# MonoNav: MAV Navigation via Monocular Depth Estimation and Reconstruction

Nathaniel Simon
*Mechanical and Aerospace Engineering*
*Princeton University*
Princeton, NJ
nsimon@princeton.edu

Anirudha Majumdar
*Mechanical and Aerospace Engineering*
*Princeton University*
Princeton, NJ
ani.majumdar@princeton.edu

***Abstract*—The small form factor of micro aerial vehicles (MAVs) makes them highly appealing for autonomous inspection, exploration, and mapping of constrained indoor environments. However, a major challenge with deploying the smallest of these platforms ($\leq$ 100 g) is their inability to carry sensors that provide high-resolution metric depth information (e.g., LiDAR or stereo cameras). Equipped only with a monocular camera and inertial measurement unit, such systems currently rely on end-to-end learning or heuristic approaches that directly map images to control inputs, and struggle to fly fast in unknown environments. In this work, we ask the following question: using *only* a monocular camera, can we create metrically accurate depth maps to leverage the powerful path planning and navigation approaches employed by larger state-of-the-art robotic systems to achieve robust autonomy in unknown environments? We present MonoNav: a fast 3D reconstruction and navigation stack for MAVs that leverages recent advances in depth prediction neural networks to enable metrically accurate 3D scene reconstruction from a stream of monocular images and poses. MonoNav uses off-the-shelf pre-trained monocular depth estimation and fusion techniques to construct a map, then searches over motion primitives to plan a collision-free trajectory to the goal. In extensive hardware experiments, we demonstrate how MonoNav enables the Crazyflie (a 37 g MAV) to navigate fast (0.5 m/s) in cluttered indoor environments.**

***Index Terms*—MAV, monocular depth estimation, 3D reconstruction, obstacle avoidance**

## I. Introduction and Related Work

The smallest class of unmanned aerial vehicles (UAVs), referred to as MAVs ($\leq$ 100 g), are well-suited for constrained indoor applications such as inspection, exploration, and mapping. However, their small size and weight restricts their ability to carry sensors that provide high-resolution metric depth information (e.g., LiDAR or stereo cameras). Instrumentation is typically limited to an inertial measurement unit (IMU) for orientation and acceleration, an optical flow camera and range sensor for position and velocity, and a forward-facing monocular camera.

To address the challenge of monocular vision-based navigation, one can train an end-to-end model which takes as input a color image and outputs a velocity setpoint. In [1], a deep convolutional neural network trained entirely in simulation is able to navigate a MAV through hallways in the real world.

N. Simon is supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-2039656.

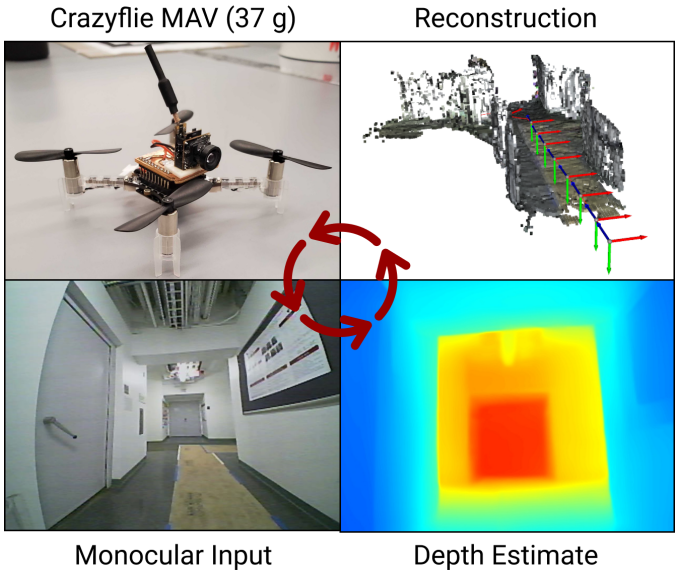


Fig. 1. MonoNav is a monocular navigation stack that leverages metric depth estimation, off-the-shelf fusers, and conventional planning to enable fast MAV navigation in constrained indoor environments.

