Autonomous Drone Project

Team 12
Jiaxiang Yang
Mohammadali Rahmati
Xuanyi Lin
Enze Wang
Yicun Song

March 2025

1 Introduction

The Sub-Terrain Challenge is a team-based project aimed at developing an autonomous system capable of efficiently exploring an underground environment. The primary objectives of this challenge are twofold: first, to locate four objects of interest (lights) as quickly as possible and accurately determine their positions within a cave-like simulation environment; second, to generate a 3D voxel-grid or mesh representation of the explored environment. This project requires seamless integration of multiple components, including perception, mapping, localization, path planning, and trajectory control.

2 System Architecture and Modules

Developing an autonomous drone requires a well-structured division of tasks among different system modules. The responsibilities of team members are distributed to optimize the efficiency of perception, planning, and control modules. Table 1 presents the breakdown of core tasks assigned to each team member.

Member Name	**Task Description**
Jiaxiang Yang	Mapping (PCD, OctoMap), Trajectory Generation
Mohammadali Rahmati	State Machine, Light Detection
Xuanyi Lin	Mapping (PCD, OctoMap), Path Planning
Enze Wang	Trajectory Generation
Yicun Song	Path Planning

Table 1: Core task distribution among team members

2.1 State Machine Module

The **State Machine module** is the core of the UAV's control system, managing transitions. It continuously evaluates **sensor inputs** and **mission objectives**, ensuring smooth transitions while dynamically updating mission parameters through interactions with **navigation and mapping nodes**.

2.2 Navigation Module

This module is responsible for autonomous movement and trajectory planning. It implements a combination of frontier-based exploration and waypoint tracking to systematically navigate unknown terrains. The navigation module integrates the **Rapidly-exploring Random Tree Star (RRT*)** algorithm, which iteratively constructs an optimal trajectory by sampling waypoints and linking them in a cost-effective manner. By continuously evaluating the occupancy grid, this module ensures that the UAV follows a collision-free path while efficiently covering unexplored areas.

2.3 Mapping and Perception Module

To build a three-dimensional representation of the surroundings, the UAV employs an **OctoMap-based volumetric mapping framework**. The depth camera provides raw image data, which is converted into a 3D point cloud. The OctoMap library then processes this data into an occupancy grid where regions are classified as *free space, occupied space, or unknown space.* The mapping module runs concurrently with the navigation system, updating the UAV's understanding of its surroundings in real time. According to Fig. 1, the generated 3D map provides a structured spatial representation of the environment.

