RAPID: Robust and Agile Planner Using Inverse Reinforcement learning for Drones

Minwoo Kim*,†,1, Geunsik Bae*,1, Jinwoo Lee[1], Woojae Shin[1], Heejung Shin[1], Myong-Yol Choi[1] and Hyondong Oh††,1

***Abstract*— In this paper, we introduce a visual learning-based planner for agile drone flight in cluttered environments, called RAPID. Leveraging a neural network without separate perception and mapping modules, the proposed planner can generate collision-free trajectories within a few milliseconds. To achieve robust and sample efficient learning, an inverse reinforcement learning (IRL) framework is used for training. Expert trajectories are generated from motion primitives in diverse geometric environments, such as narrow gaps, cubes, spheres, and trees. The IRL method allows the learning policy to fully exploit the expert dataset while exploring diverse states without an explicit reward function to learn expert-like behavior. While the proposed method is trained in a simulation environment, it can be directly applied to real-world scenarios without additional training as real-like synthetic depth images are used in training. The efficacy of the method is rigorously evaluated using quantitative metrics and validated across various simulation environments.**

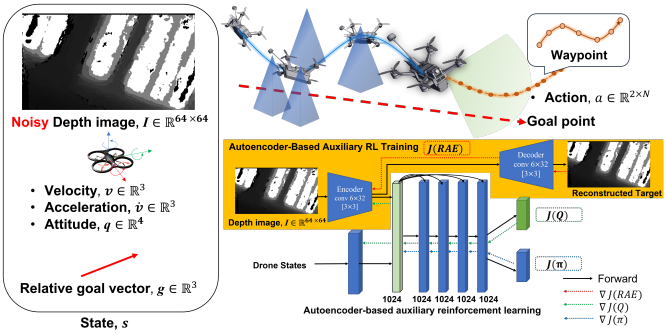


Fig. 1: Overview of RAPID. Taking a depth image, odometry, and goal vector, RAPID generates a collision-free trajectory.

I. INTRODUCTION

Small unmanned aerial vehicles (UAVs), also known as drones, are agile and compact, making them ideal for diverse applications such as search and rescue operations in disaster areas, urban indoor environment exploration, and target tracking. However, utilizing this agility in complex environments (e.g., forests and factories) is still limited due to the challenges in perception, control, and real-time motion planning. Thus, to fully exploit agility and speed, the development of high-speed visual collision avoidance algorithms in unknown and complex environments becomes a necessity.

Conventional visual collision avoidance methods rely on modular architectures combining perception and planning [1], providing certain level of performance guarantees but incurring high computational costs and latency unsuitable for agile drone flight. In contrast, neural network-based end-to-end learning integrates perception and planning into a single process, reducing latency and enabling rapid real-time planning [2].

In learning-based approaches, imitation learning (IL) is a representative method, with behavior cloning (BC) being the most widely used. BC aims to mimic expert behavior (i.e., optimal policies) but requires substantial training data, which is challenging to obtain due to time and cost constraints. Without enough data, BC suffers from issues like compounding errors in unfamiliar states [3]. Dataset aggregation (DAgger) addresses these drawbacks by incrementally collecting expert data in new environments to train a more robust policy, but acquiring additional datasets during training is generally difficult [4].

In contrast to BC methods, reinforcement learning (RL) methods enhance robustness by allowing agents to interact with environments during policy training. However, they have drawbacks such as complex reward function design, sample inefficiency compared with supervised learning, and longer training times due to the required exploration. While various RL studies have shown good performance in low-speed vision-based flight [5], directly applying RL to high-speed flight is fairly limited due to these issues.

Inverse reinforcement learning (IRL) aims to learn an underlying reward from expert behaviors and then derive an optimal policy from that learned reward. It shares similarities with the BC method in that both methods use expert datasets. However, IRL can find a better policy with fewer demonstrations, as it can mitigate the problem of compounding errors by sampling new states that are unseen in the expert dataset. A representative IRL method is generative adversarial imitation learning (GAIL) [6], which integrates IRL and RL training, making the process faster and more stable. However, it struggles with issues such as mode collapse [7], commonly occur in adversarial networks, and the biased reward problem [8] due to mistreatment of absorbing states during RL training.

