3D Gaussian Splatting for Learned Drone Navigation

Daniel Kim[1], Yikuan Fang[2], Wenshan Wang[2], Sebastian Scherer[2]

***Abstract*—Autonomous drone navigation through unstructured environments is foundational for field robotics. While depth-based flight policies generalize effectively across scenes and transfer reliably from simulation to reality, physical depth sensors impose severe payload, power, and cost penalties. Monocular RGB cameras bypass these hardware constraints, but current RGB policies tend to overfit to their specific training environments and generalize poorly. To provide the photorealistic visual diversity a monocular policy needs, we reconstruct diverse scenes from the TartanAir dataset into fully interactive, flyable environments using 3D Gaussian splatting, jointly supervised with RGB and depth losses against ground truth. We validate that this simulator remains photorealistic and geometrically accurate at novel viewpoints well beyond the original capture trajectory, making it a reliable environment for policy training. As a demonstration of this simulator, we train goal-conditioned navigation policies via privileged imitation learning using either monocular RGB or depth observations, and benchmark both against three depth-based baselines from the literature in Isaac Sim. Our depth policy achieves competitive success rates, while our RGB policy currently trails the depth-based baselines, a gap we attribute to the limited visual diversity of the current training scene library rather than a fundamental limitation of monocular navigation.**

I. Introduction

An end-to-end navigation policy replaces the classical map, plan, and control pipeline with a single network that maps drone state and an onboard observation to a motion command that a lightweight low-level controller tracks (Fig. 1), removing the sequential processing and compounding errors that separate perception, mapping, and planning modules introduce.

Depth has been the dominant observation modality for these policies: a depth image carries far less appearance variation between simulation and reality than an RGB image does, so policies trained on depth transfer zero-shot to real, previously unseen environments [1]–[4]. However, depth sensors add payload, power, and cost, and degrade outdoors on textureless surfaces and at range.

Monocular RGB cameras avoid those hardware penalties and are already ubiquitous on drones, but the appearance gap between simulation and reality is much larger and the modality suffers from scale ambiguity, so RGB policies tend to overfit to their training environments. A decade of RGB-specific work has nonetheless remained constrained relative to depth, closing the gap only by narrowing the task or the training environment, never the appearance gap itself [5]–[10].

[1]Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA. [2]Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.

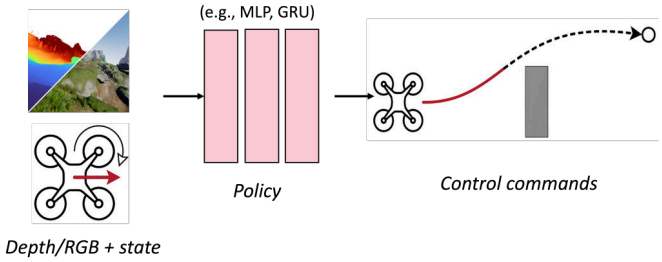


Fig. 1: An end-to-end navigation policy maps drone state and an onboard observation, either depth or RGB, to a motion command that a low-level controller tracks.

We attack this gap with two contributions. First, we build a 3D Gaussian splatting pipeline that converts captured RGB, metric depth, and pose data from TartanAir [11] into a library of flyable, photorealistic environments, jointly supervised with RGB and depth losses, and validate that its fidelity holds at novel viewpoints far from the capture trajectory. Second, we train goal-conditioned navigation policies inside this simulator via privileged imitation learning, using either RGB or depth observations, and benchmark both against three depth-based baselines from the literature as a demonstration of what the simulator enables.

II. Related Work

Two observation modalities dominate end-to-end navigation policies: depth generalizes across scenes but requires a costly sensor, while monocular RGB needs no such sensor but overfits to its training scenes. We review depth-based policies (Sec. II-A), the RGB policies that have narrowed some other axis of the problem instead (Sec. II-B), and 3D Gaussian splatting (Sec. II-C), the reconstruction method we use to attack the appearance gap directly.

