CU-Fundamental: A Comprehensive Simulation Framework for Autonomous MAV

Jialiang Wang[1]*, Yijun Huang[1], Yizhou Chen[2], Zongzhou Wu[1], Qigeng Duan[1], Zhiyu Zhou[3], Yanqi Zhao[1], Zhipeng Lin[1], Jiwen Xu[1], Jerry Tang[4], Xi Chen[1], Zhi Gao[3] and Ben M. Chen[1]
[1]*Department of Mechanical and Automation Engineering*, The Chinese University of Hong Kong, Hong Kong SAR
[2]*School of Computing and Data Science*, Hong Kong University, Hong Kong SAR
[3]*School of Remote Sensing and Information Engineering*, Wuhan University, China
[4]*Hong Kong Centre For Logistics Robotics*, Hong Kong University, Hong Kong SAR

ABSTRACT

In recent years, the development of Unmanned Aerial Vehicles (UAVs) has seen significant advancements, driven by their successful applications in various fields. Autonomous Micro Aerial Vehicles (MAVs) are particularly gaining attention due to their agility and potential for complex tasks. However, testing autonomous algorithms on real MAVs is both costly and risky, making simulation platforms essential for safe and efficient development. This paper introduces CU-Fundamental, a comprehensive simulation framework for autonomous MAVs, which supports multiple levels of simulation, including numerical simulation, Software-in-the-Loop (SITL), and Hardware-in-the-Loop (HITL). The framework integrates essential algorithms facilitating autonomous flight, including pose estimation, map building, path planning, motion planning, and low-level control. CU-Fundamental is designed to be modular and flexible, allowing for easy integration of new algorithms and components. It employs Robot Operating System (ROS) for internal communication, a web-based Ground Control Station (GCS) for advanced user interaction, PX4 firmware for flight control, and Gazebo for realistic physical simulation. By providing a flexible and modular architecture, CU-Fundamental aims to accelerate the development of robust autonomous MAV systems while minimizing risks and costs associated with real-world testing.

1 INTRODUCTION

In recent years, the development of UAVs has witnessed remarkable progress, with their applications expanding across a wide range of fields, including search and rescue, environmental monitoring, last-mile delivery, building inspection and autonomous racing [1]. Particularly, Autonomous MAVs have garnered significant attention due to their agility and potential to perform complex tasks in confined spaces such as tunnels and mines [2]. These advancements have not only enhanced the efficiency of various operations but also opened up new possibilities for innovation in the field of aerial robotics.

However, the increasing complexity of tasks that UAVs are expected to perform necessitates the development and integration of sophisticated algorithms. These algorithms encompass a wide range of functionalities, including low-level control laws for precise maneuvering, robust pose estimation for autonomous flight, accurate perception for environmental awareness, and high-level planning algorithms for optimal path generation. Validating the effectiveness of these algorithms could be costly and sometimes rather dangerous on real UAVs [3]. Therefore, simulation platforms have emerged as a crucial solution, providing a safe and efficient means to test and refine autonomous algorithms before deployment on actual platforms.

Several simulation programs have been developed to address the needs of UAV research and development, among which AirSim and PX4 Simulink are two popular platforms that have proved to be effective in various scenarios. Developed by Microsoft, AirSim is a high-fidelity simulation platform that provides realistic rendering of environments and sensor data [4]. On the other hand, PX4 Simulink focuses on the development and testing of low-level control algorithms for UAVs [5]. It provides a comprehensive environment for designing, simulating, and tuning control systems, leveraging the powerful capabilities of Simulink for model-based design and verification.

While these platforms have been instrumental in advancing UAV research, they also have limitations that need to be addressed. AirSim, for instance, excels in providing realistic sensor data, making it suitable for testing high-level perception algorithms. However, the platform is not easily extensible for secondary development, partially due to the heavy Unreal Engine it is built upon. It also has limited support for Linux, which restricts its usability makes integration with existing algorithms very challenging. Similarly, PX4 Simulink enables researchers to develop robust and reliable control algorithms through its high-performance numerical simulation. Yet, when it comes to solving practical problems that involve

