# Hybrid Object Tracking with Events and Frames

Zhichao Li[1,2], Nicola A. Piga[1], Franco Di Pietro[2], Massimiliano Iacono[2], Arren Glover[2], Lorenzo Natale[1], and Chiara Bartolozzi[2]

***Abstract*— Robust object pose tracking plays an important role in robot manipulation, but it is still an open issue for quickly moving targets as motion blur can reduce pose estimation accuracy even for state-of-the-art RGB-D-based methods. An *event-camera* is a low-latency sensor that is robust to motion blur that can act complementary to RGB-D. We propose a dual Kalman filter: the first filter estimates an object's velocity capitalising on the millisecond temporal-resolution data of the event camera, the second filter fuses the tracked object velocity with a low-frequency object pose estimated from a deep neural network using RGB-D data. The full system outputs high frequency, accurate object poses also for fast moving objects. This work targets low-power robotics by replacing high-cost GPU-based optical flow used in prior work with event cameras that inherently extract the required signal without costly processing. The proposed algorithm achieves comparable or better performance when compared to two state-of-the-art 6-DoF object pose estimation algorithms and one hybrid event/RGB-D algorithm on benchmarks with simulated and real data. We discuss the benefits and trade-offs for using the event-camera and contribute algorithm, code, and datasets to the community for to further the field of visual object tracking.**

## I. INTRODUCTION

The accurate and consistent estimation of the pose of objects is a crucial capability for robots that need to act in unconstrained and dynamic environments, as it plays a key role in navigation, localisation and object manipulation. While probabilistic methods [1], [2] form a strong baseline for object pose estimation, the current-best performance algorithms employ deep-learning[3], [4], [5], [6]. However, these algorithms are designed for offline estimation or for slowly moving targets, and accurate online object pose tracking for fast moving objects is still an open problem.

A major difficulty when performing pose estimation for fast-moving objects comes from traditional cameras having an exposure time. If the object moves more than a few pixels during the exposure period, the object appears blurred across the image. High-frequency cameras might need special illumination to cope with limited exposure time and data transfer of high-resolution images becomes a bottle-neck in many vision pipelines. In autonomous robots with limited power and computational budget, inference time might be too slow to process accurately fast moving targets, introducing delays that result in inaccurate pose estimation, and therefore unreliable robot behaviour. A solution is to combine pose estimation with object velocity estimation to solve the issue of slow inference and inaccuracy [7].

[1]Humanoid Sensing and Perception, Istituto Italiano di Tecnologia, Italy {first.last}@iit.it

[2]Event-driven Perception for Robotics, Istituto Italiano di Tecnologia, Italy {first.last}@iit.it

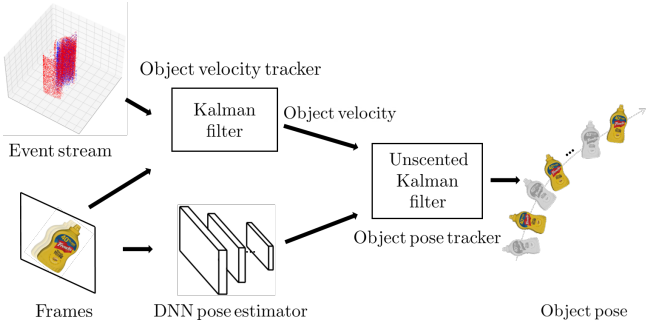



Fig. 1: Overview of the proposed dual filtering framework for fast moving object pose tracking. Velocity tracking is performed in a Kalman filter based on event-camera data. A DNN pose estimator provides low frequency object poses. An Unscented Kalman filter fuses estimated object poses with the object velocity and outputs high frequency, accurate object poses.

Event cameras don't have a fixed exposure time. Each pixel has an independent circuit that monitors illumination change and outputs data asynchronously with precise (sub-millisecond) timing. An object moving across the field of view produces *events* for each pixel that it passes over which can be disentangled in time and space. As such, event-cameras are useful sensors for solving motion blur problems. In addition, the intrinsically compressed data signal can be used to realise low-latency and low-cost visual processing [8]. However, event-cameras measure illumination *change* and not the absolute pixel brightness, and therefore are not as suited to absolute pose estimation when compared to RGB-D.

In this work we explore augmenting an RGB-D 6-DoF pose tracking algorithm with data from an event-camera to improve accuracy for fast-moving objects. In particular we take an approach similar to our previous work, ROFT [7], which demonstrated state-of-the-art performance. ROFT employs a dual Kalman filter approach: an inner loop estimates 6-DoF velocity from observations of optical flow, while an outer loop estimates the 6-DoF pose, fusing the velocity estimate with direct pose estimates from a Deep Neural Network (DOPE [3]). The optical flow used for velocity estimation is calculated from consecutive measures of dense pixel intensity images. Such a system relies on the use of power-hungry GPU hardware and is *still* subjected to poor performance if the camera imagery contains motion blur.

The event camera is a strong contender to replace optical flow computation for observations of velocity. The sparse

signal promises lightweight processing that inherently contains the illumination change signal from which velocity can be estimated, and the high-temporal resolution event-based data stream goes a long way to eliminate motion blur problems. However an observation model is still required to convert discrete spatio-temporal events into a 6-DoF velocity error signal. We therefore adopt a generative event model that has shown promising performance in 6-DoF camera pose estimation [9]. The generative event model produces a Kalman filter innovation based on the expectation of event firing-rate given the object's edge contrast and current velocity estimate.

