Early Release Report: Mastering the Quadrotor Skill Through Narrow Gaps in the Real World

Tianyue Wu[1], Guangtong Xu[2], Zihan Wang[2], Tianyang Chen[1], Fei Gao[1]

[1]Zhejiang University. [2]Huzhou Institution, Zhejiang University.

Controlling a micro quadrotor with only onboard sensing and computation perpetuates challenges despite researchers' best efforts in the community. Among the most relevant problems, a classic challenge is to servo a quadrotor to fly through body-sized narrow gaps. These gaps impose the quadrotor to utilize its asymmetrical vehicle to traverse the gaps where the quadrotor can be sharply rolled or pitched for a collision-free locomotion. However, due to the underactuated nature of the quadrotor, a particular pose can only last for a split second, posing the challenge of demanding rotor-eye coordination to take the right action at the right timing. In this paper, we present the first demonstration in the real world of quadrotor flight through narrow gaps with unknown positions and poses with very low clearances by directly mapping onboard vision and proprioception to control commands readily executed by a flight controller. The problem is solved by the paradigm of sim-to-real reinforcement learning (RL) with multi-expert observation space distillation. This end-to-end approach is preferred to the widely used modular approach due to its minimal compounding error and the scalability rendered by its purely data-driven nature. To obtain a strong policy in noisy simulation, we mitigate the fundamental hardness of RL's exploration on the restricted solution space featured by the problem with an informed reset strategy leveraging model-based trajectory optimization. Specialized training recipes and sim-to-real design choices allow the policy to control a physical quadrotor through narrow

gaps with clearances under 5cm at an unprecedented success rate. In particular, our method enables a quadrotor with a dimension 38cm×10cm (largest length measured between propeller tips) to traverse through a rectangle gap with a dimension 60cm×20cm **titled with an arbitrary roll angle (up to 90°) or pitch angle less than 70°, without the knowledge of the position and pose of the gap.**

The development of micro drones has been a history of freeing them from the protection of external measurement equipment (*1, 2*) and teleoperation to accomplishing autonomous flight with only sensors and computational resources on it (*3, 4*). This requires autonomy of perception-action closed loop at varying levels for different tasks. One of the most demanding tasks in terms of perception and motor coordination is to servo the drone through narrow gaps without any collision using only onboard sensing, as illustrated in Fig. 1. Mastering this skill can push the traversability of a drone to its limits, as one of the milestones for the drone to finally maneuver in scenarios with very restricted free space.

This setup demands the drone to exploit its asymmetrical vehicle shape and decide the right pose and timing to fly through the gap with low fault tolerance. The challenge is exacerbated when our drones, such as the most common quadrotors in everyday life, are underactuated. While these robots can fly with impressive maneuverability, the control of rotation and translation components of them are tightly coupled. In this way, the feasible action space can become very limited when the drone is faced with a body-sized narrow gap, where a careful action sequence to adjust its attitude along the flight progression is required. Therefore, there have been very few successful demonstrations to control quadrotors across narrow gaps with this whole-body control skill using only cheap onboard sensors and limited computation (*5, 6*). Researchers in (*6*) achieve real-time responses to the observations by programming a dedicated multi-stage trajectory planning pipeline for the specific rectangle gap with handcrafted state machine switching, and utilizes fixed visual landmarks for accurate state estimation. The impressive demonstrations in this work, however, come at the cost of overfitting to the setup and limited *scalability*: for instance, it can be fundamentally hard to directly extend this approach to gaps with novel geometries, and to variants of the task such as consecutive gaps traversals.

On the other hand, general full-state local trajectory optimization (TO) schemes designed for multirotors (*7*) can deal with this problem more flexibly with real-time local optimization techniques.

Beyond the local TO solutions that can result in arbitrary suboptimality, global and generalized planning methods, collectively known as Monte Carlo tree search (*8, 9*), have the potential to provide more robust solutions and enable task-generalized autonomy. However, in order to make the problem computationally tractable and to utilize robot dynamics that are typically described by rigid-body states, these methods often require state estimation (*3, 4, 6*) and a hierarchy of decision-making, e.g., trajectory planning and tracking control (*3, 6*). However, when confronted with narrow gaps where few knowledge is known in prior and only cheap onboard sensing is available, one has to compromise on the size and pose of the gap for deployment when using such a modular architecture. This is primarily attributed to compounding error, to which the problem at hand is particularly sensitive, due to artificially defined inter-module interfaces and isolation optimization objects of each module (*10, 11*). For instance, state estimates, which are far from perfect with cheap sensors and computationally lightweight algorithms, serve as an artificial interface for downstream modules; a TO module plans a trajectory by either pessimistically modeling the uncertainty on (*12*) or simply trusting in (*13*) the estimates; a controller is tuned to blindly minimizes the trajectory tracking error according to the state estimates, with no awareness of the task at hand and error of state estimation. Moreover, tiny measurement errors caused by manual calibration on the landmarks (*6*) (in our problem, the gap) can propagate and be magnified across the modules. To this end, the practitioners may conduct extensive tuning for each modular and several iterations of the system for successful deployment on a very specific setup.

