Learning to Navigate: Fully Onboard, End-to-End Navigation on Resource Constrained Quadrotor Swarms

Darren Chiu[*†], Zhehui Huang[*†], Ruohai Ge[†], and Gaurav Sukhatme[†]
[*] equal contribution, [†] University of Southern California,

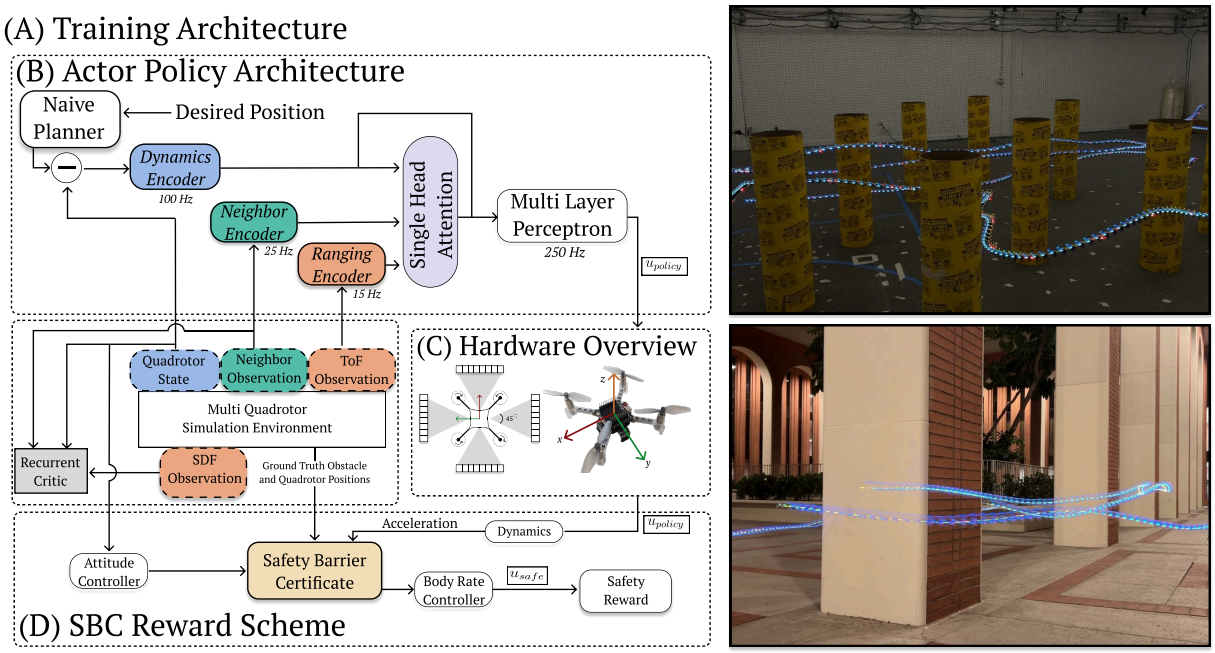

Fig. 1: Overview of the training (A) and deployment (B) architectures. During training, we leverage a larger critic model with signed distance field observations. The Safety Barrier Certificate [11] in (D) requires ground truth obstacle and quadrotor positions to find the safe control. Trajectory lapse of the outdoor and indoor deployments are shown on the right.

***Abstract*—Nano-scale unmanned aerial vehicles (UAVs) offer unprecedented agility for exploring complex environments, especially when deployed in teams. Yet their severely constrained onboard sensing, communication, and computation pose significant challenges for navigation. Most existing approaches rely on high-resolution vision or compute-intensive motion planners, rendering them infeasible for such platforms. We introduce a lightweight, safety-guided reinforcement learning (RL) framework tailored for multi-UAV navigation in cluttered spaces. Our system combines multiple low-resolution time-of-flight (ToF) depth sensors for perception with a simple motion planner and compact, attention-based deep RL policy. In simulation, our method matches the performance of two state-of-the-art baselines while using substantially fewer resources. We further demonstrate its viability on six Crazyflie quadrotors, executing fully onboard localization, perception, planning, and control.**

