Autonomous Navigation with a Monocular Event Camera

Chao Ni
University of Zurich
chao@ifi.uzh.ch

Matthias Müller
Intel
matthias.mueller@intel.com

Davide Scaramuzza
University of Zurich
sdavide@ifi.uzh.ch

Abstract

Autonomous navigation is one of the most fundamental problems in robotics. Most existing works use RGB cameras to perceive the environment, build a map, plan a trajectory, and track it with a controller. In this work, we propose a novel pipeline for autonomous navigation that leverages the stream from a single event camera, a bio-inspired sensor that captures pixel-level brightness changes at microsecond resolution. We present an end-to-end planner that takes these events as input and predicts a collision-free trajectory, which is then tracked by a model predictive controller. The training data is entirely collected in simulation; we utilize imitation learning to distill the knowledge of an artificial expert to a network policy. We show that our pipeline works with a perceptual latency of only 1*ms. The trained planner can be transferred zero-shot to a real quadrotor that is able to plan complex trajectories based on only a single event camera. The proposed approach achieves superior performance compared with image-based alternatives, reducing the perceptual latency by up to 30 times, opening up new applications for safe agile robotics.*

1. Introduction

Autonomous robot navigation has made tremendous strides over the past years. Removing the need for humans to tele-operate, monitor, or manually intervene will unlock massive applications in industry. While most existing works use RGB cameras to perceive the environment, the perception latency is the limiting factor to increase the agility (and thus the productivity) of autonomous mobile robots. Standard cameras indeed suffer from a *bandwidth-latency trade-off*: at high speeds, they require a high frame rate to reduce perceptual latency, but this introduces a significant bandwidth overhead for computationally-constrained robots; reducing the frame rate reduces the bandwidth requirements but at the cost of missing important scene dynamics (thus reducing the safety and agility of the system) due to increased perceptual latency. Besides, RGB cameras also suffer from motion blur at high speeds in dimly lit environments.

Event cameras are novel, bio-inspired vision sensors

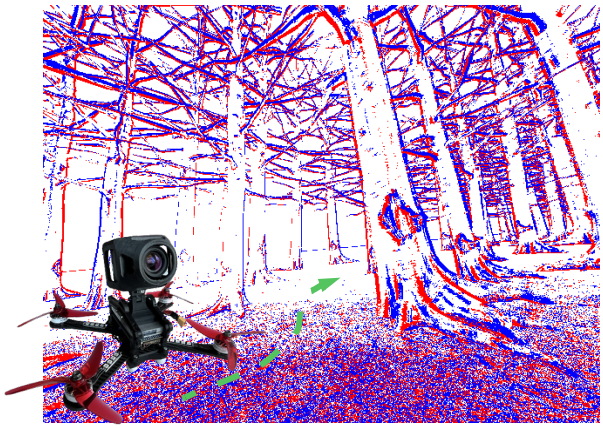


Figure 1. A novel pipeline for autonomous navigation using only the data stream from a single event camera. The image shows an event visualization of the perceived forest environment.

that have a different working principle than standard cameras. Instead of capturing images at a fixed rate, event cameras output per-pixel brightness changes at microsecond resolution and have appealing properties including sub-millisecond latency, high dynamic range, no motion blur, and low energy consumption. These properties are ideal to reduce the perceptual latency of a robotic system and, thus, achieve increased safety in high-speed scenarios, like the one in Fig. 1. For further details about event cameras, we refer the reader to [20]. In this work, we exploit event cameras for autonomous navigation of quadrotors (i.e., quadcopter drones).

Traditional methods tackle the navigation problem by separating the pipeline into perception, planning, and control modules [19, 66]: the robot first extracts appropriate perception abstractions (e.g., an estimate of the robot position, velocity, and orientation plus a local map), reasons about them, then plans a collision-free trajectory, and finally computes the control commands to execute this trajectory. This modularity enables the overall interpretability of the system. However, the sequential components add extra latency, making high speed navigation more challenging [16].

Learning-based approaches have recently been used to directly map the perception information to control outputs

	Method	Sensor	Resolution	Visual Abstraction	Perception Latency (ms)		
					Sensor	Processing	Total
(a)	FastPlanner [66] [64]	LiDAR	-	3D map	-	-	100
(b)	Reactive [19] [36]	Depth sensor	[160,120]	pointcloud	33.3	13.8	47.1
(c)	High-speed Planner [36]	RGB camera	[640,480]	SGM depth	33.3	2.74	36.0
(d)	RECON [56]	RGB camera	[160,120]	image	33.3	0	33.3
(e)	**Ours**	Event camera	[640,480]	Event tensor	1.00	1.79	**2.79**

Table 1. **Comparison of different vision-based navigation approaches.** Entries with - indicate not available or not applicable. For the column of perception latency, we report the time required for each visual abstraction to be generated, therefore including the sensor latency and the processing latency. For the RGB camera and depth sensor, the sensor latency depends on the actual hardware. Take RealSense D415 as an example, the RGB frame rate is 30 Hz and the depth sensor rate up to 90 Hz. Conversely, our approach, which takes only raw events as input and processes them into an event tensor, consumes only around 2.79ms per update.

[36, 39, 51, 52, 56]. By leveraging deep networks, the planner has been shown to achieve better generalization abilities than traditional methods and can be released zero-shot in the wild [36]. Learning-based methods are usually data-hungry. However, collecting data from the real world may cause damage to the robotic platform and takes months or even years of manual annotations [56, 57]. To mitigate this issue, other works use synthetic data [36, 38, 52]. The policy is purely trained in simulation, but needs to be deployed in the real world. This boils down to learning an effective representation of the observation such that the distribution shift from simulation to reality can be well mitigated. These requirements, along with a constrained computation resource onboard, pose major challenges for autonomous navigation.

While there exist numerous papers and datasets on the use of standard RGB cameras or depth sensors for downstream robotics tasks [19, 36, 39, 51, 52, 56, 57, 66], research on robot navigation using event cameras is limited. RGB cameras are still the main perception modality and the perception latency is limited by the nature of the frame-based sensor. For example, the RealSense D415 perceives the environment at 30 Hz, while an event camera can provide updates up to thousands Hz [20]. So far, the use of event cameras in mobile robotics has been demonstrated in high-speed state estimation [62] or dodging fast moving objects with quadrotors [17, 53]. However, utilizing event cameras for autonomous navigation in the wild remains unexplored.

