FAST-Dynamic-Vision:
Detection and Tracking Dynamic Objects with Event and Depth Sensing

Botao He[† 3,4], Haojia Li[† 3,5], Siyuan Wu[3,6], Dong Wang[1,3],
Zhiwei Zhang[1,3], Qianli Dong[3,5], Chao Xu[1,2,3], and Fei Gao[1,2,3]

Abstract— The development of aerial autonomy has enabled aerial robots to fly agilely in complex environments. However, dodging fast-moving objects in flight remains a challenge, limiting the further application of unmanned aerial vehicles (UAVs). The bottleneck of solving this problem is the accurate perception of rapid dynamic objects. Recently, event cameras have shown great potential in solving this problem. This paper presents a complete perception system including ego-motion compensation, object detection, and trajectory prediction for fast-moving dynamic objects with low latency and high precision. Firstly, we propose an accurate ego-motion compensation algorithm by considering both rotational and translational motion for more robust object detection. Then, for dynamic object detection, an event camera-based efficient regression algorithm is designed. Finally, we propose an optimization-based approach that asynchronously fuses event and depth cameras for trajectory prediction. Extensive real-world experiments and benchmarks are performed to validate our framework. Moreover, our code will be released to benefit related researches.

I. INTRODUCTION

Thanks to recent progress in aerial autonomy, UAVs have been able to fly agilely in complex environments such as mine exploration. Drones are able to perceive unknown environments and plan an exploration path autonomously. However, perception in dynamic environments, especially with high-speed objects, is still a challenging problem. For example, drones have difficulty dodging a rock falling head-on during the fast mine exploration.

For fast-moving object's avoidance, it's pivotal to track them and predict their future trajectories in a short latency. Normally, this latency is hundreds of milliseconds for most perception methods: cameras need tens of milliseconds to expose and suffer from motion blur; besides, algorithms need a sequence of frames to predict a trajectory. However, for objects with speed higher than 10 meters per second, such long latency leaves drones no time to escape. In order to reduce this latency, sensors with a higher temporal resolution

† **Equal contribution.**
1 State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China.
2 Huzhou Institute of Zhejiang University, Huzhou, 313000, China.
3 Nation Engineering Research Center for Industrial Automation (Ningbo Institute), Ningbo, 315000, China
4 School of Automation, Nanjing Institute of Technology, Nanjing, 211112, China.
5 Faculty of Robot Science and Engineering, Northeastern University, Shenyang, 110207, China.
6 Faculty of Electronic and Information Engineering, Xian Jiaotong University, Xi'an, 710049, China.
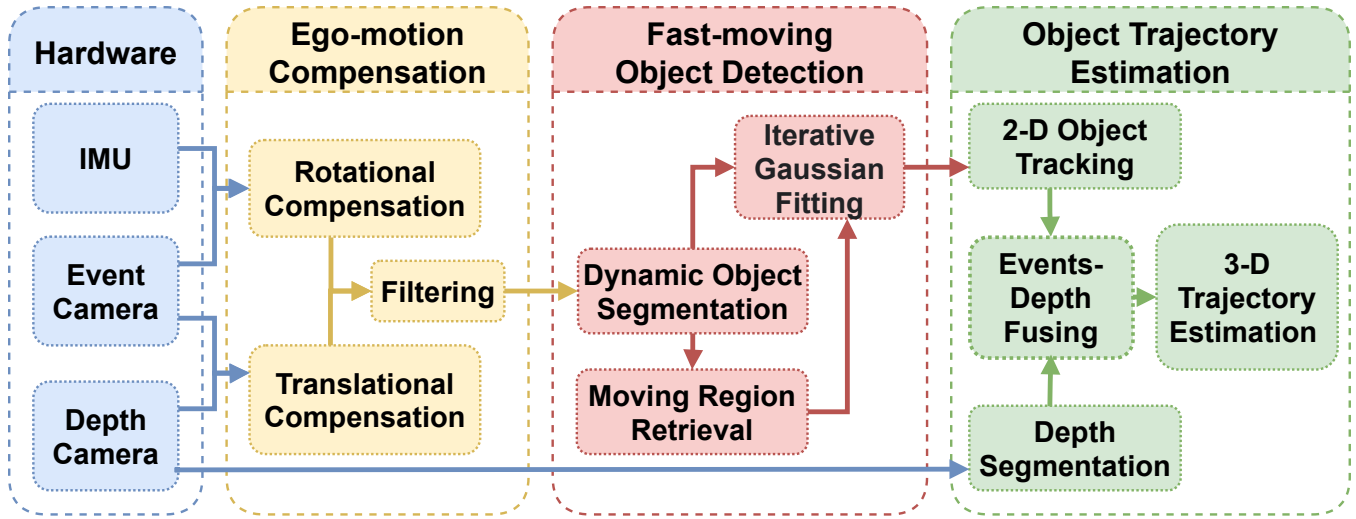Email:{cxu, fgaoaa}@zju.edu.cn

Fig. 1. A composed image of real-world dodging experiment to validate our detection and tracking system. Please refer to our video submission for more details.

are keenly demanded. Meanwhile, a real-time detection and tracking algorithm is also indispensable.

To fill this research gap, we adopt the event camera, an asynchronous motion-activated sensor providing a microsecond-level temporal resolution, for solving this problem. In this work, a complete perception system integration for this sensor is also designed. Firstly, we propose an ego-motion compensation algorithm to alleviate the noise. Then, for dynamic object detection, we develop a regression-based approach to find the region of interest (ROI). Notably, this approach is more robust and less computational demanding compared to other clustering-based methods in [1].