A. Depth-Based Navigation Policies

Because the depth modality generalizes well without requiring photorealistic appearance, simulation environments for these policies have stayed comparatively simple: obstacle courses built from primitives such as walls, cylinders, gates, and scattered boxes suffice to teach generalizable obstacle avoidance [1]–[3]. These works differ mainly in training algorithm. DiffAero backpropagates directly through a fully differentiable dynamics and rendering pipeline, combining analytic gradients with reinforcement learning to train a policy

through primitive obstacle fields [2]. A privileged reinforcement learning policy trained with time-of-arrival maps as privileged information routes around large obstacles that local depth alone cannot resolve [3]. A third policy instead imitates a privileged expert through supervised regression with DAgger-style data aggregation rather than reward-driven exploration, transferring zero-shot from simple simulated obstacle courses to real forests, snow, and collapsed buildings [1].

B. Monocular RGB Navigation Policies

Camera-only navigation has been pursued for over a decade, typically by narrowing a different axis of the problem rather than closing the appearance gap directly. Early work imitates a human pilot to learn reactive obstacle avoidance, but only by steering laterally at fixed speed and altitude through forest corridors [5]. CAD2RL trades that task constraint for an appearance one, training collision avoidance purely on domain-randomized, low-fidelity CAD renders with randomized textures that sidestep photorealism but restrict flight to indoor hallways [6]. DroNet transfers steering and collision-probability control from ground-vehicle driving and cycling data rather than learning full goal-conditioned 3D navigation, trading task generality for data never collected for flight [7]. Each narrows a different axis: task, appearance realism, or training data.

C. 3D Gaussian Splatting for Robot Navigation

3D Gaussian splatting represents a scene as a set of anisotropic 3D Gaussians, each carrying a position, covariance, opacity, and color, and renders novel views by differentiably rasterizing and alpha-compositing these Gaussians onto the image plane [12]. Because rendering stays fully differentiable and reconstruction requires only posed RGB images, a scene can be captured, reconstructed, and rendered in real time, at a photorealistic quality competitive with neural radiance fields. These properties have drawn robotics research well beyond navigation. Reconstructed Gaussian scenes have supervised policy learning for ground robots [13] and legged locomotion [14], trained drone controllers to reach visual targets or fly through gates [8]–[10], and served purely as a geometric and photometric map for classical trajectory planning and RGB-only localization [15]. Across these applications, splatting functions as a fast, differentiable, photorealistic stand-in for the real world, whether as a training environment, a planning map, or a source of synthetic supervision.

III. Method

A. 3D Gaussian Splatting Simulation Environment

Fig. 2 summarizes the pipeline: we reconstruct each TartanAir [11] scene, which pairs diverse indoor, outdoor, urban, and natural sequences with ground-truth RGB, depth, and pose, as a set of 3D Gaussians using a differentiable rasterizer [12] optimized against that scene's captured RGB and metric depth; policies then train inside the result by privileged imitation learning (Sec. III-B, Fig. 4). For a rendered view with RGB $\hat{I}$, expected depth $\hat{D} = \sum_i \mathcal{T}_i \alpha_i d_i$ (transmittance

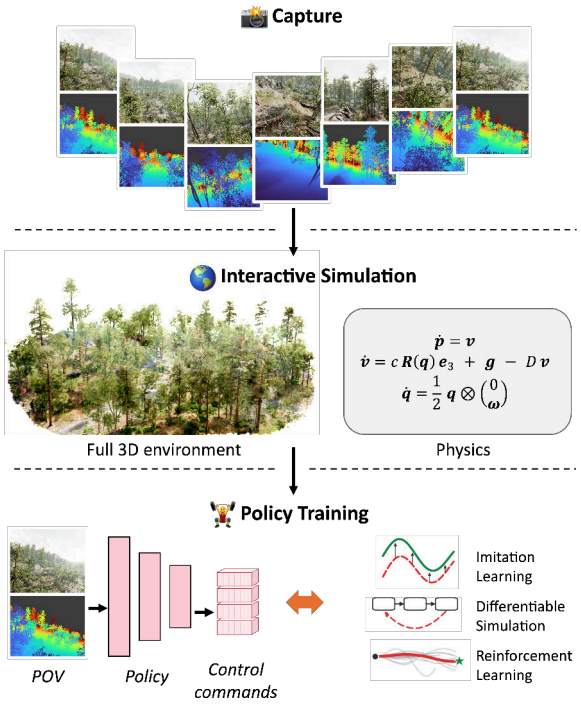


Fig. 2: System overview. Captured TartanAir views (top) reconstruct into an interactive Gaussian-splat environment with quadrotor dynamics (bottom left), inside which a policy trains end to end from RGB or depth to control commands (bottom right).

