FlyTracker: Motion Tracking and Obstacle Detection for Drones Using Event Cameras

Yue Wu* Jingao Xu* Danyang Li* Yadong Xie† Hao Cao* Fan Li† Zheng Yang*
* School of Software, Tsinghua University, Beijing, China.
† School of Computer Science, Beijing Institute of Technology, Beijing, China.
Email: ywu92@mail.tsinghua.edu.cn, {xujingao13, lidanyang1919}@gmail.com, ydxie@bit.edu.cn,
i.haocao@gmail.com, fli@bit.edu.cn, hmilyyz@gmail.com

IEEE INFOCOM 2023 - IEEE Conference on Computer Communications | 979-8-3503-3414-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/INFOCOM53939.2023.10228976

***Abstract*—Location awareness in environments is one of the key parts for drones' applications and have been explored through various visual sensors. However, standard cameras easily suffer from motion blur under high moving speeds and low-quality image under poor illumination, which brings challenges for drones to perform motion tracking. Recently, a kind of bio-inspired sensors called event cameras emerge, offering advantages like high temporal resolution, high dynamic range and low latency, which motivate us to explore their potential to perform motion tracking in limited scenarios. In this paper, we propose FlyTracker, aiming at developing visual sensing ability for drones of both individual and circumambient location-relevant contextual, by using a monocular event camera. In FlyTracker, background-subtraction-based method is proposed to distinguish moving objects from background and fusion-based photometric features are carefully designed to obtain motion information. Through multilevel fusion of events and images, which are heterogeneous visual data, FlyTracker can effectively and reliably track the 6-DoF pose of the drone as well as monitor relative positions of moving obstacles. We evaluate performance of FlyTracker in different environments and the results show that FlyTracker is more accurate than the state-of-the-art baselines.**

I. Introduction

With the rapid development of smart cities, commercial drones are more and more popular in quantities of smart applications like aerial photography, entertaining showing, transportation management, and even emergency rescue due to their flexible flight control system. Location awareness in various environments is one of the key parts for these applications, always called the six-degree-of-freedom (6-DoF) pose tracking in drone flight control, providing a drone with its location and orientation in 3D space to help the drone keep flight attitude and adjust moving path.

Solutions for 6-DoF pose tracking [1]–[3] have been researched for a long time. Among these algorithms and system, vision-based methods become one of the most attractive solutions because of the gradual maturity of computer vision processing technology and the increasing abundance of two-dimensional environmental information provided by visual sensors. For on-drone pose tracking systems, visual sensors cooperating with Inertial Measurement Unit (IMU) are widely adopted [4], [5]. These methods obtain poses of a drone through jointly optimizing the results of image feature matching and IMU motion estimation. Another type of widely-used pose tracking methods depend on the fusion of visual sensors and LiDAR [6], [7], also enabling a continuous estimation of 6-DoF poses for drones. However, in practice, drones usually fly with uneven velocity across complex environments to perform application tasks (especially applications related to rescue detection). Vision-based methods are easily suffer from low image quality caused from motion blur, poor illumination or HDR scenarios, where the performance of image feature matching decreases a lot, and further decrease the accuracy of pose estimation [8], [9]. Moreover, various kinds of moving objects may appear in the Field of View (FoV) of a drone. Moving objects in FoV, especially high-speed obstacles, not only affect the accuracy of pose estimation, but also bring crash risks to the drone. Therefore, existing vision-based methods of 6-DoF pose tracking cannot be directly applied for drone flight control in some challenging scenarios.

Recently, a kind of bio-inspired sensors called event cameras emerge and have attracted much attentions. Event cameras are asynchronous sensors that pose a paradigm shift in the way visual information is acquired. This is because they sample light based on the scene dynamics, rather than on a clock that has no relation to the viewed scene. Event cameras can offer several advantages such as very high temporal resolution and low latency, very high dynamic range, and low power consumption, which suggests that they have a large potential for robotics applications in challenging scenarios. Motivated by the aforementioned situation, we explore the practicability of combining events and standard image frames to perform 6-DoF pose tracking for drones. We consider to fuse events into vision-based methods based on two insights. First, event cameras measure the changes of scene brightness in contrast of standard cameras providing absolute brightness measurement. An event is triggered by a pixel when the log intensity (a representation of brightness) of the pixel exceeds a threshold. Thus, event-based features are more robust to the change of illumination than image-based features. Second, event cameras monitor brightness changes very fast through analog circuitry, and read out events in digital manner with a $1MHz$ clock. The microsecond resolution of which events are detected and timestamped make it easier to detect and track high-speed moving objects by using event-based features. Generally speaking, event cameras do well in sensing motion with very low latency and measure changes of brightness, while standard cameras provide absolute brightness measurement for each

pixel but with relatively high latency. Thus, Event cameras and standard cameras are complementary for developing novel algorithms capable of combining the specific advantages of both cameras to perform location aware tasks with high temporal resolution. Dynamic and Active-pixel Vision Sensor (DAVIS) is recently introduced by Brandli *et al.* [10] in this spirit. It is a sensor comprising an asynchronous event-based sensor and a standard image-based camera in the same pixel array, which is portable enough to be equipped on a drone.

In this paper, we propose a monocular 6-DoF pose tracking system, named FlyTracker, for drone flight control. In the design of FlyTracker, the heterogeneous visual data, i.e., standard image frames and asynchronous event streams, is tightly fusion for motion tracking in front end. Specifically, a brightness increment model is built from the accumulated events, producing an intuitive representation of the scene brightness change in a tiny time period. Meanwhile, a predicted brightness increment model is built from image brightness gradient and optical flow in that time period, describing the brightness increment caused by moving contours according the Event Generation Model (EGM). Then, pose changes is estimated by a joint optimization that minimizing the difference of the two models. To reduce the impact of moving objects on motion tracking accuracy, an online background-subtraction-based method is proposed to detect whether there is any moving obstacle appears in the FoV of the drone. Once a moving obstacle is found, a coarse-grained mask of the obstacle is quickly calculated that is further used to distinguish obstacle part and background part on the image plane. So that the ego-motion tracking can be performed only using visual data belong to the background part. The motion tracking methods based on the fusion of events and images can also be used to monitor the position of the detected obstacle relative to the drone, which can help the flight control system to guide obstacle avoidance. Another fusion of two parts of motion tracking results is conducted through comparing whether the drone motion relative to background is obviously different from that relative to the obstacle, which can in return verify the obstacle detection result that is not mislead by dynamic background like shaking leaves. On a whole, our system excels in three aspects:

- The tight fusion between event feature and image feature improves the tracking accuracy in conditions where a drone flies with relatively high velocity or flies under poor illumination or HDR circumstances.
- A moving object detection method distinguishes obstacle part and background part in real time, further reduces the impact on tracking accuracy that caused by moving objects appearing in FoV .
- A closed-loop structure is formed through a shallow fusion of results from ego-motion tracking and obstacle motion tracking, reducing false results provided by the moving object detection method.