Furthermore, a satisfactory solution should be capable of tracking the object in the 2D camera space and estimating its corresponding 3D trajectory [2]. To address the scale ambiguity issue, we further incorporate a depth camera to recover the scale of monocular sensing by joint optimization. Afterward, combining event and depth observations, we present an accurate trajectory estimator which significantly increases the robustness and accuracy. Our algorithm successfully balances the tight onboard computational budget and trajectory accuracy.

We perform extensive quantitative and qualitative experiments in high dynamic scenarios to validate our object detection and trajectory estimation framework, which provide a solid foundation for fast-moving object avoidance.

This paper highlights several features:

1) An advanced motion compensation method for event-detection balancing efficiency and accuracy.

Fig. 2. The overview of our detection and tracking system.

2) A 3D trajectory estimation approach that fuses event and depth information asynchronously.
3) A complete system integration with open source [1].

II. RELATED WORK

Due to its unique advantage of high temporal resolution and no motion blur, the event camera has attracted many researchers' interest [3] [4]. The first problem to solve is how to represent events. Existing event representation algorithms can be divided into two main categories. One category is classifying events of different objects into several clusters [5] [6] [7] [8] [9]. This sort of method is intuitively built on event mechanisms, but it is sensitive to noisy events. It also neglects time information which is vital for event-based detection. The other category is tracking features on time surface frames [1] [10] [11] [12], which is a 2D map only containing the latest event's timestamp while ignoring others triggered recently in each pixel. Specifically, some researchers [1] [10] introduces a mean-time image representation contains the average timestamp of the events. This mean-time image is less computational demanding than other types, e.g., exponential time surfaces[13]. Furthermore, this representation is more suitable for object detection tasks: regions containing moving objects can be obtained by merely thresholding the mean-time image. Therefore, our detection method is based on it.

To remove background events generated by rotation and translation, ego-motion compensation is necessary for moving object detection and tracking. Mitrokhin et al. [10] minimize error functions provided by spatial gradient of mean-time image to fit a parametric motion model; Gallego et al. [11] maximize a variance which represents local contrast, in other words, sharpness, on the compensated image. Zhou et al. [14] minimize an energy function. The optimization-based method is accurate. However, one drawback of this method

is that its high computational cost introduces extra latency in the perception system [1], which would lead to potential failure in our object avoidance scene. Falanga et al. [1] use IMU's angular velocity average to perform rotational ego-motion compensation. This method is less computationally demanding so that it can be applied for onboard flights while the accuracy is not guaranteed in forwarding flights. Based on this method [1], we improve the motion compensation approach by fusing depth and IMU data to implement both rotational and translational ego-motion compensation. Our method can enhance its accuracy and reliability without sacrificing computational efficiency.

For object tracking and trajectory estimation, our framework is inspired by the following studies. Su et al. [2] fit a parabolic model to estimate the 3D trajectory of a flying object from noisy 2D observation. This method requires plenty of observations due to the lack of depth information, which cannot meet the requirement for low-latency. Falanga et al. [1] apply stereo event cameras for 3D position estimation. However, this configuration does not guarantee accuracy and robustness because the high level of noise causes uncertainty in depth estimation. To obtain more accurate 3D trajectories, we design a different configuration fusing event and depth sensor onboard.

III. OVERVIEW

A. System Architecture

The pipeline of our framework is illustrated in Fig 2. There are three procedures in this framework: ego-motion compensation, object detection, and object trajectory estimation. Firstly, we implement an advanced motion compensation algorithm fusing IMU and depth data to filter out background events generated by ego-motion, including rotation and translation during flight. The mean-time image can be generated by motion-compensated events. Each pixel value of this mean-time image is the average timestamp of corresponding events. Following the motion compensation

---

[1] Our code and video can be found at https://github.com/ZJU-FAST-Lab/FAST-Dynamic-Vision

step, we detect and locate the region with the largest average timestamp in the mean-time image. This region represents the area with the fastest speed on the image plane. To obtain the region's bounding box, we introduce an iterative Gaussian fitting algorithm for the object detection step. We also present a moving region retrieval to guarantee the bounding box we get is the most accurate one. Next, the moving object's location is tracked with Kalman Filter on the 2D plane, and the object is segmented out on the depth map according to the detection result. Then, we optimize the trajectory of the object by minimizing reprojection residuals. Finally, to validate our estimation, we design a scenario in which a UAV autonomously detect and avoid objects flying towards it.

The rest of this paper is organized as follows: Section IV-A presents our advanced ego-motion compensation algorithms. Then we discuss object detection and tracking methods used in this framework (Section IV-B). In Section IV-C, we perform our 3D trajectory estimator fusing event stream and depth information. Section V depicts our real flight experiment and compares our performance with others.

B. Notation

Let $C \in \mathbb{R}^3$ denote a set of events. We use symbols $(x, y, t) \in C$ to denote an event triggered by an event camera. The symbol x, y represents the event's coordinate on the image plane, t denotes the timestamp of the event.

We represent $\xi_{i,j}$ as a set of motion-compensated events (see IV-A) which are projected onto the same pixel (i, j)

$$\xi_{ij} = \{\{x', y', t\} : \{x', y', 0\} \in C', i = x', j = y'\}. \quad (1)$$

Therefore, the event-count image pixel [10] can be denoted as $\mathcal{I}_{i,j}$ where

$$\mathcal{I}_{i,j} = |\xi_{i,j}|. \quad (2)$$

We also define the time-image as $\mathcal{T}$. Hence, pixel (i, j) in the time-image represented as $\mathcal{T}_{i,j}$, can be expressed as the average timestamp of events triggered in this position, as follows:

$$\mathcal{T}_{i,j} = \frac{1}{\mathcal{I}_{i,j}} \sum t : t \in \xi_{i,j}. \quad (3)$$

We name the normalized time-image $\mathcal{T}$ as normalized mean-time image $\mathcal{N}$, which can be computed by the following equation [1]

$$\mathcal{N}_{i,j} = \frac{\mathcal{T}_{i,j} - \min_{(i,j) \in \mathcal{T}} \mathcal{T}_{i,j}}{\max_{(i,j) \in \mathcal{T}} \mathcal{T}_{i,j} - \min_{(i,j) \in \mathcal{T}} \mathcal{T}_{i,j}}. \quad (4)$$

We use (W) as the world frame, (B) as the drone body frame. Notably, we use (E) to represent the event camera frame while (D) representing the depth camera frame. Hence, we can represent the transformation $^W\mathbf{T}_E$ of the event camera in the world frame as

$$^W\mathbf{T}_E = \begin{bmatrix} ^W\mathbf{R}_E & ^W\mathbf{t}_E \\ \mathbf{0}^\top & 1 \end{bmatrix}. \quad (5)$$

.

