

ClearNav: Perception-Aligned Quadrotor Flight Using Differential-Physics

Aditya Azad^{*1}, Arthur Huang^{*2}, and Guanrui Li¹

Abstract—Autonomous quadrotor flight in cluttered environment is an key capability for tasks like inspection and search. Navigation policies trained in simulation and transferred to hardware can fly at high speed in open space, but they can still collide as obstacle clearance shrinks. We attribute most of these collisions to a single mechanism, namely motion in directions the onboard camera does not observe. Heading that is not aligned with the direction of travel and lateral excursion during obstacle avoidance both carry the vehicle toward obstacles outside its field of view. We introduce ClearNav, a navigation policy trained with two reward terms that constrain motion to the observed direction, a forward-looking heading target and a penalty on lateral velocity. A third term penalizes speed near obstacles so that an obstacle entering the field of view remains avoidable. In simulation, across 18 held-out scenarios with 2304 paired flights per policy unseen in training, ClearNav improves success by 6.8 points, cuts the collision rate by 65% relative to the state of the arts, and flies at maximum 5m/s. For the proximity term alone under matched commanded speeds, the collision reduction still holds at 70%. The effect is significant on static scenes and consistent in direction on moving obstacles, though not significant there at the suite level, showing that safer flight in clutter is possible even on the limited sensors and compute of small drones. We also validate our navigation policy in a real-world cluttered environment, showing that it enables the quadrotor to navigate through the cluttered environment successfully and safely.

SUPPLEMENTARY MATERIAL

Website: <https://acp-lab.github.io/clearnav>

Code: <https://github.com/acp-lab/clearnav>

Video: <https://youtu.be/9hldGKQiQgc>

I. INTRODUCTION

Autonomous quadrotor flight in cluttered, confined spaces is a core requirement for search and rescue, and for inspecting forests or collapsed structures. Confined geometry restricts platform size, and thus onboard sensing and compute, so the vehicle must stay safe at small clearances under strict computational budgets.

Classical pipelines decompose navigation into mapping, planning, and control [1]–[8]. Inaccuracies propagate across stages, and the accumulated latency is too high for fast flight in clutter. Hybrid learning-plus-optimization variants recover some latency but still rely on an explicit map [9]–[11], as do pure optimization methods [12]. Their fixed-rate replanning leaves less margin as speed grows, and all demand dense sensing with a heavy SWaP burden on

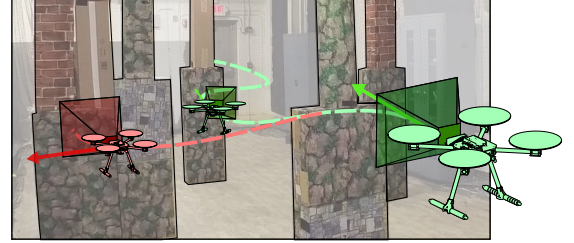


Fig. 1: ClearNav, an end-to-end navigation policy trained through differentiable physics, enables a drone to fly through cluttered environment while dodging obstacles by aligning its camera with its direction of travel and slowing down near obstacles.

small platforms. Imitation learning replaces the pipeline with a direct sensorimotor mapping [13]–[15], but the student is bounded by the expert’s performance and degrades out of distribution when the expert uses privileged information [14]. Reinforcement learning removes the expert [16]–[22], but its sampled gradients need thousands of parallel environments. Model-based variants trade sample efficiency for a world model whose long-horizon rollouts can fail to generalize [23], [24], and reward shaping reaches the policy only through indirect, high-variance sampled correlations.

Differentiable physics instead differentiates the entire rollout and updates the policy by backpropagation through the simulator [25]–[31], an idea that extends to perceptive legged control [32]. Some methods retain an RL actor or critic and are better read as first-order hybrids [33], [34], surveyed in [35]. The analytic gradients generalize to new environments from few simulation samples [27], but differentiating long unrolled rollouts is ill-conditioned and the dynamics Jacobian can become chaotic [36]. The point-mass dynamics sacrifice rotational degrees of freedom, and the privileged shaping signals are unavailable at deployment. As a result, these policies still struggle in environments that need deliberate, multi-step avoidance rather than purely reactive maneuvers: the physics-based rewards that keep training tractable give little incentive for such behavior. Later work shapes the objective with global information available at training time [25], [28], but this oracle is not used to its full potential. In our experiments these approaches fail in cluttered environments, with two recurring causes: misaligned yaw during motion, and overly wide, unsafe trajectories. The point-mass lineage [27] penalizes only the closing-speed component of velocity, leaving tangential motion free. General-purpose GPU frameworks such as DiffAero [37]

^{*}Equal contributors

¹Authors are with Worcester Polytechnic Institute, MA, USA. email: aazad@wpi.edu, gli7@wpi.edu

²The Author is with Northeastern University, MA, USA. email: huang.arth@northeastern.edu

accelerate training of such policies but are validated on agile-flight benchmarks rather than deliberate avoidance in tight clutter. We introduce a direction-agnostic speed-proximity cost as one term in a navigation objective for tight clutter, paired with heading and strafe-aware penalties.

To address this gap, we introduce **ClearNav**, a safer differentiable-physics policy for cluttered spaces. ClearNav adds three reward terms to an already-tuned differentiable-rollout objective [25], chosen so that no degree of freedom is sacrificed.

Specifically, the contributions of this work are:

- **Safer navigation in cluttered environments:** We penalize high speed near obstacles, misaligned yaw, and excessive strafing. Each penalized behavior is observable at inference time, so the policy keeps the information it needs to act on the objective during deployment.
- **Navigation around large obstacles:** Our policy navigates around obstacles across a wide range of sizes, shapes, and counts across simulations and real world experiment including large obstacles.
- **Thorough simulation experiments:** We compare against other existing approaches on static out-of-distribution scenes, a matched-velocity control, moving obstacles, and a cross-framework setting.

II. METHODOLOGY

This section describes how the policy is trained. Section II-A states the navigation problem and defines the observation and action. Section II-B introduces the differentiable point-mass dynamics that let us backpropagate through entire rollouts. Section II-C describes the Habitat training scenes and the geodesic guidance field that shapes the objective without entering the policy. Section II-D presents the three-branch recurrent policy architecture. Section II-E details the per-step objective and the three loss terms we add to it. Section II-E.7 describes the two-stage training procedure that carries the weights from an empty arena into cluttered scenes.

