

Flying in Clutter on Monocular RGB by Learning in 3D Radiance Fields with Domain Adaptation

Xijie Huang^{1,2}, Jinhan Li^{1,2}, Tianyue Wu^{1,2}, Xin Zhou²
Zhichao Han^{1,2,†}, and Fei Gao^{1,2,†}

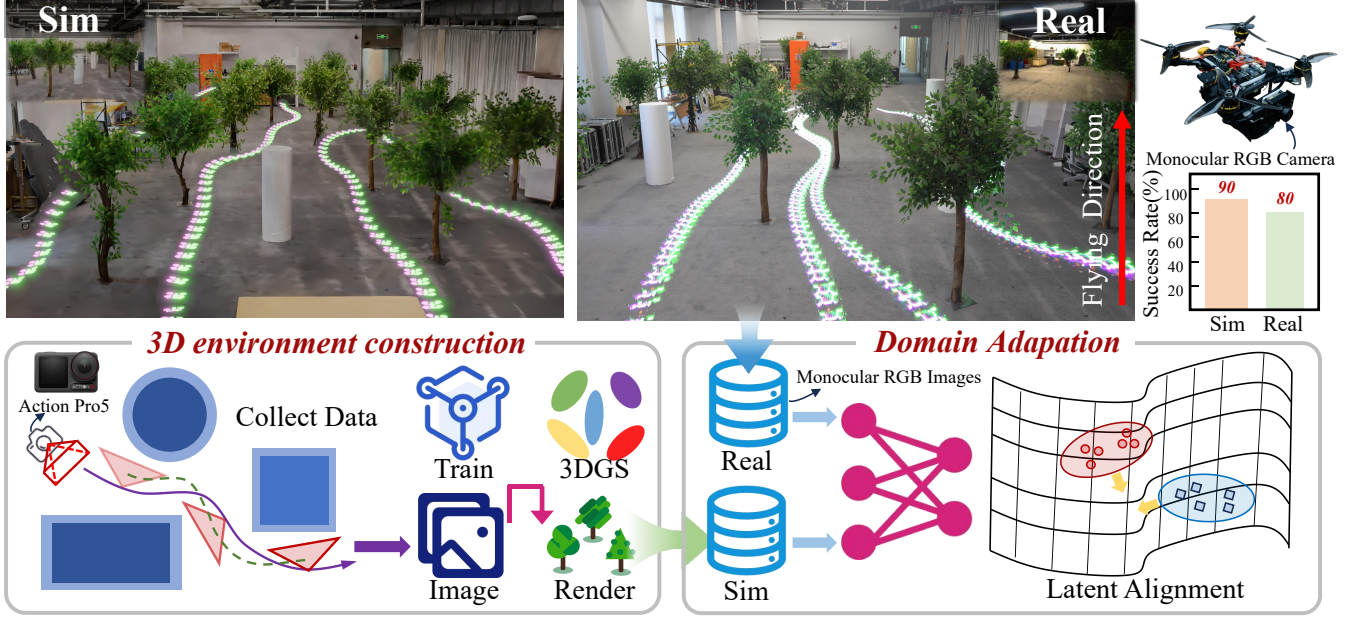


Fig. 1: **Overview of our RGB-based navigation framework.** The top panels demonstrate that our method effectively mitigates the sim-to-real gap. The bottom panel illustrates the pipeline of 3D environment construction and domain adaptation.

Abstract—Modern autonomous navigation systems predominantly rely on lidar and depth cameras. However, a fundamental question remains: Can flying robots navigate in clutter using solely monocular RGB images? Given the prohibitive costs of real-world data collection, learning policies in simulation offers a promising path. Yet, deploying such policies directly in the physical world is hindered by the significant sim-to-real perception gap. Thus, we propose a framework that couples the photorealism of **3D Gaussian Splatting (3DGS)** environments with Adversarial Domain Adaptation. By training in high-fidelity simulation while explicitly minimizing feature discrepancy, our method ensures the policy relies on **domain-invariant cues**. Experimental results demonstrate that our policy achieves robust zero-shot transfer to the physical world, enabling safe and agile flight in unstructured environments with varying illumination.

I. INTRODUCTION

For Unmanned Aerial Vehicles (UAVs), effective environmental understanding is the core of safe and robust navigation. Within the realm of perception systems, unlike depth sensors or lidar, which necessitate precise extrinsic calibration across multiple components [1], monocular RGB

cameras represent the most compact and natural perception modality. Offering distinct advantages in size, weight, and power, they provide a promising sensing solution for miniature platforms and non-rigid UAV structures where hardware constraints are strict. However, existing approaches have yet to fully exploit this potential. **Traditional methods, which decouple navigation into localization, mapping, and trajectory optimization [2],** struggle to extract reliable geometric structures from monocular imagery. While data-driven approaches offer a compelling end-to-end alternative [3, 4], the majority still depend on explicit geometric priors provided by lidar [5] or depth sensors [6]. Thus, a question arises: Is it feasible for flying robots to achieve robust autonomous navigation relying on monocular RGB information?

While collecting real-world data is costly, training in simulation serves as a more efficient alternative paradigm. In practice, learning navigation policies directly from raw monocular images in simulation presents two major challenges. First, unlike depth maps or point clouds, monocular RGB data is **inherently implicit and scale-ambiguous [7], making the extraction of implicit geometric structures and navigation information difficult to model via traditional rule-based methods.** Second, the **sim-to-real gap is particularly pronounced in the visual modality,** which severely hinders the transfer of control policies learned entirely from simu-

[†]Corresponding authors: Zhichao Han and Fei Gao

¹State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China.

²Differential Robotics, Hangzhou 311121, China.