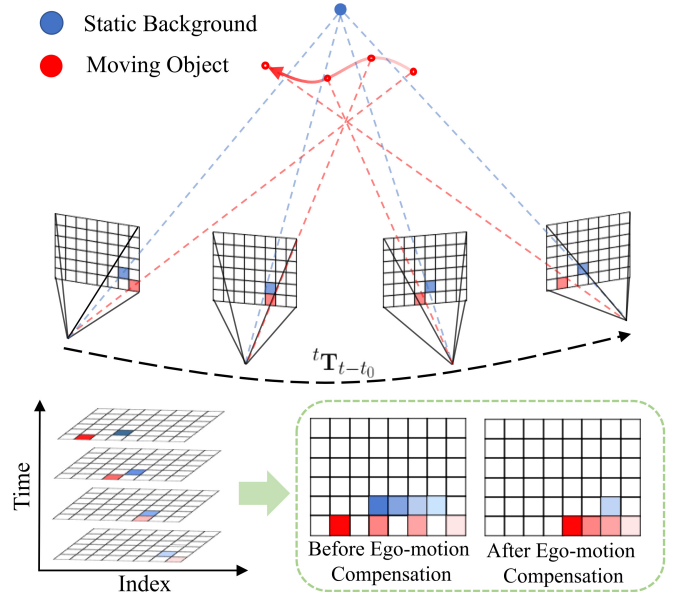

Fig. 3. Working principle of our ego-motion compensation algorithm. Both rotation and translation of the event camera are considered in $^t\mathbf{T}_{t\ t_0}$. Static background (blue) is projected to the same pixel and has a relatively low average timestamp. A moving object (red) cannot be compensated and has a wide range of timestamps. So they can be distinguished in terms of their different temporal distribution.

IV. FAST-MOVING OBJECT DETECTION AND TRACKING

A. Ego-motion Compensation

Events can be triggered either by moving objects or by the ego-motion of the camera. In order to segment objects, events generated by ego-motion (backgrounds) should be filtered out first. In previous works [1][10] [15], algorithms for ego-motion compensation are either computational demanding or not accurate enough. To eliminate this problem, we present a method considering computational efficiency, accuracy, and robustness by fusing depth and IMU data to compensate for rotational and translational ego-motion. The illustration of this section can be inferred in Fig. 3.

This section will introduce our advanced ego-motion compensation method in two steps: rotational and translational. Before projection, we store some events into an event buffer $\mathbf{B}_{t_0}$ in a small time window $\Delta t = 25$ms from timestamp t_0. Then we utilize IMU data to compute the average angular velocity $\bar{\omega}$ and orientation matrix $^W\mathbf{R}_C$ in the world frame during Δt.

1) Rotational Compensation: We apply this compensation step to eliminate events generated by the camera's rotation [1]. After getting the average angular velocity $\bar{\omega}$, we apply the Rodrigues' Rotation Formula [16] to build the rotation matrix $\mathbf{R}_e$ from relative angle $\bar{\omega}(t - t_0)$ at timestamp t. Instead of building this matrix at each timestamp t, we update it every millisecond to decrease computational costs. Then, we use this rotation matrix $\mathbf{R}_e$ to apply a warp field $\phi(\omega, t_0) : (x, y, t) \rightarrow (x_{rot}, y_{rot}, t_0)$ for every event on the image plane. [10] [1]. This event warping process can be

denoted as follows

$$C' = \phi(C)$$
$$= \phi(x, y, t - t_0) \qquad (6)$$
$$= (x_{rot}, y_{rot}, t_0), \quad \forall (x, y, t) \in C,$$

with C' being the motion-compensated events in buffer $\mathbf{B}_{t_0}$. After compensation, we project compensated events to a 2D image plane by the event camera's intrinsic matrix.

2) Translational Compensation: We apply this step to eliminate noise generated by the camera's translation.

Previous compensation methods [1] are limited by lacking depth estimation. Without unreliable depth for each pixel, they cannot compensate for ego-translation, which leads to misdetection when drones are flying fast. We solve this problem by leveraging an onboard depth camera.

We now project events on 2D camera plane (c) to 3D body frame (b) by perspective projection model with homogeneous coordinates [17]

$$d \begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \mathbf{K_E X_E}, \qquad (7)$$

where $\mathbf{X}_E$ is the event's position in the camera frame, (i, j) represents event's coordinate on the image plane, $\mathbf{K_E}$ is the intrinsic matrix. We then project this point into world frame by transform matrix $^W\mathbf{T}_E$ and apply this transitional compensation by multiplying matrix $\mathbf{T}_{t-t_0}$ as re-project it back to the camera frame,

$$\mathbf{X}'_E = {}^W\mathbf{T}_E^{-1}\mathbf{T}_{t-t_0}{}^W\mathbf{T}_E\mathbf{X}_E \qquad (8)$$

where $\mathbf{X}'_E$ is event's compensated position in the camera frame.

