**16682-A MRSD Project 2 | Individual Lab Report # 11 November 19, 2019**

**Paulo Camasmie | Team D – CuBi**

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

## Individual Progress:

Following up from my previous work of ILR#10, where I developed a point-to-point path generator for CuBi, I was requested by my team to add an important feature to my algorithm. There was an issue where CuBi was making more turns than needed to go from point A to point B. Since CuBi is a non-holonomic vehicle, care should be taken to minimize the amount of turns during its path. Eventhough the Manhattan distance (norm 1) would always be the same in a grid workd, our team had two concerns: 1) CuBi will take longer if the amount of turns are not minimized, 2) CuBi, with its long span, would tend to crash on walls if it spins unecessarily during travel. A good example can be seen in Fig 1, where CuBi path is not minimized for number of turns.

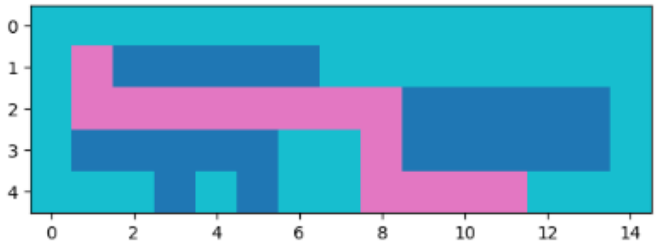


*Figure 1 Naïve path planning implementation, start row 1 x col 1, CuBi turns prematurely at 2x8 position causing one extra turn*

My idea was to simply change the search algorithm from Breadth-First-Search, to Dijkstra [1] where I could assign weights between nodes. A lower weight would represent a linear travel step, and a higher weight could be assigned to a change of states where both a turn and a linear travel step were required. The main difference between the two implementations is that on Dijkstra a priority queue would be used, instead of a standard queue.

However, I quickly realized that while a change to the search algorithm would be simple, this new state space and its graphical representation would be much larger. The reason is that we would need one node for each state and turn combination. While possible, this created unneeded complexity and would slow down the search in case we want to scale up our system.

It occurred to me that the cost of time of one turn, should be similar to the cost of the linear travel between two adjacent grid points. Also, I realized that CuBi should only turn right, turn left or go straight at each transition. I decided to use that structure and knowledge to our advantage and simplified the algorithm by only changing the state representation instead of the search algorithm. Fig 2. clearly shows the simplicity of this approach with a state and its possible transitions. In this case, the edges are un-weighted, and we could use the same algorithm as before, while keeping a small state-space.

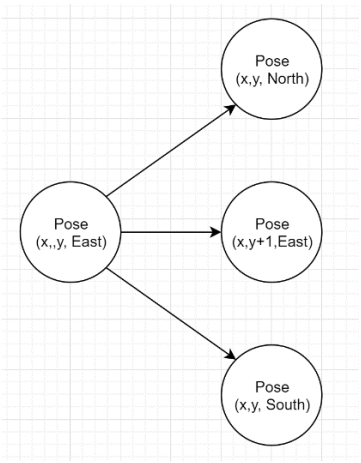


*Figure 2 Graph representation of CuBi state, in this case facing East. Notice options, =*

As a result, CuBi is now able to accomplish the same path, minimizing the number of turns required, which greatly expedited CuBi's work, Fig3. I implemented my code as a python module that could be imported into Jorge's code for CuBi's global exploration planner. I also added a feature so that the module outputs a path with explicit between states, including the orientation. That way Jorge could use this output directly to command CuBi to either move forward or turn in each direction at every time step.

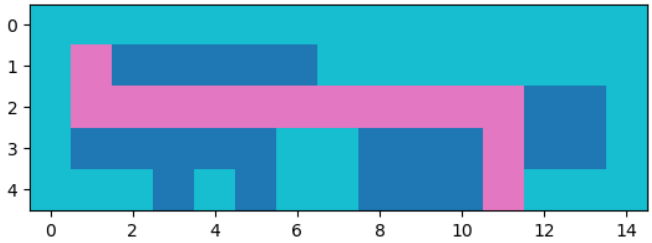


*Figure 3 CuBi's path planning of the same initial and end goals but with minimized number of turns*

**Risk Mitigation**

One of the areas that I feel is most risky—as mentioned in previous ILRs—is the potential for 3D printed parts to break. This week, it finally happened Fig4. With the the shoulder joint snapped. Luckily my team was quick to come up with a temporary solution Fig 5. using the remaining attachment points to connect the shoulder joint to the MX106 Dynamixel servo. I went ahead and worked a permanent solution making a new joint with finer structure to be replaced by FVD, Fig 6. and will inspect other joints and potential points of mechanical failure.

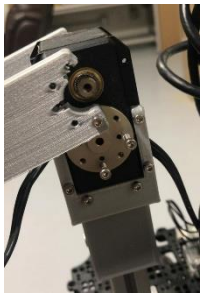

*Figure 4 broken bracket*


*Figure 5 Temporary fix*

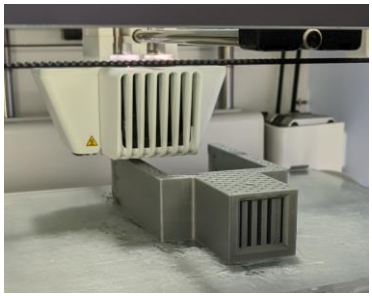

*Figure 6 Permanent fix and risk mitigation*

**Individual Challenges:**

- I spent a substantial amount of time adapting the new search algorithm. As simple as it looks and sound, I wanted to provide my team with a professional, clear and efficient implementation where the task of generating a point-to-point path was completely abstracted away and where step-by-step instructions would be received in return. It was challenging but also rewarding to provide that level of implementation.

**Individual Next Steps:**

- While visiting NREC two weeks ago, I noticed that a mobile base that was moving autonomously through the main area, would stop automatically as people walked close in front of it. I discussed that implementation with the responsible engineer who gave me some pointers of how to use the lidar data and the occupancy grid to achieve that. Eventhough we have descoped that, this is a very important feature to have in a childcare environment or in people's home. I am currently in the SLAM class and volunteered to implement that feature. Hopefully by the FVD we will be able to show it.

**Team Progress:**

**Laavanye** spent a lot of work improving our object detection considering the line conditions.

**Jorge** integrated my point-to-point path into the global exploration pipeline

**Nithin** worked in the drop-off object section of our pipeline

**Bobby** worked tirelessly into integrating and fine tuning the complete CuBi software pipeline.

**Team Challenges:**

**Laavanye** is still working very hard to eliminate false positives from our object detection algorithm.

**Nithin** helped in the efforts to mitigate that issue and came up with creative ways to seclude areas where the problems were most prominent. He also faced issues properly fine tuning the drop-off of toys since the tray would touch too low into the box, or not align properly with the box.

**Bobby and Jorge** worked as our master integrator and he faced several issues since other members implemented new features into our pipeline, new edge cases surfaced that they had to address and fix. They were able to finalize our demo 10 minutes before due time.

**Team Next Steps:**

**Laavanye** will finalize object detection robustness

**Nithin** will change the drop-off tray angle to 45 degrees instead of the current 90 so that they tray will not hit the ground, or have the risk of re-grasping an object and CuBi will be able to drop-off toys farther away from the tray which will help with not bumping into the box during the rotation reset step.

**Jorge and Bobby** will finalize the integration of our exploration policy

**As a team,** we will wrap up, perfect and validate our integration for CuBi's complete pipeline and put a good show to be proud of at the FVD and completion of our project.

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