We implement FlyTracker on a commercial drone equipped with a DAVIS and evaluate FlyTrack in various scenarios. We also compare FlyTracker with state-of-the-art vision-based motion 6-DoF tracking methods in terms of absolute trajectory error and rotation error. Evaluation results show that FlyTracker achieves the highest accuracy among the baselines, indicating FlyTracker is effective and reliable to perform 6-DoF tracking across different environments. Our contributions are summarized as follows:

- We propose FlyTracker, which develops visual sensing ability for drones of both individual and circumambient location-related contextual. By equipped with an event camera, a drone can track the 6-DoF pose of itself and monitor surrounding moving obstacles through FlyTracker.
- In FlyTracker, multilevel fusion is performed on the heterogeneous visual data, i.e., standard image frames and asynchronous event streams. Shallow data fusion of output from modules is used for detection, and tight data fusion of image features and event features is used for tracking. The multilevel data fusion effectively improves the accuracy and robustness of FlyTracker working in dynamic environments.
- In FlyTracker, we take advantages of both images and the events provided by event cameras. We propose a motion tracking method based on fusion-based photometric features according to EGM. We also propose an online background-subtraction-based method to quickly detect moving obstacles that appear in the FoV.
- Evaluations are conducted in different scenarios comparing with the state-of-the-art vision-based methods. Evaluation results show that FlyTracker makes high-accuracy flight control possible for drones in dynamic and HDR environments.

The remainder of the paper is organized as follows. We present related work in Section II. We introduce the principle of event cameras as well as event generation model in Section III. Overview and detailed system design are presented in Section IV. In Section V, we present implementation setup and performance evaluation results. Finally, we draw our conclusion in Section VI.

II. Related Work

Image-based Methods. VO methods using image frames captured by standard cameras have been explored for a long time, which can be divide into indirect measurements (i.e., feature based VO) [11]–[13] and direct measurements [14]. Indirect methods usually depend on features like SIFT [15], SURF [16] and BRIEF [17]. The recent ORB [18] combines FAST [19] and BRIEF that uses pyramid to generate multiscale representations, achieving great success in visual SLAM [12], [13]. For indirect methods, varying conditions in environments such as lighting conditions and dynamic obstacles can impede accuracy with outliers. RANSAC is employed by many works to circumvent the outliers. Buczko et al. propose an outlier detection scheme for monocular and stereoscopic [20] approaches. Direct methods are more robust with photometric calibration while being insensitive to pixel

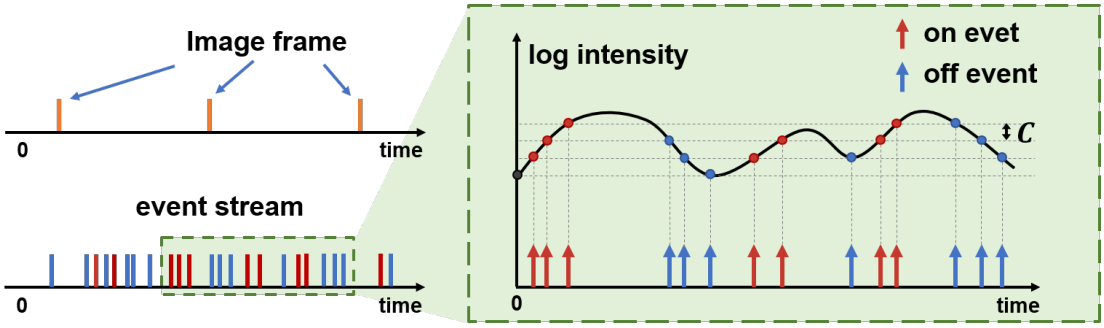
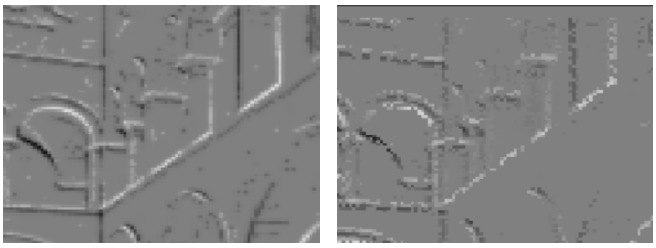


Fig. 1: Asynchronous events vers synchronous frames.

discretization artefacts. SVO [21] and DSO [14] are prominent VO algorithms that DSO uses a direct approach whereas SVO uses a semi-direct approach. DSO also uses a sparse formulation thereby decreasing computation complexity, as it samples only points of sufficient intensity gradient, and neglecting the geometric prior. By combining traits of direct and indirect methods, SVO performs a minimization of photometric errors on features of the same 3D point, where subpixel features are subsequently obtained through the relaxation of geometric constraints.

Event-based Methods. Image-based methods cannot track motion in the blind time between consecutive image frames. Some of them are high-cost since they process information from all pixels. In contrast, event cameras capture only relevant brightness information and respond asynchronously, thus, filling the blind time between consecutive image frames. Early event-based methods were simple and focused on demonstrating the low-latency and low-processing requirements of vision systems. Motions are tracked as clustered blob-like sources of events [22], circles [23] or lines [24]. Methods of tracking more complex or high-contrast scenery are demonstrated by using event-by-event adaptations of the Iterative Closest Point (ICP) algorithm [25], gradient descent [26], Mean-shift and Monte-Carlo methods [27], or particle filtering [28]. Similar to the method in [29], Tedaldi *et al.* [30] propose features consisting of local edge patterns that are represented as point sets. Method proposed in [31] considers events in overlapping spatio-temporal windows and align them using the current camera motion and scene structure, yielding motion-compensated event frames. However, the above event-based methods suffer from drift as event appearance changes over time.

