18.337 FINAL PROJECT: PERCEPTION-AWARE MULTIAGENT TRAJECTORY PLANNER USING IMITATION LEARNING

KOTA KONDO*

Abstract. Trajectory planning for unmanned aerial vehicles (UAVs) has been a focus of extensive research; however, heavy computational requirements hinder their deployment in real-world scenarios. One solution to this challenge is to use imitation learning planners that learn optimal trajectories from existing planners and mimic their behavior. This approach offers the advantage of low onboard computational requirements, making it more practical for real-world applications. While single-agent trajectory planning has been extensively studied, multiagent planners have recently gained popularity due to their broad range of applications, including package delivery. Multiagent planners can be either centralized or decentralized, with the latter being more scalable and robust to single-point failures.

Moreover, multiagent planners can be categorized as either asynchronous or synchronous, with the former being more scalable than the latter. Perception-aware trajectory planning has become increasingly popular among researchers due to its ability to gather information about the surrounding environment and use it to plan trajectories. This approach is particularly useful for agents flying in unfamiliar spaces. Although there have been numerous studies on perception-aware trajectory planning for single agents, its use in multi-agent systems is still relatively uncommon.

To facilitate the training of perception-aware multiagent trajectory planners, we implemented Message Passing Interface (MPI) on Julia, which is a standardized and portable message-passing standard designed for parallel computing architectures. We conducted a performance comparison that demonstrated MPI's advantage in parallelization.

Finally, we compared our imitation learning-based approach to optimization-based approaches and found that our imitation learning approach had not been previously applied to decentralized, asynchronous, perception-aware multiagent trajectory planners.

Key words: Julia [2], MPI, Imitation Learning, UAVs, Multiagent
Codes: https://github.com/kotakondo/18337

1. Introduction.

In recent years, multiagent UAV trajectory planning has been extensively studied [1, 4, 5, 7, 9, 12, 13, 15, 17, 18, 22, 25, 28, 31]. In real-world deployments of multiagent trajectory planning methods, it is crucial to deal with challenges such as (1) detecting and avoiding collisions with **unknown obstacles**, (2) handling **localization errors/uncertainties**, (3) achieving **scalability** to a large number of agents, and (4) enabling **fast and efficient computation** for onboard replanning and quick adaptation to dynamic environments. However, finding effective solutions to these challenges remains an open question.

One approach to address challenges such as detecting and avoiding unknown obstacles, even in the presence of localization errors and uncertainties, is to equip each agent with a sensor, typically a camera, to perceive the surrounding environment. This allows agents to gather real-time information about their surroundings, enabling them to make informed decisions and take appropriate actions to avoid collisions and navigate through dynamic environments. However, this sensor often has a limited field of view (FOV), making the orientation of the UAV crucial when planning trajectories through unknown space. Therefore, planners for flying with limited FOV sensors generally need to be perception-aware to ensure that as many obstacles or other UAVs as possible are kept within the FOV.

When scaling multiagent trajectory planners, it is important to note that, with centralized planners, each agent needs to listen to a single entity that plans all the

*Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA (kkondo@mit.edu).

trajectories [12, 15]. While this approach simplifies planning, the central entity may act as a single point of failure, and the replanning abilities of the agent depend on their ability to communicate with the central entity. Decentralized planners greatly mitigate these issues, as each agent plans its own trajectory [1, 9, 18, 22, 25, 31]. Decentralized planners are therefore generally considered to be inherently more scalable and robust to failures.

Similarly, synchronous planners such as [4, 18, 27] require all agents to wait at a synchronization barrier until planning can be globally triggered, whereas asynchronous planning enables each agent to independently trigger the planning step without considering the planning status of other agents. Asynchronous methods are typically more scalable compared to synchronous methods [9, 22, 31]. Table 1 shows the categorization of scalability of these multiagent trajectory planning approaches.

Table 1. Multiagent Trajectory Planner Category

	Synchronous	**Asynchronous**
Centralized	Not Scalable	Not Possible
Decentralized	Somewhat Scalable	**Most Scalable** (our approach)

Many optimization-based approaches [9, 22, 25, 31] have been proposed for multiagent trajectory generation. However, these approaches often require substantial computational resources, posing challenges for deployments in dynamic environments that demand fast on-the-fly replanning. To mitigate this issue, researchers have explored imitation learning (IL)-based approaches [11, 20, 24], which offer the advantage of faster replanning while still achieving close-to-optimal trajectory generation.

