This paper has been accepted for publication at the
Conference on Robot Learning (CoRL), Austin 2026

Approximate Imitation Learning for Event-based Quadrotor Flight in Cluttered Environments

Nico Messikommer, Jiaxu Xing, Leonard Bauersfeld,
Marco Cannici, Elie Aljalbout, Davide Scaramuzza
Robotics and Perception Group,
University of Zurich, Switzerland.
[nmessi, jixing, bauersfeld, cannici, aljalbout, sdavide]@ifi.uzh.ch

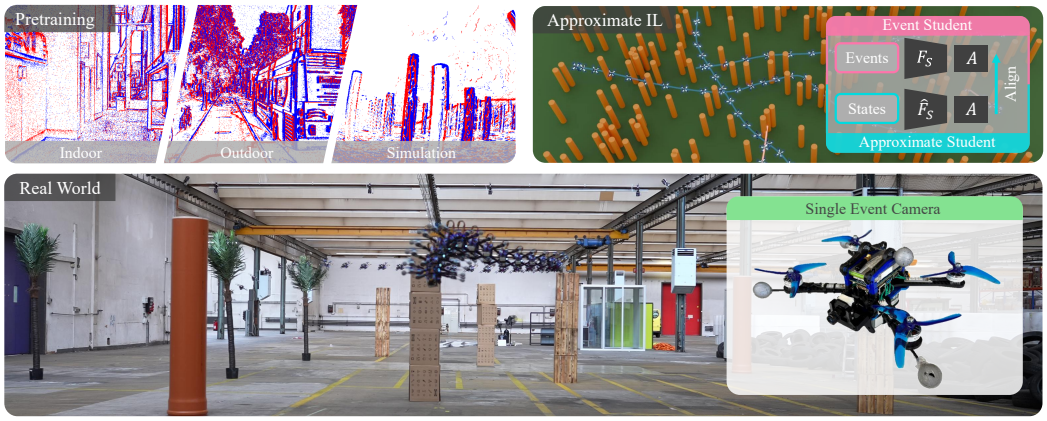


Figure 1: To fine-tune the event-based policy pretrained on offline datasets, we introduce an approximate student that receives simulated state information, eliminating the need for computationally expensive event rendering. We validate our framework by flying a quadrotor equipped with a single event camera through cluttered real-world environments using external pose information.

Abstract: Event cameras offer high temporal resolution and low latency, making them ideal sensors for high-speed robotic applications where conventional cameras suffer from motion blur. However, their widespread adoption in robot learning is severely bottlenecked by the computational cost of simulating high-frequency event data during online training. In this work, we present *Approximate Imitation Learning*, a novel framework that fundamentally resolves this bottleneck, reducing policy training time for complex, agile drone flight from 52.44 hours to just 1.86 hours—a $28\times$ computational speedup. Our key insight is to separate representation learning from policy search. We first leverage a large-scale offline dataset to learn a task-specific representation space. Subsequently, the policy is fine-tuned through online interactions that rely solely on lightweight state information, completely eliminating the need to render events during the active policy search phase. This training paradigm drastically reduces development overhead and enables event-based control policies to scale to complex environments. Furthermore, our approach eliminates the reliance on standard cameras or intermediate representations during deployment, mapping events directly to control commands. In simulation, our method matches or exceeds the performance of standard imitation learning baselines that require full online event rendering. Finally, we successfully validate the framework in the real world, demonstrating that a policy trained via this ultra-efficient paradigm enables a quadrotor to fly through highly cluttered environments at remarkable speeds of up to $9.8\ \mathrm{ms}^{-1}$.

Keywords: Imitation Learning, Event Cameras, Quadrotor

Multimedia Material A video is available at https://youtu.be/yI8wGTaoYHA

10th Conference on Robot Learning (CoRL 2026), Austin TX, USA.

1 Introduction

Standard cameras capture dense visual information of the environment, enabling impressive robot performance across a wide range of applications [1, 2, 3]. However, their performance degrades during high-speed motion due to motion blur [4, 5]. Since standard cameras integrate incoming light over a finite exposure time, any image displacement exceeding one pixel during this period results in blurred images. Consequently, perception quality deteriorates significantly, which negatively affects the downstream robot policy.

Event cameras offer a radically different sensing mechanism inspired by the biological vision system [6]. Instead of integrating the incoming light, each pixel in an event camera asynchronously triggers events if the change in intensity exceeds a given threshold. The asynchronous triggering of events reduces sensing latency to submilliseconds, compared to standard cameras that capture visual information at fixed frame rates. As a result, they provide sharp visual information even in fast motions with low latency. Furthermore, event cameras excel in scenes with very large brightness variations since they feature a high dynamic range that exceeds $120\,\mathrm{dB}$. These properties make event cameras an ideal choice to capture scene information reliably in high-speed robotic tasks.

An important application of reliable perception during high-speed motion is autonomous quadrotor flight for search-and-rescue missions, where rapid navigation in cluttered environments can be life-saving. Prior work using standard cameras has demonstrated promising results in such environments [7], but remains limited by motion blur. To improve reliability, recent work has explored mounting LiDAR sensors on quadrotors [8]. While already smaller in scale, LiDAR sensors still have a high cost and are heavier. Finally, methods using event cameras to increase sensing robustness have been proposed [9, 10]. However, most of them use a standard camera and can additionally rely on an intermediate representation, which introduces computational overhead [10].

In our work, we propose an end-to-end approach, which maps events directly to commands without any intermediate representation or standard cameras, see Fig. 1. This enables the method to directly leverage the high temporal resolution of the event camera for fast quadrotor flight. During training, to avoid the high computational cost of rendering high-frequency event data, our method first learns a task-relevant representation space from large-scale offline event datasets. This learned representation is then used within a lightweight simulator to train the control policy via the action decoder. Crucially, the learning of the behavior does not rely on rendering events, making it highly efficient.

The results in simulation confirm the effectiveness of our proposed *approximate Imitation Learning (IL)* framework. Our method achieves a success rate increase of 0.2 over the behavior cloning baseline and an increase of 0.7 to the DAgger [11] baseline. By leveraging the approximate student, our approach significantly reduces training costs, accelerating policy training by a factor of 28 (from 52.44 hours to just 1.86 hours). Finally, we validate our approach in the real world, where the quadrotor achieves speeds up to $9.8\,\mathrm{m\,s^{-1}}$ using external state information. Although we apply our method only to event data, the framework is broadly applicable and can easily be extended to other modalities, which are computationally expensive to simulate, e.g., tactile sensors, radar, or LiDAR.

2 Related Works

Robot Learning Achieving agile control with limited real-world data remains a central challenge in robot learning. While reinforcement learning (RL) has demonstrated strong performance in simulation [12] and real-world control [13, 14, 15, 16], it is often sample-inefficient and computationally expensive. Imitation learning (IL) mitigates these issues using expert demonstrations [17, 18, 19], yet methods such as DAgger [11] still require substantial interaction. Recent work improves efficiency further through pretrained visual representations [20, 21, 22]. Building on these advances, we propose an approximate IL framework for training-efficient, low-latency quadrotor navigation.

Event-Based Quadrotor Flight Event cameras are increasingly used in robotics because of their high temporal resolution and low-latency output [23, 24, 25], making them well-suited for agile

flight [26, 27, 28, 29, 30]. Early efforts focused on handcrafted, low-latency pipelines for tasks such as obstacle avoidance, achieving control loop latencies as low as $3.5\,\mathrm{ms}$ [9]. Spiking neural networks (SNNs) have been explored for their compatibility with event data, including line tracking on constrained platforms [31], real-time drone flight via ego-motion estimation [28], and dynamic obstacle avoidance using event–depth fusion [32]. In parallel, conventional deep neural networks have been applied to tasks such as high-speed gate detection in drone racing [33]. In [10], an event-based policy is trained in simulation via teacher–student distillation. While effective, their approach relies on approximate event renderings with limited temporal resolution during training. In contrast, we first learn a task-specific representation space from offline data, and then rely on an approximate student that uses state information to simulate the event-driven behavior, avoiding event rendering entirely.