A. Problem formulation

A quadrotor flies autonomously from a randomly sampled start pose to a goal position $\mathbf{p}_g \in \mathbb{R}^3$ through an unknown, cluttered environment, using only a forward-facing depth camera and proprioception. We use three frames (Fig. 2): an ENU world frame $\mathcal{W}$, a start frame $\mathcal{S}$ fixed at the reset pose, and a body frame $\mathcal{B}$ at the center of mass, with rotations $R_{ws}, R_{wb} \in SO(3)$. At each step t the policy receives the observation

$$\mathbf{o}_t = (\mathbf{s}_t, \mathbf{g}_t, D_t), \quad (1)$$

where the proprioceptive state $\mathbf{s}_t \in \mathbb{R}^7$ stacks the attitude quaternion of $\mathcal{B}$ relative to $\mathcal{S}$ and the velocity expressed in $\mathcal{S}$, both perturbed by Gaussian noise resampled every step. The goal encoding $\mathbf{g}_t \in \mathbb{R}^4$ is a clamped proportional velocity command $\mathbf{v}_t^{\text{cmd}} = \min(K_p \|\mathbf{p}_g - \mathbf{p}_t\|, v^*) (\widehat{\mathbf{p}_g - \mathbf{p}_t})$, where K_p is a proportional gain and v^* is the command speed cap, together with an inverse range $[\max(\|\mathbf{p}_g - \mathbf{p}_t\|, d_{\min})]^{-1}$, with $d_{\min} = 0.5$ m, and the hat $(\cdot)$ denotes the normalized

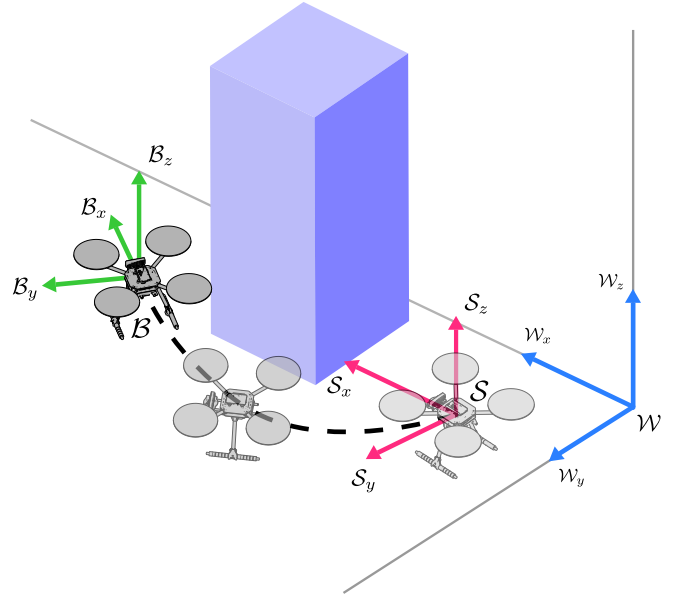


Fig. 2: Frames used in our formulation. All frames use right handed convention with positive z-axis pointing opposite to gravity at hover. World frame is indicated with blue, start frame with red and body frame with green.

unit vector of a vector. The command points straight at the goal and carries no obstacle information, so all avoidance must come from the depth image $D_t \in \mathbb{R}^{1 \times H \times W}$ of height H and width W , whose invalid returns are set to the far plane and treated as free space.

The action is a mass-normalized collective thrust vector $\mathbf{f}_t$ in $\mathcal{S}$ plus an absolute yaw setpoint ψ_t ,

$$\mathbf{a}_t = (\mathbf{f}_t, \psi_t) \in \mathbb{R}^3 \times (-\pi, \pi]. \quad (2)$$

We denote our navigation policy as π_θ , parameterized by θ . The navigation policy will take observation $\mathbf{o}_t$ and the policy's recurrent state $\mathbf{h}_t$ as input and output control action $\mathbf{a}_t$, so $\mathbf{a}_t = \pi_\theta(\mathbf{o}_t, \mathbf{h}_t)$. Since the simulator is differentiable end to end, we optimize θ directly rather than via policy gradients:

$$\min_{\theta} \mathbb{E} \left[\frac{1}{H} \sum_{t=0}^{H-1} \gamma^t \mathcal{L}(\mathbf{x}_t, \mathbf{a}_t) \right], \quad (3)$$

where $\mathbf{x}_t$ is the full simulator state, H the rollout horizon, and the expectation is over initial conditions, scenes, and the injected noise. The gradient $\partial \mathcal{L} / \partial \theta$ is obtained by backpropagation through the unrolled dynamics (Sec. II-E.7).

B. Differentiable dynamics

The vehicle is modeled as a point mass with an attitude reconstructed from the commanded thrust and yaw [25], [27], an abstraction that can struggle where significant yaw motion is required. Translational dynamics are integrated with velocity Verlet at step Δt under net acceleration $\mathbf{a}_t^{\text{net}} = R_{ws} \mathbf{f}_t + \mathbf{g}$,

$$\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_t \Delta t + \frac{1}{2} \mathbf{a}_t^{\text{net}} \Delta t^2, \quad (4)$$

$$\mathbf{v}_{t+1} = \mathbf{v}_t + \frac{1}{2} (\mathbf{a}_t^{\text{net}} + \mathbf{a}_{t+1}^{\text{net}}) \Delta t. \quad (5)$$