E-mail:{xijiehuang, fgaoa}@zju.edu.cn

lation rollouts. The complex illumination variations, diverse textures, and sensor noise ubiquitous in the physical world often induce a significant visual distribution shift, causing policies trained in simulation to degrade or fail upon real-world deployment. Existing solutions typically resort to explicit intermediate representations, such as optical flow [8], to bypass the geometric ambiguity, or employ visual domain randomization [7] to mitigate the distribution shift. Unfortunately, the former inevitably introduces modular latency and loses rich semantic information, while the latter relies on unrealistic augmentations that compromise performance.

In this paper, we propose a navigation framework designed to tackle these specific challenges. First, to address the difficulty of modeling implicit geometric structures, we leverage end-to-end reinforcement learning (RL) to learn policies directly from raw RGB images. This data-driven paradigm empowers the agent to implicitly extract reliable navigation cues from photometric inputs. Second, to overcome the visual sim-to-real gap, we introduce a high-fidelity simulation pipeline combined with domain adaptation (DA). We employ 3D Gaussian Splatting (3DGS) [9] to reconstruct real-world scenes, establishing a highly photorealistic training environment. To mitigate the prohibitive computational cost of standard 3DGS during RL training, we accelerate the pipeline via model pruning and parallelized backends [10]. These optimizations enable real-time rendering with a peak throughput of approximately 30,000 frames per second. Furthermore, inspired by [11], we explicitly integrate an adversarial domain adaptation mechanism [12] to align feature distributions between simulation and reality. Ultimately, our method achieves successful zero-shot transfer, demonstrating robust navigation capabilities in real-world scenarios with randomly distributed obstacles and varying illumination.

Overall, the main contributions of this paper can be summarized as follows:

- 1) We propose an end-to-end training paradigm for navigation using monocular RGB inputs.
- 2) We leverage model pruning and parallelized backends to accelerate 3DGS rendering.
- 3) We utilize domain adaptation to bridge the gap between the 3DGS simulation domain and the UAV camera domain, facilitating robust sim-to-real transfer.
- 4) We validate through simulation and real-world experiments that our pipeline enables safe and agile autonomous flight.

II. RELATED WORK

A. Autonomous Navigation with Geometric Sensors

Significant progress has been made in autonomous UAV navigation by leveraging sensors that provide explicit geometric information. Loquercio et al. [13] utilized imitation learning to map depth images directly to control actions, successfully demonstrating high-speed flight in wild environments. Subsequently, Zhang et al. [6] exploited a differentiable simulator to provide first-order gradient information for efficient policy updates, achieving robust obstacle

avoidance via reinforcement learning based on depth inputs. Parallel to depth-based methods, other works have focused on processing high-dimensional point clouds. For instance, a recent work [5] proposed a method to handle raw lidar data, navigating safely around thin obstacles.

However, these approaches rely on lidar and depth cameras, which are generally more expensive, heavier, and more energy-consuming. Furthermore, in scenarios demanding semantic understanding of the environment, such purely geometric methods may prove suboptimal.

B. Visual Navigation from Monocular RGB

To address the limitations of geometric sensors, research has shifted towards navigation based on monocular RGB inputs. Early works [14, 15] utilized reinforcement learning to train collision avoidance policies within simulation. Notably, CAD²RL [7] introduced a sim-to-real paradigm, leveraging extensive visual domain randomization to achieve successful transfer to the physical world. However, the limited visual fidelity of simulators failed to provide sufficiently realistic training data. This discrepancy widened the sim-to-real gap and degraded policy performance in real-world scenarios.

Another prevailing paradigm involves extracting explicit geometric features from RGB images before feeding them into the control policy. Methods utilizing depth estimation [16] or optical flow estimation [17] employ these features as intermediate representations to simplify the learning task. While effective, these additional processing modules introduce computational cost and inference latency [8]. Moreover, relying on such intermediate representations inevitably leads to information loss, thus limiting the performance of high-level tasks.

With the recent advent of 3D Gaussian Splatting (3DGS) for high-fidelity scene reconstruction, new possibilities for realistic training have emerged. Grad-Nav [18] proposed training navigation policies within photorealistic 3DGS environments. Despite its success in simulation, the method is primarily designed for simple gate-traversal tasks and relies on hand-crafted reward shaping (e.g., trajectory waypoints). This dependence constrains the agent’s exploration capabilities, often leading to sub-optimal policies.

III. TRAINING PIPELINE FOR RGB NAVIGATION TASK

A. 3DGS Formulation and Environment Integration

The original 3D Gaussian Splatting method represents the scene as a set of 3D Gaussian primitives, each parameterized by a spatial mean, a full covariance matrix, view-dependent radiance, and an opacity value. During rendering, the color of each pixel is obtained by front-to-back volumetric compositing of all Gaussians whose projected footprints cover that pixel:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \quad (1)$$

where $\mathcal{N}$ denotes the set of Gaussians intersecting the pixel, c_i is the view-dependent color (radiance) of Gaussian i , and $\alpha_i \in [0, 1]$ is its effective opacity at that pixel.

However, reinforcement learning typically requires large batch sizes during training. To satisfy this requirement, we adapt 3DGS pruning introduced in Speedy Splat [19]. For each rendered Gaussian, its projection onto the 2D image plane is an ellipse. The Gaussian-to-tile mappings are constrained strictly to the tiles that overlap with the projected ellipse, rather than the square bounding region employed in the original 3DGS framework (derived from a radius $r = \lceil 3\sqrt{\lambda_{max}} \rceil$, where λ_{max} represents the maximum eigenvalue of the projected 2D covariance matrix).