*Email address: jialiang.wang@link.cuhk.edu.hk

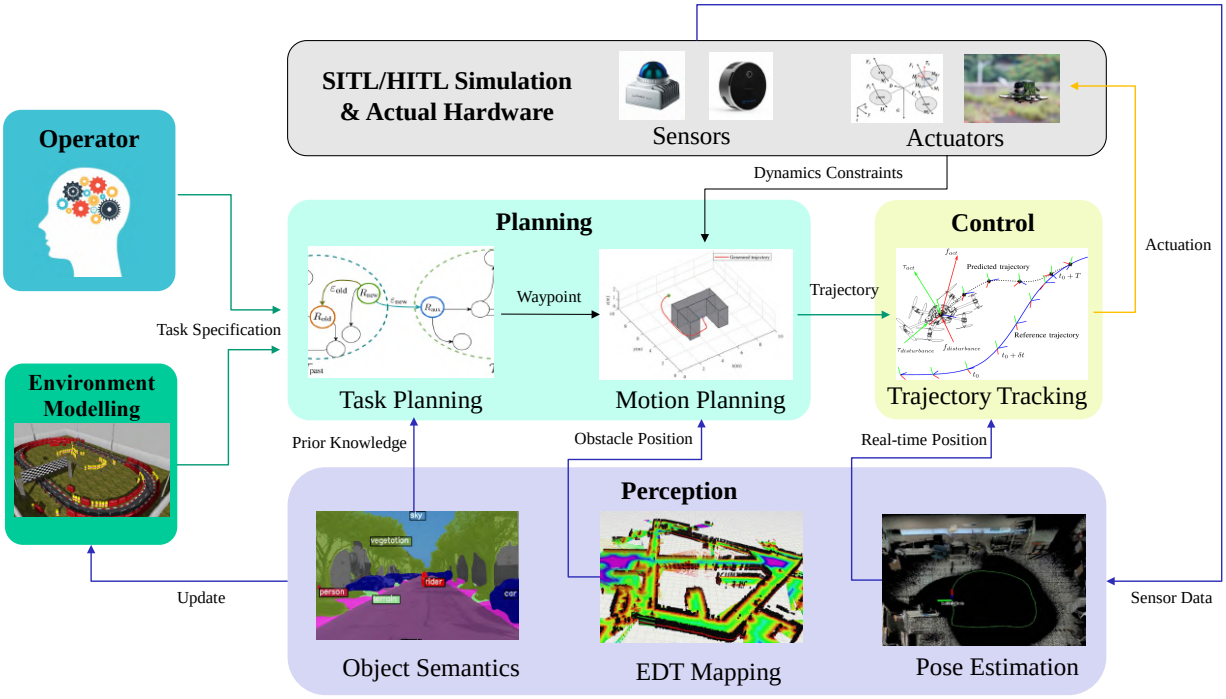


Figure 1: The architecture of CU-Fundamental. The system has built-in modules for planning, control and perception, with standard interfaces facilitating smooth communication between modules.

higher-level functionalities such as Simultaneous Localization and Mapping (SLAM) and planning, Simulink is not suitable, if not impossible, to use due to its limited capability to simulate sensor data and the complexity of integrating other algorithms into Matlab.

To test both low-level and high-level algorithms for autonomous MAVs, we introduce CU-Fundamental, a comprehensive simulation framework for autonomous MAVs. CU-Fundamental is designed to support multiple levels of simulation, including numerical simulation, SITL, and HITL, following the implementation of XTDrone [6]. It employs ROS for internal communication, a GCS for advanced user interaction, PX4 firmware for flight control, and Gazebo for realistic physical simulation. The framework has built-in support for essential algorithms like SLAM, path planning, motion planning, and control. Moreover, CU-Fundamental is designed to be modular and flexible, providing standard interfaces for integration of new algorithms and fine-tuning of algorithm parameters. This framework facilitates the fundamental workflow of developing and testing autonomous MAVs: the upper-level algorithms are first tested (in SITL mode) while taking aerodynamics (in numeric simulation) into account, then the same algorithms are deployed to the embedded flight control board and onboard computer for HITL testing. If all tests are successful, the algorithms can be deployed to real MAVs for field-testing.

2 System Architecture

An overview of the architecture of CU-Fundamental is shown in Figure 1.