To tackle the challenges of (1) **unknown objects detection and collision avoidance**, (2) **localization errors/uncertainties**, (3) **scalability**, and (4) **fast and efficient computation**, we propose an IL-based decentralized, asynchronous, perception-aware multiagent trajectory planner. Table 2 provides a comparison of the proposed approach with state-of-the-art approaches.

2. Trajectory Generation.

2.1. Expert —Optimization-based PA MA Planning. MADER [22] proposed an optimization-check-recheck scheme for decentralized, asynchronous multiagent planning. In this approach, an agent optimizes its trajectory while using received trajectories as optimization constraints. Next, the agent checks its trajectory against trajectories received in the optimization step and rechecks if it received any trajectory in the check step. To enhance robustness against communication delays, we proposed Robust MADER [9], which replaces the recheck step with a delay-check step. These frameworks allow fully decentralized asynchronous multiagent trajectory generation under real-world uncertainties and delays.

PANTHER [23] proposed a perception-aware trajectory planner for a single agent in dynamic environments, generating trajectories to avoid obstacles while keeping them in the sensor FOV. In [24], PANTHER* improved the original PANTHER with less conservatism and more optimal trajectory generation, but both were limited to tracking and avoiding only one obstacle at a time. To overcome this limitation, we

Table 2. State-of-the-art UAV Trajectory Planners

Method	Multiagent	Perception-aware
EGO-Swarm [31]		
DMPC [10]		
MADER [22]	**Yes**	**No**
decMPC [26]		
RMADER [9]		
Raptor [30]		
Time-opt [19]		
PANTHER [23]	**No**	**Yes**
PA-RHP [29]		
Deep-PANTHER [24]		
Proposed approach	**Yes**	**Yes**

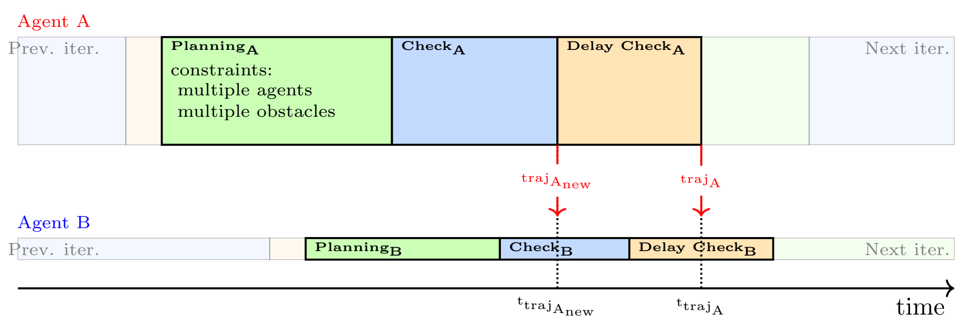


Fig. 1. Proposed trajectory optimization and deconfliction sequence: Our approach uses an imitation learning-based approach to generate trajectories for each agent, followed by a conflict detection and resolution step based on the Robust MADER framework. Each agent first generates a new trajectory in the planning step and then checks if there are any conflicts with the trajectories received from other agents. If no conflicts are detected, the agent publishes its new trajectory and begins checking for potential collisions in a delay check step. This delay check step is a sequence of checks over a period of time. Finally, if no conflicts are detected during the delay check, the agent commits to the new trajectory and publishes it. However, if conflicts are detected, the agent reverts to the trajectory from the previous iteration and discards the new trajectory. More details on the Robust MADER approach can be found in Section II of [9].

modified the optimization problem solved by PANTHER (see Appendix A) to enable tracking and avoidance of multiple obstacles, leading to a decentralized, asynchronous, perception-aware multi-agent trajectory planning system that incorporates this modified optimization approach into the RMADER deconfliction framework. Fig. 1 illustrates our approach's trajectory deconfliction scheme, which is employed by both the expert and the student.

2.2. Student —IL-based Approach. Deep-PANTHER [24] used IL to train a neural network that generates a desired position trajectory, while relying on closed-form solutions to obtain the direction where the onboard sensor should be looking (e.g., yaw on a multirotor). This closed-form yaw solution generates yaw trajectories given position trajectories, reducing the output dimension of the learned policy. However, this approach is not scalable in multi-obstacle environments since the closed-form solution only generates yaw trajectories for a single given obstacle. To address this limitation, we designed our IL-based method using a multi-layer perceptron (MLP) that generates both position and yaw trajectories. To achieve this, we increased the size of the neural network to 4 fully connected layers, each with 1024 neurons, and trained it to imitate the optimal perception-aware trajectories.