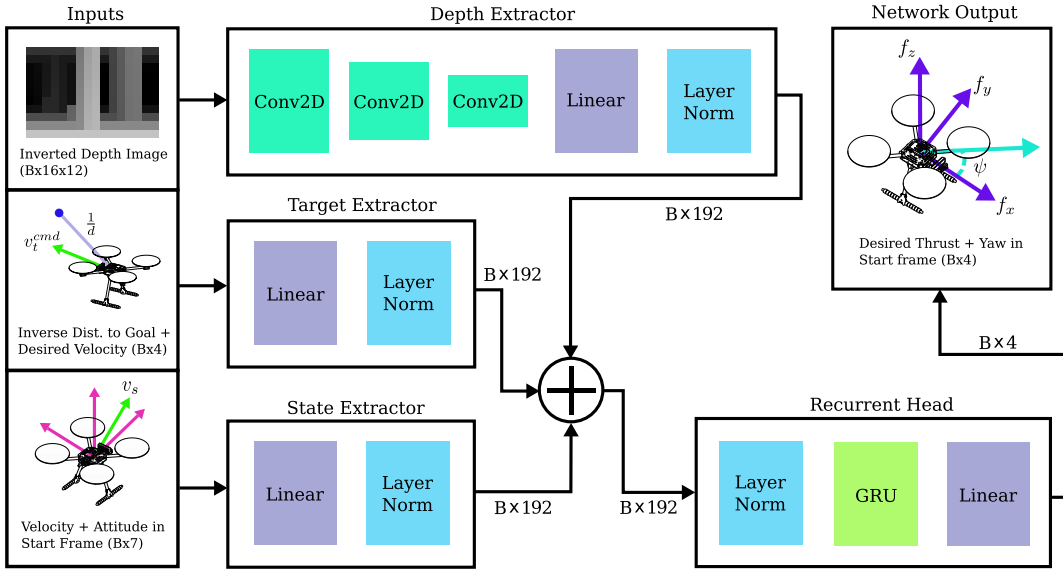


Fig. 3: Policy architecture. The three modality branches are projected to a common width n_f and summed, so the recurrent input dimension does not grow with the number of sensors.

Gravity $\mathbf{g}$ is resampled per episode from $\mathcal{N}(-9.81 \mathbf{e}_3, \sigma_g^2)$ on the vertical axis, where $\mathbf{e}_3$ is the world vertical unit vector and σ_g is the gravity noise standard deviation, which absorbs unmodeled mass and thrust-coefficient error.

Attitude is not integrated. Given the filtered thrust command $\mathbf{f}$ and yaw setpoint ψ , the body frame is reconstructed in closed form from its axes $\mathbf{x}_b, \mathbf{y}_b, \mathbf{z}_b$ so that $\mathbf{z}_b$ is along the thrust and $\mathbf{x}_b$ is as close as possible to the commanded heading $\mathbf{h} = R_{ws} R_z(\psi) \mathbf{e}_1$, where $R_z(\psi)$ rotates about the vertical axis by ψ and $\mathbf{e}_1$ is the start-frame forward axis, with $\mathbf{y}_b$ completing the right-handed orthonormal triad,

$$\mathbf{z}_b = \hat{\mathbf{f}}, \quad \mathbf{y}_b = \widehat{\mathbf{z}_b \times \mathbf{h}}, \quad \mathbf{x}_b = \widehat{\mathbf{y}_b \times \mathbf{z}_b}, \quad (6)$$

with degenerate cases handled by fixed fallbacks. Body rate $\boldsymbol{\omega}$, recovered by finite-differencing consecutive rotations, is differentiable and penalized directly.

Unrolling (5) for hundreds of steps produces the same ill-conditioning that afflicts long BPTT chains. To solve this, gradients flowing back through the BPTT rollout are shrunk by $e^{-\lambda_d T}$ over a horizon T , bounding the total gain.

C. Environment and geodesic guidance

Training scenes are built on Habitat-sim [38]. The nearest-obstacle query is exact: the closest surface point $\mathbf{c}_t$ gives a signed clearance

$$d_t = \|\mathbf{c}_t - \mathbf{p}_t\| - r, \quad (7)$$

where r is the vehicle's collision radius. An episode terminates in collision when $d_t < 0$ or on leaving the stage, and in success when $\|\mathbf{p}_g - \mathbf{p}_t\| < r_{\text{succ}}$.

$$\hat{\mathbf{u}}(\mathbf{p}) = -\widehat{\nabla \Psi(\mathbf{p})} \quad (8)$$

where Ψ is the scalar time-of-arrival potential and $\hat{\mathbf{u}}$ is its unit descent direction. This field is privileged, only shaping the objective and never an input to π_θ .

D. Policy architecture

π_θ is a three-branch encoder followed by a single recurrent cell and a linear head (Fig. 3). The depth branch inverts the range map, $\tilde{D} = 1/(D + \epsilon)$, then max-pools it to a coarse $H' \times W'$ grid. Max-pooling inverse depth preserves the closest obstacle in each cell. Three stride-1 convolutions with 32/64/128 channels and leaky-ReLU, a linear projection to n_f , and layer normalization follow. The state and goal branches are each a layer-normalized MLP to width n_f .

The three feature vectors are summed, layer-normalized, passed through a GRU cell of width n_f , and mapped by one linear layer to an unbounded thrust vector and a yaw angle squashed by a sigmoid into $(-\pi, \pi]$. The whole network is $\approx 2.2\text{M}$ parameters, which can run on many low power single board computers.

E. Objective

The per-step cost is a weighted sum of nine terms. Writing $\bar{\mathbf{v}}$ for the uniform moving average of velocity over a window T_{avg} , $\bar{\mathbf{v}}^e$ for its exponentially weighted counterpart, $\mathbf{v}_b = R_{wb}^\top \mathbf{v}$ for the velocity expressed in $\mathcal{B}$, and $\hat{\mathbf{c}}_t$ for the unit vector toward the nearest obstacle,

$$\begin{aligned} \mathcal{L} = & \lambda_v \mathcal{L}_v + \lambda_{\bar{\mathbf{v}}} \mathcal{L}_{\bar{\mathbf{v}}} + \lambda_c \mathcal{L}_c + \sum_{k \in \{a, j, \omega\}} \lambda_k \mathcal{L}_k \\ & + \lambda_{\text{oa}} \mathcal{L}_{\text{oa}} + \lambda_{\text{slip}} \mathcal{L}_{\text{slip}} + \lambda_\psi \mathcal{L}_\psi \end{aligned} \quad (9)$$

and the reward reported in the logs is $r = -\mathcal{L}$.