To sparsify primitives, each Gaussian (G_i) is assigned a lightweight importance score

$$\tilde{U}_i = (\nabla_{g_i} I_g)^2 \quad (2)$$

where I_g is the final rendered image and g_i is the 2D projected scalar value of the i -th Gaussian (G_i). During training, Gaussians are pruned according to this score.

We constructed 10 distinct scenes using video scanning, featuring the rearrangement of obstacles (e.g., adding, removing, or repositioning) for each scene. As shown in Fig. 2, for each video, we performed a COLMAP Structure from Motion (SfM) [20] reconstruction, which was subsequently processed by Speedy Splat to generate a corresponding 3DGS model. The dataset was partitioned into 9 scenes for training and 1 for evaluation.

Each scene’s scale and coordinate system were then transformed to align with the simulation world, and the point cloud geometry corresponding to each scene was extracted into Isaac Sim to render depth and detect collisions. In contrast, the RGB observation is rendered in real-time from the drone’s current position and orientation using the trained 3DGS model. This rendering is accelerated by the gsplat library [10], which utilizes vectorized, GPU-batched operations for all Gaussian projection, sorting, and compositing steps. Our pipeline achieves a peak rendering speed of 30,000 frames per second.

B. Learning the policy

1) *Actor Policy*: The network accepts two inputs: RGB images (I_{rgb}) of size 60×80 rendered in real-time by 3DGS, and a 20-dimensional UAV state vector s_t . The state vector is defined as:

$$s_t = [\mathbf{v}_b, \mathbf{R}, \mathbf{d}_g, z_c, z_t, \mathbf{a}_{last}] \quad (3)$$

where $\mathbf{v}_b \in \mathbb{R}^3$ denotes the body-frame velocity, $\mathbf{R} \in \mathbb{R}^9$ is the rotation matrix, $\mathbf{d}_g \in \mathbb{R}^3$ is the normalized goal vector, $z_c, z_t \in \mathbb{R}$ are current and target heights, and $\mathbf{a}_{last}$ corresponds to the previous action. We first employ a normalization module to standardize both input observations (I_{rgb} and s_t). Subsequently, a three-layer CNN extracts features from the normalized images, flattening them into a 4480-dimensional latent vector, $\mathbf{z}_{rgb}$. This visual embedding is then concatenated with the normalized state vector s_t . The combined representation is processed by a GRU with 512 hidden units to capture temporal dynamics. Finally, the output is passed through an MLP and a projection layer to

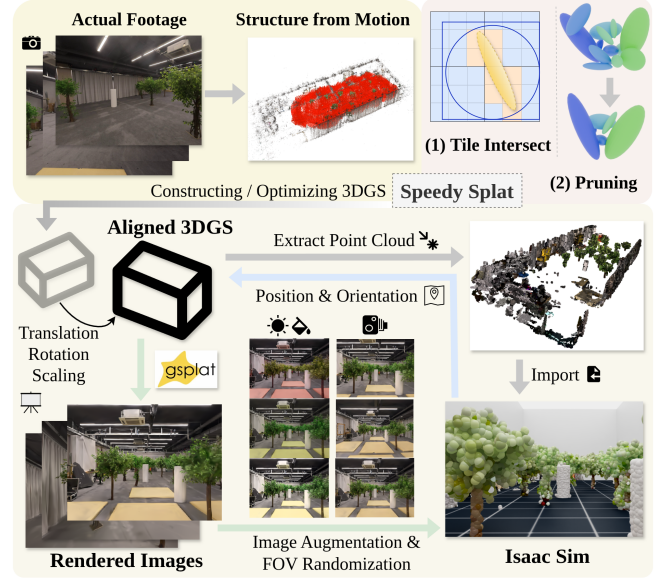


Fig. 2: **Pipeline for constructing the 3DGS-based simulation environment.** To accelerate rendering, we employ Speedy Splat for model pruning and utilize gsplat as a parallelized rasterization backend. The aligned point cloud maps are then imported into Isaac Sim to enable depth rendering and collision detection.

produce the normalized action $\mathbf{a}_t = [T, \phi, \theta, \psi]$, which is converted to torque and thrust commands via a cascaded PID controller.

2) *Critic Policy*: The Critic network shares a similar architecture to the Actor. To stabilize the training of the value function, we leverage the Asymmetric Actor-Critic paradigm, which utilizes privileged information available in simulation [21]. Specifically, in addition to the standard actor observation (I_{rgb}), the Critic’s CNN input is augmented with a 60×80 depth map (I_d) as auxiliary feature information from Isaac Sim. The effectiveness of incorporating this privileged depth information in stabilizing the training will be validated in Section V-A.

3) *Termination*: An episode terminates when (i) the UAV’s altitude (z-axis position) falls outside the predefined safety range of $[0.1\text{m}, 1.7\text{m}]$, (ii) a collision with the environment is detected, which is checked in real-time using a global, inflated point cloud occupancy map, or (iii) the UAV successfully reaches the goal. When the agent meets any termination condition, it is reset to a new starting point randomly sampled from an initial free-space region.

4) *Reward Function*: To achieve fast and stable UAV navigation in cluttered environments, we design a composite reward function. The total reward r_t at each timestep t is a weighted sum of several components:

$$r_t = \sum_i w_i r_i \quad (4)$$

where r_i is an individual reward component and w_i is its corresponding weight.