The full 6-DoF object pose estimation system includes the generative event model wrapped in a Kalman filter for 6-DoF object velocity estimation, fused with any off-the-shelf RGB-D object pose estimator (we use both DOPE [3] and DeepTrack [6] in our experiments without further pipeline modification) wrapped in an Unscented Kalman filter [10], as illustrated in Fig. 1.

The main contributions can be summarised as:

- The generative-event observation model from [9] adapted from 6-DoF camera pose estimation to 6-DoF object pose estimation.
- Integration of event-camera into a hybrid RGB-D pipeline, replacing the need for optical flow.
- Comparison to two state-of-the-art RGB-D algorithms ([7], [11]) on synthetic and real datasets comprising objects from YCB Model Set [12] - made available to the community.
- Quantitative comparison of data-driven ([13]) v.s. model-based approach for velocity estimation with event-cameras.
- Discussion on the advantages and disadvantages on the event-based integration and generative model with recommendations for future directions.

## II. RELATED WORK

Visual object pose tracking has been approached using traditional Bayesian filtering techniques, in which the prior probability distribution and the observation probability distribution are fused assuming underlying Gaussian distributions, as in Kalman filtering [1], or other more general distributions, as in Particle filtering [14]. These techniques can be optimal for all systems that are modelled precisely, but can fail due to non-modelled effects such as dynamic lighting, reflections, camera blur, deformable objects or occlusions.

Data-driven approaches, i.e. deep neural networks (DNN), have been employed for visual object pose tracking and are able to account for many difficult-to-model system characteristics, so long as a variation of these characteristics occur in the datasets used to train the models [11], [15]. Despite the impressive performance of recent DNNs for visual object pose tracking, they are fundamentally limited if the sensor data is not appropriate for the tracking task.

Event-cameras [16] are vision sensors that produce a signal in a completely novel way when compared with traditional cameras. Instead of sampling the entire pixel-array at a fixed polling rate, each pixel in an event-camera has an independent circuitry that detects *change* in light levels. Doing so has the advantage of lower latency, higher sample-rate (i.e. asynchronous), and a compressed signal. However, the visual signal is also different as image-frames are not produced and the signal is only present during object motion. Event-cameras have the potential to unlock visual tracking for tasks (e.g. fast-motion) that even state-of-the-art trackers cannot achieve due to their reliance on image-frames.

A strong body of work for object tracking with event cameras on the image plane exists [17], [18], [19], especially for static cameras, in which only the object produces any signal and simple clustering of pixels is a valid approach. Object pose estimation and tracking methods have relied on hybrid approaches which combine frames and events [13], [20]. In [13] events are processed in a DNN to give a pose differential signal, and frames (and also depth) are processed in another DNN to refine the final pose output. In [20] a event-based trajectory optimization and a frame-based alignment module were proposed for 6-DoF object pose tracking. Hybrid event-camera and frame-based camera approaches have proven to be successful also in other tasks, such as high-rate feature tracking [21], and simultaneous localisation and mapping [22]. Complex motion techniques using *only* events have been limited to camera pose estimation [9] and PTAM [23] (with mixed results), in which the full sensor array (not only signal from a single object) increase the observability of the problem and help to achieve convergence. Due to the complementary nature of camera pose tracking and object pose tracking, such methods are still influential.

In our work we propose a hybrid method on two fronts. Firstly, we introduce a novel hybrid frame-based/event-based method in which we take advantage of fast incremental updates from events, and slower but globally accurate predictions from a state-of-the-art frame-based pose estimator [3]. Secondly, our technique combines a data driven DNN for pose estimation, enabling robustness to difficult to model visual problems, with a visual-geometry-based systems modelling approach for velocity estimation from events.

## III. METHODOLOGY

Given an input stream of RGB $\{I_k\}$ and depth $\{D_k\}$ images, a stream of events $\{E_j\}$ and an initial estimate of the object pose, the proposed pipeline tracks the 6D pose and velocity of the object with respect to the camera frame, for every frame $k$. We use different indexes for the RGB-D frames and the events to indicate that they have different temporal resolutions. We also assume that a stream of (non-perfect) pose measurements $\{T_k \in \mathrm{SE}(3)\}$ is available. The poses $T_k$ can be obtained using, for example, a deep learning-based network for pose estimation. We are interested in the scenario in which the output frequency of the network is lower than that of the RGB-D source, i.e., the poses $T_k$ are available only for some $k$, which is typical for heavy processing required of pose detection.

The proposed pipeline has two Kalman filter stages:

1) (III-A) 6-DoF velocity, $\mathcal{V}_k$, estimation with an error signal generated following [9], which requires events,

$\{E_j\}$, as well as RGB, $\{I_k\}$, and D, $\{D_k\}$, information in order to estimate edge gradient magnitudes.
2) (III-B) 6-DoF pose, $x$, estimation from the fusion of velocity, $\mathcal{V}_k$, and pose observations, $\{T_k\}$.

### A. Object velocity estimation

From the stream of RGB $\{I_k\}$ and depth images $\{D_k\}$, and a stream of events $\{E_j\}$, we track the 6-DoF object velocity $\mathcal{V}$ using a Kalman Filter. We assume that each event is represented as

$$E_j = (p_j, \mathrm{u}_j, t_j), \tag{1}$$

where $p_j \in [-1, 1]$ is the event polarity, $\mathrm{u}_j \in \mathbb{R}^2$ is the Cartesian coordinate of the pixel that has emitted the event in the camera plane and $t_j \in \mathbb{R}$ is the timestamp associated to the event. The filtering process is defined as:

*1) State definition:* We draw from the Screw Theory [24] and we define the state as the *spatial* velocity $\mathcal{V}$

$$\mathcal{V} = \begin{bmatrix} v_O & \omega \end{bmatrix}, \tag{2}$$

where $\omega \in \mathbb{R}^3$ is the angular velocity of the object and $v_O \in \mathbb{R}^3$ is the velocity of a point which coincides with the origin of the camera instantaneously and moves as if it was rigidly attached to the object.