1) *Progress*: The vehicle is asked to track the routed direction at the commanded speed, penalized with a Huber loss on the magnitude of the tracking error, plus a one-sided quadratic that discourages exceeding the commanded speed:

$$\mathcal{L}_v = \text{huber}(\|\mathbf{v}^* \hat{\mathbf{u}} - \bar{\mathbf{v}}\|), \quad (10)$$

$$\mathcal{L}_{\bar{\mathbf{v}}} = (\|\mathbf{v}\| - v^*)^2_+. \quad (11)$$

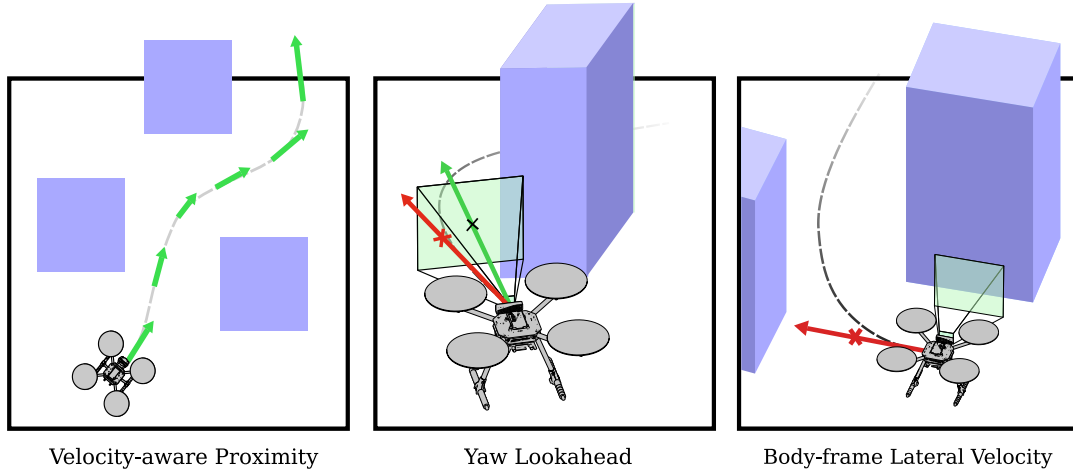


Fig. 4: Our modifications to the loss function include three components: a velocity-aware proximity penalty, which penalizes high speed near obstacles regardless of direction; a yaw lookahead reward, which encourages the policy to align its heading with the anticipated velocity direction; and a body-frame lateral-velocity penalty, which discourages sideways motion.

Using the windowed average $\bar{v}$ rather than the instantaneous velocity in (10) is what allows a lateral excursion around an obstacle without being immediately penalized for it.

2) *Collision*: The safety term combines a closing-speed penalty with a soft interior barrier,

$$\mathcal{L}_c = \underbrace{(\mathbf{v} \cdot \hat{\mathbf{c}})_+ (d_f - d)_+^2}_{\text{closing speed}} + \underbrace{\beta_1 \log(1 + e^{\beta_2 d})}_{\text{barrier}}, \quad (12)$$

Here d is the clearance d_t from (7), and $\beta_2 < 0$ so the barrier is negligible for $d \gg 0$ and grows linearly once the surface is penetrated. The closing-speed factor penalizes radial approach within a shell of thickness d_f .

3) *Smoothness*: $\mathcal{L}_a = \|\mathbf{a}\|^2$, $\mathcal{L}_j = \|\mathbf{j}\|^2$ and $\mathcal{L}_\omega = \|\boldsymbol{\omega}\|^2$ penalize acceleration, jerk, and body rate, where here $\mathbf{a}$ is the translational acceleration and $\mathbf{j}$ the jerk, making trajectories flyable.

4) *Heading*: The heading reference interpolates along the great circle from the goal direction to the smoothed velocity direction, since early in an episode there is no motion to look along,

$$\hat{\mathbf{y}}^{\text{ref}} = \text{slerp}(\widehat{\mathbf{p}_g - \mathbf{p}}, \widehat{\mathbf{v}^e}; \text{clip}(\|\mathbf{p} - \mathbf{p}_0\|, 0, 1)), \quad (13)$$

$$\mathcal{L}_\psi = -\hat{\mathbf{y}}^{\text{ref}} \cdot \hat{\mathbf{y}}, \quad (14)$$

where $\hat{\mathbf{y}}$ is the body x axis projected onto the horizontal plane, blended by distance traveled since the reset position $\mathbf{p}_0$, clamped to $[0, 1]$. $\mathcal{L}_\psi$ is gated off within 1 m of the goal, where the goal direction is ill-conditioned.

When geodesic guidance is available, the slerp reference of (13) is replaced by the routed direction sampled τ_ℓ seconds ahead of the vehicle along its smoothed velocity,

$$\hat{\mathbf{y}}^{\text{ref}} = -\nabla \Psi(\widehat{\mathbf{p} + \tau_\ell \bar{\mathbf{v}}^e}), \quad (15)$$

wherever that gradient is valid. Where the look-ahead point lies inside geometry, the slerp fallback of (13) is kept. Sampling ahead of $\mathbf{p}$ turns the heading before a bend, so the depth camera leads the path. This gradient, like $\hat{\mathbf{u}}$ in (10), only shapes training and is never an input to π_θ .

5) *Velocity-aware proximity cost*: Equation (12) penalizes only radial approach, so any velocity orthogonal to $\hat{\mathbf{c}}$ is free, and its barrier fires only once d is near zero, too late for a vehicle whose stopping distance at 5 m/s is several meters [27]. We therefore add a cost on *being fast while close*:

$$\mathcal{L}_{\text{oa}} = \|\mathbf{v}\| \exp(-2(d - \delta)_+) \quad (16)$$

where δ is an inflation radius.

The term is direction-agnostic, since $\mathcal{L}_{\text{oa}}$ depends only on $\|\mathbf{v}\|$, so the only way to reduce it near an obstacle is to slow down. Its gradient $\partial \mathcal{L}_{\text{oa}} / \partial \mathbf{v} = \hat{\mathbf{v}} \exp(-2(d - \delta)_+)$ opposes the current velocity with magnitude set by proximity, and the exponential falloff keeps $\mathcal{L}_{\text{oa}}$ negligible in open space, so it can be added without re-tuning the other weights.

6) *Body-frame lateral-velocity penalty*: (14) asks the camera to look where the vehicle is going, but thrust is commanded in $\mathcal{S}$, not $\mathcal{B}$, so lateral body velocity is no costlier than forward and a policy can satisfy $\mathcal{L}_\psi$ while flying broadside. We close this with a penalty on the body-frame lateral velocity,

$$\mathcal{L}_{\text{slip}} = v_{\text{lat}}^2 (\|\mathbf{v}\| + 1), \quad v_{\text{lat}} = (\mathbf{v}_b)_y, \quad (17)$$

where v_{lat} is the strafe-axis component of $\mathbf{v}_b = R_{wb}^\top \mathbf{v}$. The speed factor $\|\mathbf{v}\| + 1$ is detached from the autograd graph, so gradient flows only through the quadratic on v_{lat} , making fast sideways motion expensive while leaving the small transient lateral component of a coordinated turn nearly free. $\mathcal{L}_{\text{slip}}$ carries no obstacle or scan reference, so unlike $\mathcal{L}_{\text{oa}}$ it cannot be farmed by routing around geometry, and its gradient is a pure anti-strafe command leaving the forward and vertical components free.

