16682-A MRSD Project 2 | Individual Lab Report # 10 November 7, 2019

Paulo Camasmie | Team D – CuBi

Teammates: Jorge Anton, Nithin Meganathan, Changshen Bobby Shen, Laavanye Bahl

Individual Progress:

My work was focused on developing a point-to-point path generator for CuBi. So far, our team had developed a cell decomposition of the acquired environment map and implemented a centroid-by-centroid total coverage exploration. However, unlike a lawnmower, CuBi interrupts its exploration, as it finds toys that need to be collected and returned to the base. After they toy is returned, CuBi needs to resume its exploration and toy finding (toy here exemplifies any small object or clutter on the floor). Our code had no provisions to generate those intermediary paths efficiently.

My implementation involved writing code from scratch in Python. I wrote two classes that would be easily integrated as individual components of software topology.

The Class genGraph, uploads the map image, received from the LiDar, Fig.1—with values representing the centroids of the composed cells, free areas, walls and obstacles—and turn it into an occupancy grid., Fig 2. It then runs a graph building algorithm that will for each node find all nearest neighbors and build a diction of nodes as keys and accessible neighbors representing edges.

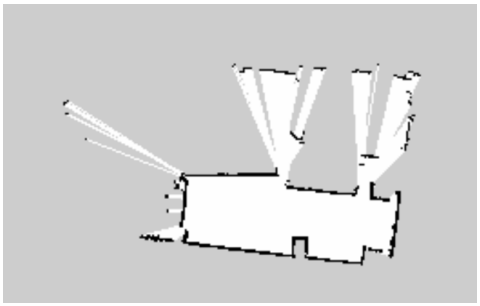

Figure 1 LiDar image of CuBi's test area

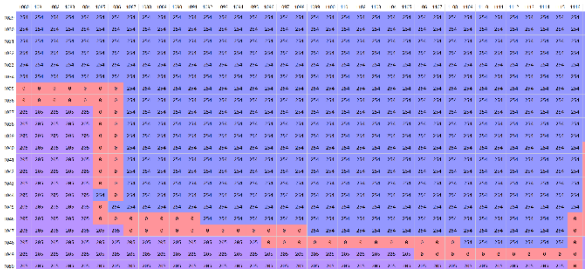

Figure 2 Occupancy grid generated from image

The Class genPath will use the generated graph and find the shortest path between the current CuBi's location and the goal location. Considering the small area that we are using right now, a simple breadth-first-search seemed appropriate. I also wrote a simple simulation in matplotlib to show a few scenarios where these new classes can be useful, as shown in Fig 3.

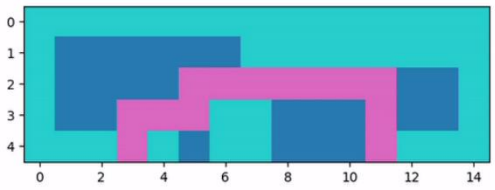

Figure 3 CuBi's path generated with Class genPath circumventing obstacle

Risk Mitigation

- I would like to test the algorithm extensively on actual hardware. Also I would like to try it on on several edge cases both in simulation and hardware. Also I want to make sure that the algorithm is scalable for larger areas.

Individual Challenges:

- I had a few challenges, since initially I was under the impression that my implementation would receive the occupancy already created. Instead, I receive an image and had to implement the complete pipeline, from image to path generation. Also generating the graph automatically was slightly challenging but I wanted to implement that, since I consider a powerful tool to have and will be useful for any current and future maps gather by our sensor, off-line or online. Once the graph is generated, searching is a solved problem and any algorithm will probably work, but I wanted to implement it from scratch, which was challenging since I had to refresh some skills, and dig into some past notes, but it was very rewarding.

Individual Next Steps:

- Path generation: Continuing the work above, I will add to the path generator two separate policies, one for when CuBi has collected a toy and it goes to the tray only by moving through cells already explored and devoid of objects on the ground. It will have a second policy for after CuBi delivered the toy and will resume its operation, where it can go through areas not cleaned yet but by following the shortest path. This will require a good integration with our state machine to know what cells have been explore, cleaned or partially cleaned.
- Going back to my previous work where I adjusted CuBi's controls, even though that worked really well in isolation, during the presentation today, I noticed that CuBi was overshooting its position and backtracking a little toward the centroid of the cell. I will re-adjust those gains to see if they are too aggressive, or if I should add a derivative component to the PD control, or if there is simply some other component in our code causing that.
- I will be working on the motion of the manipulator to complete all movements more fluidly and simultaneously

Team Progress:

Laavanye implemented robust object detection and showed that different objects are classified by its size. All objects are bounded by a virtual box, large objects not pickable by CuBi are shown in red, small ones that need to be picked are shown in blue. He also improved the object picked validation, now using the depth sensor.

Jorge implemented an algorithm to automatically segment, ads corners and outer bounds, and perform cell decomposition with centroid locations on the image of the test area. This can work in an area explored by CuBi where a map is generated by our SLAM sub-system.

Bobby implemented a coverage exploring policy for CuBi. This was successfully showed today. He also added a connection to rviz in ROS so that we could validate that CuBi has indeed covered the whole are.

Nithin explored several packages and methods for our object classification pipeline. He tried to use OpenCL Caffe [1] backend but faced challenges (below), and decided to stay with point cloud object classification by size.

Team Challenges:
Laavanye is still experiencing lots of issues with object detection considering our new test area

Bobby worked very hard to use two separate packages successfully to build a map and than to localize the robot.

Jorge had to dedicate considerable time to work with me and then with Laavanye to help integrate our codes into the state machine, since he was the most familiar with it. He also put a good effort into researching different exploration strategies.

Nithin found that he was not able to run OpenCL Caffe on Nvidia GPU since it is optimized for Intel GPU. Most Deep Learning packages are for 2D object detection/recognition and so depth information of the object is lost.

Team Next Steps:
Laavanye, and Nithin will add new light sources to our test area to try to make it more light invariant and make our object detection more robust.

Jorge and Bobby will continue working on the integration of our exploration policy. They will also try to make our code more robust.

As a team, we will try to complete the integration of our CuBi's whole pipeline so that we can validate it extensively and improve each component but in a holistic sense, for a successful FVD and completion of our project.

References:
[1] https://docs.opencv.org/3.4/d5/de7/tutorial_dnn_googlenet.html