16682-A MRSD Project 2 | Individual Lab Report # 8 October 10, 2019

Paulo Camasmie | Team D – CuBi

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

Individual Progress:

When I initially conceived and presented CuBi's idea and design to Professor John Dolan, he advised me to adopt a "caging" strategy for scooping up objects from the ground. I came back with a double paddle design and a tray. The idea was to apply a "pinball" style control, with alternating paddle movements to properly position the object with respect to the tray, and validate its pose with perception, before bringing it in. Several lengths of paddles were tested in the past, Fig 1.

For sake of simplicity though, our team decided to go with a half-tray-width paddle length and a simultaneous paddle closure. This seems to work well, however the trade-off is that sometimes an object gets trapped between the two paddles. The most common failure is for a large, flexible, and round object to get symmetrically trapped between the two paddles, Fig 2. The more uncommon issue is for the object to be trapped "in between" the two paddles, Fig 3.

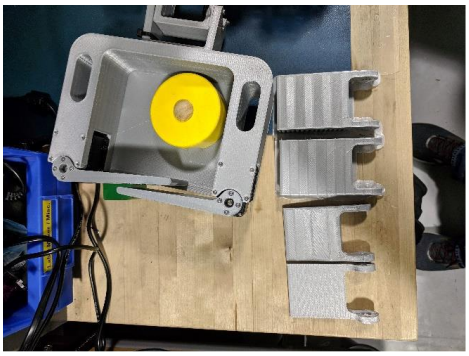


Figure 1 Paddle design options. Shown overlapping paddles for "pinball" action

Therefore, my focus during these past two weeks, has been to address this issue by detecting such failure and introducing a Failsafe routine in our State Machine. The first part was to develop a strategy for the issue. I considered using perception, however before that I investigated the ROS topics available to me, by using the rqt_graph command while running CuBi. Going over the Dynamixel reference I found that these motors do provide feedback information[1] . I then "echoed" the joinState topic and was pleasantly surprised to find a "load" feedback that reacted well to slight pressures applied to the paddle.

It can be seen by the same figures 2 and 3, the respective paddle feedback while an object is being trapped between paddles. The small amplitude curves show the motor torques, expressed as normalized loads, for operations when the object was successfully brought into the tray. It then became clear that any spikes with absolute values over 0.5 were an indication that an object was trapped for both scenarios.

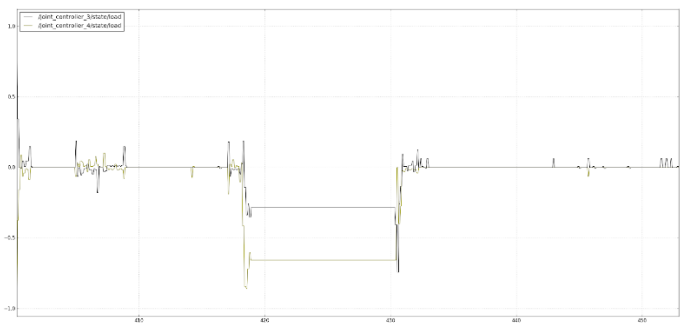
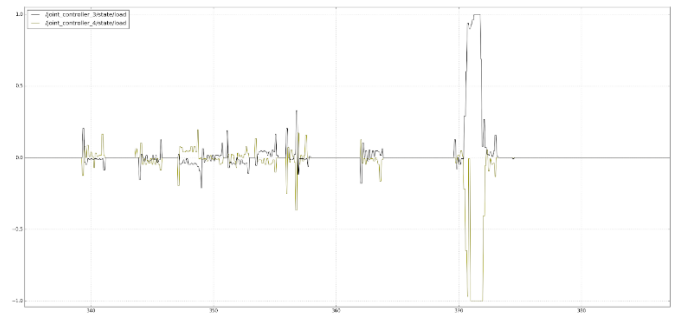


Figure 2 Double paddle jam



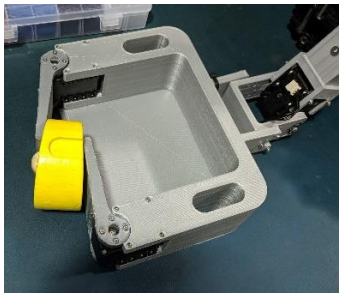


Figure 3 Opposing Paddle Jam

With the strategy and information in place, Bobby being more familiar with our current software architecture, worked with me to actual implement the code into the State Machine. We first created a subscriber to the jointState topic and wrote a callback function to check if the current load was over a torque threshold. If so, we used the existent functions in our State Machine to simply toggle the paddles open.