*2) Motion model:* We assume that the underlying dynamics of the state vector $\mathcal{V}$ is described by the following motion model:

$$\begin{aligned} \mathcal{V}_{k+1} &= \alpha \mathcal{V}_k + w, \\ w &\sim \mathcal{N}\left(0, \mathrm{diag}(Q_v, Q_\omega)\right) \end{aligned} \tag{3}$$

where the $\alpha$-scaled velocity increments $\mathcal{V}_{k+1} - \alpha\mathcal{V}_k$ are Gaussian with covariance $Q_v \in \mathbb{R}^{3\times3}$ and $Q_\omega \in \mathbb{R}^{3\times3}$ associated to the linear and angular velocity, respectively. $\alpha < 1 \in \mathbb{R}$ counteracts the particular phenomenon that events have no observation of 0 velocity as there is simply no data. The dampened motion model is a functional method, but a more appropriate observation model may result in a better solution.

*3) Measurement model:* We build upon prior work [9], and assume that an event $E_j$ is generated at pixel $\mathrm{u}_j$ if the change in brightness $|\Delta\log(I)|$ exceeds a threshold $C$, for that pixel. Here, we use $I$ as the generic intensity image as seen by the event camera. According to [9], the change in brightness can be approximated as

$$|\Delta\log(I)| \approx \nabla_{u_j}\log(I)^T \dot{\mathrm{u}}_j \Delta_{t,j}, \tag{4}$$

where $\nabla_{u_j}\log(I) \in \mathbb{R}^2$ is the image gradient evaluated in $\mathrm{u}_j$, $\dot{\mathrm{u}}_j \in \mathbb{R}^2$ is the pixel velocity, and $\Delta_{t,j}$ is the time elapsed from the last firing event at the same pixel location. Taking into account the event polarity and the aforementioned threshold, the event $E_j$ is fully specified by the following [9]:

$$-p_j \nabla_{u_j}\log(I)^T \dot{\mathrm{u}}_j \Delta_{t,j} - C \approx 0. \tag{5}$$

In practice, we substitute the intensity image $I$ with the *nearest* in time RGB image $I_k$ from the input stream. Moreover, we predict $\dot{\mathrm{u}}_j$ as a function of the estimated spatial velocity $\mathcal{V}_k$, as done in [7]:

$$\dot{\mathrm{u}} = J_u(D_k)\mathcal{V}_k, \tag{6}$$

with $J_u \in \mathbb{R}^{2\times6}$ a suitable matrix [7] depending on the RGB-D camera instrincs parameters, the pixel coordinates and the depth image $D_k$ at location $\mathrm{u}_j$.

In a Kalman filtering context, we then define the event measurement model for the $j$-th event as follows:

$$y^E_{u_j}(\mathcal{V}_k) = \left(-p_j \nabla_{u_j}\log(I_k)^T J_{u_j}(D_k)\Delta_{t,j}\right)\mathcal{V}_k - C. \tag{7}$$

Finally, we collect all events within a given temporal window in the overall measurement model

$$\begin{aligned} y^E_k(\mathcal{V}_k) &= \begin{bmatrix} \cdots \\ y^E_{u_j}(V_k) \\ \cdots \end{bmatrix} + \nu \quad u_j \in \Omega_k, \\ \nu &\sim \mathcal{N}(0, R_E), \end{aligned} \tag{8}$$

where we assume to know the set $\Omega_k$ of pixels belonging to the surface of the object that caused an event within the temporal window. The cardinality of the set is $|\Omega_k| = L$. Moreover, $\nu$ represents additive Gaussian noise with measurement noise covariance $R_E \in \mathbb{R}^{L\times L}$.

Given the definition in Eq. (5), which tells that the LHS of the equation should vanish, we match the measurement model in Eq. (8) with a set of zero-valued pseudo-measurements [25]:

$$y^E_k = 0 \in \mathbb{R}^L. \tag{9}$$

*4) Kalman filtering:* We combine the models in Eqs. (3) and (8) within a Kalman Filter [26] in order to track the object velocity $\mathcal{V}_k$. At each step $k$, the previous state is propagated through the motion model in Eq. (3) to obtain the *predicted* state $\mathcal{V}^-_k$. Then, a correction step incorporates the measurements $y^E_k$ in the state belief $\mathcal{V}_k$ according to the measurement model in Eq. (8).

