28/10/2018
Zurich, CH

Team Name: ELCA$H
Challenge #1: ELCA Swarm Drone Delivery

Solution Description

1.1. General overview - Describe quickly what ideas you realized and what «features» your algorithms cover.

We focused on simple path-planning algorithms for the drones. One is Dijkstra's algorithm for "finding the shortest path between adjacent nodes in a graph".[1] The second algorithm utilizes potential fields in order to direct the drone towards its goal destination. The parameters of each algorithm were tuned experimentally to compensate for the limited control over the drone, and the noisy sensor measurements of the position. In the end, we had limited success in navigating the drone to all waypoints and back to the "home" location. Without peripheral sensors for collision detection, real-time path-planning and collision-avoidance algorithms are virtually impossible to implement.

1.2. Challenges - Describe what challenges you faced and how you solved them.

Our main challenges can be divided into two groups: disturbance and inaccuracy on the system, as well as path planning algorithm .

The first issue was the most problematic to deal with since the noise of the system made the localization of the drone unreliable and hard to control. These disturbances originated from the carpet texture interacting with the laser range finder, and the noise in the loco position node. Giving that there was no solution to eliminate these disturbances completely, the team focused on developing a dynamic path planning to try to reduce the effect of these noises. Furthermore, after several test runs, the software algorithms were tuned to the most-optimal performance given the constraints.

The second challenge the team faced was to find a reliable and fast path planning algorithm. In the beginning, the team implemented a simple graph search method (Dijkstra's algorithm) in order to prove that drone delivery was possible. Once this was accomplished, the team moved to a more advanced and faster algorithm (potential field planning) to deal with obstacle avoidance and reduce the effect of disturbances. Even though the later algorithm worked to find the optimal path to the desired position, it failed to avoid other drones while doing so. As such, the team decided to return to our simple path planning to ensure the drones successfully deliver its parcels.

1.3. Algorithms - Describe the algorithms you used or developed and how they work.

[1] https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm

Dijkstra's algorithm iteratively searches its area for a node that minimizes its distance traveled to the node corresponding to the "goal" node. Therefore, the path traveled to the final node is optimal in some sense. The second algorithm utilizes a potential field method to avoid obstacles in its environment. A potential field is a vector field of gradients of potential energy. The gradient of this "energy" is characterized as an attractive or repulsive force that "pulls" or "pushes" the robot away from a potential well (extending to infinity for obstacles, which are considered to apply a positive "force" to the object navigating its environment, and extending to minus infinity for the goal points of the algorithm). The net effect, therefore, is that the object (a drone, in this application) is pulled towards a well of infinite negative energy. In order to consider the uncertainty in the position of the obstacles relative to the robot, the diameters of the potential wells were increased to encourage the drone to allow more space between itself and the obstacle around which it was navigating.

1.4. USPs - What makes your solution unique?

The most valuable feature of our solution is the integration of the Robot Operating System (ROS) as our framework for writing the drone's software. ROS allowed us to have a collaborative environment that covered most of the common messages and functions of drones. Furthermore, ROS benefited us from not having the need to run on the same system or even of the same architecture, which made communication from the server and our functions easier.

It is important to note that once the drone's framework was set up with ROS, any further implementation to the drone (adding new sensors or actuators) can be easily handled by adding the respective ROS libraries. In addition, any future technological improvements in the field of autonomous drones (or even specific to parcel delivery) can also be easily implemented by just adding these ROS libraries to our prototype.