Multi-Agent UAV-UGV Coordination for Autonomous Navigation

D[1], D[2], D[2]
[1]D

***Abstract*—Later**

***Index Terms*—Later**

I. Introduction

II. System Model

The proposed system consists of a UAV and a UGV operating in a shared environment, where the UAV autonomously navigates to a given goal while the UGV follows as a mobile battery station. The simulation is implemented in ROS2 and Gazebo, utilizing a 3D Point Cloud Data (PCD) map for environmental representation. The UAV's global path planning is handled using a Probabilistic Roadmap (PRM) and A* algorithm, followed by a Pure Pursuit Path Solver for smooth trajectory execution. A PID controller ensures stability during flight, integrating with UAV dynamics for precise movement. The UGV employs a Dynamic Window Approach (DWA) Controller for real-time local path planning and obstacle avoidance while maintaining a following behavior relative to the UAV. Communication between the UAV and UGV is established through ROS2 topics, enabling continuous data exchange for coordinated motion.

A. Mapping and Environment Representation

The system relies on a 3D Point Cloud Data (PCD) map for environmental perception, enabling both the UAV and UGV to navigate efficiently. As shown in the block diagram, the raw map is first converted into a structured 3D PCD map, which is then processed to extract relevant information such as free space, obstacles, and navigable areas. This processed data is used by the UAV for global path planning and by the UGV for local navigation.

For map generation, the system can utilize pre-recorded maps or real-time sensor data (such as LiDAR) to create an updated representation of the environment. The processed PCD map is fed into the UAV's global planner to compute a collision-free path, while the UGV integrates this data into the Navigation2 (Nav2) stack, allowing it to reactively follow the UAV while avoiding obstacles. This unified mapping framework ensures that both robots operate in a synchronized and environment-aware manner.

The environment is modeled as a multi-dimensional navigation space, typically using:

1) 3D Grid Representation (for UAVs): The UAV operates in a bounded volumetric space defined as $[X_{\text{UAV}}, Y_{\text{UAV}}, Z_{\text{UAV}}]$, which is discretized into voxels or occupancy grids with resolution-dependent cells for path planning [1]; thus, the number of voxels can be calculated using the following formula:

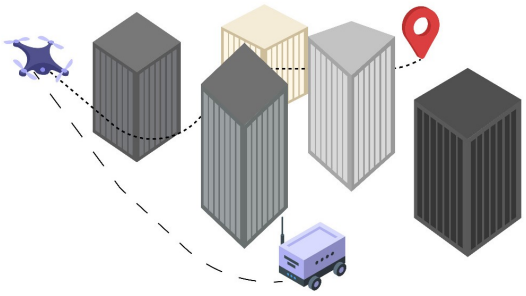

Fig. 1: Illustration of the UAV navigation in an urban environmen

$$N_v = \left(\frac{X_{\text{UAV}}}{\text{Resolution}}\right) \times \left(\frac{Y_{\text{UAV}}}{\text{Resolution}}\right) \times \left(\frac{Z_{\text{UAV}}}{\text{Resolution}}\right) \quad (1)$$

In (1). Substituting the given values:

$$N_v = \left(\frac{40}{0.5}\right) \times \left(\frac{20}{0.5}\right) \times \left(\frac{10}{0.5}\right) = 64000 \text{ voxel}$$

2) 2D Grid Representation (for Ground Robots): The ground robot moves on a planar surface defined as $[X_{\text{GR}}, Y_{\text{GR}}]$, which is discretized into a grid of cells, resulting in

$$N_c = \left(\frac{X_{\text{UGV}}}{\text{Resolution}}\right) \times \left(\frac{Y_{\text{UGV}}}{\text{Resolution}}\right) \quad (2)$$

$$N_c = \left(\frac{40}{0.5}\right) \times \left(\frac{20}{0.5}\right) = 32000 \text{ cell}$$

3) Obstacles are categorized as static (fixed objects): Buildings, walls, and trees are represented as occupancy grids, where each cell (or voxel) is marked as free or occupied [2]. The occupancy map $M(x, y, z)$ is defined as:

$$M(x, y, z) = \begin{cases} 1, & \text{if occupied} \\ 0, & \text{if free} \end{cases}$$

B. UAV Model

The UAV is described as a quadrotor system. The position vector $\mathbf{r}$, representing the center of mass in the world coordinate frame, is tied to the UAV's dynamics, as expressed in the following:

$$m\ddot{r} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}, \quad (1)$$

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \quad (2)$$

Here, m denotes the UAV's mass, and R is the rotation matrix that converts the body frame B into the world frame W using the $Z-X-Y$ Euler angle convention. The symbol I represents the moment of inertia about the UAV's center of mass, and L is the distance from the center of mass to each rotor's axis of rotation. The variables p, q, and r are the components of the angular velocity ω_{BW} within the body frame B.