$\mathcal{T}_i = \prod_{j<i}(1-\alpha_j)$ over depth-sorted Gaussians), and the corresponding ground truth I, D, the training objective adds a metric-depth term to the photometric one (Fig. 3):

$$\mathcal{L} = \mathcal{L}_{\text{photo}} + \lambda_d \mathcal{L}_{\text{depth}}, \tag{1}$$

where $\mathcal{L}_{\text{photo}}$ is the standard 3DGS photometric loss, an ℓ_1 pixel error blended with a structural-similarity term [12]. $\mathcal{L}_{\text{depth}}$ is a log-space ℓ_1 error over valid pixels $V = \{p : \epsilon < D_p < D_{\text{far}}\}$ (ϵ=10^{-3} m, D_{far}=150 m),

$$\mathcal{L}_{\text{depth}} = \frac{1}{|V|} \sum_{p \in V} \left| \log \max(\hat{D}_p, \epsilon) - \log \max(D_p, \epsilon) \right|, \tag{2}$$

upweighted near depth discontinuities so thin structures like branches and railings survive. The log space keeps near-field geometry (precisely what a policy must avoid) from being drowned out by the far field, since the same relative error costs the same at any range. A perceptual (LPIPS) term adds fine detail late in training. We ablate this recipe against a photometric-only baseline and against the 2D Gaussian splatting representation [16] in Sec. IV-D.

From each scene's reconstructed geometry we extract a Euclidean signed-distance field (ESDF) on a voxel grid, queried at arbitrary points via differentiable trilinear interpolation. From a query point $\mathbf{x}$ with signed distance $d(\mathbf{x})$, we derive two collision signals used throughout training and evaluation:

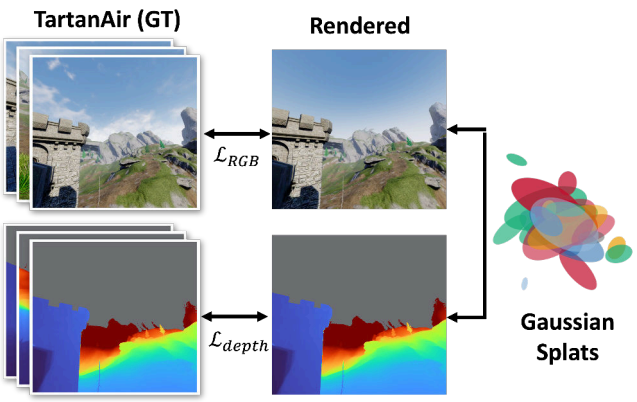


Fig. 3: Reconstructing a single scene. Stacked ground-truth RGB and depth views (left) supervise a set of 3D Gaussians (right), whose rendered RGB and depth (middle) are compared against held-out ground truth.

a hard indicator $\mathbb{1}[\,d(\mathbf{x}) < r\,]$ at the platform's physical radius r, and a differentiable soft barrier

$$\text{soft}(\mathbf{x}) = \tfrac{1}{\beta}\,\text{softplus}\big(\beta\,(r + \delta - d(\mathbf{x}))\big), \tag{3}$$

with sharpness β=4.0 and margin δ=1.0 m; it is near zero in free space and grows smoothly, with its gradient pointing away from the surface, as the platform nears or penetrates an obstacle. We set r=0.354 m, the platform's measured propeller-tip radius, throughout, shared with the policy's cost loss (Sec. III-B) and the evaluation criterion (Sec. IV).