This translational compensation matrix $\mathbf{T}_{t-t_0}$ is built from the derivative of position over time $t - t_0$

$$\mathbf{T}_{t-t_0} = \begin{bmatrix} \mathbf{I} & \mathbf{v} \cdot (t - t_0) \\ \mathbf{0}^\top & 1 \end{bmatrix}, \qquad (9)$$

with $\mathbf{v}$ being the velocity using estimation from our odometry. Due to computational cost, we update velocity $\mathbf{v}$ and timestamp t to build this matrix every millisecond.

Results of our advanced ego-motion compensation algorithm can be seen in Fig. 7. After motion compensation, we can filter out the background by simply thresholding on the normalized mean-time graph $\mathcal{N}$ (see Equation 4)

B. Object Detection

1) Dynamic Obstacle Segmentation: After ego-motion compensation, we propose a thresholding method to filter out the background in normalized mean-time image $\mathcal{N}$. Instead of using a fixed threshold, we design an adaptive one considering angular and linear velocity as $\theta(\omega, \upsilon) = \text{mean}(\mathcal{N}) + a\|\omega\| + b\|\upsilon\| + c$, where ω and υ is the magnitude of angular and linear velocity, a, b and c are parameters. We use this threshold $\theta(\omega, \upsilon)$ to classify objects and background. Let us define $\mathcal{S}$ to represent the image after motion segmentation, which is formulated as:

$$\mathcal{S}_{i,j} = \begin{cases} \mathcal{N}_{i,j}, & \mathcal{N}_{i,j} \geq \theta(\omega, \upsilon) \\ 0, & \mathcal{N}_{i,j} \leq \theta(\omega, \upsilon) \end{cases}. \qquad (10)$$

Compared to the previous approach [1], our method can preserve more information of the moving object to decrease the possibility of missed detection.

2) Iterative Gaussian fitting: After dynamic obstacle segmentation, the image is composed of moving objects and background noise. Commonly, the patterns consist of the moving object have a relatively high mean-timestamp. While some pre-processes are still to be done to make the fitting effects better. First, mean filter and morphological operations are used to eliminate salt-and-pepper noise. Next, an element-wise square is ensued to enhance the image contrast further. After all pre-processes above, the **Algorithm** 1 is proposed to extract the moving object. Initially, $\mathcal{S}(\mathbf{C}_P^0, \mathbf{L}^0)$ is the origin ROI, where $\mathbf{C}_P^0 = (x, y)$ denotes the center point of ROI and $\mathbf{L}^0 = (w, h)$ denotes all initial side lengths. In this work, $\mathbf{C}_P^0 = (x, y)$ is pointed as the pattern with the highest mean-timestamp, w and h are distributed as $1/2$ of the image width and height. After that, the optimal $\mathbf{C}_P^*$ and $\mathbf{L}^*$ are computed through an iterative Gaussian fitting process. Finally, the origin ROI $\mathcal{S}(\mathbf{C}_P^0, \mathbf{L}^0)$ is converged to optimal, denoted as $\mathcal{S}(\mathbf{C}_P^*, \mathbf{L}^*)$.

Algorithm 1: Iterative Gaussian Fitting

Input: $\mathcal{S}(\mathbf{C}_P^0, \mathbf{L}^0), K \in \mathbb{Z}_+, \delta_C > 0, \delta_\mathbf{L} > 0$
Output: $\mathcal{S}(\mathbf{C}_P^*, \mathbf{L}^*)$
begin
  while $k < K$ **do**
    $\mathcal{S}(\mathbf{C}_P^k, \mathbf{L}^k) \sim \mathcal{N}(\mu, \sigma^2)$;
    $\mathbf{C}_P^{k+1} \leftarrow \mu$;
    $\mathbf{L}^{k+1} \leftarrow 4\sigma$;
    $J_c \leftarrow J(\mathbf{D}_P^{k+1}, \mathbf{T}^{k+1})$;
    if $\|\mathbf{C}_P^{k+1} - \mathbf{C}_P^k\| < \delta_C$ **and**
      $|\mathbf{L}^{k+1} - \mathbf{L}^k| < \delta_\mathbf{L}$ **then**
      $\lfloor$ **break**
    $k \leftarrow k + 1$;
  $\mathbf{C}_P^* \leftarrow \mathbf{C}_P^k, \mathbf{L}^* \leftarrow \mathbf{L}^k$;
  return $\mathcal{S}(\mathbf{C}_P^*, \mathbf{L}^*)$;

---

3) Moving Region Retrieval: Mostly, the contour of the moving object can be extracted accurately and completely. While in some cases, especially when the scale of the moving object is too small on the image plane, the ROI can fail to converge. Hence, we seek connected components to find the region that is most likely to be the moving object in a fail-converged ROI. After this operation, the moving object is extracted accurately in the majority of cases.

C. Object Trajectory Estimation

As the moving objects are detected, a 3D trajectory can be predicted. However, the two sensors' sample frequency is different, therefore synchronization can be challenging. We are inspired by [2] and propose an optimization-based trajectory estimation and prediction system fusing event and depth observation without synchronous sampling.

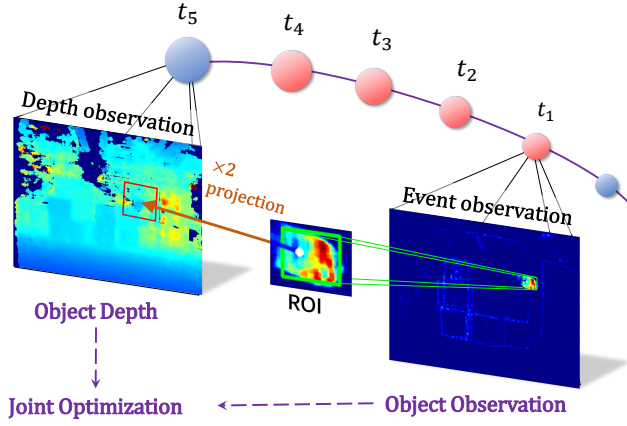An illustration of this section can be seen in Fig. 4.