For the Distance Reward, we use the change in goal distance $r_{\text{distance}} = d_{t-1} - d_t$, where d_t is the Euclidean distance to the goal. This reward is clamped per-step to the

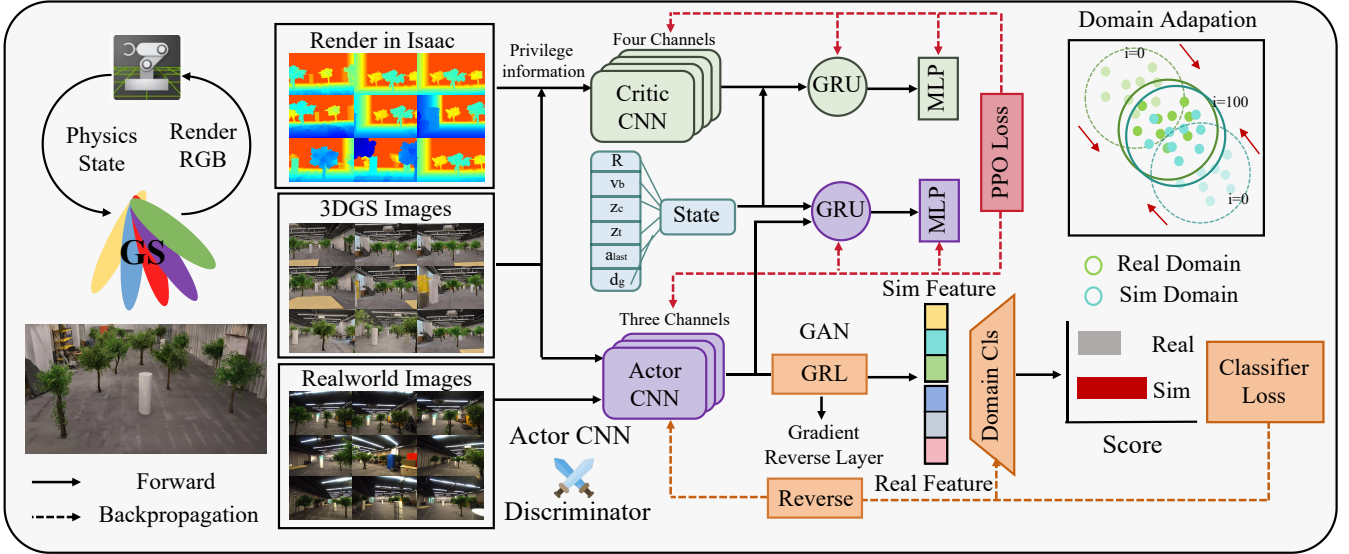


Fig. 3: **Overview of the proposed RL training framework.** The architecture features an asymmetric actor-critic structure and employs an adversarial domain adaptation module to bridge the sim-to-real gap.

maximum possible distance change, $d_{\text{step}} = v_{\text{max}} \cdot \Delta t$. The components are detailed in Table I.

TABLE I: Reward function components and their respective weights. v_{max} is set to 2 m/s in this experiment.

Reward	Equation (r_i)	Weight (w_i)
Obstacle Collision	$r_{\text{collision}}$	-80.0
Z-Velocity Penalty	$-\min(v_{b,z} , 1.0)$	-0.5
Action Magnitude	$\ \mathbf{a}_{t,0:2}\ ^2$	-0.3
Action Change	$\ \mathbf{a}_{t,0:2} - \mathbf{a}_{t-1,0:2}\ ^2$	-0.6
Distance Reward	$\text{clamp}(d_{t-1} - d_t, -d_{\text{step}}, d_{\text{step}})$	30.0
Success Reward	r_{success}	80.0
Velocity Excess	$\max(-(e^{\ \mathbf{v}_t\ - v_{\text{max}}} - 1), -5.0)$ if $\ \mathbf{v}_t\ > v_{\text{max}}$, else 0	-1.0

IV. SIM-TO-REAL TRANSFER

A. Domain Adaptation

To reconstruct high-quality 3DGS environments for training, we utilize the DJI Osmo Action 5 Pro camera¹ for data recording. However, during real-world deployment, cost and payload constraints necessitate the use of affordable and lightweight sensors. In our experiments, we employ a Hikvision industrial camera² for onboard perception. As illustrated in Fig. 3, this sensor mismatch introduces a significant gap which arises from discrepancies in lens distortion, color style, exposure, and intrinsic 3DGS reconstruction artifacts.

To bridge this gap, we employ a DA strategy to align the latent feature spaces of the two domains efficiently. It needs only approximately several minutes of real-world camera video.

As shown in Fig. 3, a discriminator is designed after the Actor CNN Encoder. It aims to distinguish whether a latent

feature vector comes from the 3DGS or the real world. Conversely, the encoder aims to generate domain-invariant features that can mislead the discriminator.

During the forward pass, the Gradient Reversal Layer acts as an identity function. In backpropagation, it reverses the gradient flowing from the discriminator to the encoder, scaled by a factor λ :

$$\frac{\partial \mathcal{L}_{\text{DA}}}{\partial \theta_E} = -\lambda \frac{\partial \mathcal{L}_{\text{DA}}}{\partial \theta_D} \quad (5)$$

where θ_E and θ_D are the parameters of the encoder and discriminator, respectively.

The discriminator is trained using a binary cross-entropy loss to maximize classification accuracy:

$$\mathcal{L}_{\text{DA}} = -\mathbb{E}_{z_s \sim \mathcal{D}_s} [\log D(z_s)] - \mathbb{E}_{z_t \sim \mathcal{D}_t} [\log(1 - D(z_t))] \quad (6)$$

where D is discriminator, $\mathcal{D}_s$ and $\mathcal{D}_t$ are the latent feature from 3DGS (source) and real-world (target).