The control inputs include two parts: the total thrust

$$u_1 = F_1 + F_2 + F_3 + F_4$$

and the moments given as follows:

$$u_2 = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}. \tag{3}$$

The state vector of the UAV is defined as:

$$x_{\text{UAV}} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T \tag{4}$$

where $\mathbf{r} = [x, y, z]^T$ and ψ are the position and the yaw angle.

C. Ground Robot Model

A ground robot operates under non-holonomic constraints, meaning it cannot move sideways like a UAV or a holonomic robot (e.g., omnidirectional wheels). It can only move forward, backward, or rotate based on its wheeled configuration. The generalized state vector for a ground robot is:

$$x_{\text{GR}} = (x, y, \theta, v, \omega)^T$$

where:

- $x, y \rightarrow$ Robot's position in a 2D coordinate frame.
- $\theta \rightarrow$ Robot's heading (yaw angle) in radians.
- $v \rightarrow$ Linear velocity in meters per second (m/s).
- $\omega \rightarrow$ Angular velocity in radians per second (rad/s).

The kinematic model of a ground robot follows:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = \omega$$

where:

- $v \rightarrow$ Linear velocity along the forward direction.
- $\omega \rightarrow$ Angular velocity determining turning rate.

In a differential drive robot (DDR), the wheel velocities determine motion:

$$v = \frac{r}{2}(\omega_L + \omega_R), \quad \omega = \frac{r}{d}(\omega_R - \omega_L)$$

where:

- $r \rightarrow$ Wheel radius.
- $d \rightarrow$ Wheel separation distance.
- $\omega_L, \omega_R \rightarrow$ Left and right wheel angular velocities.

D. UAV-UGV Channel Model

III. Methodology

A. UAV Navigation Framework

1) Global Path Planning for UAV: The UAV's navigation is handled by a Probabilistic Roadmap (PRM) and A* based global planner*, ensuring an *efficient* and obstacle-free path to the goal. PRM is initially used to generate a roadmap graph, connecting feasible waypoints while avoiding obstacles. The A* algorithm* is then applied to compute the shortest path from the start position to the goal.

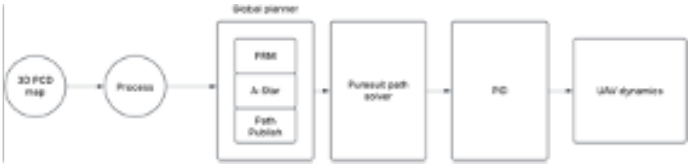

Fig. 2: UAV control flow diagram

To further optimize the generated path, a secondary path simplification step is included. This step evaluates whether the initially computed path can be shortened by checking if direct connections between intermediate waypoints are possible without encountering obstacles. If a more direct route exists, the unnecessary waypoints are eliminated, reducing overall travel distance and improving efficiency.

This dual-stage approach—PRM for roadmap generation, A* for pathfinding, and a simplification check for optimization*—ensures that the UAV follows the most efficient route while maintaining obstacle avoidance. The optimized path is then published for execution by the Pure Pursuit Path Solver, which generates smooth trajectory commands for the UAV to follow. A detailed flow is shown in Fig. 2.

2) Path Following and Control for UAV: After the global path has been determined, the UAV follows the computed trajectory using a Pure Pursuit Path Solver. This algorithm calculates a series of target waypoints along the planned path and continuously adjusts the UAV's heading and velocity to maintain smooth navigation. The Pure Pursuit method ensures that the UAV maintains a stable path, avoiding sharp turns or erratic movements.

To further enhance trajectory tracking, a PID Controller is integrated to regulate velocity and orientation. The PID controller takes feedback from UAV sensors and continuously adjusts throttle, roll, pitch, and yaw to minimize deviations from the desired path. This control mechanism improves flight stability and ensures smooth movement, even in the presence of disturbances such as wind or minor environmental variations.

The final control outputs from the PID controller are then applied to the UAV's dynamics model, executing the planned motion in the Gazebo simulation environment. This multi-stage control framework enables precise and adaptive UAV navigation, making it robust for real-world deployment scenarios.

