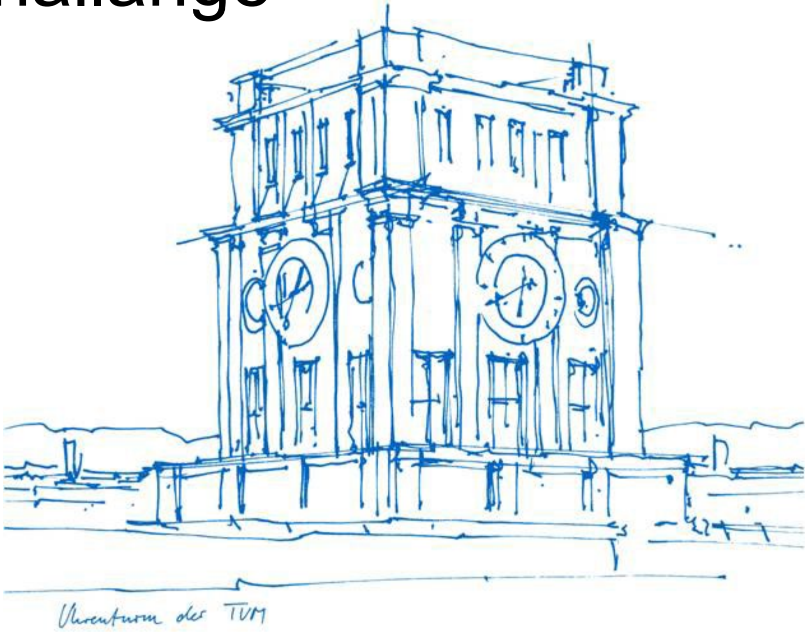
# Autonomous System:
# Group Project: Sub-Terrain Challange