In contrast to [11], which decouples the training of the discriminator and the RL agent, we integrate the domain loss directly into the PPO training objective. This approach reduces the loss of optimality caused by fragmented training, ensuring the policy retains its navigation capabilities while effectively aligning features across domains. The total loss $\mathcal{L}_{\text{total}}$ is a weighted sum of the PPO loss and the DA loss:

$$\mathcal{L}_{\text{total}} = \lambda_1 \cdot \mathcal{L}_{\text{PPO}} + \lambda_2 \cdot \mathcal{L}_{\text{DA}} \quad (7)$$

where $\mathcal{L}_{\text{PPO}}$ includes the policy surrogate loss, value function loss, and entropy term. The coefficients λ_1 and λ_2 are carefully tuned to balance task performance with feature alignment, ensuring the policy learns robust obstacle avoidance behaviors and generalizes to the real-world latent space.

B. Domain Randomization

To bridge the sim-to-real gap, we implement comprehensive randomization on dynamics and perception. Detailed parameters and distributions are provided in Table II.

¹DJI Osmo Action 5 Pro: <https://www.dji.com/osmo-action-5-pro>

²Hikvision industrial camera: <https://www.hikrobotics.com/cn/machinevision/productdetail/?id=9706>

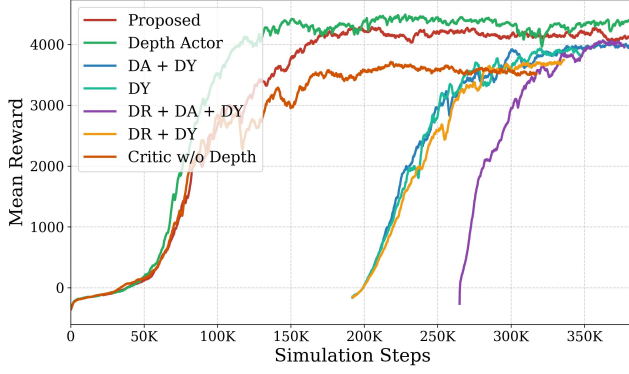


Fig. 4: **Training curves for ablation studies.** We compare the mean reward convergence across different policy inputs and sim-to-real strategies. Abbreviations: DA (Domain Adaptation), DR (Visual Domain Randomization), and DY (Dynamic Domain Randomization). Note: DY is applied as a fundamental component to mitigate the dynamics gap between simulation and reality. The DA+DY, DR+DY and DY are trained based on the Proposed, while DR+DA+DY is trained based on DR+DY.

1) *Dynamics noise randomization:* We simulate actuator imperfections by injecting Gaussian noise and introducing variable latency into the control command, similar to [22]. Latency is modeled as a moving average over a history window. Rather than using fixed parameters, both the delay magnitude (D) and noise vectors are dynamically re-sampled at randomized intervals (T) within an episode. This stochasticity forces the policy to adapt to changing disturbances rather than overfitting to static delay patterns.

2) *Perception noise randomization:* To emulate the imperfections of real-world state estimation, we introduce noise perturbations to the UAV’s linear velocity, heading vector, and altitude. Crucially, estimation errors in physical systems are rarely temporally independent. Thus, the noise dynamics are governed by Eq. 8, with a mean-reversion rate of $\theta = 0.1$:

$$\begin{aligned} n_{t+1} &= (1 - \theta)n_t + \mathcal{N}(0, \sigma^2) \\ \hat{s}_{t+1} &= s_{t+1} + n_{t+1} \end{aligned} \quad (8)$$

where s_{t+1} denotes the ground-truth state and $\hat{s}_{t+1}$ represents the policy observation.

Furthermore, we adapt the visual augmentation method from RoboSplat [23] by directly transforming the 3DGS parameters as follows:

$$C'_{dc} = \alpha \cdot C_{dc} + \beta + \epsilon \quad (9)$$

where α and β represent scale and offset factors, respectively. Additionally, we randomize the FOV to improve generalization across obstacle scales.

V. EVALUATIONS

The policy is trained using 1,024 parallel environments with a rollout length of 128 time steps per update. As illustrated in Fig. 4, our proposed method follows a three-stage training. It is important to note that DR refers specifically to visual DR. Since Dynamic Domain Randomization (DY) is a well-established standard for bridging the dynamics gap in sim-to-real transfer, it is applied as a fundamental setting

TABLE II: Domain randomization parameters and implementation details.

Parameter	Probability	Distribution / Value
Dynamics noise randomization		
Action Noise (ϵ)	1.0	$\sim \mathcal{N}(0, \sigma^2)$
Latency Delay (D)	1.0	$\sim \mathcal{U}(0\text{ms}, 80\text{ms})$
Resample Interval (T)	1.0	$\sim \mathcal{U}(10\text{ms}, 100\text{ms})$
Perception noise randomization		
Velocity State Noise (σ_{vel})	1.0	$\sim \mathcal{N}(0, 0.08)$
Direction State Noise (σ_{dir})	1.0	$\sim \mathcal{N}(0, 0.05)$
Z-axis State Noise (σ_z)	1.0	$\sim \mathcal{N}(0, 0.03)$
Color Scale (α)	1.0	$\sim \mathcal{U}(0.8, 1.3)$
Color Offset (β)	1.0	$\sim \mathcal{U}(-0.05, 0.05)$
SH Additive Noise (σ_{rgb})	1.0	$\sim \mathcal{N}(0, 0.025^2)$
Field of View (FOV)	1.0	$\sim \mathcal{U}(67^\circ, 106^\circ)$

in our real-world experiments and is not separately ablated. In the first stage, we train a baseline policy. Subsequently, we introduce DR+DY to fine-tune the policy in the second stage. Finally, we incorporate both DR+DY+DA to align latent features in the final stage. The entire training process is completed in approximately two days on a single NVIDIA L40 GPU. For real-world deployment, the final policy is executed on an NVIDIA Jetson Orin NX, achieving an inference latency of approximately 2 ms. Real-time state estimation, such as the normalized direction vector, altitude, and velocity, is derived from an Extended Kalman Filter (EKF) that fuses optical flow and IMU measurements.

