

Supplementary material for Monocular Event-Based Vision for Dodging Static Obstacles with a Quadrotor

Notes

Please note that all code described as open-source or provided will be made available after the peer review process has concluded. Thank you for your patience.

S1 Additional simulation collision metrics

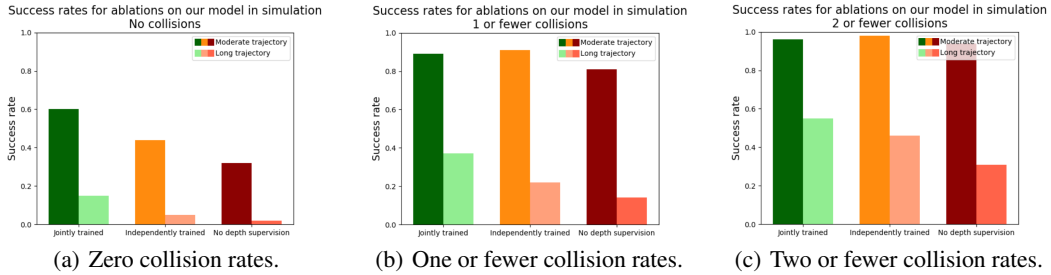


Figure S1: Rates for moderate and long trajectories in simulation of zero collisions (typically referred to just as “success rates”) and one and two collisions. On the long trajectory there is consistently better performance with the jointly-trained (proposed) model.

We present additional collision metrics from simulation rollouts for the proposed model and ablations described in Section 4.1 (jointly-trained, independently-trained, and no depth supervision models) in Figure S1. As we allow more collisions in the success rate calculation (typically “success rate” refers only to zero collisions, as in the paper) we observe a leveling out of performance among all models on the moderate trajectory, but performance remains consistently better on the long trajectory with the jointly trained model.

S2 Use of varying event camera hardware

Two quadrotor-event camera platforms were used, one with a Davis346 (resolution 260×346) and one with a Prophesee Gen3.1 VGA (resolution 480×640) (Figure 4). As mentioned in the text (Section 3.4), these two sensors have very different sensor and event-pixel characteristics; furthermore, the biases were left as the default values and not tuned for any experiment. To enable the simulation pre-trained policy to extend to a real-world Davis346 and further to the Prophesee camera, calibrated and time-synchronized event batches and depth images were gathered to fine-tune models with real data gathered from a hand-held device with both sensors (Figure S2). ROS was used to run both sensors in parallel while recording message data that includes ROS timestamps and corresponding events messages for either event camera (iniVation AG ROS driver [1] was used for running the Davis346, and Prophesee AI ROS driver [2] was used for running the Prophesee) as well as ROS timestamps and corresponding depth and “infrared1” messages from the D435 depth camera. Since the “infrared1” camera is pre-aligned with the depth images, we use these images to run multi-camera calibration. E2Calib [3] was used to generate images from either event camera



Figure S2: Example of rigidly-attached event camera (Prophesee Gen3.1 VGA) and depth camera (Intel Realsense D435) used to gather events batches and depth images for approximate post-processing calibration and time synchronization, for use in fine-tuning perception backbone from Figure 2. Note that an IR filter is used on the event camera lens so that we may enable the infrared emitter on the D435 for more-accurate indoor depth images.

at timestamps specified by the depth images (for both types, rosbag data needed to be converted to DVS EventArray message type [4, 5, 6]). With the images from E2Calib and “infrared1”, we use Kalibr [7] to perform multi-camera calibration.

As all perception models are sized according to the Davis346 resolution, for the Prophesee camera we center-crop a 260×346 sized event batch for input to the perception backbone $D(\theta)$. Note that while the simulated camera and corresponding Vid2E events output have different intrinsics than the Davis346 and Prophesee cameras, no alignment is performed during the sim-to-real transfer, and we expect the fine-tuning of the model to incorporate changes in lens geometry and distortion.

S3 State estimation for outdoor flight

As described in Figure 4, the Falcon250-Prophesee platform uses the VOXL board [8] for onboard state estimation. This board contains an independent computer, connected to the primary computer via an Ethernet cable. This visual-inertial odometry (VIO) module provides six degrees-of-freedom robot poses at 30Hz. We employed an Unscented Kalman Filter (UKF) that takes high-frequency IMU data in combination with these pose estimates to provide odometry at 150Hz. Low-level attitude and thrust controllers are run on PX4 open-source [9] control stack using the Pixhawk4 flight controller, with desired values provided by custom controllers running on the primary onboard computer.

References

- [1] iniVation AG. Dv-ros: Robotic operating system integration for inivation cameras, 2024. URL <https://gitlab.com/inivation/dv/dv-ros>. Accessed: 2024-06-10.
- [2] P. AI. Prophesee ros wrapper: Ros driver for prophesee event-based sensors, 2024. URL https://github.com/prophesee-ai/prophesee_ros_wrapper. Accessed: 2024-06-10.
- [3] M. Muglikar, M. Gehrig, D. Gehrig, and D. Scaramuzza. How to calibrate your event camera. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, June 2021.
- [4] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. A 240×180 130 db 3 μ s latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, 2014.
- [5] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128×128 120db 15 μ s latency asynchronous temporal contrast vision sensor. *IEEE journal of solid-state circuits*, 43(2):566–576, 2008.
- [6] E. Mueggler, B. Huber, and D. Scaramuzza. Event-based, 6-dof pose tracking for high-speed maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2761–2768. IEEE, 2014.
- [7] J. Maye, P. Furgale, and R. Siegwart. Self-supervised calibration for robotic systems. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 473–480. IEEE, 2013.
- [8] ModalAI VOXL Technical Docs. URL <https://docs.modalai.com/docs/datasheets/voxl-datasheet>.
- [9] L. Meier, D. Honegger, and M. Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6235–6240. IEEE, 2015.