

A Supplementary Materials

The supplementary materials consist of:

- Sec A.1: Demonstration video.
- Sec A.2: Details on the Quadrotor Dynamics for Policy Training.
- Sec A.3: Details on the Reward Formulation and hyperparameters for RL.
- Sec A.4: Details on the training configurations.
- Sec A.5: Details on the hardware configurations for the real-world experiments.
- Sec A.6: Ablation studies on the performance using different IL initialization.
- Sec A.7: Ablation studies on the improvement of our asymmetric setup.
- Sec A.8: Ablation studies on pre-trained critic function with DAgger.
- Sec A.9: Ablation studies on the history length of our approach and the baseline DAgger to handle partially observable scenarios.
- Sec A.10: Illustration of the onboard visual inputs of the policy.
- Sec A.11: Illustration of the unobservable states for the task settings.

A.1 Demonstration Video

In the video supplementary.mp4, we demonstrate the closed-loop flight of our approach relying only on visual information (with corners or images) in the real world.

A.2 Quadrotor Dynamics for Policy Training

The quadrotor is assumed to be a 6 degree-of-freedom rigid body of mass m and diagonal moment of inertia matrix $\mathbf{J} = \text{diag}(J_x, J_y, J_z)$. Furthermore, the rotational speeds of the four propellers Ω_i are modeled as a first-order system with a time constant k_{mot} where the commanded motor speeds Ω_{cmd} are the input. World \mathcal{W} and Body \mathcal{B} frames are defined with an orthonormal basis i.e. $\{\mathbf{x}_{\mathcal{W}}, \mathbf{y}_{\mathcal{W}}, \mathbf{z}_{\mathcal{W}}\}$. The frame \mathcal{B} is located at the center of mass of the quadrotor. The state space is thus 17-dimensional and its dynamics can be written as:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}}_{\mathcal{WB}} \\ \dot{\mathbf{q}}_{\mathcal{WB}} \\ \dot{\mathbf{v}}_{\mathcal{W}} \\ \dot{\boldsymbol{\omega}}_{\mathcal{B}} \\ \dot{\boldsymbol{\Omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{v}_{\mathcal{W}} \\ \mathbf{q}_{\mathcal{WB}} \cdot \begin{bmatrix} 0 & \boldsymbol{\omega}_{\mathcal{B}}/2 \end{bmatrix}^{\top} \\ \frac{1}{m} (\mathbf{q}_{\mathcal{WB}} \odot (\mathbf{f}_{\text{prop}} + \mathbf{f}_{\text{drag}})) + \mathbf{g}_{\mathcal{W}} \\ \mathbf{J}^{-1} (\boldsymbol{\tau}_{\text{prop}} - \boldsymbol{\omega}_{\mathcal{B}} \times \mathbf{J} \boldsymbol{\omega}_{\mathcal{B}}) \\ \frac{1}{k_{\text{mot}}} (\boldsymbol{\Omega}_{\text{cmd}} - \boldsymbol{\Omega}) \end{bmatrix}, \quad (3)$$

where $\mathbf{g}_{\mathcal{W}} = [0, 0, -9.81 \text{ m/s}^2]^{\top}$ denotes earth's gravity, \mathbf{f}_{prop} , $\boldsymbol{\tau}_{\text{prop}}$ are the collective force and the torque produced by the propellers, and \mathbf{f}_{drag} is a linear drag term. The quantities are calculated as follows:

$$\mathbf{f}_{\text{prop}} = \sum_i \mathbf{f}_i, \quad \boldsymbol{\tau}_{\text{prop}} = \sum_i \boldsymbol{\tau}_i + \mathbf{r}_{\text{P},i} \times \mathbf{f}_i, \quad (4)$$

$$\mathbf{f}_{\text{drag}} = -[k_{vx}v_{\mathcal{B},x} \quad k_{vy}v_{\mathcal{B},y} \quad k_{vz}v_{\mathcal{B},z}]^{\top}, \quad (5)$$

where $\mathbf{r}_{\text{P},i}$ is the location of propeller i expressed in the body frame, \mathbf{f}_i , $\boldsymbol{\tau}_i$ are the forces and torques generated by the i -th propeller, and (k_{vx}, k_{vy}, k_{vz}) [40, 41] are linear drag coefficients.