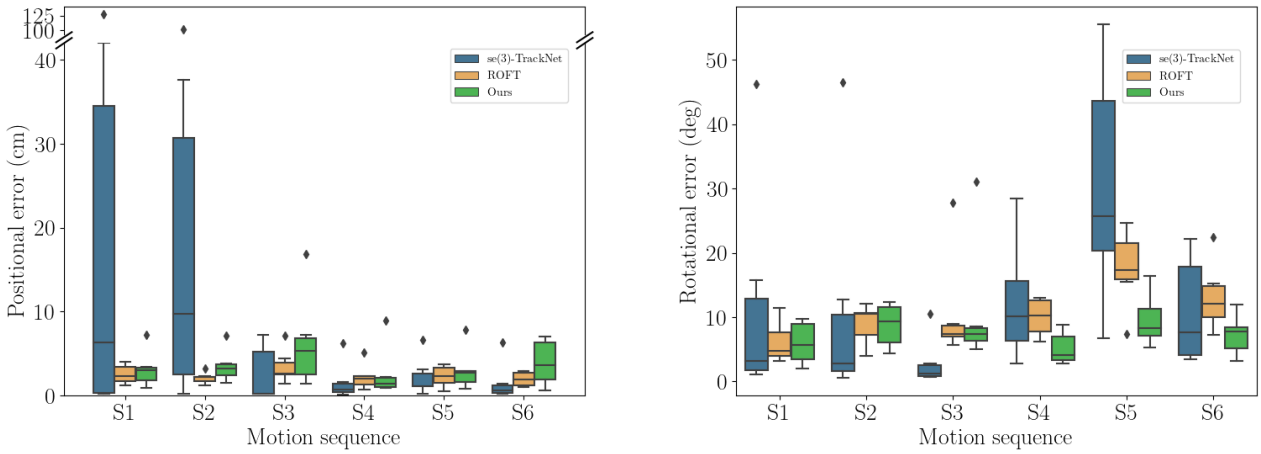
The number of events $L$ can be vary large, in the order of thousands. In order to avoid processing high dimensional measurements, which might be intractable [27], we process the events sequentially as in [28]. The actual state prediction and update equations are not reported for brevity and can be found in [26], [28].

*5) Mismatch between events and RGB-D frames:* As previously described, given an event at location $\mathrm{u}_j$, we use the nearest in time RGB-D image $I_k$ and $D_k$ in order to evaluate the image gradient and obtain depth.

In order to account for the temporal distance between the RGB-D frame and the actual firing time of the event $t_j$, we shift the pixel location $\mathrm{u}_j$, that we use to access to the RGB-D frame, as follows:

$$\mathrm{u}^s_j = \mathrm{u}_j + J_{u_j}(D_k)\mathcal{V}^-_k(t(I_k) - t_j), \tag{10}$$

where $t(I_k)$ is the timestamp associated to the RGB-D frame $I_k$ and $\mathcal{V}^-_k$ is the predicted velocity of the object (see Sec. III-A.4). As a result, we evaluate the image gradient as $\nabla_{u^s_j}(I_k)$ and depth as $D_k(\mathrm{u}^s_j)$.




(a) Position error

(b) Rotation error

Fig. 2: Tracking errors for se(3)-TrackNet, ROFT and our algorithm showing RMS error of position (a) and RMS error of rotation (b). Translation object motion along $x$, $y$, and $z$ axes correspond to S1, S2 and S3. Rotational object motion along the $z$, $y$, and $x$ axes corresponds to S4, S5, and S6.

### B. Object pose tracking

The second stage of the pipeline fuses low frame-rate pose measurements $\{T_k\}$ with the estimated velocities $\{\mathcal{V}_k\}$, available for all frames, using an Unscented Kalman Filter in order to track the 6D pose of the object. In the following, we detail the filtering process.

*1) State definition:* We define the state $x$ to be tracked over time as

$$x = [t, q], \tag{11}$$

where $t \in \mathbb{R}^3$ is the Cartesian position and $q \in \mathbb{H}$ is a unitary quaternion representing the 3D orientation.

*2) Motion model:* We assume that the state $x$ evolves according to the following constant velocity model with $v^O$ and $\omega$ as inputs:

$$\begin{aligned} x_{k+1} = \begin{bmatrix} t_{k+1} \\ q_{k+1} \end{bmatrix} = \begin{bmatrix} (I_3 + \hat{\omega}_k \Delta_T)\, t_k + v_k^O \Delta_T \\ F(\omega_k) q_k \end{bmatrix} \oplus w, \\ w \sim \mathcal{N}(0, \mathrm{diag}(Q_t, Q_q)). \end{aligned} \tag{12}$$

Here, $F(\omega)$ is the standard quaternion kinematics transition matrix [29], $Q_t \in \mathbb{R}^{3\times3}$, $Q_q \in \mathbb{R}^{3\times3}$ are the noise covariance matrix for the translational and rotational components of the state, respectively, and $\Delta_T$ is the sampling time. Furthermore, $\oplus$ represents the operation of adding noise taking into account the quaternion arithmetic [29] and $\hat{\omega}_k$ represents the skew-symmetric matrix [30] obtained from $\omega_k$. The term $\hat{\omega}_k \Delta_T t_k$ accounts for the fact that $v_k^O$ is the velocity of the object measured at the origin of the camera.

*3) Measurement model:* We collect pose measurements in a vector

$$y_k(T_k) = \begin{bmatrix} p(T_k) \\ q(T_k) \end{bmatrix}, \tag{13}$$

where $p(T_k) \in \mathbb{R}^3$, $q(T_k) \in \mathbb{H}$ are the Cartesian position and the quaternion components of the measured pose $T_k$. The measurement model relating $y_k$ with the state vector $x_k$ is expressed as

$$\begin{aligned} y_k(x_k) = \begin{bmatrix} t_k \\ q_k \end{bmatrix} \oplus \nu, \\ \nu \sim \mathcal{N}(0, \mathrm{diag}(R_t, R_q)), \end{aligned} \tag{14}$$

with $R_t \in \mathbb{R}^{3\times3}$ and $R_q$ the noise covariance matrices associated to the translational and rotational components of the state. If pose measurements are not available at time $k$, the equation simplifies to an empty measurement, i.e. the update step of the filter is skipped.