2.1 *Dynamic Modelling*

Upon finishing the design of new MAVs, the first step is to model the dynamics of the MAV. Taking advantage of Gazebo's physics engine [7], CU-Fundamental provides a dynamic model of quadrotor MAVs as shown in Figure 2.

The forces and moments on an MAV comprise the thrust force $f_i \in \mathbb{R}$ and corresponding moment $M_i \in \mathbb{R}^3$ generated by i-th rotor, the aerodynamic drag force $D \in \mathbb{R}^3$ and the gravitational force $G \in \mathbb{R}^3$. Let the distance from the center of mass to the center of each rotor be $d \in \mathbb{R}$, the total thrust f and moment $M = [M_x, M_y, M_z]^T$ generated in the body frame can be expressed as:

$$\begin{bmatrix} f \\ M_x \\ M_y \\ M_z \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -d & 0 & d \\ d & 0 & -d & 0 \\ -c_M & c_M & -c_M & c_M \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{1}$$

where c_M is the moment coefficient, which is a constant determined by the geometry of the MAV. More detailed modelling of this coefficient can be found in pp. 631-639 of [8].

Applying Newton's second law, the equations of motion

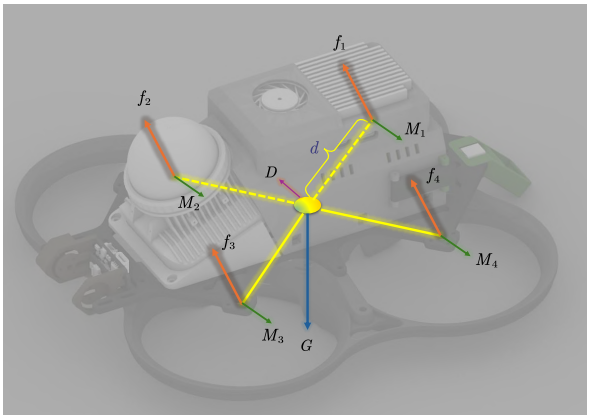


Figure 2: The dynamic model of quadrotor MAVs in CU-Fundamental.

of the quadrotor MAV can be expressed as:

$$\dot{x} = v, \tag{2}$$

$$m\dot{v} = G + D + f, \tag{3}$$

$$\dot{R} = R\hat{\Omega}, \tag{4}$$

$$J\dot{\Omega} + \Omega \times J\Omega = M, \tag{5}$$

where $R \in SO(3)$ is the rotation matrix from body frame to world frame, $\Omega \in \mathbb{R}^3$ is the angular velocity in body frame, $J \in \mathbb{R}^{3\times3}$ is the inertia matrix of the MAV, and $\hat{\Omega}$ is the skew-symmetric matrix of Ω.

2.2 *Sensor Simulation*

Sensor models are essential for the perceptual capability of autonomous MAVs. To cater for the versatile needs of various missions, CU-Fundamental provides a set of different sensors including IMU, GPS, LiDAR, and cameras.

IMU, magnetometer, barometer and GPS: Following the design of [6], the observation values of all these sensors are disturbed by an error model which consists of a random walk, a bias and a white noise. These parameters are all configurable.

Camera: CU-Fundamental supports general RGB and depth cameras provided by Gazebo plugins. The tunable parameters include camera resolution, field of view (FOV), intrinsic parameters, distortion coefficients, among others. Alternatively, users may opt to use other open-source camera simulation plugins such as the Intel Realsense simulation by [9], which essentially combines multiple RGB and depth cameras into a single plugin.

LiDAR: CU-Fundamental supports various types of LiDAR sensors, including 2D and 3D ones. 2D LiDAR is implemented as an instance of Gazebo ray sensor, which emits laser beams controlled by a certain range, FOV, resolution, number of samples and Gaussian noise. Two types of 3D LiDAR are supported: the first is mechanical LiDAR, which is implemented as a vertical array of 2D laser beams; the second is solid-state LiDAR, which is based on the working principle of Livox LiDARs [10].

2.3 *Supporting Algorithms*

Control: CU-Fundamental allows users to implement both outer-loop and inner-loop control algorithms using the functionalities of Mavros. By default, CU-Fundamental uses PX4 for attitude, position and velocity control. Attitude control is achieved using traditional PID control law, while position and velocity control are implemented as cascaded PID controllers on top of the attitude controller.