Fusion of Images and Events. Event streams and intensity images are visual information with complementary sources. Recent researches has proven that combining them is useful to improve the accuracy and robustness in various visual applications, such as feature tracking [32], ego-motion estimation [33], [34], depth prediction [35], video reconstruction [36], [37] and video frame interpolation [38]. The method in [39] is part of SLAM pipeline which uses three filters operating in parallel to jointly estimate the motion of the event camera, a 3D map of the scene, and an intensity image. Its depth



(a) brightness model using events (b) predicted model using image

Fig. 2: An example of EGM output.

estimation approach requires using an additional quantity, i.e., the intensity image, to solve for data association. In contrast, EMVS [40] is a space-sweep method that leverages the sparsity of event streams to achieve 3D reconstruction without establishing event matches or recovering intensity images. It projects events into space and then finds scene structure as local maxima of ray density. The basic idea of EMVS is also used in EVO [41]. Several fusion-based methods [32], [34], [42] exploit EGM that states how events are created when a predefined contrast threshold is reached. Corresponding experiments suggest that methods applying EGM can achieve higher accuracy than methods without EGM.

III. Background of Event-based Vision

A. Principle of Event Cameras

Event cameras are asynchronous sensors that pose a paradigm shift in the way visual information is acquired. Every pixels in an event camera respond asynchronously and independently to brightness changes in the photographed scene. Thus, the output of an event camera is a variable data sequence of digital "events". Each event represents a predefined-magnitude change of brightness (i.e., log intensity) at a pixel and a specific time. From the left figure of Fig. 1, we can see that standard cameras capture full images with fixed rates (e.g., 30 fps), while event cameras output events with different time intervals in contrast. As shown in the right figure of Fig. 1, in the design of event cameras, each pixel memorizes and monitors the log intensity from the time it gives out an event. When the change of log intensity exceeds a predefined threshold $\pm C$, the camera gives out a new event

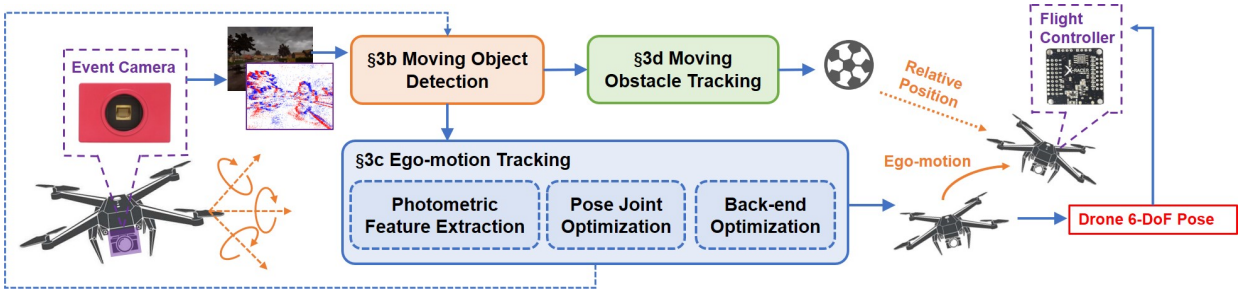


Fig. 3: System overview.

which has a polarity of "ON" or "OFF", indicating the increase or decrease of brightness, respectively.

It is obvious that event cameras are data-driven sensors which have much potential advantages over standard cameras. Since the generated events depend on motion or brightness change in the scene, the faster the motion, the more events per second are generated. In addition, events are detected with microsecond resolution and are transmitted with sub-millisecond latency. Therefore, event cameras can capture very fast motions, without suffering from motion blur which is typical to frame-based cameras. Moreover, event cameras often have much higher dynamic range (more than 120 dB) than frame-based cameras, enabling them to work effectively in very dark and very bright environments.

B. Event Generation Model

An event e_k can be represented by a quadruple (x_k, y_k, t_k, p_k) when it is triggered at pixel $u_k = (x_k, y_k)^\top$ and at time t_k with polarity $p_k \in \{-1, +1\}$. It is generated as soon as the brightness increment ΔL reaches the threshold $\pm C$ (with $C>0$), formally,

$$\Delta L(u_k, t_k) = L(u_k, t_k) - L(u_k, t_k - \Delta t_k) = p_k C, \quad (1)$$

where Δt_k is the time since the last event triggered at the same pixel. If the time interval $\Delta\tau = t_{N_e} - t_1$ is much small, the brightness increment (1) can be approximated using Taylor's expansion by $\Delta L(u_k, t_k) \approx \frac{\partial L}{\partial t}(u_k, t_k)\Delta t_k$, which interprets the events as providing information about the temporal derivative:

$$\frac{\partial L}{\partial t}(u_k, t_k) \approx \frac{p_k C}{\Delta t_k}. \quad (2)$$

This is an indirect way of measuring brightness, since events are triggered by brightness change, rather than the brightness derivative exceeding a threshold. However, The above interpretation inspires us to design physically-grounded event-based algorithms.

Accumulating the polarities of events $\{e_k\}_{k=1}^{N_e}$ with a quantity of N_e over a time interval $\Delta\tau$ produces a brightness increment image $\Delta L(u)$:

$$\Delta L(u) = \sum_{t_k \in \Delta\tau} p_k C \delta(u - u_k), \quad (3)$$

where $\delta(\cdot)$ is the Kronecker delta function used to select appropriate pixels. For small $\Delta\tau$ and given the brightness constancy assumption that usually used in optical flow, it is intuitive that $\Delta L(u)$ is caused by moving edges. In other words, the brightness increments are caused by brightness gradients $\nabla L(u) = (\frac{\partial L}{\partial x}, \frac{\partial L}{\partial x})^\top$ moving with velocity $V(u)$ over a displacement $\Delta U = V(u)\Delta\tau$ on the image plane. Formally,

$$\Delta L(u) \approx -\nabla L(u) \cdot \Delta U = -\nabla L(u) \cdot V(u)\Delta\tau. \quad (4)$$

We next denote $-\nabla L(u) \cdot V(u)\Delta\tau$ as $\hat{L}(u)$ in the subsequent content to represent a predicted brightness increment model. From this predicted model we can find that if the motion is parallel to the edge (i.e., $V(u) \parallel \nabla L(u)$), the increment vanishes and no events are generated. And if the motion is perpendicular to the edge (i.e., $V(u) \perp \nabla L(u)$), the camera generates events at the highest rate. Fig. 2 presents an example of EGM output with positive brightness changes showing as white and negative ones showing as black.