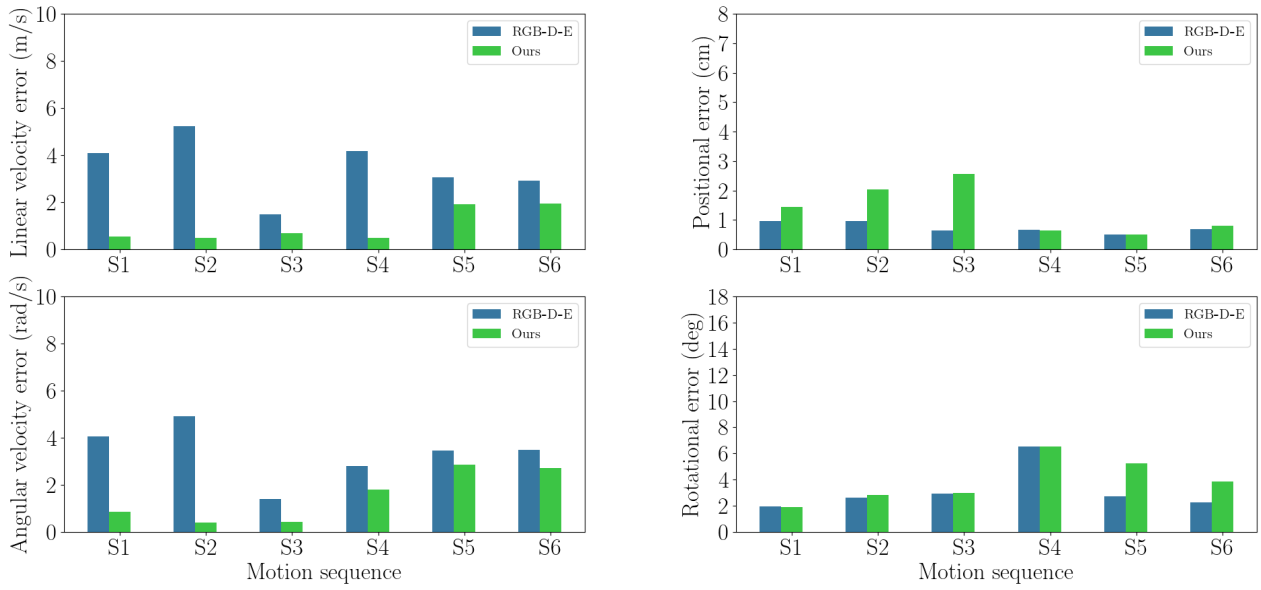
*4) Unscented Kalman filtering:* Actual tracking of the state $x_k$ is performed using a quaternion-based Unscented Kalman Filter [29] which is suitable for handling quaternions in both the state and the measurements.

The UKF state prediction and update equations are not reported for brevity and can be found in [10], [29].

## IV. EXPERIMENTS AND RESULTS

Four experiments were performed to evaluate the proposed algorithm:

- 6-DoF pose accuracy against state-of-the-art RGB-D algorithms [13], [11] on simulated datasets comprsing "003 cracker box", "004 sugar box", "005 tomato soup can", "006 mustard bottle" and "010 potted meat can" from the YCB [12] Model Set.
- 6-DoF pose accuracy and 6-DoF velocity accuracy against RGB-D-E [13] hybrid algorithm (that is not trained, and cannot operate on the above YCB objects used in the above dataset).
- An ablation study evaluating the contribution of the velocity compared to pose to the entire pipeline performance.




(a) Velocity estimation error

(b) Pose tracking error

Fig. 3: A comparison of ours and RGB-D-E for both velocity and pose tracking. The definition of S1-S6 are the same as in Fig. 2.

- Qualitative comparison on real data from an ATIS [16] calibrated against a RealSense.

Simulated datasets were generated by rendering object meshes against a background using Unreal Engine. RGB and depth was sampled at 500 Hz, and an "artificial event-camera" used the logarithmic-image-differences to produce events. Pose observations, $T_k$, are calculated at 10 Hz, and velocity estimates are computed at 60 Hz to match a typical RGB-D camera frequency.

### A. *Comparison with RGB-D object pose trackers*

We investigated the performance of event-frame fusion on 36 synthetic datasets of 6 objects from the YCB model set [12], namely "003 cracker box", "004 sugar box", "005 tomato soup can", "006 mustard bottle" and "010 potted meat can"[1]. For each object 6 datasets were created corresponding to 1 m/s translation-along the $x$, $y$, and $z$ axes (corresponding to labels S1, S2, and S3) as well as 4 rad/s rotation-around the same axes (corresponding to labels S3, S4, and S5).

The proposed algorithm was compared to ROFT [7], the main difference being the estimate of velocity coming from frame-based optical flow instead of events. In addition, the algorithm was compared to the state-of-the-art DNN-based se(3)-TrackNet [11].

