IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

FlightBench: Benchmarking Learning-based Methods for Ego-vision-based Quadrotors Navigation

Shu-Ang Yu[1*], Chao Yu[1*†], Feng Gao[1*], Yi Wu[1,2], and Yu Wang[1†]

***Abstract*—Ego-vision-based navigation in cluttered environments is crucial for mobile systems, particularly agile quadrotors. While learning-based methods have shown promise recently, head-to-head comparisons with cutting-edge optimization-based approaches are scarce, leaving open the question of where and to what extent they truly excel. In this paper, we introduce FlightBench, the first comprehensive benchmark that implements various learning-based methods for ego-vision-based navigation and evaluates them against mainstream optimization-based baselines using a broad set of performance metrics. More importantly, we develop a suite of criteria to assess scenario difficulty and design test cases that span different levels of difficulty based on these criteria. Our results show that while learning-based methods excel in high-speed flight and faster inference, they struggle with challenging scenarios like sharp corners or view occlusion. Analytical experiments validate the correlation between our difficulty criteria and flight performance. Moreover, we verify the trend in flight performance within real-world environments through full-pipeline and hardware-in-the-loop experiments. We hope this benchmark and these criteria will drive future advancements in learning-based navigation for ego-vision quadrotors. Code and documentation are available at https://github.com/thu-uav/FlightBench.**

***Index Terms*—Software Tools for Benchmarking and Reproducibility, Deep Learning Methods, Vision-Based Navigation.**

I. Introduction

Ego-vision-based navigation in cluttered environments is a fundamental capability for mobile systems and has been widely investigated [1]–[3]. It involves navigating a robot to a goal position while avoiding collisions with obstacles, using equipped ego-vision cameras [4]. Quadrotors, known for their agility and dynamism [5], present unique challenges in achieving fast and safe flight. Traditionally, hierarchical methods address this problem by decoupling it into subtasks such as mapping, planning, and control [6], optimizing the trajectory to avoid collisions. In contrast, recent works [7], [8] have demonstrated that learning-based methods can unleash the full dynamic potential of agile platforms. These methods employ neural networks to generate a sequence of waypoints [5] or motion commands [7], [8]. Unlike the high computational costs associated with sequentially executed subtasks, this end-to-end manner significantly reduces processing latency and enhances agility [5].

Figure 1: Real-world validation of FlightBench.

Despite the promising results of learning-based navigation methods, the lack of head-to-head comparisons with state-of-the-art optimization-based methods makes it unclear in which areas they truly outperform and to what degree. Traditional methods are often evaluated using customized scenarios and sensor configurations [7], [9], [10], which complicates reproducibility and hinders fair comparisons. Moreover, the absence of a quantifiable approach for scenario difficulty further obscures the analysis of the strengths and weaknesses of current methods.

In this paper, we introduce FlightBench, a comprehensive benchmark that evaluates methods for ego-vision-based quadrotor navigation. We initially incorporated a suite of representative navigation methods, encompassing both ego-vision and privileged ones for an in-depth comparative analysis. Moreover, we established three criteria to measure the difficulty of scenarios, thereby creating a diverse array of tests that span a spectrum of difficulties. Finally, we compared these methods through simulated and real-world experiments, evaluating a wide range of performance metrics to gain a deeper understanding of their specific attributes. Our experiments indicate that learning-based methods demonstrate superior performance in high-speed flight scenarios and generally offer quicker inference times. However, they encounter difficulties in handling complex situations such as sharp turns or occluded views. In contrast, our findings show that traditional optimization-based methods perform well in challenging conditions, not just in success rate and flight quality, but also in computation time, particularly when they are meticulously designed. Our analytical experiments validate the effectiveness of the proposed criteria and emphasize the importance of latency randomization for learning-based methods. In summary, our key contributions include:

1) The development of FlightBench, the first unified open-source benchmark that facilitates the head-to-head comparison of learning-based and optimization-based meth-

Manuscript received: November 25, 2024; Revised March 2, 2025; Accepted May 9, 2025.

This paper was recommended for publication by Editor Pascal Vasseur upon evaluation of the Associate Editor and Reviewers' comments. This research was supported by National Natural Science Foundation of China (No.62406159, 62325405), Postdoctoral Fellowship Program of CPSF under Grant Number GZC20240830, China Postdoctoral Science Special Foundation 2024T170496.

* Equal contribution.

† Corresponding Authors. {yuchao, yu-wang}@mail.tsinghua.edu.cn

The authors are with [1]Tsinghua University, [2]Shanghai Qi Zhi Institute.

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

ods on ego-vision-based quadrotor navigation under various 3D scenarios.
2) The proposition of tailored task difficulty and performance metrics, aiming to enable a thorough evaluation and in-depth analysis of specific attributes for different methods.
3) Detailed experimental analyses in both simulation and real-world that demonstrate the comparative strengths and weaknesses of learning-based versus optimization-based methods, particularly in difficult scenarios.

II. Related Work

A. Planning methods for navigation.

Classical navigation algorithms typically use search or sampling to explore the configuration or state space and generate a free path [11], [12]. With optimization, a multi-objective optimization problem is often formulated to determine the optimal trajectory [13], [14]. This is commonly done using gradients from local maps, such as the Artificial Potential Field (APF) [15] and the Euclidean Signed Distance Field (ESDF) [10]. On the other side, the development of deep learning enables algorithms to perform navigating directly from sensory inputs such as images or lidar [6]. The policies are trained by imitating expert demonstrations [16] or through exploration under specific rewards [17]–[19]. Learning-based algorithms have been applied to various mobile systems, such as quadrupedal robots [4], wheeled vehicles [3], [20], and quadrotors [8], [21], [22]. In this work, we examine representative methods for ego-vision-based navigation on quadrotors, including three learning-based approaches and three optimization-based methods, providing a comprehensive comparison between these categories.