IV. System Design

A. System Overview

Fig. 3 shows the overview of FlyTracker. The main target of FlyTracker is to track the ego motion of a drone as well as continuously detect relative positions of moving obstacles that appear in FoV by using a monocular event camera carried by the drone. At first, images are sent to the moving object detection module. In this module, an online background-subtraction-based method is used to detection whether there is any moving obstacle appearing in the FoV of the drone. Once a moving obstacle is found by FlyTracker, a mask of this moving object is calculated to distinguish it from the background part on the image plane. With the mask, images and events are divided into background parts and foreground parts. The background parts are then sent to the ego-motion tracking module. In this module, photometric features based on EGM are first extracted and then used to calculate brightness increment errors. Next, camera pose changes are estimated by minimizing the errors. Finally, through back-end optimization, the ego-motion tracking module outputs optimized camera poses as well as an estimated depth map. At the same time, the foreground parts are sent to the moving obstacles tracking module. Same photometric features are extracted and camera pose changes relative to the obstacle are estimated. By an invertion operation, the positions of the obstacle relative to

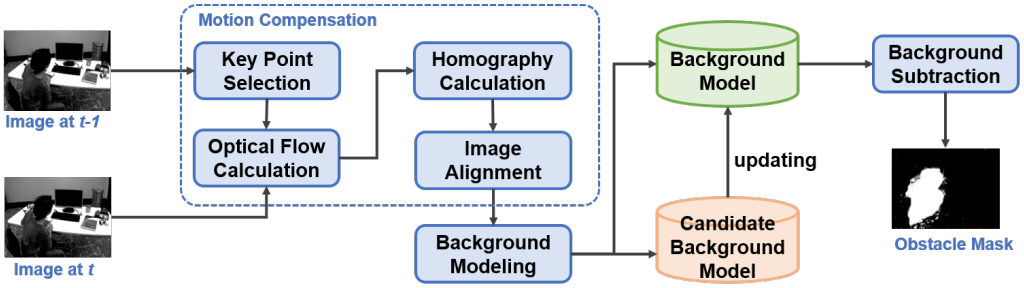


Fig. 4: Workflow of moving obstacle detection.

the drone are obtained. The camera poses changes calculated from the two tracking module are compared to eliminate false results outputting from the detection module. It should be noted that if no moving obstacle is detected, only the drone's 6-DoF poses is tracked. We test FlyTracker in scenarios with one moving obstacle each time due to the limited resolution of our event camera. Theoretically, if there appears multiple moving obstacles, multiple masks will be generated and different foreground parts are then processed separately.

B. Moving Obstacle Detection

Moving obstacle detection is the first step for a drone to avoid flight risks like crushing or tumbling. Considering the fast flight speed of drones, a mechanism that can detect surrounding moving objects in real time is much important for drones. Comparing with classification methods with pre-training operations, background subtraction, which is one of the basic tasks in visual processing, is lightweight and useful to narrowing down the searching range for object detection, recognition and tracking. Thus, an online background-subtraction-based method for moving obstacle detection is first proposed in our system. Fig. 4 shows the workflow of the online background-subtraction-based method. The proposed method mainly consists of three steps, which are motion compensation, background modeling and background subtraction.

Motion Compensation. The goal of this step is to estimate the homography matrix between a frame and its previous frame. We denote the current frame as X_t generated at time t and its previous frame as X_{t-1}. The homography matrix between them is denoted as H. We first extract key points of X_{t-1}. Specifically, the image of X_{t-1} is averagely divided with m grids, and the vertexes of each grid are selected as key points. Noted that the amount of grids should keep a trade-off between accuracy and computational complexity of homography matrix estimation. With the key points of X_{t-1}, then, three-layer Lucas-Kanade optical flow is calculated to find the corresponding points in frame X_t. Thus, H can be calculated robustly by OpenCV. Finally, we align X_{t-1} with X_t as X_a by using H.

Background Modeling. We propose a practical background model, denoted as M to perform background subtraction. We also propose a supplementary background model, denoted as $\hat{M}$, to evaluate whether an image pixel u is stable enough for background based on its supplementary age α_u. At the beginning, M and $\hat{M}$ are both set uninitialized. The process of background modeling can be divided into three steps.

- **Initialization**: Each pixels u in $\hat{M}$ is set as the corresponding pixel in the first frame X_1, which is $\hat{M}(u) = X_1(u)$. Also, the supplementary age α_u of each $\hat{M}(u)$ is set to 0 and will increase according to some conditions. Once α_u is larger than a threshold θ_α, we set $M(u) = \hat{M}(u)$ and α_u is reset to 0.
- **Updating**: When a new frame X_t appears, similarities between $\hat{M}_t(u)$, $M_t(u)$ and $X_t(u)$ comparing with a threshold θ_p is measured. Specifically, whenever $\|X_t(u) - M_t(u)\| < \theta_p$, then $M_t(u)$ is updated by a learning rate β. Formally, $M_t(u) = (1-\beta)M_{t-1}(u) + \beta X_t(u)$. When the condition of $M_t(u)$ is not met, we next measure whether $\|X_t(u) - \hat{M}_t(u)\|$ is smaller than θ_p. If $\|X_t(u) - \hat{M}_t(u)\| < \theta_p$,we set $\hat{M}_t(u) = X_t(u)$ and increase α_u with a unit value. Otherwise, we set $X_t(u) = \hat{M}_t(u)$. It should noted that whenever $\alpha_u > \theta_\alpha$, $\hat{M}(u)$ is copied to $M(u)$ and α_u is reset to 0.
- **Warpping**: The background model $M_t(u)$ and supplementary background model $\hat{M}_t(u)$ are transferred by using the homography matrix H.
- **Repeating**: The second and third steps are repeated until the end of video shoot.

Background Subtraction. With a given segmentation threshold θ_s, background subtraction is carried out pixel by pixel to obtain masks of moving objects. Formally,

$$Mask(u) = \begin{cases} 1, & \|X(u) - M(u)\| > \theta_s \\ 0, & \text{otherwise} \end{cases} . \quad (5)$$

Finally, a simple median filter is used to smooth the mask. Through the above process, moving obstacles that appear in the FoV of the drone can be quickly detect and distinguished from static background on the image plane.

C. Ego-motion Tracking

In the process of ego-motion tracking, FlyTracker only focuses on the image part of static background and events that occur in background areas. Fig. 5 shows the workflow of ego-motion tracking module. To initialize this module, we apply eight-point algorithm, which is classical algorithm in multiview geometry model, on the beginning several image frames until accumulating enough parallax. Once initialization