Moreover, CU-Fundamental offers a more sophisticated non-linear geometric controller [11, 12] as an in-place alternative to the default PID controller. Similarly, three flight modes, namely attitude controlled mode, position controlled mode and velocity controlled mode, are supported and can be switched between each other. Different from the default PID controllers, the non-linear geometric controller guarantees almost global tracking features on $SO(3)$ and enables more aggressive maneuvers with faster response time.

Perception: Perception is a board term that refers to the process of acquiring and interpreting sensory data to gain actionable insights about the environment. Among various perceptual algorithms, SLAM is one of the most crucial components because of its capability to provide accurate pose estimation and environmental mapping for both upper-level planning algorithms and low-level control. Depending on the sensor types, CU-Fundamental has built-in support for various visual and LiDAR SLAM algorithms, providing convenient choices for users. For visual SLAM, CU-Fundamental supports the ORB-SLAM series [13, 14, 15] and the VINS series [16, 17], which are both well-known open-source visual SLAM algorithms. For LiDAR SLAM, CU-Fundamental implements the LOAM family [18] and Fast-LIO [19].

Additionally, some specific mapping algorithms are included as a middle layer between SLAM and planning algorithms. For example, our previous work [20] provides a GPU-accelerated Euclidean distance transform algorithm to enhance environmental awareness of the obstacles, ensuring better motion planning performance. Finally, AI-powered object detection and segmentation algorithms are also supported to provide semantic understanding of the world. CU-Fundamental includes the YOLO series [21, 22] and provides corresponding interfaces.

Planning: A hierarchical planning framework based on our previous work [23] is implemented in the system, which consists of a task planner, a path planner and a motion planner. A task planner based on linear temporal logic [24] is first called to generate a sequence of waypoints based on mission specifications. The path planner then searches for a feasible path, which is subsequently smoothed by a motion planner to generate a trajectory consisting of a sequence of desired poses, velocities and accelerations. Moreover, CU-Fundamental includes popular motion planning algorithms

such as Ego-planner [25].

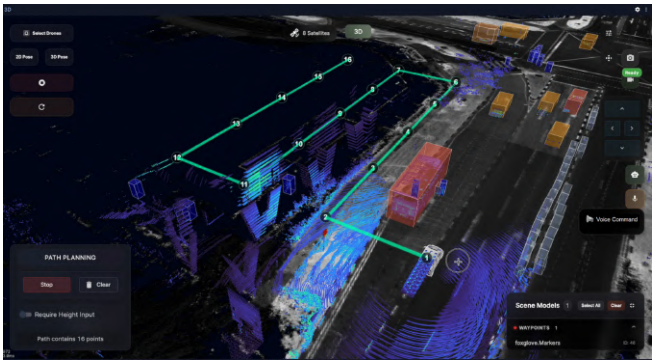


Figure 3: A Snapshot of AstroGCS visualizing the nuScenes dataset [26]. The waypoints are generated and published to onboard computer using large language models with the following prompt: "Generate a zigzag path with three rows, each row containing five waypoints".

2.4 *Interface and Communication*

The communication between different algorithms is facilitated by ROS, whose message-passing mechanism allows for efficient data exchange. In addition to the standard messages, CU-Fundamental provides a set of custom messages[1] to handle the data flow between SLAM, planning and control algorithms.

In terms of lower-level communication with the simulated flight control board, Mavlink and Mavros are used to ensure compatibility with the PX4 firmware. For example, the sensor data and MAV dynamic status simulated by Gazebo plugins are published to the flight controller through Mavlink, while other necessary information (localization results, user control commands, etc.) that are calculated in ROS are sent through Mavros. For detailed information, readers are referred to [6].

On the user side, CU-Fundamental provides a more user-friendly and modernized GCS named AstroGCS using the web-based (HTML5 and React) architecture. The GCS provides a 3D graphical interface for users to monitor the status of the MAV, visualize the environment, and interact with the system. As is illustrated in Figure 3, it also supports advanced features such as publishing planning commands and invoking AI agents to assist in decision-making. AstroGCS has access to all the data communicated in ROS through ROS Bridge (See Figure 4). Compared to Qt-based counterparts, the web-based design is easier to program and extend thanks to the rich ecosystem of web technologies. Moreover, since it is web-based, the GCS can be deployed on any device with a modern web browser and could be even accessed remotely, suggesting better cross-platform compatibility and more convenient user experience.