Team 12

Members: Xuanyi Lin, Jiaxiang Yang, Enze Wang,

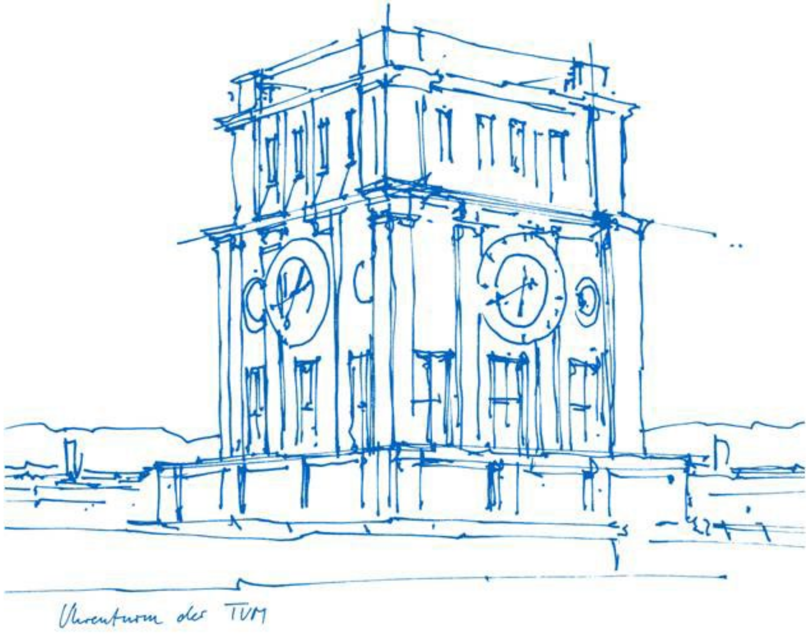
Yicun Song,  Mohammadali Rahmati

Munich, 04. March 2025



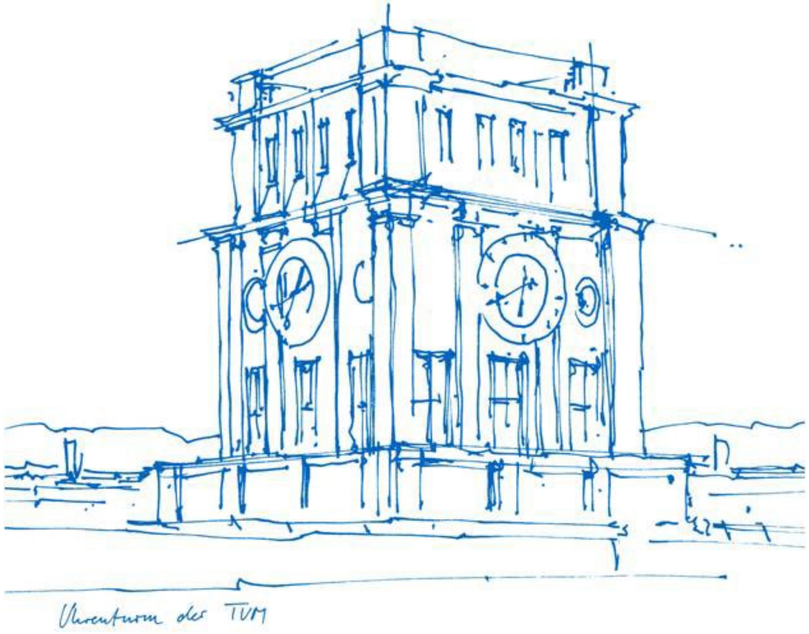
# Content

- Project Overview
- Packages
    - state_machine_pkg
    - Light_detection_pkg
    - mapping_pkg
    - pathplanning_pkg
    - trajectory_generator
- Conclusion

