Carnegie Mellon University

16-682

MRSD Project II

Task 03 Progress Review 8
Team C - COBORG

Jonathan Lord-Fonda
Teammates: Husam Wadi, Feng Xiang, Yuqing Qin, Gerry D'ascoli

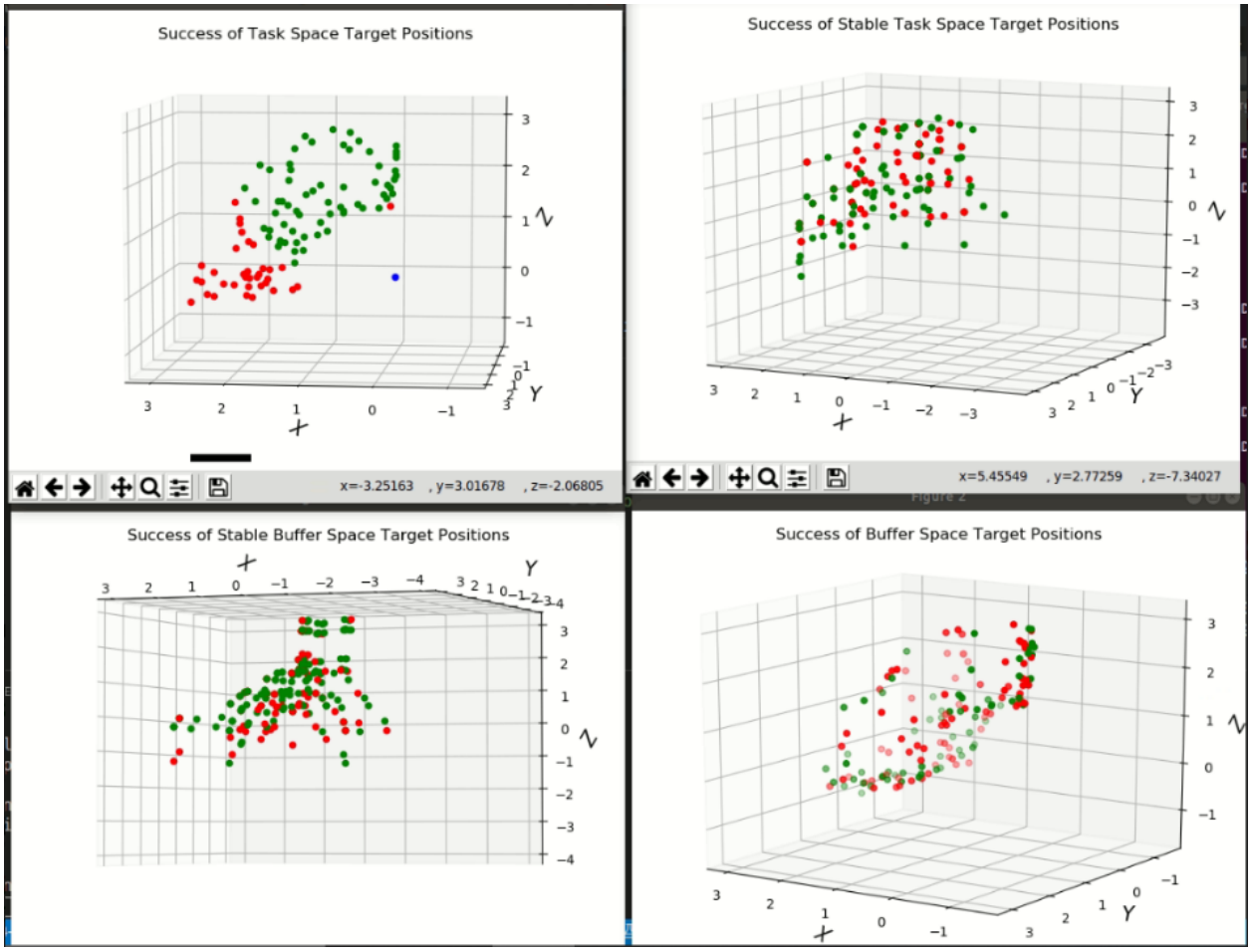
September 30, 2021

Table of Contents

1. Individual Progress

My primary tasks for this progress review were centered on creating programs to analyze our system's task space. I wrote a Python program that used Monte Carlo analysis to test a variety of points from the mechanical task space and see if MoveIt could successfully plan a path from the compact position to the given point. Figure 1 contains preliminary results from running this program for a 5-dof configuration of the Coborg arm. Each green dot was a point that could be successfully reached, while each red dot indicates a failure; our system should have a high success rate within these volumes as that is the intended region which the arm is supposed to be useful for. Unfortunately the program ran pretty slowly due to some quirks with Python's MoveIt implementation, so I began setting up a MoveIt node in C++ to connect with the original program and speed up the process. My reason for this was because the success rates we were getting didn't seem to be quite good enough and between changing link lengths and joint angles there were many arm configurations we might want to test to find a successful one. The program took a significant chunk of time though, and Jason ended up settling on a reliable solution before I had the opportunity to fix the implementation, so I left it unfinished in favor of developing a visual task space checker for Yuqing.