[1] https://github.com/CUHK-UAS-FUNDAMENTAL/usrl_msgs/

One shortcoming of AstroGCS is that it does not support showing the Gazebo simulation since Gazebo adopts a different communication and rendering strategy. Gazebo uses a server-client architecture, where the server (GzServer) is responsible for simulating the physics while the client (GzClient) is for rendering the simulation results. Since GzClient is not web-based, it cannot be directly integrated into AstroGCS. Thankfully, Gazebo provides GzBridge and GzWeb that allow users to visualize the simulation results in a web browser. CU-Fundamental modifies the GzWeb to provide advanced functionalities such as manipulating MAV models. In this way, GzWeb and AstroGCS work in accordance to provide a unified system UI in web architecture, where users could monitor the simulation results in GzWeb and manipulate drone algorithms in AstroGCS. Altogether, the architecture of the interface and communication is shown in Figure 4.

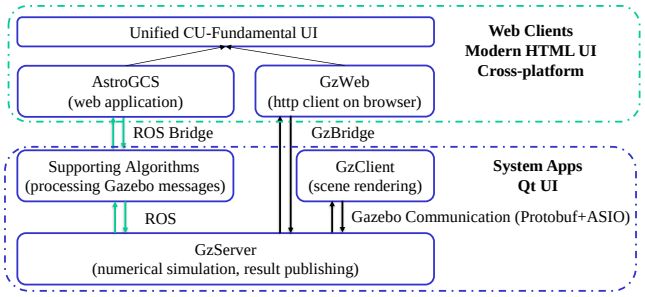


Figure 4: Communication and interface between AstroGCS, algorithms and Gazebo simulation. Components in the blue box are readily implemented in ROS and adopted by most simulation frameworks. CU-Fundamental provides additional unified web interface as marked in the green box.

3 Case Study

This study reproduces the tunnel navigation challenge in the IMAV 2025 competition, where the MAV is required to navigate through five 0.5 m × 1 m tunnels spanning a total length of 2 m. A new MAV named CU-Astro is designed for this task, which is a quadrotor MAV with a size of 216 mm × 216 mm × 95 mm. The MAV is equipped with a 3D LiDAR (Livox Mid-360) for LiDAR SLAM and detection of the tunnel walls. The overall design of CU-Astro is illustrated in the background of Figure 2.

Two necessary testing stages are required to complete the task. Firstly, the MAV should be stable enough to track a given trajectory, which is primarily achieved by tuning the control parameters. Subsequently, the high-level detection and planning algorithms are tested to automatically generate a feasible trajectory for navigation.

3.1 *Minimum-snap Trajectory Generation and Tracking*

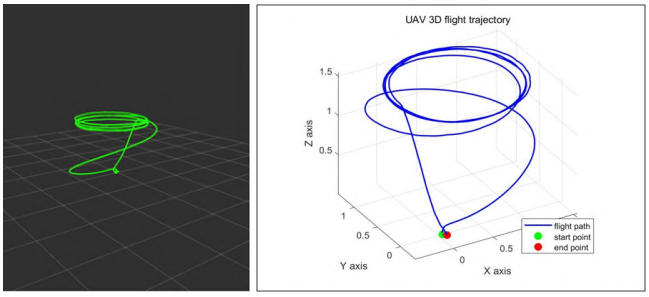
To ensure basic stability and controllability of the MAV, a minimum-snap trajectory is generated for the drone to follow [27]. Minimum-snap trajectory is a widely used

method for generating smooth flight paths for MAVs and integrated in various motion planning algorithms included in CU-Fundamental. Therefore, successful tracking of minimum-snap trajectory suggests easier integration of other algorithms in the future. Specifically, the problem could be formulated as the following optimization objective:

$$\min_{\mathbf{p}(t)} \int_{t_0}^{t_f} \alpha \left\| \mathbf{p}^{(n)}(t) \right\|^2 + \beta \left\| \Psi^{(m)}(t) \right\| dt, \quad (6)$$
$$\text{s.t. } \mathbf{p}(t_0) = \mathbf{p}_0, \mathbf{p}(t_f) = \mathbf{p}_f, \Psi(t_0) = \Psi_0, \Psi(t_f) = \Psi_f$$

