Motion Planning for Indoor Fire Fighter Drone

O. Devos 5245982
C. Espirito Santo 5557917
L. Le Ba 6291325
N. Nanvani 6140289

***Abstract*— This project develops a quadrotor drone system for emergency response in hazardous indoor environments. Using A*, RRT*, and motion primitives for motion planning, combined with PID and LQR controllers, the drone navigates efficiently to target locations. Simulations in PyBullet evaluate performance based on path length, solving time, and obstacle clearance, aiming to improve drone reliability in emergencies.**

I. INTRODUCTION

Quadrotor drones have become indispensable in emergency scenarios such as fires, gas leaks, and floods, offering a safer alternative to human intervention in hazardous environments. Their ability to navigate confined spaces and perform critical tasks makes them ideal for indoor applications, such as reaching danger zones or delivering fire-suppressing agents.

This project aims to develop a quadrotor drone system capable of navigating complex indoor spaces efficiently. To achieve this, motion planning algorithms like A*, RRT*, and motion primitives are explored, paired with PID and LQR controllers for accurate trajectory tracking. Simplifications, such as assuming a known static environment and modeling obstacles as convex shapes with safety buffers, ensure computational efficiency while maintaining reliability.

Simulations in PyBullet are used to evaluate the system in randomized building layouts, focusing on metrics such as path length, solving time, and obstacle clearance. This approach aims to enhance drone capabilities for safer and more effective emergency response operations.

II. PROBLEM DEFINITION

The selected real-world scenario involves the use of drones in emergency situations within buildings or building complexes, where drones can reduce the need for human intervention. Such applications are particularly relevant in scenarios like fires, gas leaks, or flooding, where human presence poses significant danger. Consequently, the robot morphology chosen for control is a quadrotor. The purpose of this quadrotor is to navigate from a starting point (e.g., a docking station) to a goal (e.g., the location of a fire inside a building), where it performs a designated task (e.g., extinguishing the fire by deploying water or a fire extinguisher bomb).

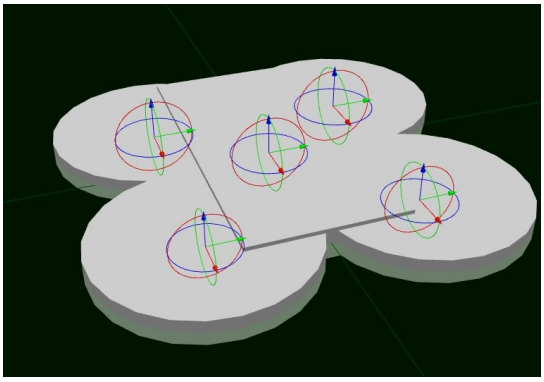

Fig. 1. URDF visualization of our drone [1]

III. DRONE MODELING AND ASSUMPTIONS

To visualize the quadrotor in the later stages within the simulation environment, the model from `gym-pybullet-drones` is employed. The propellers and aircraft are assumed to be a rigid body, and the quadrotor frame is symmetrical (as shown in Figure 1). Taking into account the model's given characteristics, as well as the well-established kinematics of a quadrotor, the following equations of motion are formulated under the assumption of keeping the drone flat:

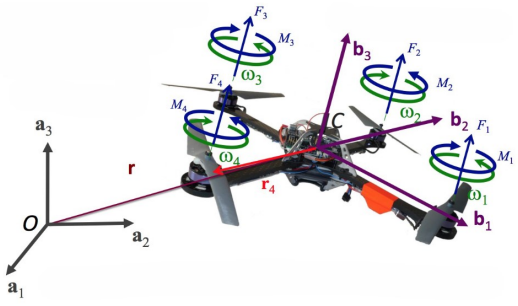

Fig. 2. FBD of a simplified quadrotor [2]

$$F_i = k_T \rho D^4 \omega_i^2 \tag{1}$$

$$M_i = k_Q \rho D^5 \omega_i^2 \tag{2}$$

For these, the variables are defined as:

- F_i: Thrust force generated per i-th propeller $i \in \{1, 2, 3, 4\}$, N.
- M_i: Torque generated per i-th propeller, Nm.
- ω_i: Angular velocity per i-th propeller, rad/s.
- k_T: Thrust Coefficient based on propeller design.
- k_Q: Torque Coefficient based on propeller design.
- ρ: Air density.
- D: Propeller diameter.

Figure 2 shows the free-body diagram of the used quadrotor and therefore visualizes the acting forces and torques which are expressed in Eq. (1) and (2).

IV. Motion Planning and Configuration Space

These equations are essential for the controller to follow the path generated by the motion planning algorithm. The motion planning algorithm plans an optimal trajectory for the robot to reach its goal via the most efficient path. In the given use case, the optimal path corresponds to the trajectory that minimizes the time required to reach the goal.

The configuration space of a quadrotor is $SE(3)$, which can be reduced to $\mathbb{R}^3 \times S^1$ via differential flatness. Here, $\mathbb{R}^3$ represents the translational position (x, y, z) and S^1 represents the yaw orientation Ψ. Differential flatness allows simplification of the configuration space by deriving the remaining variables, roll ϕ and pitch θ, from the position and yaw. Thus, the quadrotor is controlled by four independent inputs from its propellers.

V. Path Planning Approaches

Based on the assumptions of the project, we assume a fully known static map, without incorporating real-time sensor data into the planning process. The robot is treated as holonomic, and a holonomic path planning approach is chosen. To select the most suitable route, three different path planners will be explored and compared:

1) **A* Algorithm:** A graph-based planner known for its speed and optimal nature.
2) **RRT*:** A sampling-based method effective for handling high-dimensional spaces efficiently and optimally.
3) **Motion Primitives:** A combined approach using precomputed motion primitives for generating feasible trajectories.

VI. Control Strategies

Compatible controllers are essential for each planning approach:

- **A*:** The planner generates a discrete set of waypoints that are smoothed before tracking. A PID controller is used due to its simplicity and low computational intensity.
- **RRT*:** A feedforward-feedback controller such as LQR is applied to efficiently track paths in real-time.
- **Motion Primitives:** LQR is also used for tracking the dynamically feasible trajectories produced by motion primitives.

VII. Obstacle Representation and Simplification

To ensure effective trajectory planning, the obstacles within the environment are appropriately constrained and represented. Structural walls are considered fixed obstacles, while furniture and other items are overestimated as simple rectangular convex shapes. This simplification reduces computational complexity and facilitates node and vertex generation for motion planning algorithms. A buffer zone is added around obstacles to enhance safety and reliability.

VIII. Performance Evaluation

A custom loss function will be developed to validate performance. This function includes:

- Solving time (path generation).
- Execution time (time required for simulation).
- Path length.
- Weighted average distance from the closest obstacle at each timestep.

Recording these metrics provides a quantifiable basis for evaluating the effectiveness of different approaches.

IX. Simulation Environment

The robot operates in a simulated environment consisting of multiple buildings with various floor plans. The task involves flying to a designated emergency location. The assumptions include open doors and windows to avoid additional complexity. The environment will be randomized for evaluation. The PyBullet library will facilitate physics simulation and visualization.

References

[1] J. Mmyself. Ros_ardrone/ardrone_launch/launch/drone.urdf. https://github.com/mmyself/ros_ardrone/blob/master/ardrone_launch/launch/drone.urdf. GitHub repository.

[2] Sihao Sun TUDelft. Mav modelling control - ro47001. Guest Lecture.