Fig. 4. The pipeline of our optimization-based trajectory estimator. The pink ball and the cyan ball are the observation of event and depth separately. We use the ROI in the previous detection step (green box) and project it to the depth plane (red). Then we extract depth from the red area and location from the green area. After extracting all information, the trajectory is estimated by joint optimization.
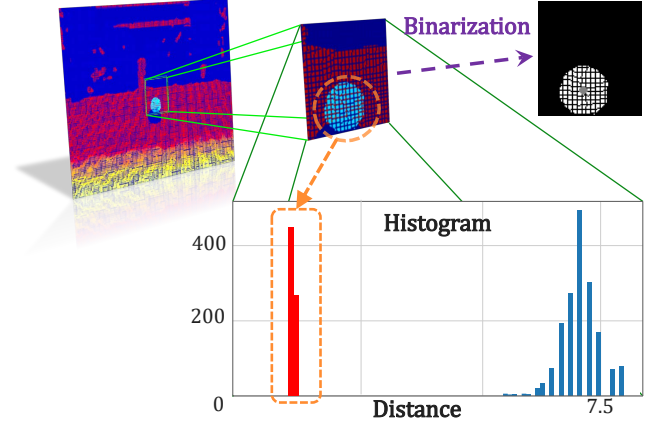

Fig. 5. The process of our object depth segmentation. The minimum distance in histogram is the depth of the object due to the obstacle is the forefront in most case. The mask is created by binarization for computing the variance.

This section introduces our entire system in three parts: (1) 2D object correspondence and tracking. (2) Object segmentation in the depth plane (3) 3D trajectory estimator fusing event and depth observation.

1) 2D Object Correspondence and Tracking: To estimate the object trajectory from detection results, we need to know which object the result corresponds to. When a new detection comes up, the algorithm will judge the time discrepancy and position deviation from the previous detection. After the correspondence is determined, the object is tracked by an Extended Kalman Filter (EKF) [18] with a linear constant acceleration model on the 2D camera plane. The EKF updates the object's central position on the 2D image plane to estimate its 3D position, velocity, and acceleration.

There are two reasons why we use this Extended Kalman Filter. Firstly, due to the event camera's limitation, the appearance of objects on the event plane is not stable and accurate, resulting in the noise of objects' 2D positions under event observation. EKF can filter this noise. Second, potential misdetection may occur, and EKF can predict the object position in this case.

2) Object Segmentation in the Depth Plane: A 3D trajectory is required for motion planning, but the tracking method only provides 2D position. Therefore, the perception of object depth is essential. We solve this by Semi-tight coupling depth segmentation. In other words, we use the detection results of the event camera to assist depth camera segmentation to decrease the computation and processing latency. The principles of our whole process are described below.

First of all, the depth map from the depth camera is registered to the event camera according to the intrinsic and extrinsic matrices of the event and depth cameras.

In practice, because the data frequency of depth and event camera is not equal, object position after projection may have a little bias. We scale the bounding box twice as the ROI area.

After previous steps, the approximate location of the object has been determined. We assume that the most dangerous obstacle is closest to us. So the object can be separated by the nearest peak in the histogram of the depth map. The process is shown as Fig. 5.

To improve the system robustness, we compute the mean and variance of segmented depth pixels. If the variance is too high, it might mean that these pixels belong to the background, which should not be considered. Otherwise, these pixels belong to the object, and we average the value to represent the camera and the object's distance.

3) 3D Trajectory Estimator Fusing Event and Depth: Although the 2D position of the object and the depth have been estimated separately, the event camera is faster than the depth camera, so we can not associate the object depth and the 2D position on the camera plane directly. Inspired by [2], a 3D optimization-based trajectory estimator fusing 2D position and depth is proposed (see Fig. 4). The most significant difference is that we fuse the depth residual into the optimization framework. Before the start, two assumptions should be stated. First, the drone has known the earth's gravity. Second, the object is in free fall, ignoring air resistance. We describe the trajectory as Equation 11. Given the initial 3D position p_{t_0}, velocity v_{t_0}, gravity vector ${}^W g$ and the start time t_0, we can predict the object 3D position at any time t expressed as $p(t)$. From object correspondence, the time t_0 when the object first appeared can be measured. We just need to obtain the initial 3D position p_{t_0} and velocity v_{t_0} to represent the whole trajectory.

$$p(t) = {}^W p_{t_0} + (t - t_0)^W v_{t_0} + \frac{1}{2}{}^W g (t - t_0)^2. \quad (11)$$

The p_{t_0} and v_{t_0} are estimated through the nonlinear optimization by minimizing the depth residual and reprojection error of event observation. Due to depth residual, the number of observations in the same period has increased so that the

convergence speed is faster and the robustness of the system is improved significantly compared to the monocular method.

At time t_k, we detect and track the object in the event camera and the predicted object's position in camera frame can be written as Equation 12. ^Ep_t means position of object in camera frame at time t_k. $^W\mathbf{R}_{E_t}$ and $^W\mathbf{t}_{E_t}$ respectively represent the rotation matrix and translation vector from world to event camera. u_t and v_t are the object position on event camera plane from 2D tracking. The residual r_E can be written as Equation 12 and 13. Meanwhile, in Equation 13, we assume the camera model is pinhole, but this model can be changed according to the actual lens.

$$^Ep_{t_k} = \begin{bmatrix} x_{t_k} \\ y_{t_k} \\ z_{t_k} \end{bmatrix} = {}^W\mathbf{R}_{E_{t_k}}(p(t_k) - {}^W\mathbf{t}_{E_{t_k}}), \quad (12)$$

$$r_E = \begin{bmatrix} \frac{x_{t_k}}{z_{t_k}} - u_{t_k} \\ \frac{y_{t_k}}{z_{t_k}} - v_{t_k} \end{bmatrix}, \quad (13)$$