A competing approach is reinforcement learning (RL) that is originally created for generalized autonomy (*14, 15*), through which we can solve many problem variants or even multiple tasks in a unified optimal control formulation and objective function design. In the past years, RL has dominated the performance charts in the world of game (*8, 16–20*) and also shed its light on physical control problems (*21–23*). With deep neural networks, RL policy also inherits the power of deep learning, where end-to-end training enables gradients to propagate through modules and the interfaces between modules need no longer to be artificially defined (*24*). In this way, the policy learns to encode the low-dimensional cues for decision-making in its hidden states from rollout samples. On the other hand, this approach takes advantage of offline computation to establish data-driven memorization for cheap deployment-time inference compared to planning methods that usually struggle in the trade-off of efficiency, generalizability, and optimality during

deployment (*9*). Therefore, in recent years, we have witnessed the emergency of such ideas of designing a policy that takes pixels as input and directly generates actions in a closed loop, on the tasks of legged locomotion (*25, 26*), drone racing (*27, 28*) and dexterous in-hand manipulation (*29*). Nevertheless, unlike the problem at hand, these systems either enjoy high degrees of freedom to react to errors (*25, 26, 29*) or a relatively large feasible solution space (*27, 28*).

Here we present a real-world-ready autonomous system that can perform narrow gap traversal, with little prior knowledge of the gap when deployed, with low clearance, e.g., under 5cm, using only onboard monocular camera and the direction of gravity readily provided accurately with an inertial measurement unit (IMU). The policy running onboard directly maps pixels to control commands of collective thrust and bodyrates at a frequency aligned with that of the image stream, performs tight coordination between the actuators, i.e., the rotors, and the vision sensor−its eye.

However, the unprecedented performance comes at the cost of the challenges of developping a unified workflow to train the policies for the problem. The challenge is twofold: (i) training a policy with high-dimensional and recurrent inputs to perform well in the noisy simulation domain, which is hard due to the complexity of the inputs and a small feasible solution space of the problem at hand, and (ii) enabling the policy to reproduce the success in simulators on a physical quadrotor despite the inevitable gap between the simulation and the real world, which can easily destroy the results for such a low-fault-tolerance task if not handled carefully. The first problem is tackled with a widely used decoupling (*23, 25, 29*) of the direct, one-stage RL, into first finding a robust solution for a Markov decision process (MDP) with oracle observations using RL and then establishing a mapping from a pixel-based recurrent input to the action, known as distillation, in a partially observable Markov decision process (POMDP) using online supervised learning (*30*). The task space is divided into subsets where a single RL policy masters in each of them, respectively. However, exploration of the solution using general RL algorithms can be difficult due to the narrow feasible space, which becomes a major hindrance to effective learning. We take advantage of the model-based TO method in simulation, where privileged information such as self and target states are available, to generate dynamically feasible and (mostly) collision-free trajectories and reset the agent to the states on the trajectories to bootstrap the exploration. This approach, compared to those directly mimicking (*10*) or synthesizing (*31*) the 'demonstrations', is expected to better leverage RL's ability to explore and exploit massively generated data. During distillation, a simple

intuition is that actions are not equally important for a successful traversal when distilling, e.g., the trajectory data far from the gap may not be as important relative to the one approaching or crossing the gap. Based on this, we compute an auxiliary conservative Q-value network (*32*) for adaptive importance weighting of samples. We find that this bridges the gap of success rates between the policies obtained by distillation and RL policies.

Due to the complex aerodynamic effects and the noisy actuation mechanism caused of micro drones that may impact the quadrotor, a perfect simulation of the quadrotor's dynamics transitions is almost impossible, thus whether the pixel-based policy deployed in the real world can reproduce the success of its simulated counterpart remains unclear for such a low fault-tolerance task. Theoretically, the states of the quadrotor during real-world flight can be biased from the states in simulation training due to the sim-to-real gap. As a result, some of these states may not be well-trained in simulation. This is particularly fatal to the problem at hand, as a single sub-optimal decision when approaching a narrow gap can lead to failure of the traversal. To this end, we conduct various types of domain randomizations and random perturbations that are not typically used in sim-to-real flight skill learning. These techniques expand the well-trained state space of the policy, making trajectories nevertheless overfit the simulator, as the key to successful deployment in the real world. Perceptual latency (*33*) due to camera transmission and decision-making are calibrated and simulated, to which our policy is sensitive.

The presented system achieves traversing gaps, where a quadrotor with a dimension of **38cm×10cm** (largest length measured between propeller tips) traverses through a rectangle gap with a dimension **60cm×20cm**, in various, prior unknown poses and positions, by directly mapping onboard sensory data to high-frequency control commands. With the presented system, we report an unprecedented success rate of 100% when the roll angle of the gap is set as $\leq 60°$, and 80% for the gap titled with roll angles more than 60°. For the pitched gaps, where the gap can inevitably disappear from the field of view (FoV), the policy achieves success rate of 100% when the pitch angle of the gap is set $\leq 30°$, and 75% when the pitch of the gap is set from 45° to 70°, in the test setting.