Vision-Based Quadrotor Flight Autonomous quadrotor navigation in cluttered environments commonly relies on onboard visual sensing, without external localization [34, 35]. Traditional model-based pipelines combine visual-inertial odometry (VIO) [36, 37] with separate planning and control modules [38, 39]. While effective in structured settings, they still suffer from brittle state estimation, delayed reactions, and limited adaptability in dynamic or perceptually degraded settings. An early RL approach uses VIO as input to a policy that predicts discrete velocity commands [40]. Subsequent methods [41, 42, 43, 18, 44] learn to predict high-level control commands directly from visual observations. To achieve tighter perception–action coupling, more recent approaches instead predict low-level control commands directly from images. Notably, the integration of VIO with a simulation-trained, motion-capture-finetuned RL policy surpassed world champion pilots in drone racing [45], and subsequent works achieve comparable agility through purely vision-based RL, without any state estimator [46, 47, 48]. In parallel, IL has improved sample efficiency and has enabled agile end-to-end flight [49, 50, 22] by minimizing the discrepancy between a learned policy and expert demonstrations [11, 51]. Nevertheless, these methods still require millions of simulated interactions, which is especially problematic for event-based vision, where rendering event streams is computationally costly. In this work, we address this limitation through our approximate imitation learning, reducing sample complexity and making event-driven agile flight more tractable.

3 Methodology

In high-speed quadrotor flight through cluttered environments, the objective is to safely navigate obstacle-dense spaces while pushing the limits of perception by maximizing flight speed. To instantiate this setting, the policy is conditioned on a commanded direction with a magnitude corresponding to the target velocity, which it tries to track while simultaneously avoiding obstacles.

To efficiently train an end-to-end policy using events, we adopt a two-stage learning approach that separates representation learning from policy search, as illustrated in Fig. 2. In the first stage, we train a mapping from events to a representation space using a pre-rendered offline dataset consisting of paired event representations and teacher actions. To ensure that the learned representation space covers task-relevant information, we apply a Behavior Cloning (BC) objective based on the actions of the teacher. We additionally incorporate a real-world dataset to train on an auxiliary depth task to improve generalization and robustness. Nevertheless, BC struggles when the training data does not adequately cover the state distribution encountered during real-world execution. To address this, we introduce a second stage that includes online student interaction in a simulator, crucially, without rendering events. Instead, we use an approximate student model that observes full state information and learns to mimic the actions of the event-based student, which is inspired by [52]. These interactions allow us to adapt the shared action decoder to a broader state distribution while avoiding the high computational cost of event rendering. In the following, we provide a detailed explanation of each training stage. The neural network architectures are introduced in the appendix.

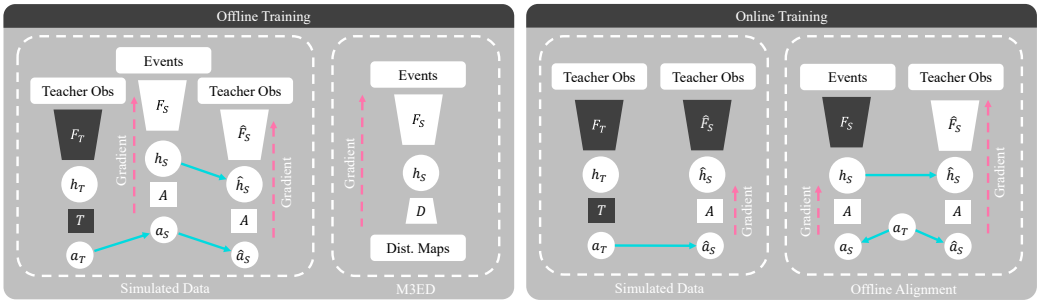


Figure 2: During offline training, teacher actions a_T supervise the event encoder F_S and the shared action decoder A. The approximate student, using teacher observations, is trained by aligning its features $\hat{h}_S$ and actions $\hat{a}_S$ with the features h_S and actions a_S of the event student. During online training, teacher observations obtained from lightweight simulations are used to fine-tune the shared action decoder A. Consistency losses in both the action and feature spaces are applied during an additional offline alignment step. For clarity, the auxiliary decoder D_o is omitted.

3.1 Offline Training

The offline dataset is generated by rolling out the teacher policy for 40,000 steps across 25 parallel environments with small stochastic perturbations applied to the teacher actions. Crucially, the expensive event rendering is performed only once and reused across multiple training runs. The event-based student encoder F_S is jointly trained with a shared action decoder A using a standard BC objective to predict the teacher action a_T. In parallel, we train an approximate student to match the latent features h_S and action outputs a_S of the event-based student using state observations. Specifically, the approximate student is supervised with an L1 loss between the latent features of its encoder $\hat{F}_S$ and the event-based encoder F_S, and an MSE loss on the action predictions, both computed using the shared decoder A.

To reduce the sim-to-real gap [53] in the event-based student encoder F_S, we leverage the M3ED dataset [54], which includes event streams and depth maps collected from three platforms: a walking quadruped, a car, and a quadrotor. Since teacher actions cannot be queried, we introduce an auxiliary depth prediction task. Specifically, the student encoder is trained to predict 1-D angular distance maps, similar to a forward-facing single-line LiDAR scan restricted to the camera's field of view, which resembles the teacher observations. For the teacher, these distance maps are horizontally aligned using the gravity vector to compensate for tilt. For the student, depth images are instead divided into three horizontal bands, and the minimum depth within 11.25° angular intervals is extracted for each band. The three resulting distance maps are then concatenated into a single 1-D observation. Using multiple bands reduces the need for gravity alignment by increasing the likelihood that at least one band captures an obstacle-free path, even when the camera faces the ground or sky. These angular distance maps are predicted with a dedicated decoder D using the features of the event encoder. Similarly, using the rendered events in the offline dataset, the gravity-aligned distance maps that serve as input to the teacher are used to jointly train a separate auxiliary decoder D_o and the event encoder F_S, further infusing task-relevant information.

The offline learning stage alternates between two update steps: a BC update and an auxiliary depth task update using simulated and real-world events from M3ED. Each training iteration begins with a batch of real-world data, which is used to update the event-based student encoder F_S and the distance decoder D. This is followed by a batch from the simulated dataset, used to update the event-based student encoder F_S, the shared action decoder A, the approximate student $\hat{F}_S$, and the auxiliary decoder D_o. Through this alternating update scheme, the learned representation is grounded in real-world event data and optimized for task relevance via supervision from expert demonstrations.

3.2 Online Training

To improve the generalization of the event policy, we further fine-tune the shared action decoder A through online interactions in simulation. Crucially, to avoid the computational cost of rendering events, we leverage the approximate student to approximate the behavior of the event-based student. We adopt the DAgger framework [55], where both the teacher and the approximate student receive the same observations. An adaptive action sampling strategy, based on the difference between their predicted actions, gradually increases the proportion of actions taken by the student. This setup enables the collection of diverse experiences for training the action decoder A across a broader state distribution, while entirely bypassing the need for rendering events during online learning.

To prevent the action decoder A from learning distinct manifolds for approximate–event student alignment and state-based behavior fine-tuning, we add an offline alignment update. In this update, the action decoder is supervised to predict teacher actions from offline features generated by the event and approximate student. In addition, the approximate student encoder is updated via feature consistency between the event and the approximate student, improving the robustness of the task decoder to small feature perturbations. This alignment update step adds minimal overhead, since the event features h_S for the offline data can be pre-computed once and reused for all fine-tuning runs.

3.3 Event Rendering

To improve render efficiency, we follow [10] and directly generate event representations, eliminating the computationally intensive step of synthesizing sparse event sequences. In contrast to [10], which generates single-channel representations, our proposed approach truthfully constructs a multi-channel event tensor that accumulates the events in a given temporal window for each polarity [56]. Importantly, our method generates event representations for multiple videos in parallel using vectorized operations, significantly accelerating the simulation.

Given a high-frame-rate video with images $I_0, I_1, ...I_T$, we first compute the log-intensity for each frame and subtract the initial frame I_0, establishing it as the reference. Each normalized log-intensity frame is floor-divided by the contrast threshold C, assigning each pixel at each timestep to a discrete *contrast band*. To detect band transitions, we compute the temporal difference of band indices between consecutive frames. Crucially, an event is only triggered when the intensity curve exits the current band in the opposite direction from which it previously entered. This condition is met at timesteps where the second-order difference is zero, indicating that the logarithmic difference exceeded the contrast threshold C. To obtain the final event representation, a simple reshaping and summing is required. The event generation steps are summarized in Alg. 1.