I. Introduction

Unmanned aerial vehicles (UAVs) are increasingly used in domains such as surveillance [9], search and rescue [6], and planetary exploration [3]. Nano UAVs offer distinct advantages over larger aerial robots. Their compact and lighter form factor allows them to traverse narrow spaces safely and scale up to larger swarms [8]. Additionally, their minimal computational footprint makes them viable candidates for extraterrestrial missions, where radiation-resistant computing architectures are critical [1]. However, these benefits come with severe constraints on sensing, processing power, and communication, making navigation and control significantly more challenging than their larger counterparts. State-of-the-art UAV navigation approaches predominantly rely on high-dimensional vision-based inputs

TABLE I: Performance comparison across methods, scenarios, and planner versions. All rates are fractions over $N = 50$ trials with 8 agents; collision rates, distances, speeds are $\mu \pm \sigma$. A-A coll. denotes agent to agent collision fractions, whereas agent to obstacle fractions are denoted by A-O coll.

Method	Scenario	Agent-level ↑	Overall*↑	Incomplete*↓	A–A coll. ↓	A–O coll. ↓	Avg. Dist. ↓	Avg. Vel. ↑
Our Method	*Straight Line*	**0.975**	**0.880**	0.000 ± 0.000	0.080 ± 0.0685	0.120 ± 0.0584	12.448 ± 0.226	0.495 ± 0.007
	Swap Goal	0.8647	**0.510**	0.206 ± 0.408	0.089 ± 0.135	0.082 ± 0.115	13.689 ± 0.712	0.443 ± 0.021
ES2 [12]	*Straight Line*	0.975	0.760	0.2 ± 0.405	0.020 ± 0.035	0.120 ± 0.075	13.689 ± 1.330	0.924 ± 0.162
	Swap Goal	**0.896**	0.456	0.280 ± 0.453	0.042 ± 0.098	0.074 ± 0.112	13.022 ± 1.442	1.20 ± 0.08
HDSM [10]	*Straight Line*	0.765	0.3	0.16 ± 0.370	0.215 ± 0.195	0.027 ± 0.068	13.022 ± 0.196	0.414 ± 0.011
	Swap Goal	0.721	0.327	0.03 ± 0.184	0.271 ± 0.249	0.002 ± 0.016	14.921 ± 1.062	0.127 ± 0.013

* Denotes that the statistic was calculated across runs, whereas others were calculated across agents.

from depth cameras [12] or LiDAR sensors [10]. Furthermore, these methods often assume access to ample computational resources, memory for map-building, and high-bandwidth communication for trajectory sharing. Specifically, many rely on parallelized computations operating at gigahertz clock speeds, enabling iterative or sampling-based motion planning or sufficient memory to build maps. However, such assumptions do not hold for resource-constrained systems. This work aims to address the challenge of deploying collision avoidance and navigation methods in resource-constrained robotic systems.

Our main contributions are as follows: i) We propose an end-to-end, learning-based autonomy framework for nano-UAV swarms that reliably tracks arbitrary trajectories in densely cluttered environments while avoiding collisions with both static obstacles and neighboring agents. ii) Building on Huang et al. [4], we fuse a global planner with a safety-based local controller to substantially improve robustness and overall system performance. iii) We benchmark against two state-of-the-art methods, HDSM [10] and Ego-Swarm2 [12] and achieve comparable performance but with dramatically lower onboard compute and sensing requirements. We demonstrate our method in both indoor and outdoor environments, running entirely on the severely resource-constrained Crazyflie platform for perception, localization, planning, and control.

II. Method

A. Problem Formulation

We consider a 3D environment with N homogeneous robots and an unknown number of obstacles with arbitrary positions and dimensions. Each robot lacks prior knowledge of the environment and relies solely on onboard resources. Robots face strict hardware constraints: (i) single-core processors at sub-gigahertz speeds, (ii) memory $\approx$ 1MB, and (iii) communication bandwidth of the order of Mbps. The goal is to develop an efficient and robust control method that enables all N robots to reach designated goals within a time limit while avoiding collisions with obstacles and neighboring robots.