I then updated our State Machine by creating a new state called "Back Off", Fig 4. I modified the code to switch states when the callback function above indicated that an object was trapped. The paddles are instructed to open, and the robot is in a failstate mode. In the State Machine loop, I added a routine that if the machine was in the "Back Off" state, to call a function that backs the robot off 10cm, and then switches states back to "Toy Search" state. For now, this is a simple heuristic that gives the robot a chance to try to pick the object again, possibly from a different angle.

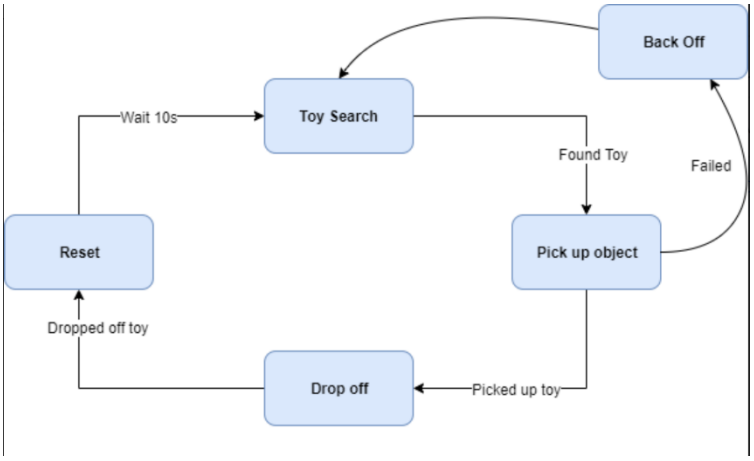


Figure 4 Updated State Machine diagram with Back Off state

Risk Mitigation

- Addressing some of the issues raised in my previous ILR, I went through the mechanical assembly of CuBi, and tighten many of the mechanical joints of the robot. There is still room for improvement in that area
- I am a bit concerned with the robustness of our State Machine, since we are dealing with a lot of asynchronous ROS messages, I think it is important to validate the robot behavior more truly and on various scenarios

Individual Challenges:

- My biggest challenge was interpreting our code and software architecture. Since I had mostly ben focused on the hardware side of our project, I was unfamiliar with some of the idiosyncrasies of our code. I found many disparate files, in different formats and package locations, that needed to be launched separately to make CuBi work. I also found more dependencies than I envisioned. I asked our team to document a software architecture showing how all files were related, and a State Machine diagram. With that information at hand and a couple of paired programming sections, I find myself now well equipped to contribute further to the code.

Individual Next Steps:

- Further validate the new "Backoff" state of the state machine, making sure that it had not brought unintended consequences to the overall behavior of the robot
- Work on the gains of the robot mobile base and manipulator. Also work on combining multiple sequential movements, into simultaneous actuation, with the objective of speeding up the overall behavior of the robot, for a more fluid motion and time saving operation

Team Progress:

Laavanye implemented a failsafe mode for the robot to validate if the robot was successful in grasping an object. He implemented a routine that takes an image of the tray during the robot booting, and then for every subsequent grasp, the robot would take an image of the tray and compare a histogram of colors (pixel values) against the empty tray. In case the histogram is similar to the empty tray, then the robot failed to grasp, and the State Machine would switch to the "Backoff" state and subsequently search for toys again

Jorge worked with Bobby developing an exploration strategy, where a given map with walls and static obstacles would be decomposed in cells. Each cell's centroid would be considered a way point, through which a global path would be generated. As the robot follows the global path—since each cell is small enough to be contained in its field of view—the robot would run a local planner to grasp each toy in that cell before proceeding to the next cell. **Bobby** also collaborated with me on the failsafe grasping as described in this report.

Nithin worked on a trade study of our sensors, Intel RealSense, versus Hokuyo Lidar, by comparing if we should use the RealSense for mapping, obstacle detection, or both. He also considered if we should fuse information from both sensors for improved localization. So far, the verdict seems to be that we will use the Hokuyo for mapping of the environment and the RealSense for local obstacle avoidance. We will also use the RealSense for the detection and collection of the objects on the floor, as initially intended.

Team Challenges:

Laavanye had to come up with a light-invariant, robust and the most efficient way to classify whether or not an object was grasped by our robot. By rebooting the image every time and using a histogram of RGB values, he succeeded at the task

Bobby had challenges in coming up with an exploration strategy that will match a lown-mower efficiency and simplicity but that is also robust to several static objects in the environment

Jorge and Nithin found a substantial challenge to re-run our SVD, since recent code changes by different mem had brought new bugs into our pipeline

Team Next Steps:

Laavanye will integrate the failsafe mode into our State Machine

Jorge, and Bobby will implement the cell decomposition and trajectory planning. They will also start implementing the ROS Navigator stack into our code

Nithin will continue implementing mapping and localization for our robot. He will also implement obstacle detection by using the Intel RealSense

References:

[1] http://docs.ros.org/melodic/api/sensor_msgs/html/msg/JointState.html