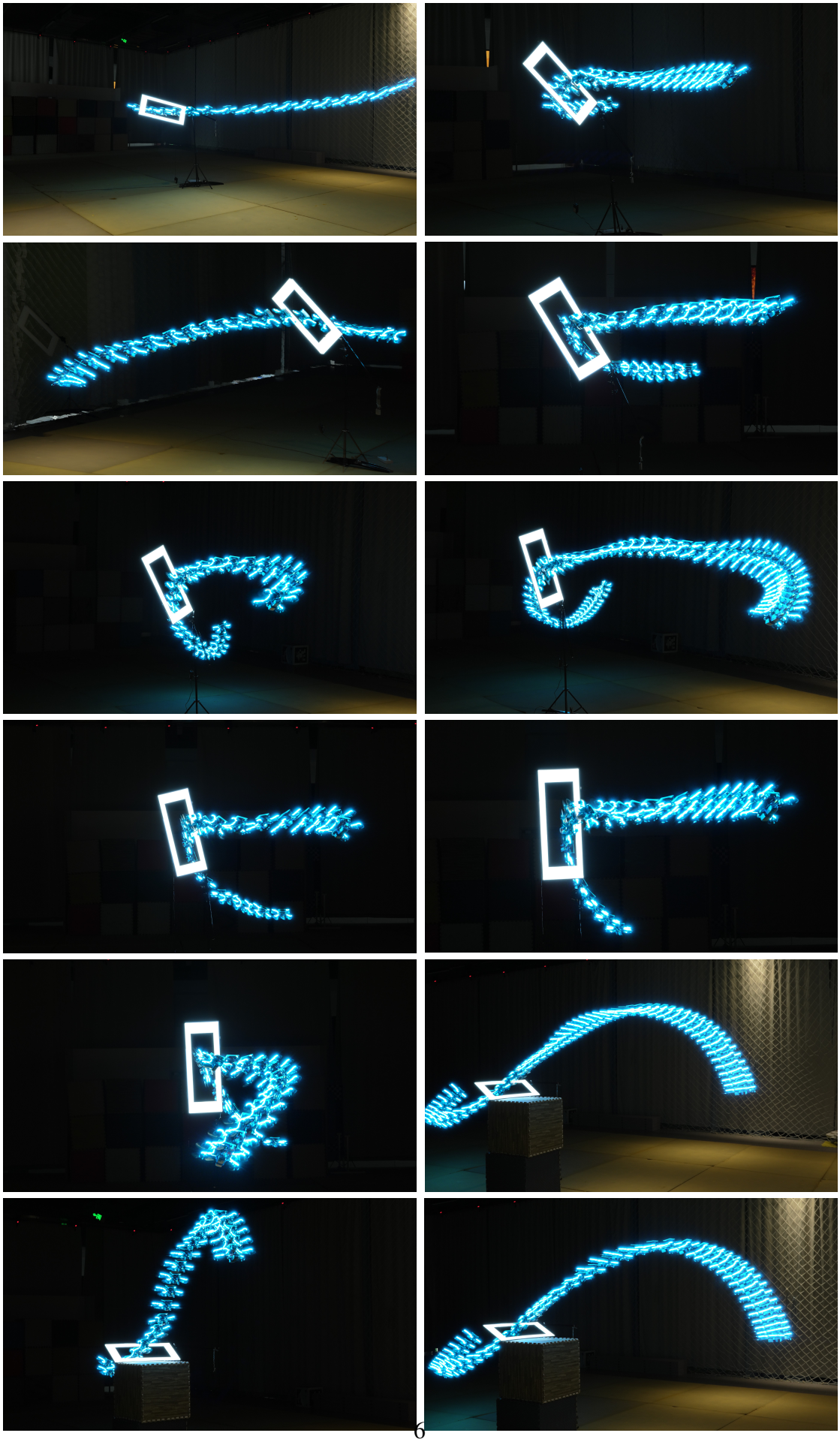

Figure 1: **Real-world rollouts of the policy with only onboard sensing.**

Results

Our main interest is whether the quadrotor can successfully traverse the narrow gap, so the main metric is the success rate of narrow gap traversal, i.e., in terms of whether the quadrotor can fly through the closed area formed by the narrow gap without collision between the vehicle and the gap instance. In other words, we are not expecting the quadrotor to achieve a specified attitude at a waypoint, so our object function (see (1)) is essentially discriminant, distinct from the work where the task is simplified as pose constraints at waypoints in the three-dimensional Special European group (SE(3)). The formulation in (*7*) shares a similar spirit with ours.

In this early release report, we present the results from a quadrotor platform with dimension of 38cm×10cm, equipped with a monocular camera with a FoV of 82°×72°. The camera is used to capture the gap instance with a resolution of 1280×1024 and the image is downsampled to 320×256 before inputting the neural network policy. The flight controller is used to execute the thrust and bodyrates commands given by the policy and provide measurements of the gravity direction with little error. Analogous to First-Person View (FPV) activities, the used gravity measurements, i.e., roll and pitch measurements, replace the pilot's information about the ground horizon and about the surrounding buildings a priori. A blind proportional differential (PD) controller recovers the quadrotor into a near hovering attitude after it traverses the gap and then an optical flow module is used to stabilize the vehicle. The computation is conducted on an 16G memory onboard jeston Orin NX.

Traversal through a Rectangular Narrow Gap with Low Clearances

We train a policy on a single rectangular gap with its passable area of dimension 20cm×60cm, where the control tolerance is only 4.5cm considering the 11cm height of the vehicle. Considering the manufacturing error of the frame and the deformation of the frame after a period of use, the frame width and the passable area of simulated images are randomized at around ±1cm.

We report the statistics results of both real-world experiments calculated under 60 trials. We distinguish between different settings to present our statistics: roll angles from (1) 0°to 20°, (2) 20°to 45°, (3) 45°to 60°, (4) 60°to 80°, and (5) 80°to 90°, and pitch angle from (6) 0°to 30°, (7) 30°to 45°and (2) 45°to 70°. The real-world rollout trajectories of the policy in the real world are

visualized in Fig. 1. The success rates and the number of trials are presented in Tab. **??**. We note that, some of these positions are at an angle more than 45° from the center axis of the gap and at a distance greater than 5m from the center of the gap. From the perspective of state estimation in a traditional modular pipeline, greater distance from the target leads to high estimation uncertainty, e.g., because of the limited resolution of the pixel inputs the algorithm's inherent ineptitude in this case, thus making the initial decision-making significantly suboptimal, which a data-driven, end-to-end approach can naturally handle.

