Taming Event Cameras with Bio-Inspired Architecture and Algorithm: A Case for Drone Obstacle Avoidance

Danyang Li*, *Student Member, IEEE,* Jingao Xu*, *Member, IEEE,* Zheng Yang†, *Fellow, IEEE,* Yishujie Zhao, Hao Cao, *Student Member, IEEE,* Yunhao Liu, *Fellow, IEEE,* Longfei Shangguan, *Member, IEEE*

Abstract—Fast and accurate obstacle avoidance is crucial to drone safety. Yet existing on-board sensor modules such as frame cameras and radars are ill-suited for doing so due to their low temporal resolution or limited field of view. This paper presents **BioDrone**, a new design paradigm for drone obstacle avoidance using stereo event cameras. At the heart of BioDrone are three simple yet effective system designs inspired by the mammalian visual system, namely, a chiasm-inspired event filtering, a lateral geniculate nucleus (LGN)-inspired event matching, and a dorsal stream-inspired obstacle tracking. We implement BioDrone on FPGA through software-hardware co-design and deploy it on an industrial drone. In comparative experiments against two state-of-the-art event-based systems, BioDrone consistently achieves an obstacle detection rate of >90%, and an obstacle tracking error of <5.8cm across all flight modes with an end-to-end latency of <6.4ms, outperforming both baselines by over 44%.

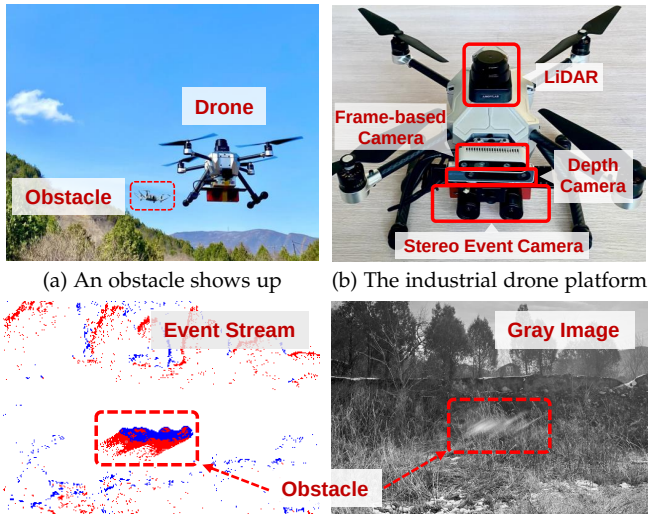
Index Terms—Mobile Computing; Drone-based Applications; Obstacle Avoidance; Event Camera; Bio-inspired Design

1 Introduction

Drones are among the most disruptive inventions in the past few years, spawning many novel applications including aerial imaging [1], [2], [3], last-mile delivery [4], [5], [6], sky networking [7], [8], and industrial inspection [9], [10], [11]. Despite their huge market value, safety remains a crucial challenge for drones, particularly for those high-speed drones in industrial and urban applications. For instance, DJI's industrial drones cruise at up to $25m/s$ [12], and the relative speed between two Amazon delivery drones can reach $30m/s$ [13]. Drone collisions with obstacles (e.g., birds [14], drones [15]) will not only cause financial loss but also threaten human safety [16], [17], which sets a strong barrier for drone adoption.

Fast and accurate obstacle detection and localization plays a key role in drone obstacle avoidance – the lower the detection latency, the more time the drone could take to react; and a higher localization accuracy increases the likelihood the drone can dodge them. Existing solutions primarily rely on frame-based cameras [18], [19] and radars [20], [21]. However, the low spatial-temporal sampling resolution of these on-board sensors makes it challenging for drones to perceive obstacles timely or localize them accurately.

For instance, the sampling interval of a typical frame-based camera varies from $20ms$ to $50ms$, during which an obstacle can move up to $40cm$ (given a $20m/s$ relative speed). As a result, we are expected to see severe motion blurring on each image (Fig.1c). Such motion blurring will fail the vision algorithms, impairing both obstacle detection and localization accuracy. The radar-based solutions, on the other hand, suffer from high miss detection rates due to their limited field of view (FoV) [22], [23].



(a) An obstacle shows up (b) The industrial drone platform

(c) A comparison of captured event stream and ordinary gray image (without versus with obvious motion blur)

Fig. 1. Snapshot of an obstacle avoidance maneuver.

Drone obstacle avoidance with event cameras. Event cameras, inspired by biological vision systems, asynchronously report pixel-level intensity changes. Endowed with microsecond resolution, event cameras are able to capture high-speed motions without blurring (Fig.1c). Hence event cameras are envisioned to be an ideal solution to challenging vision tasks such as high-speed motion tracking [24], [25], and simultaneous localization and mapping (SLAM) [26].

To better understand the potential of event cameras for obstacle avoidance, we reimplement event-based systems

- *A preliminary version of this article appeared in International Conference on Mobile Computing and Networking (ACM MobiCom 2023)*
- *Danyang Li, Jingao Xu, Zheng Yang, Yishujie Zhao, Hao Cao, and Yunhao Liu are with the School of Software and BNRist, Tsinghua University, Beijing, China, 100084. E-mail: {lidanyang1919, xujingao13, hmilyyz, chillpill995, nyz1500, yunhaoliu}@gmail.com*
- *Longfei Shangguan is with the Department of Computer Science, University of Pittsburgh. E-mail: longfei@pitt.edu*
- **Danyang Li and Jingao Xu are co-primary authors*
- *†Zheng Yang is the corresponding author*

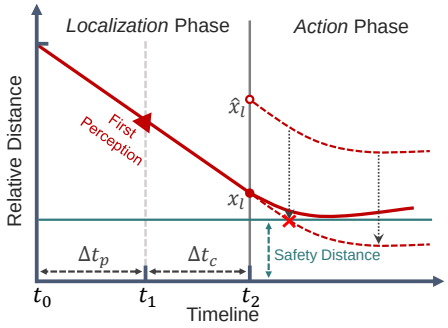


Fig. 2. Illustration of Obstacle Avoidance

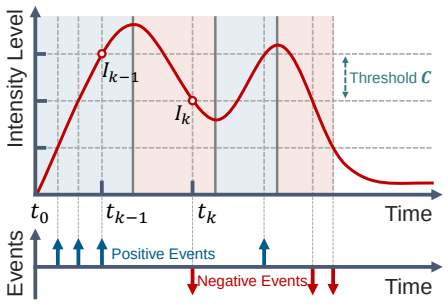


Fig. 3. Principle of Event Camera

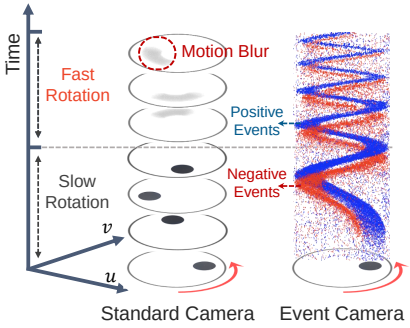


Fig. 4. Comparison of Different Camera

[23], [27], [28], [29], [30], [31], [32] and evaluated their performance. Our benchmark studies (§2.2) reveal that these systems face fundamental challenges in high-speed drone flight scenarios, as detailed below.

• **Event burst impairs drone obstacle detection.** Event cameras are hyper-sensitive to environmental change. For instance, a slight change in lighting can lead to a remarkable change in pixel-wise intensity, resulting in hundreds of event reports. In practice, the scene in the camera's view changes rapidly due to drone movement; thus we will see an event burst where thousands of events are reported within a short time and obstacle-triggered events are easily buried by massive numbers of environment-triggered events.

• **Event sticking delays drone obstacle localization.** Conventional vision algorithms are designed for frame-based cameras and cannot be directly applied to event streams for obstacle localization because the output of an event camera is not an image but a stream of asynchronous events. To address this issue, the current practice periodically sticks a lot of scattered events into a compact image and applies image-based algorithms (e.g., stereo triangulation [33] or deep neural networks [27], [28], [30], [31]) that are both computationally demanding. Repeating these operations would cause significant delays in obstacle localization.

Although existing solutions (e.g., Baseline-I [23]) achieve high obstacle *detection* accuracy by using a monocular event camera. The obstacle *localization* performance, however, drops significantly in high-speed scenarios (i.e., $20m/s$) due to the increasing task complexity and growing data volume.

Given that the event camera is a kind of bio-inspired vision sensor, we ask a question: Could we tackle the above challenges by studying how animals process binocular visual signals for efficient obstacle localization? To answer this question, we resort to bionics and take a comprehensive study (§3.1) on (i) how binocular visual signals are transmitted from the retina to the visual cortex in the mammalian visual system; and (ii) how they are rapidly filtered, matched, and spatio-temporal corrected through the visual pathway.

Our Work. In this paper, we leverage the **Bio**logical lessons learned from mammalian visual system and propose BioDrone, a **Drone**-oriented obstacle avoidance system. BioDrone features three key designs to fully unleash binocular event cameras' potential for obstacle localization. It is implemented on FPGA by software-hardware co-design, incorporating on-chip intelligence [34], [35], as detailed below.

• On system architecture front, we imitate how mammal's *visual pathway* processes binocular visual signals and propose a visual-pathway-inspired signal processing pipeline for binocular event streams. Unlike the current practice where event streams are processed separately and not fused until the final triangulation stage, BioDrone fuses binocular event streams at an early stage, enabling the subsequent event filtering, matching, and localization modules to take full advantage of binocular information (§3.3).

• On system algorithm front, we first introduce a Chiasm-inspired Event Filtering (CEF) algorithm to quickly filter out environment-triggered events from the massive amount of events with a very low false positive rate (§4.1). We then propose a Lateral Geniculate Nucleus (LGN)-inspired Event Matching (LEM) algorithm to determine the obstacle spatial location using a unique spatio-temporal event representation (§4.2). Moreover, we implement a Dorsal stream-inspired Obstacle Tracking (DOT) algorithm, which adeptly balances historical states with real-time observations to optimize obstacle trajectory and predicts its location (§4.3).

• On system implementation front, we implement BioDrone on a commercial Xilinx Zynq-7020 [36] chip. We design exclusive logic circuits, on FPGA, to parallelize the pixel-wise event processing, expediting the software stack (§5).

We deploy BioDrone on a drone testbed and further integrate it into ArduPilot [37], a widely-used open-source drone flight controller. We conduct extensive experiments with various types of obstacles and in different flying speed settings both indoors and outdoors. We compare the end-to-end obstacle localization accuracy and latency of BioDrone with two state-of-the-art (SOTA) event camera-based drone obstacle avoidance systems Baseline-I [23] (Science Robotics'20) and Baseline-II [29] (IROS'18). Evaluation results show that BioDrone achieves >90% obstacle detection rate across all fight modes, outperforming both baselines by >10%. BioDrone further achieves <5.8cm tracking error with <6.4ms latency, outperforming baselines by >44%.

In summary, this paper makes following contributions.