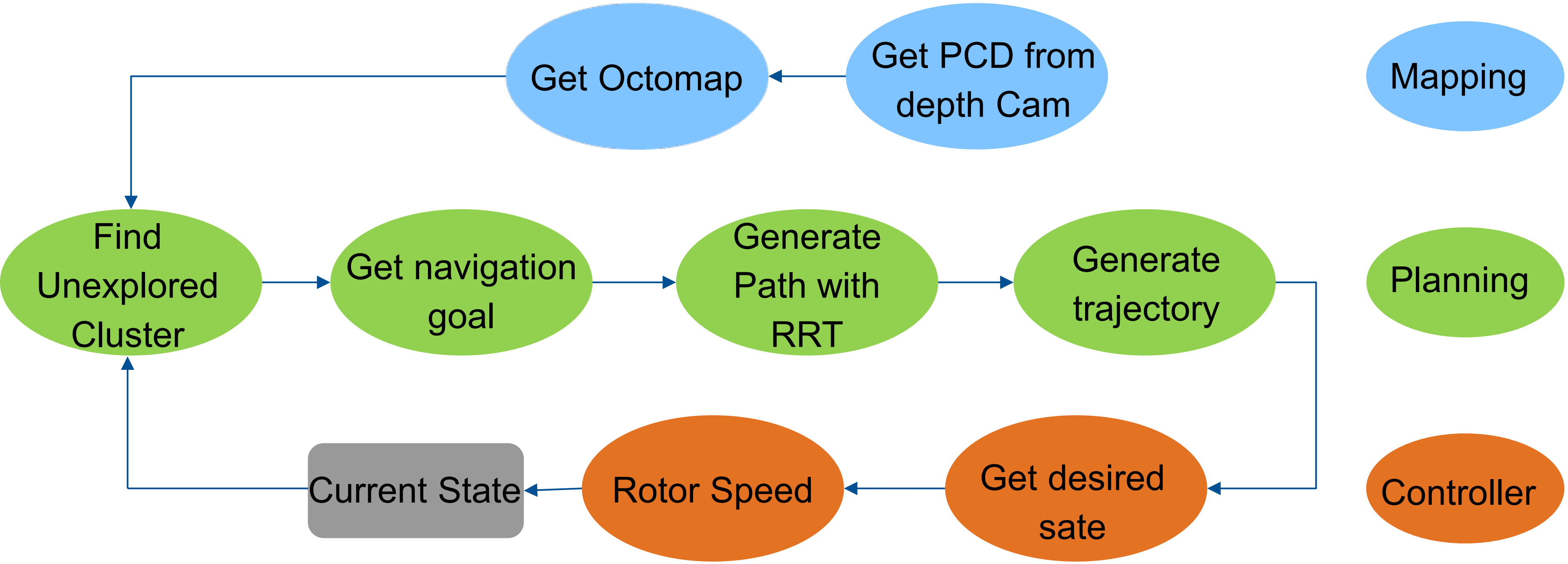


# Project Overview

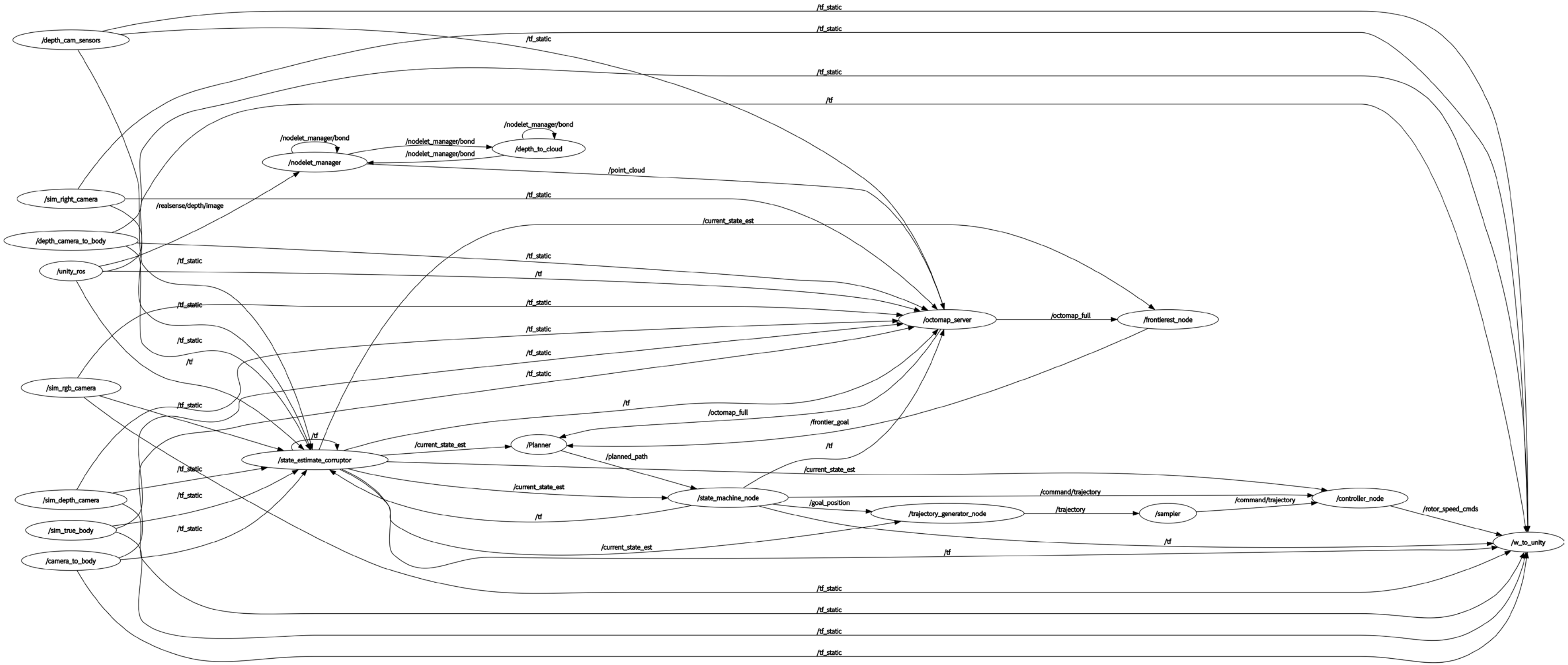
- Goal: Autonomous system operates in a cave environment
- Object detection, environment modeling, path planning
- Tech Stack: ROS + Unity Simulation
- Computer Vision, Robot dynamic, Path Planning