7) *Training*: We roll N agents forward for H steps, accumulate the discounted cost of (3), and backpropagate through the whole unrolled graph in a single backward pass, covering the dynamics, the clearance query, and the

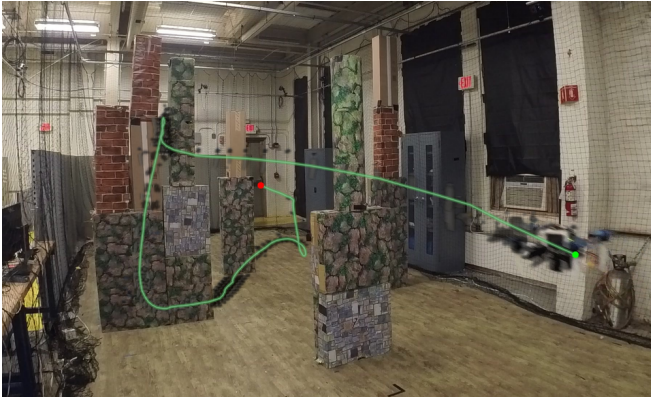


Fig. 5: Real world trajectory flown by the ClearNav policy. The drone moves left to avoid the first obstacle, then after looking at it tries to go below, yaws to the right to avoid the second, wider obstacle, and continues to the goal.

reward. Parameters are updated with AdamW under a cosine-annealed learning rate and gradient-norm clipping at $g_{\max}$, and after each iteration the simulator and recurrent states are detached, truncating the graph while preserving physical continuity. Agents that terminate are reset individually, with the discount and recurrent state zeroed for those rows only.

Training proceeds in two stages that carry the weights forward and continue the global iteration counter. The first stage uses an empty walled arena so the policy learns the thrust-to-motion mapping and goal reaching, and the second stage uses cluttered scenes with time-of-arrival guidance and the full objective.

III. EXPERIMENTAL RESULTS

We validate our methods by answering four questions:

- Q1** Does ClearNav help on the scenes it was trained on (Sec. III-B)?
- Q2** Does it help out of distribution, on obstacle configurations, densities, arena sizes, and speed bands never seen in training (Sec. III-C)?
- Q3** Does the benefit survive when the confounding factors are controlled (Sec. III-D)?
- Q4** Does the benefit survive when the obstacles move (Sec. III-E)?

The first is answered in the negative on the training distribution, where the proximity term adds nothing measurable (Sec. III-B). The bundle’s gains appear only out of distribution (Sec. III-C). A few static scenarios where the baseline matches or beats it, and a weaker, non-significant effect on moving obstacles (Sec. III-E), bound the scope of the claim.

A. Experiment Setup

We compare four policies:

- **B** (baseline): the original DepthNav reward, i.e. (9) with the three new terms disabled (avoidance driven by $\mathcal{L}_c$ alone); stage-1 cruise speed $\sim \mathcal{U}[1, 5]$ m/s.
- **ClearNav** (ours): the full objective activating $\mathcal{L}_{oa}$, $\mathcal{L}_{slip}$, and the look-ahead yaw of (15); cruise speed



Fig. 6: Real world trajectory flown by the baseline policy. The drone drifts toward the net to the side of the course, crashes into it, and never reaches the goal.

$\sim \mathcal{U}[0.5, 10]$ m/s, with a longer stage-1 episode limit for the slower end.

- **OA-only** (ablation): $\mathcal{L}_{oa}$ alone on top of B; appears only in the matched-velocity control (Sec. III-D) and speed sweep (Sec. III-E), where the full bundle was not re-run.
- **DA** (DiffAero reference): the v1 policy trained in the DiffAero simulator [37], evaluated in that simulator on the same scenario specifications. Sec. III-F explains why this benchmarks whole stacks rather than the reward term.

This is not a single-variable ablation. ClearNav differs from B in three objective terms ($\mathcal{L}_{oa}$, $\mathcal{L}_{slip}$, and the look-ahead yaw reference) and the commanded-speed distribution used during stage-1 training. The latter directly causes the mean-speed gap in every table below. Sec. III-D isolates it with a control experiment. OA-only provides one ablation point, the proximity term alone. Separating $\mathcal{L}_{slip}$ from the look-ahead yaw reference is not attempted here.

All evaluations use held-out scenes, with seeds disjoint from training and no shared scene, layout, or start-goal pair. Both policies run under the same environment seed, so poses, goals, and obstacle placements (and trajectories, Sec. III-E) are *paired*. An episode is a success within $r_{\text{succ}} = 0.5$ m of the goal, a collision on geometry contact or leaving the stage, or a timeout otherwise. The collision radius is 0.25 m, below the 0.3 m used in training. Statistical statements are Wilcoxon signed-rank tests over *scenarios*, with each scenario’s paired success rate as one matched observation. We include footage of the simulation environments in the supplementary video for the reader.

B. In-distribution evaluation

During training, baseline and OA-only are indistinguishable on a held-out set from the stage-1 scene family, sampled every 100 iterations at 80 episodes and commanded $[1, 5]$ m/s. Both converge to $\approx 99\%$ success and $\approx 1\%$ collision, with identical mean speed ≈ 2.5 m/s. On the training distribution the proximity term thus buys nothing measurable, and costs nothing: it does not slow the policy,

degrade goal-reaching, or destabilize training. Whether the bundle matters must be decided out of distribution.

C. Static generalization suite

To probe generalization we built an 18-scenario suite sweeping obstacle count $|\mathcal{O}| \in [5, 80]$, shape mix, arena size $L \in [10, 25]$ m, commanded speed band $[0.5, 1.5]$ to $[7, 10]$ m/s, and spatial clustering. Obstacles form a 2D Gaussian blob centered on the goal, so every path must traverse the clutter. The goal sits at the arena center and the vehicle spawns on a cylinder of radius $L/2$. Each scenario is $16 \text{ scenes} \times 8 \text{ rollouts} = 128$ paired episodes per policy, 2304 total, with a 1500-step, 30 s limit.