Recently, non-adversarial learning-based IRL methods

*Equal contributions. †Project lead. ††Corresponding author.
[1]Minwoo Kim, [1]Geunsik Bae, [1]Jinwoo Lee, [1]Woojae Shin, [1]Heejung Shin, [1]Myong-Yol Choi, and [1]Hyondong Oh are with Department of Mechanical Engineering, Ulsan National Institute of Science and Technology, Republic of Korea red9395@unist.ac.kr, baegs94@unist.ac.kr, jinwoolee2021@unist.ac.kr, oj7989@unist.ac.kr, godhj@unist.ac.kr, mychoi@unist.ac.kr, h.oh@unist.ac.kr

such as inverse soft Q-imitation learning (IQ-learning) [9] have improved performance and stability, but still faces problems like the curse of dimensionality in vision-based continuous action spaces and biased reward. To solve these problems, least square inverse Q-learning (LSIQ) [10] introduces Q-function clipping and proper absorbing state treatment, enhancing stability. However, in high-speed collision avoidance scenarios, these issues are more challenging, as considering feasible flight attitudes is crucial, making absorbing state treatment particularly difficult. Therefore, this paper addresses the issue by replacing implicit absorbing state treatment with explicit absorbing state treatment.

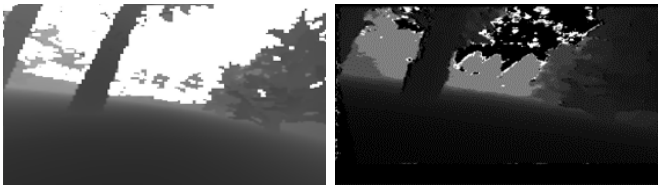
In addition to these specific challenges in IRL methods, the general use of high-dimensional encoders in RL presents difficulties related to sample inefficiency and training instability, which become much severe in IRL [9]. Since IRL involves not only learning a policy but also inferring an underlying reward function from demonstrations, it adds another layer of complexity. This dual objective intensifies sample inefficiency, as demonstrations provide limited and sometimes ambiguous feedback. As a result, the encoder must concurrently adapt to both the inferred rewards and policy updates, causing greater training instability and slower convergence compared to standard RL training.

Even if all these challenges are addressed, applying the trained neural network to real-world scenarios still faces the persistent issue of the sim-to-real gap, a fundamental challenge in learning-based approaches [2]. This gap is especially pronounced in vision-based planners, where the use of visual information amplifies the problem. Additionally, differences in noise characteristics, dynamics, and platform discrepancies between simulation and real-world environments further complicate real-world deployment. To develop policies that perform well in actual environments, it is crucial to account for these factors during the training process to ensure robust performance outside of controlled simulations.

A. Contributions

The purpose of this paper is to develop a generalizable and sample-efficient collision avoidance algorithm using high dimensional visual information, performing reliably in real-world scenarios without a single real-world data. In this paper, we propose a learning-based planner called ***R**obust and **A**gile **P**lanner using **I**nverse reinforcement learning for **D**rones (**RAPID**)*, which generates agile collision-free trajectories in cluttered environments. In summary, the contributions of this work are threefold:

- Through integration of inverse soft Q-learning with a proper absorbing state treatment significantly improves the drone's evasive maneuvers;
- The introduction of autoencoder-based auxiliary loss function helps mitigate a state complexity when using a high dimensional visual input, allowing faster learning; and
- Overcome the sim2real gap problem by adopting a domain randomization method using a domain invariant feature representation.



(a) Simulated depth (b) Stereo depth image

Fig. 2: Depth images from simulation. Stereo depth images are used for training.

The rest of the paper is organized as follows. Section II introduces the proposed method. Section III shows the simulation environments, dataset acquisition process, and an in-depth comparison with the baseline methods.