where $\mathbf{p}(t) \in \mathbb{R}^3$ is the position of the MAV at time t, $\Psi(t)$ is the yaw angle, $\mathbf{p}_0$ and $\mathbf{p}_f$ are the initial and final positions, Ψ_0 and Ψ_f are the initial and final yaw angles. n and m control the order of derivatives. α and β are scaling factors. For minimum-snap trajectory generation, $n = 4$ and $m = 2$ are used, which results in minimized differential thrust in the physical system, thereby reducing the energy consumption and improving the smoothness of the flight. To track the generated trajectory, a typical cascaded PID controller is used.

In the simulation stage, a minimum-snap trajectory is generated to pass through four waypoints located at the corners of a square with a size of 1 m × 1 m. The square is placed on a horizontal plane at a height of 1.5 m.



(a) Simulation (b) Real flight

Figure 5: Minimum-snap trajectory generated and tracked in simulation and real flight.

After the workflow of the algorithms is successfully verified, the same algorithms are deployed to the onboard computer. The control parameters used in the simulation serve as the initial values for the real flight and are further fine-tuned using the PX4 Autotune. Figure 5 shows the minimum-snap trajectory generated and tracked in both simulation and real flight. It is noted that these two trajectories are highly similar, indicating that the algorithms are successfully transferred from simulation to real flight. The real flight data is recorded by PX4 and shown in Figure 6. While there's some delay in the response of the MAV, the final trajectory aligns well with the planned.

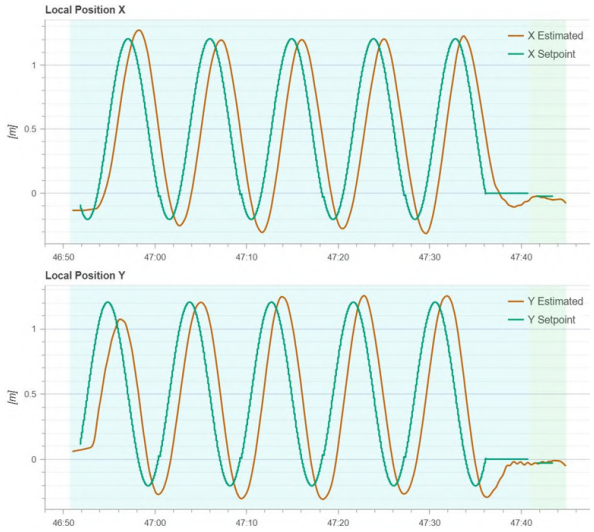


Figure 6: Real flight data record of tracking results.

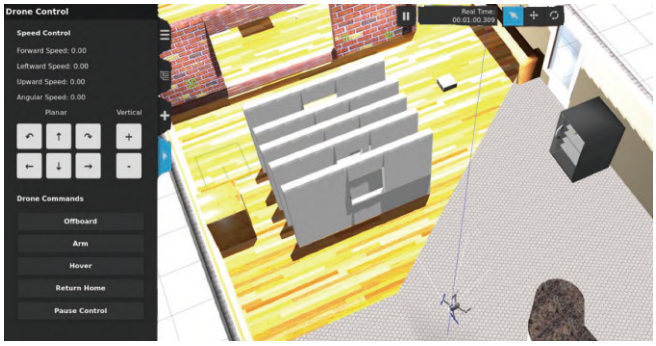
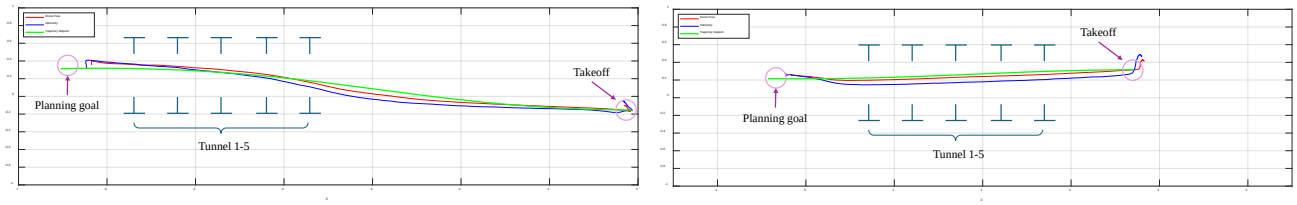


Figure 7: Gazebo simulation of the tunnel navigation challenge. The scene rendering is captured from GzWeb. The left side menu also shows a part of the GzWeb interface, which allows users to manipulate the MAV directly and visualize the simulation results.