To improve the generalization of such systems, [2] and [3] propose general-purpose policies that are trained and deployed across a range of environments and embodiments. However, the resulting policies are limited in the speed and robustness with which they can operate in unknown environments, and are also constrained to planar motion. Another approach is to leverage miniature sensors such as a 2.28 g ToF sensor for depth sensing [4], and a 4.4 g GAP8 embedded processor for end-to-end vision-based collision prevention [5]. While promising, the low-resolution depth information limits the ability to operate in cluttered environments.

In this work, we ask the following question: using *only* a monocular camera, can we obtain depth maps with sufficient metric accuracy to enable local 3D reconstruction of the MAV's environment? Such an ability would enable use of motion planning and navigation techniques used by larger state-of-the-art UAVs [6] [7]. We hypothesize that a modular pipeline consisting of depth estimation, local mapping via

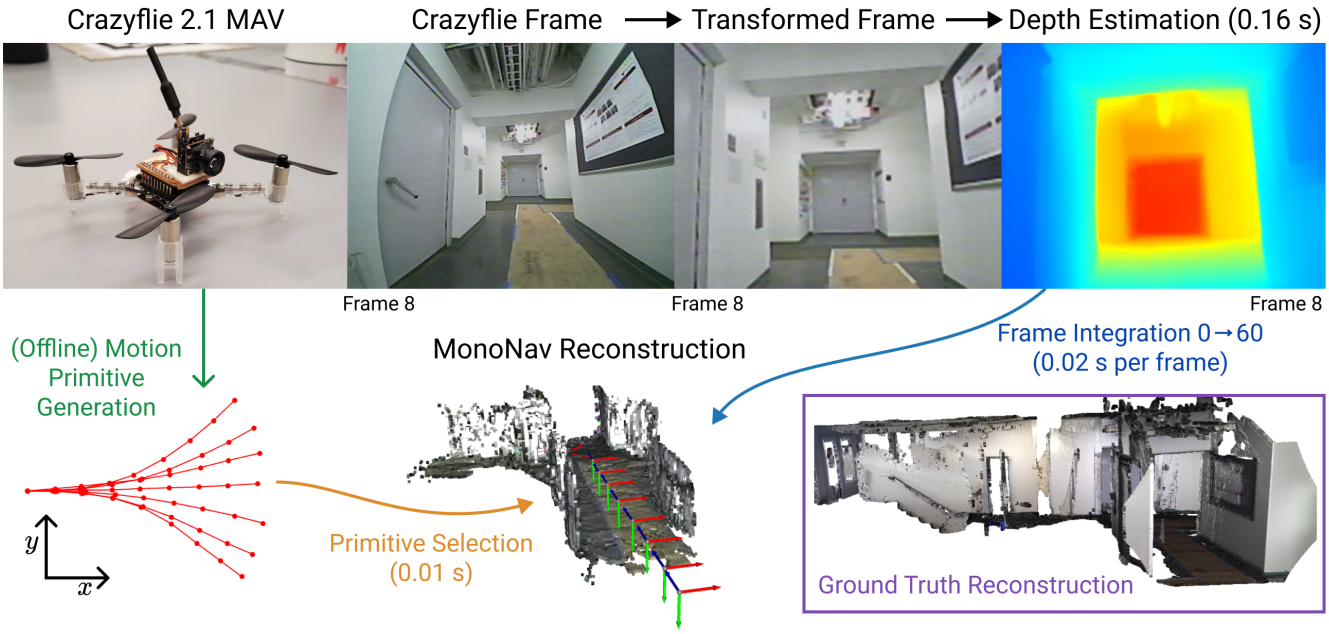


Fig. 2. MonoNav converts a series of RGB images into depth estimates (top, left to right), then fuses them into a 3D reconstruction (bottom middle). MonoNav then selects from collision-free motion primitives (bottom left) to navigate to a goal position.

fusion, and planning will enable significantly faster flight and more robust generalization to unseen environments [8]. In addition, this modular approach allows one to directly leverage improvements in depth estimation and motion planning without having to retrain an end-to-end policy from scratch. Finally, such an approach affords the ability to easily incorporate new objectives into the navigation stack (e.g., tracking a target object or constraining the drone's camera angle for cinematic applications).