II. RAPID

A. Preliminaries

The vision-based path planning problem can be modeled as an infinite-horizon Markov decision process (MDP). The MDP is composed of $(s, a, p(s_0), s', p(s', s|a), r(s, a), \gamma)$, where s is a state, a is an action, $p(s_0)$ is an initial state distribution, s' is a next state, $p(s', s|a)$ is a transition probability, $r(s, a)$ is a reward function and $\gamma \in [0, 1]$ is a discount factor. π is a stochastic policy that takes an action a given a state s. Data from the expert policy will be denoted as D_{π_E}, while data from the learner policy will be denoted as D_π. Additionally, the expert data distribution is represented as d_{π_E}, and the learner data distribution is represented as d_π. A detailed explanation regarding states and actions will be discussed in the following section.

B. States and Actions

1) States: The policy network $\pi(a_t|s_t)$, generates an action a_t at time step t. The state space s_t is defined as:

$$s_t = [I_t, v_t, \dot{v}_t, q_t, g_t],$$

where a depth image $\boldsymbol{I} \in \mathbb{R}^{64\times64}$, velocity $\boldsymbol{v} \in \mathbb{R}^3$, acceleration $\dot{\boldsymbol{v}} \in \mathbb{R}^3$, attitude quaternion $\boldsymbol{q} \in \mathbb{R}^4$, and a relative goal vector $\boldsymbol{g} \in \mathbb{R}^3$ (i.e., the difference between the goal and the current position of the drone).

In general, domain-invariant features such as depth images are used to bridge the gap between simulation and real-world environments. However, there are differences between the depth images from the simulation and real-world environments. Therefore, these differences need to be addressed to overcome the sim-to-real gap. To this end, a stereo depth image similar to a real depth image is synthesized using the semi-global matching (SGM) method and used for training. The two images can be visually compared in Fig. 2.

2) Actions: The action a_t consists of N waypoints ahead, each separated by a fixed time interval T. Each waypoint represents a relative position from the previous point, expressed in cylindrical coordinates to reduce action-space complexity. Specifically, each waypoint is defined by a relative distance

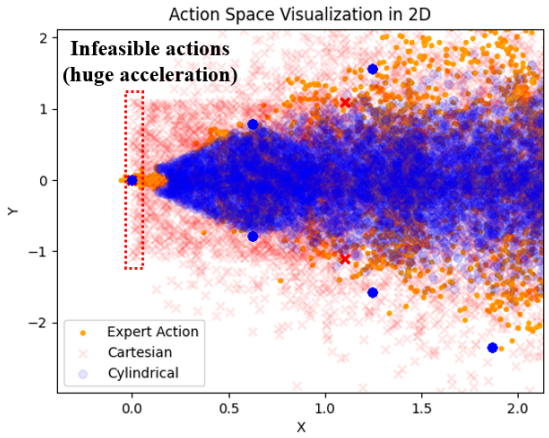


Fig. 3: Action space representation with different coordinate systems.

Δr_i and a relative angle $\Delta\psi_i$ from the previous point. The raw waypoints output from the policy network are:

$$a_t^{raw} = \{(\Delta r_1, \Delta\psi_1), (\Delta r_2, \Delta\psi_2), \ldots, (\Delta r_N, \Delta\psi_N)\},$$

where $i = 1, 2, \ldots, N$ indicates an indice of the waypoints. These waypoints are transformed into Cartesian coordinates using cumulative heading angles. Let $\theta_0 = \psi_t$ be the initial heading angle of the drone. The cumulative heading angle for the i-th waypoint is:

$$\theta_i = \theta_{i-1} + \Delta\psi_i, \quad \text{for } i = 1, 2, \ldots, N.$$

The position of the i-th waypoint in Cartesian coordinates is then calculated recursively:

$$\mathbf{p}_i = \mathbf{p}_{i-1} + \Delta r_i \begin{bmatrix} \cos\theta_i \\ \sin\theta_i \end{bmatrix}, \quad \text{with } \mathbf{p}_0 = \begin{bmatrix} x_t \\ y_t \end{bmatrix}.$$

Accordingly, the final transformed action vector a_t is:

$$a_t = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\},$$

where each $\mathbf{p}_i$ is the absolute position of the i-th waypoint in Cartesian coordinates.