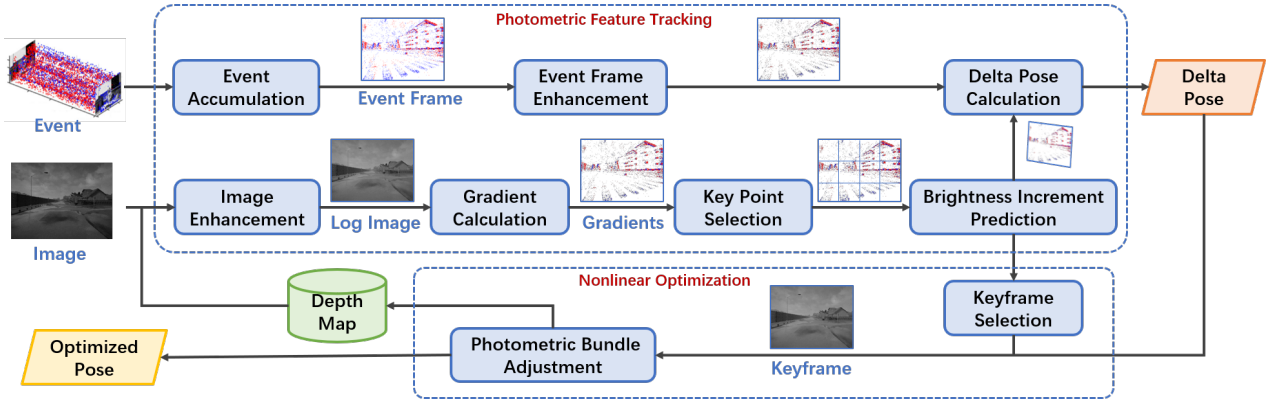


Fig. 5: Workflow of ego-motion tracking.

is successful, these image frames become keyframes and map points are generated. Then, Bundle Adjustment (BA) is performed on the keyframes to refine the map. We select median depth to normalize the scale since it is known that scale in monocular visual odometry is arbitrary. After initializing the module, photometric features of images and events are deeply fused to estimate camera poses.

Extracting Event Feature. Events with a quantity of N_e are first accumulated to build an event frame (i.e., brightness increment image $\Delta L(u)$). N_e should be selected carefully as a small N_e can not provide sufficient SNR while a large N_e may introduce much accumulation blur. In practice, FlyTracker enhances the event frames by selecting slightly large N_e that ensures high SNR and then weighting the polarities of each event to decrease accumulation blur. That is, $\Delta L(u) = \sum_{t_k \in \Delta\tau} w_k p_k C \delta(u - u_k)$ and w_k subjects to a Gaussian distribution with expectation of $\frac{1}{2}\Delta\tau$. Thus, we obtain the photometric feature $\Delta L(u)$ of events which is also named brightness increment model in EGM.

Extracting Image Feature. According to EGM, brightness increments are caused by moving edges. So image contours are selected for brightness increment prediction. Specifically, an image frame is first enhanced by logarithmic transformation and normalization. Then, brightness gradient of the image is calculated. Next, we select 20% pixels with the largest brightness gradient in randomly divided and nonoverlapping areas, which are more likely belong to contours. If given predicted pixel velocities (i.e., optical flow) to the selected key points, FlyTracker obtains a predicted brightness increment model $\hat{\Delta} L(u)$, which can be fused with brightness increment model to estimate camera pose changes.

Camera Pose Estimation. The pixel velocity $V(u)$ of each key point is related to the camera's linear velocity ν and angular velocity ω, denoted by a 1×6 vector $\xi = (\nu^\top, \omega^\top)^\top$ for convenience, and further related to camera pose change ΔT. According to pinhole camera model, a 3D point $U = (U_x, U_y, U_d)^\top$ is mapped into a image point $u = (u_x, u_y)^\top$ described by $u = \pi(U)$, where U_d is scene depth and π is the camera intrinsic. Let the relative motion between the viewing camera and the scene be described by

$$\frac{dU}{dt} = -\hat{\omega}(t)U(t) - \nu(t), \tag{6}$$

where $\hat{\omega}$ is the cross-product matrix of ω. Giving the corresponding point depth d_u, the relationship between the pixel velocity and the camera velocity is

$$V(u) = J(u, d_u)\xi, \tag{7}$$

where $J(u, d_u)$ is the 2×6 image Jacobian matrix:

$$J(u, d_u) = \begin{bmatrix} -d_u^{-1} & 0 & u_x d_u^{-1} & u_x u_y & -1 - u_x^2 & u_y \\ 0 & -d_u^{-1} & u_y d_u^{-1} & 1 + u_x^2 & -u_x u_y & -u_x \end{bmatrix}. \tag{8}$$

By combining (7) and (4), we get a predicted brightness increment model as

$$\hat{\Delta} L(u) = -\nabla L(u) \cdot J(u, d_u)\xi\Delta\tau. \tag{9}$$

It is worth noting that camera pose T and camera velocity ξ are global quantities shared by all image pixels u. With brightness gradients $\nabla L(u)$ and depth information d_u known (after the depth map is initialized), we cast the camera pose estimation problem as a joint optimization by minimize the difference between the brightness increment model (which comes from events) and the predicted brightness increment model (which comes from images). Formally,

$$\min_{\Delta T, \xi} \left\| \frac{\Delta \hat{L}}{\|\Delta \hat{L}\|_2} - \frac{\Delta L}{\|\Delta L\|_2} \right\|_{\mathcal{H}}, \tag{10}$$

where $\|\cdot\|_{\mathcal{H}}$ is Huber norm. Finally, we use Ceres solver to minimize the above joint optimization and get the camera pose changes relating to each event frame. We denote the camera pose changes of each event frame as delta camera pose in this paper.

Back-end Optimization. Back-end optimization of camera poses and depth map is designed at the back end through using keyframes of images. After the initialization of the ego-motion tracking module, a keyframe is created when the selected key points decrease over 30% that may due to moving forward or rotation. FlyTracker keeps a sliding window with

TABLE I: Evaluation Scenarios

Scenario Name	Scenario Type	Ground Truth Source	Description
Desk	static	OptiTrack	a desk with books, pens and a monitor
Office	static	OptiTrack	a typical office room
Building	static	RGBD Camera	building channels with some weak-texture walls and doors
Atrium	simulated	simulated	simulate an ancient architecture by ESIM [43]
Reader Moving	dynamic	OptiTrack	a person is moving in an office room
Day Flying	dynamic	RGBD Camera	outdoor with moving cars and tested obstacles
Night Flying	dynamic	RGBD Camera	outdoor with tested obstacles in low light

5 keyframes. For the keyframes in the sliding window, non-linear optimization is performed through Photometric Bundle Adjustment (PBA), which is usually used in direct methods of visual odometry. Specifically, let the photometric error of a point u in reference image frame $\mathcal{K}_i$ observed in a target frame $\mathcal{K}_j$ be defined as:

$$E_{uj} = \left\| \mathcal{K}_{\hat{L}_i}(u) - \mathcal{K}_{\hat{L}_j}(u') \right\|_{\mathcal{H}}, \tag{11}$$