# System Architecture



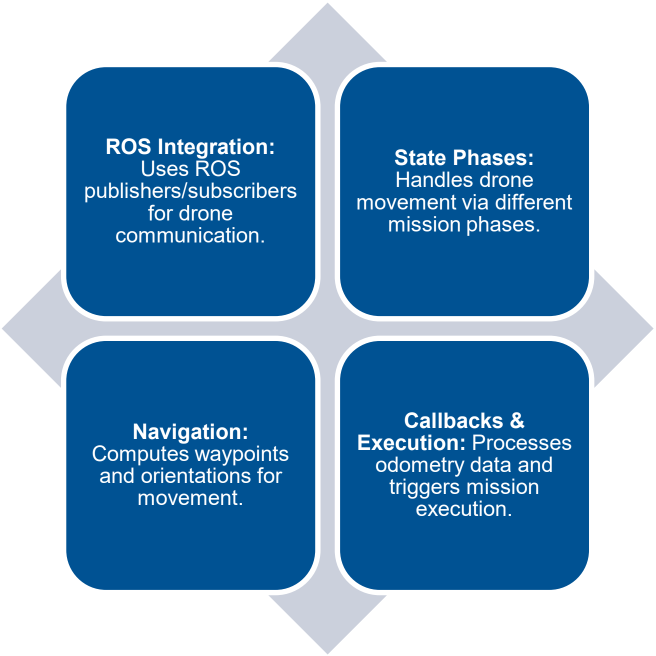
# RQT Graph


/tf_static
/depth_cam_sensors
/tf_static
/tf_static
/tf_static
/tf
/nodelet_manager/bond
/nodelet_manager/bond
/nodelet_manager/bond
/depth_to_cloud
/nodelet_manager/bond
/nodelet_manager
/point_cloud
/sim_right_camera
/tf_static
/realsense/depth/image
/current_state_est
/depth_camera_to_body
/tf_static
/unity_ros
/tf_static
/tf
/tf_static
/tf_static
/octomap_server
/octomap_full
/frontierest_node
/tf_static
/tf_static
/tf
/tf_static
/tf_static
/sim_rgb_camera
/tf_static
/tf
/octomap_full
/frontier_goal
/tf
/current_state_est
/Planner
/state_estimate_corruptor
/planned_path
/tf
/current_state_est
/tf_static
/current_state_est
/command/trajectory
/sim_depth_camera
/tf_static
/state_machine_node
/command/trajectory
/controller_node
/goal_position
/sim_true_body
/tf
/trajectory_generator_node
/trajectory
/sampler
/rotor_speed_cmds
/tf_static
/tf
/camera_to_body
/current_state_est
/tf
/w_to_unity
/tf_static
/tf_static
/tf_static
/tf_static

# State_machine_pkg

## Purpose and Key Components

- Manages a drone's mission using a structured state machine.
- Receives path data, sets goal waypoints, and executes sequential movements.
- **State Phases:**
  1. Takeoff
  2. Transit to Cave
  3. Hovering
  4. Descent
  5. Rotation
  6. Straight Movement



# State_machine_pkg

## Workflow and Execution Flow

**Initialization:**

- Define **goal points** for navigation.
- Subscribe to **odometry & path updates**.
- Publish **desired drone states**.

**Mission Execution:**

1. Fetch **next waypoint**.
2. Compute **yaw** for orientation.
3. Publish **goal position**.
4. Monitor **target fulfillment**.
5. Transition to **next phase**.