We simulate quadrotor dynamics with a standard 13-dimensional body-rate model: state $\mathbf{s} = (\mathbf{p}, \mathbf{v}, q, \boldsymbol{\omega})$ (position, velocity, attitude quaternion, body-frame angular rate), action $\mathbf{u} = (c, \boldsymbol{\omega}_{\text{cmd}})$ (mass-normalized collective thrust along body $+z$ and commanded body rates), rigid-body translation under gravity and linear drag, quaternion kinematics, and a first-order lag τ_ω=0.05 s on the rate response. We integrate with fourth-order Runge–Kutta at 50 Hz. During training we randomize the rendered observation itself with multiplicative-plus-additive depth noise, depth dropout, RGB pixel noise, and brightness jitter, so the student never sees a clean image twice.

B. Imitation Learning Policy

Our policy architecture and multi-hypothesis imitation objective follow Loquercio et al. [1], with three changes: recovery data gathered by re-rendering logged near-miss states offline, rather than by online expert relabeling as in their DAgger loop; a cost target that evaluates clearance on the predicted trajectory itself via the differentiable ESDF of Sec. III-A; and pose-perturbation augmentation bounded by each scene's measured reconstruction fidelity.

Training proceeds in two stages. First, a depth-input expert policy [2] flies in simulation alongside a multi-hypothesis point-mass trajectory optimizer; from each rollout in which the expert reaches the goal with positive clearance, we sample labeled states every 0.4 s, each carrying its K best expert trajectories in cost order. Second, a student policy trained on this corpus (described below) flies closed-loop without a safety guard, and we re-query the expert at the states preceding its near-misses (clearance below 1.2 m, sampled 0.5–2.5 s before the event), re-rendering each logged pose offline so recovery is demonstrated from exactly those states. The student is retrained from scratch on the combined corpus, and we report that recovery-augmented policy throughout.

The student is a multi-hypothesis trajectory network adapting the mixture policy head of [1]: an ImageNet-pretrained MobileNetV2 backbone encodes the RGB or depth observation, a small MLP encodes the body-frame velocity and goal-direction vector, and a linear head decodes their concatenation into M=3 hypotheses, each a body-frame trajectory $\hat{T}_m \in \mathbb{R}^{T\times 3}$ of T=10 waypoints at 0.1 s spacing (a 1.0 s lookahead) paired with a predicted cost $\hat{c}_m$.

Training uses the relaxed winner-takes-all objective of [1]: each expert trajectory votes for its closest hypothesis by mean-squared waypoint error $L_{k,m} = \frac{10}{3T}\sum_t \|T^*_{k,t} - \hat{T}_{m,t}\|_2^2$, the winning hypothesis absorbs most of that expert's loss, and unvoted hypotheses take a small pull (ε=0.05) toward the best expert so unused modes do not collapse. Because inference selects $\arg\min_m \hat{c}_m$, the cost head must rank hypotheses correctly. Here we depart from [1]: rather than regressing cost onto the expert planner's cost of the matched expert trajectory, the target folds each hypothesis's best-expert error together with a geometric clearance penalty g_m, the soft barrier of Sec. III-A accumulated along the *predicted* trajectory, normalized per sample,

$$u_m = \min_k L_{k,m} + \lambda_{\text{geo}}\, g_m, \qquad c^*_m = \frac{u_m}{\text{mean}_{m'}\, u_{m'} + \eta}, \tag{4}$$

and the total loss adds a squared-error regression of $\hat{c}$ onto the detached c^*, so that selecting the lowest-cost hypothesis at deployment favors trajectories that are both accurate and collision-free. We additionally re-render each labeled state from several nearby perturbed poses [17], re-expressing the expert's waypoints in each perturbed body frame so the student learns to recover from small drift; perturbation magnitude is capped per scene by its measured reconstruction fidelity (Sec. IV-B). At deployment, a lightweight differential-flatness controller converts the selected hypothesis's lookahead waypoint into the thrust and body-rate command our dynamics model expects.