437 A.3 Reward Formulations for RL Trainings.

438 The reward components are formulated as follows:

$$\begin{aligned}
r_t^{\text{prog}} &= \lambda_1 (d_{\text{Gate}}(t-1) - d_{\text{Gate}}(t)), \\
r_t^{\text{perc}} &= \lambda_2 \exp(\lambda_3 \cdot \delta_{\text{cam}}^4), \\
r_t^{\text{act}} &= -\lambda_3 \|a_t - a_{t-1}\|, \\
r_t^{\text{br}} &= -\lambda_4 \|\omega_t\|, \\
r_t^{\text{pass}} &= c_1, \quad \text{if robot passes the next gate,} \\
r_t^{\text{crash}} &= -c_2, \quad \text{if robot crashes (gates, ground)}.
\end{aligned} \tag{6}$$

439 Here $d_{\text{Gate}}(t)$ denotes the distance from the robot’s center of mass to the center of the next gate to
440 pass, δ_{cam} is the angle between the camera’s optical axis and the direction towards the center of the
441 next gate. a represents the control command, and ω the bodyrate. $\lambda_1, \lambda_2, \lambda_3, \lambda_4, c_1, c_2$ are different
442 positive hyperparameters.

443 In our experiments, we employ identical hyperparameters for both state-based teacher training and
444 vision-based RL fine-tuning to ensure a fair comparison. These parameters are determined based
445 on iterations, shown in Table. 4 in both simulation and real-world experiments, aiming to achieve
optimal and smooth performance for the state-based policy.

Reward Name	Symbol	Value
Progress reward	λ_1	0.5
Perception-aware reward	λ_2	0.025
Command smoothness reward	λ_3	2e-4
Body rate penalty	λ_4	5e-4
Gate passing reward	c_1	10
Collision penalty	c_2	4

Table 4: Parameters for RL training.

447 A.4 Training Configurations.

448 For state-based teacher training, we employ a policy network consisting of a two-layer MLP, each
449 layer containing 256 neurons, with a final layer outputting a 4-dimensional vector using a *tanh*
450 activation function. In imitation learning, a 3-layer Temporal Convolutional Network (TCN) is
451 utilized to encode the 32 timestamps of perceptual inputs. The length of the temporal embedding is
452 128, followed by another two-layer MLP to output the control command. For imitation learning, we
453 employ a batch size of 512, and convergence typically occurs after collecting 5M data samples over
454 approximately 100 epochs. We incorporate a linear decay in the learning rate, starting at 1e-3 and
455 decreasing to 1e-5 at 50 epochs, remaining unchanged for the remainder of the training process.

456 A.5 Hardware Configurations.

457 We deploy our approach in the real world using a high-performance racing drone with a maximum
458 thrust-to-weight ratio (TWR) of 5.78. However, for our experiments, we have limited the TWR
459 to 2.7. We use a modification of the *Agilicious* platform [42] for the real-world deployment. We
460 have replaced the onboard computer with an RF receiver, which is connected directly to the flight
461 controller¹ and takes care of parsing the collective thrust and bodyrate commands from the computer.
462 Additionally, we also mount an ultra-low latency camera feed sender, which sends the live video
463 stream to the base computer. This configuration is similar to the one used by professional drone
464 racing pilots.

¹<https://www.betaflight.com>

465 A.6 Training Performance with different IL Initializations.

466 In this experiment, we test different percentages (ranging from 10% to 90%) of the DAgger policies and fine-tune them using our proposed approach. The detailed training plots are illustrated in Fig. 7. Notably, even with only 15% (1.5M) of the data samples used for policy training, we observe significant benefits (> 20 rewards) compared to training from scratch (< 0 rewards). Furthermore, as we allocate a larger portion of data samples for pre-training, our approach demonstrates the ability to enhance performance to a higher level. However, we observe that improvement plateaus and converges at the same level after surpassing 45% pre-training, as DAgger training ultimately results in similar policies for bootstrapping.