Algorithm 1 Event Representation Generation

Input Log frames $\{L_0, L_1, \ldots, L_T\}$, contrast C
Output Event representation E_T

$\Delta L \leftarrow \{L_t - L_0 \mid t = 0, \ldots, T\}$
$\text{BandID} \leftarrow \lfloor \Delta L / C \rfloor$
$\Delta\text{BandID} \leftarrow \text{BandID}_{1:T} - \text{BandID}_{0:T-1}$
$\text{BandID} \leftarrow \text{NonZero}(\Delta\text{BandID})$
$\Delta^2\text{BandID} \leftarrow \text{BandID}_{1:F} - \text{BandID}_{0:F-1}$
$E_T \leftarrow \sum_{\text{axis}} \left[\Delta^2\text{BandID} = 0\right]$

Except for a few corner cases related to specific initial conditions, the resulting event representations are equivalent to those derived from sparse event sequences generated with ESIM [57]. While our approach reduces average runtime by 34% compared to generating event representations with ESIM's CUDA-based implementation, its primary advantage lies in the significantly improved memory efficiency, illustrated in the appendix. Compared to ESIM, our method achieves a reduction of more than four times in peak GPU memory usage, removing a key bottleneck for large-scale event simulation. A more detailed and visual explanation of our event rendering approach are provided in the appendix.

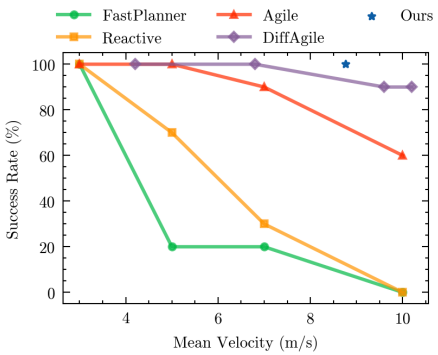


Figure 3: Illustrative comparison of mean velocity versus success rates reported by baseline methods using absolute depth.

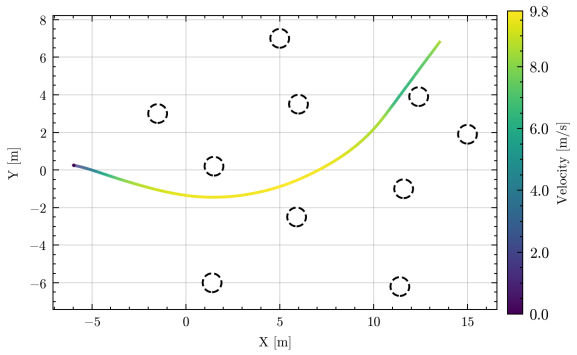


Figure 4: Top-down view of a real-world flight trajectory alongside approximate obstacle locations. Fig. 5 A shows the first half of this trajectory.

4 Experiments

We evaluate our approach in simulation using Flightmare (Sec. 4.1) and validate its real-world performance on a quadrotor platform similar to [49] (Sec. 4.2), with platform details in the appendix.

4.1 Simulation

Setup We construct a simulated forest environment where trees are modeled as tall cylinders with radii ranging from $0.2\,\mathrm{m}$ to $0.5\,\mathrm{m}$. These cylinders are placed within a $100\,\mathrm{m}\times100\,\mathrm{m}$ world box rendered in a natural outdoor scene, see appendix. Tree positions are sampled from a Poisson point process, with the Poisson delta controlling the tree density across environments. For each Poisson delta value (0.04 and 0.05), we sample 10 environments, resulting in 20 test environments. In each test episode, the quadrotor starts at the center of the left edge of the environment and receives a direction command pointing to the right. A flight is successful if the quadrotor travels more than $40\,\mathrm{m}$ in the commanded direction without crashing into obstacles or leaving the world box.

Results As shown in Tab. 1, our proposed behavior fine-tuning via lightweight simulation leads to a 0.2 higher success rate (1.0 vs. 0.8) than Behavior Cloning (BC). When increasing the tree density to a Poisson delta of 0.05, the performance of our approach decreases from 1.00 to 0.90, while still maintaining a 0.20 advantage over BC. Despite access to online event rendering, the number of rendering steps used for our approach is insufficient for DAgger [55] to learn a robust mapping from events and state information to control commands. Moreover, the equal performance of our approximate IL and the BC+DAgger baseline indicates that our approximation with lightweight states closely matches real event renderings, while substantially reducing the training time, as illustrated in Fig. 6. Finally, our event student performs comparably to the teacher policy, which relies on highly informative distance maps, confirming that the teacher–student performance gap is small.

The advantage of our approximate IL method over BC, the best-performing baseline under the same training conditions, is also evident when only onboard inputs are available, i.e., previous actions and commanded direction. Without access to the quadrotor state, our method improves the success rate by 0.3 compared to BC (0.5 vs. 0.2), as reported in Tab. 2. Crucially, both approaches rely on the same number of pre-rendered event representations, since our method avoids costly online event rendering by leveraging approximate policy updates in simulation.

Ablations To validate our design choices, we conduct ablation studies in simulation using an event-based policy with access to state information in environments generated with a Poisson delta of 0.04. In addition to the success rate, we report the action MSE between the event student and teacher, and between the approximate student and teacher, as well as the L1 distance between the features of the two student encoders h_S and $\hat{h}_S$ (Feature L1). As shown in Tab. 3, removing the M3ED training

Figure 5: Our end-to-end policy demonstrates reliable flight in complex (A/B) real-world environments using only a monocular event camera and onboard compute.

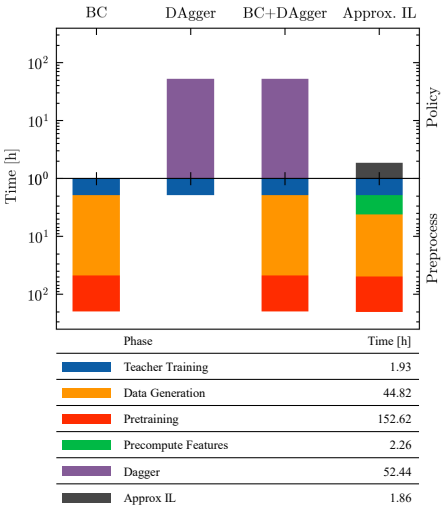


Figure 6: Training time grouped into a *Preprocess* and a *Policy* stage.

Table 1: Simulation results for methods using offline (Off.) and/or online (On.) training input.

		Poisson Delta: 0.04		Poisson Delta: 0.05	
Methods	Training Input	Success Rate	Mean Velocity [$\frac{\mathrm{m}}{\mathrm{s}}$]	Success Rate	Mean Velocity [$\frac{\mathrm{m}}{\mathrm{s}}$]
BC	Off. Event	0.80	8.53	0.70	8.13
DAgger [55]	On. Event	0.30	6.66	0.10	5.72
BC+DAgger	Off. Event + On. Event	1.00	9.00	0.90	8.47
Approx IL (Ours)	Off. Event + On. State	1.00	8.76	0.90	8.53
State Teacher (Ours)	On. State + Dist. Map	1.00	8.77	0.90	8.60

stage with the auxiliary distance map prediction task (BC w/o M3ED) significantly reduces the success rate to 0.30 and results in the largest action difference to the teacher. This observation suggests that the event and approximate students overfit to the offline action distribution. Adding M3ED during offline training (BC) improves the success rate to 0.80 at the cost of a larger feature difference between the event and approximate students. Simply using the approximate student during online training to fine-tune the action decoder through online interactions (Approx IL w/o Off. Align.) decreases the success rate to 0.50, despite lower action MSE. This can be explained by the action decoder leveraging encoder features specific to the approximate student, resulting in separate feature manifolds for the event and approximate students. Introducing offline alignment during online training mitigates this issue by encouraging the action decoder to remain robust to feature perturbations. The resulting model (Approx IL) achieves the best overall performance, with a success rate of 1.00 and the lowest action and feature differences. To assess the alignment between the approximate and event student, we compute the action MSE on events rendered from the approximate student rollout states. The resulting MSE of 0.00199 is lower than that obtained on teacher rollout states (0.00249), indicating good alignment between the two students. Additional ablations on learnable parameters, different levels of state information, and the distance map decoder are provided in the appendix.

