

# Event-based Highly Dynamic Obstacle Avoidance

Shifan Zhu

Hochul Hwang

Shangqun Yu

University of Massachusetts Amherst



Figure 1. Left image: the robot dodged the obstacle through a series of operation including event accumulation, denosing, obstacle tracking, trajectory prediction, and avoidance. Top right image: robot perspective of the predicted trajectory of the dynamic obstacle in event frame. Bottom right image: robot perspective of the environment.

## Abstract

Recent development in the robotics community enabled legged robots to maneuver agilely in complex environments. However, having these dynamic systems to avoid fast-moving obstacles in daily living scenarios is still a remaining challenge that limits deployment in real-world applications. Especially, although accurate detection of highly dynamic obstacles under dynamic robot motions is critical for successful obstacle avoidance, this problem is complex and challenging due to motion blur and the high latency of traditional image sensors. More recently, event cameras have shown great potential in alleviating relevant issues. In this paper, we propose a dynamic obstacle avoidance framework that consists of dynamic obstacle detection and trajectory prediction algorithms without using other sensors, such as depth cameras. Firstly, we present an accurate dynamic obstacle detection and tracking algorithm based on threshold event data. We then utilize Random

sample consensus (RANSAC) to track and predict the trajectory of the obstacle positions in 2D pixel coordinates. Finally, we introduce our avoidance framework that operates with predicted 2D obstacle positions. We perform extensive real-world experiments to validate our avoidance framework which performs 61.9% of obstacle avoidance success rate of a kicked ball. Our code and video are publicly available at: <https://github.com/DARoSLab/DynamicObjectsAvoidance>.

## 1. Introduction

Technological advancements in artificial intelligence, robotics, computer vision, and computation hardware have facilitated research in safety-ensure robot navigation by avoiding obstacles based on perception [7, 17, 18, 20]. However, recognizing highly dynamic obstacles (e.g., high-speed car, kicked ball) and avoiding them with high precision is still challenging by utilizing standard cameras as

they internally possess an average latency of tens of milliseconds. This may not be fast enough to avoid the high-speed obstacle for a short time interval. Moreover, such obstacles may arouse motion blur which raises the bars for successful and robust detection.

Recently, the event camera [10] [6], known as the neuromorphic camera, was introduced with the potential to solve such issues. Unlike traditional cameras that capture a whole image by using a shutter, an event camera reports local changes in brightness for each pixel independently and asynchronously. By capturing independent local events, unlike the traditional frame-based cameras, event cameras provide benefits such as high dynamic range, low latency, low power consumption, and improved motion blur [12]. Here, we exploit the event-based neuromorphic camera to distinguish between dynamic and static obstacles and utilize the low-latency feature of the event camera to achieve state-of-the-art fast-moving obstacle avoidance for our quadruped system.

Despite the advantages of event cameras, the data obtained from event cameras usually contains more noise and has a relatively low resolution (sparse) compared to dense pixel data obtained from frame-based cameras. In addition, event data is camera motion dependent which means the edges parallel to the motion will not generate events because the events are triggered by the brightness change from the relative motion of the camera. Furthermore, event cameras can only capture the relative changes in brightness. In other words, event cameras do not store any visual detail such as color and texture. Owing to the distinct sensing mechanisms of RGB and event cameras, RGB image-based obstacle avoidance algorithm cannot be directly applied to event data.

Here, we divide the dynamic obstacle detection algorithm into two situations to fully take advantage of the event camera and reflect their sensing mechanisms. (1) Static robot motion: dynamic obstacle detection is relatively easy under static robot motion. The motion-dependent sensing mechanism of the event camera naturally only captures the event data corresponding to dynamic obstacles. Therefore, the background is automatically filtered out. (2) Dynamic robot motion: we will conduct further research for obstacle detection when robot is moving in the future.

In short, we present a framework to detect and track dynamic obstacles. The obstacle trajectory is computed and predicted based on tracked obstacle positions. The obstacle avoidance mechanism is proposed based on 2D obstacle position without using any additional sensors (e.g., depth camera) to provide the 3D location of dynamic obstacles. Our contributions are summarized as follows:

- We propose an event-based dynamic obstacle avoidance framework to detect and avoid dynamic obstacles

without using any additional sensors such as a depth camera.