To approach this problem, we propose a novel pipeline for autonomous navigation that leverages the data stream from a single event camera. The policy is trained purely in simulation and transferred zero-shot to the real world. We utilize Imitation Learning (IL) [44] to train the navigation policy: the demonstrations are collision-free trajectories with given boundary conditions; the policy takes a stream of events and directly predicts the trajectory in terms of way points. The trajectory is then executed by an optimal controller to increase the generalizability of the system. In addition, we further study the importance of different components for training a successful quadrotor navigation planner. Our findings show that state information (i.e., an inertial sensor) is not necessary for predicting a collision-free trajectory, and a 1ms time window of events from a single event camera is sufficient to plan the trajectory.

Contributions: In summary, we develop the first end-to-end approach that utilizes a single event camera and predicts collision-free trajectories for agile quadrotor navigation. The proposed approach achieves superior performance compared with frame-based alternatives, greatly reducing the perception latency by up to 30 times. This opens up the potential of event cameras being used in other high-speed, safety-critical robotic tasks such as autonomous driving. In addition, we implement a series of ablation experiments and evaluate different modalities for perception-based navigation, and create a common benchmark for navigation tasks.

2. Related Work

2.1. Visual Navigation

Modularized System. Traditional methods separate visual navigation into perception, planning, and control modules [3, 12, 19, 54, 66]. Zhou *et al*. [66] built an environment map from the perceived point cloud and solved an optimization problem for safe path planning. Florence *et al*. [19] forgo building the map and propose to operate directly on the latest depth information for planning, therefore reducing the latency by a large margin. However, a sequential pipeline still neglects the interactions between different components and errors compound [65].

Data-driven Approaches. Recent works in vision-based navigation include many learning-based approaches [8, 9, 27, 36, 39, 51, 51, 52, 56]. These approaches directly map the visual observation to the control output. Among them Imitation Learning (IL) [44] and Reinforcement Learning (RL) [60] are the two main paradigms. IL learns the planner from expert demonstrations such as human experts [39, 51], samplers [36, 55], and privileged teacher networks [8], while RL encodes the navigation problem as a Markov Decision Process (MDP) and guides the agent with task-specific rewards

[9, 52, 56]. Meng *et al*. [39] use a few demonstrations to build the environment map and encode the visual observation to a latent space, the robot navigates through targeting the most similar demonstrated image as waypoints. Similar to our work, Loquercio *et al*. [36] proposed to learn the visual navigation planner by imitating a Monto-Carlo sampler sampling from a global collision-free trajectory. We simplify the network by removing the state and replacing the traditional vision sensor with an event camera, reducing the perception latency by a huge margin.

Some works collect training data directly in the real world [56, 57]. Shah *et al*. [56] proposed to navigate by exploring the environment and building the map simultaneously, using a goal-conditioned latent model, training with large scale offline real world data. The dataset is extended in [57], and a general navigation model is trained by combining datasets from different platforms. However, all above mentioned method use RGB or depth images as visual information. Tab. 1 compares the sensors and latency of different approaches. In our work, we use an event camera and imitation learning to tackle the navigation problem.

2.2. Event-based Vision

Event Representation. Events are usually transformed to intermediate representations to facilitate the feature learning from events. Alternative representations include time surface [1, 33, 37], voxel grid [2, 61, 63] and reconstructed images [50]. We refer to [20] for more details. In this work, we utilize the voxel grid representation to sum up positive and negative event polarities; we construct a 3-channel event tensor by adding event count as third channel.

Event-based Depth Estimation. Depth as an intermediate representation is important for robotics applications [7, 67]. Works in depth prediction with event cameras have experienced a surge in past years, both in the optimization-based domain [31, 46, 48, 49, 68, 69] and the deep learning domain [26, 71]. Zhu *et al*. [71] encoded the events with a spatial-temporal voxel grid and learned the motion information from an unsupervised fashion. Hidalgo *et al*. [26] used a monocular event camera to predict the depth and used both synthetic and real data for training. However, we show in the experiments that depth is not necessary for training a navigation policy. We leverage the data stream from a single event camera and train the planner directly from the event tensor without having an intermediate depth map.

Datasets. Compared with abundant image datasets, datasets based on event cameras are scarce. Available datasets include [10, 13, 18, 23, 30, 45, 58, 70]. Among them Zhu *et al*. [70] proposed the first dataset with synchronized stereo event cameras, with accurate ground truth depth and pose. Gehrig *et al*. [23] address the need for large scale outdoor stereo event camera datasets. However, datasets for autonomous navigation with event cameras do not yet exist. In our work, we resort to simulation to create the navigation dataset; we will publicly release it upon acceptance. We render high temporal resolution video in Flightmare [59] and use ESIM [47] to simulate events.

Event Camera for Robotics. Event cameras have recently attracted attention in the robotics community [20]. Early works using events for robotics focus on low-dimensional tasks [4, 6, 11, 14, 15, 21, 24, 25, 40, 41, 43]. Event cameras have also been demonstrated in more complex robotics platforms [17, 42, 53, 62]. Evasive maneuvers with quadrotors are presented in [17, 42, 53] with the help of event cameras. To dodge fast moving obstacles with event cameras, Falanga *et al*. [17] proposed to segment the moving obstacles from a single stream of events and fly the quadrotor into a collision-free direction. Along the same line of work, Sanket *et al*. [53] leveraged deep learning to estimate the ego-motion, optical flow and segmentation simultaneously. Our work extends the usage of event cameras to autonomous navigation and utilizes neural networks for achieving collision-free planning.

3. Method

An overview of our approach is shown in Fig. 2. The main building block of our method is a visual planner, a neural network that takes as input perceptual data coming from a sensing device, in our case a monocular event-based camera, and directly outputs waypoint positions, while also avoiding potential collisions with obstacles in the way. Contrary to previous work, our method does not predict intermediate representations, such as a depth map, a feature which we empirically validate in Sec. 4.3 where we show that it provides the best tradeoff between accuracy and latency. Such an end-to-end planner, combined with the low perceptual latency of the event-based camera, reduces the processing latency and enables agile navigation by shortening the reaction time. We present the detailed components of our approach in the following sections.