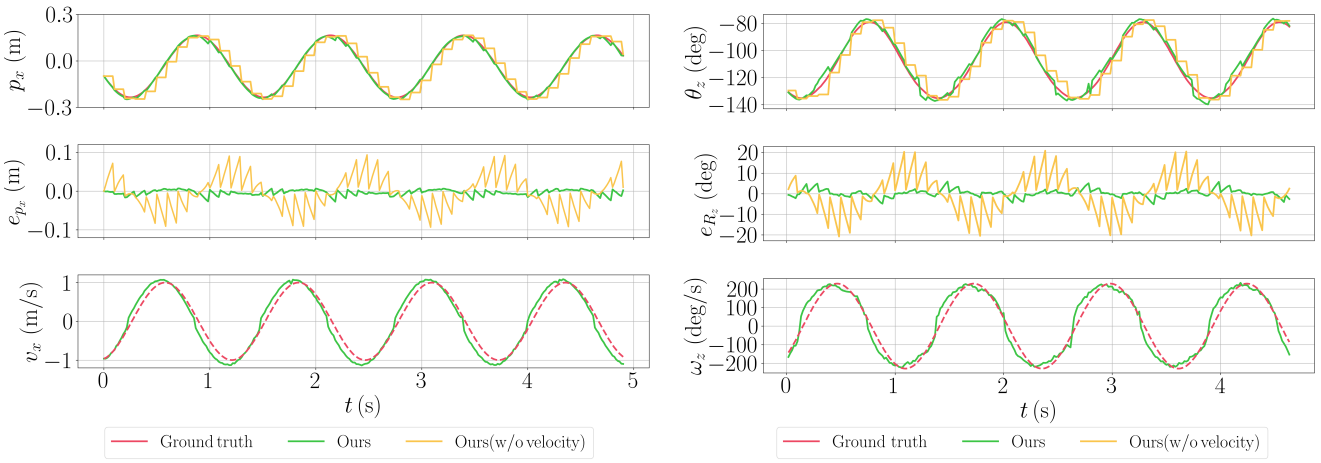
The datasets have fast object motion, but due to the synthetic production, **do not have motion blur**. Therefore, se(3)-TrackNet may struggle as it is not designed for large jumps in object position between frames. If the relative pose change of the object is larger than presented at training time the data-driven method can fail. Instead ROFT is designed for fast motions, and should not fail due to real-world limitations of RGB-D cameras (i.e. motion blur).

Our method achieved comparable positional tracking performance to ROFT (sometime with higher, sometimes lower performance), and obtained more stable tracking results compared to se(3)-TrackNet, which fails for some objects in translations, as shown in Fig. 2. The proposed algorithm shows promising performance for rotational estimation compared with ROFT indicating that the events may be more informative for rotation estimation. Indeed, the optical flow used by ROFT assumes linear visual motion between consecutive frames, while events can continuously measure the non-linear visual change associated with object rotations.

The translations along $x$ and $y$ axes (S1 and S2) most likely exceed the velocities of data on which se(3)-TrackNet was trained, instead velocities along $z$ (S3) were probably included in training data; as indicated by the tracking failures and successes in Fig. 2.

In general the result in Fig. 2 indicates that events are an appropriate visual signal to use for 6-DoF object velocity estimation and are comparable to the the optical flow commonly used in traditional computer vision. Further evaluation is required with real camera data to understand the magnitude of degradation of ROFT due to motion blur (not present in synthetic datasets) - as we did not test ROFT on real data in this work.

[1]chosen as a DOPE model trained on these models is publicly available.




(a) Translation trajectories

(b) Rotation trajectories

Fig. 4: Ablation study showing the benefit of events on the pose estimation smoothness.(a): sample trajectories from a sequence involving translation along the $x$-axis of the object "010 potted meat can". (b): sample trajectories from a sequence involving rotation around the $z$ axis of the object "006 mustard bottle".

### *B. Comparison to hybrid RGB-D and Events pose trackers*

The sole state-of-the-art in event-based 6-DoF pose estimation is the hybrid RGB-D-E algorithm [13]. Similar to ours, it is also a two step method, however in RGB-D-E both steps are data driven convolutional neural networks. The first step in RGB-D-E is an estimate of pose velocity from events in a small time window. Differently for ours, the second step is a *refinement* that combines the updated pose and the RGB-D information to give a final pose estimate. The refinement network requires a minimum error between the event updated pose and the RGB-D observation to avoid failure.

RGB-D-E is a data-driven approach and the publicly available model weights are trained only on a single *dragon* object and therefore cannot operate on the YCB objects used in the datasets discussed in Section IV-A. Instead, to enable a valid comparison we generated a further set of sequences with the *dragon* object mesh using an identical approach as outlined above. However, as DOPE was not trained on the *dragon* we instead used DeepTrack [6], which is trained on the *dragon*, as the source of $T_k$. To have a fair comparison with our pipeline, we set the output rate of RGB-D-E to 60 Hz.

As the algorithms have a similar two-step approach we compare both velocity estimation, see Fig.4a, and pose estimation, see Fig. 4b. The proposed algorithm (model-based approach) achieves a lower velocity estimation error compared to RGB-D-E (data-driven CNN). Additional data could be used to train the velocity estimation network to improve performance, however as there is a known mathematical model that relates visual flow and 6-DoF velocity (i.e. the image Jacobian $J_u$ in equation (6) in Section III), perhaps a data-driven approach at this level is less beneficial to the full pipeline. In addition, the data-driven CNN has to be trained for each object specifically, while the model-based approach is object-agnostic.

TABLE I: Average RMSE of all motion sequences by different selected disable components

| Methods | $e_p(cm)$ | $e_r(deg)$ |
|---|---|---|
| Ours | **3.76** | **7.95** |
| Ours (w/o DOPE) | 8.00 | 21.47 |
| Ours (w/o velocity) | 4.63 | 9.86 |

However, the final pose from RGB-D-E has a higher accuracy than our proposed algorithm, see Fig. 4b. The refinement network of RGB-D-E was run at 60 Hz compared to the 10 Hz of DeepTrack used by our algorithm, which gives RGB-D-E an advantage at this second stage of pose estimation. In addition, the RGB-D-E refinement network assumes that the prior pose and RGB-D observation have only a small error between them, while our tracking pipeline we can handle observations globally (i.e. across the image plane) as we rely on a deep neural network for 6D object pose estimation as the source of our measurements. Therefore our proposed algorithm can use the object detector also for pose initialisation and failure recovery.

The combined result considering Fig.4a and Fig. 4b indicates that the strongest component for RGB-D-E is not in the extraction of velocities from events, but in the refinement network.