**Outcome:**

- Enables autonomous drone navigation through defined waypoints.
- Ensures **smooth transitions** between mission states.
- Uses **ROS messaging** for real-time control.

# Light_detection_pkg

## Semantic Mask Node

**Goal:** Identify the areas where lights are present and filter out everything else.

**1.Receives Images**:

1. It subscribes to two image topics:
    1. A **semantic image** (color-coded image showing objects detected by a model).
    2. A **depth image** (providing distance information for each pixel).

**2.Extracts Light Mask**:

1. It processes the **semantic image** to detect lights.
2. It creates a **mask** by filtering pixels of a specific color (representing lights).
3. This mask is applied to the **depth image**, keeping only the depth values where lights exist.

**3.Publishes a Filtered Depth Image**:

1. The modified depth image is sent as a **masked depth image**, which contains only light-related depth information.
2. This image is published so the second node can process it.

# Light_detection_pkg

## Light Detector Node

**Goal:** Convert the filtered depth image into 3D points and detect the location of the lights.

1. **Receives the Masked Depth Image**:
    1. It subscribes to the **masked depth image** from the first node.
2. **Converts Depth Image to 3D Points**:
    1. It processes the depth image to create a **Point Cloud** (a 3D representation of detected lights).
    2. It transforms this into world coordinates (real-world positions).
3. **Finds the Center of Lights**:
    1. It calculates the **center** of the detected light sources in 3D space.
4. **Filters Out Duplicates**:
    1. It checks if this light was already detected before (to avoid duplicates).
5. **Publishes the Light's 3D Position**:
    1. It publishes the **position of the light** as a PointStamped message.
    2. This can be used by other systems (e.g., a robot that needs to navigate using detected lights).

# Mapping Package

## Key Optimization Strategies

🌍 3D Point Cloud Generation (depth_image_proc)

- Converts depth images into 3D point clouds for environment mapping.
- High-precision spatial representation aids navigation.

🌍 Voxel Grid Mapping (OctoMap)

- Converts point clouds into occupancy grids for dynamic path planning.
- Supports real-time environment updates and obstacle avoidance.

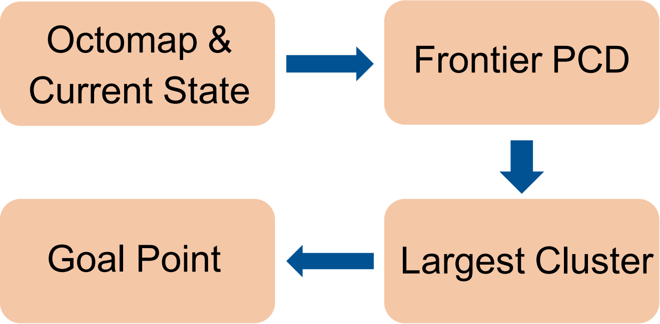
🌍 Semantic Camera for Object Detection

- Identifies key objects (e.g., lights) with reduced false positives.
- Provides semantic-level information to support path planning.

# Pathplanning Package

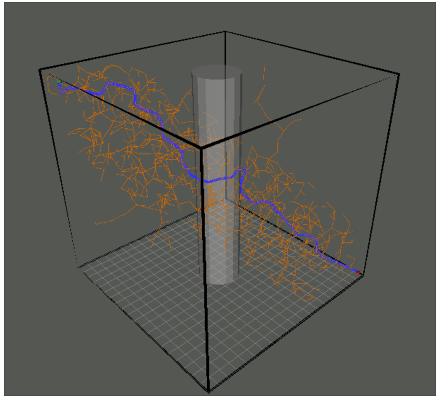
Efficient autonomous exploration in unknown environments by identifying boundary regions.

### Path Planning Workflow:




1. **Identify Boundaries & Compute Target Point**
• Use OctoMap occupancy grid and current state to detect the unexplored area's boundary.

2. **Path Planning with RRT**
•Set UAV's current position as the start and boundary center as the goal.
•Randomly sample feasible points, expanding the RRT tree while ensuring obstacle avoidance.