We note that, in our real-world experiments, the performance of the policy with a roll of the gap under 60° is nearly perfect. The performance degrades to around 80%∼90% from perfect when the roll angle is larger, due to the fact that the quadrotor's capability to adjust in the direction along the short edge of the rectangular gap decreases as the roll angle becomes larger, and therefore the robustness of the policy relatively decreases. For example, when the roll angle of the gap reaches 90°, even if a small thrust is applied, the collective force on the quadrotor as it approaches the gap points to the long side of the rectangular gap above the quadrotor, and thus we observe that most of the failures come from the upper portion of the quadrotor colliding with the gap; on the contrary, when the gap is titled nearly horizontally, there is more maneuverability in the direction of its acceleration along the short side of the rectangular gap. The failures also occasionally come from poor timing of attitude adjustments that cause the quadrotor to drop altitude rapidly, thus resulting in the propellers scraping against the gap's short side, a situation that, while still capable of completing the flight, is judged as a failure.

The average success rate when the pitch angle of the gap is set ≤30° is perfect, while degrading to around 75% when the pitch angle is larger. Large pitches of the gap pose challenges for the forward FoV-based vision system. By observing the flight trajectory, i.e., Fig. 1, we can tell that the policy first drives the quadrotor forward or upward by increasing the pitch angle, while then delicately adjusting its pitch angle and thrust for a decelerated horizontal locomotion to finally dive through the gap. In this way, it becomes a trade-off between keeping observation of the gap and preparation for diving, i.e., the quadrotor may have to lose perception of the whole gap and go blind for a period of time before preparing for traversing, which can be exacerbated with large pitches of the gap. We note that the persistence of blind control can degrade the performance of policy training in simulation and, more importantly, greatly increase the gap between the performance of

the policy in simulation and the real world.

Method and Materials

Problem Formulation

We formulate the control problem at hand in this section. The vehicle of the quadrotor is modeled as a single rigid body. The *collider* $C(\cdot)$ of this rigid body, which is a function of its state $\mathbf{x}$ in special Euclidean group (SE(3)), is used to determine if any collision from the rigid body happens. The quadrotor is considered a discrete-time dynamical system with continuous states and control inputs. The goal is to find the optimal control sequence to drive the quadrotor through a plane without collision, on which a closed region with geometry $\mathcal{G}$, called gap, is the only free space on the plane. To cover the practical situation where the gap has a thickness, we assume that the plane has a fixed thickness d, which is set as 2cm in this work. Accordingly, we divided the free space into three separate regions, i.e., the gap region $\mathcal{F}^g$, the space $\mathcal{F}^0$ in which the quadrotor initially locates, and the other side w.r.t the plane, $\mathcal{F}^1$. The above spatial relations are visualized in Fig. 2.

Since the gap can be very narrow, the robot has to adjust its attitude thus utilizing the asymmetry body to traverse through the gap. Therefore, the problem to solve is known as the *whole-body control* problem of quadrotors (*7, 34*). The goal in this work is to develop policies that solve this whole-body control problem in a closed loop, which is formulated as follows:

$$
\begin{aligned}
&\exists\, T < \infty, \ \text{ subject to:} \\
&\quad \mathbf{x}_0 = \mathbf{x}_{\text{init}},\ \mathbf{x}_{k+1} = \mathbf{x}_k + f(\mathbf{x}_k, \mathbf{u}_k), \\
&\quad C\left(\mathbf{x}_k\right) \in \mathcal{F} := \mathcal{F}^g \cup \mathcal{F}^0 \cup \mathcal{F}^1, \forall k, \\
&\quad C\left(\mathbf{x}_0\right) \in \mathcal{F}^0,\ C\left(\mathbf{x}_T\right) \in \mathcal{F}^1, \\
&\quad \mathbf{u}_k \sim \pi(\cdot | o_{0:k}),
\end{aligned}
\tag{1}
$$

where $f(\cdot)$ is the system's discrete-time dynamics function, $\{\mathbf{u}_k\}$ is the control sequence along time as the decision variable, π is the policy for decision-making conditioned on historical observations $o_{0:k}$.

We assume a textureless world, i.e., without textures and references except for the physical instance of the gap, similar to the setup in (*6*). Thus, our exteroceptive observation is defined as

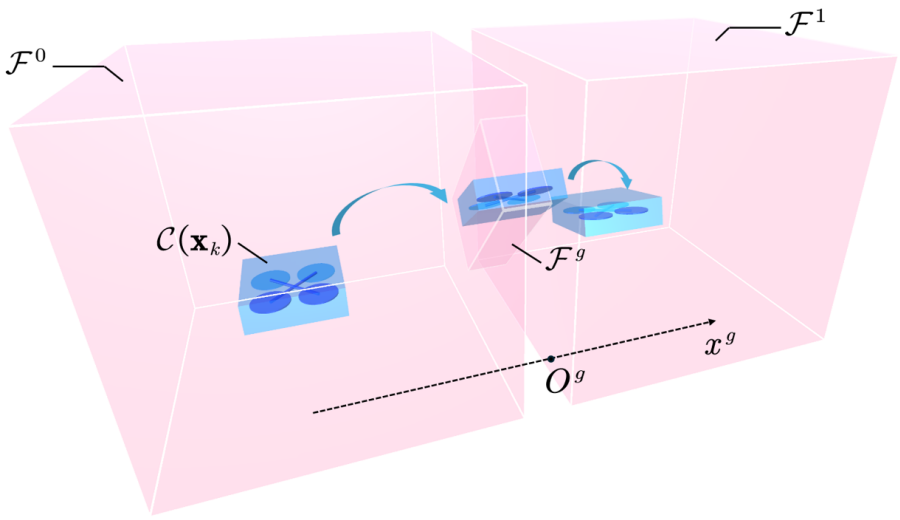

Figure 2: **Illustration of the problem of whole-body control through a gap.**

appearance of the gap instance obtained via, e.g., the object segmentation. In contrast, FPV flyers often rely on buildings or horizons to estimate and utilize the drone's roll and pitch angles, taking into account its anisotropic actuation. We simply employ the readily available information from IMU on the quadrotor to obtain such information. We see FPV-level flight that takes real-world textures and references into account as an exciting future direction.