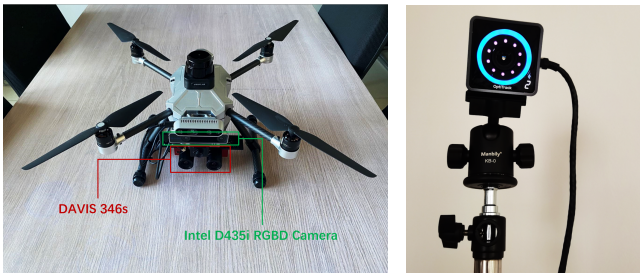
where $\mathcal{K}_{\hat{L}_i}$ and $\mathcal{K}_{\hat{L}_j}$ represent the point brightness in $\mathcal{K}_i$ and $\mathcal{K}_j$, respectively. Also, u' stands for the projected point in $\mathcal{K}_j$ of u. It should be noted that the projection depends on (1) the point's depth d_u, (2) the camera intrinsics π, and the camera pose change T_ij involved with $\mathcal{K}_i$ and $\mathcal{K}_j$. Thus, PBA is to minimize full photometric error over all keyframes and key points to optimize camera poses and the depth map, that is

$$\min \sum_{i \in \mathcal{K}} \sum_{u \in N_u} \sum_{j \in obs(u)} E_{uj}. \tag{12}$$

In the objective function, i runs over all keyframes in the sliding window $\mathcal{K}$, u runs over all selected key points N_u in keyframe $\mathcal{K}_i$, and j runs over all keyframes $obs(u)$ in which u is visible. PBA can also be implemented by using Ceres solver.

D. Moving Obstacle Tracking.

The basic idea of the moving obstacle tracking module is similar to that of ego-motion tracking module. FlyTracker focuses on the foreground parts of images and events in this module. First, photometric features of image and events are extracted according to EGM. Then, delta camera poses relative to the moving obstacle are calculated like the same process in ego-motion tracking module. Next, FlyTracker inverses the delta camera poses to get the obstacle pose changes relative to the camera. Considering that the moving object may not be a rigid body, we only track the obstacle's position relative to the camera while discarding orientation information. Module initialization is carried out at the phase of detecting moving objects and no back-end optimization is designed in this module. In order to eliminate false detection results outputting from the moving object detection module, we compare the camera traces in the same duration that calculated from the above tracking module. The detection results are verified true if the two traces are obviously different.



(a) drone with cameras (b) motion capture system

Fig. 6: Experiment devices.

V. Implementation and Evaluation

A. Evaluation Setup

Data Collection. We evaluate FlyTracker in different environments including static scenarios (with no moving object) and dynamic scenarios (with moving objects or HDR scenarios). Visual data are collected with a stereo DAVIS 346 carried by an AMOV-P450 drone, and ground truth are collected by a motion capture system with four OptiTrack cameras and an Intel D435i RGB-Depth-Camera. Evaluation scenarios are summarized in Table I and experiment devices are shown in Fig. 6.

Baseline Methods. We compare FlyTracker with image-only, event-only and information-fused VO methods. Three baselines are selected which are ORB-SLAM, DSO, and USLAM.

- ORB-SLAM [12] is the best-known image-based and indirect visual SLAM system that depends on ORB features. We implement the monocular VO part of ORB-SLAM, which mainly includes its tracking module and local mapping module.
- DSO [14] is an image-based and direct monocular VO method. It combines a fully direct probabilistic model with consistent and joint optimization of model parameters such as inverse depth and camera motion. That is, the basic idea of DSO is to estimate camera motion by minimizing photometric errors between keyframes.
- USLAM [33] is an indirect monocular SLAM pipeline that combines events, images and IMU measurements. It treats event frames as normal images and tracks FAST corners separately on event frames and image frames.
- FlyTracker create an event frame using $20k$ events in the ego-motion tracking module and using $5k$ events in the

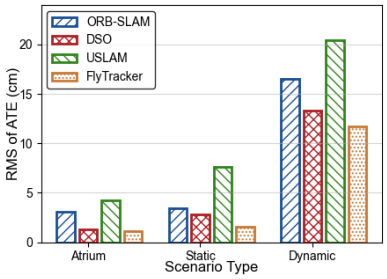


Fig. 7: Overall performance comparison.

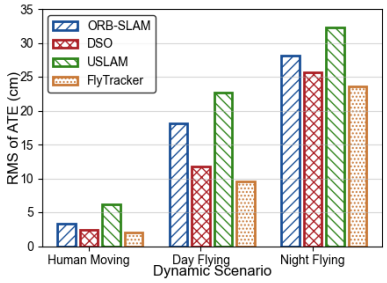


Fig. 8: ATE under dynamic scenarios.

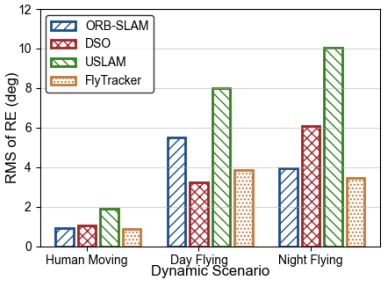


Fig. 9: RE under dynamic scenarios.

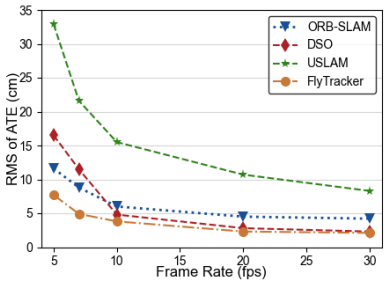


Fig. 10: ATE under different frame rates.

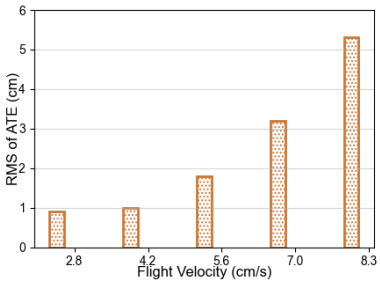


Fig. 11: ATE with flight velocities.

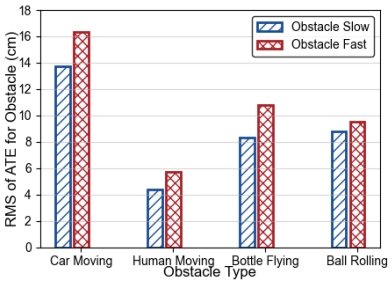


Fig. 12: ATE of obstacle motion tracking.

moving obstacle tracking module. Each event frame has 50% events overlapping with its previous or next event frame. Images are generated at a fixed rate of $30fps$. Other parameters are set with $m = 1024$, $\beta = 0.8$, $\theta_\alpha = 15$, $\theta_p = 18$ and $\theta_s = 50$.