3.1. Visual Planner

The visual planner Φ takes observations $\mathcal{X}$ as input and outputs a sequence of waypoints $\mathcal{T} \in \mathbb{R}^{N\times 3}$. These waypoints are expressed in the local body frame of the drone. We write this input-output relation as

$$\mathcal{T} = \Phi(\mathcal{X}) \tag{1}$$

While this formalism is general, and could be used for any sensor modality, we focus here on using events from a monocular event camera. Later in Sec. 4.3, we will show that this sensor is actually sufficient for successful, lightweight planning, without the need for other sensor modalities such as depth $\mathcal{D}$, images $\mathcal{I}$ or the drone state $\mathcal{S}$.

Event cameras output a stream of events (t_i, x_i, y_i, p_i), each consisting of time, position, and polarity. An event

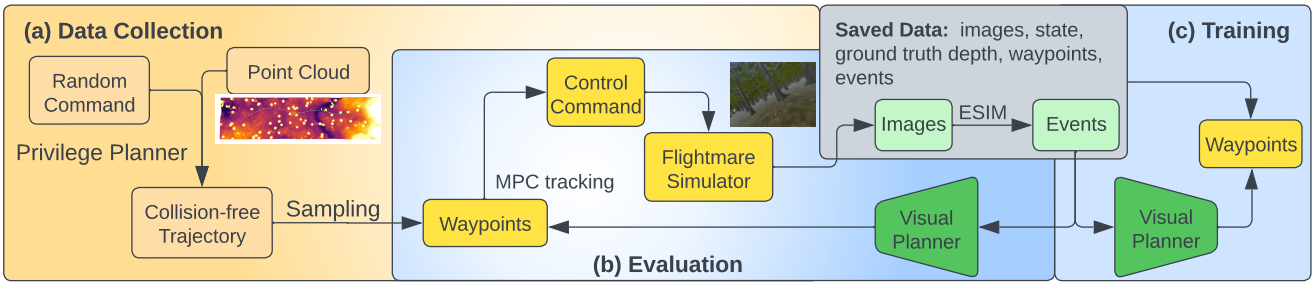


Figure 2. System Overview. During data collection, the privilege planner [35] has access to the ground truth point cloud of the environment and can plan a global collision-free trajectory given a random command. From the collision-free trajectory, sequences of waypoints are sampled and tracked by a Model Predictive Controller (MPC). The control command is sent to Flightmare [59]. In the data collection phase, we collect high frequency images, ground truth depth map, quadrotor state, and the tracked waypoints to the training buffer. To simulate events in Flightmare, we use ESIM [47]. The perception backbone is a pretrained MobileNet-V3 and the network is trained in a supervised fashion to predict the waypoints.

is triggered at time t_i if the brightness change at (x_i, y_i) exceeds a threshold $\pm C$, i.e. when,

$$p_i[L(\mathbf{x}_i, t_i) - L(\mathbf{x}_i, t_i - \Delta t_i)] > C \tag{2}$$

The observations are sequence of raw events $\mathcal{E} = \{e_i\}_{i=1}^{N}$. Before ingesting events, we first map them to a dense "image-like" representation, a practice which allows us leverage existings powerful neural network architectures, and pretraining. We opt for a dense grid-like representation, due to its ease of use and high performance on down-stream tasks [22, 71] with dimensions $B \times H \times W \times 3$, where B is the number of temporal bins and 3 are the per-pixel features. The first two features are computed by assigning events within a time window ΔT into B bins by:

$$E(x, y, t, p) = \sum_{\substack{x_i=x,\\ y_i=y,\\ p_i=p}} p_i \max(0, 1 - |t - t_i^*|), \tag{3}$$

where $t_i^* = \frac{B-1}{\Delta T}(t_i - t_0)$ is the normalized timestamp and $p \in \{-1, 1\}$. We finally compute the third channel by computing the total number of events within the time interval ΔT. We implement the planner as a MobileNet-V3 [29] pretrained on ImageNet. We empirically show in Sec. 4.1 that such a simple solution is enough for planning trajectories with remarkable obstacle avoidance performance and high level of robustness to challenging motion and lighting conditions. Fig. 2(c) shows the visual planner that only takes an event tensor as input.

3.2. Learning in simulation

To train the visual planner Φ, by forcing it to imitate an expert planner. This planner is another algorithm that generates waypoints $\mathcal{T}$ but can do so with privileged information. In our case, similar to [36] we base our expert on the method in [35].

Expert and Training Setup. The expert optimizes for a global collision-free trajectory $\mathcal{T}_G$ given a boundary condition, and sends the trajectory to the Flightmare [59] simulator. Flightmare uses the Unity Engine as its rendering backend and can generate images with high frequency. The data collection is performed in a virtual forest environment with a size of 100m by 100m and trees distributed according to a Poisson distribution [5] with a minimum distance between any two trees as 3 meters. A visualization of a collision-free trajectory generated by the demonstrator is provided in Fig. 3. These trajectories are collected in a training buffer during a data collection phase, and then used in a later stage to train the visual planner so that it learns to imitate the privilege planner but without having access to global information. We randomly spawn the expert quadrotor at the beginning of the trajectory rollout, and set a random reference direction. At each planning step t_p, we sample 1 second of waypoints $\mathcal{T}_{g,\text{exp}} \in \mathcal{R}^{N\times 3}$ from the collision-free trajectory $\mathcal{T}_G$ and track the trajectory with the MPC. We send the sampled waypoints to the MPC at 20 Hz and the low-level controller runs at 100 Hz. We also log this trajectory for later training the visual planner. In total, we collect data from 300 trajectories, each lasting 6 seconds, resulting in 30 min of flight. Collection at 20 Hz results in 36'000 samples.