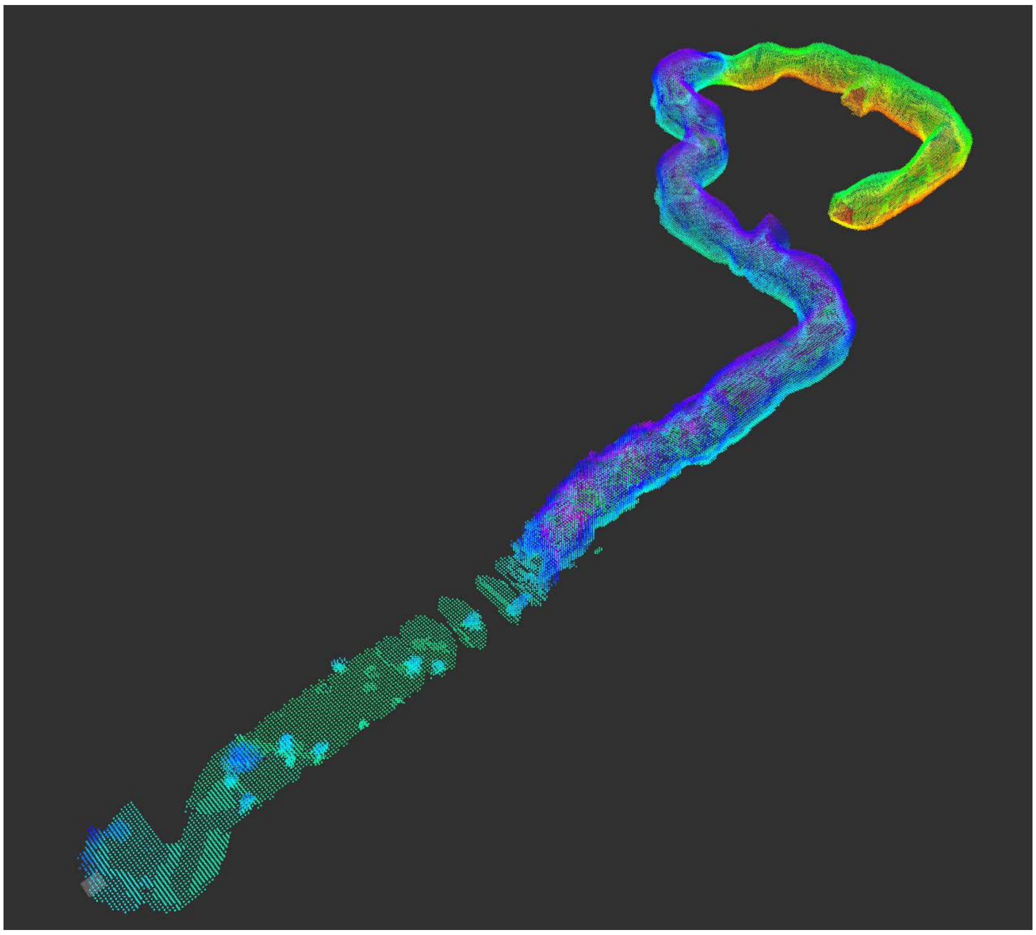

Figure 1: 3D Mapping generated by the UAV.

2.4 Vision and Object Detection Module

The Light Detection Package consists of two ROS nodes: the **Semantic Mask Node** and the **Light Detector Node**, which work together to process depth and semantic segmentation images for detecting light sources. Integrated into the UAV's perceptual system, which includes **semantic and depth cameras**, this module identifies and tracks objects within the environment. It processes visual inputs to detect key features, such as light sources or obstacles, using a color-based filtering technique that converts RGB images into a binary mask to isolate relevant regions. Once detected, the objects are localized within the global reference frame, enabling the UAV to make informed decisions.

2.5 Trajectory Planning and Execution Module

After identifying waypoints, the system must generate a feasible trajectory. The **mav_trajectory_generation** package is used to optimize smooth, dynamically feasible paths. This package employs polynomial trajectory representations to minimize sudden changes in acceleration, ensuring stable UAV movement. The computed trajectory is sampled and executed in coordination with the UAV's control system.

3 Generated ROS Nodes

3.1 Frontier Exploration Node

This node enables the UAV to systematically explore unknown areas by detecting the boundaries between mapped and unexplored regions. It evaluates the occupancy grid and identifies frontiers that warrant further investigation. Once a frontier is selected, it is passed to the navigation module for execution.

3.2 Planner Node

The planner node is crucial for computing collision-free paths. It integrates **Obstacle-aware Motion Planning Library (OMPL)** and **Fast Collision Library (FCL)** to dynamically adjust the UAV's movement based on environmental constraints. This node ensures that the planned path maintains a safe distance from obstacles while prioritizing efficient coverage of the target area.

3.3 State Machine Node

The state machine operates through a finite set of flight phases, ensuring structured navigation:

- **ascend_step**: Takeoff from the initial position.
- **transit_cave**: Move towards the cave entrance.
- **stationary**: Maintain a hovering state at specific waypoints.
- **descent**: Initiate the landing sequence.
- **rotate_heading**: Adjust the UAV's orientation.
- **move_straight**: Move forward in a straight trajectory.

3.3.1 Key Functions

- **State Management:** Handles phases like *ascend, transit, hover, descent, rotation, and straight movement* to execute the mission.
- **Path Navigation:** Processes planned paths, dynamically adjusts waypoints, and computes yaw angles for accurate directional control.
- **Real-Time Execution:** Uses a **ROS timer** to manage state transitions via specialized functions (`performAscent()`, `moveIntoCave()`, etc.).
- **Waypoint & Sensor Integration:** Retrieves UAV **odometry data**, processes planned paths, and validates goal completion.
- **ROS Communication:** Publishes trajectory commands (`desired_state_pub_`), receives state updates (`current_state_sub_`), and broadcasts transformations.