Figure 3 shows the action space with different coordinate systems. In this study, the appropriate action space range should encompass the expert action range while minimizing exploration. As shown in Fig. 3, actions represented with Cartesian coordinates generate infeasible actions, particularly producing straight-line actions in the y-direction, which negatively affects learning. However, when using cylindrical coordinates, the action range becomes more aligned with that of the expert, eliminating the initial infeasible actions. Although the proposed action representation might seem unconventional, this design confines the action range within a specific boundary and stabilizes the training.

C. Trajectory Generation and Control

Given discrete waypoints generated by the planner, it is necessary to convert them into a continuous and differentiable trajectory for smooth flight. The trajectory $\boldsymbol{\tau}(t)$ can

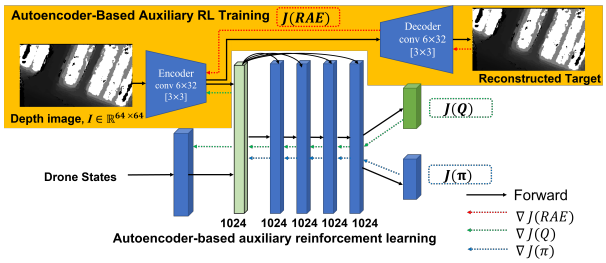


Fig. 4: Autoencoder-based auxiliary reinforcement learning with skipping connection networks.

be represented as a distinct function along each axis as:

$$\boldsymbol{\tau}(t) := [\tau_x(t), \tau_y(t), \tau_z(t)]^T.$$

For each axis, the trajectory can be represented as an M^{th}-order of piecewise polynomial function with N time intervals:

$$\tau_\mu^k(t) = \textstyle\sum_{i=0}^{M} \sigma_{\mu,i}^k (t - t_0 - kT)^i,$$
$$t_0 + (k-1)T \le t < t_0 + kT,$$

where $\mu \in \{x, y, z\}$ and $k = 1, \cdots, N$.

Each polynomial segment must start and end at specified waypoints and ensure a smooth transition by maintaining the continuity of the j^{th} derivative at each intermediate waypoint. Moreover, the first segment should initiate from the drone's current position, velocity, and acceleration. The trajectory that minimizes the integral of the jerk squared can be found by solving the optimization problem as:

$$\min_{\boldsymbol{\sigma}^1, \boldsymbol{\sigma}^2, \cdots, \boldsymbol{\sigma}^N} J = \int_{t_0}^{t_N} \|\dddot{\boldsymbol{\tau}}(t)\|^2 dt,$$

where $\boldsymbol{\sigma}^k \in \mathbb{R}^{(M+1)\times 3}$ represents the coefficients of the k^{th} polynomial segment. The objective J can be analytically computed by integrating the polynomial, and it is formulated as constrained quadratic programming (QP). A closed-form solution is obtained by mapping the polynomial coefficients to the derivatives at each segment boundary, as outlined in [11].

In this study, we adopt the method from [11], representing the trajectory as a 5^{th}-degree polynomial while ensuring continuity in both velocity and acceleration at the waypoints (i.e., $M = 5$ and $j = 1, 2$). To achieve precise tracking of the generated trajectory, we employ a geometric controller based on [12].

D. Sample Efficient Training with Image Reconstruction

1) Autoencoder-Based Auxiliary Loss function: Learning with high-dimensional inputs complicates the RL policy, as the agent should also learn a compact latent representation to embed high-dimensional states. Additionally, the images used for training contain a significant amount of noisy information. Fitting a high-capacity encoder using only a scarce reward signal with large input noise is sample inefficient and prone to suboptimal convergence.