### *C. Ablation studies*

To validate the effectiveness of different components in our algorithm, they are selectively disabled and the algorithm is tested against all motion sequences of the datasets comprising objects from the YCB Model Set. The average RMSE of position and rotation tracking are reported in Table I and, in

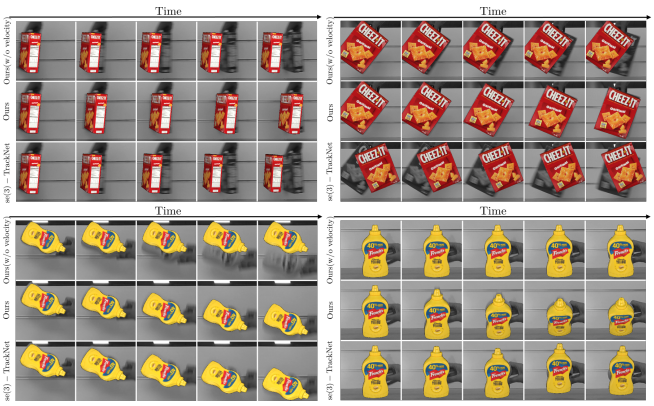



Fig. 5: Real Datasets taken with an ATIS and RealSense stereo pair. Top row: qualitative results for the object "003 cracker box" during translation along the $x$ axis (on the left) and rotation about the $z$ axis (on the right). Bottom row: qualitative results for the object "006 mustard bottle" during translation along the $y$ axis (on the left) and rotation about the $x$ axis (on the right).

combination with Fig. 4, indicate the benefit of both proposed components of the pipeline.

Without the outer loop fusing velocities with pose observations, the integrated pose will drift as expected. Instead without velocity estimation from the events the error in pose is clearly shown in Fig. 4 and further evidence for the benefit of events in the pipeline is shown in Section IV-D. The full pipeline produces smooth trajectories without drift.

### *D. Validation on the real dataset*

Real datasets were used to validate the approach considering all additional sources of noise from real sensors. The ATIS Gen3 (640 x 480) and a RealSense D415 RGB-D camera were calibrated as a stereo pair and the depth information was used to re-project events onto the RealSense camera image plane. Fig. 5 shows the "003 cracker box" object in a $x$-axis translation motion and a rotation about the $z$-axis, as well as the "006 mustard bottle" object in a $y$-axis translation motion and rotation about the $x$-axis. These samples are taken from real-world sequences and can also be found in the video provided as supplementary material. The objects are moved as fast as possible by a human operator within the visual field-of-view. The pose estimated by the algorithm is used to render the object mesh onto the image, which can be qualitatively compared to the RGB image taken from the RealSense to evaluate the algorithm performance.

The advantage of using events can be seen by comparing "Ours (w/o velocity)" with "Ours". While our algorithm shows a majority overlap between the projected pose and the RGB image for all frames over time, the pose-only method does not update the pose between frames (10 Hz), and therefore the pose lags behind the true position between detections.

se(3)-TrackNet has a result similar to that of "Ours (w/o velocity)" for the cracker box, however the cause is different. The fast-motion induces motion-blur in the images, which causes tracking failure as the relative 6D pose is not estimated correctly. Even though the se(3)-TrackNet has a high frequency, it is the traditional RGB-D sensor that causes the failure. The mustard bottle is tracked correctly for the $y$-axis translation, but for the rotation about the $x$-axis the motion-blur of the frame may have caused the inherent features that se(3)-TrackNet uses to be distorted and tracking is unsuccessful.

The qualitative evaluation on real data demonstrates the sensor is as important for the tracking as the algorithm, and that using an event-camera is a promising approach as they do not suffer from motion blur as do traditional cameras. The real datasets also prove that the algorithm is robust to the noise and visual artefacts that come along with real data.

## V. LIMITATIONS

Results indicate that events are an appropriate replacement for optical flow in a dual Kalman filter approach, as shown by comparable performance between ROFT and our proposed algorithm. The event-camera improves robustness to high-speed motion and motion-blur problems but requires additional hardware in the system. While multi-camera setups are becoming the norm for commercial devices (e.g. RealSense or smart phones), simple hardware set-ups avoid additional problems with calibration and additional points of failure. However, due to the relatively young field of computer vision with event-cameras, further improvements could be made to object pose detection with event data, and in doing so could lead to a event-camera only 6-DoF solution.

Despite achieving strong results for velocity estimation with event-cameras, the observation model [9] for events is still unsatisfactory for several reasons and is the first point of improvement for the algorithm. While technically sound, the model requires multiple events to fire at the same pixel location and discards any information for pixels which only fire a single time. In practice and anecdotally, we have found that the timing of the first pixel is the most accurate, while the timing of successive events contain the most noise and uncertainty. In disregarding the information of the first event firing, much useful information is left unused by the velocity estimation algorithm.

In addition the velocity error signal used for correction in the Kalman filter is calculated as a temporal error, and is only one dimensional, while the visual sensor is two dimensional in space. Increasing the dimensionality of the velocity error signal increases the observability of the system and leads to more accurate results. Due to this, the proposed algorithm requires some tuning of the Kalman filter measurement uncertainty for the velocity estimation to ensure accurate convergence. Therefore, while we show all motions are capable of being tracked, some *a-priori* knowledge on the types of motion help for stronger convergence. For a more robust algorithm this issue would need to be addressed, and we suggest a two dimensional error vector would be the first option to investigate.

The measurement model implemented requires both image gradients and depth to operate. Therefore the velocity obser-

vations are limited by the frame-rate of these other sensors, which also limits the frequency of operation. It is possible to keep a similar architecture, but swap out the measurement model to measure velocity with a different method, e.g. using both the spatio-temporal information (and not only temporal) from the event data.