3.2 *Tunnel Detection and Navigation*

After the MAV's mobility is verified, the next is to integrate high-level algorithms including object detection, SLAM and motion planning algorithms are tested to find feasible solutions. In the simulation stage, a Gazebo environment as shown in Figure 7 is first created to simulate the tunnel environment. Then the MAV is manually guided using proper external localization and motion planner. As CU-Fundamental has built-in algorithms and interfaces for these functionalities, the development process is significantly accelerated. Preliminary experiments suggest that a combination of Fast-LIO2, Ego-planner and PID controller can achieve successful navigation provided that correct waypoints are given.

To automatically generate waypoints, a simple object detection method based on 3D point cloud is implemented. The algorithm first filters the point cloud to remove noisy points,



(a) Simulation (b) Real flight

Figure 8: Visualization of the trajectory in simulation and real flight. The drone pose (red), odometry (blue) and trajectory setpoints (green) are shown in the figure. The Z-axis is omitted in the plot since the tunnel is high enough to tolerate relatively large altitude errors.

then applies a DBSCAN clustering algorithm to detect and segment the tunnel walls. The center of the tunnel is then estimated as the average of the points in each cluster. After the MAV is guided to the center of the first tunnel wall, the location of the furthermost center is used as the planning goal.

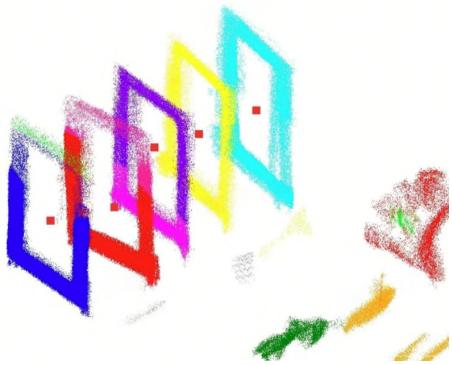


Figure 9: DBSCAN clustering of tunnel walls. The center is marked as red points.

The aforementioned algorithms are first tested in simulation and then deployed for real flight. Since the detection algorithm is based on the relatively easy-to-simulate LiDAR sensor (only geometry of the pointcloud is required), there is little simulation-to-real gap. The final flight trajectory is illustrated in Figure 10.

To qualitatively evaluates the function of CU-Fundamental, Figure 8 compares the odometry, actual drone pose and trajectory setpoints in both simulation and real flight. The statistics of the corresponding errors are reported in Table 1. Odometry error is defined as the Euclidean distance between the drone pose and the odometry while tracking error is that between the drone pose and the trajectory setpoints.

As is indicated in the figure and table, the error pattern in real flight is similar to that in simulation, suggesting the practical usability of CU-Fundamental as a simulation framework for real-world applications. Both odometry and tracking errors are statistically below 10 cm, which is acceptable for the task since the drone size is 21.6 cm compared to the tunnel size of 50 cm. The maximum tracking error is recorded at the end of the flight in both cases, which is likely due to the accumulated error in the odometry. To overcome this issue, it is advised that the MAV should adopt some advanced algorithms to readjust its location with respect to the final tunnel.

Figure 10: Composite image of the drone trajectory.

Table 1: Odometry error and tracking error (Unit: m)

Error type	Odometry error		Tracking error	
	simulation	real	simulation	real
Min	0.0217	0.0447	0.0037	0.0022
RMSE	0.0962	0.0855	0.0316	0.0655
Max	0.2577	0.1449	0.0575	0.1647

4 Conclusion

This paper presents CU-Fundamental, a comprehensive simulation framework for autonomous MAVs. The framework supports multiple levels of simulation, including numerical simulation, SITL, HITL, and integrates essential algorithms for autonomous flight. CU-Fundamental is based on modular design, implementing standard interfaces for various SLAM, planning and control algorithms. It also innovates the user interface by providing web-based AstroGCS

and GzWeb, which allows for advanced user interaction and easier secondary development. The framework is tested in a case study of tunnel navigation, demonstrating its effectiveness in simulating and testing autonomous MAV algorithms. The future work will focus on further enhancing the modularity and extensibility of CU-Fundamental, as well as improving simulation realism to better reflect real-world scenarios.