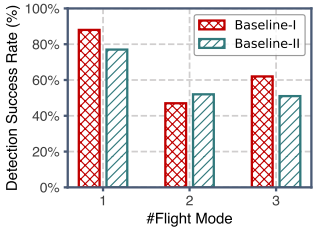
(1) We systematically study both the conventional sensor- and event camera-based drone obstacle avoidance systems, and reveal the fundamental limitations of these solutions.

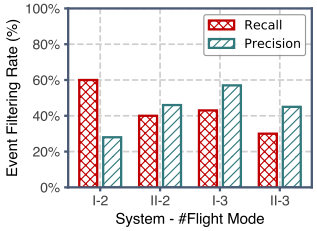
(2) We design various bio-inspired components in BioDrone to unleash the potential of event cameras for obstacle avoidance, including a human visual pathway-inspired event processing architecture, a chiasm-inspired event filtering module, a LGN-inspired event matching mechanism, and a dorsal stream-inspired obstacle tracking algorithm.

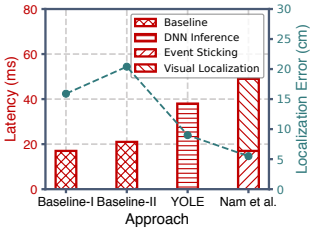
(3) We fully implement BioDrone through software-hardware co-design and deploy it on an industrial drone, conducting a head-to-head comparison with two SOTA systems. The evaluation results show the feasibility and efficiency of BioDrone to realize fast obstacle avoidance for high-speed drones.

Flight Mode	Event Generating Speed (e/ms)	
	Background	Obstacle
#1 (Normal Speed) $2m/s \le v_t \le 4m/s$ $5^\circ/s \le \omega \le 15^\circ/s$	103 (34.5%)	195 (**65.5%**)
#2 (Rapid Translation) $18m/s \le v_t \le 25m/s$ $5^\circ/s \le \omega \le 15^\circ/s$	675 (**71.4%**)	270 (28.6%)
#3 (Rapid Rotation) $2m/s \le v_t \le 4m/s$ $75^\circ/s \le \omega \le 95^\circ/s$	1091 (**75.9%**)	346 (24.1%)

Event Generating Speed Comparison







(a) Event Generating Speed (b) Obstacle Detection Rate (c) Event Filtering Rate (d) Localization Accuracy-Latency

Fig. 5. Performance of existing event-based solutions at different translation and rotation speed settings.

2 Motivation

2.1 Drone Obstacle Avoidance Primer

As illustrated in Fig.2, obstacle avoidance consists of *localization* and *action* two phases. During the *localization* phase, suppose an obstacle shows up abruptly at t_0 and is perceived by a drone at t_1 after a perception delay Δt_p. Upon detecting the obstacle, the drone takes Δt_c to localize the obstacle. Noting that the drone still follows its planned trajectory to move before localizing the obstacle at t_2. Afterward, the on-board flight controller changes the drone's trajectory to dodge the obstacle (i.e., keep a safe distance from it) in the *action* phase.

Both the localization delay ($\Delta t_l = \Delta t_p + \Delta t_c$) and localization error ($\Delta x = \hat{x}_l - x_l$) are crucial to drone obstacle avoidance. A long delay Δt_l leaves the drone very short time to react, and a large localization error Δx misleads the flight controller to execute a wrong obstacle avoidance maneuver. For instance, as depicted by the red dotted line in Fig.2, evasion commands issued by the flight controller based on the inaccurate localization result $\hat{x}_l$ cannot make the drone dodge the obstacle (with its real location at x_l) successfully. To ensure the success of collision avoidance in high-speed scenarios, it is crucial to minimize both the localization delay Δt_l and action bias Δx.

2.2 Event Camera for Obstacle Avoidance

Event cameras are bio-inspired sensors that work differently from frame-based cameras. Instead of capturing images at a fixed rate, an event camera measures per-pixel brightness changes asynchronously, resulting in a stream of events at *microsecond* resolution [26].

Principle of Event Camera. Event cameras feature intelligent pixels, akin to photoreceptor cells in retinas, that independently trigger events. When a pixel detects a change in intensity, it generates an event $e_k = (\boldsymbol{x}_k, t_k, p_k)$, which records the trigger time t_k, the pixel's spatial location $\boldsymbol{x}_k = (u, v)$, and the polarity p_k, indicating whether the intensity change is towards brighter or darker. Specifically, as shown in Fig.3, let t_{k-1} be the last time an event was triggered at pixel $\boldsymbol{x}_k$, with I_{k-1} as the intensity at that time. A new event is triggered at time t_k when the intensity difference $||I_k - I_{k-1}||$ exceeds a threshold C. Fig.4 compares event cameras with conventional cameras. Unlike conventional cameras, which capture frames at a fixed rate and suffer from motion blur, event cameras continuously output brightness changes as a stream of events in space-time.

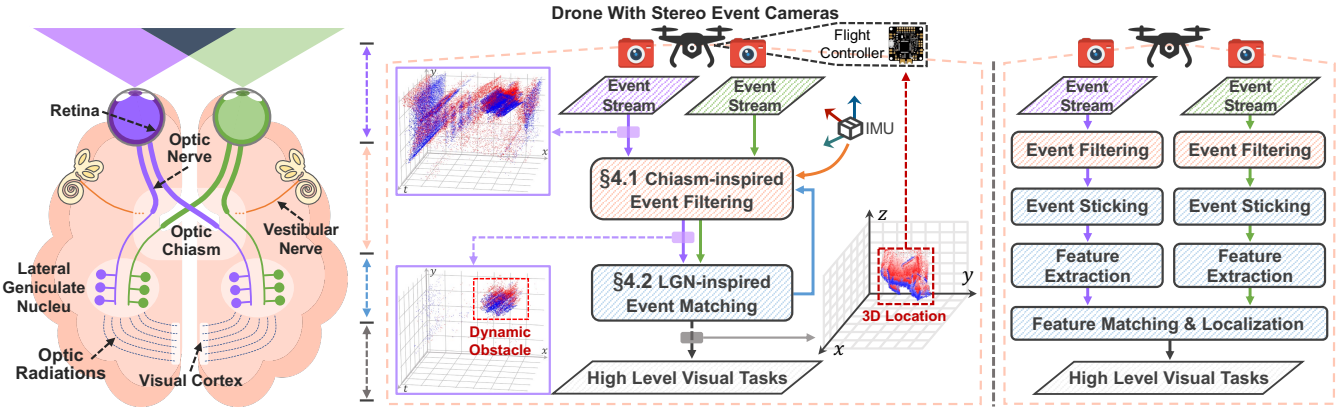
Opportunities and challenges. Compared with frame-based camera and radar, event camera is a natural choice for obstacle avoidance due to the following advantages: (i) their high temporal resolution allows an extremely low perception delay (i.e., microsecond-level Δt_p), enabling motion blur-free measurements as opposed to frame-based cameras; and (ii) the output of event camera is sparse compared to an entire frame captured by a conventional camera, resulting in lower processing delay (lower Δt_c). Exploiting these properties, some previous works investigated the use of stereo event cameras to track obstacles for drones [23], [32]. However, the high speed (i.e., both translation and rotation speed) of drones brings new issues that challenge the obstacle avoidance performance:

- **C1: Event burst impairs drone obstacle detection.** Events captured by an event camera can be classified into two categories: environment-triggered and obstacle-triggered events. The former is generated due to the ego-motion of the event camera, while the latter is caused by the appearance of obstacles. To detect and further localize an obstacle, a system needs to identify obstacle events from massive events. To this end, existing solutions apply IMU-based ego-motion compensation algorithms to filter out environment-triggered events [23], [26], [29]. However, as shown in Fig.5a, the number of events generated per millisecond surged from around 300 to 1,500 and the new additions are mainly environment events. Such a burst of environmental events overwhelm obstacle events and degrade existing algorithms' performance.

To validate the above analysis, we conduct an obstacle detection experiment under different flight modes. As shown in Fig.5b, the detection rate of two SOTA solutions, Baseline-I [23] and II [29], drops to $< 60\%$. To better understand the reasons for failure cases, we further examine the event filtering performance of these two baselines in two high-speed flight modes (i.e., mode 2 and mode 3). We observe that both systems achieve a very low event filtering rate (recall and precision $< 60\%$ in Fig.5c), which confirms our analysis.

- **C2: Event sticking delays drone obstacle localization.** Once the obstacle is detected, the drone has to localize it in 3D space. Typically, localization is more time-consuming than detection due to the additional operations involved. For instance, Baseline-I requires binocular parallax optimization, matching, and triangulation after detection, which is more computationally intensive as outlined in [23].

Moreover, conventional vision algorithms (e.g., stereo triangulation [33]) or DNNs cannot be directly applied as the output of an event camera is not fix-rate frames but a stream of asynchronous events. To solve this issue, the current practice proposes to (i) stick all generated events within a time window (e.g., $< 10ms$) into an image and then apply image-based algorithms (Fig.6c); or (ii) design event data-oriented DNNs (e.g., spiking neural networks [39]) for object localization. However, as depicted in Fig.5d, although the localization accuracy is boosted, either the sticking operations, the stereo visual algorithms, or DNN



(a) (b) (c)

Fig. 6. System architecture comparison. (a) Human binocular visual pathway. (b) BioDrone's architecture inspired by (a). (c) System architecture of conventional event-based systems [23], [29], [38], where binocular event streams are processed separately and follow traditional visual localization workflow (i.e., from feature extraction to matching and then stereo triangulation).

inference introduces significant delays, leaving the drone no time to react.

In summary, although event cameras hold great potential for delay-sensitive tasks such as drone obstacle avoidance, there still lack effective algorithms and system support to fully unleash their potential.

3 Bio-Inspired Architecture

Our system architecture and algorithms are inspired by the *biological visual pathway*. In this section, we first introduce the biological visual pathway in mammalian visual system and describe how visual information is filtered, processed, and transmitted from retina to brain through the pathway. We then present the lessons learned and explain how we leverage these insights to design BioDrone.

3.1 Biological Visual Pathway

As illustrated in Fig.6a, light entering eyes is refracted by the cornea and lens and then simulates photoreceptor cells on the *retina* to produce visual signals. The *optic nerves* carrying those visual signals from both eyes cross at the *optic chiasm*, which localizes at the base of the hypothalamus of the brain [40]. Additionally, the *vestibular nerves* that transmit human motion information, interact with the optic nerves at the optic chiasm and select which necessary visual signals will be further carried forward to the thalamus for subsequent processing [41]. Afterwards, the filtered visual signals enter the *lateral geniculate nucleus* (LGN) are re-organized and spatio-temporally correlated to achieve a 3D representation of environment [42]. Subsequently, these integrated visual representations are transmitted to the *visual cortex* via the *optic radiations*. From there, the *dorsal stream* extends to the *posterior parietal cortex*, traditionally identified as the "where" pathway due to its role in processing spatial attributes of objects [43].