3.4 Waypoint Navigation Node

To execute planned trajectories, this node manages UAV movement between predefined waypoints. It adjusts velocity and orientation to ensure precise navigation, making real-time corrections when necessary. This node ensures that the UAV follows its designated path without deviation.

3.5 Sampler Node

This node is responsible for processing trajectory points at optimized intervals. By sampling trajectory commands at controlled frequencies, it reduces computational load while maintaining smooth motion execution. It also interacts with external position-hold services to ensure stability during trajectory execution.

3.6 Depth-to-Cloud Node

Transforming raw depth images into 3D point clouds, this node plays a critical role in environmental perception. It utilizes the **depth_image_proc** package to efficiently process image frames and generate accurate spatial representations for navigation and mapping.

3.7 Octomap Server Node

Acting as the central mapping framework, this node maintains a **probabilistic occupancy map** that continuously updates as new data is received. It enables real-time responses to environmental changes, ensuring up-to-date spatial awareness.

3.8 Semantic Mask Node

Function: Filters depth data based on semantic segmentation.
Inputs:

- `/realsense/semantic/image_raw` (Semantic image)
- `/realsense/depth/image` (Depth image)
- `/realsense/depth/camera_info` (Camera calibration)

Output: `masked_depth_image` (Depth image filtered by semantic mask)
Process: Converts images to OpenCV format, applies masking, and publishes the masked depth image.

3.9 Light Detector Node

Function: Extracts 3D coordinates of detected light sources.
Inputs:

- `/masked_depth_image` (Processed depth image)
- `/realsense/depth/camera_info` (Camera calibration)

Output: `detected_points` (3D coordinates of detected lights)
Process: Converts depth to PointCloud, transforms to world coordinates, filters redundant detections, and publishes detected points.

4 Implementation and Testing

To validate system performance, extensive simulations were conducted in Unity and ROS environments. The UAV was subjected to various underground scenarios, including irregular cave structures and narrow passageways. Through iterative testing, the system's navigation and mapping accuracy were refined. Challenges such as segmentation faults in the exploration process were addressed by optimizing memory management in the OctoMap server.

5 Conclusion

This project successfully developed an autonomous UAV system for navigating and mapping environments. By leveraging advanced perception systems, state-of-the-art path-planning algorithms, and real-time occupancy mapping, the system demonstrated robust autonomous exploration capabilities. Future enhancements may include integrating machine learning techniques for adaptive navigation and refining obstacle avoidance strategies for complex terrains.
The final QRT graph representing the project's results is shown in Figure 2.

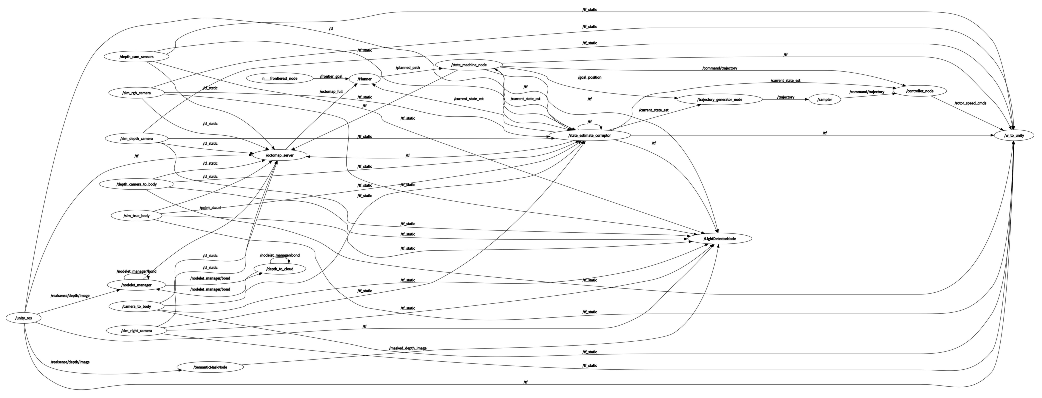

Figure 2: Final QRT Graph of the Project