Autonomous Systems:
Group Project: Sub-Terrain Challange

Bingkun Huang, Haowen Shi, Siyan Li, Weili Tang, Zhenjiang Li

1 Introduction

This project is part of the Sub-Terrain Challenge in the Autonomous Systems course at TUM. The objective is to develop a system that can autonomously explore a cave environment, detect and locate four objects of interest (lights), and generate a 3D voxel-grid or mesh representation of the environment. The implementation involves a ROS-based framework, integrating perception, path planning, and control using a quadrotor and a Unity-based simulation. This document provides an overview of the system architecture, software components, team contributions, challenges, and results, including a ROS graph, figures.

2 Project Goals

The project aims to develop an autonomous system capable of exploring a cave environment, detecting and locating four objects of interest (lights) as quickly as possible. Additionally, the system must generate a 3D representation of the environment using either a voxel-grid or a mesh-based approach. Key objectives include:

Implementing a perception pipeline to process depth images and generate point clouds. Developing path and trajectory planning algorithms for autonomous navigation. Ensuring seamless ROS integration via a simulation bridge for real-time communication. Designing a state machine for robot operation, handling take-off, navigation, and landing.

3 System Overview

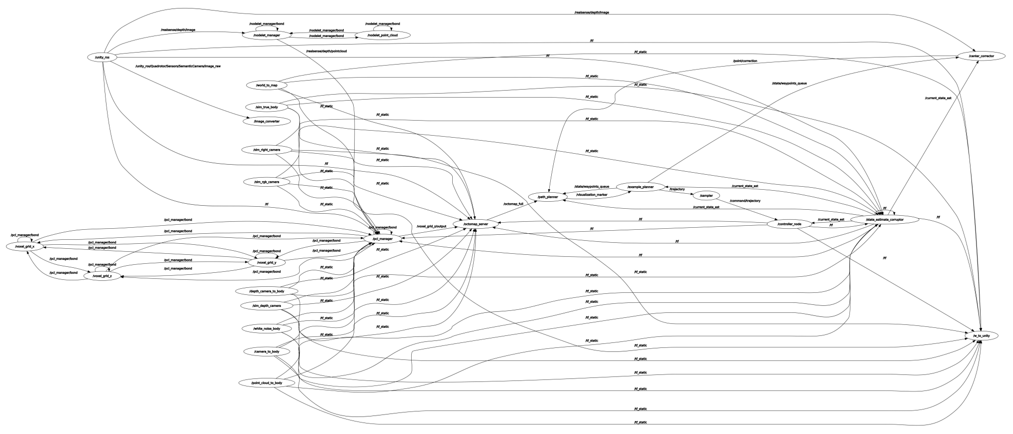
3.1 Mav Trajectory Generation

The mav trajectory generation [1] package provides tools for polynomial trajectory generation and optimization, specifically designed for micro aerial vehicles, such as quadrotors. It implements both linear and nonlinear trajectory optimization methods based on minimum snap or minimum jerk approaches, ensuring smooth, dynamically feasible paths for MAV navigation.

The package supports waypoint-based trajectory generation, where users define key positions, and the optimizer calculates continuous polynomial paths. It allows time allocation optimization, ensuring efficient speed profiles. Additionally, it integrates feasibility checks for velocity, acceleration, and thrust constraints to prevent actuator saturation.

3.2 Eigen Catkin

Eigen catkin is a wrapper for the Eigen linear algebra library in ROS (Robot Operating System), ensuring seamless integration within the catkin build system.



(a) Image 1

Figure 1: Two side-by-side images

In UAV (Unmanned Aerial Vehicle) navigation and control, many critical data structures rely on Eigen, including:

- Waypoints: Stored as Eigen::Vector3d (x, y, z) or Eigen::MatrixXd for multiple waypoints.
- Current Position: UAV position, velocity, and acceleration are represented using Eigen::Vector3d or Eigen::Affine3d.
- Trajectory Optimization: Uses Eigen::MatrixXd for storing trajectory points and performing optimization computations.