Table I shows the innovations raise mean success from 90.3% to 97.1% while cutting the collision rate from 8.2% to 2.9%, a 65% relative reduction. The improvement is consistent, not driven by a few scenarios. ClearNav is better on 17 of 18 scenarios and worse on none, with one exact tie. This is significant under a Wilcoxon signed-rank test over scenarios, Holm-corrected $p = 3 \times 10^{-4}$. Pooling episodes gives 2237/2304 against 2080/2304.

The gain is largest where (16) predicts: dense clutter, clustering, and the tight arena. In these scenes, the baseline loses most, while the near-saturated very-sparse scenario shows no difference. The tight-cluster case is ClearNav's weakest, with a higher collision rate, plausibly dominated by its higher speed.

This is achieved while flying faster, not slower. ClearNav's mean speed of 3.39 m/s against B's 2.24 m/s is a 52% increase, the opposite of the usual safety-speed trade-off. Most of the speed gap comes from the wider stage-1 training band rather than the objective terms. Whether the safety gain is attributable to the objective terms or the training band is resolved by Sec. III-D.

D. Controlling for commanded speed

To separate the proximity term from the training speed band, we re-ran a subset of the suite with the commanded band forced to $\mathcal{U}[2, 4]$ m/s for both policies on the same scenes and seeds, using OA-only rather than the full-bundle ClearNav. The full bundle was not re-run under matched velocity, so this controls the confound for $\mathcal{L}_{\text{oa}}$ alone, not the bundle. Since $\mathcal{L}_{\text{slip}}$ and the look-ahead yaw reference are not speed-band-dependent in the same way, we take what it establishes for $\mathcal{L}_{\text{oa}}$ as a lower bound on the bundle.

Table II shows success improving from 90.6% to 97.2% and collisions falling from 8.4% to 2.6%, a 70% relative reduction, over 512 paired episodes on the four re-run scenarios. The direction and magnitude of Table I survive. The two policies still differ in speed under a matched command, 2.73 vs 2.04 m/s, so the control is partial, but it rules out the strong confound: with OA-only still faster, it collides less often, which a speed-only explanation does not predict.

E. Moving obstacles

$\mathcal{L}_{\text{oa}}$ is computed from an instantaneous clearance without modeling obstacle motion, and neither policy saw a moving

obstacle in training, so this suite probes transfer on both counts. Obstacles are relocated each control step on trajectories seeded identically for both policies. Eleven scenarios cover linear drift, crossing traffic, orbits about the goal, oscillating pillars, and a dense fast swarm, which we showed in the supplementary video. Five more mix movers with a fixed static layout. Each scenario is $8 \text{ scenes} \times 3 \text{ rollouts} = 24$ paired episodes per policy with a 600-step limit. We report the suite with obstacle-obstacle separation enforced and, secondarily, without it. The correction is deterministic and pairing is preserved in both variants.

Table III shows the results. With separation enforced, mean success improves from 64.0% to 73.9% and collisions fall from 36.0% to 26.1%, winning 7 scenarios, losing 3 and tying 1, but this does not reach significance at $n = 11$, $p = 0.084$. Without separation, the effect is of similar magnitude, $68.6\% \rightarrow 76.1\%$ with $p = 0.19$, and in the mixed scenarios ClearNav leads 80.0% to 71.7%, again not significant. We interpret this as no evidence that ClearNav hurts on moving obstacles and a weak, consistent-in-sign suggestion that it helps, with 24 episodes per scenario too few to establish it. The mechanism is also plausibly limited: it slows near a wall, but if drone approaches to the obstacle at 5m/s, the safe action may be to accelerate away.

The crossing-traffic sweep in Table IV isolates obstacle speed. The sweep is run on OA-only to characterize the proximity term's transfer to dynamic obstacles. Both policies degrade sharply with crossing speed, from 80–95% success at 1 m/s to 33–38% at 5 m/s, and the ordering between them is not stable. Neither policy handles fast dynamic obstacles, unsurprising since neither saw one in training, and the proximity term does not change this.

F. Cross-framework comparison

The proximity cost (16) is a direction-agnostic term we introduce (Sec. II-E.5). For context, the DA columns of Table I evaluate the DiffAero reference policy [37] on the same 18 scenario specifications in the DiffAero simulator. On the 18 ClearNav scenarios, DA's mean success is 61.8% against 90.3% for B and 97.1% for ClearNav. This is not a model comparison. The stacks differ in renderer, depth resolution, dynamics, termination criteria, and episode limits, so the numbers benchmark whole stacks, not the reward term.

G. Real world flight

We also fly the ClearNav policy on a physical quadrotor to confirm that the simulator-trained behavior transfers to real hardware, with the policy running onboard on a Jetson Orin NX (Fig. 5). The scene holds four obstacles of different shapes along the path to the goal. The first is narrow, so the drone moves left to avoid it, then once it has looked at the obstacle it tries to go below it. Since there is ground below it does not move further and it keeps its altitude low while trying to yaw around the second obstacle, which is a wider variety, so the drone intentionally yaws to the right to clear it before continuing the rest of the trajectory to the goal. The baseline policy was not successful under the same

TABLE I: Static generalization suite: 18 scenarios, 128 episodes per scenario per policy. *DA* = DiffAero v1 (cross-simulator; Sec. III-F), *B* = DepthNav baseline, *ClearNav* = ours (baseline+ $\mathcal{L}_{\text{oa}}$ + $\mathcal{L}_{\text{slip}}$ + look-ahead yaw of (15)). Best of B/ClearNav in bold.