Additionally, we added a Long Short-Term Memory (LSTM) [6] feature-extraction network to the MLP, inspired by the ground-robot motion planning approach [3]. This allowed the neural network to accept various numbers of obstacles and agents as input, whereas traditional feedforward neural networks can only handle a fixed number of obstacles. LSTM can take as many obstacles and agents as possible and generate a fixed length of the latent output, which we feed into the fully connected layers.

It is also worth noting that IL-based approaches are more scalable in practice than optimization-based approaches. As the number of agents and obstacles in the environment increases, optimization-based approaches need to include more constraints in the optimization, leading to significant computational requirements. On the other hand, IL-based approaches are able to handle larger-scale environments with little to no additional computational overhead with the use of LSTM.

In summary, we first fed the predicted trajectories of obstacles and received other agents' trajectories to the LSTM, which outputs a fixed-size vector h. We then combine h with the agent's own state and feed this into the fully connected layers. The architecture of the neural network is summarized in Fig. 2.

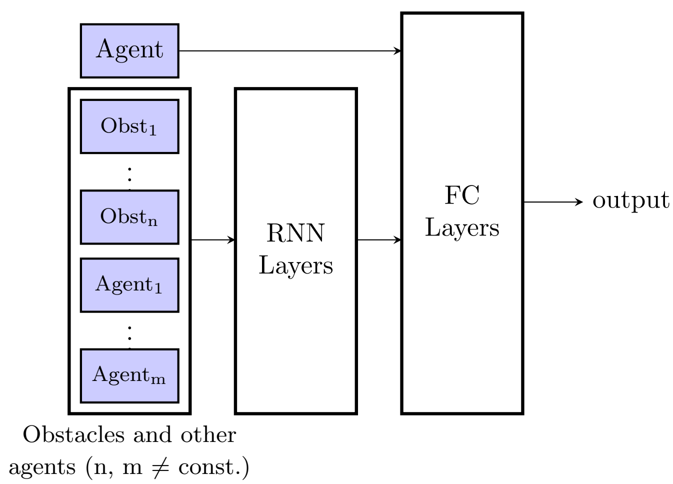


Fig. 2. Student Network Architecture

Table 3 shows the comparison of the state-of-the-art perception-aware trajectory planners. Our Expert approach is the first perception-aware multiagent trajectory planner that generates position and yaw coupled trajectory while tracking multiple obstacles, and our Student achieves much faster computation time, leveraging an IL-based planner.

[1]Deep-PANTHER [24] generates only position trajectory, and yaw trajectory is generated by closed-form solution based on the position trajectory.

Table 3. State-of-the-art Perception-aware Obstacle Tracking Trajectory Planners

Method	Tracking Multi-obstacles	Multi-agents	Trajectory	Planning
[21]	No	No	Only Position	Optimization-based (slow & not scalable)
[14]	No	No	Position & Yaw	Optimization-based (slow & not scalable)
PANTHER / PANTHER* [23, 24]	No	No	Position & Yaw	Optimization-based (slow & not scalable)
Deep-PANTHER [24]	No	No	Only Position [1]	IL-based (faster & scalable)
Expert	Yes	Yes	Position & Yaw	Optimization-based (slow & not scalable)
Student (proposed)	Yes	Yes	Position & Yaw	IL-based (faster & scalable)

2.3. Obstacle Sharing. As shown in Fig. 3, each agent detects and tracks obstacles and shares their predicted trajectories with other agents. This obstacle-sharing architecture allows the agents to have a better understanding of the surrounding environment as a team.