Methodology Overview

We follow the general sim-to-real reinforcement learning paradigm where we train neural network policies in simulation and then deploy the policy to the real world in a zero-shot manner.

We decouple the exploration problem and the deep learning problem in the deep reinforcement learning method in a teacher-student paradigm (*23, 25, 29*): we first train multiple specialist oracle policies each of which masters in a subset of the task distribution using RL on a Markov Decision Process (MDP), and then train a recurrent pixel-based policy that establishes mapping from high-dimensional temporary observations to the action using interactive imitation learning (*30*) on a Partially Observable Markov Decision Process (POMDP). By regressing the action produced by the multiple oracle policies, the knowledge of the oracle policies is distilled into the recurrent pixel-based policy, as illustrated in Fig. 3.

We choose such an approach due to several reasons. A direct application of DRL can suffer from inefficiency or ineffectiveness when it tackles the problems with (i) high-dimensional or complex

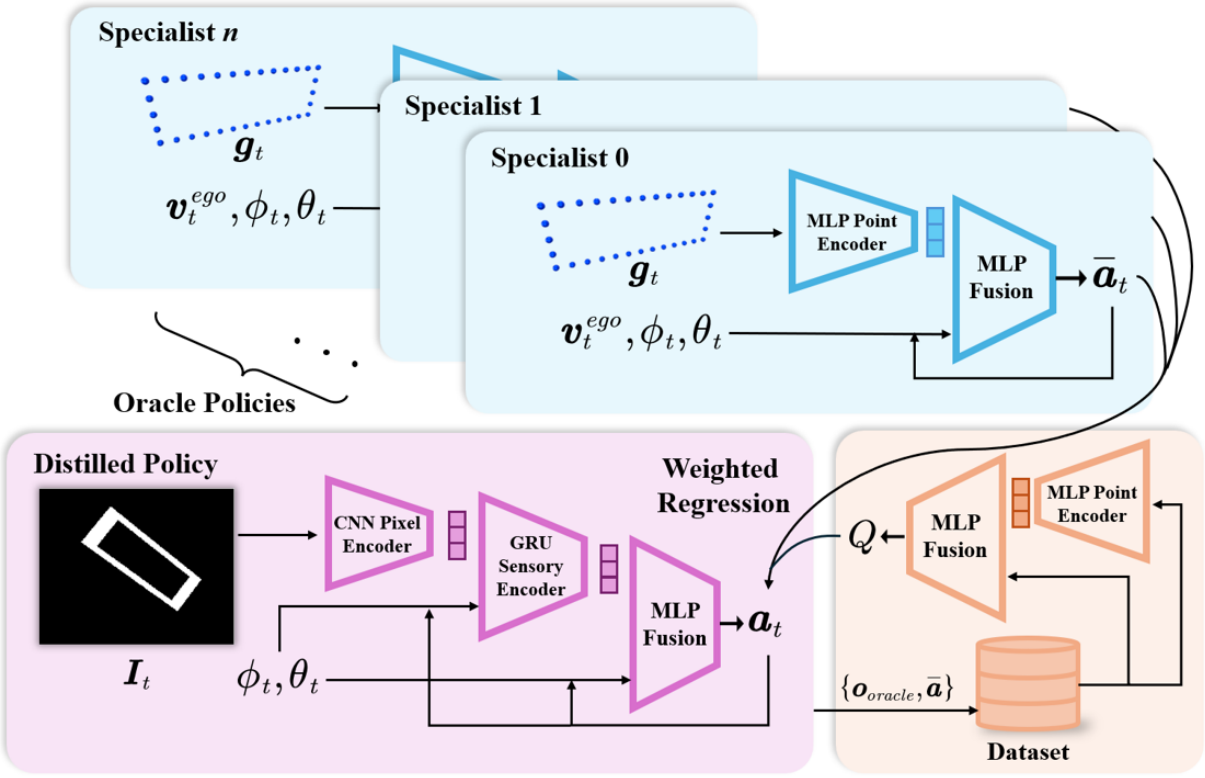

Figure 3: **Training Method Overview.**

recurrent inputs, (ii) wide task variants that require diverse responses, and (iii) a restricted solution space or sparse rewards, which makes (i) even more severe. To overcome (i), we design an oracle observation space with a low-dimensional exteroceptive surrogate of the pixel observations and construct an MDP to approximate the original POMDP. If the optimal solution of the MDP is close to that of the POMDP, this decoupled solution can approximately recover the optimal solution of the original POMDP (*25*). To overcome (ii), we train multiple RL specialists with oracle observations to master subsets of the MDP, and then distill them into a unified policy. We mitigate the issue (iii) on the surrogate MDP using an informed reset strategy and avoid inefficient explorations directly on the complex POMDP.