B. System Design

The system design consists of three parts: 1) simulation design; 2) hardware design; 3) algorithm design.

1) Simulation Design: The simulation runs in an $8 \times 8 \times 5\ m^3$ room with a central $6 \times 4 \times 5\ m^3$ obstacle region. Robots spawn on one side and must traverse to the opposite side, either all toward a shared goal or each to an individual goal, following smooth point-to-point trajectories generated by a naive planner that ignores collision avoidance.

2) Hardware Design: Given the hardware's computational and memory constraints, we adopt the approach of [2], using a flow deck and IMU for localization, and four VL53L5CX 8×8 ToF matrix sensors for obstacle detection (Figure 1(C)).

3) Algorithm Design: We train a decentralized end-to-end RL-based control policy using asynchronous independent proximal policy optimization [7]. At each time t, robot i observes $o_i^t = (e_i^t,\ \eta_i^t,\ \zeta_i^t)$, where e_i^t is the robot's own state, η_i^t lists up to K neighbor relative positions and velocities, and ζ_i^t is a 32-dimensional vector derived from four ToF sensors. Actions $a_i^t \in [0, 1]^4$ are normalized rotor thrusts. The reward incorporates a collision penalty through a safety barrier certificate (SBC) module [11]. To satisfy onboard constraints, the actor uses a single-layer encoder with single-head attention, while the critic employs two layers with multi-head attention. More details are shown in Figure 1.

III. Experiments

a) Simulation Experiments: We conduct simulated trials comparing our learned method against two multi robot planners: Ego-Swarm2 (ES2) [12] and HDSM [10]. All three methods are tasked with navigating a straight line lateral (from -x to +x) scenario and a variation where the goals are swapped across the y axis. Table I provides a detailed breakdown of metrics. We present two definitions of success rate: agent-level and overall. We define agent level as the fraction of agents

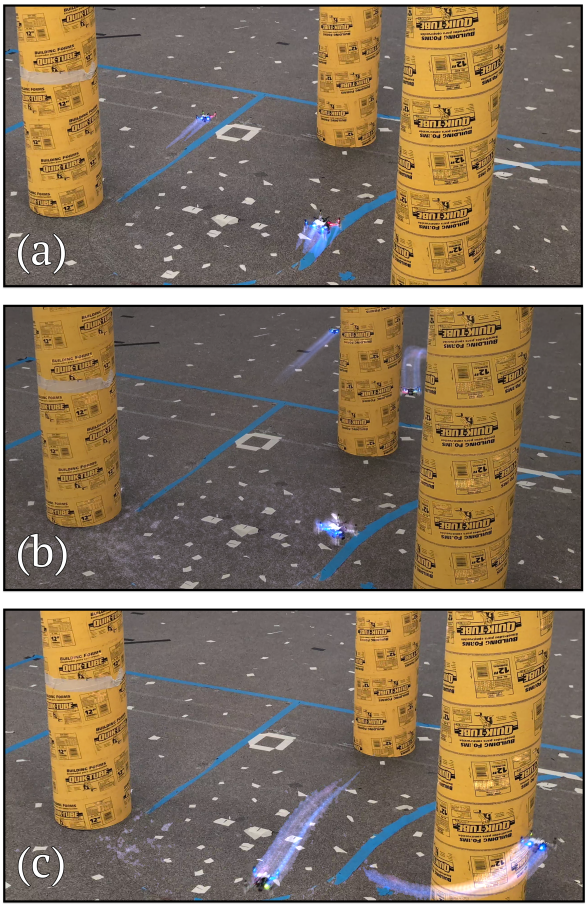

Fig. 2: High conflict negotiation on hardware. The quadrotors are faced with intersecting trajectories with the presence of obstacles. The right outgoing quadrotor waits for the incoming to pass before swerving to the right to dodge.