Table I: A comparison of FlightBench to other open-source benchmarks for navigation.

Benchmark	3D Scenarios	Classical Methods	Learning Methods	Sensory Input
MRBP 1.0 [23]	✗	✓	✗	LiDAR
Bench-MR [24]	✗	✓	✗	-
PathBench [25]	✓	✓	✓	-
Gibson Bench [26]	✗	✗	✓	Vision
OMPLBench [27]	✓	✓	✗	-
RLNav [28]	✗	✓	✓	LiDAR
Plannie [29]	✓	✓	✓	-
MP design [30]	✓	✓	✗	-
FlightBench (Ours)	✓	✓	✓	Vision

B. Benchmarks for navigation.

Several benchmarks exist for non-sensory-input navigation algorithms. OMPL [27] and Bench-MR [25] primarily focus on sampling-based methods, while PathBench [25] evaluates graph-based and learning-based methods. Plannie [29] offers sampling-based, heuristic, and learning-based methods for quadrotors. The most closely related work is MP design [30], which lacks vision-input baselines and does not consider the visual perception challenges in its proposed ECS metric. For methods with sensory inputs, most benchmarks are primarily designed for 2D scenarios. MRBP1.0 [23] and RLNav [28] evaluate planning methods using laser-scanning data for navigation around columns and cubes. GibsonBench [26] features a mobile agent equipped with a camera, navigating in interactive environments. As outlined in Tab. I, there's a notable lack of a benchmark with 3D scenarios and ego-vision inputs to assess and compare both classical and learning-based navigation algorithms, a gap that FlightBench aims to fill.

III. FlightBench

This section outlines the components of FlightBench, as summarized in Fig. 2. To design a set of `Tasks` with distinguishable characteristics for assessment, we propose three criteria, named task difficulty metrics, and develop ten tests across three scenarios based on these metrics. We integrate various representative `Baselines` to examine the strengths and features of both learning-based and optimization-based navigation methods. Furthermore, we establish a comprehensive set of performance `Evaluation Metrics` to facilitate quantitative comparisons. The next subsections provide an in-depth look at the Tasks, Baselines, and Evaluation Metrics.

A. Tasks

1) Task Difficulty Metrics: Each task difficulty metric quantifies the challenge of a test configuration from a specific perspective. In quadrotor navigation, a test is configured by the obstacles-laden scenario, start point, and end point. We utilize the topological guide path $\mathcal{T}$, which comprises interconnected individual waypoints [12], to establish the quantitative assessment. In FlightBench, we propose three main task difficulty metrics: Traversability Obstruction (TO), View Occlusion (VO), and Angle Over Length (AOL).

a) Traversability Obstruction: Traversability Obstruction (TO) measures the flight difficulty due to limited traversable space caused by obstacles. We use a sampling-based approach [9] to construct sphere-shaped flight corridors $\{B_0, \ldots, B_{N_T}\}$, where N_T is the number of spheres representing traversable space along path $\mathcal{T}$. Fig. 3(a) illustrates the primary notations for computing these corridors. The next sampling center $\mathbf{p}_{\text{hi}}$ is chosen from existing spheres $\{B_0, \ldots, B_{i-1}\}$ along $\mathcal{T}$. We sample K candidate centers from a 3D Gaussian distribution $\mathcal{D}$ around $\mathbf{p}_{\text{hi}}$. Each candidate sphere B_{cand} is defined by its center $\mathbf{p}_{\text{cand}}$ and radius $r_{\text{cand}} = ||\mathbf{p}_{\text{cand}} - \mathbf{n}_{\text{cand}}||_2 - r_d$, where $\mathbf{n}_{\text{cand}}$ is the nearest obstacle and r_d is the drone radius. For each B_{cand}, we compute S_{cand}:

$$S_{\text{cand}} = k_1 V_{\text{cand}} + k_2 V_{\text{inter}} - k_3(\mathbf{d}\cdot\mathbf{z}) - k_4||\mathbf{d} - (\mathbf{d}\cdot\mathbf{z})\mathbf{z}||_2 \quad (1)$$

where $k_1, k_2, k_3, k_4 \in \mathbb{R}^+$, V_{cand} is the volume of B_{cand}, V_{inter} is the overlap with B_{i-1}, $\mathbf{d} = \mathbf{p}_{\text{cand}} - \mathbf{p}_{\text{hi}}$, and $\mathbf{z}$ is the unit vector along $\mathbf{p}_{\text{hi}} - \mathbf{p}_i$. The sphere with the highest S_{cand} is selected as the next sphere. This process repeats until path $\mathcal{T}$ is fully covered. Occlusion challenges mainly occur in narrow spaces, so sphere radii $\{r_1, \ldots, r_{N_T}\}$ are sorted in ascending order. The traversability obstruction metric $\mathbb{T}$ is defined in Eq. (2), where R represents the sensing range.

$$\mathbb{T} = \frac{1}{N_T} \sum_{i=1}^{\lfloor N_T/2 \rfloor} \frac{R}{r_i}. \quad (2)$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