During rollout, we also collect the quadrotor's current state information, images, ground truth depth, and events from a stereo camera with a field of view (FOV) of 90 degrees, tilt angle of 30 degrees, and a baseline of 10 cm. Each training data sample s_i in the buffer thus has the following elements: quadrotor state $\mathcal{X}$, stereo images $\mathcal{I}_l, \mathcal{I}_r$, ground truth depth $\mathcal{D}$, sampled waypoints $\mathcal{T}$, and simulated events $\mathcal{E}_l, \mathcal{E}_r$.

Visual Planner. The visual planner does not have access to the global pointcloud, and thus needs to be able to imitate the expert using only events. To enable this, we use ESIM [47] to generate events from video with a high tem-

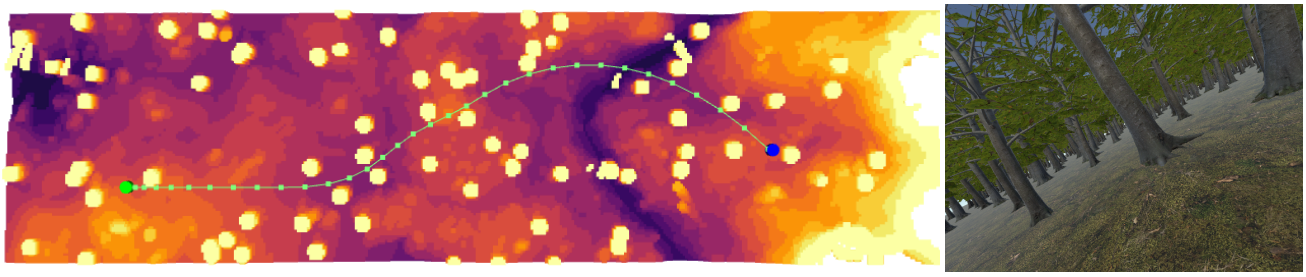


Figure 3. Visualization of the training environment. On the left is the visualization of trajectory planned by the demonstrator [35]: the planner has access to the ground truth point cloud and can plan a collision-free trajectory given the boundary conditions. On the right is a camera image showing the density of the environment.

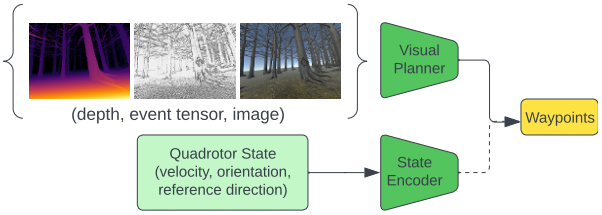


Figure 4. Architecture of the planner. Note that we use a dashed line connecting the state source. We ablate on the state information and show comparison in Sec. 4.

poral resolution. Using the collected labels $\mathcal{T}_{\text{exp}}$ from the expert, we train the student to imitate those labels from events alone, by minimizing the following objective:

$$\theta^* = \arg\min_{\theta} \sum_i \|\mathcal{T}^i_{\text{g,exp}} - R\Phi(\mathcal{E}_i)\|_2^2, \tag{4}$$

where i is a training sample index, and θ are the parameters of the visual planner neural network. The transform R maps the trajectory output by our visual planner to the global frame.

We train the planner using the Adam optimizer [32] with a learning rate of $3 \cdot 10^{-5}$, a batch size of 32 and without data augmentation.

3.3. Model Predictive Control (MPC)

The sampled trajectory $\mathcal{T}_g$ is tracked by an model predictive control (MPC). To compute the control command, the MPC solves the following optimization problem in a receding horizon fashion:

$$\underset{\boldsymbol{u}(\cdot)}{\text{minimize}} \sum_{i=0}^{N-1} ||\boldsymbol{x}_i - \mathcal{T}_{g,i}||^2_{\mathbf{R}} + ||\boldsymbol{u}_i||^2_{\mathbf{Q}}, \tag{5}$$

while satisfying system constraints. The variables $\boldsymbol{x}_i$ and $\boldsymbol{u}_i$ are the quadrotor state and input at step i. The trajectory $\mathcal{T}_g$ is either sampled from the global collision-free trajectory $\mathcal{T}_G$, or predicted by the visual planner $\mathcal{T}_g = R\Phi(\mathcal{E})$. The sampled trajectory has a horizon of $N = 10$, with a total duration 1 second. At each time step, the MPC computes a sequence of actions $[\boldsymbol{u}(0), \boldsymbol{u}(1), ...\boldsymbol{u}(N-1)]$, but only the first command is executed.

4. Experimental Results

4.1. Simulation Experiments

To ablate the importance of different perception modalities and planner states, we conduct a series of experiments. For simplicity, we denote E as event representation, I as image and S as quadrotor state. In this section, we aim to answer the following question: **Q1:***What kind of perception modality is most suitable for agile navigation?* From this we ask ourselves more specifically: **Q1.1:** *Do we need a stereo camera, e.g. depth information, for agile navigation?* **Q1.2:** *Do we need state information for planning trajectories?* Finally, we investigate the question: **Q2:** *What's the optimal event window size and planner update frequency to achieve the best planning performance?* To investigate these questions, we collect several sensors data in our simulator, including quadrotor state, depth maps, and images, which is discussed next.

Test Setup. We evaluate the navigation task in an evaluation environment in Flightmare with the same size as the training environment, but with trees spawned following a Poisson distribution [5] with density of $\sigma = 25^{-1}$ tree m^{-2}. During rollout, initial position and orientation of the quadrotor are randomly chosen and remain fixed for all experiments.

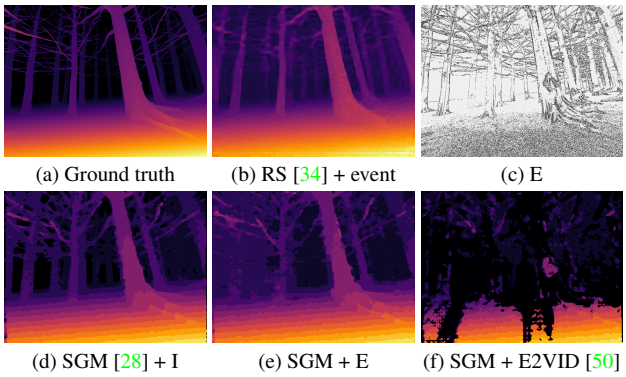
4.2. Metrics and Baselines.