Dynamic Obstacles To further assess the generalization of our approximate imitation learning framework, we evaluate our policy and the BC+DAgger baseline in simulation with dynamic obstacles moving in random directions at constant velocities sampled between $0.5\,\mathrm{m\,s^{-1}}$ and $1.5\,\mathrm{m\,s^{-1}}$. Obstacle initial positions are sampled with a Poisson Delta of 0.04. Without retraining, our policy achieves a success rate of 0.9, as shown in Fig. 7. These results demonstrate that the learned policy can reliably avoid moving obstacles despite being trained only on static obstacles.

Methods	Poisson Delta: 0.04	
	Success Rate	Mean Velocity [$\frac{\mathrm{m}}{\mathrm{s}}$]
BC	0.20	8.08
Approx IL	0.50	8.17

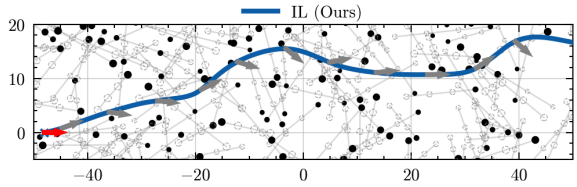
Table 2: Simulation results without quadrotor state.

Methods	Success Rate	Action MSE		Feature L1
		Event-Teacher	Approx-Teacher	Event-Approx
BC w/o M3ED	0.30	0.0596	0.0505	0.664
BC	0.80	0.0543	0.0667	0.748
Approx IL w/o Off. Align.	0.50	0.0248	0.0126	0.748
Approx IL	1.00	0.0122	0.0108	0.568

Table 3: Ablation study for the approximate IL method.

Representative Comparison To put the performance of our method into context with existing work, we plot the results reported in related work [7] in Fig. 3 and include the performance of our method for reference. Although the forest density and evaluation metrics are the same across methods, the underlying simulators and environments differ. Furthermore, the reported baselines FastPlanner [58], Reactive [59], Agile [49], and DiffAgile [7] use depth information in addition to state information. This represents a significant advantage over our monocular event camera setup. Consequently, this comparison should be interpreted as a representative reference to relate the different success rates and achieved velocities, rather than as a direct one-to-one comparison.

Training Time Fig. 6 shows the time required by each baseline for the individual training steps, grouped into a *Preprocess* and a *Policy* stage. The *Preprocess* stage consists of steps that can be reused across multiple policy trainings and therefore do not significantly contribute to the development time for a new policy. In contrast, the *Policy* stage includes steps that must be executed for every new policy. Since all evaluated methods rely on IL, they require the training of a teacher policy using lightweight state and perception observations (see appendix), which is inexpensive and takes $1.93\,\mathrm{h}$. Except for DAgger, all methods also require $44.82\,\mathrm{h}$ to generate offline event data (Data Generation) and $152.62\,\mathrm{h}$ for the subsequent offline training (Pretraining). The offline dataset can be reused across any *Policy* stage configurations, while the pretrained network remains reusable as long as the policy architecture is unchanged. To reduce training time in the *Policy* stage, our proposed approximate IL precomputes event features in $2.26\,\mathrm{h}$ for all offline samples, since its event encoder remains frozen during policy fine-tuning. The substantial advantage of our approximate IL becomes evident in the *Policy* stage, where the reliance of DAgger methods on online event rendering leads to a training time of $52.44\,\mathrm{h}$, whereas approximate IL requires only $1.86\,\mathrm{h}$. This corresponds to a reduction in policy training time by approximately a factor of 28.



Methods	Success Rate	Mean Velocity [$\frac{\mathrm{m}}{\mathrm{s}}$]
BC+DAgger	0.9	8.76
Approx IL	0.9	8.47

Figure 7: Simulation results with dynamic obstacles moving randomly at $0.5\,\mathrm{m\,s^{-1}}$ to $1.5\,\mathrm{m\,s^{-1}}$.

4.2 Real-World

Results Our event-based policy is capable of flying reliably through indoor environments with varying levels of clutter using quadrotor state estimates from an external positioning system (VICON), as shown in Fig. 5. Tested in five environments with obstacle counts that increase from sparse to dense, reaching up to ten obstacles, the quadrotor equipped with a single event camera successfully navigates trajectories through the cylindrical obstacles with a 5/5 success rate. The policy reaches a peak speed of $9.8\,\mathrm{m\,s^{-1}}$ in these indoor tests, as illustrated in Fig. 4, which shows the velocity profile along the trajectory together with the approximate obstacle locations. This strong performance achieved by an end-to-end event policy is likely supported by the inclusion of real-world event data

during the pretraining stage of our approximate IL framework, which helps facilitate transfer from simulation to the real world.

Limitations Quadrator crashes at high speed with onboard event cameras are costly, preventing real-world evaluation of underperforming baselines. Even an external safety guard struggled to stabilize the drone at high speeds and provided no protection against obstacle collisions. Furthermore, while our vision-based end-to-end variant outperforms the best baseline, it still lags behind our approach using external state estimation. This gap could be reduced through auxiliary state-prediction tasks during offline training or by leveraging IMU data during offline training while omitting it during online fine-tuning within our approximate IL framework.

5 Conclusion

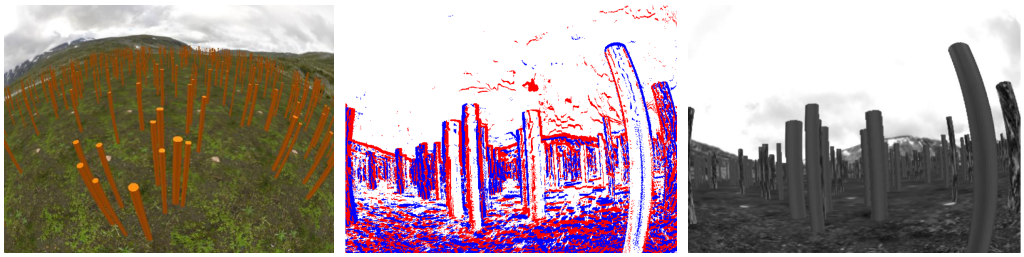
While event cameras offer compelling advantages, such as high temporal resolution and low latency, they remain underused in robotics due to the high computational cost of simulating events. In this work, we propose an approximate imitation learning framework that significantly reduces the number of required event renderings using offline datasets from simulation and the real world. To increase the generalization of the policy, we introduce an approximate student that predicts the actions of the event-based student using privileged state information. Since this state information can be simulated efficiently, the policy can be fine-tuned through online interactions without rendering events. To further accelerate the validation of event-based policies, we also propose a fully parallelized method for directly generating event representations from high-speed videos. Experiments in both simulation and the real world demonstrate that our approach enables fast and agile quadrotor flight through cluttered environments using a single event camera, without relying on intermediate representations. While we apply our approximate IL to event data, the framework naturally extends to other modalities that are expensive to simulate, such as tactile sensors, radar, or LiDAR.

Acknowledgments

This work was supported by the European Union's Horizon Europe Research and Innovation Programme under grant agreement No. 101120732 (AUTOASSESS), and the European Research Council (ERC) under grant agreement No. 864042 (AGILEFLIGHT).

6 Appendix

6.1 Simulation Environment



(a) (b) (c)

Figure 8: The simulation environment consists of tall cylindrical obstacles placed in a natural scene (a). The scene is captured by a simulated event camera mounted on the quadrotor (b). For reference, the corresponding grayscale images are also shown (c).

Figure 8 (a) illustrates the simulation environment, which features cylindrical orange obstacles embedded within a natural scene. High-framerate grayscale images, shown in Fig. 8 (c), are efficiently converted by our vectorized event rendering pipeline into event stacks, as depicted in Fig. 8 (b). This process preserves fine-grained temporal dynamics while enabling efficient event-based perception.