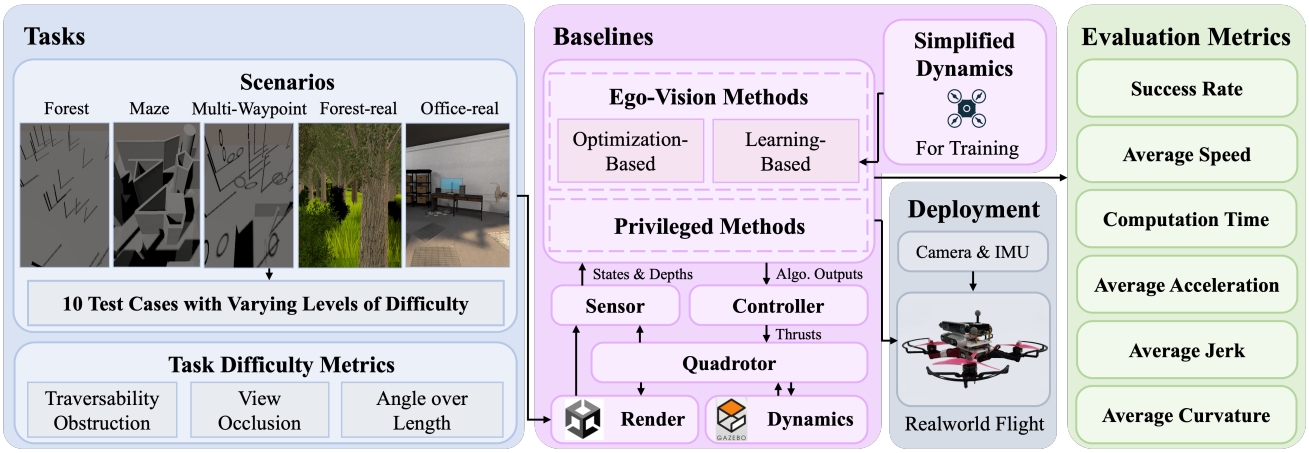
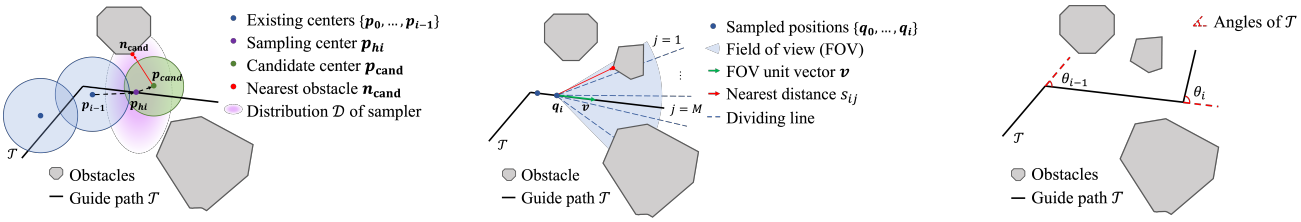


Figure 2: An overview of the FlightBench. FlightBench consists of three main components: (1) Tasks, featuring three scenarios categorized into eight difficulty levels. (2) Baselines, the core benchmarking platform supporting five ego-vision-based methods and two privileged methods. (3) Evaluation Metrics, offering a thorough suite of performance assessment metrics.



(a) Traversability Obstruction (b) View Occlusion (c) Angle over Length

Figure 3: Illustration of the task difficulty metrics

b) View Occlusion: In ego-vision-based navigation tasks, a narrow field of view (FOV) can limit the drone's perception, posing a challenge to the perception capabilities of various methods [31], [32]. We use the term view occlusion (VO) to describe the extent to which obstacles block the FOV in a given scenario. The more obstructed the view, the higher the view occlusion. As shown in Fig. 3(b), we sample drone position $\{\mathbf{q_i}\}$ and FOV unit vector $\{\mathbf{v_i}\}$ along $\mathcal{T}$ with $i \in \{1, \cdots, N_V\}$. For each sampled pair $\{\mathbf{q_i}, \mathbf{v_i}\}$, we divide FOV into M parts and calculate the distance s_{ij} between the nearest obstacle point and drone position q_i in each part j. The view occlusion $\mathbb{V}$ can be represented as Eq. (3), where m_j is a series of weights, which gives higher weight to obstacles closer to the center of the view.

$$\mathbb{V} = \frac{1}{N_V} \sum_{i=1}^{N_V} \sum_{j=1}^{M} m_j \frac{R}{s_{ij}}. \tag{3}$$

c) Angle Over Length: For a given scenario, frequent and violent turns in traversable paths pose challenges for the agility of planning. We employ the concept of Angle Over Length (AOL) denoted as $\mathbb{A}$ to quantify the sharpness of a path. The AOL $\mathbb{A}$ is defined by Eq. (4), where N_{AOL} signifies the number of angles depicted in Fig. 3(c), θ_i represents the i-th angle within the topological path $\mathcal{T}$, and L stands for the length of $\mathcal{T}$.

$$\mathbb{A} = \frac{1}{L} \sum_{i=1}^{N_{AOL}} \left(\exp\left(\frac{\theta_i}{\pi/6} \right) - 1 \right). \tag{4}$$

Table II: Task difficulty score of each test case.

Scenarios	Test Cases	TO	VO	AOL
Forest	1	0.76	0.30	7.64×10^{-4}
	2	0.92	0.44	1.62×10^{-3}
	3	0.90	0.60	5.68×10^{-3}
Maze	1	1.42	0.51	1.36×10^{-3}
	2	1.51	1.01	0.010
	3	1.54	1.39	0.61
MW	1	1.81	0.55	0.08
	2	1.58	1.13	0.94
Forest-real	1	1.76	1.47	0.02
Office-real	1	0.76	0.58	8.45×10^{-3}

2) Scenarios and Tests: As illustrated in Fig. 2, our benchmark incorporates specific tests based on five scenarios: `Forest`, `Maze`, `Multi-Waypoint`, `Forest-real`, and `Office-real`. These scenarios were chosen for their representativeness and frequent use in evaluating quadrotor navigation methods [8], [9]. Within these scenarios, we developed ten tests, each characterized by varying levels of difficulty. The task difficulty scores for each test are detailed in Tab. II.