3.2 Bio-lessons

We've learned two biological lessons from visual pathway:

• **L1: Early integration of binocular visual signals.** Binocular visual signals are integrated at an early stage (i.e., at optic chiasm instead of brain). This allows visual signal filtering and matching to take full advantage of the binocular information. In contrast, current practice [23], [29] process, filter, and extract visual features from each event stream independently, as illustrated in Fig.6c.

• **L2: Fast processing of low-level visual tasks.** Low-level visual tasks, such as object detection and localization, are swiftly executed during the signal transmission through the visual pathway. This involves the binocular visual signals being filtered at the optic chiasm, matched at the LGN, and undergoing motion analysis in the dorsal stream. In contrast, the visual cortex is dedicated to more complex, high-level visual tasks like object recognition or segmentation.

3.3 Overview of BioDrone

BioDrone shares a similar architecture with the biological visual pathway to unleash the potential of event cameras, as shown in Fig.6b. We explain the functional units below.

• From the architecture perspective, following lesson-L1, BioDrone features a visual-pathway-inspired signal processing pipeline, fusing binocular event streams at an early stage, which allows the obstacle detection and localization tasks to combine and fully leverage the binocular event information.

• From the algorithm perspective, following lesson-L2, BioDrone attempts to mimic how *optical chiasm*, *LGN*, and *dorsal stream* process binocular visual signals and designs three heuristic algorithms. Specifically, BioDrone proposes a *Chiasm-inspired Event Filtering* (CEF) mechanism for event filtering and obstacle detection, an *LGN-inspired Event Matching* (LEM) module to localize obstacles from the integrated event stream, and a *Dorsal stream-inspired Obstacle Tracking* (DOT) algorithm aimed at optimizing and predicting the movement of obstacles.

Finally, the resulting obstacle trajectory and predicted location from DOT will be (i) utilized to direct the flight controller in executing appropriate evasive maneuvers; (ii) fed back to CEF to provide a priori information for subsequent event filtering.

4 Bio-Inspired Algorithm Design

In this section, we describe three bio-inspired algorithms for event filtering (§4.1), event matching (§4.2), and obstacle tracking (§4.3).

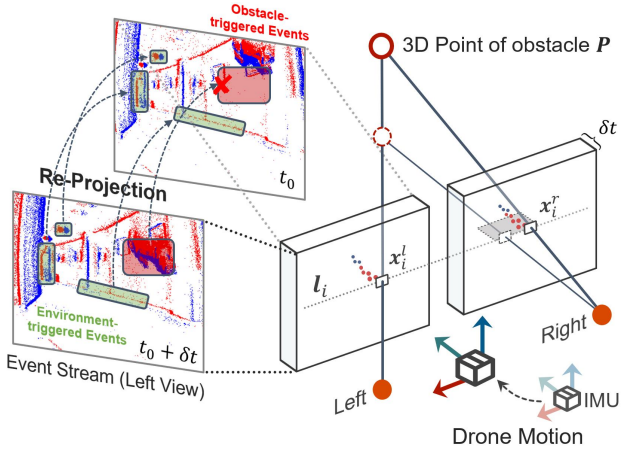


Fig. 7. Illustration of the chiasm-inspired event filtering scheme. Left: the distinction between environment- and obstacle-events under ego-motion instruction; Right: the binocular constraint that an obstacle event should satisfy.

4.1 Chiasm-Inspired Event Filtering

Optic nerves carrying visual signals from eyes and vestibular nerves carrying motion signals from cochleas cross at optic chiasm. Like a busy intersection, optic chiasm is the rendezvous point where binocular visual information gets fused and filtered under the guidance of proprioceptive motion information. Typically, about 1,200,000 photoreceptors on human retina generate visual signals per second, yet merely around 1,700 of them would pass through optic chiasm [44].

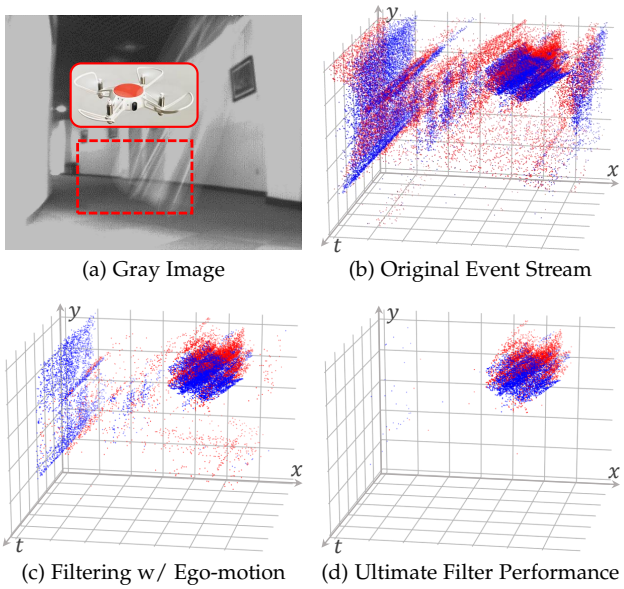
Motivated by the chiasm's ultra-efficient signal filtering performance, we design a chiasm-inspired event filter that leverages the drone's IMU perception data (simulating the vestibular motion signals) to pick up obstacle events in binocular event streams. The insight behind this mechanism lies in two-fold: (i) IMU could be leveraged to infer the ego-motion of event cameras, which provides a priori knowledge to cull environment-triggered events. Just like in our daily life, when your head is turning right, your righter visual field becomes clear while the lefter blurs, and vice-versa; and (ii) the collaborative use of binocular event streams would further improve the filtering performance as the spatial relationship (i.e., pose transformation) between the stereo event cameras provides an additional constraint. For instance, a single eye is less sensitive to the depth change of a moving object compared to two eyes.

4.1.1 *Event Filtering Based On Ego-motion Instruction*

We begin by explaining how events are filtered using the motion information from IMU sensors. As shown in Fig.7, consider a batch of events $\mathcal{E}$ and IMU data $\mathcal{I}$ collected within a short time window $[t_0, t_0 + \delta t]$. For any event $e_i = (\boldsymbol{x}_i, t_i, p_i)$ occurring within this window, we estimate its past location $\hat{\boldsymbol{x}}_0$ at time t_0 based on the drone's motion. There are two scenarios:

• *Environment-triggered event.* If e_i is caused by a stationary environment feature, its back-projected pixel location $\hat{\boldsymbol{x}}_0$ should align with its location $\boldsymbol{x}_0$ at t_0, as the apparent change is solely due to the drone's motion.

• *Obstacle-triggered event.* On the other hand, if e_i is triggered by a moving obstacle, the back-projected location $\hat{\boldsymbol{x}}_0$ will not match $\boldsymbol{x}_0$, since the projection only accounts for the drone's movement and not the obstacle's motion.



(a) Gray Image (b) Original Event Stream

(c) Filtering w/ Ego-motion (d) Ultimate Filter Performance

Fig. 8. Step-by-step event filtering performance.

Modeling. To formalize this process, we define the key variables illustrated in Fig.7. In short time periods, camera rotation typically generates more events than translation [23]. Therefore, we primarily focus on compensating for ego-rotation when filtering events. Each event e_i is warped to the image plane at time t_0:

$$\hat{\boldsymbol{x}}_0 = \boldsymbol{K}\boldsymbol{R}_i\boldsymbol{K}^{-1}\boldsymbol{x}_i, \tag{1}$$

where $\boldsymbol{K}$ is the camera's intrinsic matrix, and $\boldsymbol{R}_i$ represents the rotation matrix describing the pose transformation from time t_i to t_0, provided directly by the IMU. While localization algorithms (e.g., Kalman Filter) can offer more accurate motion estimates, we opted for raw IMU data to maintain real-time performance and minimize accumulated drift error by using short time windows.

For each pixel $\boldsymbol{x}$ in this image plane, we collect the events mapped to that location as $\mathcal{E}'_{\boldsymbol{x}}$. Then, we construct a time-image representation by calculating the average timestamp of the events at each pixel:

$$T_{\boldsymbol{x}} = \frac{1}{|\mathcal{E}'_{\boldsymbol{x}}|}\sum_{\mathcal{E}'_{\boldsymbol{x}}} t_i. \tag{2}$$

Finally, to distinguish between environment-triggered and obstacle-triggered events, we compute a score $\rho_{imu}(\boldsymbol{x})$ for each pixel based on the time differences:

$$\rho_{imu}(\boldsymbol{x}) = \frac{T_{\boldsymbol{x}} - \overline{T}}{\delta t}, \tag{3}$$

where $\overline{T}$ is the average event time over all pixels in the image. A higher score indicates a greater likelihood that the event is caused by an obstacle, as opposed to the static environment.

In the filtering process described, we utilize rotation information from the IMUs of both the left and right cameras independently, applying the filtering algorithm to the events in each view. In Fig.8, we present an example of event filtering. Fig.8a and Fig.8b show the scene captured by a conventional camera and the raw event stream from the event camera, respectively. Fig.8c shows the event filtering result based on ego-motion instruction.

4.1.2 *Event Filtering Based On Binocular Consistency*

To further enhance the filtering of environment-triggered events, we introduce a binocular consistency constraint. The key insight is that a pair of obstacle-triggered events captured by a rigidly attached stereo camera should satisfy the epipolar constraint [45].

Modeling. At time t_i, let the predicted obstacle location be $\boldsymbol{P}_i = (X_i, Y_i, Z_i)$, as determined by dorsal stream-inspired obstacle tracking (detailed in §4.3). The coordinate of an event $\boldsymbol{e}_i^l = (\boldsymbol{x}_i^l, t_i^l, p_i^l)$ from the left camera can be transformed to the right camera's frame of reference using:

$$\boldsymbol{x}_i^r = \boldsymbol{\pi}(\boldsymbol{T}_{rl}, Z_i \boldsymbol{K}^{-1} \boldsymbol{x}_i^l), \tag{4}$$

where $\boldsymbol{T}_{rl}$ is the transformation matrix from left to right camera, and $\boldsymbol{\pi} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ projects a 3D point $\boldsymbol{P}$ onto the image plane given a pose transformation $\boldsymbol{T}$:

$$\boldsymbol{\pi}(\boldsymbol{T}, \boldsymbol{P}) = \frac{1}{Z} \boldsymbol{K} \boldsymbol{T} \boldsymbol{P}, \ \boldsymbol{P} = [X, Y, Z]^T. \tag{5}$$

Due to inherent uncertainties in obstacle localization, we account for potential errors in the estimated position $\boldsymbol{P}_i$ by considering additional pixels around $\boldsymbol{x}_i^r$ along the epipolar line. Since the event cameras are horizontally aligned and face the same direction, the epipolar line is horizontal, so the search area includes pixels $\boldsymbol{x}_i^r \pm [\delta x, 0]^T$, where δx represents the uncertainty, typically set to 5 pixels. The consistency score for each event e_i^l is defined as:

$$\rho_{bi}(\boldsymbol{x}_i^l) = \min_{t^r} |t_i^l - t^r|, \tag{6}$$

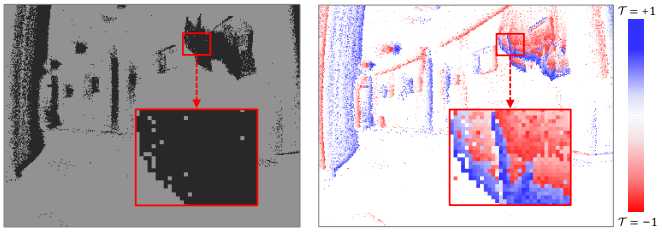
where t^r is the timestamp of events within the search area on the right image plane. A smaller $\rho_{bi}(\boldsymbol{x}_i^l)$ indicates a higher likelihood of finding a corresponding event pair, suggesting that e_i^l is more likely triggered by an obstacle.