6.2 Network Architecture

Our framework employs three networks: a teacher, an event-based student, and an approximate student. The teacher is implemented as a three-layer multilayer perceptron (MLP) with ReLU activations and a final tanh layer, following standard designs used in quadrotor control [16]. To efficiently process event representations and capture temporal dynamics, the event-based student uses an EfficientNet-B0 encoder [60], without the final classification layer and with two recurrent GRU layers inserted after the third and fourth stages of the backbone. The encoded spatial features are flattened through global average pooling and subsequently passed through the *Projection Layers*, which are implemented as a two-layer MLP with ReLU activations. The encoder of the approximate student is a lightweight three-layer MLP with ReLU activations. Both students share the same action decoder, which employs a fully connected *Fusion Layer* to merge the event features $h_s^{t_i}$ with the features produced by the *Vector Layers* from auxiliary inputs, namely the direction command $\bar{v}_{\text{cmd}}^{t_i}$, the previous action $a_S^{t_{i-1}}$, and, optionally, the quadrotor state s_{t_i}. The fused features are then further processed by a two-layer *Action Head* with ReLU activations and a final tanh layer. The processing pipeline of the event student is shown in Fig. 9. Both auxiliary decoders, D and D_o, are implemented as four-layer MLPs with ReLU activations. All networks are trained using the Adam optimizer [61], with a learning rate scheduled via cosine annealing with warm restarts [62]. The input to the event-based student consists of event representations with a resolution of 213×160, obtained by downsampling the coordinates of an event stream originally at 640×480. Importantly, this integer-division downsampling preserves certain high-resolution spatial details that would not be captured by a native 213×160 event camera, which requires the original event representations to be rendered at 640×480.

6.3 Training and Evaluation Details

Training Offline training alternates between update steps on M3ED and simulated data. The offline loss weights are set to 1 for the distance-map loss, 100 for the teacher-student action MSE, 1 for the approximate-student action MSE, and 1 for the approximate-student feature L1 loss. During online training, the DAgger action loss has weight 1, while the offline alignment uses weights of 0.5 for the student-approximate-student action MSE and 1 for the corresponding feature L1 loss. For the offline alignment, we sample 2000 offline samples per DAgger iteration, with the sampled subset varying across iterations.

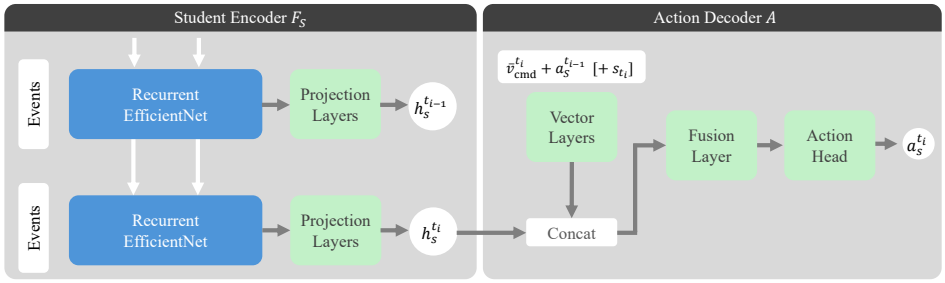


Figure 9: The event student receives events in the form of event representations, which are first encoded into features $h_S^{t_i}$ by the event student encoder F_S. The student encoder F_S consists of a recurrent EfficientNet [60] followed by the *Projection Layers*. These event features are then concatenated with the outputs of the *Vector Layers*, which encode auxiliary inputs, i.e., the direction command $\bar{v}_{t_i}$, the previous action $a_S^{t_{i-1}}$, and, optionally, the state information s_{t_i}. The combined features are fused in the *Fusion Layer*, and the final actions $a_S^{t_i}$ are produced by the *Action Head*.

Evaluation We use a single training seed and report the mean performance across 10 environments, following the protocol of [7, 49]. For a fair comparison, the online DAgger baselines, DAgger and BC+DAgger, are trained using the same number of rendering steps (10^6) as used to generate the simulated offline dataset. The key difference is that the online DAgger baselines require computationally expensive event rendering for every training run, whereas our approximate IL approach and other offline methods, such as BC, can reuse the offline dataset across runs.

6.4 Event Rendering

Training end-to-end robot policies with visual inputs in simulation requires a large number of environment steps, which is particularly problematic for event-based policies because training time scales with the cost of simulating events. This cost is high, as accurate event simulation requires rendering high-frame-rate videos that capture the fine temporal resolution needed to truthfully simulate event sequences.

Each event sequence consists of individual events represented as tuples $(\mathbf{x_k}, p_k, t_k)$, where $\mathbf{x_k}$ is the pixel location, $p_k \in \{+1, -1\}$ is the polarity, and t_k the timestamp. To simulate events, event generation is commonly approximated using the noise-free model of [6]. Specifically, once a high-frame-rate video is synthesized, events are generated by evaluating the change in logarithmic intensity $L = \log(I_t)$ at a single pixel $\mathbf{x_k}$ over time. An event is triggered at timestep t_k if the logarithmic intensity change from the last triggered event at timestep t_{k-1} exceeds a given contrast threshold C, as shown in Eq. 1.

$$\Delta L(\mathbf{x_k}, t_k) = L(\mathbf{x_k}, t_{k-1}) - L(\mathbf{x_k}, t_k) = p_k C. \tag{1}$$

Here, the polarity p_k indicates the direction of the intensity change. In popular video-to-event methods [63, 57], the frame rate of the underlying video is chosen such that the maximum optical flow between two frames does not exceed one pixel.

In our use case, the event stream is ultimately converted into a dense spatiotemporal representation, which is then processed by a lightweight neural network. Thus, generating a full event sequence only to later transform it into a dense representation introduces significant and unnecessary computational overhead. Inspired by [10], we follow the idea of directly generating event representations, effectively bypassing the intermediate step of synthesizing sparse event sequences. This design has two main advantages: (i) the direct generation of event representations reduces computation time and memory usage by avoiding explicit event sequence simulation, and (ii) it removes the need for high temporal fidelity, as the event representation uses a coarse discretization of the time dimension. In contrast to [10], which generates single-channel representations, our proposed approach truthfully constructs a multi-channel event tensor that accumulates the events in a given temporal window for

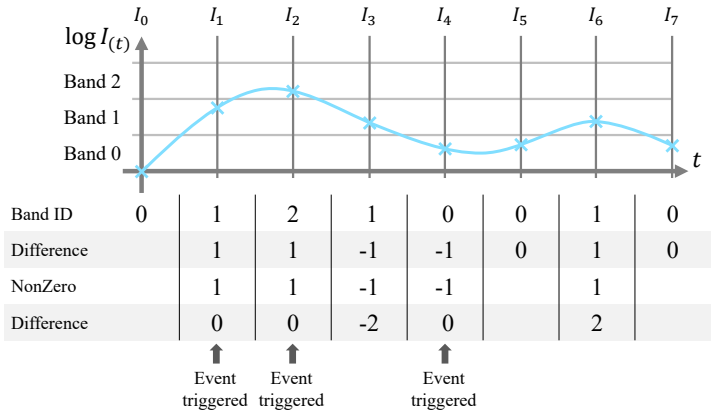


Figure 10: Our vectorized event generation first quantizes log intensities into bands and computes band differences between adjacent timesteps. Neighboring non-zero values are then differenced again. Events are triggered where this second subtraction yields zero.

each polarity [56]. Importantly, our method generates event representations for multiple videos in parallel using vectorized operations, significantly accelerating the simulation.

The core idea behind our event representation method is to leverage fast, vectorized operations, as illustrated in Fig. 10, which shows the log-intensity curve over time alongside the key processing steps. Given a high-frame-rate video with images $I_0, I_1, ...I_7$, we first compute the log-intensity for each frame and subtract the intensity of the initial frame I_0, establishing it as the reference. In the next step, each normalized log-intensity frame is floor-divided by the contrast threshold C, assigning each pixel at each timestep to a discrete *contrast band* based on its current intensity level. To detect band transitions, we compute the temporal difference of band indices between consecutive frames. Steps where the band index remains unchanged, i.e., the difference equals zero, indicate no threshold crossing and are discarded. Crucially, an event is only triggered when the intensity curve exits the current band in the opposite direction from which it previously entered. To identify this, we compute a second-order difference of the band indices over time, using only frames with valid, non-zero first-order differences. An event is triggered at timesteps where this second difference is zero, indicating that the logarithmic difference exceeded the contrast threshold C. The polarity of the triggered event is determined from the sign of the first band ID difference. To obtain the final event representation, a simple reshaping and summing along a given axis is required.

To account for multiple events being triggered at a given pixel within a single timestep, we take the absolute value of the band ID difference in the first differencing step, which corresponds to the number of threshold crossings. Additionally, to simulate a non-zero initial reference value, a constant offset can be subtracted from the log-intensity at the normalization stage, allowing for flexible initialization of the reference state.