- We set experiments with a real-world quadruped robot to validate our algorithm and show experimental results indicating the effectiveness of our proposed method.

## 2. Related Works

Vision-based obstacle avoidance is a challenging problem that has been studied extensively for a relatively long time period. We sort the well-established works in this section. The obstacle avoidance problem can be divided into several processes, including dynamic obstacle detection and tracking, trajectory prediction, and obstacle avoidance. In this section, we divide previous works into two categories: RGB camera-based methods and event camera-based methods.

### 2.1. RGB-based Methods

Viet et al. [23] utilized an RGB-based method to tackle the problem of obstacle avoidance by calculating a lookup table. Although their method enables the robot to successfully navigate in a simple environment, it is not generally applicable as it requires an additional calculation of the lookup table in every new environment and it will fail in complex environments containing high-texture grounds. Singh et al. [19] proposed a machine learning-based method to detect predefined objects and define them as obstacles. However, this limits the categories of the obstacle to the training dataset. Su et al. [22] estimated the position, velocity, and orientation of the quadrotor using an Unscented Kalman Filter (UKF) based loosely coupled visual-inertial fusion module. The visual measurements are obtained from an optical flow-based velocity estimator that utilizes measurements from a downward-facing camera and a sonar. Then a ball with LEDs is detected and tracked by applying a simple intensity threshold.

These methods are limited to slow-moving obstacles because they cannot respond to highly dynamic obstacles timely due to the low frame rate of conventional RGB cameras. Besides, the highly dynamic objects may also cause image blur which further increases the failure ratio of object detection and tracking.

### 2.2. Event-based Methods

To mitigate the limitation of the standard camera, Delbruck et al. [2] [3] utilized the event camera to build a fast self-calibrating goalie with a high update rate and low latency at low CPU load.

Neural network-based methods are also investigated to solve this problem. El Shair et al. [4] presented a hybrid ob-

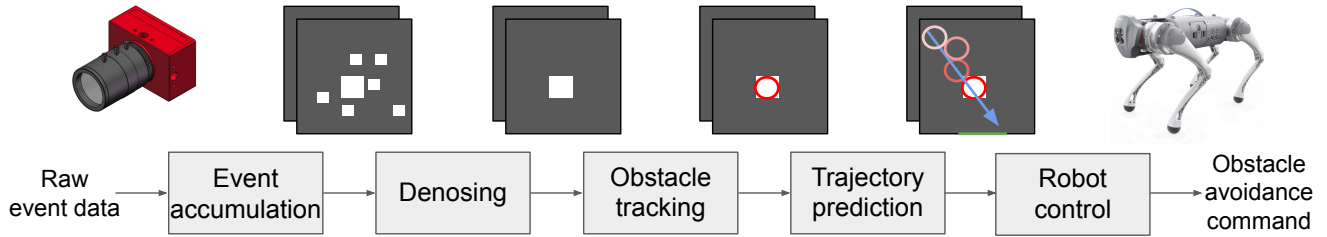


Figure 2. The overview of our detection, tracking, trajectory prediction, and avoidance system. The circles are the locations of the dynamic obstacle. The redder the circle, the closer it is to the current time. The array is the predicted trajectory direction.

ject detection and tracking approach that uses both frame-based camera data and event-based camera data. They first developed a learning-based method to classify from the frame camera data. Then, event masks are extracted from the event stream guided by the frame data to perform inter-frame tracking. Jiang et al. [9] fused an offline-trained detector and online-trained tracker to complement each other to provide an accurate object-tracking framework. Sanket et al. [16] validated their event-based moving object segmentation and avoidance algorithm on a quadrotor by using shallow neural networks. They trained a neural network to deblur the event frames and then passed the frames into another network that performs both segmentation and optical flow of independently moving dynamic objects. They further developed a control policy based on the estimation from the network. Ramesh et al. [13] [14] presented an object tracking method using a local sliding window technique for reliable tracking. Objects are initially detected using a global sliding window to find regions of interest (ROIs) which is only used during the initialization of an object or when the tracking fails to enable real-time performance. Finally, overlap success and center location error metrics were used for quantitative evaluation on a short indoor data sequence.