ACKNOWLEDGEMENTS

This work was supported in part by the InnoHK of the Government of the Hong Kong Special Administrative Region via the Hong Kong Centre for Logistics Robotics and in part by the Research Grants Council of Hong Kong SAR (Grant Nos: 14217922, 14209623 and 14209424).

REFERENCES

[1] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace sciences*, 91:99–131, 2017.

[2] Julius A Marshall, Wei Sun, and Andrea L'Afflitto. A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems. *Annual Reviews in control*, 52:390–427, 2021.

[3] Adriano Bittar, Helosman V Figuereido, Poliana Avelar Guimaraes, and Alessandro Correa Mendes. Guidance software-in-the-loop simulation using x-plane and simulink for uavs. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 993–1002. IEEE, 2014.

[4] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics: Results of the 11th international conference*, pages 621–635. Springer, 2017.

[5] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 6235–6240. IEEE, 2015.

[6] Kun Xiao, Shaochang Tan, Guohui Wang, Xueyan An, Xiang Wang, and Xiangke Wang. Xtdrone: A customizable multi-rotor uavs simulation platform. In *2020 4th International Conference on Robotics and Automation Sciences (ICRAS)*, pages 55–61. IEEE, 2020.

[7] Russell Smith et al. Open dynamics engine. 2005.

[8] Brian L Stevens, Frank L Lewis, and Eric N Johnson. *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons, 2015.

[9] PAL Robotics. Intel realsense gazebo ros plugin, 2019. Available at `https://github.com/pal-robotics/realsense_gazebo_plugin`. Last accessed: 2025-07-08.

[10] Francesco Vultaggio, F d'Apolito, C Sulzbachner, and P Fanta-Jende. Simulation of low-cost mems-lidar and analysis of its effect on the performances of state-of-the-art slams. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 48:539–545, 2023.

[11] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Control of complex maneuvers for a quadrotor uav using geometric methods on se (3). *arXiv preprint arXiv:1003.2005*, 2010.

[12] Taeyoung Lee, Melvin Leok, and N Harris McClamroch. Geometric tracking control of a quadrotor uav on se (3). In *49th IEEE conference on decision and control (CDC)*, pages 5420–5425. IEEE, 2010.

[13] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[14] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[15] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE transactions on robotics*, 37(6):1874–1890, 2021.

[16] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE transactions on robotics*, 34(4):1004–1020, 2018.

[17] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors, 2019.

[18] Ji Zhang, Sanjiv Singh, et al. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014.

[19] Wei Xu and Fu Zhang. Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter. *IEEE Robotics and Automation Letters*, 6(2):3317–3324, 2021.

[20] Yizhou Chen, Shupeng Lai, Jinqiang Cui, Biao Wang, and Ben M Chen. Gpu-accelerated incremental euclidean distance transform for online motion planning of mobile robots. *IEEE Robotics and Automation Letters*, 7(3):6894–6901, 2022.

[21] Marko Bjelonic. YOLO ROS: Real-time object detection for ROS, 2016–2018. Available at `https://github.com/leggedrobotics/darknet_ros`. Last accessed: 2025-07-08.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[23] Lele Xi, Xinyi Wang, Lei Jiao, Shupeng Lai, Zhihong Peng, and Ben M Chen. Gto-mpc-based target chasing using a quadrotor in cluttered environments. *IEEE Transactions on Industrial Electronics*, 69(6):6026–6035, 2021.

[24] Yizhou Chen, Ruoyu Wang, Xinyi Wang, and Ben M Chen. Sampling-based path planning under temporal logic constraints with real-time adaptation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3700–3706. IEEE, 2023.

[25] Xin Zhou, Zhepei Wang, Hongkai Ye, Chao Xu, and Fei Gao. Ego-planner: An esdf-free gradient-based local planner for quadrotors. *IEEE Robotics and Automation Letters*, 6(2):478–485, 2020.

[26] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020.

[27] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *2011 IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.