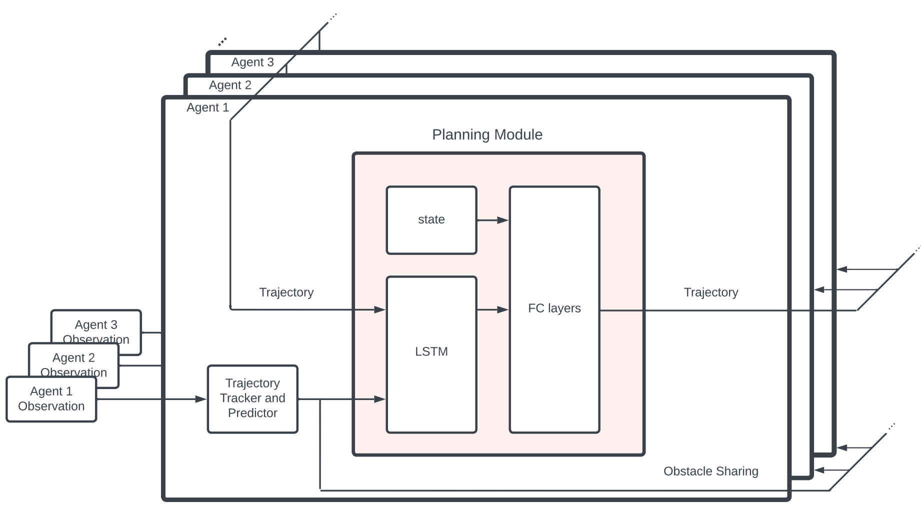


Fig. 3. Student Planning and Sharing Trajectory Architecture

3. Parallel Training.

3.1. Training Setup. We used the student-expert IL learning framework, where our expert approach provides demonstrations, and the student is trained so that its neural network can reproduce the provided demonstrations. The student was trained in an environment containing multiple dynamic obstacles following a random-

Table 4. Data Collection Time for 100 trajectories

		Data Collection Time [s]
Not Parallelized		320.4
MPI Parallelized	2 processors	181.6
	5 processors	86.8

ized trefoil-knot trajectory, with a randomized terminal goal. We first used Behavior Cloning (BC) to collect data and train the student, but its performance was sub-optimal compared to the expert. Therefore, we employed the Dataset-Aggregation algorithm (DAgger) [16] to refine the policy training using BC. The performance comparison is given by Table 5, and Section 4 provides the detailed analysis. We used Adam [8] as an optimizer, and we normalized our observation and trajectory to make it easy for the neural network to learn. Additionally, we introduced a weighted loss function between position and yaw. During the training process, we found that it was more difficult to train the yaw trajectory than the position trajectory, and thus we weighted the yaw loss function. In our training, we set the weight α to 70. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pos}} + \alpha \mathcal{L}_{\text{yaw}} \tag{3.1}$$

3.2. Julia MPI performance comparison. To accelerate the training process of perception-aware multiagent trajectory planners, we utilized the Julia MPI Package to implement parallelized training, which enables us to employ parallelized decentralized training, as depicted in Fig. 4. We performed a performance comparison and evaluated the training time using 1 (non-parallelized), 2, and 5 processors. Table 4 demonstrates that decentralized training completes much faster. Specifically, the two-processor and five-processor training completes 1.75 and 3.68 times faster than the non-parallelized training, respectively.

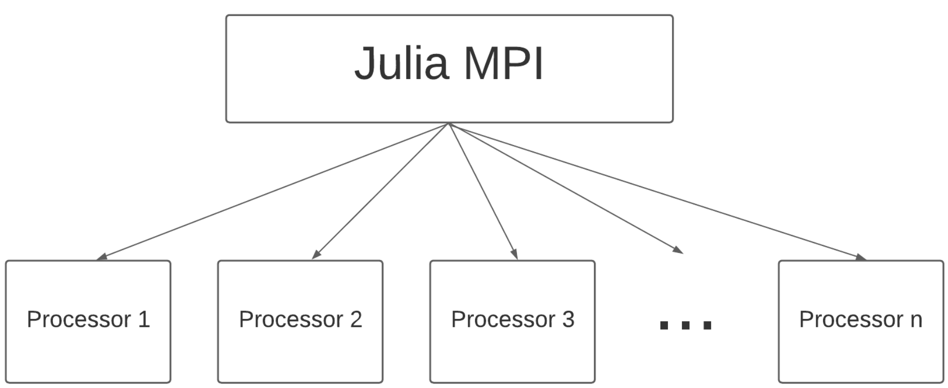


Fig. 4. Julia [2] MPI architecture

4. Simulation Results.

4.1. Expert vs. Student in single-agent, single-obstacle environment. Table 5 compares the average performance of the expert and the student in a simulation environment with a single dynamic obstacle that follows a trefoil trajectory while the agent flies diagonally to avoid obstacles. The comparison is based on the average cost of trajectories and computation time, and the student with BC and DAgger achieves about 6000 times faster computation time with little performance loss. Fig. 6 shows the simulation environment.

Table 5. Expert vs. Student

	Avg. Cost	**Computation Time [ms]**
Expert	1317.0	**5363.4**
Student (BC)	2055.4	**0.5634**
Student (BC + DAgger)	1550.3	**0.8978**