IV. Experiments

A. Navigation Performance

We evaluate navigation policies in Isaac Sim, whose GPU-parallel physics follows the approach introduced by Isaac Gym [18], on a 17-scenario benchmark spanning photorealistic and procedurally generated obstacle fields. We compare our recovery-augmented policy (Sec. III-B), trained with either an RGB or a depth observation, against three depth-based baselines from the literature. For each method we report a *success rate*, meaning the drone reached the goal without dropping below zero clearance at the platform's physical

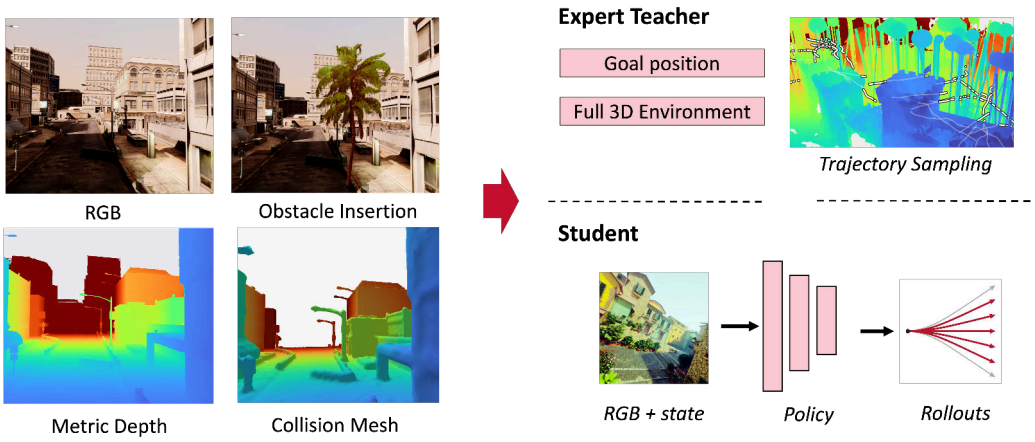


Fig. 4: Imitation learning inside the reconstructed environment. Given an RGB or depth observation (with inserted obstacles) and a goal position, a student policy is supervised by an expert teacher's sampled trajectories; the student predicts a fan of candidate control-command rollouts from observation and state alone.

radius, alongside mean per-scenario minimum clearance and mean flight velocity (Table I).

TABLE I: Navigation benchmark (17 scenarios)

Method	Success Rate ↑	Min. Clearance ↑	Avg. Vel.
Baselines (depth)			
DiffAero [2]	**16/17 (94%)**	**+0.37 m**	2.12 m/s
DepthNav [3]	14/17 (82%)	+0.30 m	2.31 m/s
Agile Autonomy [1]	5/17 (29%)	−0.01 m	2.50 m/s
Ours			
Depth	10/17 (59%)	+0.33 m	2.28 m/s
RGB	5/17 (29%)	−0.09 m	2.51 m/s

Depth's advantage holds even before RGB enters the comparison: trained on an identical corpus and recipe, our depth policy clears the benchmark at twice the rate of our RGB policy, with positive rather than negative mean minimum clearance, consistent with the appearance gap discussed in Sec. II. Our depth policy also lands between the strongest and weakest baselines, trailing DiffAero and DepthNav but well ahead of Agile Autonomy, which suggests the gap our RGB policy still faces owes more to observation modality than to our training recipe. Fig. 5 and Fig. 6 show the broader comparison: our policies fly at mean velocities comparable to the depth baselines, but our RGB policy's flown trajectories cut closer to obstacles than any depth-based method's.

B. Reconstruction Fidelity Away from the Capture Trajectory

A navigation policy flies through the simulator, not just along its capture path, so we test whether reconstruction quality holds away from that path. Fig. 7 samples novel views at increasing lateral offset and tracks rendering and depth accuracy as offset grows; both degrade gradually rather than collapsing, so the reconstruction supports flight through the volume, not just along the capture path.