However, the neural network-based detecting and tracking result are usually time-consuming. For instance, the proposed resulting sensing latency was 60 ms [16]. Approaches that utilize the event data structure have been explored. To better exploit the topological structure, [1] [24] interpret the event cloud in the form of space-time graphs. In particular, Bi et al. showed that such compact graph representation requires less computation and memory than conventional Convolutional neural networks (CNNs) while achieving superior results to the state-of-the-art in various datasets. Zhou et al. [24] proposed a space-time event graph representation to jointly optimized event-based multi-motion segmentation and motion model fitting. An energy function is minimized to calculate the fitting motion model. Falanga et al. [5] reduced the latency to 3.5 ms by exploiting the temporal information of events to distinguish dy-

namic and static objects. They demonstrated the effectiveness of their moving obstacle avoidance algorithm also on a quadrotor. Mondal et al. [12] used graph-spectral clustering to detect moving objects from event data, then used silhouette analysis to automatically determine the number of moving objects in the data. Mitrokhin et al. [11] proposed a motion compensation model that enables the detection of objects in a scene by finding inconsistencies in the resulting model and then tracking them using a Kalman filter. They tested their approach on a dataset collected on a moving platform comprising several sequences of varying lighting conditions. The objects were labeled at the time instances of the captured RGB frames. They evaluated their tracking performance based on a success rate of the percentage of objects detected with at least 50% overlaps.

Although the above methods can achieve dynamic obstacle detection within milliseconds, they still need to estimate the robot’s ego-motion. In dynamic environments, using event data to estimate the poses of fast-moving robots is a very challenging problem.

### 3. Methodology

The overall pipeline of our framework is illustrated in Fig. 2. Our cascaded framework consists of multiple steps: event data accumulation, data denoising, obstacle detection and tracking, trajectory prediction, and robot control. We first accumulate the sparse events to support obstacle detection in the next few steps. We then implement an event image denoising algorithm to filter out background noise. Then, the dynamic obstacle detection algorithm is introduced to segment dynamic objects. Based on the sequence of dynamic obstacle positions, we calculate the obstacle direction and predict its future trajectory. The trajectory prediction is converted to  $y$ -axis velocity commands which are provided to the robot to avoid the dynamic obstacle.

The rest of this paper is organized as follows: Section 3.1 presents the event data processing. Then Section 3.3 discusses object detection and tracking methods used in this framework. In Section 3.4, we present the control strategy

---

**Algorithm 1** Obstacle detection, prediction, and avoidance

---

**Input: Event data****Output: Control command**

```
1:  $position = 0, cnt = 0$ 
2: for  $e$  in  $events$  do
3:   Continue if event is not supported by adjacent events
4:    $position.x += e.x$ 
5:    $position.y += e.y$ 
6:    $cnt += 1$ 
7: end for
8:  $position.x /= cnt$ 
9:  $position.y /= cnt$ 
10: Push position in a queue
11: if  $position.queue.size() > 20$  then
12:   Run RANSAC to find fitting line
13: end if
14: Predict obstacle position by extending the line
15: Calculate avoidance command
16: return Command
```

---

of robot avoidance.

### 3.1. Event Capture

An event  $e_i = (u_i, t_i, p_i)$  is triggered by the event camera at the position  $u_i = (x_i, y_i)$  at time  $t_i$  when the brightness change is larger/smaller than a given threshold  $\pm H$ . The polarity  $p_i$  describes the sign of the change. Due to the sparsity of the events, we capture the events by accumulating a stream of events in a fixed time window  $\Delta t$  of a few milliseconds. Specifically, we accumulated the events of  $\Delta t$  in the range of  $2000\mu s$  to  $8000\mu s$  while testing different detection algorithms. We then obtain a set of events  $E = \{e_i\}_{i=0}^{N-1}$  for detecting the obstacle and run obstacle detection when the accumulated number of events within a time window is larger than a threshold (set to 50 in the experiments). The intensity threshold is set to keep events that are triggered within 2 milliseconds (see Fig. 3). Under the assumption that triggered events should be close to each other, a background noise filter is then applied to only keep events that are supported by adjacent events.

### 3.2. Obstacle Detection

Obstacle detection can be divided into two situations depending on whether the robot is in static motion or dynamic motion.