Scenario	O	L	v^* band	Success rate (%)			Collision rate (%)			Mean speed (m/s)	
				DA	B	ClearNav	DA	B	ClearNav	B	ClearNav
Baseline	30	25	3.0–6.0	75.4	89.8	96.1	12.7	10.2	3.9	2.36	3.53
Sparse	10	25	3.0–6.0	90.6	94.5	99.2	2.3	5.5	0.8	2.40	3.59
Dense	60	25	3.0–6.0	53.6	87.5	95.3	24.9	11.7	4.7	2.33	3.46
Spheres Heavy	30	25	3.0–6.0	73.8	90.6	96.9	13.1	8.6	3.1	2.37	3.51
Cubes Heavy	30	25	3.0–6.0	82.1	91.4	99.2	6.9	8.6	0.8	2.36	3.56
Short Dist	20	15	3.0–6.0	66.4	89.1	99.2	17.0	10.9	0.8	2.12	3.39
Slow Targets	30	25	1.5–3.0	57.7	87.5	99.2	4.0	7.8	0.8	1.73	2.10
Fast Targets	30	25	5.0–8.0	56.1	90.6	96.9	39.8	9.4	3.1	2.55	4.42
Clustered	40	25	3.0–6.0	51.1	87.5	92.2	26.3	10.2	7.0	2.32	3.49
Baseline Reseed	30	25	3.0–6.0	79.3	90.6	99.2	12.6	7.0	0.8	2.33	3.53
Very Dense	80	25	3.0–6.0	34.5	87.5	93.8	28.8	10.9	6.2	2.32	3.43
Very Sparse	5	25	3.0–6.0	94.6	100.0	100.0	2.2	0.0	0.0	2.42	3.56
Pure Spheres	30	25	3.0–6.0	78.8	91.4	99.2	11.0	7.0	0.8	2.36	3.54
Pure Cubes	30	25	3.0–6.0	82.0	92.2	99.2	6.3	6.2	0.8	2.33	3.58
Tight Arena	12	10	3.0–5.0	72.5	89.1	100.0	6.3	10.2	0.0	1.93	2.87
Ultra Fast	30	25	7.0–10.0	26.8	90.6	94.5	72.0	9.4	5.5	2.65	5.09
Ultra Slow	30	25	0.5–1.5	1.4	87.5	98.4	2.7	4.7	1.6	1.02	0.93
Tight Cluster	50	25	3.0–6.0	36.1	87.5	89.1	38.1	8.6	10.9	2.33	3.45
mean				61.8	90.3	97.1	18.2	8.2	2.9	2.24	3.39

TABLE II: Matched-velocity control, run on OA-only rather than the full-bundle ClearNav. Both policies are commanded the same $v^* \sim \mathcal{U}[2, 4]$ m/s band, removing the commanded-speed difference between the two training configurations. 128 episodes per scenario per policy. The full-bundle ClearNav was not re-run under matched velocity, so this controls the confound for the proximity term alone, not for the bundle; see Sec. III-D.

Scenario	Success (%)		Collision (%)		Speed (m/s)	
	B	OA-only	B	OA-only	B	OA-only
Baseline	89.1	100.0	9.4	0.0	2.03	2.71
Sparse	98.4	98.4	1.6	1.6	2.09	2.74
Dense	87.5	96.1	11.7	3.1	2.03	2.73
Very Dense	87.5	94.5	10.9	5.5	2.01	2.75
mean	90.6	97.2	8.4	2.6	2.04	2.73

conditions. We also tested the baseline policy in the same real-world scenario, but it flew too broadly and grazed the nets enclosing the flying area, or crashed directly into the net. We therefore conclude that ClearNav successfully transferred to real-world scenario.

IV. CONCLUSION

We presented ClearNav, a navigation policy for autonomous quadrotor flight in cluttered and confined spaces, where a small platform carries only a single forward-facing depth camera and proprioception. Three reward terms proposed in this paper on penalizing high speed near obstacles regardless of direction, turning the heading toward the geodesic gradient sampled ahead of the vehicle, and making sideways flight expensive makes ClearNav a safer policy than the baseline policies trained in differentiable physics simulator. Every behavior the objective penalizes is observable at inference. On the training distribution the proximity term costs nothing measurable. On 18 held-out static scenarios, ClearNav raises success from 90.3% to 97.1% and cuts

TABLE III: Moving-obstacle suite. Obstacles are relocated at the control rate on trajectories seeded identically for both policies. 24 episodes per scenario per policy. Primary columns enforce obstacle–obstacle separation; the right-hand pair repeats without it. v_o is a range for orbiters ($|\omega|r$, $|\omega| \in [0.3, 0.8]$ rad/s) and oscillators (speed varies within an episode up to the quoted peak).

Scenario	O	v_o	Success (%)		Collision (%)		Succ., no sep.	
			B	Ours	B	Ours	B	Ours
Slow Drift	10	0.5	95.8	100.0	4.2	0.0	91.7	95.8
Crossing	10	2.0	70.8	70.8	29.2	29.2	79.2	70.8
Orbiting	12	0.9–8.8	45.8	41.7	54.2	58.3	45.8	62.5
Oscillating	12	0–4.8	66.7	95.8	33.3	4.2	62.5	100.0
Dense Swarm	18	1.3	66.7	83.3	33.3	16.7	79.2	91.7
Crossing v1	10	1.0	66.7	100.0	33.3	0.0	83.3	87.5
Crossing v3	10	3.0	54.2	66.7	45.8	33.3	50.0	70.8
Crossing v4	10	4.0	66.7	58.3	33.3	41.7	70.8	66.7
Crossing v5	10	5.0	29.2	58.3	70.8	41.7	37.5	58.3
Swarm Fast	18	2.5	58.3	50.0	41.7	50.0	83.3	62.5
Drift Fast	10	2.0	83.3	87.5	16.7	12.5	70.8	70.8
mean			64.0	73.9	36.0	26.1	68.6	76.1
<i>mixed: static obstacles + movers</i>								
Static Drift	6+8	0.8	95.8	95.8	4.2	4.2	–	–
Static Crossing	6+8	2.0	87.5	87.5	12.5	12.5	–	–
Pillars Swarm	10+10	1.5	66.7	70.8	33.3	29.2	–	–
Dense Mixed	12+12	1.5	62.5	62.5	37.5	37.5	–	–
Static Orbit	8+8	0.9–8.8	45.8	83.3	54.2	16.7	–	–
mean			71.7	80.0	28.3	20.0	–	–

TABLE IV: Crossing-traffic speed sweep, run on OA-only. All five columns share obstacle seed 102, so the obstacle field and headings are identical and only the speed magnitude changes. Rates in %.

	obstacle speed v_o (m/s)				
	1	2	3	4	5
B success	83.3	79.2	50.0	70.8	37.5
OA-only success	95.8	54.2	58.3	70.8	33.3
B collision	16.7	20.8	50.0	29.2	62.5
OA-only collision	4.2	45.8	41.7	29.2	66.7

collisions by 65%, winning 17 of 18 while flying faster. On matched velocity control isolation the proximity term preserves a 70% collision reduction. On moving obstacles the effect points the same way but does not reach significance. We also flew the ClearNav policy on a physical quadrotor with the policy running onboard, and it navigated through a cluttered scene safely that the baseline could not, confirming that the behavior transfers to real hardware. One limitation is privileged supervision. Since the collision and proximity losses use an exact omnidirectional clearance query while the policy observes only a single 89° depth camera, so a learned maneuver need not be supported by an observation, a gap we plan to address in future work.