Online Reinforcement Learning from Gap Points

Observation and Action Space

We choose n^g (n^g is 32 in our implementation) 3D points $\boldsymbol{g}_t$ uniformly sampled along the gap edge, which is considered enough to express gaps with any shape, as shown in Fig. 3, and transform it

into the body frame to surrogate the egocentric pixel information. Roll and pitch angles ϕ_t and θ_t of the vehicle can be estimated with high precision (within 2 deg of error) from a single IMU, so we adopt them as input. We also use oracle information, the body-frame linear velocity $\mathbf{v}_t$. We also take the last taken action $\mathbf{a}_{t-1}$ as inputs. Note that we do *not* include the position as input and all the observations are ego-centric. The outputs are low-level control commands, collective thrust and body rate, to be executed by a flight controller embedded in the quadrotor.

Reward Function

RL optimizes the problem in the form of $\max_\pi \mathbb{E}_{\boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{o}_t)} \left[\sum_t \gamma^t r_t\right]$, where the immediate reward r_t should be designed to find the solution of (1), such that the policy π conditioned only on current (oracle) observation $\boldsymbol{o}_t$ can find the optimal control sequence in a closed loop.

- *traversing reward:*

$$\mathbb{I}\left[\left|x_t^g\right| \leq l^C \text{ and } \|\mathbf{p}_t - \mathbf{p}^g\| \leq d \text{ and } C\left(\mathbf{x}_k\right) \in \mathcal{F}\right] \cdot (x_t^g - x_{t-1}^g), \tag{2}$$

  where $\mathbb{I}[\cdot]$ is the indicator function, x^g is the 1-axis coordinate established parallel to the gap normal vector with the origin at the center of the gap's thickness, as illustrated in Fig. 2, $l^C > 0$ is a threshold set according to the size of the collider, $\mathbf{p}^g$ is the geometric center of the gap, and $d > 0$ is also a threshold. The indicator approximately determines if the vehicle is traversing the gap at the moment, i.e., $C\left(\mathbf{x}_k\right) \cap \mathcal{F}_g \neq \varnothing$. Optimizing this term is equivalent to find the solution of (1).

- *shaping reward:*

$$\mathbb{I}\left[\left|x_t^g\right| > l^C\right] \cdot \left(\left\|\boldsymbol{p}_{t-1} - \boldsymbol{p}^g\right\| - \left\|\boldsymbol{p}_t - \boldsymbol{p}^g\right\|\right). \tag{3}$$

  This reward, while not always describing the right motion of the quadrotor, helps to alleviate the exploration problem to some extent.

- *jerky motion penalties:*

$$-(\lambda_{\text{mag}} \cdot \|\boldsymbol{a}_t\| + \lambda_{\text{var}} \cdot \|\boldsymbol{a}_t - \boldsymbol{a}_{t-1}\|), \tag{4}$$

  where $\lambda_{\text{mag}}, \lambda_{\text{var}}$ greater than 0 are the penalty coefficients for the magnitude and variation of actions. These penalties are used to encourage a smooth motion. We find this term of reward

and tuning the corresponding weight are very crucial for the policy to perform similar smooth motions as model-based methods (*7*).

- *aggressiveness constraint:*

$$\mathbb{I}\left[\|\boldsymbol{v}_t\| \leq v_{\max}\right] \cdot \left(-\exp\left(\|\boldsymbol{v}_t\| - \boldsymbol{v}_{\max}\right) + 1\right). \tag{5}$$

  This reward is set to constrain the velocity of the motions.

- *distillation-awareness regularization:*

$$\begin{aligned} &\lambda_{\text{vis}} \cdot \mathbb{I}\left[\left|x_t^g\right| > l\right] \cdot \left|\boldsymbol{g}_t^{\text{vis}}\right| - \lambda_{\boldsymbol{v}} \cdot \mathbb{I}\left[\left|x_t^g\right| > l\right] \cdot \left\|\boldsymbol{v}_t^x - \boldsymbol{v}_t^g\right\| \\ &-\lambda_{\mathbf{n}} \cdot \mathbb{I}\left[\left|x_t^g\right| \leq l\right] \cdot \left\|\boldsymbol{v}_t^x - \mathbf{n}^g\right\|, \end{aligned} \tag{6}$$

  where $\left|\mathbf{g}_t^{\text{vis}}\right|$ is the number of visible gap points, i.e., the gap points in the limited field of view (FoV), and $\boldsymbol{v}_t^x$, $\boldsymbol{v}_t^g$ and $\mathbf{n}^g$ are the direction of body x-axis of the quadrotor (the directed centerline of FoV), the direction pointing from the body to the gap center, and the normal of the gap plane, respectively. We call the rewards above as distillation-awareness reward since it helps to *reduce the discrepancy between the gap point and pixel observations*. Specially, the first two terms encourage actions improving the visibility of the whole gap, and the last term regularizes the behavior of the quadrotor when it is approaching the gap and the gap becomes invisible.

We note the weight of the traversing reward is significantly larger than those of others.

Termination Condition

An episode terminates when one of the following happens: (i) $C\left(\mathbf{x}_t\right) \in \mathcal{F}^1$, i.e., the quadrotor has crossed the gap, (ii) the agent is out of an artificially defined world box, (iii) the quadrotor collides with the gap plane, and (iv) the episode steps achieves a certain number.

Policy Representation

The policy is represented by a neural network consisting of a simple gap point encoder and a feedforward output network. The point encoder receiving the gap points as input is a multilayer perceptron (MLP) with an intermediate global max-pooling layer to encode features that are

invariant to the order of points (*35*). The feedforward network is an MLP that fuses the point features and other observations to output actions.