The `Forest` scenario serves as the most common benchmark for quadrotor navigation. We differentiate task difficulty based on obstacle density and establish three tests, following the settings used by Agile [5]. In the `Forest` scenario, TO and VO metrics increase with higher tree density. AOL is particularly low due to sparsely spanned obstacles, making this

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Table III: Characteristics of the navigation methods for quadrotors. "RL" denotes reinforcement learning. "IL" represents imitation learning. "GM" and "EM" refer to Grid Mapping and ESDF Mapping, respectively. The control level indicates the part of the control stack used by the baseline.

	Method Type	Priv. Info.	Decision Horizon	Mapping	Planning	Control Stack: Traj.	Control Stack: Waypoint	Control Stack: Motion Cmd
SBMT	Samp.-based	✓	Global	Planning Module				MPC
LMT	RL	✓	Local	Policy Network				
Fast-Planner	Opti.-based	✗	Global	GM+EM	Planning Module		MPC	
EGO-Planner	Opti.-based	✗	Local	GM	Planning Module		MPC	
TGK-Planner	Opti.-based	✗	Global	GM	Planning Module		MPC	
Agile	IL	✗	Local	Policy Network				MPC
LPA	RL+IL	✗	Local	Policy Network				
NPE	RL	✗	Local	Policy Network				MPC

scenario suitable for high-speed flights [5], [9].

The `Maze` scenario consists of walls and boxes, creating consecutive sharp turns and narrow gaps. Quadrotors must navigate these confined spaces while maintaining flight stability and perception awareness [33]. We devise three tests with varying lengths and turn complexities for `Maze`, resulting in discriminating difficulty levels for VO and AOL.

The `Multi-Waypoint` (`MW`) scenario involves flying through multiple waypoints at different heights sequentially [34]. This scenario also includes boxes and walls as obstacles. The `MW` scenario is relatively challenging, featuring the highest TO in test 1 and the highest AOL in test 2.

We create two more realistic scenarios: `Forest-real` and `Office-real`. The `Forest-real` scenario features a dense arrangement of textured trees and shrubs on a grassy field. The structure of the obstacles is similar to `Forest` but with higher density. The `Office-real` has a structure similar structure to *Maze*, incorporating shelves, desks, and chairs to form narrow gaps and sharp turns. Both scenarios also feature realistic lighting conditions, such as outdoor natural lighting for `Forest-real` and indoor overhead lighting for `Office-real`.

B. Baselines

We evaluate representative vision-based navigation methods in FlightBench, including three learning-based approaches (**Agile** [5], **NPE** [35], **LPA** [7]) and three optimization-based approaches (**Fast-Planner** [10], **TGK-Planner** [36], **EGO-Planner** [37]). To establish an upper bound, we also integrate two privileged methods with access to environmental information (**SBMT** [12], **LMT** [38]). We offer two perception module options: ground-truth depth and SGM depth [39]. These methods produce varying outputs, which are then processed by the same control stack (see Fig. 4) with an MPC controller [40]. The characteristics of each method are detailed in Tab. III. For code, parameters, and implementation details, please visit our website[1].

C. Evaluation Metrics

We represent quadrotor states as a tuple $(\mathbf{x}(t), \mathbf{v}(t), \mathbf{a}(t), \mathbf{j}(t))$, where t denotes time, $\mathbf{x}(t)$ denotes the position, and $\mathbf{v}(t) = \dot{\mathbf{x}}(t)$, $\mathbf{a}(t) = \dot{\mathbf{v}}(t)$, $\mathbf{j}(t) = \dot{\mathbf{a}}(t)$ are the velocity, acceleration, and jerk in the world frame, respectively. T denotes the time taken to fly from the starting point to the end point.

[1] https://thu-uav.github.io/FlightBench

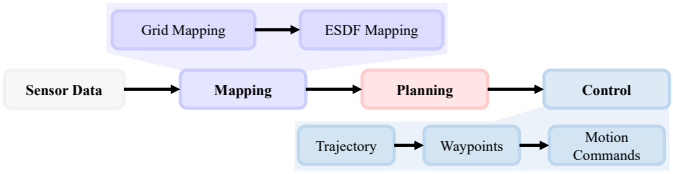


Figure 4: A generic processing pipeline for hierarchical navigation systems on quadrotors.

First, we integrate three widely used metrics [5], [9] into FlightBench. **Success rate** measures if the quadrotor reaches the goal within a 1.5 m radius without crashing. **Average speed**, defined as $\frac{1}{T}\int_0^T ||\mathbf{v}(t)||_2 \, \mathrm{d}t$, reflects the achieved agility. **Computation time** evaluates real-time performance as the sum of processing times for mapping, planning, and control. Additionally, we introduce **average acceleration** and **average jerk** [10], [37], defined as $\bar{\mathbf{a}} = \frac{1}{L}\int_0^T ||\mathbf{a}(t)||_2^2 \, \mathrm{d}t$ and $\bar{\mathbf{j}} = \frac{1}{L}\int_0^T ||\mathbf{j}(t)||_2^2 \, \mathrm{d}t$, respectively. Average acceleration indicates energy consumption, while average jerk measures flight smoothness [41]. These metrics, though not commonly used in evaluations, are crucial for assessing practicability and safety in real-world applications. However, average acceleration and jerk only capture the dynamic characteristics of a flight. For instance, higher flight speeds along the same trajectory result in greater average acceleration and jerk. To assess the static quality of trajectories, we propose **average curvature**, inspired by Bench-MR [24]. Curvature is calculated as $\kappa(t) = \frac{|\mathbf{v}(t)\times\mathbf{a}(t)|}{|\mathbf{v}(t)|^3}$, and average curvature is defined as $\bar{\kappa} = \frac{1}{L}\int_0^T \kappa(t)\mathbf{v}(t) \, \mathrm{d}t$.