Metrics. We pick 100 random starting points to fly the quadrotor. A rollout is considered successful if the quadrotor flies forward 20 meters without crashing into a tree. We report the success rate among all these attempts. Now we list the baselines considered during the experiments.

Direct Baseline (DB): This baseline does not use a neural network and blindly navigates the quadrotor straight, i.e. along a constant reference direction.

	Method	epe (px)	valid %	inference (ms)
(a)	RS + E	**0.47**	**99.8**	39.6
(b)	SGM + E	4.55	96.2	2.74
(c)	SGM + I [36]	4.97	87.3	2.74
(d)	SGM + E2VID [50]	3.98	11.5	-

Table 2. **Deep network provides better depth estimation than SGM.** Depth estimation with different approaches and modalities. The results are reported for inverse depth (disparity). In the Method column, we denote E for events, I for images, RS for RAFT-Stereo [34], E2VID for reconstructed images from events. We report with RS + E if the disparity is estimated by RAFT-Stereo and events. We use the standard evaluation matrix: end-point-error (epe), and valid percentage of pixels that have positive disparity prediction.



(a) Ground truth (b) RS [34] + event (c) E

(d) SGM [28] + I (e) SGM + E (f) SGM + E2VID [50]

Figure 5. Different Perception Modalities. We visualize different depth information (a, b, d, e, f) and event representation (c). In addition, we also include the RGB image as another modality in our experiments. Notation: We denote E for events, I for images, RS for RAFT-Stereo [34], E2VID for reconstructed images from events. We report with RS + E if the disparity is estimated by RAFT-Stereo and events.

Blind Planner: The blind (B) planner represents the planner that is trained on images padded to zero, and the quadrotors state detailed in Fig. 4. It thus learns a minimum of navigation capabilities from its own state.

Depth-based Planners: Computer vision can provide useful intermediate representation for robotics [67] that can help facilitate training the downstream policy [7]. Among many intermediate representations, depth is one of the most important ones. But is this also true when the perception modality is events? To investigate the importance of depth information, we introduce three baselines for getting depth from the event camera: (i) ground truth (GT); (ii) our adaptation of RAFT-Stereo [34] (RS) that maps stereo events to depth; (iii) semi-global matching [28] (SGM) that finds matching features between the left and right event tensors. For each one of these baselines, we train them once with state (+S) and once without (no symbol). To make a better comparison of these baselines, we additionally evaluate them in their ability to reconstruct accurate depth, and report these results in Tab. 2 and visualize them in Fig. 5. We can see that there is a clear tradeoff between accuracy and latency, both of which are crucial for successful navigation in real-time.

Vision-based Planners: To get a comprehensive understanding of the informativeness of different perception modalities for depth prediction, we compare two perception sources: event representation (Mono E) and RGB image (Mono I). Again we differentiate between planners that use state and do not use state with (+S).

4.3. Results

Accurate depth map does not improve the performance. The detailed performances are shown in Tab. 3. In Tab. 3 we see that the success rate between modalities only ranges between 0.67 (SGM+I+S) and 0.73 (GT+S). This indicates that the perception modality does not have a major impact on the behavior of the planner. We find that event-based visual inputs (Mono E+S) perform as well as ground truth depth maps. We can see that the accuracy of the depth does not play a major role, since highly accurate depth maps from (RS+E+S) and less accurate depth from (SGM+E+S) achieve the same success rate. Surprisingly, we find that even changing the depth map method from RS to SGM only has a minor impact on performance. We therefore conjecture that depth is only used to a minor extent.

State information prevents the planner learning from exteroception. To test this hypothesis we even remove the depth information in the test environment (h) and find that the performance only drops by 3%. This confirms our hypothesis, and thus leads us to believe that the planner focuses primarily on the state to plan its trajectory. Since the waypoints have a high correlation with the state, e.g. quadrotor velocity, the planner tends to exploit the state and ignores the perception information. However, this is unsustainable especially in more complex environments, and we thus want the planner to focus more on its visual inputs. We therefore repeat the experiments in Tab. 3 without state, and report those results in Tab. 4. In this setting we see that our planner actually uses perception, because the planner trained on (RS+E), when deprived of its inputs, experiences a performance drop of 47.9%, from 73% to a very low success rate of 38%. We find that this simple modification improves the success rate of almost all baselines by between 1% to 5%.

Using events improves success rates over using images. We also compare the difference between images and events. Motivated by [36], we compare SGM on stereo cameras for both images and events. Fig. 5 provides a visualization comparing these two modalities. Comparing (f) and (g) in Tab. 3 and Tab. 4 shows that using monocular events

	Abstraction	Train	Evaluate	Success
(a)	depth	RS + E + S	RS + E + S	0.71
(b)		RS + E + S	SGM + E + S	0.67
(c)		SGM + E + S	SGM + E + S	0.70
(d)		SGM + I + S	SGM + I + S	0.67
(e)		GT + S	GT + S	**0.73**
(f)	event	Mono E + S	Mono E + S	**0.73**
(g)	image	Mono I + S	Mono I + S	0.70
(h)	blind	RS + E + S	B + S	0.68
(i)		B + S	B + S	0.66
(j)		-	DB + S	0.57

Table 3. **Success rate for approaches trained with state information.** Notation: RS for RAFT-Stereo [34], E for events, S for state, I for images, Mono for monocular, B for blind, DB for direct baseline.

	Abstraction	Train	Evaluate	Success	Improvement
(a)	depth	RS + E	RS + E	0.73	0.02
(b)		RS + E	SGM + E	0.71	0.04
(c)		SGM + E	SGM + E	0.59	-0.11
(d)		SGM + I	SGM + I	0.72	0.05
(e)		GT	GT	**0.77**	0.04
(f)	event	Mono E	Mono E	0.74	0.01
(g)	image	Mono I	Mono I	0.60	-0.1
(h)	blind	RS + E	B	0.38	-0.3
(i)		-	DB	0.57	0.00

Table 4. **Success rate for approaches trained without state information.** Notation: RS for RAFT-Stereo [34], E for events, I for images, Mono for monocular, B for blind, DB for direct baseline.

gives a boost over using images directly. Especially, when the state is not provided, our image-based planner degrades by a full 14%. We conjecture that the motion information in the events give the planner ample information to perform well in the task of collision-free navigation, while images, being static representation, do not carry this information.