Evaluation Metrics. FlyTracker and baselines are evaluated by two standard metrics: Root Mean Square (RMS) of Absolute Trajectory Error (ATE) and RMS of Rotation Error (RE). Monocular methods are operated with a scale transformation to get estimated absolute trajectory and tested using data from the left camera. Each absolute trajectory error and rotation error are calculated by comparing the estimated poses of keyframes and the ground truth.

B. Overall Performance

Fig. 7 shows the overall performance of FlyTracker compared with the three baselines. The average RMS of ATE under three static scenarios and three dynamic scenarios, as well as RMS of ATE from the simulated visual data are presented in the figure, respectively. From the figure, it can be seen that FlyTracker outperforms the three baselines in all tested scenarios. ATEs of all methods in dynamic scenarios are much higher than those in static scenarios, suggesting that dynamic scenarios are always challenging for vision-based tracking solutions. However, FlyTracker also has the lowest ATE in dynamic scenarios among baselines, indicating that the multilevel fusion method of FlyTracker can effectively improve tracking accuracy in different environments. Though USLAM fuses with the most sensors, its performance is not desirable since the data association problem it has addressed is a much challenging problem for feature-based methods. This is because each event carries little information and it is hard to identify which events are triggered by the same scene point. DSO performs better than both ORB-SLAM and USLAM for it can work relatively reliably in scenarios lacking corners or with many weak-texture areas, where is not friendly to feature-based methods. When the illumination in a scenario changes fast or the captured images are obvious blur, the tracking accuracy of DSO will also decrease.

Then, we compare FlyTracker with baselines in dynamic scenarios, including indoor environment with moving objects, outdoor environment with moving objects and weak lighting environment. Fig. 8 and Fig. 9 show the evaluation results in terms of ATE and RE, respectively. In general, FlyTracker outperforms the baseline in all scenarios. Obviously, poor illumination has the most impact on motion tracking, since the image quality decrease a lot under this condition. The brightness gradients become indistinct and may disappear in some areas, which makes the quantity of selected key points decrease for FlyTracker and further decrease tracking accuracy. However, FlyTracker has ATE lower by 16.0%, 7.8% and 26.9%, and RE lower by 12.4%, 43.3% and 65.5% than ORB-SLAM, DSO and USLAM in the night flying scenario, respectively.

Considering that high motion speed of the camera usually makes the scenery captured in images largely apart, we compare motion tracking accuracy of the baselines when image frames are more different with each other. The comparison is performed through decreasing the frame rate of visual data collected in the office room. Evaluation results are shown in Fig. 10. It can be seen from the figure that tracking accuracy of all methods decreases with the decrease of image frame rate.

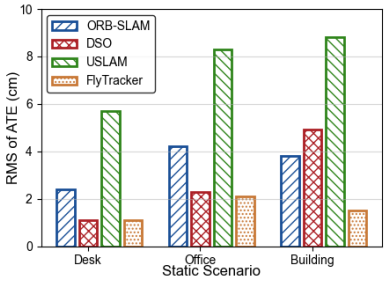


Fig. 13: ATE under static scenarios.

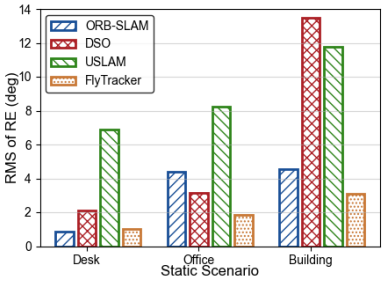


Fig. 14: RE under static scenarios.

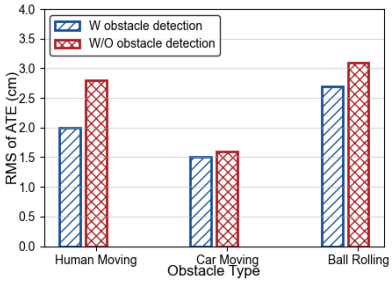


Fig. 15: ATE with and without obstacle detection.

Specifically, The ATE of FlyTracker is $2.1cm$ at $30fps$ and increases to $5.7cm$ when frame rate decreases to $5fps$, which is still higher than those of baselines at $5fps$. It is because for FlyTracker, events are largely generated between image frame, making the photometric features still can be tracked in the blind time between image frames.

C. Robustness Study

We conduct a evaluation in a static outdoor environment to study the impact of drone flight velocity on the performance of motion tracking. Fig. 11 shows the RMS of ATE under different drone flight velocities. It is demonstrated that tracking accuracy decreases with the increase of flight velocity. The RMS of ATE is $5.3cm$ at $8.3cm/s$, which is higher by 83% than the ATE at $2.8cm/s$. However, the tracking performance can still satisfy plenty of applications.

In addition, we study the performance of motion tracking for moving obstacles. Four typical obstacles are selected, which are a moving car, a walking person, a flying bottle thrown by people and a rolling basketball. RMS of ATE for obstacle tracking is evaluated when they move with a relatively slow speed and a relatively fast speed (but not same for different kinds of obstacle). Evaluation is conducted in the same outdoor environment and the results are shown in Fig. 12. As a whole, tracking accuracy is higher when the obstacle moves at a fast speed than that at a slow speed. Tracking accuracy of the moving car is the lowest since the distance between the car and the drone is the farthest, that brings slightly higher error in depth estimation. Tracking the walking person achieves the highest accuracy. The reason may that the walking person occupies a large area in FoV, and the abounding texture of the person produces evident brightness gradients.

D. Module Study

We compare FlyTracker with baselines in static scenarios, in which there is no effect of moving obstacle. Fig. 13 and Fig. 14 show the RMS of ATE and RE under four static scenarios, respectively. We can see that FlyTracker outperforms the baselines under the four scenarios both in terms of trace estimation and rotation estimation. The ATE of FlyTracker is $1.5cm$ in building scenario, which is lower by 60.5%, 69.4% and 82.9% than ORB-SLAM, DSO and USLAM, respectively. The performance of FlyTracker benefits from the advantage of tight fusion of events and images.

Finally, we study whether the moving obstacle detection affect the motion tracking accuracy. Evaluation is conducted in a scenario with a walking person, a moving car and a rolling basketball, and FlyTracker perform motion tracking with and without the moving obstacle detection module. Fig. 15 shows that tracking ATE is lower with the detection module than that without the detection module in all conditions, demonstrating that the detection module does improves tracking accuracy for FlyTracker. The improvement with a moving car is smaller than that with a walking person and a rolling ball, as the car is the smallest in the FoV of the drone.