Together, these six metrics provide a comprehensive comparison of learning-based algorithms against optimization-based methods for ego-vision-based quadrotor navigation. Our extensive experiments will demonstrate that while learning-based methods excel in certain metrics, they also have shortcomings in others.

IV. Experiments

Using FlightBench, we carried out extensive experiments to address the following research questions:

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Table IV: Performance evaluation of different methods under tests with the highest AOL within each scenario. The highest performing values for each metric are highlighted in bold, with the second highest underlined.

Scen.	Metric	Privileged		Optimization-based			Learning-based		
		SBMT	LMT	TGK	Fast	EGO	Agile	LPA	NPE
Forest	Success Rate ↑	0.80	**1.00**	0.90	0.90	**1.00**	0.90	**1.00**	0.90
	Avg. Spd. (ms^{-1}) ↑	**15.25**	11.84	2.30	2.47	2.49	3.058	8.96	2.03
	Avg. Curv. (m^{-1}) ↓	**0.06**	0.07	0.08	**0.06**	0.08	0.37	0.08	0.13
	Avg. Acc. (ms^{-3}) ↓	28.39	10.29	0.25	**0.19**	0.83	4.93	9.96	0.98
	Avg. Jerk (ms^{-5}) ↓	4.27×10^3	8.14×10^3	**1.03**	3.97	58.39	937.02	1.14×10^4	1.82×10^3
Maze	Success Rate ↑	0.60	**0.9**	0.50	0.60	0.20	0.50	0.30	0.50
	Avg. Spd. (ms^{-1}) ↑	8.73	**9.62**	1.85	1.99	2.19	3.00	8.35	1.98
	Avg. Curv. (m^{-1}) ↓	0.31	**0.13**	0.17	0.23	0.33	0.68	0.21	0.39
	Avg. Acc. (ms^{-3}) ↓	60.73	26.26	**0.50**	0.79	1.91	15.45	37.30	2.12
	Avg. Jerk (ms^{-5}) ↓	6.60×10^3	4.64×10^3	**6.74**	9.62	80.54	2.15×10^3	4.64×10^3	2.38×10^3
MW	Success Rate ↑	0.70	**0.90**	0.40	0.80	0.50	0.60	0.50	0.40
	Avg. Spd. (ms^{-1}) ↑	5.59	**6.88**	1.48	1.73	2.13	3.05	6.72	1.50
	Avg. Curv. (m^{-1}) ↓	0.47	0.30	0.46	0.32	0.62	0.67	**0.26**	0.71
	Avg. Acc. (ms^{-3}) ↓	80.95	31.23	1.07	**0.97**	5.06	16.86	36.77	2.84
	Avg. Jerk (ms^{-5}) ↓	9.76×10^3	1.66×10^4	25.52	**22.72**	155.83	2.07×10^3	6.19×10^3	2.53×10^3
Forest-real	Success Rate ↑	0.90	**1.00**	0.70	0.90	0.90	**1.00**	0.90	0.90
	Avg. Spd. (ms^{-1}) ↑	**14.11**	11.20	2.40	2.42	2.44	3.06	9.70	2.16
	Avg. Curv. (m^{-1}) ↓	**0.04**	0.10	0.09	0.09	0.26	0.24	0.13	0.15
	Avg. Acc. (ms^{-3}) ↓	28.56	10.90	**0.18**	0.29	1.82	4.87	16.49	1.92
	Avg. Jerk (ms^{-5}) ↓	4.37×10^3	8.20×10^3	3.09	**2.25**	94.70	862.3	6.72×10^3	1.89×10^3
Office-real	Success Rate ↑	0.80	**0.90**	0.70	0.70	0.50	0.50	0.80	0.60
	Avg. Spd. (ms^{-1}) ↑	9.54	**10.45**	1.86	2.01	2.51	3.01	9.32	1.95
	Avg. Curv. (m^{-1}) ↓	**0.06**	0.09	0.19	0.18	0.25	0.33	0.09	0.36
	Avg. Acc. (ms^{-3}) ↓	39.08	11.24	**0.49**	0.59	3.10	8.41	20.64	3.38
	Avg. Jerk (ms^{-5}) ↓	5.40×10^3	4.52×10^3	**7.28**	9.16	285.91	1.03×10^3	7.11×10^3	1.82×10^3

- What are the main advantages and limitations of learning-based methods compared to optimization-based methods?
- How does navigation performance vary across different scenario settings?
- How much does the system latency introduced by the practical evaluation environment affect performance?
- Is there a consistent correlation between performance in simulation and those demonstrated in real-world?

A. Setup

For simulating quadrotors, we use Flightmare [42] with Gazebo [43] as its dynamic engine. To mimic real-world conditions, we develop a simulated quadrotor model equipped with an IMU sensor and a depth camera, calibrated with real flight data. To simulate real-world communication delays, all data transmission in the simulation uses ROS [44]. All simulations are conducted on a desktop PC with an Intel Core i9-11900K processor and an Nvidia 3090 GPU. Each evaluation metric is averaged over ten independent runs.

We validate the sim-to-real capabilities of our benchmark through real-world experiments on the Q250 platform. Q250 quadrotor integrates an NVIDIA Orin NX for computation and utilizes MAVLink to interface with the PX4 flight controller for low-level control. The implementation details and full experimental results are available on our website1.

B. Benchmarking Flight Performance

1) Flight Quality: To systematically assess the strengths and weaknesses of various methods on ego-vision navigation, we conduct evaluations across all tests within five distinct scenarios. Tab. IV displays the results for tests with the highest AOL in each scenario. We evaluate these methods using our proposed evaluation metrics, and computation time will be addressed separately in a subsequent discussion. In these experiments, we standardize the expected maximum speed at $3m/s$ for a fair comparison. Exceptions are SBMT, LMT, and LPA, whose flight speeds cannot be manually controlled.