Monocular events have low perception latency. To investigate the optimal event horizon and the effect of the planning frequency on the navigation performance, we conduct ablation studies and show the results in Tab. 6. We vary the horizon from 1ms to 30ms, and train the planner for each setting. Fig. 7 visualizes the event representation with different horizons. We show that even a horizon of 1 ms events is sufficient to fly the quadrotor with a success rate of 0.79! Since the quadrotor flies at a speed of about 5 m/s, the fast motion brings significant brightness change, and thus creates enough events even within a very short period of time. Considering that the latency of a standard camera is usually larger than 30 ms (see Tab. 5), our work shows the great potential of event cameras for high-speed autonomous applications. In addition, we also show the generalization ability

	Sensor	Latency (ms)
(a)	RealSense D415 RGB	33.3
(b)	RealSense D415 Depth	33.3
(c)	Prophesee Gen 3.1	**1.0**

Table 5. **The event camera provides superior performance in perception latency.** The RealSense D415 camera updates RGB frames at 30Hz, and updates depth at 30Hz and up to 90 Hz. The event camera requires a horizon of only 1ms for the planner to work.

		Planning step (ms)				
	Horizon (ms)	10	30	50	100	1000
(a)	1	**0.79**	**0.76**	0.76	0.68	0.20
(b)	3	0.74	0.65	**0.79**	**0.78**	0.35
(c)	5	0.67	0.66	0.65	0.68	0.20
(d)	10	0.73	0.73	0.69	0.73	0.15
(e)	30	0.77	0.73	0.74	0.72	**0.36**

Table 6. **Ablation study on event stream horizon and planner update rate.** The fast motion of quadrotor brings enough events even within a very short period of time. When the planning frequency is very low (1Hz), the performance drops significantly. Note that the horizon MPC needs is 1.0 seconds, having a very slow planning rate results in inconsistent tracking for MPC, thus resulting performance degeneration.

Horizon (ms)	1	3	5	10	20	30
Success	0.7	0.67	0.63	0.63	**0.74**	0.71

Table 7. **Generalization ability with different event horizon.** The planner is trained with horizon 20ms, but evaluated under different horizons. The performance does not degrade too much while having different horizons.

of the planner when evaluating with different densities of events. Tab. 7 shows that the learned planner is robust to different horizon of events. As a visual demonstration, we show in Fig. 6 a successful closed-loop rollout in simulation. The quadrotor plans a trajectory through some trees correctly and finishes the rollout successfully. In addition, using the images often leads to larger sim-to-real gap [36] due to a larger domain shift.

4.4. Real-World Experiments

Real-world planning. We use a monocular Prophesee Gen3.1, and collect sequences of events in the real world to show that our approach can bridge the sim-to-real gap. Due to the lack of ground truth, we define a planned trajectory as successful if the predicted trajectory aligns with the collision-free direction. For example, if the tree is on the left and the planned trajectory leans towards the right, that

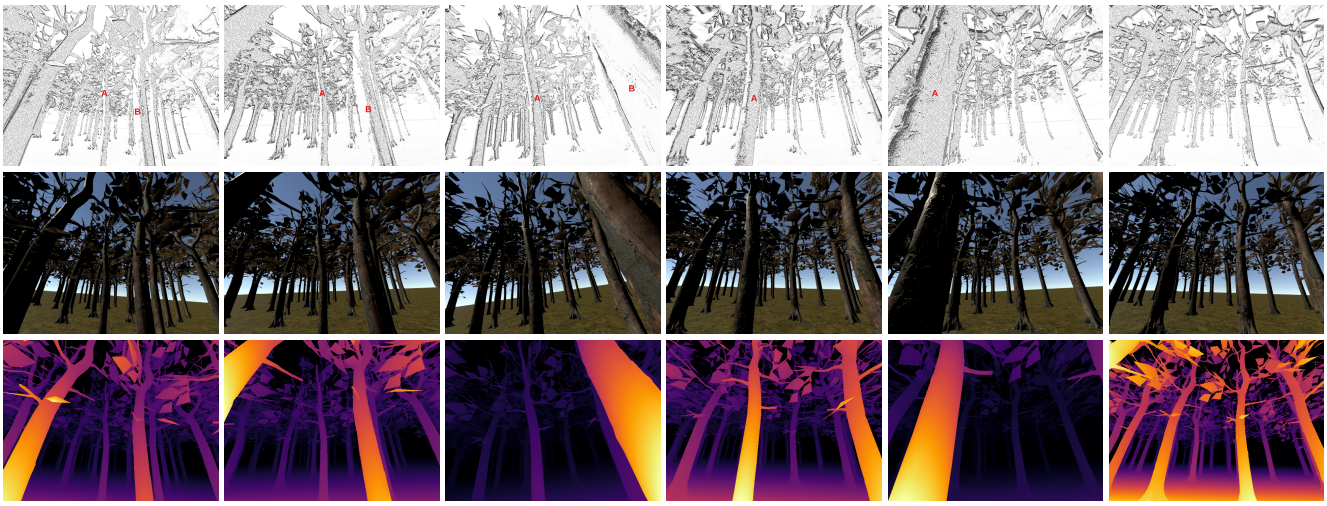


Figure 6. Illustration of a successful closed-loop flying with a monocular event camera in simulation. From top to bottom are event tensor, RGB image, and the ground truth depth observed at different time steps. Note that only the events are used for the planner. To indicate the flying progress, we denote two trees by red letters A and B. At $t = 0$, tree A is in the straight line front of the quadrotor, flying straight will lead to a collision. The learned planner predicts trajectories that turn towards the right, and the quadrotor successfully fly through the space between tree A and tree B. (red, in top row)

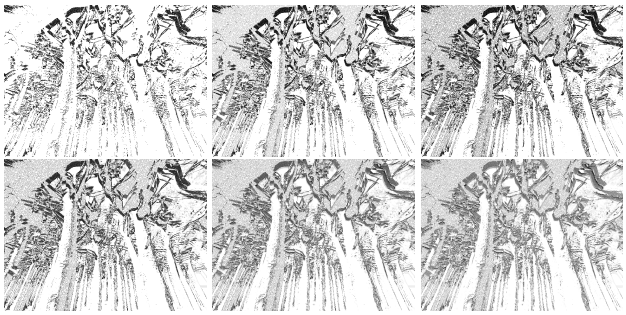


Figure 7. Visualization of event representation with different horizons. Top to bottom, left to right: 1ms, 3ms, 5ms, 10ms, 20ms, 30ms.