A. Ablation Study

In this section, we conduct a series of ablation studies to validate the effectiveness of using privileged information (i.e., Depth) in the Critic’s training and the efficacy of the rich semantic information from the RGB input.

We compare the learning curves and final success rates of four experimental setups, as shown in Fig. 4: 1) **Critic without depth:** Removing privileged information from the Critic; 2) **Depth Actor:** A baseline using ground-truth depth for the Actor; 3) **Proposed:** Our full method (RGB-Actor, Privileged-Critic);

First, our method achieves reward levels competitive with the Depth-based baseline. It suggests that our policy successfully extracts implicit geometric features and traversability cues directly from high-dimensional photometric data. This result validates the feasibility of achieving robust navigation relying solely on monocular RGB information. Consequently, it serves as a lightweight yet viable alternative, eliminating the need for expensive sensors (e.g., LiDAR or depth cameras) in defined environments.

Next, we validate the effectiveness of privileged information by comparing our proposed method against the No-Privilege setup. As shown in Figure 4, without privileged depth, the policy fails to converge effectively, achieving a reward 20% less than Proposed. This strongly indicates that privileged information provides the Critic with an accurate spatial understanding. It allows the Critic to form a more stable and accurate value function, which in turn provides a

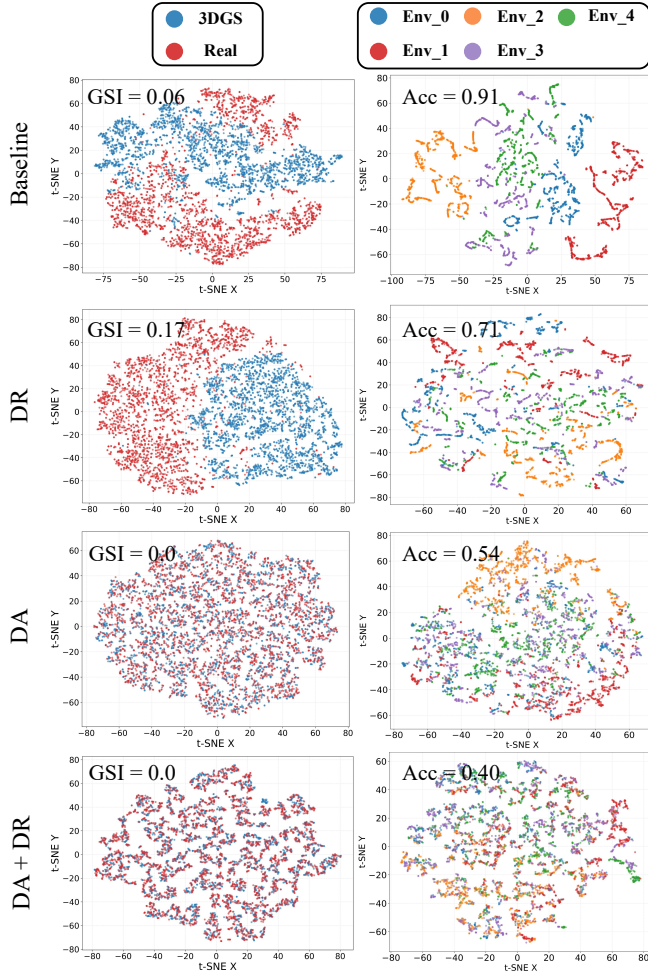


Fig. 5: **t-SNE visualization of the latent feature space.** The left column evaluates domain alignment between 3DGS and Real-world inputs (measured by GSI), while the right column illustrates feature discriminability across different data-augmented environments (measured by Classification Accuracy).

more reliable learning signal for the Actor.

B. Visualizing the Sim-to-Real Gap

To intuitively evaluate the efficacy of our method in bridging the sim-to-real gap and understanding the internal representations, we employ t-SNE to project the high-dimensional latent features extracted by the encoder of the Actor network into a two-dimensional manifold for visualization.

1) *Alignment between Simulation and Reality:* We first evaluate the sim-to-real alignment by visualizing the feature embeddings of the original 3DGS rendered images against real-world captured images. We employ the Geometric Separability Index (GSI) [24] to quantify the degree of region overlap, where a lower GSI indicates better mixing.

As illustrated in the left column of Fig. 5, the Baseline model exhibits a distribution with a GSI of 0.06, indicating a lack of shared feature representation. Interestingly, applying DR leads to a clearer separation between the simulation and real-world clusters, which increases the GSI to 0.17. This phenomenon suggests that the real-world observations

remain out-of-distribution to the encoder. In contrast, our DA method successfully aligns the two distributions, making them virtually indistinguishable. This confirms that the encoder has learned to extract domain-invariant features from diverse visual representations.

As demonstrated in the bottom-left panel of Fig. 5, the latent features, trained through a combination of DA and DR, exhibit a highly categorized distribution. Clusters representing similar underlying visual properties become tightly concentrated, while those corresponding to distinct feature types are more spatially segregated. This dual approach not only achieves effective domain alignment between simulation and reality but also enhances the discriminability of different features within the same domain.

2) *Difference between Generalization and Alignment:* To further investigate the underlying mechanisms of different training strategies, we visualize the latent vectors of images from five distinct data-augmented environments (Env 0-4). Additionally, we use a simple classifier (Logistic Regression) on these features to quantify their distinguishability via classification accuracy (Acc), where a lower Acc means better alignment.