As shown in Tab. IV, the privileged methods, with a global awareness of obstacles, set the upper bound for motion performance in terms of average speed and success rate. In contrast, the success rate of ego-vision methods in the Maze and MW scenarios is generally below 0.6, indicating that our benchmark remains challenging for ego-vision methods, especially at the perception level.

Learning-based methods, known for their aggressive maneuvering, tend to fly less smoothly and consume more energy. They also experience more crashes in areas with large corners, as seen in the Maze and MW scenarios. When performing a large-angle turn, an aggressive policy is more likely to cause the quadrotor to lose balance and crash. Optimization-based methods are still competitive or even superior to current learning-based approaches, particularly in terms of minimizing energy costs. By contrasting the more effective Fast-Planner with the more severely impaired TGK-Planner and EGO-Planner, we find that global trajectory smoothing and enhancing the speed of replanning are crucial for improving success rates in complex scenarios.

Remark. *Learning-based methods tend to execute aggressive*

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Table V: Computation time of different baselines. T_{map}, T_{plan}, T_{ctrl}, T_{tot} stands for mapping time, planning time, control time, and total time, respectively.

		SBMT	LMT	TGK	Fast	EGO	Agile	LPA	NPE
Desktop	T_{tot} (ms)	3.189×10^5	2.773	11.960	8.196	3.470	5.573	1.395	4.942
	T_{map} (ms)	-	1.607	3.964	7.038	2.956	0.338	0.399	2.107
	T_{plan} (ms)	2.589×10^5	1.167	7.994	1.155	0.510	5.115	0.995	2.833
	T_{ctrl} (ms)	-	-	0.002	0.003	0.003	0.119	-	0.003
Onboard	T_{tot} (ms)	-	13.213	37.646	39.177	24.946	27.458	12.293	17.144
	T_{map} (ms)	-	2.768	27.420	36.853	24.020	1.175	4.313	4.532
	T_{plan} (ms)	-	10.445	10.211	2.310	0.910	26.283	7.980	12.598
	T_{ctrl} (ms)	-	-	0.015	0.014	0.016	-	-	0.014

and fluctuating maneuvers, yet they struggle with instability in scenarios with high challenges related to VO and AOL.

2) Impact of Flight Speed and Computation Time: This section evaluates all methods in Test Case 2 of the Forest scenario. As illustrated in Fig. 5, we examine the success rate of each method at different average speeds, excluding three methods where speed is non-adjustable. Learning-based methods exhibit agile obstacle avoidance and operate near dynamic limits due to their end-to-end architecture. In contrast, optimization-based methods struggle at high speeds, as their hierarchical pipeline latency often causes the quadrotor to overshoot obstacles before a new path is generated. Privileged methods achieve higher success rates and faster flights, highlighting the potential for improving current ego-vision-based approaches.

We also analyze computation times for various planning methods on both desktop and onboard platforms, breaking down the results into mapping, planning, and control stages (see Tab. V, Fig. 4). For learning-based methods, mapping time includes image-to-tensor conversion and state pre-processing. Because of the compact neural network structure, learning-based methods generally require less computation time. Among optimization-based approaches, mapping is the most time-intensive stage, particularly on the onboard platform. Fast-Planner has the longest mapping time due to ESDF map construction. Similar to the conclusions of [5], computation time directly impacts maximum flight speed. Lighter architectures, which allow higher replan frequencies, improve obstacle reaction and enable faster flights.

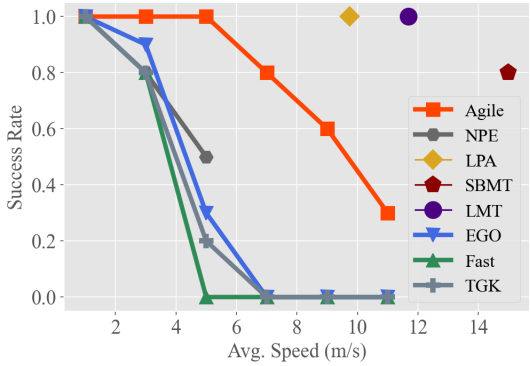


Figure 5: Success rate with different flight speeds.

Remark. *Learning-based methods, with their compact neural network architectures, excel in high-speed flight by achieving superior performance and faster planning. Well-crafted optimization-based methods can achieve comparable planning speed.*

C. Analyses on Effectiveness of Different Metrics

To demonstrate how various task difficulties influence different aspects of flight performance, we calculate the correlation coefficients between six performance metrics and three difficulty metrics for each method across multiple tests. The value at the intersection of the horizontal and vertical axes represents the absolute value of the correlation coefficient between the two metrics. A higher value indicates a stronger correlation. Fig. 6 presents the average correlations for privileged and ego-vision-based methods, separately evaluating the impacts on agility and partial observation.

The results for privileged methods shown in Fig. 6(a) indicate that AOL and TO have a significant impact on the baseline's motion performance. The correlation coefficients between AOL and average curvature, velocity and acceleration are all above 0.85, indicating that AOL describes the sharpness of the trajectory well. More specifically, high AOL results in high curvature and acceleration of the flight trajectory, as well as lower average speed. TO, indicating task narrowness, is a crucial determinant of flight success rates. In contrast to privileged methods where global information is accessible, as shown in Fig. 6(b), ego-vision-based methods primarily struggle with partial perception. Field-of-view occlusions and turns challenge real-time environmental awareness, making VO highly correlated with success rates.