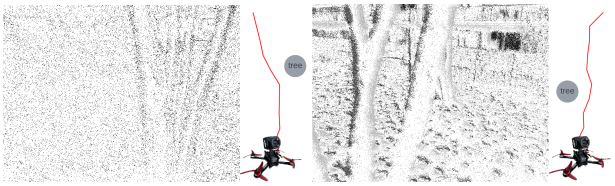


Figure 8. Trained Planner transfers zero-shot in the real world. We provide two visualizations with real-world events and a top-down view of predicted waypoints (red curve). The predicted waypoints lead to collision-free areas while observing obstacles.

would be considered a successful plan. As shown in Fig. 8, the predicted waypoints lead to collision-free areas while observing trees. More real-world experiments can be found in the supplementary material.

5. Conclusion

We have presented a novel pipeline for autonomous navigation, that takes streams of a single event camera and predicts a collision-free trajectory. We show that our planner can fly a quadrotor with only 1 ms horizon of events while achieving a success rate of 79%. This is about 30x faster than conventional depth or RGB camera pipelines and the success rate is similar or better.

While this is an exciting step towards event-based agile autonomous systems, there are multiple opportunities for future work. For example, solving the optimization problem to find collision-free trajectories is computationally expensive and the generalization to new environments is limited. One interesting avenue could be to explore deep neural networks for planning to reduce latency and improve generalization. In order to ensure a consistent representation of the environment over time, the event processing pipeline may benefit from recurrent modules. We hope that this work will inspire more research in the space of low-latency perception for autonomous navigation.

References

[1] Md Ahad, Atiqur Rahman, JK Tan, H Kim, and S Ishikawa. Motion history image: its variants and applications. *Machine Vision and Applications*, 23(2):255–281, 2012. 3

[2] Patrick Bardow, Andrew J Davison, and Stefan Leuteneg-

ger. Simultaneous optical flow and intensity estimation from an event camera. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 884–892, 2016. 3

[3] Andrew J Barry, Peter R Florence, and Russ Tedrake. High-speed autonomous obstacle avoidance with pushbroom stereo. *Journal of Field Robotics*, 35(1):52–68, 2018. 2

[4] Hermann Blum, Alexander Dietmüller, Moritz Milde, Jörg Conradt, Giacomo Indiveri, and Yulia Sandamirskaya. A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor. *Robotics Science and Systems, RSS 2017*, 2017. 3

[5] Robert Bridson. Fast poisson disk sampling in arbitrary dimensions. *SIGGRAPH sketches*, 10(1):1, 2007. 4, 5

[6] Andrea Censi. Efficient neuromorphic optomotor heading regulation. In *2015 American Control Conference (ACC)*, pages 3854–3861. IEEE, 2015. 3

[7] Bryan Chen*, Alexander Sax*, Francis Lewis, Silvio Savarese, Amir Zamir, Jitendra Malik, and Lerrel Pinto. Robust policies via mid-level visual representations: An experimental study in manipulation and navigation. In *4th Annual Conference on Robot Learning, CoRL 2020*, Proceedings of Machine Learning Research. PMLR, 2020. 3, 6

[8] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020. 2

[9] Tao Chen, Saurabh Gupta, and Abhinav Gupta. Learning exploration policies for navigation. *arXiv preprint arXiv:1903.01959*, 2019. 2, 3

[10] Wensheng Cheng, Hao Luo, Wen Yang, Lei Yu, Shoushun Chen, and Wei Li. Det: A high-resolution dvs dataset for lane extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 3

[11] Jorg Conradt, Raphael Berner, Matthew Cook, and Tobi Delbruck. An embedded aer dynamic vision sensor for low-latency pole balancing. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 780–785. IEEE, 2009. 3

[12] Shreyansh Daftry, Sam Zeng, Arbaaz Khan, Debadeepta Dey, Narek Melik-Barkhudarov, J Andrew Bagnell, and Martial Hebert. Robust monocular flight in cluttered outdoor environments. *arXiv preprint arXiv:1604.04779*, 2016. 2

[13] Pierre de Tournemire, Davide Nitti, Etienne Perot, Davide Migliore, and Amos Sironi. A large scale event-based detection dataset for automotive. *arXiv preprint arXiv:2001.08499*, 2020. 3

[14] 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. 3

[15] 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. 3

[16] Davide Falanga, Suseong Kim, and Davide Scaramuzza. How fast is too fast? the role of perception latency in high-speed sense and avoid. *IEEE Robotics and Automation Letters*, 4(2):1884–1891, 2019. 1

[17] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020. 2, 3

[18] Tobias Fischer and Michael Milford. Event-based visual place recognition with ensembles of temporal windows. *IEEE Robotics and Automation Letters*, 5(4):6924–6931, 2020. 3

[19] Pete Florence, John Carter, and Russ Tedrake. Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Algorithmic Foundations of Robotics XII*, pages 304–319. Springer, 2020. 1, 2

[20] 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. 1, 2, 3

[21] Francesco Galluppi, Christian Denk, Matthias C Meiner, Terrence C Stewart, Luis A Plana, Chris Eliasmith, Steve Furber, and Jörg Conradt. Event-based neural computing on an autonomous mobile platform. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2862–2867. IEEE, 2014. 3

[22] Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5633–5643, 2019. 4

[23] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954, 2021. 3

[24] Arren Glover and Chiara Bartolozzi. Event-driven ball detection and gaze fixation in clutter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2203–2208. IEEE, 2016. 3

[25] Arren Glover and Chiara Bartolozzi. Robust visual tracking with a freely-moving event camera. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776. IEEE, 2017. 3

