Alex Brinkman

Team D: Project HARP (Human Assistive Robotic Picker)

Teammates: Abhishek Bhatia, Lekha Mohan, Feroze Naina, Abhishek Bhatia

ILR #7: Progress Report

Submitted 2/12/2015

1. Individual progress

These past two weeks I have devoted my time to polishing the simulations for the Amazon Picking Challenge video submission, characterizing the configuration space, and creating the online grasp planning process.

We had to demonstrate our ability to control the arm in our simulations to avoid cord tangle issues while avoiding collision objects. The arm planner found many solutions that flipped the arm from ‘left shoulder’ to ‘right shoulder’ configurations which resulted acrobatic motion plans and would have broken the vacuum hose attached to the end effector. To make the arm well-behaved, I restricted the angles of the shoulder pan and tilt joints, elbow joint, and wrist pitch and yaw and found a configuration that would allow us to reach the full configuration space and prevent the left-to-right shoulder flip. Once the arm was better behaved, we worked together to generate visually pleasing simulations highlighting localization, perception, executive task plan for the picking task, and executive task planning for the stowage task.

After limiting the UR5 joints, I found it very difficult and time consuming to place the shelf and bins so the arm could reach the entire workspace. I realized the arm had self-collisions with our Kinect mount and wrist 2 joint when accessing the left side of the shelf. I needed a way to ensure we can reach the full configuration space a priori to determine shelf, arm, mount, and end effector geometries. I decided I would sweep the arm through a large number of goal poses in a 3D grid to find the approximate edge of the configuration space to greatly reduce development and debugging time. Any valid position found from the brute-force search was saved to a pointcloud file for reference. Figure 1 shows the results of this sweep where the green points show the small portion limited by the Kinect/wrist self-collisions and the black cubes approximates the full configuration space.

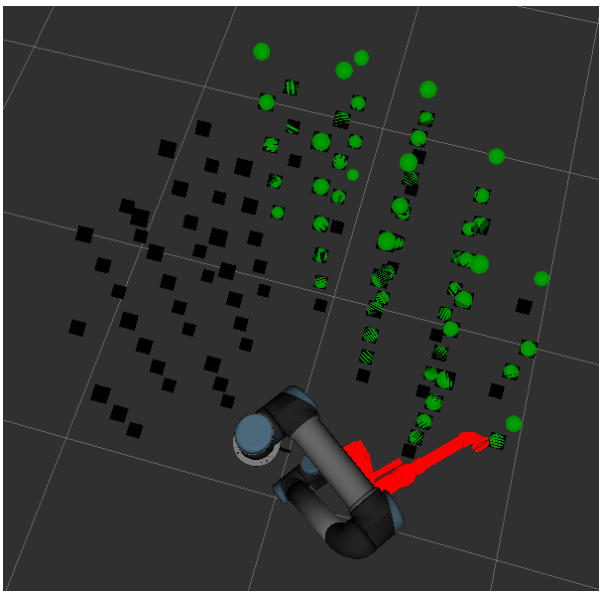

Figure 1: Collision Free Configuration Space

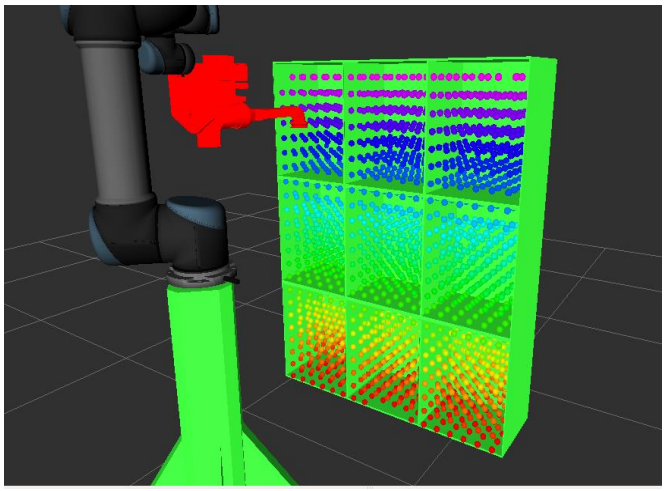

Figure 2: Configuration Space within the Shelf

This map allowed me to move the shelf around and place it so that the arm can access the entirety of the shelf bins. Once placed, I wanted to see if we can reach into the collision model and access every point inside the shelf with high resolution. The rainbow pointcloud shown in Figure 2 proves that the problem of reach is solved at least for this shelf location.

