MRSD Project 2

Individual Lab Report #10

Parv Parkhiya

November 07th, 2019

Team H:
PhoeniX
Teammates:
Shubham Garg
Akshit Gandhi
Zhihao (Steve) Zhu

Individual Progress

The first task for this progress review, I worked on was streamlining plane detection and opening detection pipeline. In the last progress review, I merged my plane detection code with Shubham's opening detection. But merging was done by simply creating an instance of LaneDetection class inside OpeningDetection class. While both of them were using the same pointcloud for processing, there were 2 separate pointcloud callbacks. I properly merged both the code by using just one callback and carefully merging the private variables of both the classes so that we don't use duplicate variables storing the same thing. This task was pretty straight forward and went smoothly.

The next major task for our entire team was to integrate various software packages to a central entity to plan a sequential mission. We decided to use the behavior tree framework for the same. The behavior tree framework is an alternative way of doing the same thing as the state machine with a notable difference. The state machine is structured to jump from one state to another based on the input without keeping track of from which state it has come from. (Similar to "go to" statement in programming). But for the kind of mission, we would like to do, we want to return back to where the call was made. (Similar to function callback in programming)

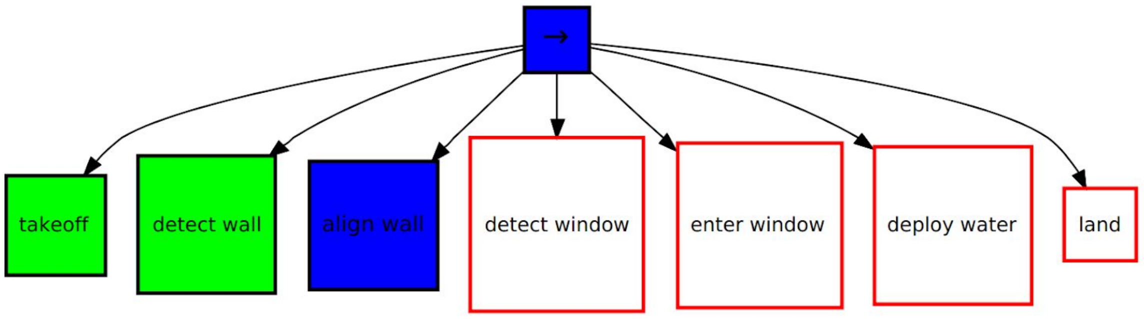


Figure 1: Behavior Tree for UAV mission

Airlab's core stack has the code implementation of the behavior tree framework but it's designed in a self-contained passive manner. In other words, while it jumps from one action to another action, it doesn't make the call to the respective node rather it's the responsibility of the node to listen to the particular topic and tell the behavior tree that whether it's running, it has succeeded or failed. I designed the overall mission in the behavior tree framework which was a straightforward task. The behavior tree used can be seen in figure 1. Major work was updating on the nodes. Akshit and I choose different nodes to make it compatible with the behavior tree framework. Specifically, I worked on modifying the wall plane detection node and opening detection node to interface with the behavior tree.

I also worked on finishing the attachment of the extinguishing system to the drone. Connecting various pipes and solving the water dripping problem by creating small cork for the nozzle. The final state can be seen in figure 2.

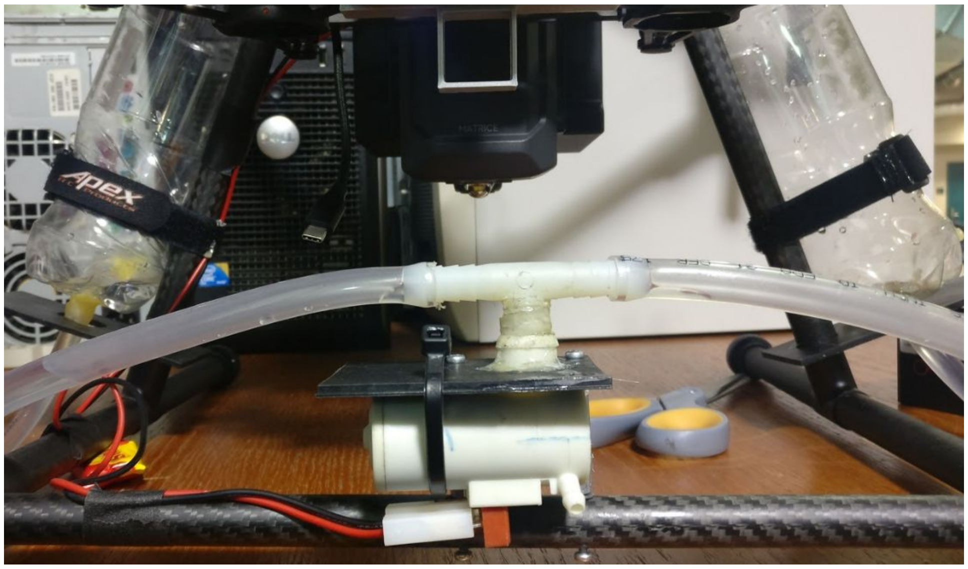
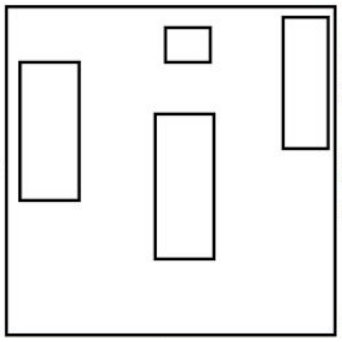
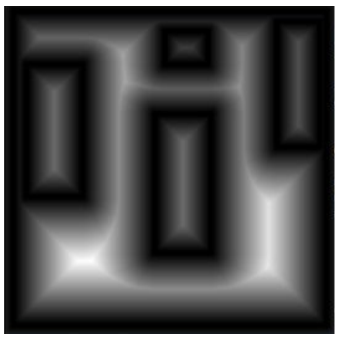