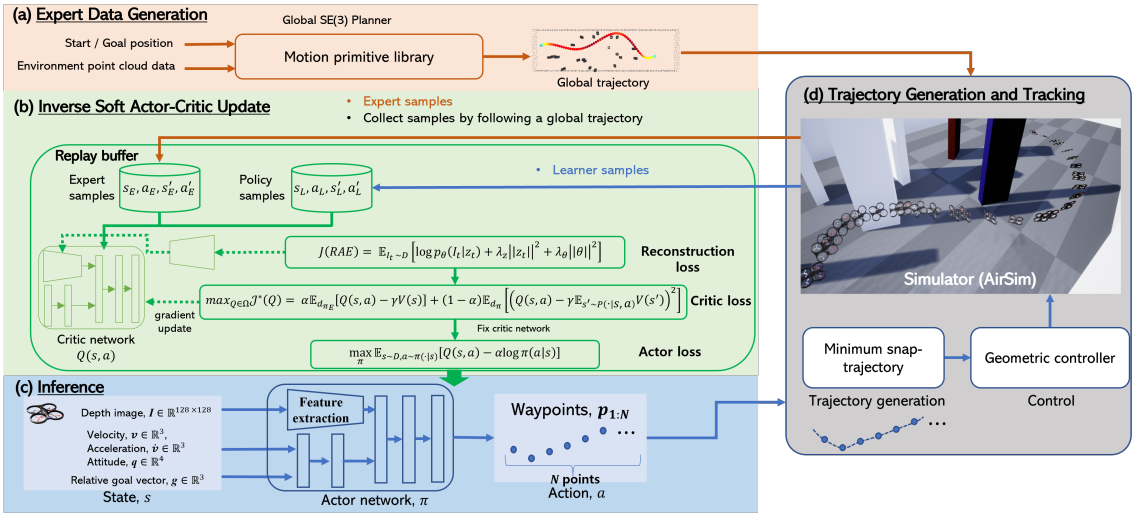


Fig. 5: End-to-end learning diagram of inverse soft Q-imitation learning. The learning framework is composed of four parts: (a) Expert data generation, (b) inverse soft actor-critic update, (c) inference, and (d) trajectory generation and tracking.

In this paper, we aim to use an autoencoder (AE) to mitigate the following challenges: i) effectively embed high-dimensional inputs and ii) address noisy image inputs. As demonstrated in previous studies, jointly learning the policy and auxiliary objectives has been shown to improve performance [13]. Following the prior work, we utilized unsupervised training through an image-based AE. In practice, AE is composed of two parts: encoder and decoder. A convolutional encoder g_ϕ maps an image observation I_t to a low-dimensional latent vector z_t, and the deconvolutional decoder f_θ reconstructs z_t back to the original state I_t. To achieve more stable training with better performance, L_2 penalty on the learned representation z_t and weight-decay on the decoder parameter θ are imposed. The objective function of reconstruction autoencoder $J(RAE)$ is given as:

$$J(RAE) = \mathbb{E}_{I_t\sim D}[\log p_\theta(I_t|z_t) + \lambda_z||z_t||^2 + \lambda_\theta||\theta||^2], \quad (1)$$

where $z_t = g_\phi(I_t)$, λ_z and λ_θ are hyperparameters. The proposed approach is summarized in Fig. 4.

2) Skipping Connection Networks: Although deeper networks generally perform better on complex tasks by introducing inductive biases, simply adding more layers in deep reinforcement learning (DRL) does not yield the same benefits as in computer vision tasks. This is because additional layers can decrease mutual information between input and output due to non-linear transformations. To overcome this issue, skip connections can be used to preserve important input information and enable faster convergence. We applied the deeper dense RL (D2RL) [14] network, which incorporates such skip connections into DRL, to the high-speed collision avoidance problem. This allowed us to gain the advantages of deeper neural networks while achieving faster learning. The skip connection network is illustrated in Fig. 4.

E. Learning Robust Policy Without Explicit Reward

The IRL algorithm used in this paper is least square inverse Q-learning (LSIQ) [10] which directly learns a Q-function using an implicit reward formulation. Previously, it was necessary to train two neural networks simultaneously in the reward-policy domain. However, by utilizing the soft Q-function concept [15], the reward function can now be derived from the Q-function $Q(s,a)$ and the value function $V(s)$ as:

$$r(s,a) = Q(s,a) - \gamma\mathbb{E}_{s'\sim P(\cdot|s,a)}[V^\pi(s')]. \quad (2)$$

In LSIQ, the use of the inverse Bellman operator allows the objective function L_ρ to be transformed from the reward-policy space to the Q-policy space, denoted as J_ρ:

$$\max_{r\in R}\min_{\pi\in\Pi} L_\rho(r,\pi) = \max_{Q\in\omega}\min_{\pi\in\Pi} J_\rho(Q,\pi),$$

where $\Omega = \mathbf{R}^{S\times A}$ is the space of Q-functions. The transformation enables faster learning without additional reward network training. Furthermore, LSIQ improves the stability of implicit learning methods effectively by properly handling absorbing states and clipping the Q-function during training.