Remark. *High VO and AOL significantly challenge learning-based methods, as these factors heavily impact the ego-vision-based method's ability to handle partial observations and sudden reactions.*

D. Impact of Latency on Learning-based Methods

Beyond the algorithmic factors previously discussed, learning-based methods face considerable challenges transitioning from training simulations to real-world applications. Latency significantly impacts sim-to-real transfer, particularly when simplified robot dynamics are used to enhance high-throughput RL training. We evaluate the impact of latency in a ROS-based environment, where asynchronous node communication introduces approximately $45ms$ of delay.

Tab. VII details the performance of RL-based methods under the most challenging test, i.e., test 2 of the Multi-Waypoint

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Table VI: HITL experiment results

Scene	TO	VO	AOL	EGO Avg. Spd. ↑ (ms^{-1})	EGO Avg. Curv. ↓ (m^{-1})	EGO Avg. Acc. ↓ (ms^{-3})	EGO Avg. Jerk ↓ (ms^{-5})	Agile Avg. Spd. ↑ (ms^{-1})	Agile Avg. Curv. ↓ (m^{-1})	Agile Avg. Acc. ↓ (ms^{-3})	Agile Avg. Jerk ↓ (ms^{-5})
1	0.98	0.57	3×10^{-4}	1.39	0.54	1.40	4.85×10^{2}	1.45	0.65	2.11	3.99×10^{3}
2	1.81	1.82	0.027	1.32	0.92	1.68	5.64×10^{2}	1.47	1.61	5.62	6.64×10^{3}
3	2.07	1.85	0.040	1.39	1.55	5.20	7.70×10^{2}	1.51	1.62	7.82	8.36×10^{3}

(a) Correlation heatmap for privileged methods

(b) Correlation heatmap for ego-vision methods

Figure 6: Correlation coefficients between three difficulty metrics and six evaluation metrics.

Table VII: Performance of RL-based methods under test 2 of Multi-Waypoint scenario.

	Train w/ latency		Train w/o latency	
	Success Rate ↑	Progress	Success Rate ↑	Progress
LMT	**0.9**	**0.96**	0.3	0.57
LPA	**0.5**	**0.73**	0.0	0.32

scenario, with the highest VO, AOL, and the second-highest TO. To further evaluate the baseline's performance at low success rates, we introduce *Progress*, a metric ranging from 0 to 1 that reflects the proportion of the trajectory completed before a collision occurs. The "train w/ latency" column displays results from training with simulated and randomized latencies between $25ms$ to $50ms$, whereas the "train w/o latency" column serves as a control group. "Train w/ latency" significantly improves both success rate and progress by more than 50% for two methods, emphasizing the importance of incorporating randomized latency in more realistic simulation environments and real-world deployments.

Remark. *Integrating latency randomization into the training of RL-based methods is essential to enhance real-world applicability.*

E. Real-world Flight

We selected two representative methods for real-world experiments: EGO, from the optimization-based category, and Agile, from the learning-based category. First, we conducted full-pipeline experiments in cluttered environments to evaluate the deployment capabilities of our benchmark (Fig. 1). During these flights, all state estimation, sensing, computation, and control were executed entirely onboard.

Next, we performed Hardware-In-The-Loop (HITL) experiments, where simulated visual perception guided the drone's flight, allowing us to quantitatively validate the consistency between simulation and real-world performance. Tab. VI presents flight results across three real-world test cases. In more challenging scenarios (characterized by higher TO, VO, and AOL), the quadrotor exhibited increased curvature, acceleration, and jerk. Furthermore, in these scenarios, Agile demonstrated greater curvature, acceleration, and jerk compared to EGO, leading to less smooth flight. These findings confirm that the performance trends observed in simulation closely align with real-world outcomes. For further details, please refer to our video.

Remark. *Our benchmark supports full-pipeline deployment, ensuring that flight performance trends remain consistent and order-preserving across simulation and real-world conditions.*

V. Conclusion

In this paper, we introduce FlightBench, an open-source benchmark for evaluating both learning-based and optimization-based approaches in ego-vision quadrotor navigation. By incorporating three task difficulty metrics, FlightBench enables quantitative comparisons across diverse scenarios and test cases. Our experiments demonstrate that learning-based methods excel in high-speed flight and inference efficiency but require improvement in handling complexity and robustness. In contrast, optimization-based methods, while slower, consistently generate smooth and flyable trajectories. Correlation analysis confirms that our task difficulty metrics effectively capture performance variations across different test cases. Additionally, real-world experiments validate the consistency between simulation and actual quadrotor performance. We hope FlightBench will serve as a catalyst for advancing learning-based quadrotor navigation.

References

[1] J. Xiao, P. Pisutsin, and M. Feroskhan, "Collaborative target search with a visual drone swarm: An adaptive curriculum embedded multistage reinforcement learning approach," *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

[2] X. Xiao, J. Biswas, and P. Stone, "Learning inverse kinodynamics for accurate high-speed off-road navigation on unstructured terrain," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 6054–6060, 2021.

[3] K. Stachowicz, D. Shah, A. Bhorkar, I. Kostrikov, and S. Levine, "Fastrlap: A system for learning high-speed driving via deep rl and autonomous practicing," in *Conference on Robot Learning*. PMLR, 2023, pp. 3100–3111.

[4] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on robot learning*. PMLR, 2023, pp. 403–415.

[5] 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. eabg5810, 2021.

[6] J. Xiao, R. Zhang, Y. Zhang, and M. Feroskhan, "Vision-based learning for drones: A survey," 2024.

[7] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza, "Learning perception-aware agile flight in cluttered environments," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1989–1995.

[8] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[9] Y. Ren, F. Zhu, W. Liu, Z. Wang, Y. Lin, F. Gao, and F. Zhang, "Bubble planner: Planning high-speed smooth quadrotor trajectories using receding corridors," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6332–6339.