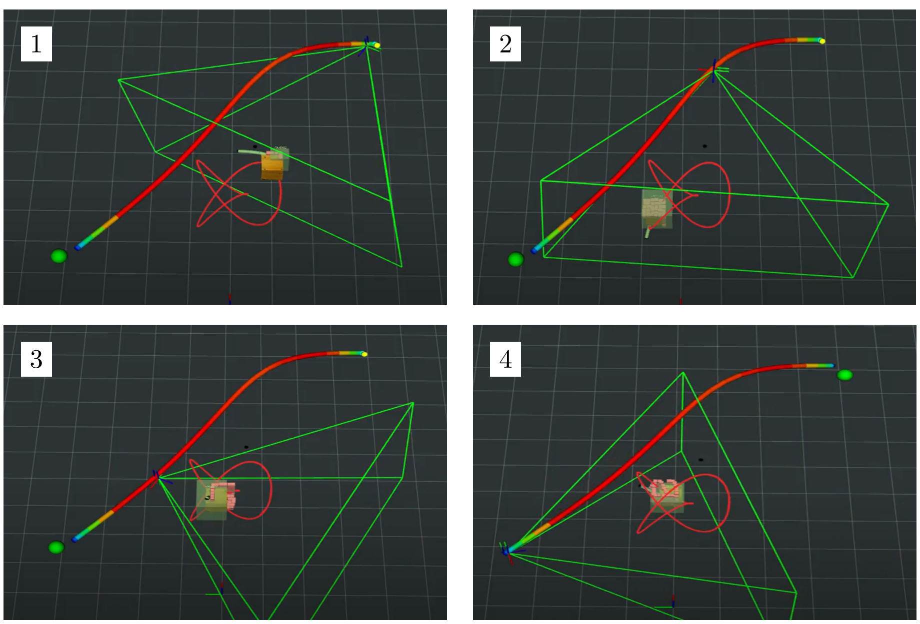


Fig. 5. Student single-agent, single-obstacle, simulation result: We made the imitation learning-based planner (student) fly around a trefoil-trajectory dynamic obstacle. The agent started at the top-right corner and was commanded to fly to the down-left.

4.2. Multiagent and multi-obstacle benchmarking. We also tested the expert and the student in two different environments: one with one agent and two obstacles, and another with three agents and two obstacles. To conduct the experiment, we positioned the agents in a 3.0 m radius circle and had them exchange positions diagonally, as shown in Fig. 4. We set the maximum dynamic limits to 2.0 m/s, 10.0 m/s^2, and 30.0 m/s^3 for velocity, acceleration, and jerk, respectively.

We conducted all simulations on an Alienware Aurora r8 desktop running Ubuntu 20.04, which is equipped with an Intel® Core™ i9-9900K CPU clocked at 3.60 GHz with 16 cores and 62.6 GiB of RAM.

Table 6 and Fig. 7 compare the average performance of the expert and student in two different environments: (1) one agent with two obstacles, and (2) three agents with two obstacles. The metrics used to evaluate the performance are as follows:

1. Computation time: the time it takes to replan at each step.
2. Sucess rate: the rate at the agents successfully reach the goal without collisions.
3. Travel time: the time it takes for the agent to complete the position exchange.
4. FOV rate: the percentage of time that the agent keeps obstacles within its FOV when the agent is closer than its camera's depth range.
5. Number of continuous FOV detection frames: the number of consecutive frames that an obstacle is kept within the FOV of the agent.
6. Dynamic constraints violation rate: the violation rate of the maximum velocity, acceleration, jerk, and yaw rate.