However, in problems like high-speed collision avoidance, where terminal states are frequently encountered, learning absorbing states recursively through bootstrapping proved to be highly unstable. While the LSIQ paper presents bootstrapping and analytical approaches as separate methods for learning, our tests showed that combining the bootstrapping approach for normal states with the analytical approach for absorbing states is much more effective. Details regarding the proof of inverse bellman operator for absorbing states and reward range can be found in [9], [10].

F. Inverse Soft Actor-Critic Update

To train a policy, soft actor-critic (SAC) is used. SAC methods require an expert and learner dataset. The expert dataset is collected offline using a global planner, whereas the

learner dataset is collected online through agent interaction with an environment. These two datasets are stored in one replay buffer. In the replay buffer, expert data and learner data are sampled at the same ratio. The objective of Q-function is given as:

$$J(Q) = \alpha \mathbb{E}_{d_{\pi_E}}[Q(s,a) - \gamma V(s)] + (1-\alpha)\mathbb{E}_{d_\pi}\left[(Q(s,a) - \gamma \mathbb{E}_{s' \sim p(\cdot|s,a)} V(s'))^2\right], \quad (3)$$

where α is a regularizer constant and is set to 0.5.

In the continuous action space, there is no direct way to get an optimal policy. Instead, an explicit policy π is used to approximate π_Q by using the SAC method. With a fixed Q-function, the policy π is updated using the following equation:

$$\max_{\pi} \mathbb{E}_{s \sim D, a \sim \pi(\cdot|s)}[Q(s,a) - \alpha \log \pi(a|s)], \quad (4)$$

where D is a replay buffer. The whole process is described in Fig. 5.

III. Results

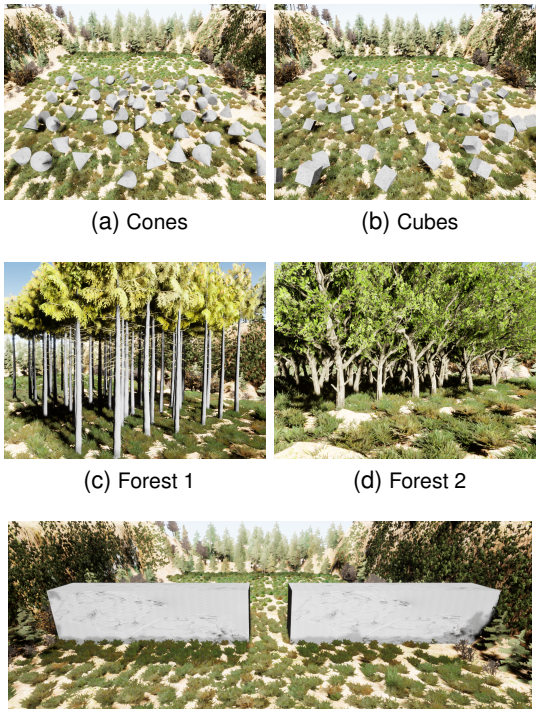
A. Data Acquisition and Training

To enhance generalization performance, a variety of training environments (e.g., trees, cones, and spheres) are generated as shown in Fig. 6. The AirSim simulator is used for map building and testing the algorithm. For data acquisition, a motion primitive-based expert planner is employed, which necessitates prior knowledge of the map. Point cloud data (PCD) is first gathered to construct a global trajectory, after which local trajectories are sampled by considering an obstacle cost. The overview of the data collection process can be found in Fig. 5(a).

To achieve generalization performance, domain randomization techniques are applied. First, in each episode, the drone's learning is initiated from a random starting position. Additionally, about 10 percent noise is added to the drone's controller gain values to introduce randomness. To enhance the robustness of the encoder during the learning process, the image random shuffling technique is used. Furthermore, it is discovered that network initialization plays an important role during the learning process. Commonly used initialization like Xavier initialization makes the learning unstable. To address this issue, an initialization technique with zero mean [13] is applied to create a straight path initially. If the drone collides with an obstacle or reaches the goal point, the episode is terminated, and the map is replaced every 15 episodes.