4.1.3 *Put Together*

To produce a better filtering result, we integrate both filtering methodologies, necessitating the normalization of their respective scores first. Let the normalized scores of ρ_{imu} and ρ_{bi} be represented as ρ'_{imu} and ρ'_{bi}, respectively. We then employ a linear combination of these normalized scores:

$$\rho(\boldsymbol{x}) = \alpha \rho'_{imu}(\boldsymbol{x}) + (1-\alpha)\rho'_{bi}(\boldsymbol{x}), \alpha = \frac{\omega}{\omega + kv} \in [0, 1], \tag{7}$$

where v (in m/s) and ω (in $^\circ/s$) are the translational and rotational speeds of the camera, respectively. The parameter k serves as a weight to balance the influence of the camera's translational and rotational speed on the filtering process. A larger k places more emphasis to the translational component, making the filtering more sensitive to binocular consistency ρ'_{bi}, while a smaller k shifts the focus towards ego-rotation instruction ρ'_{imu}. In practice, since events generated by rotation tend to dominate [23], [29], we empirically set $k = 0.8$ to achieve a balanced integration of both filtering methods. If $\rho(\boldsymbol{x}) \geq \tau_{threshold}$, the pixel is classified as part of a moving obstacle; otherwise, it is considered part of the background. $\tau_{threshold}$ is a manually set threshold, ranging from 0.4 to 0.6 in our configuration. A higher $\tau_{threshold}$ results in more thorough background event filtering, but setting it too high risks incorrectly discarding obstacle-related events. The final result of event filtering is shown in Fig.8d.



(a) Event Sticking (b) Polarity Time-surface

Fig. 9. Event representation method comparison.

4.2 LGN-Inspired Event Matching

Visual signals passing through the optic chiasm are spatio-temporally correlated at LGN in order to obtain a 3D representation of the object. As shown in Fig.10, the architecture of LGN is characterized by six distinctive layers. The inner two layers are magnocellular layers that are responsible for detecting object motion and size (i.e., coarse feature), while the outer four layers are parvocellular layers for detecting the object's color and contour (i.e., fine details) [42]. Such a six-layer folding architecture supports a plethora of anatomical calculations without involving those computationally intensive spatial and temporal correlations.

Inspired by this elegant structure, we propose a neural-enhanced event matching algorithm, as elaborated below.

- First, we propose a novel event stream representation, namely polarity time-surface (§4.2.1), that maps the 3D event stream to the 2D space without sacrificing the valuable event features. Such a design can expedite feature matching without hurting the matching accuracy.
- Second, similar to LGN, we propose a six-layer hierarchical event feature extraction and matching algorithm (§4.2.2) that can localize the obstacle based on the binocular polarity time-surface timely and accurately.

4.2.1 *Spatio-Temporal Representation of Events*

An effective representation of event streams is crucial for feature extraction and matching. The current practice sticks consecutive events into an event image every tens of milliseconds (Fig.9a) and applies conventional vision algorithms to each event image. However, such a design discards the rich spatial-temporal information hidden in the event stream and thus achieves inferior performance.

In BioDrone, we propose a lightweight representation of event streams, namely, Polarity Time-Surface (P-TS), that can well retain rich spatio-temporal information. P-TS is a 2D map where each pixel value represents both the polarity and timestamp of the event. For instance, as shown in Fig.9b, the red and blue color indicates two different polarities of the event while the darkness of the color shows the time this event being captured. We leverage an exponential decay kernel to prioritize recent events over past ones, mirroring how the LGN prioritizes fresh visual signals over older inputs [42]. Compared to uniform decay, the exponential decay enhances the pixel gradients at the most recent event locations, thereby accentuating the spatiotemporal features of the obstacle's current state in the P-TS representation. Specifically, for each pixel $\boldsymbol{x} = (u, v)^T$, its polarity time-surface presentation is formally defined as:

$$\mathcal{T}(\boldsymbol{x}, t) = \rho_{\text{last}}(x) \cdot \exp\left(-\frac{t - t_{\text{last}}(\boldsymbol{x})}{\eta}\right), \tag{8}$$

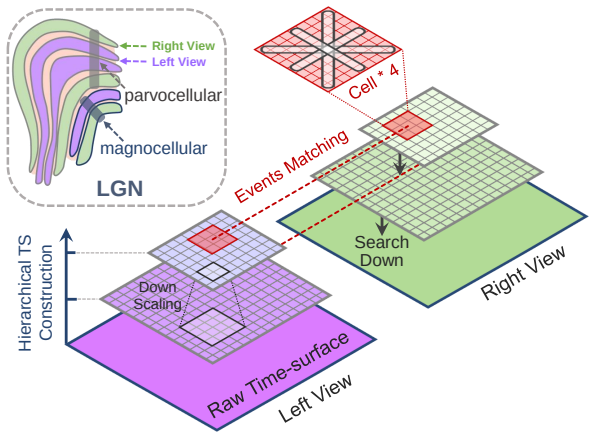


Fig. 10. Illustration of our proposed LEM algorithm.

where $t_{\text{last}}(\boldsymbol{x})$ and $\rho_{\text{last}}(x)$ are the timestamp and polarity of the event showing up at pixel $\boldsymbol{x}$; η is the decay rate. The parameter $\rho_{\text{last}}(x)$ provides an additional polarity constraint for event matching. Compared to the sticked event image (Fig.9a), the proposed P-TS retains the fine-grained texture of the obstacle, making it easily distinguishable.

To accelerate the generation of P-TS from event streams, we adopt two principal strategies: (1) **Pixel pruning**, focusing solely on pixels experiencing obstacle-related events filtered through CEF, and within the current time window (i.e., $t - t_{\text{last}}(\boldsymbol{x}) < \delta t$). (2) **Hardware acceleration**, accomplished by developing a custom P-TS acceleration module on an FPGA, facilitating the efficient parallel processing of dense events within the stream.

In the generation of the P-TS, it is crucial to balance the size of the time window δt to manage its effect on both feature matching latency and accuracy. A smaller time window may lead to excessively sparse events in the P-TS, resulting in incomplete feature representations (e.g., discontinuous obstacle edges), which can reduce matching precision. Conversely, increasing the time window captures more complete features but requires longer event accumulation and increases the number of features to be processed, affecting real-time performance. In our configuration, the P-TS time window is kept consistent with the filtering algorithm (§4.1.1), with $\delta t = 10$ms, to strike a balance between overall reliability and real-time performance.

The transformation of discrete events into the P-TS enables spatiotemporal correlations that enhance feature extraction and matching. While this raises concerns about potentially compromising the efficiency gained from the inherent sparsity of event data, the P-TS preserves spatial sparsity by encoding only background-filtered events within a short time window. As shown in Fig.9b, non-obstacle pixels (white regions) are excluded from further processing. Additionally, by encoding multiple events at the same pixel location into a single temporal feature, the P-TS significantly reduces both storage requirements and computational load.

4.2.2 *Fast Event Feature Matching*

Next, we run a feature extraction and matching on the P-TS maps for obstacle localization. However, sweeping the entire P-TS maps for feature extraction and matching would consume significant amount of time. We thus resort to the lessons learned from LGN and design a pyramidal P-TS hierarchy to expedite feature extraction and matching.

On a high level, we build two 3-layer P-TS pyramids based on the P-TS map obtained from left and right event streams, respectively, as shown in Fig.10. In the P-TS pyramid, the bottom layer $\mathcal{T}_0$ (i.e., the original P-TS map) is subsampled by a factor of k to obtain the next pyramid level $\mathcal{T}_1$. And $\mathcal{T}_1$ is then subsampled in the same way to obtain $\mathcal{T}_2$. We are expected to see a sequence of reduced resolution P-TS map on the pyramidal P-TS hierarchy, with a growing reception field. Each pixel on the top layer $\mathcal{T}_2$ corresponds to a reception field of $k^2 \times k^2$. By sweeping the $\mathcal{T}_2$ map, we can essentially reduce the searching space by k^4.

Pyramidal P-TS hierarchy generation. Let $\mathcal{T}_0^l$ and $\mathcal{T}_0^r$ be the left and right P-TS map, respectively. Without losing generality, we take the left P-TS map for algorithm description. We down-sample $\mathcal{T}_0$ by a factor of k:

$$\mathcal{T}_{i+1}(\boldsymbol{x}, t) = \mathcal{T}_i(k\boldsymbol{x} - \frac{k}{2}[1,1]^T, t), \tag{9}$$

where $\mathcal{T}_{i+1}(i \in \{0,1\})$ is the down-sampled P-TS. Each pixel in $\mathcal{T}_{i+1}$ represents a $k \times k$ region in previous P-TS, taking the value of the region's center pixel.

Feature Extraction. In contrast to conventional visual features, P-TS based features require a tailored approach that accounts for event generation characteristics to facilitate effective matching. We propose a *cell operator* to enhance stereo event feature matching, as illustrated by the red block in Fig.10. This operator comprises four directional segments (i.e., a horizontal, a vertical, and two diagonal), each spanning 9 pixels. This design reflects edge textures of event from various motion directions, aligning with the principle of LGN's magnocellular's responsiveness to moving edges.

For feature comparison, we compute the Sum of Absolute Differences (SAD) between pairs of cell operators. Within each operator, pixel scores are uniformly averaged, providing a measure of feature similarity. This cell operator accentuates attention to the edge textures of moving objects, resonating with the innate properties of event cameras and the characteristics of high-speed moving obstacles.

Feature matching. We perform hierarchical feature matching in a top-to-down direction and coarse-to-fine granularity (i.e., from $\mathcal{T}_2$ to $\mathcal{T}_0$). Specifically, the hierarchical matching process consists of a global (on upper layer) and a local (on lower layer) two stages.

In the global stage, we initiate an epipolar search on the downsampled $\mathcal{T}_2^r$ to reduce the search scope. For a feature point x^l in the left view, its corresponding match x^r in the right view must fulfill the epipolar constraint:

$$(\boldsymbol{x}^l)^T \boldsymbol{F} \boldsymbol{x}^r = 0, \tag{10}$$

where $\boldsymbol{F}$ represents the fundamental matrix. This equation restricts the search to the epipolar line.