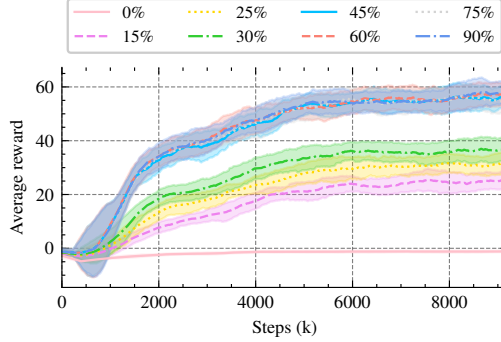


Figure 7: Rewards visualization for the SplitS track using policies from pixels with ResNet. By varying percentages (from 10% to 90%) of DAgger policies in fine-tuning using our proposed approach. Our approach reveals substantial benefits, even with 15% of data samples for policy training.

482 A.7 Ablation study on Asymmetric Critic Formulation

483 In stage III of our approach, the visuomotor policy undergoes fine-tuning using an asymmetric critic setup. In this experiment, we ablate how the critic configurations, as demonstrated in Fig. 3, can impact policy performance. As depicted in Fig. 8, RL fine-tuning with an asymmetric critic function achieves the highest reward within the same sample budget. At the same time, as shown previously, including privileged knowledge in the training process can also lead to better performance when handling partial observations.

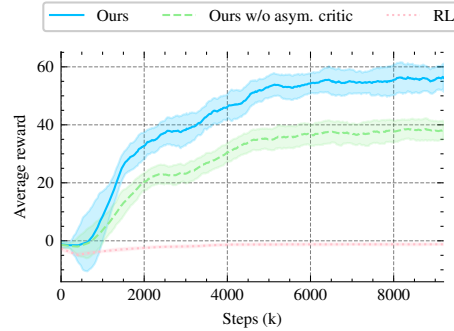


Figure 8: Comparing rewards across different RL configurations for the SplitS track using ResNet embedding, we find that utilizing an asymmetric critic makes the learning process more efficient. As a result, we have selected this configuration as the default setting for our other experiments.

494 A.8 DAgger Training for Critic Function

495 In our approach, we bootstrapped the policy training using only a pre-trained actor policy. To study the effect of how the critic function changes the performance of policy training, we conducted an experiment to compare the rewards using both pre-trained actor and critic functions in our adaptive fine-tuning setting. The pre-trained critic here is similar to actor training in the second stage, where we also learn to estimate the value using the information from the asymmetric critic used in the third stage. Since the student actor and critic are not interacting in the second stage, we could train a critic function individually, and the network training converges in the end.

505 We then fine-tune the same IL policy but use pre-trained and untrained critic functions to perform our adaptive finetuning. In Fig. 9, we visualize the results. It is clear that the differences between these two configurations are quite small. The reason behind this is that in the initial stage, the critic function will be mostly updated, as it utilizes privileged state information, making it quite sample-efficient. Hence, pre-training does not significantly boost performance, and we would need to invest more computing resources to train it. Therefore, in the end, we decided not to pursue this option.

History Length	SR%		MGE [m]		LT [s]	
	Dagger	Ours	Dagger	Ours	Dagger	Ours
4	0	0	-	-	-	-
8	28	84	0.64	0.29	8.34	7.92
16	58	97	0.52	0.27	8.31	7.83
32	100	100	0.27	0.22	8.26	7.68
64	100	100	0.28	0.21	8.27	7.65
Average	57	76	0.43	0.25	8.30	7.77

Table 5: Ablation study on history length of the policy observations using raw pixels. We could clearly find out by using more history observations, that the policy improvement will get improved. Notably, our approach consistently outperforms baseline methods across all history lengths

511 A.9 Ablation on Varying History Lengths.

512 In Table. 5 we detailed the results of our approach
513 against the best baseline methods DAgger on vary-
514 ing different history lengths. It is evident that by in-
515 corporating more historical information, the student
516 could achieve a higher success rate. More impor-
517 tantly, in all of these cases, our approach achieves
518 both better performance and success rate. The en-
519 hanced performance of our RL-based approach in
520 partially observable situations can be attributed to
521 our asymmetric actor-critic setup, where additional
522 state information is provided for value estimations
523 during environment interactions.