Owing to the motion-dependent sensing mechanism of an event camera, dynamic obstacle detection is relatively simple in static situations. We can segment the foreground as the obstacle by averaging the previously filtered event data. Specifically, we utilize the time surface map as a representation of event data to segment the dynamic obstacles. A time surface is a 2D map where each pixel stores

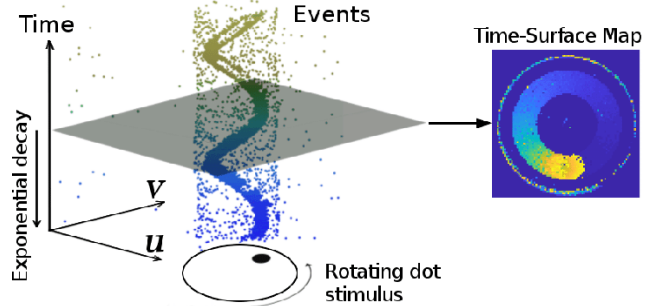


Figure 3. Left: output of an event camera when viewing a rotating dot. Right: Time- surface map at a time  $t$ , which essentially measures how far in time (with respect to  $t$ ) the last event spiked at each pixel  $x = (u, v)$ . The brighter the color, the more recently the event was generated. Figure adapted from [25].

the timestamp of the last spiked event (Shown in Figure X), then a time surface image can be generated where each pixel stores a grayscale value that decays with respect to the corresponding timestamp. The segmented detection results are shown in Fig. 1 and Fig. 6.

### 3.3. Obstacle Trajectory Prediction

Under the assumption that the legged robot and the obstacle move on the ground, we can estimate the trajectory without the depth camera. This will guarantee better performance because using a depth camera will sacrifice the low latency feature of the event camera and the motion blur of the depth camera will decrease the accuracy of the obstacle detection performance. To achieve collision-free obstacle avoidance, we only need to calculate and predict a 2D trajectory on the pixel coordinate. And if the dynamic obstacle is predicted to collide with the right side of the robot, the robot should move to the left side and vice versa.

A sequence positions of detected dynamic obstacle are utilized to track the trajectory and predict its future position. For obstacle trajectory calculation and prediction, we assume the speed of dynamic obstacles is high, so the trajectory should be a line. Under this assumption, the straightforward way is to predict the trajectory based on obstacle's previous position and current position. However, the position is updated in a high rate with a large noise, so the change of two adjacent positions cannot represent the direction of the trajectory.

To solve this issue, we utilize a Random sample consensus (RANSAC) algorithm to track the trajectory of the moving obstacles. Specifically, we store a sequence of positions of detected obstacles and use RANSAC to filter out outliers and compute the trajectory. The trajectory is shown in Fig. 1 and Fig. 6.



Figure 4. We kicked a number 5 soccer ball toward the robot and calculate the success rate of robot being able to dodge the ball.

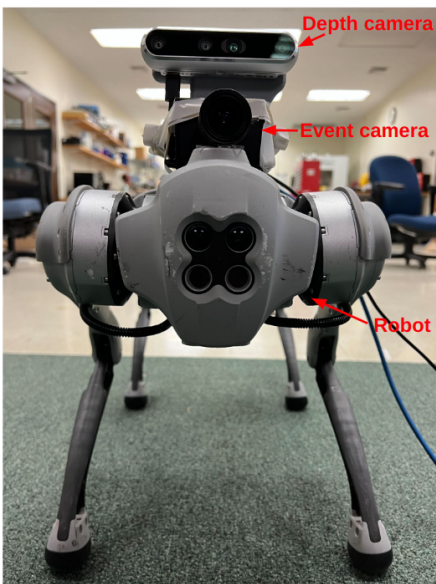


Figure 5. System configuration of the Unitree Go1 robot and the event camera. The Intel RealSense Depth Camera D455 was only used to visualize the robot-perspective viewpoint and not used for the obstacle avoidance framework.

### 3.4. Avoidance Behavior Control

The trajectory prediction output from the previous step is converted to a velocity command for the robot. Based on the prediction, we first select the action to avoid the dynamic obstacle among a predefined finite set of actions  $A = \{left, right, steady\}$ . For example, when the pre-

diction of the obstacle trajectory lies on the left side of the robot as shown in Fig 1, we command the robot to move right and vice versa. Specifically, the robot is commanded to move in 0.7 m/s for 1 s along the coronal plane (side-ways) unless the *steady* action is selected. Here, we ignore the triggered events during the avoidance movement and set a steady time as 2 s for the robot to stay still after the avoidance command to reduce the chance of detecting events based on ego-motion. We utilize the Go1 robot’s built-in motion controller developed by Unitree Robotics [15] to execute the high-level control of the velocity, body height, orientation, and gait type of the robot.