The local stage involves searching within a $k \times k$ area on $\mathcal{T}_1^r$, corresponding to the matched point from $\mathcal{T}_2^r$, and repeating this process on $\mathcal{T}_0^r$ for finer matching. Once matched feature points x^l and x^r are identified, the depth of the point is calculated as:

$$d = \frac{f * baseline}{disp}, disp = ||\boldsymbol{x}^l - \boldsymbol{x}^r||, \tag{11}$$

where f is the focal length and $baseline$ is the distance between two optical centers.

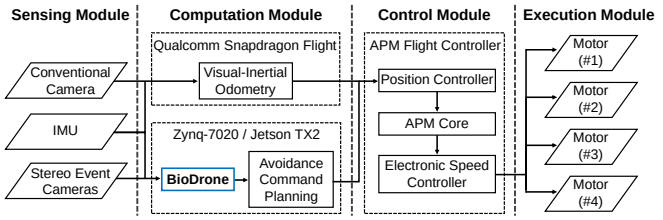


Fig. 11. Implementation of BioDrone on a drone platform for drone obstacle avoidance.

4.3 Dorsal Stream-Inspired Obstacle Tracking

The dorsal stream, integral to the visual processing system, facilitates spatial awareness and action guidance. In obstacle avoidance, it exhibits two key functions: (i) *predictive coding*, which leverages current visual and memory-based information to anticipate the location of obstacles, thereby aiding action planning, and (ii) *neural plasticity*, utilizing accumulated experiences to refine the motion analysis, thus enhancing the capability of obstacle tracking.

Inspired by the dorsal stream's efficient features in motion processing, we introduce a dorsal stream-inspired algorithm for obstacle tracking. This algorithm is designed to determine obstacle motion states (i.e., location and velocity), which are essential for drone path planning algorithms (e.g., artificial potential field [46], [47]). In alignment with the dorsal stream's mechanisms, our algorithm (i) constructs the motion model of obstacle for predicting its 3D location and (ii) integrates a gain vector to balance the tracking stability and responsiveness.

Specifically, we first compute the obstacle's location in the camera coordinate system, denoted as $\boldsymbol{p}_c$:

$$\boldsymbol{p}_c = d_{\text{obs}} \boldsymbol{K}^{-1} \boldsymbol{x}_{\text{c}}, \tag{12}$$

where $\boldsymbol{x}_{\text{c}}$ is the centroid of the obstacle, and the depth d_{obs} is determined as the average depth of the closest 10% of pixels on the obstacle. Then, $\boldsymbol{p}_c$ is transformed into $\boldsymbol{p}_w$, which represents the location in the world coordinate system:

$$\boldsymbol{p}_w = T_{wc} \boldsymbol{p}_c. \tag{13}$$

Here, T_{wc}, the transformation matrix from the camera coordinate system to the world coordinate system, is derived by integrating data from the drone's IMU [48]. To reduce the drift caused by long-term accumulation of IMU data, the origin of the world coordinate system is set at the drone's location upon initial detection of the obstacle.

Subsequently, we perform a recursive prediction of the obstacle's state:

$$\hat{\boldsymbol{\theta}}(t_i|t_{i-1}) = \boldsymbol{\phi}(t_i) \hat{\boldsymbol{\theta}}(t_{i-1}), \tag{14}$$

$$\boldsymbol{\phi}(t_i) = \begin{bmatrix} \mathbf{I}_{3\times3} & \Delta t \cdot \mathbf{I}_{3\times3} \\ 0_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}, \tag{15}$$

where $\boldsymbol{\phi}(t_i)$ is the linear motion model. The state vector $\hat{\boldsymbol{\theta}} = [\hat{\boldsymbol{p}}_w, \hat{\boldsymbol{v}}_w]^T$ is composed of the estimated location and velocity, and both set to 0 initially.

The gain vector $\mathbf{K}(t_i)$, dynamically modulates the influence between new observations and past trajectory on the state estimate, is calculated as:

$$\mathbf{K}(t_i) = \frac{\boldsymbol{P}(t_{i-1}) \boldsymbol{\phi}(t_i)^T}{\lambda + \boldsymbol{\phi}(t_i) \boldsymbol{P}(t_{i-1}) \boldsymbol{\phi}(t_i)^T}, \tag{16}$$

where $\boldsymbol{P}(t_{i-1})$ is the error covariance matrix. λ is the forgetting factor, set to 0.95, balancing the latest measurements

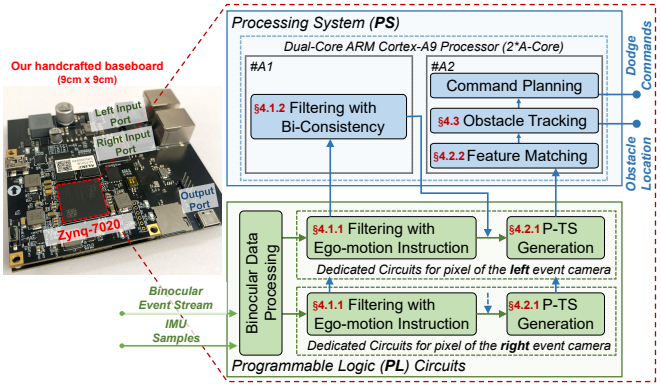


Fig. 12. Implementation of BioDrone on a Zynq chip.

with the existing state estimate.

Upon receiving an updated observation of the obstacle, the residual between the observed and estimated locations is computed as follows:

$$\mathbf{e}(t_i) = \boldsymbol{p}_w(t_i) - \hat{\boldsymbol{p}}_w(t_i|t_{i-1}). \tag{17}$$

Utilizing the gain vector and the observation residual, the state estimate is updated as follows:

$$\hat{\boldsymbol{\theta}}(t_i) = \hat{\boldsymbol{\theta}}(t_{i-1}) + \mathbf{K}(t_i)\mathbf{e}(t_i), \tag{18}$$

and the error covariance matrix is revised for the upcoming iteration:

$$\boldsymbol{P}(t_i) = \frac{1}{\lambda}(\boldsymbol{P}(t_{i-1}) - \mathbf{K}(t_i)\boldsymbol{\phi}(t_i)\boldsymbol{P}(t_{i-1})). \tag{19}$$

By iteratively applying Eq.14-19, we continuously update the trajectory of the obstacle while simultaneously predicting its future location.

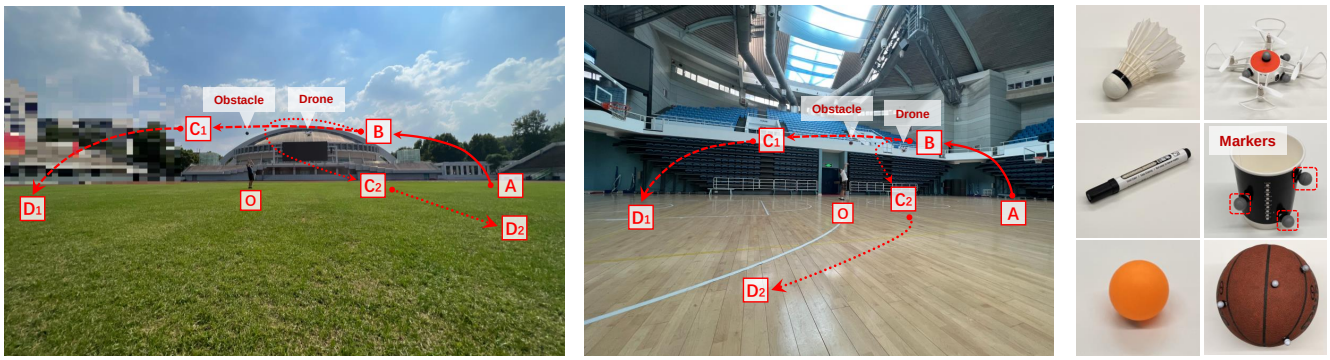
In summary, the dorsal stream-inspired obstacle tracking algorithm compensates for the drone's irregular motion (Eq.13) while assuming that obstacles follow continuous motion within a shot time window. This design offers two key benefits: (i) it allows for rapid estimation of the obstacle's current location from past states, ensuring low latency. and (ii) with the integration of a forgetting factor, it prioritizes recent data, which is crucial for real-time obstacle avoidance, where the immediate state of the obstacle is more relevant than the stability of its global trajectory.

5 Implementation

BioDrone's efficient algorithm design enables deployment on most general-purpose computing units, while more powerful devices are better equipped to handle real-time obstacle avoidance in highly dynamic drone flights. Moreover, as BioDrone must handle simultaneous event triggers, it is inherently well-suited for parallel acceleration on platforms such as FPGA and GPU. In this section, we first describe its standard implementation on drone platforms (§5.1), followed by how heterogeneous computing platforms (i.e., Xilinx Zynq-7020) accelerate BioDrone (§5.2).

5.1 BioDrone's On-Board Implementation

• **Hardware:** We deploy BioDrone on an AMOVLAB P450-NX drone. Fig.11 shows the diagram of the drone obstacle avoidance system. The drone is equipped with two on-board computational units: (i) a Qualcomm Snapdragon Flight for monocular visual-inertial odometry (VIO); and



(a) Outdoor experiments (b) Indoor experiments (c) Different Obstacles

Fig. 13. Experimental scenarios of BioDrone. The red lines show the drone's movement trajectory.

(ii) Xilinx Zynq-7020 chip or Nvidia Jetson TX2 (accompanied with an AUVIDEA J90 carrier board) running BioDrone's obstacle detection and localization software stack. The obstacle states (§4.3) are fed to the ArduPilot Mega (APM) flight controller for route planning. The drone testbed is equipped with a pair of front-facing DAVIS-346 event cameras. These two cameras are mounted with a baseline separation of $6cm$. The horizontal and vertical FoV of the event camera is 120° and 100°, respectively, with 346 × 260 pixels QVGA resolution.

• **Software:** The algorithms are implemented on the robotics operating system (ROS) in C++. We use the open source event camera driver [49] to stream event outputs, and the avoidance algorithm proposed in [23] to plan avoidance commands based on the trajectory and predicted location of obstacle. To reduce latency, we implement the obstacle localization and avoidance algorithms in the same ROS module, so that no message exchange is needed between drivers and the position controller.

5.2 Software and Hardware Co-Design

We implement BioDrone on a Xilinx Zynq-7020 through software-hardware co-design, as shown in Fig.12. It consists of a processing system (PS) and a programmable logic (PL) two modules. The PS features a dual-core ARM Cortex-A9 processor (i.e., #A1 and #A2), while PL is for hardware acceleration through FPGA. We also manufacture a baseboard for data input/output and voltage adaption.

• **PL:** We design exclusive logic circuits on FPGA to accelerate those event operations suitable for parallel and pipeline execution, i.e., data denoising, ego-motion-based filtering (§4.1.1), and P-TS generation (§4.2.1), on PL.

• **PS:** Before loading specific tasks, we first exploit a core-isolation strategy to isolate the computing resources of #A1 from PS, reducing the impact of CPU scheduling on task execution to better match the PL pipelines. We realize it by building a Linux OS with boot parameter `isolcpus=<cpu #A1>`. We execute the binocular-based filtering which requires frequent memory access and cannot be easily implemented through FPGA, on #A1. The feature matching, obstacle tracking, and command planning tasks are executed on #A2.