VI. Conclusion

This paper proposes the design and implementation of FlyTracker, a 6-DoF pose tracking system for drone flight control to work in challenging scenarios. FlyTracker exploits the idea of event generation model and further design an accurate motion tracking method by tightly fusion event feature and image feature. To reduce the impact of moving objects on tracking accuracy, a real-time method for moving object detection is proposed to distinguish foreground and background part. Extensive evaluations are conducted in real environment, and the results show that FlyTracker can effectively and reliably track the 6-DoF pose of the drone as well as monitor relative positions of moving obstacles across various scenarios.

The main limitation of our work is the time consumption that FlyTracker performs motion tracking. Though the moving obstacle detection is real time and tracking the motion of obstacles is very fast, FlyTracker is not a real time motion tracking system due to the large number of parameters used in back-end optimization. In our future work, we will combine the FlyTracker with an edge framework, leveraging the computation ability of edge server to speed up the process of motion tracking of FlyTracker.

Acknowledgment

The work of Yue Wu is partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62202262. The work of Fan Li is partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62072040. Zheng Yang is the corresponding author.

REFERENCES

[1] G. Chi, Z. Yang, J. Xu, C. Wu, J. Zhang, J. Liang, and Y. Liu, "Wi-drone: Wi-fi-based 6-dof tracking for indoor drone flight control," in *ACM International Conference on Mobile Systems, Applications, and Services*, 2022, pp. 1–1.

[2] B. Pardhasaradhi and L. R. Cenkeramaddi, "GPS spoofing detection and mitigation for drones using distributed radar tracking and fusion," *IEEE Sensors Journal*, vol. 22, no. 11, pp. 11 122–11 134, 2022.

[3] J. Park, D. H. Kim, Y. S. Shin, and S.-h. Lee, "A comparison of convolutional object detectors for real-time drone tracking using a PTZ camera," in *International Conference on Control, Automation and Systems*, vol. 1, no. 1, 2017, pp. 696–699.

[4] R. Mur-Artal and J. Tardos, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2016.

[5] 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, 2018.

[6] D. Li, J. Xu, Z. Yang, Q. Zhang, Q. Ma, L. Zhang, and P. Chen, "Motion inspires notion: Self-supervised visual-LiDAR fusion for environment depth estimation," in *ACM International Conference on Mobile Systems, Applications, and Services*, 2022, pp. 1–1.

[7] J. Park, K. Joo, Z. Hu, C. Liu, and I. Kweon, "Non-local spatial propagation network for depth completion," in *European Conference on Computer Vision*, 2020, pp. 120–136.

[8] L. O. Russo, G. A. Farulla, M. Indaco, S. Rosa, D. Rolfo, and B. Bona, "Blurring prediction in monocular slam," in *IEEE Design and Test Symposium*, 2013, pp. 1–6.

[9] F. Zhong, S. Wang, Z. Zhang, C. Chen, and Y. Wang, "Detect-SLAM: Making object detection and SLAM mutually beneficial," in *IEEE Winter Conference on Applications of Computer Vision*, 2018, pp. 1001–1010.

[10] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240 x 180 130 db 3us latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[11] A. Pretto, E. Menegatti, M. Bennewitz, W. Burgard, and E. Pagello, "A visual odometry framework robust to motion blur," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 2250–2257.

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[13] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An open-source SLAM system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[14] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[15] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[16] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *European Conference on Computer Vision*, 2006, pp. 404–417.

[17] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *European Conference on Computer Vision*, 2010, pp. 778–792.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: an efficient alternative to SIFT or SURF," in *IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571.

[19] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, 2006, pp. 430–443.

[20] M. Buczko and V. Willert, "Monocular outlier detection for visual odometry," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 739–746.

[21] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 15–22.

[22] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *IEEE International Symposium on Circuits and Systems*, 2007, pp. 845–848.

[23] Z. Pacoret, R. Benosman, S. Ieng, and R. S, "Asynchronous event-based high speed vision for microparticle tracking," *Journal of Microscopy*, vol. 245, no. 3, pp. 236–244, 2012.

[24] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. Douglas, and T. Delbruck, "A pencil balancing robot using a pair of aer dynamic vision sensors," in *IEEE International Symposium on Circuits and Systems*, 2009, pp. 781–784.

[25] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier, "Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1081–1089, 2012.

[26] Z. Ni, S.-H. Ieng, C. Posch, S. Regnier, and R. Benosman, "Visual tracking using neuromorphic asynchronous event-based cameras," *Neural Computation*, vol. 27, no. 4, pp. 925–953, 2015.

[27] 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, 2015.

[28] A. Glover and C. Bartolozzi, "Robust visual tracking with a freely-moving event camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 3769–3776.

[29] A. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1–1.

[30] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza, "Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS)," in *International Conference on Event-based Control, Communication, and Signal Processing*, 2016, pp. 1–7.

[31] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *British Machine Vision Conference*, 2017, pp. 1–12.

[32] 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.

[33] 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.

[34] S. Bryner, G. Gallego, H. Rebecq, and D. Scaramuzza, "Event-based, direct camera tracking from a photometric 3d map using nonlinear optimization," in *International Conference on Robotics and Automation*, 2019, pp. 325–331.

[35] D. Gehrig, M. Ruegg, M. Gehrig, J. Carrio, and D. Scaramuzza, "Combining events and frames using recurrent asynchronous multimodal networks for monocular depth prediction," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2822–2829, 2021.

[36] L. Pan, R. Hartley, C. Scheerlinck, M. Liu, X. Yu, and Y. Dai, "High frame rate video reconstruction based on an event camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2519–2533, 2022.

[37] C. Scheerlinck, N. Barnes, and R. Mahony, "Continuous-time intensity estimation using event cameras," in *Asian Conference on Computer Vision*, 2018, pp. 308–324.

[38] S. Tulyakov, D. Gehrig, S. Georgoulis, J. Erbach, M. Gehrig, Y. Li, and D. Scaramuzza, "Timelens: Event-based video frame interpolation," in *International Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1–14.

[39] H. Kim, S. Leutenegger, and A. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *European Conference on Computer Vision*, 2016, pp. 349–364.

[40] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "Emvs: Event-based multi-view stereo—3d reconstruction with an event camera in real-time," *International Journal of Computer Vision*, vol. 126, no. 1, pp. 1394–1414, 2017.

[41] 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. 6, pp. 593–600, 2017.

[42] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger, "Interacting maps for fast visual interpretation," in *International Joint Conference on Neural Networks*, 2011, pp. 770–776.

[43] H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," in *Conference on Robot Learning*, 2018, pp. 1–14.