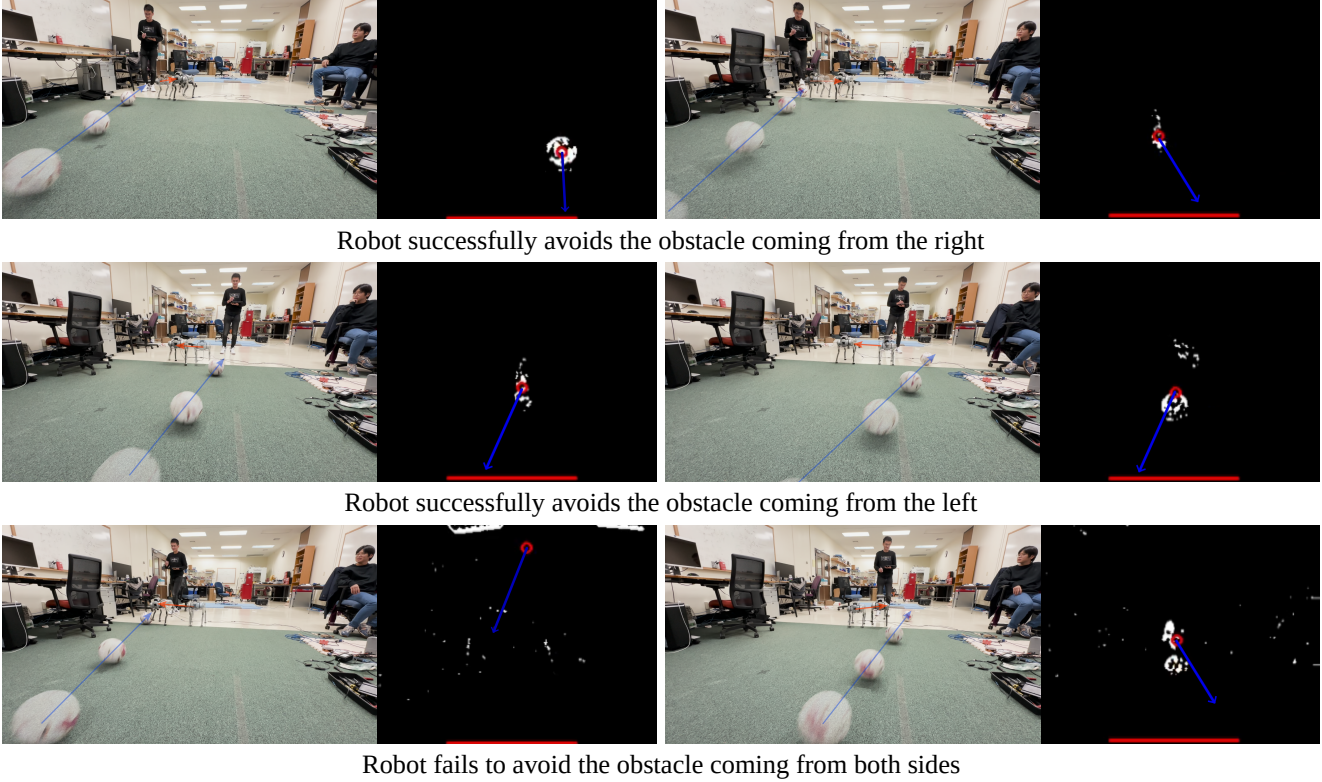
## 4. Experimental result

In this section, we provide details on the setup for the dynamic obstacle avoidance experiment in Section 4.1 and demonstrate the results in Section 4.2.

### 4.1. Experiment setup

We tested our dynamic obstacle avoidance framework using the Go1 quadruped robot. The IniVation DVXplorer Lite event camera is mounted on top of the *head* of the robot facing forward. We additionally installed an Intel RealSense Depth Camera D455 on top of the robot for recording purposes. Note that our algorithm does not require any other information such as depth other than events. The hardware platform including the robot and the cameras are well-depicted in Fig. 5.