• **Data flow in-between:** We further leverage the physical-level direct memory access (DMA) technique [50] to transmit intermediate data among PL, #A1, and #A2. Compared with network-level solutions such as PL-PS ethernet interface [51] and OpenAMP [52], DMA ensures data interaction processes would not be interrupted by CPU scheduling.

TABLE 1
Different Drone Flight Mode Configurations

Flight Mode* (Trajectory)	Speed		Event Generating Speed (e/ms)
	Translation (m/s)	Rotation (°/s)	
1 ($A \rightarrow B \rightarrow C_1 \rightarrow D_1$)	2.0–6.0	0–10.0	100–400
2 ($A \rightrightarrows B \rightarrow C_1 \rightrightarrows D_1$)	15.0–26.5	0–10.0	350–1200
3 ($A \rightarrow B \curvearrowright C_2 \rightarrow D_2$)	2.0–6.0	20–100	900–2100

* $\rightarrow$: acceleration; $\rightarrow$: uniformity; $\rightarrow$: deceleration.

6 Evaluation

6.1 Experimental Methodology

Field studies. We conduct field studies both indoors and outdoors as shown in Fig.13. The performance is evaluated in three flight modes, as defined in Table 1. The drone follows planned trajectories to move, and four volunteers throw six different types of obstacles toward the drone during the drone's movement.

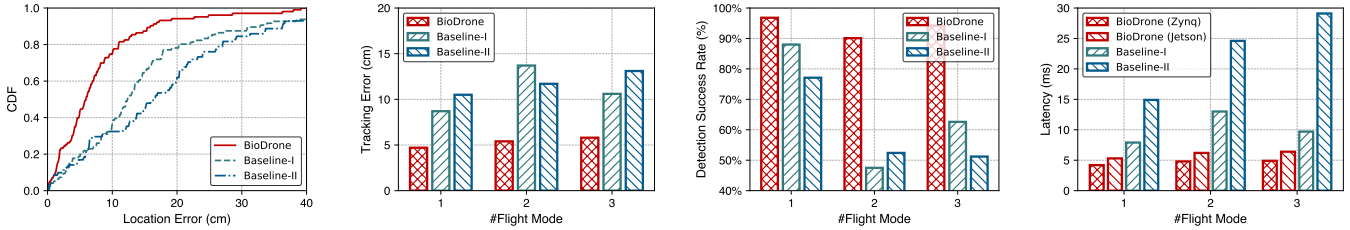
Repeatability. Before conducting experiments under each flight mode, we program a series of pre-determined flight commands into the on-board APM flight controller, enabling the drone to follow the planned trajectory, speed, and acceleration, to make our experiments repeatable.

Metrics and Ground truth. The drone logs its localization results with timestamps. We download these logs and evaluate end-to-end (E2E) localization latency Δt_l and error Δx (defined in §2). Indoors, an OptiTrack motion capture system could provide $<1mm$ localization ground truth. Outdoors, since we cannot deploy OptiTrack to obtain ground truth, we collect event streams and run an advanced yet heavy event-based object localization and segmentation neural network [53] offline. The results are taken as ground truth. We also log event classification results reported by it to examine BioDrone's event filtering performance.

Baselines. We compare the accuracy and latency of BioDrone with Baseline-I [23] and -II [29]. We also compare the LEM module in BioDrone with ESVO [32]. As these baselines are not implemented on FPGA, we thus implement BioDrone on the drone's onboard Nvidia Jetson TX2 for a fair comparison with them.

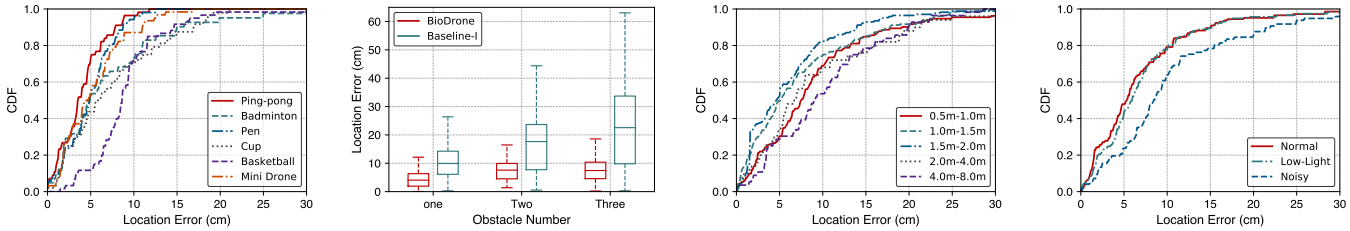
6.2 Overall Performance

Obstacle localization and tracking. We first evaluate the accuracy of localization and trajectory tracking for obstacles. As illustrated in Fig.14a, the performance of BioDrone in obstacle single-point localization is compared with two other systems. BioDrone achieves an average localization error of $7.5cm$, outperforming Baseline-I and Baseline-II, which



(a) Localization Error (b) Tracking Error (c) Objection Detection Rate (d) Localization Latency

Fig. 14. Overall Performance Comparison



(a) Impact of Obstacle Type (b) Impact of Obstacle Number (c) Impact of Obstacle Distance (d) Impact of Scene Dynamic

Fig. 15. System Robustness Evaluation

exhibit average errors of 15.9cm and 20.4cm, respectively. Furthermore, when comparing the three systems across various flight modes in terms of Average Trajectory Error (ATE), Fig.14b illustrates that BioDrone outperforms the two baselines by at least 45.9%, 53.8% and 44.6% in flight modes 1, 2 and 3, respectively.

Obstacle detection. As shown in Fig.14c, BioDrone achieves obstacle detection rates of 96.8%, 90.1%, and 94.2% in three distinct flight modes, outperforming the baseline by over 10% in low-speed (mode 1) and by more than 51% in high-speed (modes 2 & 3) scenarios. Unlike related works that predominantly use ego-motion instruction for event filtering, BioDrone significantly boosts its detection efficiency by incorporating a binocular consistency constraint.

End-to-end latency. We further evaluate the E2E latency, covering the obstacle detection, localization, and tracking phases. As shown in Fig.14d, BioDrone achieves an E2E latency of under 6.4ms on the general-purpose Jetson platform. When deployed on the Zynq platform, the latency is reduced by over 20.7%, owing to our FPGA-based hardware architecture, which facilitates parallel and pipelined processing of events, significantly enhancing throughput and efficiency. On the same Jetson platform, BioDrone outperforms baseline methods, reducing latency by over 32.9% in flight mode 1. As flight speeds increase, the latency for Baseline-I and II rises significantly, while BioDrone outperforms both baselines in flight modes 2 and 3 by more than 52.3% and 34.1%, respectively.

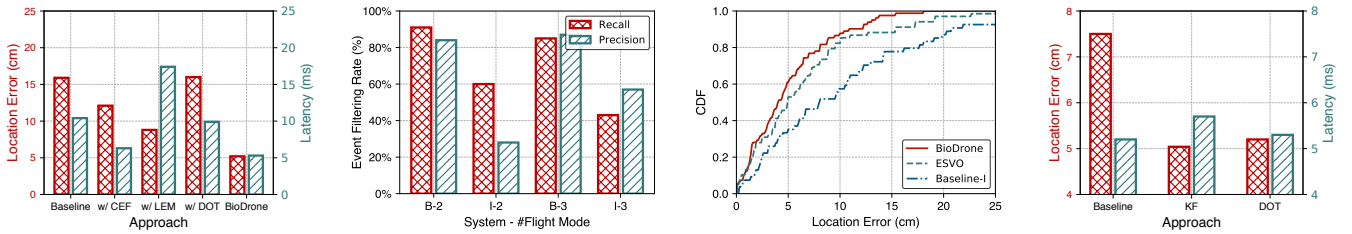
6.3 System Robustness Evaluation

Impact of obstacle type. We assessed how different types of obstacles (Fig.13c, varying in form factor and texture) affect performance. The findings, displayed in Fig.15a, reveal that smaller and textured obstacles such as ping-pong, and mini-drone result in lower average localization errors of 4.1cm, and 5.3cm, respectively. In contrast, obstacles with fewer textures and larger volumes (e.g., a basketball) tend to exhibit greater localization errors. This is due to two factors: (i) the lack of texture directly reduces the number of events generated within the obstacle's pixel space, and (ii) larger obstacles trigger events that are more widely spaced along their edges. Together, these factors reduce the effective features captured by the cell operator (§4.2.2) for stereo matching, reducing the robustness of the localization.

Impact of obstacle quantity. In this experiment, two volunteers are asked to throw multiple (2-3) obstacles toward the drone; we examine the detection and localization results for each obstacle individually. As depicted in Fig.16b, BioDrone outperforms Baseline-I by >40% in all settings. As the number of obstacles grows, we observe a slight increase (around 3cm) in BioDrone's localization error. In contrast, the localization error of Baseline-I grows dramatically to 24.68cm. The results demonstrate that the LEM module could extract spatio-temporal features of different obstacles and thus distinguish them from each other. On the contrary, Baseline-I simply clusters events for triangulation, making it difficult to separate obstacles close to each other.

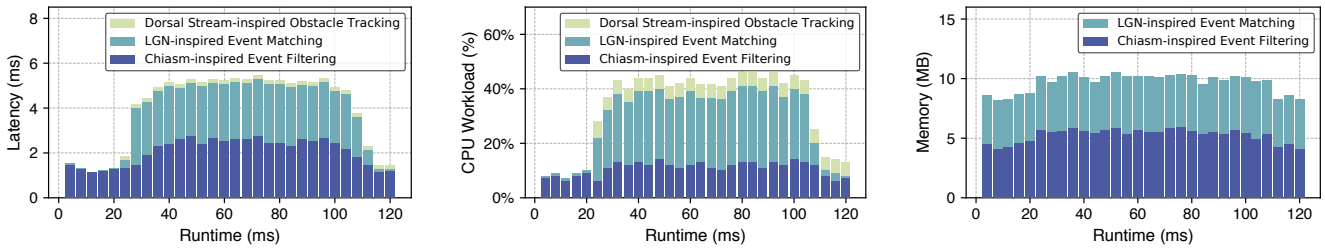
Impact of obstacle distance As shown in Fig.15c, when the obstacle appears at around 1.0-2.0m, BioDrone achieves the highest localization accuracy where the average location error is 6.58cm, and the average location error will slightly increase (within 9.5cm though) as the distance increases. Generally, a longer distance fails to generate sufficient events, making the feature matching more challenging.