Finally, in order to operate the algorithm in real-time further optimisation, and possibly trade-off between accuracy and computational complexity is required.

## VI. CONCLUSIONS

In this work, we propose a hybrid method that uses events for object velocity estimation and RGB-D for object pose estimation, with information fused in a dual Kalman filter implementation for tracking of fast moving objects in 6-DoF. We show that the proposed algorithm is comparable to a optical flow method for velocity estimation, but should be more light-weight and more robust to motion-blur - a necessity for highly dynamic robots. In addition the proposed algorithm has higher velocity estimation accuracy, which does not require training on each individual object, compared to the state-of-the-art that uses events for 6-DoF object velocity estimation. We have also validated that the proposed algorithm successfully tracks fast moving objects in a real-world hybrid event-RGB-D stereo pair setup.

We have discussed the major limitations of the system to provide direction for the community to further improve on event-based 6-DoF tracking. The datasets and code are also provided open-source.

## REFERENCES


[1] J. Issac, M. Wüthrich, C. G. Cifuentes, J. Bohg, S. Trimpe, and S. Schaal, "Depth-Based Object Tracking Using a Robust Gaussian Filter," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 608–615.

[2] M. Wüthrich, P. Pastor, M. Kalakrishnan, J. Bohg, and S. Schaal, "Probabilistic Object Tracking using a Range Camera," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 3195–3202.

[3] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects," in *Conference on Robot Learning (CoRL)*, 2018. [Online]. Available: https://arxiv.org/abs/1809.10790

[4] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[5] W. Chen, X. Jia, H. J. Chang, J. Duan, L. Shen, and A. Leonardis, "FS-Net: Fast Shape-based Network for Category-Level 6D Object Pose Estimation with Decoupled Rotation Mechanism," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 1581–1590.

[6] M. Garon, D. Laurendeau, and J.-F. Lalonde, "A Framework for Evaluating 6-DOF Object Trackers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 582–597.

[7] N. A. Piga, Y. Onyshchuk, G. Pasquale, U. Pattacini, and L. Natale, "ROFT: Real-Time Optical Flow-Aided 6D Object Pose and Velocity Tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 159–166, 2022.

[8] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-Based Vision: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022.

[9] G. Gallego, C. Forster, E. Mueggler, and D. Scaramuzza, "Event-based Camera Pose Tracking using a Generative Event Model," *arXiv preprint arXiv:1510.01972*, 2015.

[10] E. A. Wan and R. V. D. Merwe, "The Unscented Kalman Filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[11] B. Wen, C. Mitash, B. Ren, and K. E. Bekris, "se(3)-TrackNet: Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 367–10 373.

[12] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, "The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research," in *2015 International Conference on Advanced Robotics (ICAR)*, 2015, pp. 510–517.

[13] E. Dubeau, M. Garon, B. Debaque, R. d. Charette, and J.-F. Lalonde, "RGB-D-E: Event Camera Calibration for Fast 6-DOF Object Tracking," in *2020 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2020, pp. 127–135.

[14] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "PoseRBPF: A Rao–Blackwellized Particle Filter for 6-D Object Pose Tracking," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1328–1342, 2021.

[15] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented Autoencoders: Implicit 3D Orientation Learning for 6D Object Detection," *International Journal of Computer Vision*, vol. 128, no. 3, pp. 714–729, 2020.

[16] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," in *2008 IEEE International Symposium on Circuits and Systems*, may 2008, pp. 2130–2133.

[17] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based Feature Tracking with Probabilistic Data Association," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4465–4470.

[18] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 8, pp. 1710–1720, 2014.

[19] A. Glover and C. Bartolozzi, "Robust Visual Tracking with a Freely-moving Event Camera," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3769–3776.

[20] H. Li and J. Stueckler, "Tracking 6-DoF Object Motion from Events and Frames," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 14 171–14 177.

[21] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Asynchronous, Photometric Feature Tracking using Events and Frames," in *Computer Vision – ECCV 2018*. Springer International Publishing, 2018, pp. 766–781.

[22] 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 Robotics and Automation Letters*, vol. 3, no. 2, pp. 994–1001, 2018.

[23] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza, "EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 593–600, 2017.

[24] R. Featherstone, *Rigid Body Dynamics Algorithms*. Springer, 2014, ch. 2, pp. 10–13.

[25] P. Richards, "Constrained Kalman Filtering Using Pseudo-measurements," in *IEE Colloquium on Algorithms for Target Tracking*, 1995, pp. 75–79.

[26] Y. Bar-Shalom, T. Kirubarajan, and X.-R. Li, *Estimation with Applications To Tracking and Navigation: Theory, Algorithms and Software*. USA: John Wiley & Sons, Inc., 2002.

[27] C. McManus and T. Barfoot, "A Serial Approach to Handling High-Dimensional Measurements in the Sigma-Point Kalman Filter," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011. [Online]. Available: http://www.roboticsproceedings.org/rss07/p29.html

[28] D. Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. John Wiley & Sons, 2006.

[29] A. C. Chiella, B. O. Teixeira, and G. A. S. Pereira, "Quaternion-based robust attitude estimation using an adaptive Unscented Kalman Filter," *Sensors*, vol. 19, no. 10, p. 2372, 2019.

[30] S. Andrilli and D. Hecker, "Vectors and Matrices," in *Elementary Linear Algebra*, Fifth Edition ed., S. Andrilli and D. Hecker, Eds. Boston: Academic Press, 2016, pp. 1–83.