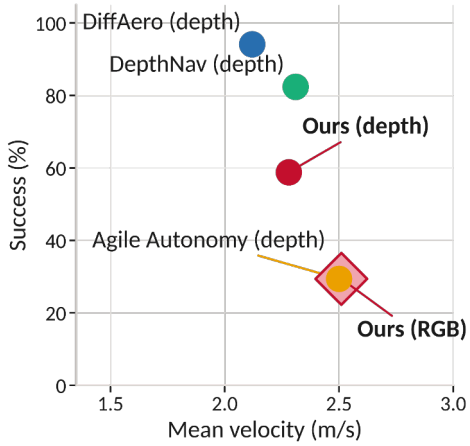


Fig. 5: Success rate vs. mean flight velocity across methods.

C. Qualitative Results

Fig. 8 shows what the aggregate numbers of Sec. IV-D look like on individual frames. Plain 3DGS reaches the sharpest

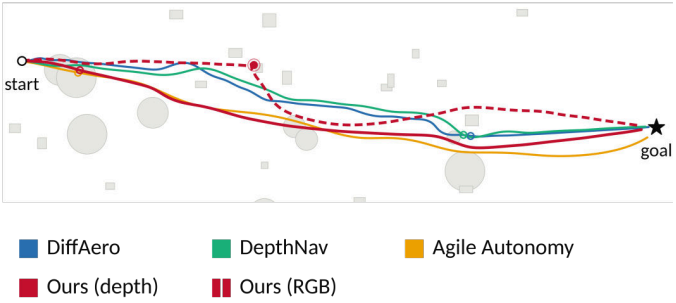
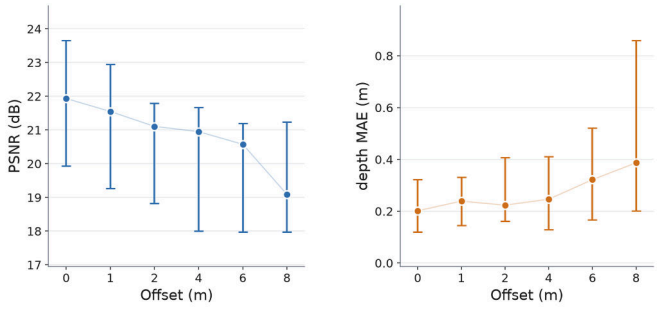


Fig. 6: Representative flown trajectories through the same obstacle field.



(a) PSNR vs. offset (b) Depth MAE vs. offset

Fig. 7: Rendering and geometric fidelity as a function of lateral offset from the capture trajectory.

storefront lettering of the three, 27.4 dB against our 27.1 dB, mirroring the sub-half-decibel trade the aggregate ablation reports, but the reverse trade shows in geometry: red floating-artifact overlays cover 13.5% of 2DGS's pixels and 2.5% of plain 3DGS's, against 0.7% for our recipe.

D. Reconstruction Recipe Ablation

We next ask whether our recipe (Sec. III-A) is worth its extra supervision, and whether the gain holds across representations. On a held-out split of one scene (1,569 training / 225 held-out views), we cross two representations, 3D and 2D Gaussian splatting [16], with two recipes, a photometric-only baseline and our recipe with metric-depth supervision (Table II).

Both photometric-only baselines win on PSNR, but each pays for it in geometry: plain 3DGS's depth error is more than eleven times larger than our recipe's and over a tenth of its pixels are floaters, and plain 2DGS is worse still on both counts. Adding the metric-depth term costs under half a decibel of PSNR relative to plain 3DGS in exchange for an order-of-magnitude improvement in geometric accuracy, which is the property a training environment for a collision-

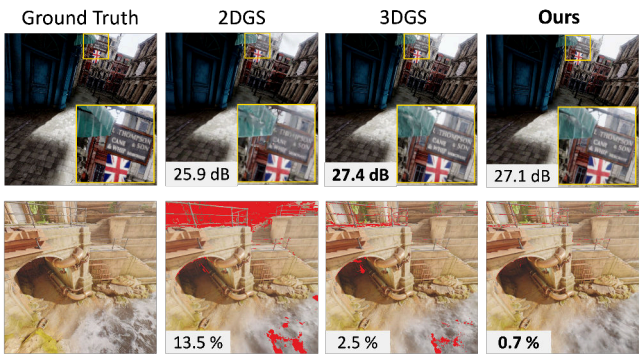


Fig. 8: Ground truth, 2DGS, plain 3DGS, and our recipe rendered from the same viewpoint. Top: RGB detail with PSNR. Bottom: floating-artifact overlay (red) with floater fraction.