REFERENCES

- [1] F. Gao, Y. Lin, and S. Shen, "Gradient-based online safe trajectory generation for quadrotor flight in complex environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3681–3688.
- [2] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.
- [3] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 2872–2879.
- [4] B. Zhou, J. Pan, F. Gao, and S. Shen, "Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1992–2009, 2021.
- [5] S. Xu, D. Honegger, M. Pollefeys, and L. Heng, "Real-time 3d navigation for autonomous vision-guided mavs," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 53–59.
- [6] C. Richter, A. Bry, and N. Roy, *Polynomial Trajectory Planning for Aggressive Quadrotor Flight in Dense Indoor Environments*. Cham: Springer International Publishing, 2016, pp. 649–666.
- [7] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in se(3)," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2439–2446, 2018.
- [8] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger, "Fast frontier-based information-driven autonomous exploration with an mav," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 9570–9576.
- [9] G. Tang, W. Sun, and K. Hauser, "Learning trajectories for real-time optimal control of quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3620–3625.
- [10] Y. Wu, X. Sun, I. Spasojevic, and V. Kumar, "Deep learning for optimization of trajectories for quadrotors," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2479–2486, 2024.
- [11] Y. Jiang, Y. Zhan, H. V. Gupta, C. Borde, and J. Geng, "A self-supervised learning approach with differentiable optimization for uav trajectory planning," in *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [12] F. Kong, W. Xu, Y. Cai, and F. Zhang, "Avoiding dynamic small obstacles with onboard sensing and computation on aerial robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7869–7876, 2021.
- [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] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," *8th Conference on Robot Learning (CoRL)*, 2024.
- [15] J. Low, M. Adang, J. Yu, K. Nagami, and M. Schwager, "Sous vide: Cooking visual drone navigation policies in a gaussian splatting vacuum," *IEEE Robotics and Automation Letters*, vol. 10, no. 5, pp. 5122–5129, 2025.
- [16] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1205–1212.
- [17] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Mueller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, pp. 982–987, 08 2023.
- [18] A. Romero, A. Shenai, I. Geles, E. Aljalbout, and D. Scaramuzza, "Dream to fly: Model-based reinforcement learning for vision-based drone flight," in *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [19] Z. Han, X. Huang, Z. Xu, J. Zhang, Y. Wu, M. Wang, T. Wu, and F. Gao, "Reactive aerobatic flight via reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 10, no. 10, pp. 11 014–11 021, 2025.
- [20] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating Agile Flight from Pixels without State Estimation," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024.
- [21] G. Malczyk, M. Kulkarni, and K. Alexis, "Reinforcement learning for active perception in autonomous navigation," in *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [22] J. Zhang, C. Lei, C. Dai, K. Hoi, L. Wang, Z. Han, and F. Gao, "High-speed vision-based flight in clutter with safety-shielded reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 11, no. 8, pp. 9559–9566, 2026.
- [23] A. E. Verraest, S. Bahnam, R. Ferede, G. C. de Croon, and C. de Wagter, "Skydreamer: Interpretable end-to-end vision-based drone racing with model-based reinforcement learning," arXiv:2510.14783, 2025.
- [24] E. Vytla, B. Kalavakolanu, A. Perrault, and M. McCrink, "Dreaming falcon: Physics-informed model-based reinforcement learning for quadcopters," arXiv:2511.18243, 2025.
- [25] J. Lee, A. Rathod, K. Goel, J. Stecklein, and W. Tabib, "Quadrotor navigation using reinforcement learning with privileged information," in *2026 IEEE International Conference on Robotics and Automation (ICRA)*, 2026.
- [26] J. Heeg, Y. Song, and D. Scaramuzza, "Learning quadrotor control from visual features using differentiable simulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 4033–4039.
- [27] Y. Zhang, Y. Hu, Y. Song, D. Zou, and W. Lin, "Learning vision-based agile flight via differentiable physics," *Nature Machine Intelligence*, vol. 7, no. 6, pp. 954–966, Jun 2025.
- [28] Y. Su, F. Yu, Y. Hu, X. Niu, L. Zhang, F. Sun, and D. Zou, "Vector field augmented differentiable policy learning for vision-based drone racing," *IEEE Robotics and Automation Letters*, vol. 11, no. 5, pp. 5829–5836, 2026.
- [29] H. Zhong, P. Chi, J. Zhao, S. Yuan, X. Gao, T.-M. Nguyen, and L. Xie, "Panodp: Learning collision-free navigation with panoramic depth and differentiable physics," arXiv:2603.07644, 2026.
- [30] F. Li, S. Wang, Y. Huang, F. Sun, S. Wu, Y. Yan, D. Zou, and W. Yu, "Simple but stable, fast and safe: Achieve end-to-end control by high-fidelity differentiable simulation," arXiv:2604.10548, 2026.
- [31] X. Wang, Z. Chen, N. Cheng, Z. Yin, and X. Shen, "Learn for variation: Efficient aav trajectory learning through a differentiable wireless world model," arXiv:2603.18853, 2026.
- [32] J. Y. Luo, Y. Song, V. Klemm, F. Shi, D. Scaramuzza, and M. Hutter, "Residual policy learning for perceptive quadruped control using differentiable simulation," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025, pp. 1–8.
- [33] 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," in *2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025, pp. 7941–7948.
- [34] E. Xing, V. Luk, and J. Oh, "Stabilizing reinforcement learning in differentiable multiphysics simulation," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [35] R. Newbury, J. Collins, K. He, J. Pan, I. Posner, D. Howard, and A. Cosgun, "A review of differentiable simulators," *IEEE Access*, vol. 12, pp. 97 581–97 604, 2024.
- [36] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman, "Gradients are not all you need," arXiv:2111.05803, 2021.

- [37] X. Zhang, R. Wang, Y. Ren, J. Sun, H. Fang, J. Chen, and G. Wang, "Diffaero: A gpu-accelerated differentiable simulation framework for efficient quadrotor policy learning," arXiv:2509.10247, 2025.
- [38] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A Platform for Embodied AI Research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.