Similarly, the depth residual r_D is expressed as Equation 14 and 15 with the rotation $^W\mathbf{R}_{D_{t_i}}$ and translation $^W\mathbf{t}_{D_{t_i}}$ from world frame to depth camera frame at time t_i. d_{t_i} is the depth from depth camera observation at time t_i. It should be indicated that the r_E and r_D are independent and various.

$$^Dp_{t_i} = \begin{bmatrix} x_{t_i} \\ y_{t_i} \\ z_{t_i} \end{bmatrix} = {}^W\mathbf{R}_{E_{t_i}}(p(t_i) - {}^W\mathbf{t}_{E_{t_i}}), \quad (14)$$

$$r_D = z_{t_i} - d_{t_i}, \quad (15)$$

$$\min_{^Wp_{t_0}, {}^Wv_{t_0}} \sum_{k=0}^{N} ||r_E||^2 + \sum_{i=0}^{M} ||r_D||^2 \quad (16)$$

Then this problem can formulate nonlinear optimization problem as Equation 16 to obtain the trajectory parameter $(^Wp_{t_0}, {}^Wv_{t_0})$. For better robustness, we use the Huber loss.

V. EXPERIMENT AND EVALUATION

A. Implementation Details

We present our real-world experiment (see section V-D) on a modified flight platform, carrying an iniVation DVXplorer dynamic vision sensor and an Intel Realsense D435i depth

DJI Manifold2-C

Intel Realsense D435i

DJI Frame Wheel F450

CUAV Nora Autopilot Flight Controller
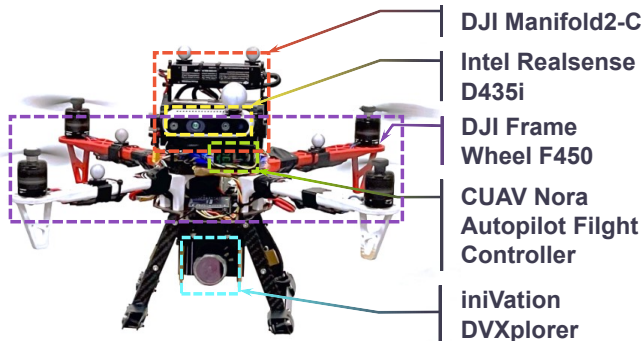
iniVation DVXplorer

Fig. 6.   Overview of our UAV system

camera. A DJI Manifold2-C computer running Ubuntu 16.04 is mounted in our UAV for computational supports. We use a CUAV Nora Autopilot Flight Controller running the PX4 flight stack. To alleviate disturbance from the motion capture system's infrared light on the dynamic vision sensor, we add an infrared filter on the lens surface of the DVXplorer camera. The overall weight (including LiPo battery and propellers) is 1.99 kg, with dimensions being $570 \times 570 \times 270$ mm. An overview of our flight platform can be seen in Fig. 6.

B. Evaluation of Ego-motion Compensation

To demonstrate the robustness of our method, we put our system into high dynamic scenarios, where the UAV flies at a speed of over 5 m/s. Three algorithms are applied in two scenes, one with no moving object and another has an object that moves over 10 m/s (see TABLE I). To ensure efficiency and accuracy, we hope the process has lower time consumption. Moreover, the contrast between the moving object and the background is deemed to be as high as possible, which is critical for detection algorithms. We call this the Relative Contrast. To derive this, we define the manually marked bounding box of the moving object in a motion-compensated image as $\mathcal{M}$ (marked as green in Fig. 7(a) on the RGB frame). We depict the rest part of the image as $\mathcal{B}$. Then, relative contrast η is defined as:

$$\eta = \frac{\max\limits_{(i,j)\in\mathcal{M}} \mathcal{M}_{i,j} - \max\limits_{(i,j)\in\mathcal{B}} \mathcal{B}_{i,j}}{\max\limits_{(i,j)\in\mathcal{M}} \mathcal{M}_{i,j}}. \quad (17)$$

Notice, since the sensors are imperfect, noise is introduced. It is meaningless to compute the relative contrast on an image that has much noise because noise often has the timestamp from oldest to newest, so they are more likely to be selected for computation of relative contrast instead of the moving object or background. Therefore, we apply unified denoising for images after ego-motion compensation by all three methods before computing the relative contrast. Besides, the relative contrast can only be computed in Scene 2 because scene 1 does not have moving objects.

Usually, these indicators cannot be met at the same time, so the trade-off between performance and efficiency is indispensable. In this work, we sacrifice a little efficiency under the promise of real-time. Table I lies the results of the comparison of our method against [1] and [10]. The table indicates that our method largely outperforms [10] while lower than [1] in several million-seconds. At the same time, the mean value and variance of our output image are closer to the optimization-based method [10] than [1]'s. Moreover, our method has the highest contrast between the moving object and the background, which provides convenience for object detection.

C. Evaluation of Trajectory Estimation

To validate the accuracy of estimated trajectories, we compare our fusing method with the monocular method[2] in the same scenarios. Our ground truth is provided by a

TABLE I
COMPARISON OF EGO-MOTION COMPENSATION ALGORITHMS

Experiment	Algorithm	Time (ms)			Relative Contrast (%)			Real-time	Dynamic-flight
		min	avg	max	min	avg	max		
Scene 1	BetterFlow [10]	1173.2	10620.4	32461.9	-	-	-	No	Yes
	Falanga [1]	**2.7**	**7.1**	**20.6**	-	-	-	Yes	No
	ours	4.1	12.9	22.4	-	-	-	**Yes**	**Yes**
Scene 2	BetterFlow [10]	554.2	16587.6	110559.0	17.7	27.8	36.1	No	Yes
	Falanga [1]	**1.4**	**4.6**	**16.4**	-1.2	1.7	3.9	Yes	No
	ours	4.3	8.9	20.7	**22.4**	**32.9**	**51.4**	**Yes**	**Yes**

Vicon motion capture system. We perform two estimation algorithms in two different scenarios, one with the drone flying fast forward and the other with the drone swinging forward. The drone flies at 2 m/s, and a ball is thrown at about 12 m/s from one side to another in both scenes. The modules of detection and data association are fully consistent. Due to the fast motion of the ball, there are seven detections on the event camera and three segmentations of the depth map in 0.18 seconds in the forward scenario. The swinging scenario lasts for 0.16 seconds with six detections on event and three on depth.