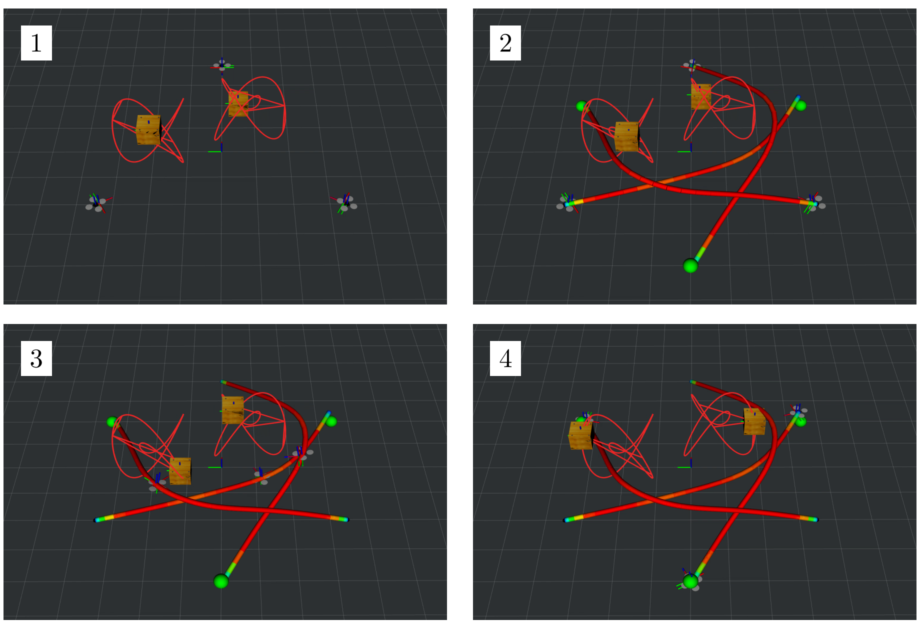


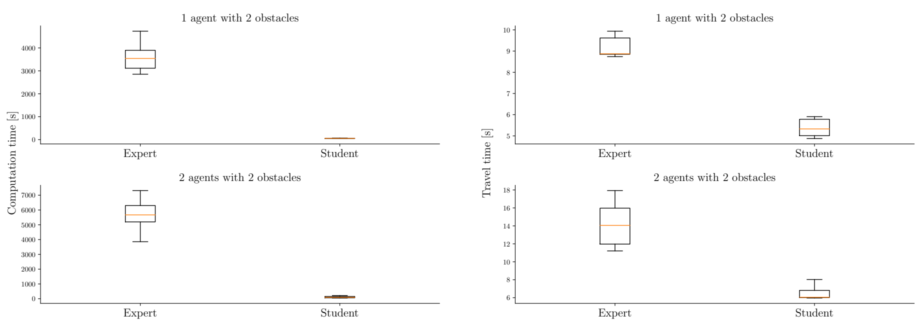
Fig. 6. Student mingle-agent, mingle-obstacle, simulation result: We made three imitation learning-based (student) agents fly around two dynamic obstacles. They started at the top-right corner and was commanded to fly to the down-left. For simplicity, we omitted FOV tripods visualization.

Both the expert and the studnet achieve successful position exchange with the two dynamic obstacles, with similar performance. However, the student significantly outperforms the expert in terms of computation time, completing the task in only 57 ms compared to the much slower expert.

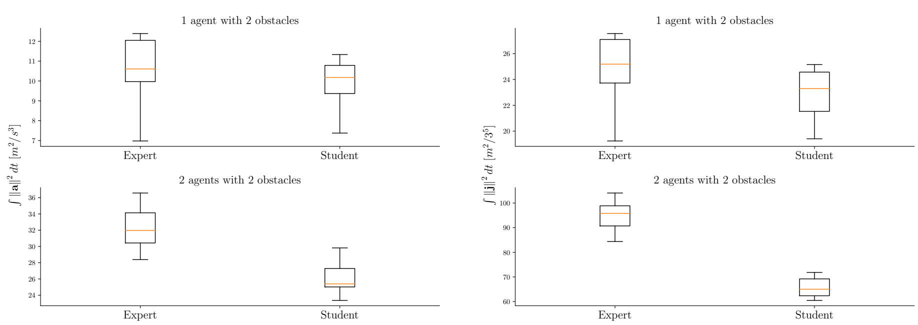
In the more complex environment with three agents and two obstacles, the expert and the student both achieve a high success rate, while the expert does not complete

Table 6. Benchmarking

Env.	Method	Avg. Compu. Time [ms]	Success Rate [%]	Avg. Travel Time [s]	FOV Rate [%]	Avg. of Max # Conti. FOV Detection Frames	Dyn. Constr. Violation Rate [%]
1 agent + 2 obst.	**Expert**	**3456**	**100**	7.9	29.0	19.8	0
	Student	**57**	**100**	4.5	28.0	31.0	10.3
3 agents + 2 obst.	**Expert**	**6212**	**0**	13.0	19.6	65.7	0.0
	Student	**119**	**80**	5.8	25.0	35.3	5.4



(a) Computation time (b) Travel time



(c) Trajectory smoothness (Acceleration) (d) Trajectory Smoothness (Jerk)

Fig. 7. Results of flight simulations. (a) The student's computation time is much faster than that of the expert, and (b) the student's travel time is also much shorter; this is mainly because of the faster computation time. (c-d) Since the student achieves faster replanning, it does not need to stop as the expert does, and that leads to smoother trajectory generation.

any position exchange. The reason for this is that when agents spend too much time optimizing their trajectory, the constraints used in the optimization become outdated by the time the optimization is complete, resulting in trajectory conflicts during the check and delay check steps. The expert suffers from long computation time, and as a result, almost none of the trajectories it generates pass the check and delay check steps, leading to a low success rate.