B. UGV Navigation Using NAV2 and DWA Planner

Unlike the UAV, which follows a predefined global path, the UGV dynamically adjusts its trajectory using the NAV2 stack and the Dynamic Window Approach (DWA) planner. The NAV2 stack provides a flexible and modular navigation system that allows the UGV to autonomously move while reacting to environmental changes.

The DWA planner, used within NAV2, is responsible for local path planning and obstacle avoidance. DWA operates

Fig. 3: UGV control flow diagram

by evaluating possible velocity commands within a defined window, selecting the most optimal movement that avoids obstacles while keeping the UGV aligned with the UAV. This reactive approach allows the UGV to dynamically follow the UAV without requiring a strict predefined path.

The process follows these steps:

- **Map Processing:** The processed 3D PCD map is provided as input to NAV2, enabling the UGV to understand the navigable space.
- **Localization:** The UGV continuously tracks its position using sensor fusion (e.g., odometry and LiDAR-based SLAM).
- **Path Planning:** DWA evaluates velocity and trajectory options, ensuring that the UGV moves safely while staying in proximity to the UAV.
- **Obstacle Avoidance:** If an obstacle is detected, DWA recalculates the best trajectory while maintaining synchronization with the UAV.

By using NAV2 and DWA, the UGV is able to autonomously adjust its path while ensuring efficient coordination with the UAV. This method enhances the system's *flexibility*, allowing it to operate in dynamic and cluttered environments.

D. Communication and Coordination Between UAV and UGV

Coordination between the UAV and UGV is established through ROS2 communication mechanisms, ensuring real-time data exchange for synchronized movement. The UAV continuously publishes its position and planned path via ROS2 topics, which the UGV subscribes to in order to dynamically adjust its movement.

The communication process includes:

- Position Sharing: The UAV broadcasts its current location and planned trajectory.
- Velocity Commands: The UAV can send suggested velocity updates for smoother coordination.
- Feedback Loop: The UGV sends periodic status updates, allowing for adjustments in movement synchronization.

This communication framework enables the UGV to reactively follow the UAV, maintaining an optimal distance while navigating obstacles. The use of ROS2 for decentralized coordination ensures that both robots can operate *independently* while staying in sync, making the system scalable for future enhancements such as multi-UAV and multi-UGV coordination.

E. Path Planning Algorithms

1. Global Path Planning Using PRM + A:* The UAV computes its global trajectory using a Probabilistic Roadmap (PRM) combined with A*.

Step 1: Probabilistic Roadmap (PRM) Generation

- PRM constructs a graph representation of the environment by randomly sampling N points in free space [**?**].
- Nodes are connected if their direct path is collision-free, forming an undirected graph.

$$G = (V, E)$$

where

$$V = \{v_1, v_2, \ldots, v_n\}, \quad E = \{(v_i, v_j) \mid \textit{ValidEdge}(v_i, v_j)\}$$

and *ValidEdge* ensures obstacle-free connections.

Step 2: Path Search Using A*

- A* finds the shortest path using the cost function [**?**]:

$$f(n) = g(n) + h(n)$$

where:

- $g(n)$ is the travel cost from the start node to n.
- $h(n)$ is the estimated heuristic cost to the goal (Euclidean distance).

2. Trajectory Execution Using Pure Pursuit: The Pure Pursuit Path Solver generates smooth steering commands by computing a look-ahead point ahead of the UAV's current position [**?**].

The heading correction is calculated as:

$$\delta = \tan^{-1}\left(\frac{2L\sin(\theta)}{d}\right)$$

where:

- L is the UAV's wheelbase equivalent.
- θ is the angle between the current heading and the target point.
- d is the look-ahead distance.

The Pure Pursuit output provides target velocities and orientation commands.

3. Dynamic Window Approach (DWA): Unlike the UAV, the UGV follows the UAV dynamically using DWA within the Nav2 stack. DWA optimizes velocity commands by evaluating the cost function [**?**]:

$$G(v, \omega) = \alpha \cdot \textit{heading}(v, \omega) + \beta \cdot \textit{clearance}(v, \omega) + \gamma \cdot \textit{velocity}(v, \omega)$$

Where:

- *Heading* term ensures goal alignment.
- *Clearance* term prevents obstacles.
- *Velocity* term ensures efficient movement.

The selected velocity (v, ω) enables real-time obstacle avoidance while maintaining synchronization with the UAV.

4. PID Controller: : [**?**].
[**?**].

IV. Results

V. Conclusion

References

[1] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

Vehicle	Average Time (s)	Average Energy (J)
UAV	41.61	6383
UGV+UAV	22.88	3498

Table I: Average time and energy

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.