Fig. 8 shows a comparison of fusing two cameras versus a monocular event camera which is configured as [2]. Fig. 8 states that the result of fusing two sensors is significantly superior to only monocular event intuitively. In both scenarios, the trajectories estimated by the monocular method are opposite in the x-direction. It is mainly due to fewer detection times and lack of depth truth. We compute the APE(Absolute Pose Error) of the estimated trajectory with the reference. The detailed result is shown in TABLE II. This comparison demonstrates that the accuracy of our method is

TABLE II
COMPARISON OF TRAJECTORY ESTIMATION METHOD

Experiment	Method	APE (m)			
		Mean	Min	Max	RMSE
Fast Forward	Mono[2]	0.725	**0.152**	1.218	0.784
	Fusion	**0.194**	0.169	**0.285**	**0.201**
Swing	Mono[2]	0.956	0.345	0.957	0.717
	Fusion	**0.424**	**0.248**	**0.562**	**0.436**

much higher in these fast scenarios.

D. Real-world Experiment

We present several throwing-ball experiments with onboard sensors both indoor and outdoor, with larger and smaller balls, bright (240 ∼ 1100 lux) and dim (8 ∼ 10 lux), swinging and moving environments. The main goal of these experiments is to validate our detection, tracking, and 3D trajectory estimation system onboard highlighted in the dynamic object avoidance scene.

One experiment is to throw a ball of unknown size at a

(a) RGB   (b) BetterFlow[10]   (c) Falanga[1]   (d) ours

(e) origin   (f) BetterFlow[10] after denoising   (g) Falanga[1] after denoising   (h) ours after denoising
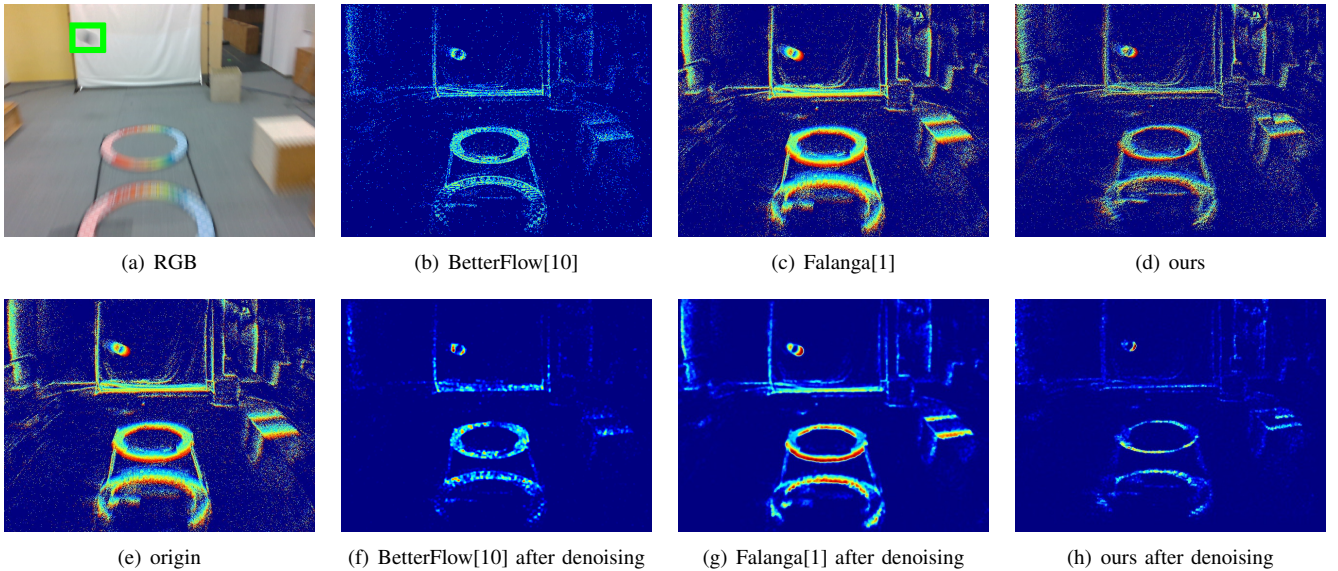
Fig. 7. Comparison of our ego-motion compensation method with [10] and [1] (Section IV-A). Fig. 7(a) is RGB image, which contains a flying ball (green) and two colorful banners; 7(e) is the original mean-time graph $\mathcal{T}$; 7(b) is the mean-time graph $\mathcal{T}$ generated by [10], 7(c) is the performance using algorithm from [1]; 7(d) is our ego-motion method fusing IMU and depth data to compensate rotational and translational motion. 7(f) 7(g) 7(h) are all three compensated images after unified denosing.