5. Conclusions. In conclusion, our work has addressed the critical issue of trajectory deconfliction in perception-aware, decentralized multiagent planning. We first presented an optimization-based perception-aware, decentralized, asynchronous multiagent trajectory planner (expert) that enabled teams of agents to navigate uncertain environments while avoiding obstacles and deconflicting trajectories using perception information. Although these methods achieved state-of-the-art performance, they suffered from high computational costs, making it difficult for agents to replan at high rates.

To overcome this challenge, we presented a learning-based planner that was trained with imitation learning (IL). This approach is a computationally-efficient deep neural network and achieved a computation speedup of up to 6000 times faster than optimization-based approaches, while maintaining high performance. This speedup enables scalability to a large number of agents, making the student a promising approach for large-scale swarm coordination. We used Julia MPI for parallelized training which led to 3.68 times faster training.

Moving forward, our future work will focus on larger-scale simulations and hardware flight experiments to demonstrate the scalability and performance of the student in complex environments with many agents and obstacles. Additionally, we will explore how to integrate the student with other state-of-the-art perception systems, such as SLAM, to enable even more robust and accurate perception-aware multiagent trajectory planning. Ultimately, our work has demonstrated the potential for learning-based approaches to address critical challenges in decentralized multiagent trajectory planning, and we believe that these approaches will play an essential role in enabling the deployment of multiagent systems in real-world applications.

Appendix A. Multi-obstacle Optimization Formulation.

To enable tracking multiple obstacles and agents we modified the FOV term given in Section 4 in [23] as the following.

$$-\alpha_{FOV}\sum_{i}^{n}\{\int_{0}^{T}(\text{inFOV}(\text{obstacle}_i))^3 dt\} \tag{A.1}$$

where α_{FOV} is the weight, n is the number of obstacles, T is the total time of the trajectory, inFOV() returns a higher number when obstacle$_i$ is in FOV.

REFERENCES

[1] S. Batra, Z. Huang, A. Petrenko, T. Kumar, A. Molchanov, and G. S. Sukhatme, *Decentralized control of quadrotor swarms with end-to-end deep reinforcement learning*, in Conference on Robot Learning, PMLR, 2022, pp. 576–586.

[2] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, *Julia: A fresh approach to numerical computing*, SIAM review, 59 (2017), pp. 65–98, https://doi.org/10.1137/141000671.

[3] M. Everett, Y. F. Chen, and J. P. How, *Motion planning among dynamic, decision-making agents with deep reinforcement learning*, in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 3052–3059, https://doi.org/10.1109/IROS.2018.8593871.

[4] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, *A distributed multi-robot coordination algorithm for navigation in tight environments*, arXiv preprint arXiv:2006.11492, (2020).

[5] Y. Gao, Y. Wang, X. Zhong, T. Yang, M. Wang, Z. Xu, Y. Wang, Y. Lin, C. Xu, and F. Gao, *Meeting-merging-mission: A multi-robot coordinate framework for large-scale communication-limited exploration*, in 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 13700–13707, https://doi.org/10.1109/IROS47612.2022.9981544.

[6] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural Computation, 9 (1997), pp. 1735–1780.

[7] J. Hou, X. Zhou, Z. Gan, and F. Gao, *Enhanced decentralized autonomous aerial swarm with group planning*, ArXiv, abs/2203.01069 (2022).

[8] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[9] K. Kondo, R. Figueroa, J. Rached, J. Tordesillas, P. C. Lusk, and J. P. How, *Robust mader: Decentralized multiagent trajectory planner robust to communication delay in dynamic environments*, arXiv preprint arXiv:2303.06222, (2023).

[10] C. E. Luis, M. Vukosavljev, and A. P. Schoellig, *Online trajectory generation with distributed model predictive control for multi-robot motion planning*, IEEE Robotics and Automation Letters, 5 (2020), pp. 604–611, https://doi.org/10.1109/LRA.2020.2964159.

[11] B. Park and H. Oh, *Vision-based obstacle avoidance for uavs via imitation learning with sequential neural networks*, International Journal of Aeronautical and Space Sciences, 21 (2020), pp. 768 – 779.

[12] J. Park, J. Kim, I. Jang, and H. J. Kim, *Efficient Multi-Agent Trajectory Planning with Feasibility Guarantee using Relative Bernstein Polynomial*, in 2020 IEEE International Conference on Robotics and Automation (ICRA), May 2020, pp. 434–440, https://doi.org/10.1109/ICRA40945.2020.9197162. ISSN: 2577-087X.

[13] P. Peng, W. Dong, G. Chen, and X. Zhu, *Obstacle avoidance of resilient uav swarm formation with active sensing system in the dense environment*, arXiv preprint arXiv:2202.13381, (2022).