who arrived within distance δ of the goal collision free. The overall success rate is defined as the fraction of entire runs where all agents arrived within distance δ of the goal collision free. In both cases, we use a threshold distance of $\delta = 0.25m$. The incomplete rate is found by taking the fraction of runs where *all* agents arrive within δ of the goal and were collision free. The remaining statistics (agent collisions, distance, velocity) were all found as a fraction over the number of agents. In the *Straight Line* scenario, our method achieves both the highest overall success rate (0.88) and the highest agent-level success rate (0.975) while completing the task fully. While ES2 performs similarly in terms of the agent-level success rate, the planner fails to complete 20% of runs i.e. 20% contained at least one agent not reaching the goal or colliding. In comparison to HDSM, our method reported a higher agent to obstacle collision rate, but was able to trade-off showcasing a much lower agent to agent collision rate. The overall distance traveled for

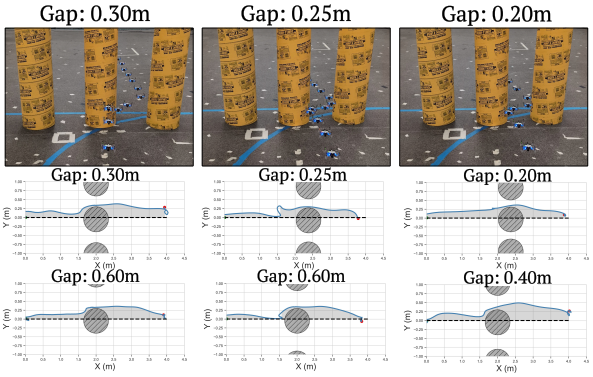

Fig. 3: The learned controller is tasked with flying through decreasing gap sizes. During training, we limit the environment generation to only produce gap sizes as small as $0.15m$. The real world deployment shows reliable flight down to gaps as small as $0.2m$.

our method is also the lowest against the two planners – supporting the previous qualitative claim on using the minimum distance planner.

The *Swap Goal* scenario was conducted to simultaneously stress the obstacle and agent navigation abilities. ES2 achieves the highest agent-level success rate (0.896), likely due to their trajectory sharing schemes. Despite the amount of information exchanged between agents in ES2, our policy only falls behind by 3%. In terms of overall success rates, the policy outperforms ES2 by 5%, suggesting that while our method relies on much more limited information it still yields effective multi agent navigation. Most importantly, we do not rely on building a voxelized representation of the environment, such as in both planning based methods. Due to the complexities, the HDSM planner suffers the most in the *Swap Goal* scenario. The results show high agent to agent collision rates (0.27) and the lowest success rates (0.327). We believe this is due to the way the synchronous planning is implemented – which especially suffers in tighter corridor navigation. When examining flight speeds, it is important to note that our policy travels significantly slower than ES2. This occurs because our training environment randomly plans trajectories using a desired velocity sampled from a normal distribution, $\mathcal{N} \sim (\mu = 0.5, \sigma = 0.1)$. Thus, to remain well within the distribution, the planned trajectories used in benchmarks were fixed at $v_{desired} = 0.5$. We chose a normal distribution centered at $v = 0.5$ to accommodate for the limits of the hardware platform, but to accommodate for sim to real variations. To fly faster, one can train the policy on higher velocity trajectories as done in [5] to achieve higher flight speeds with a suitable platform.