As shown in the right column of Fig. 5, the Baseline model produces clearly isolated clusters for each environment, resulting in a high classification accuracy of 91%. This indicates that the model heavily relies on environment-specific styles (e.g., texture and lighting) rather than task-relevant semantic information. Introducing DR reduces the distances between clusters and lowers the accuracy to 71%. This suggests that DR achieves robustness primarily by forcing the model to adapt to a wider range of visual variations.

Most notably, although our DA method was explicitly trained to align only raw simulation data with real-world data, it achieves tight clustering across all five unseen augmented environments. The classification accuracy drops sharply to 54%, confirming that the adversarial training effectively removes style-related information. When combining DA with DR, the accuracy is further reduced to 40%. This comparison highlights a key difference in their mechanisms: unlike DR, which relies on expanding the data coverage to include more styles, DA forces the network to learn universal geometric and semantic features that are invariant to domain shifts.

C. Real-world Flight

1) *Ablation Study of Sim-to-Real Methods in Real-World Scenarios:* To further evaluate the transfer performance of various sim-to-real approaches, we conducted validation experiments in real-world scenarios. We established five comparative settings: Baseline, DY, DY+DR, DY+DA, and our proposed Method (combining DR+DA+DY).

As illustrated in Table III, the Proposed Method demonstrates superior robustness against environmental noise and achieves smooth navigation during sim-to-real transfer. In comparison, the DY+DR method fails to generalize effectively, suggesting that standard visual DR alone is insufficient

to cover the complex visual distribution of real-world scenes.

Furthermore, although the DY+DA method demonstrates the feasibility of navigation, it exhibits a suboptimal success rate. The primary reason is its lack of robustness to environmental interference, such as sensor noise or light condition differences, resulting in frequent collisions with complex obstacles such as foliage. Finally, real-world state estimation, computed via EKF-based fusion of downward optical flow and IMU data, is inherently noisy and uncertain. Thus, without DY, the Baseline fails to bridge the dynamics gap, exhibiting significant instability in the physical environment.

TABLE III: Ablation study on real-world capabilities. We evaluate each variant based on its ability to mitigate the dynamic gap, resist visual interference, and perform visual sim-to-real transfer.

Method	Mitigate Dynamic Gap	Visual Robustness	Visual Sim-to-Real
Baseline	×	×	×
DY	✓	×	×
DY + DR	✓	✓	×
DY + DA	✓	×	✓
DR + DA + DY (Proposed)	✓	✓	✓

2) *Efficient sim-to-real transfer*: To validate the effectiveness of our sim-to-real approach, as illustrated in Fig. 1, we conduct comparative experiments in both real-world and simulated environments. Each experimental set consisted of 10 test trajectories. We achieve a 90% success rate in simulation and an 80% success rate in the real-world environment. The high success rate demonstrates the successful sim-to-real transfer with our pipeline. The primary cause of failure in the real world was collisions resulting from foliage swaying induced by propeller downwash—a dynamic factor currently unmodeled in the simulation.

Furthermore, we evaluate the robustness and zero-shot generalization of our policy through a continuous round-trip navigation task. Crucially, this experiment is conducted as a single, uninterrupted flight session without landing the UAV or resetting the control algorithm. To prevent map memorization, we physically reshuffled the positions of obstacles (including trees and pillars) during the transition between flight legs. As illustrated in Fig. 6, the left panels depict the forward trajectories, while the right panels show the subsequent return flights, with each flight conducted in a randomly reconfigured environment. Despite the random changes in obstacle distribution, our policy consistently generates smooth, collision-free trajectories. This further validates that our policy successfully extracts generalized visual features, ensuring robust performance across diverse and unseen obstacle configurations.

3) *Robustness under visual change*: To further evaluate the visual robustness of our policy, we design a flight task characterized by rapidly changing illumination, as shown in Fig. 7. Each frame corresponds to an abrupt shift in lighting conditions. This experiment serves as a stress test for the feature extractor. The results suggest that our framework, driven by DA and DR, effectively extracts semantic geometry

despite varying illumination. Instead of relying on low-level pixel intensities that are susceptible to illumination changes, the encoder extracts consistent representations. As a result, the policy exhibits sustained robustness, generating stable control commands despite photometric variations. A more expressive and intuitive experimental demonstration is provided in the supplementary video.

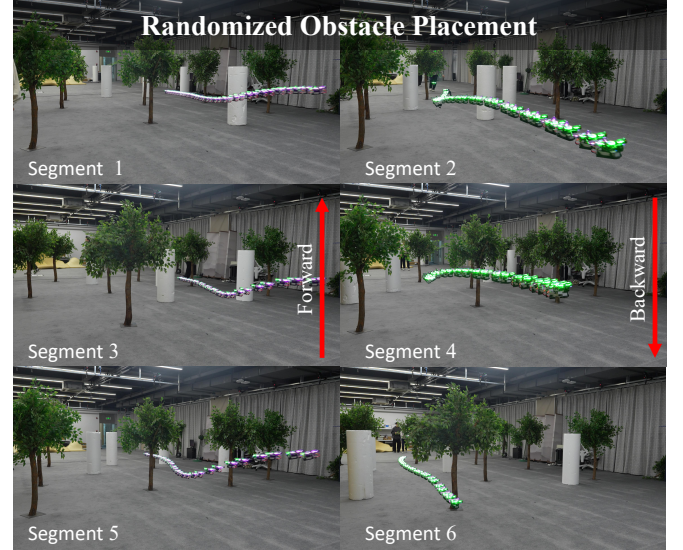


Fig. 6: **Demonstration of continuous navigation across varying environments.** The mission is divided into six sequential flight segments. Upon reaching the goal, the UAV initiates a return flight and obstacle positions are randomly redistributed. Left: Forward flight trajectory. Right: Return flight trajectory.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a framework combining 3DGS and domain adaptation for monocular RGB-based navigation. We leveraged end-to-end reinforcement learning to extract implicit cues from monocular data. By employing a pruning strategy and parallelized backends, we accelerated the 3DGS rendering speed to 30,000 frames per second. Through experiments in both simulated and real-world environments, we validated that our method achieves effective sim-to-real transfer by incorporating domain adaptation. Notably, the learned policy demonstrated robust navigation capabilities in environments with randomly distributed obstacles and maintained stability under varying illumination conditions.