### A. Statement of Contributions

Our primary contribution in this work is to demonstrate that, surprisingly, a monocular system combined with state-of-the-art depth estimation techniques can perform local 3D reconstruction with sufficient quality to enable fast MAV navigation in unknown environments. We present MonoNav: a navigation stack that leverages pre-trained transformer-based models for monocular depth estimation [9] [10] [11] in combination with off-the-shelf fusion and planning techniques. To enable the use of pre-trained models without any fine-tuning, we propose an image processing pipeline that minimizes domain shift by performing lightweight image transformations. We perform experiments demonstrating that our framework enables the Crazyflie (a 37 g MAV) to navigate fast (0.5 m/s) in cluttered indoor environments with fully onboard sensing and offboard computation. To our knowledge, MonoNav is the first monocular navigation stack implementation with distinct mapping and planning stages evaluated in hardware.

## II. Technical Approach

MonoNav is comprised of concurrent monocular mapping and planning processes (Fig. 2), which we describe below. To operate real-time, MonoNav assumes access to an offboard computer (e.g., a ground station or a larger ground-based robot).

### A. Monocular depth estimation and mapping

The monocular mapping process is broken into two stages: metric depth estimation and fusion. We leverage recent advances in monocular depth estimation [9] [10], which produce depth estimates from a single image [9] or a short history of past images and robot poses [10]. One of our key goals is to use *pre-trained* models without any fine-tuning. However, this is challenging due to the domain shift that arises from the difference between the MAV's camera intrinsics and the camera used for training the depth estimation models (to decrease latency, MAV cameras typically use fish-eye lenses). To tackle this, we propose a lightweight image pre-processing step (Fig. 2 - top center) that transforms a source image from the MAV's camera to a target image that appears *as though* it was taken with the camera used to train the pre-trained models; this can be achieved with a standard image processing library since both cameras intrinsics are known. We then pass the transformed image to a pre-trained model which performs monocular depth estimation.

In contrast to prior techniques that only produce depth images up to an unknown scaling factor, recent approaches [9] [10] produce absolute estimates of depth for every pixel. MonoNav uses ZoeDepth [9] for per-frame metric depth estimation. Combined with the drone's pose estimates from optical flow odometry, this enables the use of off-the-shelf depth fusion [12] to create a Truncated Signed Distance Function (TSDF) representation of the environment. We represent the 3D map using Open3D's VoxelBlockGrid representation, a