Impact of environmental dynamic. We further assess BioDrone's performance in low-light conditions (i.e., nighttime with illumination levels below 30 lux) and in noisy environments (i.e., 1-3 pedestrians randomly crossing the camera's field of view). The results are shown in Fig.15d. As seen, even in low-light conditions, the average localization accuracy remains consistent (a minor decrease within 10%), due to the HDR of event camera that still manages to capture sufficient events in dark environments. However, in noisy environments, there is a 33% decrease in average localization accuracy because BioDrone sometimes interprets dynamic pedestrians as foreground obstacles. A potential solution to this problem could be the integration of depth-based filtering algorithms, which is left as a future work.



(a) Impact of Different Module (b) Chiasm-inspired Events Filtering (c) LGN-inspired Events Matching (d) DS-inspired Obstacle Tracking

Fig. 16. Ablation Study



(a) System Latency (b) CPU Workload (c) Memory Usage

Fig. 17. System Efficiency on the Drone

6.4 Ablation Study

Contributions of each module. BioDrone encompasses three pivotal modules: the Chiasm-inspired Event Filtering (CEF), the LGN-inspired Event Matching (LEM), and the Dorsal stream-inspired Obstacle Tracking (DOT). In our experiment, we assess the individual and combined impacts of these modules on BioDrone's performance. We integrate these modules into Baseline-I respectively, evaluating changes in localization accuracy and end-to-end latency. According to Fig.17a, Baseline-I, without these modules, records a localization error of 15.9cm and a latency of 10.4ms. The incorporation of the CEF module leads to a reduction in localization error to 11.6cm and a decrease in latency to 6.3ms. Adding the LEM module reduces the localization error to 8.8cm, albeit with a slight increase in delay due to the absence of an efficient event-filtering mechanism. And the integration of the DOT module, in place of the conventional Kalman filter, results in a latency decrease of 0.4ms while maintaining accuracy. Upon fully incorporating CEF, LEM, and DOT into Baseline-I, the system reaches optimal performance, minimizing both localization error and E2E latency.

Performance of CEF. We compare CEF with the filtering module of Baseline-I in high-speed mode (flight modes 2 and 3). We denote CEF in mode 2 and mode 3 as B-2 and B-3, respectively. Likewise, the filtering module of Baseline-I as I-2 and I-3, respectively. In Fig.16b, a higher recall means more obstacle-triggered events are preserved while a higher precision indicates more background events are removed. As seen, the recall of CEF is $\geq$ 85% and its precision is $\geq$ 84% under all flight modes. In contrast, the filter module of Baseline-I achieves an inferior recall of 43% under flight mode 3. Even worse, the precision further drops to 28% under flight mode 2. This result demonstrates the efficacy of CEF in event filtering.

Performance of LEM. We evaluate the performance of LEM by comparing it with the localization module in ESVO [32] and Baseline-I. As shown in Fig.16c, LEM reduces localization error by 23.8% compared to ESVO where event features are matched using naive block-matching operations. Besides, compared with Baseline-I, which exploits event clustering and triangulates the spatial location of an object at cluster-level, LEM reduces the localization error by 55.1%.

Performance of DOT. We evaluate the enhancement in obstacle tracking introduced by the proposed DOT algorithm. In this experiment, a BioDrone setup without any tracking algorithm serves as the Baseline, while Kalman Filter (KF) implementations from Baseline-I and -II are used for comparison. As illustrated in Fig.16d, DOT, compared to the Baseline, reduces localization error by 30.6% and achieves precision comparable to KF, with only a marginal increase in computation delay of 0.12ms. In contrast, the KF approach requires an average latency of 0.46ms, which heightens the risk of obstacle avoidance failure.

6.5 System Efficiency Study

As a drone-oriented obstacle avoidance system, it is important to achieve a balance among computing latency, CPU workload and memory usage, to ensure the system effectively working on resource-constrained drone devices. We analyzed a representative 120ms obstacle avoidance scenario, logging system latency, CPU workload, and memory usage as shown in Fig.17. The drone detects an obstacle at 24ms and promptly initiates an avoidance maneuver, with the obstacle leaving the drone's field of view after 110ms.

- *0-24ms*: Before the obstacle is perceived, LEM and DOT remain inactive, contributing negligible computing latency and CPU workload. CEF operates during this phase with a latency under 0.15ms and CPU workload below 10%.
- *24-110ms*: Upon obstacle detection, CEF experiences increased computing latency and CPU workload due to heightened event activity, yet maintains a low latency of 2.65ms and CPU usage under 13%. Concurrently, LEM actively localizes the obstacle, adding 2.42ms of latency and approximately 25% CPU workload. DOT, while persistently tracking and predicting obstacle positions, incurs a maximum of 0.2ms computational latency, 5% CPU workload, and negligible memory usage.

• *110-120ms*: As the drone successfully avoids the obstacle, the modules revert to their pre-$24ms$ states in terms of latency and CPU workload. Across the entire process, the memory footprint of the three modules remains <11 MB. Throughout the avoidance procedure, BioDrone reserves over 50% of CPU resources for higher-level tasks.

In addition, in comparison to traditional drone-based industrial applications, BioDrone requires the use of stereo event cameras, which introduces additional resource consumption. Specifically, (i) the weight of the event cameras has a marginal impact on the overall power consumption of the drone. In our configuration, two DAVIS 346 event cameras weigh a total of 200g, which is considerably lower than the maximum payload capacity of industrial drones (e.g., the P450 drone, >2.2kg). As event cameras continue to improve with better integration and lighter designs, this impact is expected to diminish further. (ii) Moreover, while event cameras do consume some power (typically less than 100mW [26]), their exceptionally low energy demand makes them highly suitable as sensors for drone applications.

7 Related work

Obstacle avoidance with traditional sensors. Nowadays, fast and safe obstacle avoidance has attracted great interest from both academia and industry [54]. Current research predominantly utilizes frame-based cameras (i.e., monocular [55] and stereo systems [56]), depth cameras [57], millimeter-wave radars [58], and LiDAR [59]. However, these approaches typically presume that obstacles are either stationary or exhibit only slow relative motion (i.e., less than $5m/s$), and fall short for high-speed drones (i.e., relative speeds exceeding $20m/s$). This limitation is rooted in the inherent properties of the sensors and is not readily addressable through algorithmic enhancements.

Event-based algorithms and systems. Event cameras, heralding significant advantages over frame-based cameras, provide *high temporal resolution, low latency, and high dynamic range*. Recent years have seen an upsurge in research developing algorithms and systems utilizing event cameras [26], such as scene reconstruction [32], SLAM [60], object tracking [23], [29], and HDR image reconstruction [61]. Among these, Baseline-I [23] emerges as a significant drone obstacle avoidance solution, employing IMU data to eliminate background events and enhance obstacle detection, closely aligning with our research. Our work, BioDrone, diverges from Baseline-I's monocular setup, embracing a binocular configuration for obstacle localization. This transition from detection to localization, alongside the shift in hardware setup, presents new challenges in fully exploiting the capabilities of event cameras for drone obstacle avoidance.

Bio-inspired design for event-based vision. Biological principles drive the design of event camera pixels and some event processing algorithms, such as spiking neural networks (SNN [39]), spatiotemporal oriented filters (STOF [62]), and spike-timing dependent plasticity (STDP [63]). In general, current innovations mainly mimic the working principles of the human visual cortex and design sophisticated algorithms for high-level object recognition [64], segmentation [38], and understanding [65]. Albeit inspiring, these bio-inspired systems are not the optimal solution for obstacle avoidance-related tasks due to the large computational overhead. In BioDrone, we find those delay-sensitive tasks are not executed at the visual cortex but exactly at the earlier *binocular visual pathway*. We take the bio-lessons learned from it and design BioDrone for fast obstacle detection, matching, and tracking.

8 Conclusions

We have presented the design and implementation of BioDrone, a solution to support fast and accurate drone obstacle detection and localization using event cameras. BioDrone exploits biological knowledge behind human visual systems and designs a visual pathway-inspired architecture, a chiasm-inspired event filtering module, an LGN-inspired event matching mechanism, and a dorsal stream-inspired obstacle tracking algorithm to unleash the full potential of event cameras. We fully implement BioDrone on a Zynq chip through software-hardware co-design. Extensive evaluations conducted on an industrial drone demonstrate its superior performance. Through BioDrone, we present that the bio-inspired design paradigm produces simple yet effective solutions to potentially replace heavy-weight ones, adding a new solution dimension for sensing problems with strict restrictions on accuracy, latency, and computation.

9 Acknowledgments

We sincerely thank the MobiSense group and the anonymous reviewers for their insightful comments. This work is supported in part by the NSFC under grant No. 62372265, No. 62302254, and No. 62402276.

References

[1] S. Jha, Y. Li, S. Noghabi, V. Ranganathan, P. Kumar, A. Nelson, M. Toelle, S. Sinha, R. Chandra, and A. Badam, "Visage: enabling timely analytics for drone imagery," in *ACM MobiCom*, 2021.

[2] A. Jain, Z. Kapetanovic, A. Kumar, V. N. Swamy, R. Patil, D. Vasisht, R. Sharma, M. Swaminathan, R. Chandra, A. Badam *et al.*, "Low-cost aerial imaging for small holder farmers," in *Proceedings of the ACM Compass*, 2019.

[3] A. Balasingam, K. Gopalakrishnan, R. Mittal, M. Alizadeh, H. Balakrishnan, and H. Balakrishnan, "Toward a marketplace for aerial computing," in *Proceedings of the ACM DroNet*, 2021.

[4] Y. Ma, N. Selby, and F. Adib, "Drone relays for battery-free networks," in *Proceedings of the ACM Sigcomm*, 2017.

[5] W. Wang, L. Mottola, Y. He, J. Li, Y. Sun, S. Li, H. Jing, and Y. Wang, "Micnest: Long-range instant acoustic localization of drones in precise landing," in *Proceedings of the ACM Sensys*, 2022.

[6] K.-L. Wright, A. Sivakumar, P. Steenkiste, B. Yu, and F. Bai, "Cloudslam: Edge offloading of stateful vehicular applications," in *Proceedings of the IEEE/ACM SEC*, 2020.

[7] R. K. Sheshadri, E. Chai, K. Sundaresan, and S. Rangarajan, "Skyhaul: A self-organizing gigabit network in the sky," in *Proceedings of the ACM MobiHoc*, 2021.

[8] S. Chinchali, A. Sharma, J. Harrison, A. Elhafsi, D. Kang, E. Pergament, E. Cidon, S. Katti, and M. Pavone, "Network offloading policies for cloud robotics: a learning-based approach," *Autonomous Robots*, vol. 45, no. 7, pp. 997–1012, 2021.

[9] A. J. B. Ali, Z. S. Hashemifar, and K. Dantu, "Edge-slam: edge-assisted visual simultaneous localization and mapping," in *Proceedings of the ACM Mobisys*, 2020.

[10] J. Xu, H. Cao, Z. Yang, L. Shangguan, J. Zhang, X. He, and Y. Liu, "Swarmmap: Scaling up real-time collaborative visual slam at the edge," in *Proceedings of the USENIX NSDI*, 2022.

[11] Y. Chen, H. Inaltekin, and M. Gorlatova, "AdaptSLAM: Edge-assisted adaptive slam with resource constraints via uncertainty minimization," in *Proceedings of the IEEE INFOCOM*, 2023.