We created an experimental environment as shown in Fig. 4 to have the quadruped robot avoid a soccer ball kicked by one of the researchers. In each trial of the experiment,



Robot successfully avoids the obstacle coming from the right

Robot successfully avoids the obstacle coming from the left

Robot fails to avoid the obstacle coming from both sides

Figure 6. Top row: the robot dodged the obstacle coming towards its right side. Middle row: the robot dodged the obstacle coming towards its left side. Bottom row: two failure cases where the algorithm didn't make the right prediction due to noise.

Table 1. Obstacle avoidance experiment trials.

Direction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Right	✓	✓	✗	✓	✓	✗	✓	✓	✓	✗	✗	✓	✗	-	✓	✓	-	-	✓	✓	-
Left	✓	✗	✗	✗	✓	✓	✗	✗	✓	✓	✗	✗	✓	✗	✓	-	✗	✗	✓	✓	✗

Obstacle direction	Detection	Avoidance*
Right	21/21	16/21
Left	21/21	10/21

\* Among the 16 & 10 success avoidance when the obstacle was approaching to the Right & Left side of the robot, there are 4 & 1 times that the robot did not react to the approaching ball which is predicted not to collide with the robot.

Table 2. Detection and avoidance success rate.

the kicker sat on a chair and kicked the ball near the robot. The kicker minimized movement while kicking the ball to prevent the event camera to capture the events generated by the kicker. The robot was re-positioned to the initial position after each trial.

## 4.2. Results

In order to quantify the performance of our obstacle avoidance algorithm, we count the number of successes of avoidance and report the success rate. If the robot fails to avoid the ball and collide, we count the trial as a failure which is similar to counting the number of collisions, one of the commonly used metrics in navigation [17, 21].

In total, 21 kicks were aimed toward the right side (robot perspective) of the robot and another 21 kicks were aimed toward the left side. We show the obstacle avoidance sequence in Table 1. Our dynamic obstacle avoidance algorithm enabled the quadruped robot to avoid the approaching ball 16 out of 21 trials on the right side and 10 out of 21 trials on the other side as shown in 4.

Out of 42 total kicks, the robot has achieved 100% detection rate and it successfully dodged the moving ball 26

times. We have visualized some of the success and failure cases on Fig. 6. Due to the high responsiveness of event cameras, our algorithm is able to accurately detect the fast moving obstacle throughout the experiment. However, event cameras also suffer from noise in the environment. Failure cases are all due to the noise being picked by the event camera which results in wrong prediction of the trajectory.

## 5. Conclusion and Discussion

We propose an obstacle avoidance framework for a quadruped robot to avoid highly-dynamic obstacles based on events. Our event-based framework enables detection and trajectory prediction of high-speed obstacles within 2 milliseconds without estimating ego-motion. Nevertheless, the agility of the legged robot limits the time to avoid it. To quantify the performance of our algorithm, we measure the success rate of avoiding a ball kicked by a researcher and achieved 62% success of avoidance.

## 6. Future Work

We plan to improve our algorithm's robustness to noise in our future work. We expect to implement learning-based filters to significantly decrease the noise level and consequently improve the success rate of avoidance. We also plan to implement obstacle detection even when the robot is in motion. Other researchers have been estimating the ego-motion to compensate for the event data to segment the dynamic obstacles [8]. However, this requires accurate pose estimation, which is extremely challenging with event data under highly dynamic robot motion in a dynamic environment. We will release the limitation of ego-motion estimation by separating the dynamic obstacles based on local optical flow consistency. We also plan to leverage sparse-to-fine optical flow algorithms to provide high-speed and accurate dynamic obstacle detection.

To gain further improvement, we will utilize time surface images to fully interpret temporal information for future work. We also plan to advance our algorithm to plan paths under conditions having multiple approaching dynamic obstacles while navigating.