b) Hardware Experiments: We evaluate our policy in the real world by tasking a single quadrotor to navigate

through progressively narrower gap sizes. In Figure 3 we show the trajectories of a single quadrotor flying through gaps ranging from sizes of $0.6m$ to $0.2m$, whereas the planned trajectories are shown in the dotted. The policy consistently maintains a close yet safe distance to the seen obstacles, allowing it to tightly navigate corridors. This demonstrates that the trajectory tracking reward and obstacle collision penalty work well together for this scenario. As such, our experiments indicate that we are able to reliably fly through a $0.2m$ gap, roughly 2 times the quadrotor diameter. Figure 2 shows a challenging real-world deployment scenario involving three quadrotors navigating through a triangular arrangement of obstacles. The quadrotors are initially positioned on intersecting paths, one approaching from the incoming direction, and two departing along outgoing trajectories. The incoming quadrotor must select a maneuver that allows it to bypass the central obstacle without obstructing the outbound paths of the two others. As it begins its approach, the right-side outgoing quadrotor adjusts its trajectory preemptively, creating space to ensure safe passage. This interaction unfolds without centralized communication; instead, each agent operates using only instantaneous state exchanges—namely, positions and velocities—to adjust control actions. This scenario highlights our approach's capability for agile, autonomous conflict resolution in tight, spatially constrained conditions.

IV. Conclusion

Our experiments demonstrate, for the first time, that teams of Crazyflie quadrotors—despite their severe onboard sensing, communication, and compute constraints—can navigate collaboratively using only onboard resources. By integrating a control-barrier-function–based reward for collision avoidance, we significantly improve safety without sacrificing maneuverability. Comparative evaluations show that our approach matches the performance of state-of-the-art navigation methods while dramatically reducing sensor, communication, and computational requirements.

References

[1] Jean-Pierre de la Croix, Federico Rossi, Roland Brockers, Dustin Aguilar, Keenan Albee, Elizabeth Boroson, Abhishek Cauligi, Jeff Delaune, Robert Hewitt, Dima Kogan, et al. Multi-agent autonomy for space exploration on the cadre lunar technology demonstration. In *2024 IEEE Aerospace Conference*, pages 1–14. IEEE, 2024.

[2] Carl Friess, Vlad Niculescu, Tommaso Polonelli, Michele Magno, and Luca Benini. Fully onboard slam for distributed mapping with a swarm of nano-drones. *IEEE Internet of Things Journal*, 2024.

[3] Mostafa Hassanalian, D Rice, and A Abdelkefi. Evolution of space drones for planetary exploration: A review. *Progress in Aerospace Sciences*, 97:61–105, 2018.

[4] Zhehui Huang, Zhaojing Yang, Rahul Krupani, Baskın Şenbaşlar, Sumeet Batra, and Gaurav S Sukhatme. Collision avoidance and navigation for a quadrotor swarm using end-to-end deep reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 300–306. IEEE, 2024.

[5] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-level drone racing using deep reinforcement learning. *Nature*, 620 (7976):982–987, 2023.

[6] Balmukund Mishra, Deepak Garg, Pratik Narang, and Vipul Mishra. Drone-surveillance for search and rescue in natural disaster. *Computer Communications*, 156:1–10, 2020.

[7] Aleksei Petrenko, Zhehui Huang, Tushar Kumar, Gaurav Sukhatme, and Vladlen Koltun. Sample factory: Egocentric 3d control from pixels at 100000 fps with asynchronous reinforcement learning. In *International Conference on Machine Learning*, pages 7652–7662. PMLR, 2020.

[8] James A. Preiss, Wolfgang Honig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. page 3299–3304. IEEE Press, 2017. doi: 10.1109/ICRA.2017.7989376. URL https://doi.org/10.1109/ICRA.2017.7989376.

[9] Hazim Shakhatreh, Ahmad H Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7:48572–48634, 2019.

[10] Charbel Toumieh and Dario Floreano. High-speed motion planning for aerial swarms in unknown and cluttered environments. *IEEE Transactions on Robotics*, 40:3642–3656, 2024. doi: 10.1109/TRO.2024.3429193.

[11] Li Wang, Aaron D Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. *IEEE Transactions on Robotics*, 33 (3):661–674, 2017.

[12] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, and Fei Gao. Swarm of micro flying robots in the wild. *Science Robotics*, 7(66):eabm5954, 2022. doi: 10.1126/scirobotics.abm5954. URL https://www.science.org/doi/abs/10.1126/scirobotics.abm5954.