# Trajectory Generator

## Trajectory Generation with mav_trajectory_generation

🚀 **Objective:**

• Generate smooth, optimized UAV trajectories using **mav_trajectory_generation**.

• Ensure feasible paths that respect velocity = 30 m/s and acceleration constraints = 10 m/$s^2$.

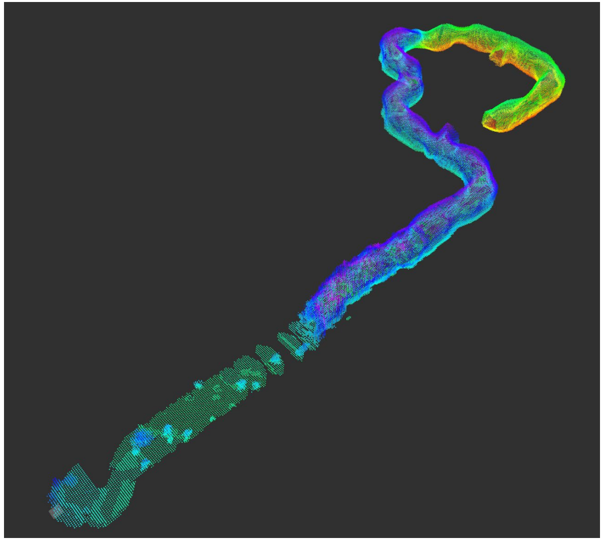
**Key Steps:**

1. **Define Waypoints:** Specify UAV positions and constraints.
   goal_position (from pathplanning) current position (from the state_estimate_corruptor).
2. **Set Optimization Parameters:** Choose polynomial degree (N=10) and derivative (order = 4/SNAP) for optimization.
3. **Optimize Trajectory:** Solve for a **minimum-snap** trajectory using nonlinear methods.
4. **Sample & Visualize Trajectory:** Extract trajectory points and visualize in **RViz**.

# Conclusion

We have completed the core parts expected in the assignment.

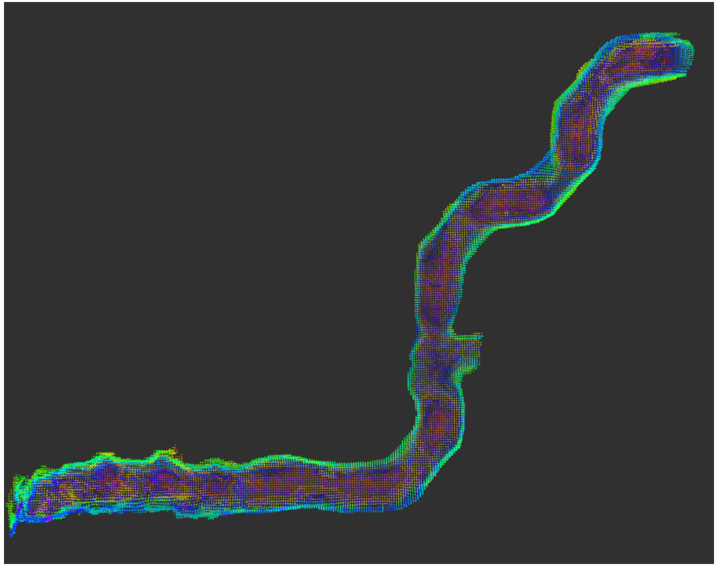
- Successfully working perception pipeline
- Successfully working path and trajectory planning
- Successfully arriving at the entrance of the cave
- Successfully finding two objects of interest
- Successfully building a voxel-grid/mesh representation of the cave environment



# Conclusion

## But we also have some details should have to be completed

- Sometime the drone can not build the full voxel-grid/mesh representation of the cave environment
- We can only find two objects of interest
- The drone have a low speed in the cave to build the voxel-grid/mesh representation.



Thank you very much for your patience