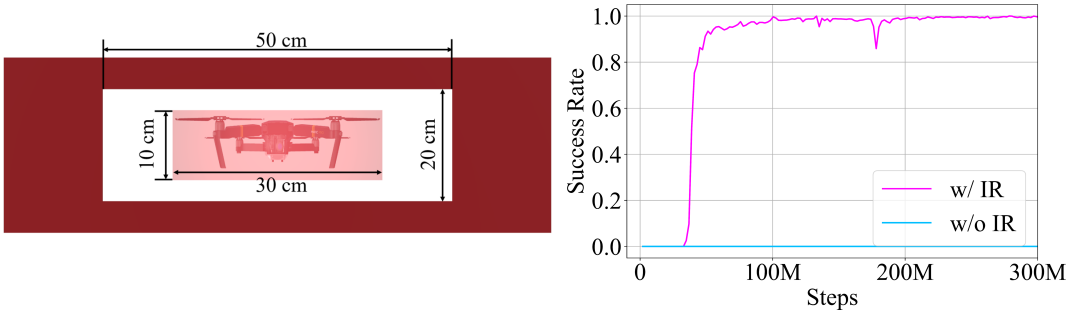

Figure 4: **A motivating example of IR in low-noise dynamics.** The left figure is a visualization of the sizes of the collider vs. the gap. The right figure is the success rate evolution.

Informed Reset

We begin the introduction of our reset scheme, called the informed reset (IR), with a motivating example in Fig. 4, which is conducted in a *low-noise* dynamics domain. In this example, the collider of the quadrotor is modeled as a cuboid and is trained to fly through a rectangular gap tilted so that the long side is *vertical* to the ground. From the results in Fig. 4, the policy is allowed for rapid mastery of this flight skill once the initial success occurs in the IR-enabled case, whereas directly training the policy in such a challenging task leads to an exploration failure.

The pipeline of IR is as follows. First, we define the task distribution at triple levels: the gap geometry, the gap pose, and the initial state of the quadrotor relative to the gap. We divide each level into subspaces, e.g., we have i kinds of gap, divide the pose into j disjoint sets, and divide the initial state into k disjoint sets, thus obtaining $i \times j \times k$ task subspaces.

Then, we follow the quotient space-based method for $\mathrm{SE}(3)$ trajectory optimization in our previous work (*7*), where the gap is represented by polytopes, to generate one trajectory for each task space. These trajectories are generated efficiently compared to other model-based methods thanks to the advanced computational techniques proposed in (*7*). In aggregate, $i \times j \times k$ trajectories are collected offline.

We propose to reset the agent in the state randomly sampled from the corresponding trajectories of the task subspace, which includes the position, attitude, velocity and acceleration. We set a 50% probability of sampling the states on these trajectories, and the rest follows a randomized sampling

of initial states. States that satisfy the condition of reward (2) are not sampled. The yaw angle of the reset state is imposed to let the quadrotor be visible to the target. This method shares a similar spirit with those in (*36, 37*), which use informed states to reset the agent. The idea behind this reset scheme is logically intuitive: the high-quality trajectories impose the exploration distribution around some crucial states and starting the episode from these states can make random exploration actions more likely to be highly rewarded. These beneficial actions are *reinforced* and more likely to be followed in the future, which gradually biases the explored distribution.

Implementation

We use a single Intel i9-14900K centralized processing unit (CPU) to simulate the dynamics transitions, and an NVIDIA RTX 4090 graphic processing unit (GPU) for update and inference of neural networks and image rendering. We use the proximal policy optimization (PPO) (*38*) algorithm that is demonstrated to be powerful with a large throughput of data. The collider of the quadrotor is modeled as a 30cm $\times$ 30cm $\times$ 11cm cuboid.

Online Distillation via Supervised Learning

As we assume a textureless and instance-only world, where the policy does not rely on the texture of the world or background visual references to make decisions (*6*), a binary mask of the gap instance is chosen as the pixel input of the recurrent policy. Beyond the problem formulation, this object-level visual modality renders less sim-to-real gap compared to the raw images (*39–41*).

The current pixel input $\mathbf{I}_t$ are encoded using a lightweight convolutional network and the obtained visual feature is concatenated with roll/pitch angles and the last action to input into a one-layer recurrent network, gated recurrent unit (GRU), to maintain a memory. This memory is necessary since a partial or whole gap can vanish from the FoV during flight, when the policy has to handle the situation according to the memory. A following feedforward MLP module fusing the features output by the GRU and the vectorized inputs at current timestamp generates the action.

Advantage-Weighted DAgger for Biased Regression

We use the dataset aggregation (DAgger) algorithm (*30*) as a meta-algorithm to train the pixel-based policy. DAgger is an online/interactive IL method that effectively suppresses the covariate shift problem of offline learning, both in theory (*30*) and practice (*42*). Typically, DAgger uses the behavior cloning loss with uniform weights for every sample. However, intuitively, not all samples are alike in importance. Therefore, we first train a pixel-based policy with uniformly weighted regression and reuse the collected data in this distillation stage to fit a conservative Q network *with the oracle observation space and the oracle policy's action as the condition* using Implicit Q-Learning (IQL) (*32*), a widely employed offline RL algorithm. Then for each sample, we compute its weight according to the advantage-weighted behavior cloning (AW-BC) formulation in (*31*) and reset the policy entropy (*43*) of the pixel-based policy for a fine-tuning stage.