Our proposed event generation method enables the parallel generation of event representations for multiple video streams through efficient vectorized operations implemented in PyTorch [64]. Except for a few corner cases related to specific initial conditions, the resulting event representations are numerically equivalent to those obtained by first generating full event sequences with ESIM [57] and subsequently converting them to dense representations. While our approach reduces average runtime by 34% compared to generating event representations with ESIM's CUDA-based implementation, its primary advantage lies in the significantly improved memory efficiency, illustrated in Fig. 11. Compared to ESIM, our method substantially reduces the maximum GPU memory usage, which ultimately determines whether event simulations at scale are feasible. Consequently, ESIM fails to scale to higher resolutions and larger numbers of environments, restricting the comparison in Fig. 11 to half the final resolution (240×320) and at most 15 environments. In contrast, our method supports

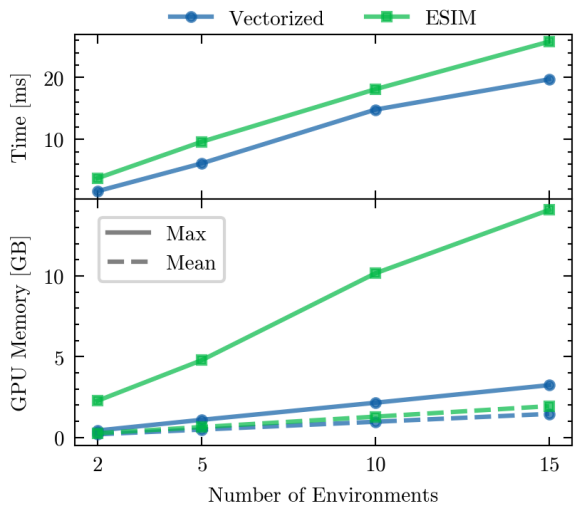


Figure 11: The maximum and mean GPU memory usage and the required computation time are shown for our vectorized event generation (Vectorized) and ESIM's GPU-based event generation (ESIM) across different numbers of environments. Our method leads to a 34% reduction in mean runtime (top) and significantly lowers peak GPU memory usage (bottom).

large-scale parallel simulation environments, making it well-suited for training event-based control policies at scale.

6.5 Teacher Training

The teacher policy is trained using Proximal Policy Optimization (PPO)[65] within the Flightmare simulator [50]. Its observation space includes the commanded direction in the local frame, the z-position of the quadrotor, orientation represented as a flattened rotation matrix, linear and angular velocities, and the previous action command. To perceive obstacles, the teacher also receives an angular distance map that encodes the closest obstacle distances across 10 angular bins. Each bin spans 11.25°, covering a 120-degree horizontal field of view. The episode terminates if the quadrotor leaves the simulation boundary, collides with an obstacle, or reaches the maximum number of steps. The teacher is trained for 8000 PPO update steps, with each update performed on rollouts of 250 steps collected from 100 parallel environments.

The total reward r_t combines several weighted components, with their respective factors λ_* omitted in Eq. 2 for clarity:

$$r_{\mathrm{t}} = r_{\mathrm{prog}} + r_{\mathrm{act}} + r_{\mathrm{br}} + r_{\mathrm{perc}} + r_{\mathrm{obs_dist}} + r_{\mathrm{crash}} \tag{2}$$

$$\begin{aligned}
r_{\mathrm{prog}} &= \tanh\left(\frac{\bar{v}_{\mathrm{quad}} \cdot \bar{v}_{\mathrm{cmd}}}{\|\bar{v}_{\mathrm{cmd}}\|} + 1\right) \\
&\quad \cdot \tanh\left(\|\bar{v}_{\mathrm{cmd}}\| - \|\bar{v}_{\mathrm{quad}} - \bar{v}_{\mathrm{cmd}}\| + 1\right) \\
&\quad \cdot \frac{1}{4} \min\left(\frac{\|\bar{v}_{\mathrm{quad}}\|}{\|\bar{v}_{\mathrm{cmd}}\|}, 1\right) - \left|v^z_{\mathrm{quad}}\right|, \\
r_{\mathrm{act}} &= -\|\bar{a}_{t_i} - \bar{a}_{t_{i-1}}\|, \\
r_{\mathrm{br}} &= -\|\bar{a}_{\omega}\|, \\
r_{\mathrm{perc}} &= \frac{\bar{v}_{\mathrm{cmd}} \cdot \bar{d}_{\mathrm{heading}}}{\|\bar{v}_{\mathrm{cmd}}\| \|\bar{d}_{\mathrm{heading}}\|}, \\
r_{\mathrm{obs_dist}} &= -\frac{1}{N_{\mathrm{obs}}} \sum_{i}^{N_{\mathrm{obs}}} \exp\left(-d_{\mathrm{obs}_i} - (\phi_{\mathrm{heading}} - \beta_{\mathrm{obs}_i})^2\right) \\
r_{\mathrm{crash}} &= -\|\bar{v}_{\mathrm{quad}}\| - 1 \quad \text{if crash ceiling, ground or obstacle,}
\end{aligned} \tag{3}$$

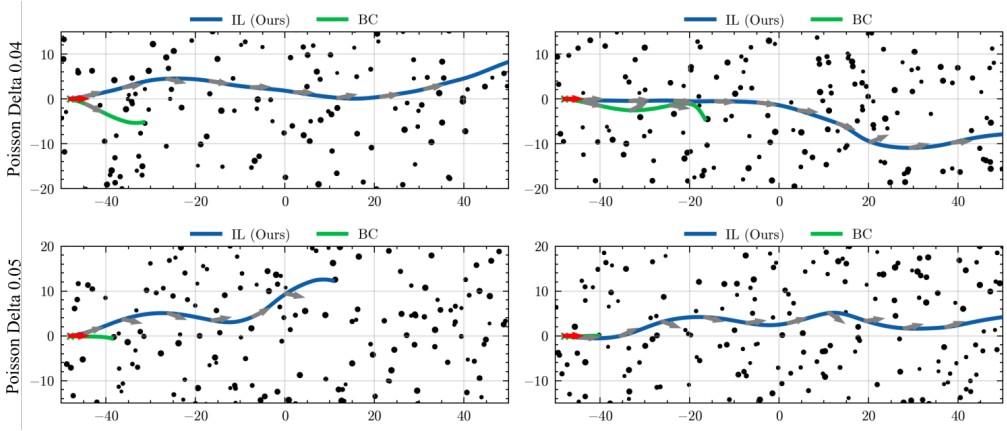


Figure 12: Our proposed method yields a policy that consistently flies longer trajectories in simulation than BC, the closest baseline applicable in the same training setting.

where r_{prog} encourages the quadrotor velocity $\bar{v}_{\text{quad}}$ to align with the commanded direction $\bar{v}_{\text{cmd}}$ to promote goal-directed flight, r_{act} penalizes large changes in consecutive actions $\bar{a}_{t_i}$ to ensure smooth control, r_{br} discourages high body-rate actions $\bar{a}_{\omega}$ to support stable flight dynamics, r_{perc} rewards alignment of the yaw direction of the quadrotor $\bar{d}_{\text{heading}}$ with the commanded direction, $r_{\text{obs_dist}}$ exponentially penalizes the distance d_{obs_i} to nearby obstacles N_{obs} inside the field of view $(\phi_{\text{heading}} - \beta_{\text{obs}_i})$, and r_{crash} applies a penalty upon collisions or when the quadrotor exits the simulation boundaries. Overall, the proposed teacher formulation yields a stable state-based policy capable of reliably flying in highly cluttered environments.

6.6 Simulated Flight Trajectories

Example flight trajectories in four evaluation environments are visualized in Fig. 12, comparing our proposed approximate imitation learning method against behavioral cloning (BC), the best-performing baseline trained under identical conditions.

6.7 Ablations

To validate our design choices, we conduct several ablation studies in simulation using the standard model, i.e., the event-based policy with access to state information. The evaluation environments are generated with a Poisson delta of 0.04.