[14] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, *Vision-based minimum-time trajectory generation for a quadrotor uav*, in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 6199–6206, https://doi.org/10.1109/IROS.2017.8206522.

[15] D. R. Robinson, R. T. Mar, K. Estabridis, and G. Hewer, *An Efficient Algorithm for Optimal Trajectory Generation for Heterogeneous Multi-Agent Systems in Non-Convex Environments*, IEEE Robotics and Automation Letters, 3 (2018), pp. 1215–1222, https://doi.org/10.1109/LRA.2018.2794582. Conference Name: IEEE Robotics and Automation Letters.

[16] S. Ross, G. Gordon, and D. Bagnell, *A reduction of imitation learning and structured prediction to no-regret online learning*, in Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[17] G. Ryou, E. Tal, and S. Karaman, *Cooperative Multi-Agent Trajectory Generation with Modular Bayesian Optimization*, in Robotics: Science and Systems XVIII, Robotics: Science and Systems Foundation, June 2022, https://doi.org/10.15607/RSS.2022.XVIII.060, http://www.roboticsproceedings.org/rss18/p060.pdf (accessed 2022-07-08).

[18] B. Sabetghadam, R. Cunha, and A. Pascoal, *A distributed algorithm for real-time multi-drone collision-free trajectory replanning*, Sensors, 22 (2022), https://doi.org/10.3390/s22051855, https://www.mdpi.com/1424-8220/22/5/1855.

[19] I. Spasojevic, V. Murali, and S. Karaman, *Perception-aware time optimal path parameterization for quadrotors*, in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 3213–3219, https://doi.org/10.1109/ICRA40945.2020.9197157.

[20] A. Tagliabue, D.-K. Kim, M. Everett, and J. P. How, *Demonstration-efficient guided policy search via imitation of robust tube mpc*, 2022 International Conference on Robotics and Automation (ICRA), (2021), pp. 462–468.

[21] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, *Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor*, IEEE Robotics and Automation Letters, 2 (2017), pp. 1762–1769, https://doi.org/10.1109/LRA.2017.2702198.

[22] J. Tordesillas and J. P. How, *MADER: Trajectory planner in multi-agent and dynamic environments*, IEEE Transactions on Robotics, (2021).

[23] J. Tordesillas and J. P. How, *PANTHER: Perception-aware trajectory planner in dynamic environments*, arXiv preprint arXiv:2103.06372, (2021).

[24] J. Tordesillas and J. P. How, *Deep-panther: Learning-based perception-aware trajectory planner in dynamic environments*, IEEE Robotics and Automation Letters, 8 (2023), pp. 1399–1406, https://doi.org/10.1109/LRA.2023.3235678.

[25] C. Toumieh, *Decentralized multi-agent planning for multirotors: a fully online and communication latency robust approach*, arXiv preprint arXiv:2304.09462, (2023).

[26] C. Toumieh and A. Lambert, *Decentralized Multi-Agent Planning Using Model Predictive*

Control and Time-Aware Safe Corridors, IEEE Robotics and Automation Letters, 7 (2022), pp. 11110–11117, https://doi.org/10.1109/LRA.2022.3196777. Conference Name: IEEE Robotics and Automation Letters.

[27] R. Van Parys and G. Pipeleers, *Distributed model predictive formation control with inter-vehicle collision avoidance*, in 2017 11th Asian Control Conference (ASCC), IEEE, 2017, pp. 2399–2404.

[28] Z. Wang, C. Xu, and F. Gao, *Robust trajectory planning for spatial-temporal multi-drone coordination in large scenes*, in 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022, pp. 12182–12188, https://doi.org/10.1109/IROS47612.2022.9982032.

[29] X. Wu, S. Chen, K. Sreenath, and M. W. Mueller, *Perception-aware receding horizon trajectory planning for multicopters with visual-inertial odometry*, IEEE Access, 10 (2022), pp. 87911–87922, https://doi.org/10.1109/ACCESS.2022.3200342.

[30] B. Zhou, J. Pan, F. Gao, and S. Shen, *Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight*, IEEE Transactions on Robotics, 37 (2021), pp. 1992–2009, https://doi.org/10.1109/TRO.2021.3071527.

[31] X. Zhou, J. Zhu, H. Zhou, C. Xu, and F. Gao, *EGO-Swarm: A Fully Autonomous and Decentralized Quadrotor Swarm System in Cluttered Environments*, Nov. 2020, https://doi.org/10.48550/arXiv.2011.04183, http://arxiv.org/abs/2011.04183 (accessed 2022-07-05). arXiv:2011.04183 [cs] version: 1.