524 This setup significantly mitigates the challenges of
525 partial observability, thereby improving the robust-
526 ness and effectiveness of the learning process.

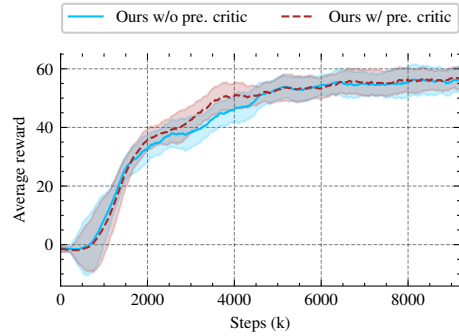


Figure 9: Reward visualization for the SplitS track using policies from pixels with ResNet w/ and w/o pre-trained critic function.

527 A.10 Onboard Image Visualization

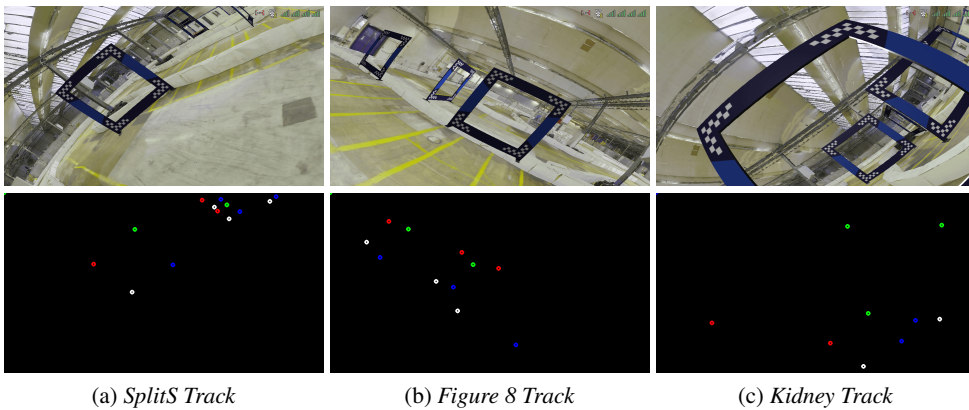


Figure 10: *Top*: Visualization demonstrates the render image input for our visuomotor policies in simulation. *Bottom*: Sparse corners visualization. Our learned visuomotor policies, relying solely on perceptual inputs, showcase their acquired capabilities for achieving robust but agile flying performance across three distinct tracks.

528 To significantly reduce the sim-to-real gap, we gather LiDAR and image data within our indoor
529 testing arena and construct a digital twin for all our experiments. In Fig 10, visualizations of the

530 images and the corners of our policies on three different racing tracks are depicted. It is noteworthy
531 that for corner generation, there is a 20% probability of missing data per corner, with ± 10 pixels of
532 noise applied. For a detailed view of real-world flights, please see the accompanying videos.

533 A.11 Unobservable States Illustration

534 For imitation learning, the policy usually needs to infer action from only partially observable states,
535 here we demonstrate one detailed example for corner-based racing in *SplitS Track* in Fig. 11. To
536 avoid the fact that the policy will need to infer actions from unobservable states, we utilize the
537 history of the observations and the asymmetric setup for RL training.



Figure 11: Illustration of one corner observation in the real world racing track. There are certain timesteps where there exist no meaningful corner projections at all. Hence we emphasize the necessity of introducing history information to handle unobservable perceptual states.