By providing a standardized Eigen version, eigen catkin prevents dependency conflicts and ensures stable, efficient matrix operations in UAV applications within the ROS ecosystem.

3.3 Eigen Check

Eigen checks is a ROS-compatible utility library designed to validate and debug Eigen-based numerical computations. It ensures the correctness of matrices and vectors used in robotics applications, particularly in UAVs, autonomous systems, and control algorithms.

In UAV applications, eigen checks helps verify the correctness of waypoints, positions, velocities, attitudes, and control inputs, reducing errors in navigation and optimization processes. By providing robust validation functions, eigen checks improves the reliability of Eigen-based computations in robotics software.

3.4 Catkin Simple

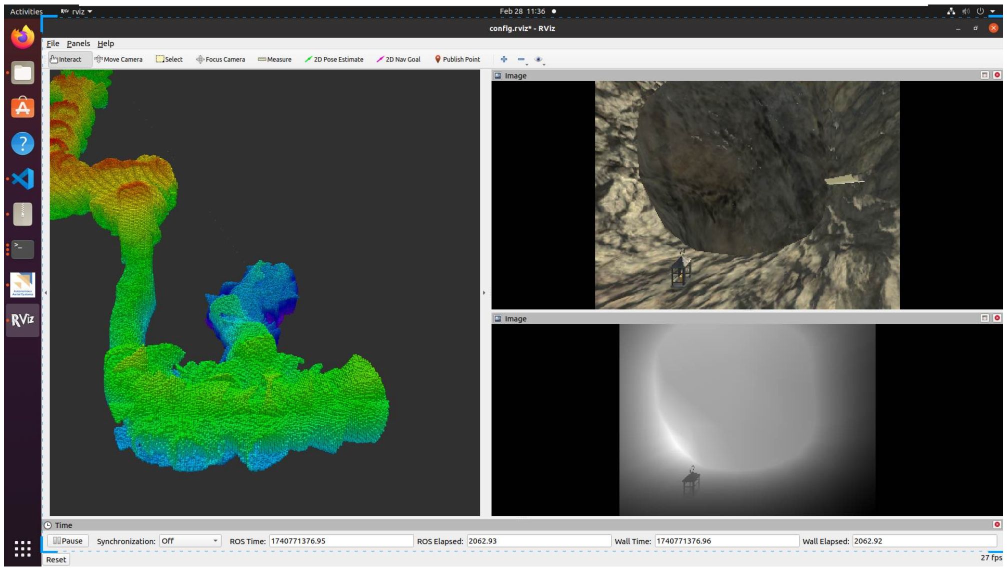
Catkin simple is a lightweight CMake wrapper designed to simplify the build process for ROS (Robot Operating System) catkin packages. It reduces boilerplate CMake code, making package management easier and more readable. Key Features:

- Simplifies CMakeLists.txt: Reduces the complexity of manually handling dependencies, libraries, and executables.
- Easy Library and Executable Creation: Provides concise functions like cs add library() and cs add executable(), replacing long CMake commands.

- Automatic Dependency Management: Automatically links required ROS and system dependencies, reducing manual setup.
- Better Build System Integration: Works seamlessly with catkin make, catkin build, and catkin tools.

3.5 ROS Noetic OctoMap

ROS Noetic OctoMap is the ROS Noetic integration of OctoMap, enabling 3D octree-based mapping and environment representation for robot navigation, UAV obstacle avoidance, and path planning. It builds probabilistic 3D occupancy grids from sensor data (e.g., LiDAR) and optimizes storage with adaptive resolution. In ROS Noetic, octomap server can be used to publish 3D maps, which integrate with MoveIt!, FCL collision detection, and SLAM for efficient environment modeling and motion planning. OctoMap remains a leading 3D mapping solution in Noetic, widely used for mobile robotics applications.



Synchronization: Off ROS Time: 1740771376.95 ROS Elapsed: 2062.93 Wall Time: 1740771376.96 Wall Elapsed: 2062.92

(a) Image 1

Figure 2: Two side-by-side images

3.6 Flexible Collision Library

4 Implementation

5 Results

6 Contributions

References

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