TABLE II: Reconstruction recipe ablation (single scene, held-out views)

	Rendering Quality			Geometric Fidelity	
Arm	**PSNR ↑**	**SSIM ↑**	**LPIPS ↓**	**Depth MAE (m) ↓**	**Floaters ↓**
Vanilla 3DGS	**29.35**	**0.8706**	0.215	0.974	11.5%
Vanilla 2DGS	29.07	0.8656	0.228	1.466	13.0%
2DGS + depth	28.26	0.8518	0.213	0.212	1.5%
3DGS + depth (ours)	28.89	0.8676	**0.184**	**0.085**	**0.4%**

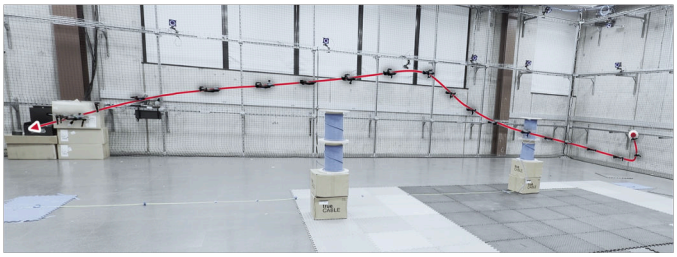


Fig. 9: Real-world flight of the depth-based DiffAero policy, using motion-capture state estimation and a time-of-flight depth sensor. The recovered flight path (red) clears both obstacles inside the motion-capture cage.

avoidance policy actually needs; it also costs a little SSIM, though LPIPS, the perceptual metric our recipe's own loss targets, favors our recipe over every baseline. That trade holds across representations, but not identically: adding depth supervision to 2DGS still leaves its floater rate at 1.5%, nearly four times our recipe's 0.4%, so depth supervision helps every representation but 3DGS realizes it best.

E. Real-World Deployment

As a preliminary hardware check, we deployed the depth-based DiffAero policy (Sec. IV-A) on a Starling 2 Max drone, using motion-capture for state estimation and a time-of-flight (ToF) sensor in place of the simulated depth image. The policy outputs were converted into target velocity commands and tracked by a PX4 controller; an OptiTrack system provided the quadrotor's state, and the ToF image was downsampled to 16×9 before being fed to the policy as its depth observation. Fig. 9 shows the resulting flight path clearing two obstacles. Because this test used motion-capture pose rather than onboard estimation, a PX4 velocity controller rather than our flatness controller, and the depth-based baseline rather than our own student policies, it validates only the hardware integration of a depth-based navigation policy under external state estimation; a fully onboard, monocular flight test remains the deployment this pipeline is ultimately built toward (Sec. V).

V. Discussion and Future Work

Our central result is a gap, not a wall: on the same corpus and recipe, the RGB policy clears 5/17 scenarios against the depth policy's 10/17, and every scenario the RGB policy clears is one its depth counterpart also clears. A gap of this shape is what we would expect from a training corpus that has not yet

seen enough visual variation, not from a modality that cannot support navigation at all.

That reading rests on a 17-scenario benchmark, a reconstruction ablation run on a single held-out scene (Sec. IV-D), and reconstruction-fidelity and navigation results measured entirely in simulation, aside from the preliminary hardware check of Sec. IV-E. A wider, more visually diverse scene library, an ablation repeated across scenes, and a flight test on a physical monocular platform would each test whether these results hold at scale; the last of these is the evaluation this pipeline was ultimately built toward, since a depth-free navigation stack only pays off once it flies on hardware that has no depth sensor to fall back on. The reconstruction and policy-training pipeline (Sec. III-A, Sec. III-B) places no requirement on TartanAir specifically, so both directions, a larger scene library and real-world flight, are natural next steps.