[26] Javier Hidalgo-Carrió, Daniel Gehrig, and Davide Scaramuzza. Learning monocular dense depth from events. In *2020 International Conference on 3D Vision (3DV)*, pages 534–542. IEEE, 2020. 3

[27] Noriaki Hirose, Fei Xia, Roberto Martín-Martín, Amir Sadeghian, and Silvio Savarese. Deep visual mpc-policy learning for navigation. *IEEE Robotics and Automation Letters*, 4(4):3184–3191, 2019. 2

[28] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007. 6

[29] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019. 4

[30] Yuhuang Hu, Jonathan Binas, Daniel Neil, Shih-Chii Liu, and Tobi Delbruck. Ddd20 end-to-end event camera driv-

ing dataset: Fusing frames and events with deep learning for improved steering prediction. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020. 3

[31] Hanme Kim, Stefan Leutenegger, and Andrew J Davison. Real-time 3d reconstruction and 6-dof tracking with an event camera. In *European conference on computer vision*, pages 349–364. Springer, 2016. 3

[32] Diederik P. Kingma and Jimmy Lei Ba. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. 5

[33] Xavier Lagorce, Garrick Orchard, Francesco Galluppi, Bertram E Shi, and Ryad B Benosman. Hots: a hierarchy of event-based time-surfaces for pattern recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1346–1359, 2016. 3

[34] Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *International Conference on 3D Vision (3DV)*, 2021. 6, 7

[35] Sikang Liu, Nikolay Atanasov, Kartik Mohta, and Vijay Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 2872–2879. IEEE, 2017. 4, 5

[36] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59), 2021. 2, 3, 4, 6, 7

[37] Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, and Vincent Lepetit. Speed invariant time surface for learning to detect corner points with event-based cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10245–10254, 2019. 3

[38] Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 672–678. IEEE, 2020. 2

[39] Xiangyun Meng, Yu Xiang, and Dieter Fox. Learning composable behavior embeddings for long-horizon visual navigation. *IEEE Robotics and Automation Letters*, 6(2):3128–3135, 2021. 2, 3

[40] Moritz B Milde, Olivier JN Bertrand, Harshawardhan Ramachandran, Martin Egelhaaf, and Elisabetta Chicca. Spiking elementary motion detector in neuromorphic systems. *Neural computation*, 30(9):2384–2417, 2018. 3

[41] 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

[42] Elias Mueggler, Nathan Baumli, Flavio Fontana, and Davide Scaramuzza. Towards evasive maneuvers with quadrotors using dynamic vision sensors. In *2015 European Conference on Mobile Robots (ECMR)*, pages 1–8. IEEE, 2015. 3

[43] Erich Mueller, Andrea Censi, and Emilio Frazzoli. Low-latency heading feedback control with neuromorphic vision sensors using efficient approximated incremental inference. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 992–999. IEEE, 2015. 3

[44] T Osa, J Pajarinen, G Neumann, JA Bagnell, P Abbeel, and J Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018. 2

[45] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. *Advances in Neural Information Processing Systems*, 33:16639–16652, 2020. 3

[46] Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Emvs: Event-based multi-view stereo. 2016. 3

[47] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. Esim: an open event camera simulator. In *Conference on robot learning*, pages 969–982. PMLR, 2018. 3, 4

[48] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. 2017. 3

[49] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2016. 3

[50] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):1964–1980, 2019. 3, 6

[51] Stéphane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J Andrew Bagnell, and Martial Hebert. Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE international conference on robotics and automation*, pages 1765–1772. IEEE, 2013. 2

[52] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016. 2, 3

[53] Nitin J Sanket, Chethan M Parameshwara, Chahat Deep Singh, Ashwin V Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657. IEEE, 2020. 2, 3

[54] Sebastian Scherer, Sanjiv Singh, Lyle Chamberlain, and Mike Elgersma. Flying fast and low among obstacles: Methodology and experiments. *The International Journal of Robotics Research*, 27(5):549–574, 2008. 2

[55] L. Schmid, C. Ni, Y. Zhong, , R. Siegwart, and O. Andersson. Fast and compute-efficient sampling-based local exploration planning via distribution learning. *IEEE Robotics and Automation Letters*, 7(3):7810–7817, 2022. 2

[56] Dhruv Shah, Benjamin Eysenbach, Nicholas Rhinehart, and Sergey Levine. Rapid Exploration for Open-World Navigation with Latent Goal Models. In *5th Annual Conference on Robot Learning*, 2021. 2, 3

[57] Dhruv Shah, Ajay Sridhar, Arjun Bhorkar, Noriaki Hirose, and Sergey Levine. Gnm: A general navigation model to drive any robot. *arXiv preprint arXiv:2210.03370*, 2022. 2, 3

[58] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1731–1740, 2018. 3

[59] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Conference on Robot Learning*, pages 1147–1157. PMLR, 2021. 3, 4

[60] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018. 2

[61] Stepan Tulyakov, Francois Fleuret, Martin Kiefel, Peter Gehler, and Michael Hirsch. Learning an event sequence embedding for dense event-based deep stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1527–1537, 2019. 3

[62] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate slam? combining events, images, and imu for robust visual slam in hdr and high-speed scenarios. *IEEE Robotics and Automation Letters*, 3(2):994–1001, 2018. 2, 3

[63] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10081–10090, 2019. 3

[64] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 2

[65] Zichao Zhang and Davide Scaramuzza. Perception-aware receding horizon navigation for mavs. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2534–2541. IEEE, 2018. 2

[66] Boyu Zhou, Fei Gao, Luqi Wang, Chuhao Liu, and Shaojie Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019. 1, 2

[67] Brady Zhou, Philipp Krähenbühl, and Vladlen Koltun. Does computer vision matter for action? *Science Robotics*, 4(30):eaaw6661, 2019. 3, 6

[68] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3d reconstruction with a stereo event camera. In *Proceedings of the European conference on computer vision (ECCV)*, pages 235–251, 2018. 3

[69] Alex Zihao Zhu, Yibo Chen, and Kostas Daniilidis. Real-time time synchronized event-based stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 433–447, 2018. 3

[70] Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018. 3

[71] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 989–997, 2019. 3, 4