B. Simulation Results

To quantitatively evaluate the proposed model, the RAPID planner is compared with the state-of-the-art learning-based planner, agile autonomy [2], and the conventional BC method. Agile autonomy learns collision-free trajectories using DAgger with relaxed winner-takes-all (R-WTA) loss, overcoming multi-modality problem in conventional BC method. Trajectories generated by agile autonomy is originally tracked using model predictive control (MPC) as its



(a) Cones (b) Cubes (c) Forest 1 (d) Forest 2 (e) Narrow gap

Fig. 6: Various training environments.

controller. However, in this comparison, to evaluate collision-free trajectory generation based on the state, minimum-snap trajectory generation is employed for path planning, and a geometric controller is used to track the generated trajectories. The purpose of using MPC is to consider dynamics to follow trajectories that might otherwise be infeasible, but the geometric controller cannot impose such constraints. Instead, this limitation is partially compensated for by adding velocity and acceleration constraints during the trajectory generation process. BC model uses a pretrained MobileNetV3 [16] structure and 1D-convolutional layers, identical to the structure used in agile autonomy.

The experiments are conducted under varying conditions based on map density, obstacle sizes, and shapes. The map density represents the number of trees per unit area. Trees are randomly placed with the diameter of $1.0 - 1.5m$ over $60m \times 60m$ map. The evaluation metrics are set to mission progress (MP), velocity, and traveled distance. MP represents progress toward a goal from a starting position. Figure 7 shows the traveled trajectories with varying map densities. The straight baseline is used to show that collisions occur easily when avoidance commands are not generated. As shown in Fig.7, the BC algorithm shows the lowest performance. This is mainly due to the tendency to generate straight trajectories, leading to negative relation between map density and MP. Agile autonomy shows a consistent performance across all densities, compared with BC methods. Although agile autonomy shows a strong tendency to avoid collisions, its excessive tendency to make large directional

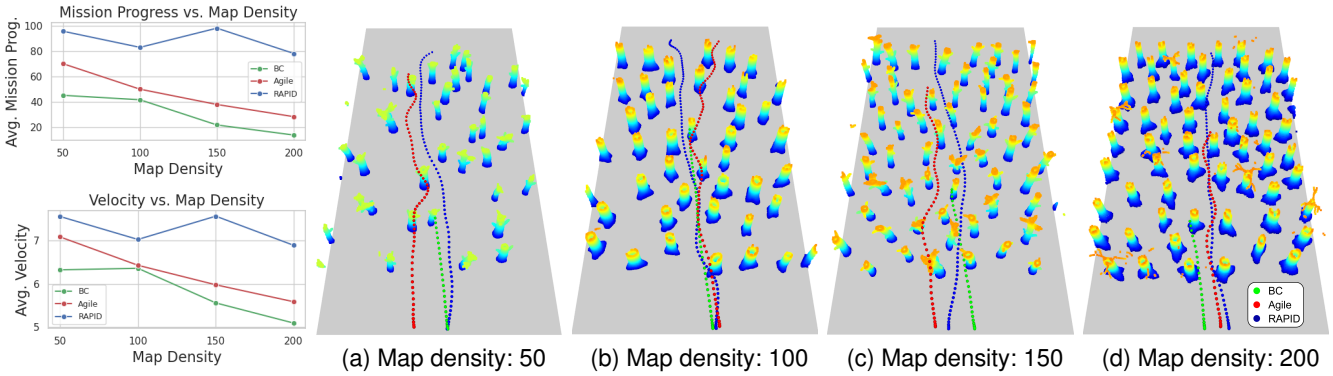


Fig. 7: Baseline comparison in cluttered environments. Forest environments with various types and sizes are used for testing.

TABLE I: Evaluation on various maps. The results are averaged over five trials with different starting positions and goals.