Next, the grasp planner was broken into two parts consisting of an offline process that generates a small set of point normals from the item ground truth and an online process that reads the point normals, projects them onto the pose estimate of the item, and generates an ordered list of suction cup locations. The Point Cloud Library and Eigen library were used to load the point normal for a test object and apply the transformations to the points and normal vectors to translate them into the scene. I created a metric for weighting each normal to generate the ordered list. Once in the world frame, I can take the dot product of the transformed normal with the world Z axis to prefer points pointed upwards, which is the best option for a suction system. Since the sign of the normal is ambiguous, I needed a way to prioritize points on the top face over the bottom face of the item. The Z world coordinate was scaled to (0,1) and the final weight taken as the product of the dot product and scaled height. Once the weights were calculated for all normal, the list was sorted by decreasing weight.

To generate an end effector pose, I know I want the suction cup to be coplanar with plane tangent to the normal at the point but this does not fully constrain the desired grasp pose. I had to take the cross product with the world Y axis using the Eigen::Quaterinion.FromTwoVectors() method and condition the sign of the resulting vector to fully constrain the rotation of the grasp pose. This process was executed on the ordered list of normals to build the grasp plan. Figure 3 shows the end effector aligning to a grasp pose on a test item.

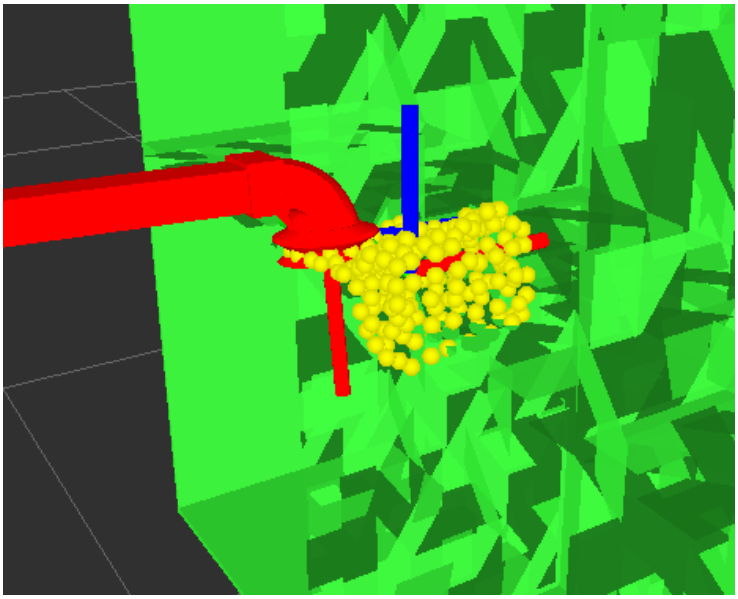

Figure 3: System executing the Grasp Plan

I developed the online grasp planner using a few test cases to numerically confirm the transformations were computed correctly and visualized the results in Rviz to aid development. The development was quick and did not present any significant stumbling blocks.

2. Challenges

The main challenges I uncounted since the last progress review was making the arm motion plans well behaved, managing the codebase, and creating a tool to condition the pointnormals for grasping, and getting the PERCH perception code to work. One of the downsides of the arm planner is that the success feedback of the arm plan is a boolean and does not provide details on why the planner could not find a valid path. So while we were creating the simulations it was very difficult to understand why the arm could not go to the locations requested. This was addressed by mapping the configuration space and figuring out the internal collision with the current Kinect mount design. The other challenge was finding the correct joint limits to keep the end effector from tangling the hose. This required a lot of iterations but does not need to be completed again.

Managing the codebase is becoming are more difficult task. As the number of test cases, configurations, demos, and servers increases managing the various branches take a lot of time. We decided to do some house cleaning and enforced consistency in naming conventions and content across the various demo and feature branches.

The online grasp planner works only if every normal in the database is a valid suction point. Currently, we are working on a tool using the PCL visualizer to manually select normals to exclude from a randomly clustered set of normals but it is taking longer than expected to understand the transforms and create the tool.

The PERCH perception pipeline is provided to use from SBPL. The code would not properly build on the lab computer but we eventually got it to work on Rick's computer. There is still a lot of work to get PERCH working on our item dataset.

3. Teamwork

I worked a lot with Rick to generate the simulations for the APC video submission. Rick used his Mac and iMovie to create the voice and texts layovers on the simulations. Lekha and I worked on the online and offline grasp planners. Feroze and I worked to refactor the codebase and document the changes. Rick and Abhishek worked closely on various available perception ROS packages and getting PERCH to work. Rick developed a tool to manually create scene ground truths for benchmark testing.

4. Plans

Going forward we need to get CMU contracts to issue a PO for the UR5 as the highest priority. Our development will begin to suffer if we do not acquire the UR5 by the end of next week (week of 2/15/2016.) Once we have the UR5, we will evaluate our hardware and quantify the deviation in our localization. We will continue to improve our perception approach and get PERCH working. I will be focusing on getting the SBPL planner to work, which is currently having excessive internal collisions. We will need to create a way to calibrate the Kinect mounting location and hope to have that ready for the next progress review. Finally, we will have the offline and online grasping tools finished and integrated into the system architecture.