## References

- [1] Yin Bi, Aaron Chadha, Alhabib Abbas, Eirina Bourtsoulatze, and Yiannis Andreopoulos. Graph-based object classification for neuromorphic vision sensing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 491–501, 2019. 3
- [2] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor. *Frontiers in neuroscience*, 7:223, 2013. 2
- [3] Tobi Delbruck and Patrick Lichtsteiner. Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *2007 IEEE international symposium on circuits and systems*, pages 845–848. IEEE, 2007. 2
- [4] Zaid El Shair and Samir A. Rawashdeh. High-temporal-resolution object detection and tracking using images and events. *Journal of Imaging*, 8(8), 2022. 2
- [5] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020. 3
- [6] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020. 2
- [7] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *CoRR*, abs/1702.03920, 2017. 1
- [8] Botao He, Haojia Li, Siyuan Wu, Dong Wang, Zhiwei Zhang, Qianli Dong, Chao Xu, and Fei Gao. FAST-dynamic-vision: Detection and tracking dynamic objects with event and depth sensing. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3071–3078. IEEE. 7
- [9] Rui Jiang, Xiaozheng Mou, Shunshun Shi, Yueyin Zhou, Qinyi Wang, Meng Dong, and Shoushun Chen. Object tracking on event cameras with offline–online learning. *CAAI Transactions on Intelligence Technology*, 5(3):165–171, 2020. 3
- [10] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128× 128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *IEEE Journal of Solid-State Circuits*, 43(2):566–576, 2008. 2
- [11] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018. 3
- [12] Anindya Mondal, Shashant R, Jhony H. Giraldo, Thierry Bouwmans, and Ananda S. Chowdhury. Moving object detection for event-based vision using graph spectral clustering. *CoRR*, abs/2109.14979, 2021. 2, 3
- [13] Bharath Ramesh, Shihao Zhang, Zhi Wei Lee, Zhi Gao, Garrick Orchard, and Cheng Xiang. Long-term object tracking with a moving event camera. In *Bmvc*, page 241, 2018. 3
- [14] Bharath Ramesh, Shihao Zhang, Hong Yang, Andres Ussa, Matthew Ong, Garrick Orchard, and Cheng Xiang. e-tld: Event-based framework for dynamic object tracking. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(10):3996–4006, 2020. 3
- [15] Unitree Robotics. Unitree robotics go1 robot, 2022. 5
- [16] Nitin J. Sanket, Chethan M. Parameshwara, Chahat Deep Singh, Ashwin V. Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodge: Embodied AI for high-speed dodging on A quadrotor using event cameras. *CoRR*, abs/1906.02919, 2019. 3
- [17] Mingyo Seo, Ryan Gupta, Yifeng Zhu, Alexy Skoutnev, Luis Sentis, and Yuke Zhu. Learning to walk by steering: Perceptive quadrupedal locomotion in dynamic environments. *arXiv preprint arXiv:2209.09233*, 2022. 1, 6

- [18] Kavan Singh Sikand, Sadegh Rabiee, Adam Uccello, Xuesu Xiao, Garrett Warnell, and Joydeep Biswas. Visual representation learning for preference-aware path planning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 11303–11309. IEEE, 2022. 1
- [19] Rajmeet Singh, Tarun Kumar Bera, and Nizar Chatti. A real-time obstacle avoidance and path tracking strategy for a mobile robot using machine-learning and vision-based approach. *SIMULATION*, page 00375497221091592, 2022. 2
- [20] Maks Sorokin, Jie Tan, C Karen Liu, and Sehoon Ha. Learning to navigate sidewalks in outdoor environments. *IEEE Robotics and Automation Letters*, 7(2):3906–3913, 2022. 1
- [21] Maks Sorokin, Jie Tan, C. Karen Liu, and Sehoon Ha. Learning to navigate sidewalks in outdoor environments. *IEEE Robotics and Automation Letters*, 7(2):3906–3913, 2022. 6
- [22] Kunyue Su and Shaojie Shen. Catching a flying ball with a vision-based quadrotor. In *International Symposium on Experimental Robotics*, pages 550–562. Springer, 2017. 2
- [23] Chau Nguyen Viet and Ian W Marshall. Vision-based obstacle avoidance for a small, low-cost robot. *ICAR 2007*, 2007. 2
- [24] Yi Zhou, Guillermo Gallego, Xiuyuan Lu, Siqi Liu, and Shaojie Shen. Event-based motion segmentation with spatio-temporal graph cuts. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 3
- [25] Yi Zhou, Guillermo Gallego, and Shaojie Shen. Event-based stereo visual odometry. *IEEE Transactions on Robotics*, 37(5):1433–1450, 2021. 4