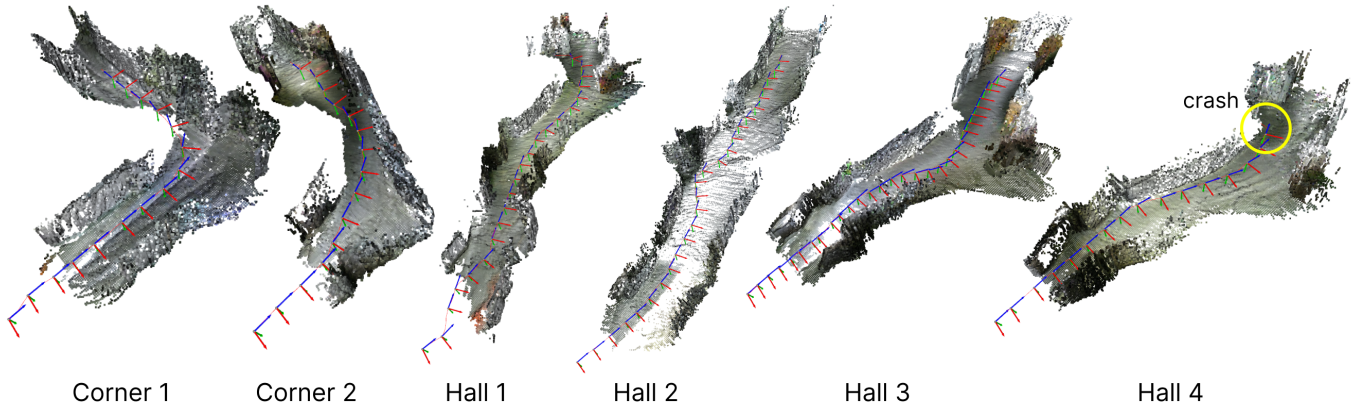


Fig. 3. MonoNav produces metric reconstructions for planning real-time, enabling navigation and collision avoidance indoors. Six representative scenes are shown above, including corners and longer hallways. Coordinate frames represent the drone's pose as flown through the trajectory.

globally sparse yet locally dense data structure which first discretizes the world coarsely into blocks, and stores blocks containing surfaces in a hashmap [13]. These blocks are further divided into voxels, which have TSDF weights and values. We perform TSDF fusion on each collected image to construct a local map. This local fusion process corrects for per-frame errors in depth estimation and also provides a memory of previously seen portions of the environment.

### B. Navigation

At each timestep, the robot has access to the map in the form of a VoxelBlockGrid. For collision-free navigation towards a goal, we use motion primitives; in principle, other planning approaches (e.g., A*, RRT*) could work.

The motion primitives and open-loop velocity setpoints are generated in a single offline step and stored in a trajectory library. From a desired constant speed $V$, maximum yawrate $A$, and horizon $T$, we define our motion primitives as Dubins' car inputs, with forward velocity $\dot{x}_{\text{sp}}(t) = V$ and yawrate $\dot{\psi}_{\text{sp}}(t) = A \sin(\pi t/T)$. This ensures that yawrates are zero at the beginning and end of each primitive for smooth flight. We integrate the inputs to determine the spatial trajectory used in planning. By varying $A$, we generate our library of primitives.

At runtime, the robot considers the set $\mathcal{T}$ of available trajectories $\tau \in \mathbb{R}^{n\times 3}$, each consisting of $n$ position waypoints. We also define the set $\mathcal{V}_o$ of occupied voxel coordinates $v_o \in \mathbb{R}^3$ in the VoxelBlockGrid, as well as the minimum distance $D(\tau, x)$ from any point along the trajectory $\tau$ to a coordinate $x \in \mathbb{R}^3$:

$$D(\tau, x) = \min_{0\le i<n} ||\tau_i - x||_2. \tag{1}$$

At each navigation step, we select a motion primitive according to the following objective function, which takes into account the goal position, $x_g \in \mathbb{R}^3$, and the minimum acceptable distance to any obstacle, $c \in \mathbb{R}_{>0}$:

$$\tau^* = \arg\min_{\tau\in\mathcal{T}} D(\tau, x_g) \quad \text{subject to} \quad D(\tau, v_o) \ge c, \quad \forall v_o \in \mathcal{V}_o. \tag{2}$$

In practice, we determine the set $\mathcal{V}_o$ by filtering all voxels in the VoxelBlockGrid by thresholds for weight and TSDF value. We exhaustively compute the distances from all trajectory points to the goal position and to every occupied voxel.

## III. Experiments

We implement MonoNav on the Crazyflie 2.1, a MAV configured as in [14] [15]. The Crazyflie is outfitted with a Flow deck v2 for position and velocity estimation and a Wolfwhoop WT05 RGB camera. Our offboard computer, which has a GeForce RTX 4090 GPU, communicates with the MAV and receives the analog video stream over radio. The Wolfwhoop camera suffers from significant 'barrel distortion' due to its fish-eye lens; we transform this image to the desired camera intrinics using OpenCV's undistortion and warp affine functions.