6.7.1 Learnable Network Parameters

In this ablation, we examine which components of the network should remain trainable during approximate IL, as summarized in Tab. 4. In the default configuration, we fine-tune the *Vector Encoder* and the *Fusion Layer*, which leads to the longest mean flight distance before a crash. When all parameters of the action decoder are updated (Full Action Decoder), the policy still reaches $94.1\,\text{m}$ and maintains a high speed of $9.14\,\text{m}\,\text{s}^{-1}$. Reducing further the number of trainable parameters, either by updating only the biases (Only Biases) or by updating only the final layer (Last Layer), results in slower flight speeds and significantly shorter flight distances, e.g., $46.67\,\text{m}$ when updating only the last layer. This degradation can be explained by the reduced adaptability of the policy that results from limiting the set of trainable parameters. Overall, the best performance is achieved when updating the *Vector Encoder* and the *Fusion Layer*, which suggests that there is an optimal trade-off between adapting the policy during the approximate IL phase and preserving the knowledge learned during the pretraining stage.

Methods	Mean Distance [m]	Mean Velocity [$\frac{\mathrm{m}}{\mathrm{s}}$]
Only Biases	76.60	7.76
Last Layer	46.67	8.44
Full Action Decoder	94.10	9.14
Vector+Fusion (Default)	110.3	8.76

Table 4: Ablation of trainable action decoder parameters.

Methods	Success Rate	Mean Velocity [$\frac{\mathrm{m}}{\mathrm{s}}$]
w/o State	0.50	8.17
w/o Vel.	0.70	11.55
w/o Transl. Vel.	0.80	12.91
w/o Ang. Vel.	1.00	9.04
w State	1.00	8.76

Table 5: Ablation of varying state information levels.

6.7.2 Quadrotor State Observations

To assess the effect of the quadrotor state observations, we ablate policies that receive varying levels of state information, as reported in Tab. 5. Naturally, providing full state information leads to the highest robustness with a success rate of 1.0. The removal of angular velocities (w/o Ang. Vel.) has little impact, as the resulting policy still achieves a perfect success rate. In contrast, excluding translational velocities (w/o Transl. Vel.) reduces the success rate to 0.8 while increasing the mean flight speed to $12.91\,\mathrm{m\,s^{-1}}$, reflecting the difficulty in controlling the speed without absolute velocity observations. The differing impact of removing angular versus translational velocities can be explained by the ability of the event camera to infer rotational motion through its high temporal resolution, whereas the absolute scale information required to estimate translational velocity is inherently missing in monocular sensing. When both angular and translational velocities are removed (w/o Vel.), the success rate drops further to 0.7, approaching the performance obtained without any state information (w/o State), which reaches 0.5.

6.7.3 Distance Map Estimation

To provide insight into the angular distance map estimation, we visualize sample predictions alongside the corresponding ground truth in Fig. 13. Importantly, the distance maps and associated auxiliary decoders D and D_o are used only during the pretraining phase and are discarded during action inference of the event student. In Fig. 13 (A), we show the three ground truth distance bands together with the corresponding predictions for the M3ED dataset. Even though this sample comes from a hold-out validation split, the auxiliary decoder D accurately predicts the angularly binned distances in an automotive scene. On simulation data, the auxiliary decoder D_o reliably predicts gravity-aligned angular distance maps, as illustrated in Fig. 13 (B). Finally, Fig. 13 (C) shows a prediction of a gravity-aligned angular distance map in the real-world test environment onboard a hovering quadrotor. These qualitative examples demonstrate that the auxiliary decoders can robustly predict angular distance maps using only events, providing evidence that the feature space of the event encoder captures meaningful distance information.

6.8 Hardware Setup

We validate our approach in the real world using a quadrotor platform and onboard computational resources similar to those used in [49], as shown in Fig 14. The quadrotor is equipped with Hobbywing XRotor 2306 motors and 5-inch propellers, resulting in a total weight of approximately $780\,\mathrm{g}$. To obtain an accurate drone model in simulation, the inertial parameters and thrust mapping are identified, while selected Flightmare parameters are randomized to account for sim-to-real differences. Onboard computation is handled by an NVIDIA Jetson TX2, mounted on a ConnectTech Quasar carrier board. Our neural network outputs control commands in the form of collective thrust and angular rates, which are converted to individual rotor commands using the BetaFlight flight controller. For event-based perception, we use the DVXplorer Micro sensor with a VGA resolution of 640×480 pixels, weighing approximately $16\,\mathrm{g}$ without a lens. Although the DVXplorer Micro includes an onboard IMU, we do not use its inertial measurements in our approach. Using TensorRT, a forward pass of our event network processing 213×160 event representations takes on average

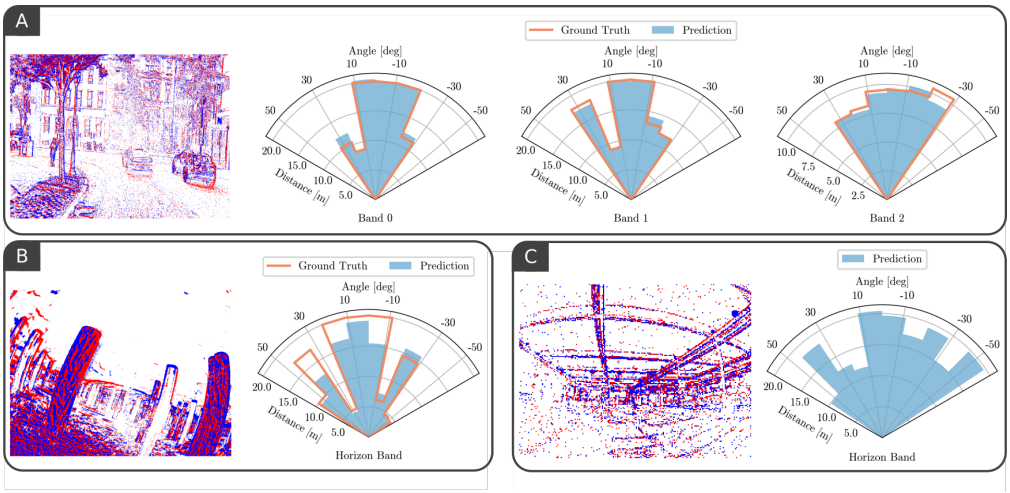


Figure 13: The ground truth distance maps and corresponding predictions are shown for the M3ED dataset (A), simulation data (B), and the real-world test environment (C). Importantly, the distance maps and associated auxiliary decoders are used only during pretraining and are not involved in action inference of the event student.

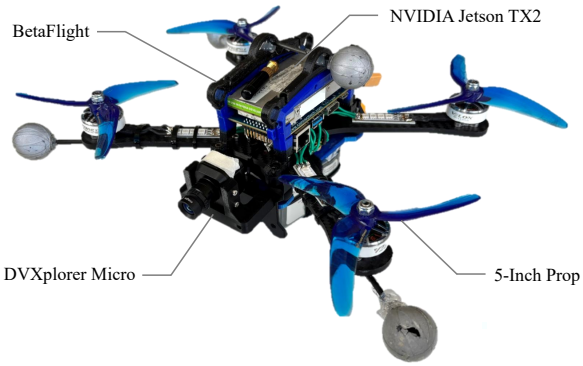


Figure 14: The platform used in the real-world experiments features an NVIDIA Jetson TX2 running the event-based student, which processes events from a DVXplorer Micro sensor.

$11.6\,\mathrm{ms}$ on the Jetson TX2, providing sufficient margin to meet the $20\,\mathrm{ms}$ requirement of the $50\,\mathrm{Hz}$ control frequency.

6.9 Extension Beyond Event Cameras

Our approximate imitation learning framework assumes that offline training establishes a shared task-relevant latent space between the approximate student and student, preserving factors such as motion, obstacle distance, and free space while discarding event-camera-specific appearance or lighting cues. Extending the framework to other sensor modalities, therefore, requires the approximate student and student observations to share sufficient task-relevant information, while allowing sensor-specific information to be removed during alignment. If such shared information is absent, the offline alignment can not establish a common task-relevant representation, and the online training phase with the approximate student may degrade student performance. In this case, however, the observations themselves may provide insufficient information to solve the task.

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2017.

[2] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[3] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. In *2019 international conference on robotics and automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[4] S. Sun, G. Cioffi, C. de Visser, and D. Scaramuzza. Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events. *IEEE Robotics and Automation Letters*, 2021.

[5] X. Wu, S. Chen, K. Sreenath, and M. W. Mueller. Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry. *IEEE Access*, 2022.

[6] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020. doi:10.1109/TPAMI.2020.3008413.