[12] DJI, "DJI Industrial Drones," https://www.dji.com/products/industrial, 2020.

[13] Amazon, "Amazon Drones Swarm," https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011, 2021.

[14] D. Mail, "When eagles attack! drone camera mistaken for rival," www.dailymail.co.uk/video/news/video-1154408/Golden-Eagle-attacks-drone-cameramistaking-rival.html, 2016.

[15] CNet, "Hawk attacks drone in a battle of claw versus machine," www.cnet.com/news/this-hawk-has-no-love-for-your-drone/, 2016.

[16] U. Ali, H. Cai, Y. Mostofi, and Y. Wardi, "Motion-communication co-optimization with cooperative load transfer in mobile robotics: An optimal control perspective," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 2, pp. 621–632, 2018.

[17] N. Garg and N. Roy, "Enabling self-defense in small drones," in *Proceedings of the ACM HotMobile Workshop*, 2020, pp. 15–20.

[18] T. Eppenberger, G. Cesari, M. Dymczyk, R. Siegwart, and R. Dubé, "Leveraging stereo-camera data for real-time dynamic obstacle detection and tracking," in *Proceedings of the IEEE/RSJ IROS*, 2020.

[19] F. Wimbauer, N. Yang, L. Von Stumberg, N. Zeller, and D. Cremers, "Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera," in *Proceedings of the IEEE/CVF CVPR*, 2021.

[20] D. Hutabarat, M. Rivai, D. Purwanto, and H. Hutomo, "Lidar-based obstacle avoidance for the autonomous mobile robot," in *Proceedings of the IEEE ICTS*, 2019.

[21] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proceedings of the IEEE ICRA*, 2016.

[22] J. Xu, G. Chi, Z. Yang, D. Li, Q. Zhang, Q. Ma, and X. Miao, "Followupar: Enabling follow-up effects in mobile ar applications," in *Proceedings of the ACM MobiSys*, June 24-July 2 2021.

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

[24] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *Proceedings of the IEEE ICRA*, 2017.

[25] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *Proceedings of the Springer ECCV*, 2016.

[26] G. Gallego, T. Delbrück, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis *et al.*, "Event-based vision: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 1, pp. 154–180, 2020.

[27] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Asynchronous convolutional networks for object detection in neuromorphic cameras," in *IEEE/CVF CVPR Workshops*, 2019.

[28] B. He, H. Li, S. Wu, D. Wang, Z. Zhang, Q. Dong, C. Xu, and F. Gao, "Fast-dynamic-vision: Detection and tracking dynamic objects with event and depth sensing," in *IEEE/RSJ IROS*, 2021.

[29] A. Mitrokhin, C. Fermüller, C. Parameshwara, and Y. Aloimonos, "Event-based moving object detection and tracking," in *Proceedings of the IEEE IROS*, 2018.

[30] Y. Nam, M. Mostafavi, K.-J. Yoon, and J. Choi, "Stereo depth from events cameras: Concentrate and focus on the future," in *Proceedings of the IEEE/CVF CVPR*, 2022, pp. 6114–6123.

[31] 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.

[32] Y. Zhou, G. Gallego, and S. Shen, "Event-based stereo visual odometry," *IEEE Transactions on Robotics*, 2021.

[33] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[34] N. Pham, H. Jia, M. Tran, T. Dinh, N. Bui, Y. Kwon, D. Ma, P. Nguyen, C. Mascolo, and T. Vu, "Pros: an efficient pattern-driven compressive sensing framework for low-power biopotential-based wearables with on-chip intelligence," in *ACM MobiCom*, 2022.

[35] A. Bakar, R. Goel, J. de Winkel, J. Huang, S. Ahmed, B. Islam, P. Pawełczak, K. S. Yıldırım, and J. Hester, "Protean: An energy-efficient and heterogeneous platform for adaptive and hardware-accelerated battery-free computing," in *Proceedings of the ACM SenSys*, 2022.

[36] Xilinx, "Xilinx Zynq-7000 SoC," https://www.xilinx.com/products/silicon-devices/soc/zynq-7000, 2022.

[37] Ardupilot, "ArduPilot Mega," https://ardupilot.org/, 2018.

[38] A. Mitrokhin, Z. Hua, C. Fermuller, and Y. Aloimonos, "Learning visual motion segmentation using event surfaces," in *IEEE CVPR*, 2020.

[39] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," *Neural networks*, 2019.

[40] G. Jeffery, "Architecture of the optic chiasm and the mechanisms that sculpt its development," *Physiological Reviews*, vol. 81, no. 4, pp. 1393–1414, 2001.

[41] K. E. Cullen, "The vestibular system: multimodal integration and encoding of self-motion for motor control," *Trends in neurosciences*, vol. 35, no. 3, pp. 185–196, 2012.

[42] C. Tailby, S. K. Cheong, A. N. Pietersen, S. G. Solomon, and P. R. Martin, "Colour and pattern selectivity of receptive fields in superior colliculus of marmoset monkeys," *The Journal of Physiology*, vol. 590, no. 16, pp. 4061–4077, 2012.

[43] M. Mishkin, L. G. Ungerleider, and K. A. Macko, "Object vision and spatial vision: two cortical pathways," *Trends in neurosciences*, 1983.

[44] L. Erskine and E. Herrera, "The retinal ganglion cell axon's journey: insights into molecular mechanisms of axon guidance," *Developmental biology*, 2007.

[45] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *International journal of computer vision*, 1998.

[46] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments," in *2015 IEEE international conference on robotics and automation (ICRA)*, 2015.

[47] A. Singletary, K. Klingebiel, J. Bourne, A. Browning, P. Tokumaru, and A. Ames, "Comparative analysis of control barrier functions and artificial potential fields for obstacle avoidance," in *IEEE/RSJ IROS*, 2021.

[48] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *Proceedings of the IEEE Transactions on Robotics*, 2018.

[49] UZH, "Event Camera Driver," https://github.com/uzh-rpg/rpg_dvs_ros, 2022.

[50] Xilinx, "Direct Memory Access," https://www.xilinx.com/products/intellectual-property/axi_dma.html, 2022.

[51] X. Inc. (2022) Mpsoc ps and pl ethernet example projects. [Online]. Available: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/478937213/MPSoC+PS+and+PL+Ethernet+Example+Projects

[52] OpoenAMP. (2022) Openamp project. [Online]. Available: https://www.openampproject.org/

[53] I. Alonso and A. C. Murillo, "Ev-segnet: Semantic segmentation for event-based cameras," in *IEEE/CVF CVPR*, 2019.

[54] P. Fraga-Lamas, L. Ramos, V. Mondéjar-Guerra, and T. M. Fernández-Caramés, "A review on iot deep learning uav systems for autonomous obstacle detection and collision avoidance," *Remote Sensing*, 2019.

[55] Z. Zhang, Y. Cao, M. Ding, L. Zhuang, and J. Tao, "Monocular vision based obstacle avoidance trajectory planning for unmanned aerial vehicle," *Aerospace Science and Technology*, 2020.

[56] V. S. Kalogeiton, K. Ioannidis, G. C. Sirakoulis, and E. B. Kosmatopoulos, "Real-time active slam and obstacle avoidance for an autonomous robot based on stereo vision," *Cybernetics and Systems*, 2019.

[57] D. Wang, W. Li, X. Liu, N. Li, and C. Zhang, "Uav environmental perception and autonomous obstacle avoidance: A deep learning and depth camera combined solution," *Computers and Electronics in Agriculture*, 2020.

[58] H. Yu, F. Zhang, P. Huang, C. Wang, and L. Yuanhao, "Autonomous obstacle avoidance for uav based on fusion of radar and monocular camera," in *IEEE/RSJ IROS*, 2020.

[59] N. Baras, G. Nantzios, D. Ziouzios, and M. Dasygenis, "Autonomous obstacle avoidance vehicle using lidar and an embedded system," in *Proceedings of the ACM MOCAST*, 2019.

[60] H. Rebecq, T. Horstschäfer, 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*, 2016.

[61] M. Mostafavi, L. Wang, and K.-J. Yoon, "Learning to reconstruct hdr images from events, with applications to depth and flow prediction," *International Journal of Computer Vision*, 2021.

[62] G. Orchard, R. Benosman, R. Etienne-Cummings, and N. V. Thakor, "A spiking neural network architecture for visual motion estimation," in *2013 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, 2013, pp. 298–301.

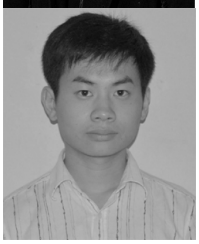
[63] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, 2008.

[64] Y. Nan, R. Xiao, S. Gao, and R. Yan, "An event-based hierarchy model for object recognition," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.

[65] L. A. Camuñas-Mesa, T. Serrano-Gotarredona, S.-H. Ieng, R. Benosman, and B. Linares-Barranco, "Event-driven stereo visual tracking algorithm to solve object occlusion," *IEEE transactions on neural networks and learning systems*, 2017.

Danyang Li received the B.E. degree in School of Software from Yanshan University in 2019 and the M.E. degree in School of Software from Tsinghua University in 2022. He is currently pursuing his Ph.D. degree in School of Software, Tsinghua University. His research interests include Internet of Things and mobile computing.

Jingao Xu received his B.E. and Ph.D. degree in School of Software from Tsinghua University in 2017 and 2022, respectively. He is now a Postdoc research fellow in School of Software, Tsinghua University. His research interests include Internet of Things and mobile computing.



Zheng Yang is an associate professor at Tsinghua University. He received a B.E. degree in computer science from Tsinghua University in 2006 and a Ph.D. degree in computer science from Hong Kong University of Science and Technology in 2010. His main research interests include Internet of Things and mobile computing. He is the PI of National Natural Science Fund for Excellent Young Scientist and has been awarded the State Natural Science Award (second class).

Yishujie Zhao received his B.E. and B.F.A. degrees from Tsinghua University at 2022, now his is pursuing a M.E. degree at School of Software, Tsinghua University under the supervision of Prof. Zheng Yang.

Hao Cao received his B.E. degree in College of Intelligence and Computing from Tianjin University in 2019. He is now a Ph.D. candidate in School of Software, Tsinghua University. His research interests include Internet of Things and mobile computing.

Yunhao Liu received his B.S. degree in Automation Department from Tsinghua University, and an M.A. degree in Beijing Foreign Studies University, China. He received an M.S. and a Ph.D. degree in Computer Science and Engineering at Michigan State University, USA. Yunhao is now MSU Foundation Professor and Chairperson of Department of Computer Science and Engineering, Michigan State University, and holds Chang Jiang Chair Professorship at Tsinghua University.

Longfei Shangguan received the B.S. degree from the School of Software, Xidian University, Shanghai, China, in 2011, and the Ph.D. degree from the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Hong Kong, in 2015. He is currently a researcher with Microsoft. His research interests include wireless networks, mobile systems, and low-power communication.