Figure 1 - Task Space Check



This figure shows the monte-carlo task space checker for the mechanical system. The four different plots represent four different spaces that the arm was checked for. The “Task Space” plot (top left) was a basic analysis of our task space. The “Buffer Space” plot (bottom right) was a thick, three-inch shell around our task space that enabled us to see how much leeway we had; if our system had complete success in the task space but low success in the buffer space then we’d know we were pushing our limits. From these two plots the “Stable Task Space” (top right) and “Stable Buffer Space” (bottom left) take a selection of points from the “Task Space” and “Buffer Space” plots and transform them according to our stability requirements. This allows us to estimate how freely the user will be able to move while the system maintains control of the panel.

The visual task space checker for Yuqing was built on a similar concept as the mechanical task space checker but required a completely different, albeit simpler implementation. Since reverse kinematics did not need to be run for the vision aspects I simply created a program that plotted three-dimensional volumes in plotly. The program allowed the user to place cameras at any given location and orientation and then showed a series of plots with blue volumes being the required task spaces and yellow volumes being the cameras’ cones of vision. Seeing how enveloped the blue volumes were by

the yellow volumes is a good measure of how successful we are in covering the task space, as well as which areas we are failing in.

After Yuqing and Jason had some time to try different solutions for their respective systems with their respective programs and create trade studies for those solutions, we met to negotiate between the three aspects, vision, mechanical, and requirements/user needs, which was represented by myself. Jason pointed out that his mechanical solution did well in almost the entire task space, but failed consistently near the bottom. Yuqing's vision solutions struggled with both the vertical and horizontal components. As a first step we reduced the total vertical angle requirement to only cover pure horizontal up to ten degrees from vertical. The reasoning behind this change was that it would be fairly expensive to cover just a few degrees more for the vision system when there was really no reason to force coverage below horizontal. Additionally, it would be extremely uncomfortable to hold a panel directly vertical; it's far more comfortable to hold an object a few degrees off of vertical. After that Yuqing went back to check different combinations of requirements and configurations to find a zone of agreement. To this end we reduced the allowable horizontal angle from +/-30 degrees to +/-10 degrees and reduced the allowable hand spread from 25 degrees down to 17 degrees. Our reasoning for allowing these adjustments was that the angle requirements were written as "nice to haves;" there was no strong reason for them. Additionally, if the task required the person standing at an odd angle to hold the panel it is likely because they are in a cramped, cluttered environment. It would be difficult, if not impossible, for the arm to function under such conditions anyway, so we deemed it acceptable to reduce the horizontal task space requirements given those facts.

Beyond these primary tasks I created a test plan for the entire semester, adding interim tests to proof out each function as we wrap it into our system, and modified our Fall Validation Demonstration plans to account for the updated system we are building this semester.

2. Challenges

There were two main challenges that I encountered this review cycle. The first of which was that writing the C++ node took far longer than I anticipated and so it ended up being obsoleted before it was finished. The good news is that it ended up not being required. The other problem that I encountered was the limited capability of even more advanced vision system solutions. If we had more time, budget, etc and a large task space was important to our customer

base, then it might make sense to sink extra resources into overhauling the system dramatically. As it stands though we developed a good compromise that minimized the number of development resources required while still providing all of the functionality required by our system.

3. Teamwork

Jason's work during this previous cycle was primarily testing a variety of arm configurations for our task space. He did this by running the programs I created to simulate the kinematics, as well as mocking up some systems physically out of cardboard to get a better feel for how they moved. Beyond this he laid out a trade study for the configurations and built a prototype of the best configuration.

Gerry's work during this previous cycle included implementing Octomap for voxel-based object detection and integrating it with the existing point cloud library on our system. Additionally he built up the framework for the resolved rate node for advanced actuated manipulation.

Yuqing's work during this previous cycle included simulating and iterating on the Coborg's visual task space. She also determined the ideal solution by developing a trade study to analyze our options and negotiated with Jason and I to come to a working solution. Additionally she prepared YOLO Tiny v4 and v3 and tested them to ascertain their utility for our project.

Husam's work during this previous cycle primarily included creating, sourcing, and assembling parts for the updated system framework. Part of this was supplying brackets for the vision system and brackets and HEBI motors for the mechanical system. Additionally he monitored the project timeline and kept the kanban board up-to-date and kept members on task.

4. Plans

Before the next progress review I will finish debugging the smart manipulation node, as well as evaluate new control methods that can be added to improve functionality. Beyond this I'll be testing the running system and tuning the gains to provide responsive action that feels right to the user.