Despite these results, we acknowledge that training within a limited set of scenes currently limits the policy’s adaptability to arbitrary environments. However, our method provides a cost-effective foundation for scaling up simulation data. Future work will focus on utilizing this framework to scale up training across large-scale, diverse 3DGS datasets. By integrating this with VLA [25] paradigms, we aim to bridge the gap towards universal navigation policies capable of handling complex, unknown scenarios.



Fig. 7: **Flight performance under rapidly changing illumination.** The lights are turned on and off rapidly, creating extreme distractions. The resulting smooth trajectory demonstrates the robustness of our policy in such challenging conditions.

REFERENCES

- [1] S. Suresh, H. Qi, T. Wu, T. Fan, L. Pineda, M. Lambeta, J. Malik, M. Kalakrishnan, R. Calandra, M. Kaess *et al.*, “Neuralfeels with neural fields: Visuotactile perception for in-hand manipulation,” *Science Robotics*, vol. 9, no. 96, p. eadl0628, 2024.
- [2] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu *et al.*, “Swarm of micro flying robots in the wild,” *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022.
- [3] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, “Demonstrating agile flight from pixels without state estimation,” *arXiv preprint arXiv:2406.12505*, 2024.
- [4] T. Wu, Y. Chen, T. Chen, G. Zhao, and F. Gao, “Whole-body control through narrow gaps from pixels to action,” in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 11 317–11 324.
- [5] G. Xu, T. Wu, Z. Wang, Q. Wang, and F. Gao, “Flying on point clouds with reinforcement learning,” *arXiv preprint arXiv:2503.00496*, 2025.
- [6] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, “Back to newton’s laws: Learning vision-based agile flight via differentiable physics,” *arXiv preprint arXiv:2407.10648*, 2024.
- [7] F. Sadeghi and S. Levine, “Cad2rl: Real single-image flight without a single real image,” *arXiv preprint arXiv:1611.04201*, 2016.
- [8] Y. Hu, Y. Zhang, Y. Song, Y. Deng, F. Yu, L. Zhang, W. Lin, D. Zou, and W. Yu, “Seeing through pixel motion: learning obstacle avoidance from optical flow with one camera,” *IEEE Robotics and Automation Letters*, 2025.
- [9] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, “3d gaussian splatting for real-time radiance field rendering,” *ACM Transactions on Graphics*, vol. 42, no. 4, July 2023. [Online]. Available: <https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/>
- [10] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa, “gsplat: An open-source library for gaussian splatting,” *Journal of Machine Learning Research*, vol. 26, no. 34, pp. 1–17, 2025.
- [11] H. Yu, C. De Wagter, and G. C. E. de Croon, “Depth transfer: Learning to see like a simulator for real-world drone navigation,” *IEEE Robotics and Automation Letters*, 2025.
- [12] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation,” 2015. [Online]. Available: <https://arxiv.org/abs/1409.7495>
- [13] 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.
- [14] A. Singla, S. Padakandla, and S. Bhatnagar, “Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge,” *IEEE transactions on intelligent transportation systems*, vol. 22, no. 1, pp. 107–118, 2019.
- [15] A. Al-Kaff, F. García, D. Martín, A. De La Escalera, and J. M. Armingol, “Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for uavs,” *Sensors*, vol. 17, no. 5, 2017. [Online]. Available: <https://www.mdpi.com/1424-8220/17/5/1061>
- [16] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao, “Depth anything v2,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 21 875–21 911, 2024.
- [17] Z. Zhang, A. Gupta, H. Jiang, and H. Singh, “Neuflow v2: High-efficiency optical flow estimation on edge devices,” *arXiv preprint arXiv:2408.10161*, vol. 6, p. 12, 2024.
- [18] 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. [Online]. Available: <https://arxiv.org/abs/2503.03984>
- [19] A. Hanson, A. Tu, G. Lin, V. Singla, M. Zwicker, and T. Goldstein, “Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives,” in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 21 537–21 546. [Online]. Available: <https://speedysplat.github.io/>
- [20] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [21] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, “Asymmetric actor critic for image-based robot learning,” *arXiv preprint arXiv:1710.06542*, 2017.
- [22] Z. Han, X. Huang, Z. Xu, J. Zhang, Y. Wu, M. Wang, T. Wu, and F. Gao, “Reactive aerobatic flight via reinforcement learning,” *arXiv preprint arXiv:2505.24396*, 2025.
- [23] S. Yang, W. Yu, J. Zeng, J. Lv, K. Ren, C. Lu, D. Lin, and J. Pang, “Novel demonstration generation with gaussian splatting enables robust one-shot manipulation,” *arXiv preprint arXiv:2504.13175*, 2025.
- [24] J. Greene, “Feature subset selection using thornton’s separability index and its applicability to a number of sparse proximity-based classifiers,” in *Proceedings of annual symposium of the pattern recognition association of South Africa*, 2001.
- [25] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi *et al.*, “Open-vla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024.