[10] B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen, "Robust and efficient quadrotor trajectory generation for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3529–3536, 2019.

[11] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE transactions on Robotics and Automation*, vol. 13, no. 6, pp. 814–822, 1997.

[12] R. Penicka and D. Scaramuzza, "Minimum-time quadrotor waypoint flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5719–5726, 2022.

[13] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2012.

[14] H. Ye, N. Pan, Q. Wang, C. Xu, and F. Gao, "Efficient sampling-based multirotors kinodynamic planning with fast regional optimization and post refining," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3356–3363.

[15] J. Sfeir, M. Saad, and H. Saliah-Hassane, "An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment," in *2011 IEEE international symposium on robotic and sensors environments (ROSE)*. IEEE, 2011, pp. 208–213.

[16] F. Schilling, J. Lecoeur, F. Schiano, and D. Floreano, "Learning vision-based flight in drone swarms by imitation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4523–4530, 2019.

[17] S. Liu, M. Xu, P. Huang, X. Zhang, Y. Liu, K. Oguchi, and D. Zhao, "Continual vision-based reinforcement learning with group symmetries," in *Proceedings of The 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, J. Tan, M. Toussaint, and K. Darvish, Eds., vol. 229. PMLR, 06–09 Nov 2023, pp. 222–240. [Online]. Available: https://proceedings.mlr.press/v229/liu23a.html

[18] J. Heeg, Y. Song, and D. Scaramuzza, "Learning quadrotor control from visual features using differentiable simulation," *arXiv preprint arXiv:2410.15979*, 2024.

[19] M. Xi, H. Dai, J. He, W. Li, J. Wen, S. Xiao, and J. Yang, "A lightweight reinforcement-learning-based real-time path-planning method for unmanned aerial vehicles," *IEEE Internet of Things Journal*, vol. 11, no. 12, pp. 21 061–21 071, 2024.

[20] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov, "Learning to explore using active neural slam," *arXiv preprint arXiv:2004.05155*, 2020.

[21] J. Xing, A. Romero, L. Bauersfeld, and D. Scaramuzza, "Bootstrapping reinforcement learning with imitation for vision-based agile flight," *arXiv preprint arXiv:2403.12203*, 2024.

[22] I. Geles, L. Bauersfeld, A. Romero, J. Xing, and D. Scaramuzza, "Demonstrating agile flight from pixels without state estimation," *Robotics Science and Systems online Proceedings*, no. 20, p. online, 2024.

[23] J. Wen, X. Zhang, Q. Bi, Z. Pan, Y. Feng, J. Yuan, and Y. Fang, "MRPB 1.0: A unified benchmark for the evaluation of mobile robot local planning approaches," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 8238–8244.

[24] E. Heiden, L. Palmieri, L. Bruns, K. O. Arras, G. S. Sukhatme, and S. Koenig, "Bench-MR: A motion planning benchmark for wheeled mobile robots," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4536–4543, 2021.

[25] A.-I. Toma, H.-Y. Hsueh, H. A. Jaafar, R. Murai, P. H. Kelly, and S. Saeedi, "Pathbench: A benchmarking platform for classical and learned path planning algorithms," in *2021 18th Conference on Robots and Vision (CRV)*. IEEE, 2021, pp. 79–86.

[26] F. Xia, W. B. Shen, C. Li, P. Kasimbeg, M. E. Tchapmi, A. Toshev, R. Martín-Martín, and S. Savarese, "Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 713–720, 2020.

[27] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[28] Z. Xu, B. Liu, X. Xiao, A. Nair, and P. Stone, "Benchmarking reinforcement learning techniques for autonomous navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9224–9230.

[29] L. Rocha and K. Vivaldini, "Plannie: A benchmark framework for autonomous robots path planning algorithms integrated to simulated and real environments," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2022, pp. 402–411.

[30] Y. S. Shao, Y. Wu, L. Jarin-Lipschitz, P. Chaudhari, and V. Kumar, "Design and evaluation of motion planners for quadrotors in environments with varying complexities," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 10 033–10 039.

[31] J. Tordesillas and J. P. How, "PANTHER: Perception-aware trajectory planner in dynamic environments," *IEEE Access*, vol. 10, pp. 22 662–22 677, 2022.

[32] X. Chen, Y. Zhang, B. Zhou, and S. Shen, "APACE: Agile and perception-aware trajectory generation for quadrotor flights," *arXiv preprint arXiv:2403.08365*, 2024.

[33] J. Park, Y. Lee, I. Jang, and H. J. Kim, "Dlsc: Distributed multi-agent trajectory planning in maze-like dynamic environments using linear safe corridor," *IEEE Transactions on Robotics*, 2023.

[34] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing with deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1205–1212.

[35] Y. Zhang, C. Yan, J. Xiao, and M. Feroskhan, "NPE-DRL: Enhancing perception constrained obstacle avoidance with nonexpert policy guided reinforcement learning," *IEEE Transactions on Artificial Intelligence*, vol. 6, no. 1, pp. 184–198, 2025.

[36] H. Ye, X. Zhou, Z. Wang, C. Xu, J. Chu, and F. Gao, "TGK-Planner: An efficient topology guided kinodynamic planner for autonomous quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 494–501, 2021.

[37] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-Planner: An esdf-free gradient-based local planner for quadrotors," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 478–485, 2021.

[38] R. Penicka, Y. Song, E. Kaufmann, and D. Scaramuzza, "Learning minimum-time flight in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7209–7216, 2022.

[39] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2007.

[40] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[41] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3259–3278, 2022.

[42] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Proceedings of the 2020 Conference on Robot Learning*, 2021, pp. 1147–1157.

[43] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[44] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.