In extensive hardware experiments, we test MonoNav in 15 runs across 10 unique constrained hallway settings. These settings vary in complexity, ranging from straight sections, T-intersections, curved walls, and open spaces with columns. We defined the set of motion primitives (Fig. 2, bottom left) by $T = 1.0$ s, $V = 0.5$ m/s and $A \in \{-0.7 + 0.2\bar{3}\}_{i=0}^{6}$ rad/s. We set the distance threshold to $c = 0.5$ m and the goal position to $x_g = (-5, -0.4, 10)$. The camera has a measured lag of 0.12 s, per-frame depth estimation with ZoeDepth takes 0.11-0.16 s, fusion takes 0.02 s, and motion primitive selection takes 0.01 s. Camera readings, depth estimation, and integration occur at 3-4 Hz and replanning occurs at 1 Hz. It should be noted that both fusion and planning take longer as more voxels are added to the map.

## IV. Results

In 15 runs across 10 unique indoor settings, (six of which are shown in Fig. 3), MonoNav navigates successfully and avoids obstacles. Of the 15 runs, MonoNav crashed once (Fig. 3 Hall 4), and was prematurely terminated once (Fig. 3 Hall 1). In both cases, MonoNav turned into a wall or dead-end that was previously occluded and thus not perceived as an

obstacle (and is hence missing from the reconstruction). The goal position $x_g = (-5, -0.4, 10)$ induced a leftward bias into the navigation, which is reflected in the trajectories.

Of the errors in reconstruction, most are caused by noise in the camera feed. Discolored blobs in frames, though temporary, are often integrated as occupied voxels, as are overexposed pixels when the MAV is exposed to a bright light. This can cause early self-termination as the MAV avoids a phantom obstacle.

### A. Scheduled Experiments

In addition to demonstrating MonoNav's performance in challenging indoor environments, we propose the following experiments: A) Compare MonoNav against a baseline monocular navigation approach. To demonstrate the robustness and performance of MonoNav, it should be compared against state of the art end-to-end monocular navigation approaches [3] under identical conditions. B) Compare reconstruction quality against baseline depth estimation pipelines. MonoNav uses ZoeDepth, which infers depth from a single image. Other methods such as SimpleRecon [10] use a series of frames, poses, and metadata to infer depth. Such 'multi-view stereo' approaches could improve MonoNav's reconstruction accuracy.

### B. Conclusion

We introduce MonoNav, a navigation stack for monocular robots that generates a metric reconstruction using pre-trained depth estimation models and off-the-shelf fusion methods, enabling the use of conventional path planning techniques for collision-free navigation. We use MonoNav to enable a 37 g MAV to navigate numerous indoor environments at 0.5 m/s.

## REFERENCES


[1] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[2] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine, "Gnm: A general navigation model to drive any robot," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7226–7233.

[3] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine, "ViNT: A Foundation Model for Visual Navigation," *arXiv preprint arXiv:2306.14846*, 2023.

[4] V. Niculescu, H. Müller, I. Ostovar, T. Polonelli, M. Magno, and L. Benini, "Towards a multi-pixel time-of-flight indoor navigation system for nano-drone applications," in *2022 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE, 2022, pp. 1–6.

[5] D. Palossi, F. Conti, and L. Benini, "An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs," in *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 604–611.

[6] S. Tang and V. Kumar, "Autonomous flight," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 29–52, 2018.

[7] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, p. eabg5810, 2021.

[8] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *Science Robotics*, vol. 8, no. 79, p. eadf6991, 2023.

[9] S. F. Bhat, R. Birkl, D. Wofk, P. Wonka, and M. Müller, "Zoedepth: Zero-shot transfer by combining relative and metric depth," *arXiv preprint arXiv:2302.12288*, 2023.

[10] M. Sayed, J. Gibson, J. Watson, V. Prisacariu, M. Firman, and C. Godard, "SimpleRecon: 3D reconstruction without 3D convolutions," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*. Springer, 2022, pp. 1–19.

[11] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 3, pp. 1623–1637, 2020.

[12] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE international symposium on mixed and augmented reality*. Ieee, 2011, pp. 127–136.

[13] W. Dong, Y. Lao, M. Kaess, and V. Koltun, "Ash: A modern framework for parallel spatial hashing in 3D perception," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 5, pp. 5417–5435, 2022.

[14] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, "Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 6008–6014.

[15] A. Majumdar, "Introduction to Robotics at Princeton," *irom-lab.princeton.edu*.