Autonomous Quadrotor using RRT with Constraint

Dejun Guo, Xiang He, Shantnu Kakkar, Roya Sabbagh Novin

Introduction

Nowadays, unmanned aerial vehicles (UAVs) are applied to various tasks, such as search and rescue or surveillance. Quadrotor have been studied for use in such tasks due to the advantage in control that multiple rotors gives. Because of this, quadrotor can hover and through complex environments that traditional UAVs cannot navigate, for example in a damaged structure. Many problems need to be dealt with to achieve the autonomous navigation for quadrotor in such tasks. How to get to the goal in a smooth path and avoid the obstacles as well as the dangerous area is the crucial issue we will focus on in this project.

Firstly, a map will be built based on an indoor environment which the user can edit through an interface. Secondly, a set of way points will be generated based on bi-directional goal-biased RRT algorithm. Since this path might not be the shortest, we will apply a post-processing algorithm based on line-of-sight segmentation, to further shorten this path. Finally, a path smoothing algorithm based on 4th-order B-spline curves will be applied to make the path agile and dynamically-feasible (that is satisfy velocity and acceleration constraints). This smooth final path will guide the quadrotor from the starting point to the goal while avoiding the dangerous area and the obstacles.

1. Map Generating and Visualization

Map generating will be done in MATLAB, using MATLAB GUI. A separate program will be created with GUI to help user drawing a new map easily, including map resizing, basic line, polygon, and circle element representing obstacles in the map. Map is stored temporary as picture to help user reviewing it. Then a cost map will be asked to point dangerous area on the generated map picture. After these two steps, three map files will be generated, a high resolution map, a lower resolution map and a cost map which has danger costs centered at the point user just pointed out. Map generating is done in MATLAB by reading the map picture.

3D visualization is built in ROS with OpenGL. Visualization node reads the stored high resolution map file and represents it in 3D. Playback node will publish the real-time position and orientation of the quadrotor, based on planning result. Visualization node subscribes the localization and trajectory data and visualizes the quadrotor along the trajectory towards the goal.

2. Path Planning

In this project, the aim is to solve a collision free Kinodynamic motion planning problem of a quadcopter, which autonomously navigate inside an environments with dangerous zones from a start location to goal. By Kinodynamic planning, we are referring to problems in which motion planning must satisfy non-holonomic and/or dynamic constraints.

The path should avoid collision with obstacles but it is allowed to pass dangerous zones. However, this should be avoided as much as possible. For this purpose, first, we need to express the concept of danger more specifically. For instance, in the case of this project, areas with high altitude can be dangerous for the quadrotor. Next, some extra cost can be defined for these zones regarding the level of danger.

Another contribution which will be considered in this project is implementing an algorithm for narrow passages, such as doors or windows in a closed environment. That will be solved using OBPRM, Gaussian sampling, bridge planner or similar techniques.

3. Path Smoothing

There are two requirements when deciding which curve function to use for the path smoothing function. The first is Local Support. When fitting a curve to the way-points, there are cases when the generated curve intersects with the obstacles. However, it is normally observed that not whole of the curve, instead only a part of it gets intersected. Hence it is logical to think that we need to reshape only that part of the curve which collides with the obstacle. To achieve this, we need a curve that can be reshaped only at certain regions. After a thorough literature survey and feasibility study, it sounds like that B-spline curve meets this requirement. The second requirement is the need of continuity of velocity, acceleration and radius of curvature throughout or path. Since, 4th order B-spline curves are c2 continuous, they can be used to satisfy velocity, acceleration and radius of curvature constraints. Another advantage of choosing B-spline curve is that they can be easily implemented using De-Boors recursion algorithm, thus not adding much of any overhead.

For implementing this algorithm, we will add Pseudo control points to the way points that we get after the post-processing step. There will be cases when addition of Pseudo control points won't be sufficient to avoid collision. In those cases, we will re-position the way-points which are responsible for the collision.