Sim-to-Real Techniques for Low-Clearance Goal-Conditioned Flight

For low fault-tolerant seam traversal, specialized sim-to-real techniques need to be explored. In general, such techniques fall into two ideas (*22*), one is to reduce the trajectory error between the simulated rollout and that in the physical world by making the dynamics transitions or sensor readings from the simulator approximate those in the physical world (*4*), and the other is to hone the policy trained in simulation more robust, e.g., through domain randomization (*44*), to overcome the inherent discrepancies between simulation and the real world for a successful deployment. We introduce the following variants based on these two ideas and explain insights for them, which collaboratively contribute to the policy to finally reach an unprecedented success for precise flight in the real world:

Perturbation Force

A key to improving the success rate of the trained policy during deployment is to apply perturbations to the simulated quadrotor. Instead of representing unmodeled dynamics (*44*), our aim is to force the policy not to overfit certain features in the simulation, as well as to extend the range of states the policy supports. For example, in pixel-based policy learning, we do not expect the policy to learn to make decisions that rely exclusively on processing historical action inputs, which approximates

overfitting the dynamics transitions, while trivializing higher-fidelity pixel inputs; the unobservable perturbation force can encourage the policy to look more at the exterceptive perception. We apply unrealistic perturbation forces to maintain, e.g., tens of simulation steps, on each axis with a force of $1\mathrm{m/s}^2$ to $2\mathrm{m/s}^2$ with some probability. Importantly, we only apply forces when the quadrotor is not close to the gap where it may begin to sharply adjust its attitude for traversal, which should be generally considered in other goal-conditioned tasks.

Response Simulation for the Flight Controller

Instead of simulating the low-level dynamics of a real quadrotor, including the execution of the flight controller and the propeller force generation, we directly fit the structured parameters to compute the response of collective thrust and bodyrates at the moment from historical desired thrusts and bodyrates output by the policy. In particular, we collect real flight data, including teleoperation data, rollout data by the trained policy in a hardware-in-the-loop setup, and aggressive trajectory tracking control data. This not only makes the simulation step more efficient to compute, but also results in more accurate dynamics transitions with less system identification efforts, as well as benefiting the controllability and interpretability of the simulation errors.

Maintenance Randomization

Previous works (*45*) randomize the outputs of the policy by multiplying randomization factors on them before they are input to the dynamics model. We maintain these randomization factors for a period of time, such as tens of steps, instead of refreshing them at every simulation step. This can make it easier for the policy to encounter novel but relevant states, resulting in a larger space of states supported by the policy. Specifically, we randomize the thrust and bodyrates output by the policy, as implemented in (*45*), before applying them to calculate the current response. We also randomize rotor drag coefficients largely instead of identifying them accurately. The fitted parameters for response calculation are also randomized.

Perceptual Latency Simulation

Perceptual latency is one of the most important properties of the system for the task at hand. We calibrate the image transmission and preprocessing time, i.e., the time from when the camera starts capturing the image to when it is input to the policy, as well as the time for decision-making, i.e., neural network inference, and model the latency in the simulation.

References and Notes

1. D. Mellinger, V. Kumar.
2. D. Mellinger, N. Michael, V. Kumar .
3. X. Zhou, *et al.* .
4. E. Kaufmann, *et al.* .
5. G. Loianno, C. Brunner, G. McGrath, V. Kumar .
6. D. Falanga, E. Mueggler, M. Faessler, D. Scaramuzza.
7. Z. Wang, X. Zhou, C. Xu, F. Gao .
8. D. Silver, *et al.* .
9. B. Rivière, J. Lathrop, S.-J. Chung .
10. M. Bojarski, *et al.* (2016).
11. Y. Hu, *et al.* (2023).
12. S. Yang, G. J. Pappas, R. Mangharam, L. Lindemann (2023).
13. Y. Ren, S. Liang, F. Zhu, G. Lu, F. Zhang (2023).
14. R. S. Sutton, A. G. Barto, *et al.* (1998).
15. R. S. Sutton, D. McAllester, S. Singh, Y. Mansour (1999).
16. V. Mnih, *et al.* (2015).
17. C. Berner, *et al.* (2019).
18. J. Schrittwieser, *et al.* (2020).
19. D. Silver, *et al.* (2018).
20. P. R. Wurman, *et al.* (2022).

21. P. Abbeel, A. Coates, M. Quigley, A. Ng (2006).

22. J. Hwangbo, *et al.* (2019).

23. J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, M. Hutter (2020).

24. Y. LeCun, Y. Bengio, G. Hinton (2015).

25. A. Agarwal, A. Kumar, J. Malik, D. Pathak (2023).

26. D. Tirumala, *et al.* (2024).

27. I. Geles, L. Bauersfeld, A. Romero, J. Xing, D. Scaramuzza (2024).

28. J. Xing, A. Romero, L. Bauersfeld, D. Scaramuzza (2024).

29. T. Chen, *et al.* (2023).

30. S. Ross, G. Gordon, D. Bagnell (2011).

31. A. Kumar, *et al.* (2022).

32. I. Kostrikov, A. Nair, S. Levine (2021).

33. D. Falanga, S. Kim, D. Scaramuzza (2019).

34. T. Wu, Y. Chen, T. Chen, G. Zhao, F. Gao (2024).

35. C. R. Qi, H. Su, K. Mo, L. J. Guibas (2017).

36. T. Salimans, R. Chen (2018).

37. X. B. Peng, P. Abbeel, S. Levine, M. Van de Panne (2018).

38. J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov (2017).

39. M. Müller, A. Dosovitskiy, B. Ghanem, V. Koltun (2018).

40. T. Gervet, S. Chintala, D. Batra, J. Malik, D. S. Chaplot (2023).

41. M. Dalal, *et al.* (2024).

42. Y. Pan, *et al.* (2017).

43. T. Haarnoja, A. Zhou, P. Abbeel, S. Levine.

44. O. M. Andrychowicz, *et al.* (2020).

45. E. Kaufmann, L. Bauersfeld, D. Scaramuzza (2022).