Figure 2: Extinguishing system attached to UAV

Finally, I worked on the global pre-planner to generate preliminary high-level paths for the UAV-AGV system. The idea here is that since the mission (at MBZIRC competition) will start from outside of the building, the system first needs to reach the building and also it needs to maintain some distance between them to avoid the collision. While we aspect the space to be free apart from the building, there could be a blocked region in the arena where we don't want our robots to go. The global pre-planner takes an abstract layout map of the area with start position and goal location (which is the building) and generate optimum paths that are within the arena and are the safest distance from the known structural obstacles. Initially, I was thinking of using RRT based method to get path but I realized that paths generated by RRT would be non-optimum and non-straight. While there are methods to correct that we decided to do uniform discretization and use the A* algorithm to get the optimal path that is not fluctuating.

I coded the A* algorithm from scratch in C++. A lot of effort was also on reading the map from the .png file using the OpenCV library. I also added the functionality of using distance transform as a cost in the A* to incentivize the paths that stays an optimal distance away from the obstacles. The heuristic for A* is chosen as the L2 norm between the current location and goal location. The final result for the global pre-planner can be seen in figure 3.


Input Map


Distance Transform

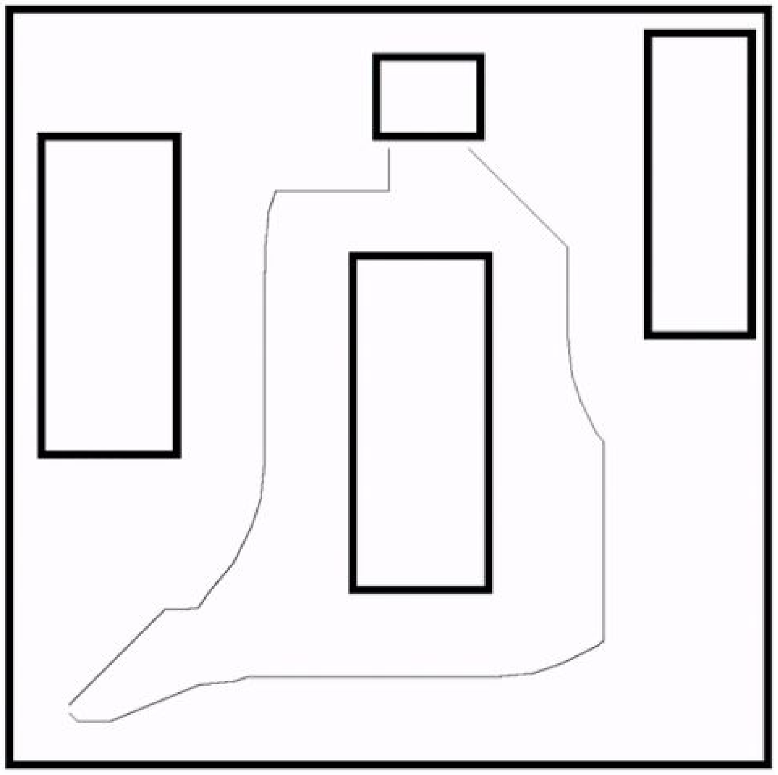

Map with optimal paths from start to goal

Figure 3: Global pre-planner output

Challenges

One of the major challenges I faced was regarding the behavior tree. The behavior tree framework expects that each node returns its status (say Running) at high frequency. But nodes that I was handling where processing pointcloud and the code takes a couple of seconds to run. This delay makes the behavior tree think that the node is no longer active. As a solution to this, I created a separate ROS callback that would publish the node status. However, it didn’t solve the issue. After some debugging, I realized that ROS internally treats various callbacks synchronously which means that all the callbacks of a node wait till the slowest one finished its processing. But we want the status callback to run at a higher frequency so that the behavior tree doesn't time out. I was finally able to solve the problem by using an asynchronous ROS callback mechanism where each callback runs on multiple threads without waiting for other callbacks to finish running.

Teamwork

Akshit and I worked together to better understand the behavior tree and creating a pipeline between various nodes. We also performed multiple mission flight tests together until the window fall on the drone breaking the last set of propellers. Shubham, Akshit and I with the help of Kevin, reattached the UR5 arm on Husky, along with all of the electronics and powering systems. Steve and I set up the environment during various mission test. All the team members helped in setting up the tent which we might use in the future specifically in outdoor testing

Future Plans

I would be working on helping Shubham in writing the behavior tree for Husky. I would also work on improving the opening detection since the current system still gives false window detection sometimes. We also need to perform more flight tests but that would only we possible once new propellers arrive. I will also look into fixing the water pump powering problem and the potential solution for the same. As a team, we need to work on fixing the remaining problems and start performing the FVD mission and make it more robust and reliable.