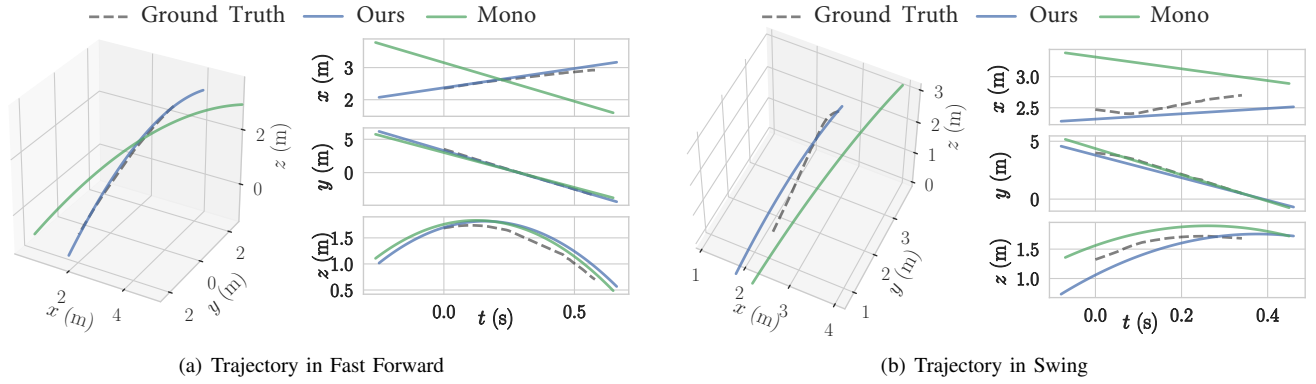(a) Trajectory in Fast Forward

(b) Trajectory in Swing

Fig. 8.   The estimated trajectories of our fusing method with the monocular method in [2]. The dotted line is the ball position obtained by motion capture, the green line is the method in [2] and the blue line is our fusing method.

hovering UAV, which would move upward after detecting the ball to avoid the collision. A ball with a diameter of 21cm was thrown at a distance ranging from 8 to 10 meters at speeds from 7.0 to 12.0 m/s like Fig. 1. In this experiment, background events are triggered by in-situ vibration and rotation. We did this experiment in several scenes to evaluate the performance under different ambient illumination levels.

Another experiment is to dodge a throwing ball while the UAV is flying forward. The ball was thrown at the same position and speed compared to the last experiment. Different from that one, plenty of background events were triggered by the UAV's translational motion, making the moving ball harder to detect. Eventually, our system successfully tackles this challenge with remarkable performance (please refer to our attached video).

VI. CONCLUSION

In this paper, we present a novel perception system for solving dynamic object avoidance problems. It achieves a computational-friendly while accurate motion compensation for event-based object detection. It also presents a robust 3D trajectory estimator leveraging both event and depth data. The system has been tested in real-world experiments to prove its advantages.

Nevertheless, there is still room for improvement in some aspects. Integrating avoidance algorithm based on motion planning with our perception system is one of the most promising improvements. In this way, a carefully generated trajectory such as [19] [20] could consider static and dynamic scenes, avoiding performance and flight smoothness.

REFERENCES

[1] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, p. eaaz9712, 2020.

[2] K. Su and S. Shen, "Catching a Flying Ball with a Vision-Based Quadrotor," in *2016 International Symposium on Experimental Robotics*, ser. Springer Proceedings in Advanced Robotics, D. Kuli, Y. Nakamura, O. Khatib, and G. Venture, Eds. Springer International Publishing, 2017, vol. 1, pp. 550–562.

[3] G. Gallego, T. Delbruck, G. M. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, pp. 1–1, 2020.

[4] T. Brosch, S. Tschechne, and H. Neumann, "On event-based optical flow detection," *Frontiers in Neuroscience*, vol. 9, 2015.

[5] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom. (ICRA)*, 2017, pp. 4465–4470.

[6] E. Piatkowska, A. N. Belbachir, S. Schraml, and M. Gelautz, "Spatiotemporal multiple persons tracking using dynamic vision sensor," in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2012, pp. 35–40.

[7] X. Lagorce, C. Meyer, S. 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, Aug. 2015.

[8] H. Li and L. Shi, "Robust event-based object tracking combining correlation filter and cnn representation," *Frontiers in Neurorobotics*, vol. 13, p. 82, 2019.

[9] C. Brandli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck, "ELiSeD An event-based line segment detector," in *2016 Second International Conference on Event-Based Control, Communication, and Signal Processing (EBCCSP)*. IEEE, 2016, pp. 1–7.

[10] A. Mitrokhin, C. Fermuller, C. Parameshwara, and Y. Aloimonos, "Event-Based Moving Object Detection and Tracking," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, 2018, pp. 1–9.

[11] G. Gallego, H. Rebecq, and D. Scaramuzza, "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation," in *Proc. of the IEEE Intl. Conf. on Pattern Recognition (CVPR)*. IEEE, 2018, pp. 3867–3876.

[12] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, "Semi-dense 3d reconstruction with a stereo event camera," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 235–251.

[13] Y. Zhou, G. Gallego, and S. Shen. Event-based Stereo Visual Odometry. [Online]. Available: http://arxiv.org/abs/2007.15548

[14] Y. Zhou, G. Gallego, X. Lu, S. Liu, and S. Shen. (2020) Event-based motion segmentation with spatio-temporal graph cuts. [Online]. Available: https://arxiv.org/abs/2012.08730

[15] A. Mitrokhin, C. Ye, C. Fermller, Y. Aloimonos, and T. Delbruck, "EV-IMO: Motion Segmentation Dataset and Learning Pipeline for Event Cameras," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst. (IROS)*, 2019, pp. 6105–6112.

[16] W. Stanley, "Quaternion from rotation matrix," *Journal of Guidance and Control*, vol. 1, no. 3, pp. 223–224, 1978.

[17] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge: Cambridge University Press, 2004.

[18] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[19] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "Ego-planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.

[20] Z. Wang, X. Zhou, C. Xu, and F. Gao. Geometrically Constrained Trajectory Optimization for Multicopters. [Online]. Available: http://arxiv.org/abs/2103.00190