Algorithm	**Avg. MP [m]**				**Avg. Velocity [m/s]**				**Avg. Distance [m]**				**Avg. Cost [km/s^2]**			
Map Density [tree/m^2]	50	100	150	200	50	100	150	200	50	100	150	200	50	100	150	200
Straight	37.2	26.0	7.8	6.4	7.48	4.7	5.6	4.8	22.3	15.6	17.8	10.2	138.0	30.0	48.0	32.0
BC	45.2	41.7	21.9	13.9	6.3	6.4	5.6	5.1	34.9	32.7	19.6	14.4	86.8	82.8	55.2	43.2
Agile Autonomy	70.2	50.1	38.0	28.4	7.1	6.4	6.0	5.6	54.8	39.8	29.7	22.3	138.5	104.5	80.4	61.2
RAPID (Ours)	**95.8**	**83.1**	**98.2**	**78.3**	7.6	7.0	7.6	6.9	63.9	58.6	67.4	52.4	159.7	165.3	174.1	144.4

changes leads to collisions. However, the RAPID algorithm shows the best collision avoidance performance across all maps. While RAPID uses the same training data as BC, it achieves significantly better performance, demonstrating that the proposed framework is highly effective for vision-based planning. Table I shows the results of five experiments conducted on each map, showing that the RAPID algorithm consistently outperforms the comparison algorithms. The study of agile autonomy [2] highlighted the limitations of experts in finding feasible trajectories during high-speed flight and suggested the potential of RL approaches. The proposed method leverages the advantages of IRL, combining expert data with environmental interaction to achieve robust performance.

ACKNOWLEDGMENT

This research was supported by National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (2023R1A2C2003130), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1A6A1A03040570), Unmanned Vehicles Core Technology Research and Development Program through the NRF, and Unmanned Vehicle Advanced Research Center (UVARC) funded by the Ministry of Science and ICT, the Republic of Korea (2020M3C1C1A01082375).

REFERENCES

[1] K. Zhu and T. Zhang, "Deep Reinforcement Learning-Based Mobile Robot Navigation: A Review," Tsinghua Science and Technology, vol. 26, no. 5, pp. 674–691, 2021.

[2] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning High-Speed Flight in the Wild," Science Robotics, vol. 6, no. 59, p. 5810, 2021.

[3] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," pp. 661–668, 2010.

[4] S. Ross, G. Gordon, and D. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," International Conference on Artificial Intelligence and Statistics, pp. 627–635, 2011.

[5] M. Kim, J. Kim, M. Jung, and H. Oh, "Towards Monocular Vision-Based Autonomous Flight Through Deep Reinforcement Learning," Expert Systems with Applications, vol. 198, p. 116742, 2022.

[6] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," Advances in Neural Information Processing Systems, vol. 29, 2016.

[7] H. Thanh-Tung and T. Tran, "Catastrophic Forgetting and Mode Collapse in GANs," 2020 international joint conference on neural networks (ijcnn), pp. 1–10, 2020.

[8] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, "Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning," arXiv preprint arXiv:1809.02925, 2018.

[9] D. Garg, S. Chakraborty, C. Cundy, J. Song, and S. Ermon, "IQ-Learn: Inverse Soft-Q Learning for Imitation," Advances in Neural Information Processing Systems, vol. 34, pp. 4028–4039, 2021.

[10] F. Al-Hafez, D. Tateo, O. Arenz, G. Zhao, and J. Peters, "Ls-iq: Implicit reward regularization for inverse reinforcement learning," arXiv preprint arXiv:2303.00599, 2023.

[11] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," pp. 649–666, 2016.

[12] T. Lee, M. Leok, and N. H. McClamroch, "Geometric Tracking Control of a Quadrotor UAV on SE (3)," IEEE Conference on Decision and Control, pp. 5420–5425, 2010.

[13] D. Yarats, A. Zhang, I. Kostrikov, B. Amos, J. Pineau, and R. Fergus, "Improving sample efficiency in model-free reinforcement learning from images," in Proceedings of the aaai conference on artificial intelligence, vol. 35, no. 12, 2021, pp. 10 674–10 681.

[14] S. Sinha, H. Bharadhwaj, A. Srinivas, and A. Garg, "D2rl: Deep dense architectures in reinforcement learning," arXiv preprint arXiv:2010.09163, 2020.

[15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," International Conference on Machine Learning, pp. 1861–1870, 2018.

[16] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., "Searching for mobilenetv3," in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1314–1324.