[7] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin. Learning vision-based agile flight via differentiable physics. *Nature Machine Intelligence*, 7(6):954–966, June 2025. ISSN 2522-5839. doi:10.1038/s42256-025-01048-0. URL http://dx.doi.org/10.1038/s42256-025-01048-0.

[8] Y. Ren, F. Zhu, G. Lu, Y. Cai, L. Yin, F. Kong, J. Lin, N. Chen, and F. Zhang. Safety-assured high-speed navigation for mavs. *Science Robotics*, 10(98):eado6187, 2025.

[9] D. Falanga, K. Kleber, and D. Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robotics*, 5(40):eaaz9712, 2020.

[10] A. Bhattacharya, M. Cannici, N. Rao, Y. Tao, V. Kumar, N. Matni, and D. Scaramuzza. Monocular event-based vision for obstacle avoidance with a quadrotor. In *8th Annual Conference on Robot Learning*, 2024.

[11] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[12] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, J. Humplik, M. Wulfmeier, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, et al. Learning agile soccer skills for a bipedal robot with deep reinforcement learning. *Science Robotics*, 9(89):eadi8022, 2024.

[13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26):eaau5872, 2019.

[14] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39(1):3–20, 2020.

[15] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt. On the role of the action space in robot manipulation learning and sim-to-real transfer. *IEEE Robotics and Automation Letters*, 2024.

[16] J. Xing, I. Geles, Y. Song, E. Aljalbout, and D. Scaramuzza. Multi-task reinforcement learning for quadrotors. *IEEE Robotics and Automation Letters*, 2024.

[17] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[18] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. ViNT: A foundation model for visual navigation. In *7th Annual Conference on Robot Learning*, 2023. URL https://arxiv.org/abs/2306.14846.

[19] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[20] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[21] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[22] J. Xing, L. Bauersfeld, Y. Song, C. Xing, and D. Scaramuzza. Contrastive learning for enhancing robust scene transfer in vision-based agile flight. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5330–5337. IEEE, 2024.

[23] B. Forrai, T. Miki, D. Gehrig, M. Hutter, and D. Scaramuzza. Event-based agile object catching with a quadrupedal robot. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12177–12183, 2023. doi:10.1109/ICRA48891.2023.10161392.

[24] Z. Wang, F. Cladera, A. Bisulco, D. Lee, C. J. Taylor, K. Daniilidis, M. A. Hsieh, D. D. Lee, and V. Isler. Ev-catcher: High-speed object catching using low-latency event-based neural networks. *IEEE Robotics and Automation Letters*, 7(4):8737–8744, 2022. doi:10.1109/LRA.2022.3188400.

[25] T. Delbruck and M. Lang. Robotic goalie with 3 ms reaction time at 4 *Frontiers in Neuroscience*, Volume 7 - 2013, 2013. ISSN 1662-453X. doi:10.3389/fnins.2013.00223. URL https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2013.00223.

[26] S. Stroobants, J. Dupeyroux, and G. C. de Croon. Neuromorphic computing for attitude estimation onboard quadrotors. *Neuromorphic Computing and Engineering*, 2(3):034005, 2022.

[27] S. Stroobants, C. De Wagter, and G. C. De Croon. Neuromorphic attitude estimation and control. *IEEE Robotics and Automation Letters*, 2025.

[28] F. Paredes-Vallés, J. J. Hagenaars, J. Dupeyroux, S. Stroobants, Y. Xu, and G. de Croon. Fully neuromorphic vision and control for autonomous drone flight. *Science Robotics*, 9(90): eadi0591, 2024.

[29] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermüller, D. Scaramuzza, and Y. Aloimonos. Evdodgenet: Deep dynamic obstacle dodging with event cameras. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10651–10657, 2020. doi:10.1109/ICRA40945.2020.9196877.

[30] S. Vemprala, S. Mian, and A. Kapoor. Representation learning for event-based visuomotor policies. *Advances in Neural Information Processing Systems*, 34:4712–4724, 2021.

[31] A. Vitale, A. Renner, C. Nauer, D. Scaramuzza, and Y. Sandamirskaya. Event-driven vision and control for uavs on a neuromorphic chip. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 103–109. IEEE, 2021.

[32] X. Zhang, J. Tie, J. Li, Y. Hu, S. Liu, X. Li, Z. Li, X. Yu, J. Zhao, Z. Wan, et al. Dynamic obstacle avoidance for unmanned aerial vehicle using dynamic vision sensor. In *International Conference on Artificial Neural Networks*, pages 161–173. Springer, 2023.

[33] K. F. Andersen, H. X. Pham, H. I. Ugurlu, and E. Kayacan. Event-based navigation for autonomous drone racing with sparse gated recurrent network. In *2022 European Control Conference (ECC)*, pages 1342–1348. IEEE, 2022.

[34] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, et al. Fast, autonomous flight in gps-denied and cluttered environments. *Journal of Field Robotics*, 35(1):101–120, 2018.

[35] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66):eabp9742, 2022.

[36] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE, 2015.

[37] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. Svo: Semidirect visual odometry for monocular and multicamera systems. *IEEE Transactions on Robotics*, 33(2): 249–265, 2016.

[38] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2):267–274, 2010.

[39] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart. Receding horizon" next-best-view" planner for 3d exploration. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 1462–1468. IEEE, 2016.

[40] F. Sadeghi and S. Levine. Cad2rl: Real single-image flight without a single real image. *Robotics: Science and Systems*, 2016.

[41] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 133–145. PMLR, 29–31 Oct 2018.

[42] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.

[43] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. In *International Conference on Robotics and Automation (ICRA)*, 2023. URL https://arxiv.org/abs/2210.03370.

[44] A. Sridhar, D. Shah, C. Glossop, and S. Levine. NoMaD: Goal Masked Diffusion Policies for Navigation and Exploration. *arXiv pre-print*, 2023. URL https://arxiv.org/abs/2310.07896.

[45] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620(7976):982–987, 2023.

[46] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza. Demonstrating agile flight from pixels without state estimation. *Robotics: Science and Systems (RSS)*, 2024.

[47] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza. Bootstrapping reinforcement learning with imitation for vision-based agile flight. *8th Conference on Robot Learning (CoRL)*, 2024.

[48] A. Romero, A. Shenai, I. Geles, E. Aljalbout, and D. Scaramuzza. Dream to fly: Model-based reinforcement learning for vision-based drone flight. *arXiv preprint arXiv:2501.14377*, 2025.

[49] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. Learning high-speed flight in the wild. In *Science Robotics*, October 2021.

[50] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Proceedings of the 2020 Conference on Robot Learning*, pages 1147–1157, 2021.

[51] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *IJCAI*, pages 4950–4957, 2018.

[52] N. Messikommer, J. Xing, E. Aljalbout, and D. Scaramuzza. Student-informed teacher training. *International Conference on Learning Representations*, 2025.

[53] E. Aljalbout, J. Xing, A. Romero, I. Akinola, C. Garrett, E. Heiden, A. Gupta, T. Hermans, Y. Narang, D. Fox, D. Scaramuzza, and F. Ramos. The reality gap in robotics: Challenges, solutions, and best practices. *arXiv preprint*, 2025.

[54] K. Chaney, F. Cladera, Z. Wang, A. Bisulco, M. A. Hsieh, C. Korpela, V. Kumar, C. J. Taylor, and K. Daniilidis. M3ed: Multi-robot, multi-sensor, multi-environment event dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 4015–4022, June 2023.

[55] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[56] 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*, 129(4): 900–920, 2021.

[57] H. Rebecq, D. Gehrig, and D. Scaramuzza. {ESIM}: an Open Event Camera Simulator. *Conf. on Robotics Learning (CoRL)*, 10 2018.

[58] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen. Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters*, 4(4):3529–3536, 2019.

[59] P. Florence, J. Carter, and R. Tedrake. Integrated perception and control at high speed: Evaluating collision avoidance maneuvers without maps. In *Algorithmic Foundations of Robotics XII: Proceedings of the Twelfth Workshop on the Algorithmic Foundations of Robotics*, pages 304–319. Springer, 2020.

[60] M. Tan and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[61] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *Int. Conf. Learn. Representations (ICLR)*, 2014.

[62] I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with restarts. *Int. Conf. Learn. Representations (ICLR)*, 2017.

[63] D. Gehrig, M. Gehrig, J. Hidalgo-Carrió, and D. Scaramuzza. Video to Events: Recycling video datasets for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2020.

[64] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[65] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.