VI. Conclusion

We build a 3D Gaussian splatting pipeline that reconstructs TartanAir captures into flyable, photorealistic simulation environments, jointly supervised with RGB and metric depth, and validate that reconstruction fidelity holds at novel viewpoints away from the capture trajectory. Inside this simulator we train goal-conditioned navigation policies via privileged imitation learning from RGB or depth observations and benchmark both against three depth-based baselines from the literature. The depth policy lands between the strongest and weakest of those baselines; the RGB policy trails, a gap we attribute to the limited visual diversity of the current training library (Sec. V), which argues for a larger reconstructed scene library and a flight test on a physical monocular platform.

Acknowledgment

This work was made possible by the Robotics Institute Summer Scholars (RISS) program. The authors thank Rachel Burcin, John Dolan, and the RISS sponsors for organizing the program.

References

[1] 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, eabg5810, Oct. 13, 2021.

[2] X. Zhang, R. Wang, Y. Ren, *et al.* "DiffAero: A GPU-Accelerated Differentiable Simulation Framework for Efficient Quadrotor Policy Learning." (Sep. 12, 2025).

[3] J. Lee, A. Rathod, K. Goel, J. Stecklein, and W. Tabib. "Quadrotor Navigation using Reinforcement Learning with Privileged Information." (Mar. 5, 2026).

[4] A. Bhattacharya, N. Rao, D. Parikh, *et al.* "Vision Transformers for End-to-End Vision-Based Quadrotor Obstacle Avoidance." (Apr. 1, 2025).

[5] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, *et al.* "Learning Monocular Reactive UAV Control in Cluttered Natural Environments." (Nov. 7, 2012).

[6] F. Sadeghi and S. Levine. "CAD2RL: Real Single-Image Flight without a Single Real Image." (Jun. 8, 2017).

[7] A. Loquercio, A. I. Maqueda, C. R. del-Blanco, and D. Scaramuzza, "DroNet: Learning to Fly by Driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[8] Q. Chen, J. Sun, N. Gao, J. Low, T. Chen, and M. Schwager, *GRaD-Nav: Efficiently learning visual drone navigation with gaussian radiance fields and differentiable dynamics*, 2025.

[9] J. Low, M. Adang, J. Yu, K. Nagami, and M. Schwager, *SOUS VIDE: Cooking visual drone navigation policies in a Gaussian Splatting vacuum*, 2025.

[10] A. Quach, M. Chahine, A. Amini, R. Hasani, and D. Rus. "Gaussian Splatting to Real World Flight Navigation Transfer with Liquid Networks." (Oct. 16, 2024).

[11] W. Wang *et al.*, "TartanAir: A dataset to push the limits of visual SLAM," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[12] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d Gaussian splatting for real-time radiance field rendering," in *ACM Transactions on Graphics (SIGGRAPH)*, 2023.

[13] Z. Xie, Z. Liu, Z. Peng, W. Wu, and B. Zhou, "Vid2Sim: Realistic and interactive simulation from video for urban navigation," in *CVPR*, 2025, pp. 1581–1591.

[14] A. Escontrela, J. Kerr, A. Allshire, *et al.*, *GaussGym: An open-source real-to-sim framework for learning locomotion from pixels*, 2025.

[15] T. Chen, O. Shorinwa, J. Bruno, *et al.* "Splat-Nav: Safe Real-Time Robot Navigation in Gaussian Splatting Maps." (Jan. 11, 2025).

[16] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d Gaussian splatting for geometrically accurate radiance fields," in *ACM Transactions on Graphics (SIGGRAPH)*, 2024.

[17] A. Tagliabue and J. P. How. "Tube-NeRF: Efficient Imitation Learning of Visuomotor Policies from MPC using Tube-Guided Data Augmentation and NeRFs." (Feb. 26, 2024).

[18] V. Makoviychuk, L. Wawrzyniak, Y. Guo, *et al.*, *Isaac Gym: High performance GPU-based physics simulation for robot learning*, 2021.