1/28/2016

Progress Review 7

Individual Lab Report #6

Abhishek Bhatia

Team D: Team HARP (Human Assistive Robotic Picker)
Teammates: Alex Brinkman, Feroze Naina, Lekha Mohan, Rick Shanor

I. Individual Progress

The official rules for the Amazon Picking challenge were out during December and the major update was that the time for picking task was reduced from 20 mins as of last year to 15 mins, to carry out the whole operation. The second major update was addition of another task ‘Stowage’. We had analysed PR2’s performance and concluded that it was not fast enough for the whole operation and might not give us a competitive edge for the competition [1].

Hence, for the Progress Review 7, team’s major challenge was to transit from PR2 to UR5. Over the break we did a lot of research on UR5 and figured out pros and cons for both the platforms PR2 and UR5. PR2 is a great research platform and has a lot of functionality, moveable base, moveable spine, 2 arms (presents working in parallel capability) but these functionalities also increase complexity and as far as Amazon Picking Challenge is concerned, we do not require these extra functionalities. UR5 has no moveable base, but can access the complete shelf space, including all the shelf bins, moreover is much faster as compared to PR2. The biggest plus is the ROS support that gave us the confidence in transitioning from PR2 to UR5.

My main task for this transition was to work with Alex and modify our state controller to have UR5 as the platform. There were certain challenges associated with this task, the major challenge was that PR2 was based on ROS Groovy and UR5 on ROS Indigo. Alex started with setting up the baseline for our system with UR5. He developed a basic script that launched UR5 in Rviz. I took over from there to make UR5 work with interactive markers. Basic idea was to add two interactive markers within the configuration space of PR2 such that the arm can move between the two markers. The major problem with this was setting up the transforms properly. With PR2 on ROS Groovy, we were using the 'tf' package as part of our TFUtils file to keep track of multiple coordinate frames over time. With ROS Indigo, we discovered that the TFUtils file was not giving us the transforms correctly possibly because of some incompatibilities between 'tf' features and ROS Indigo. I then modified the 'tf' based framework to 'tf2' based, considering ‘tf2’ being the advanced package with similar framework. But, with 'tf2' package I was getting weird errors mostly related to 'tf2' package not being recognized. By experimentation, I figured out a similar package 'tf2_ros' that had the same functionality as 'tf2', and used it to generate and track these coordinate transforms. My script was getting compiled but I was still getting errors with the function 'wait and lookup' (an important functionality that is being utilized by our state controller) that keeps waiting for transforms and returns as soon as it gets one. This was later fixed after going through 'tf2_ros' based tutorials, I figured out that the error was in the way I was passing the parameters to the tf2_ros lookup transform function [2].

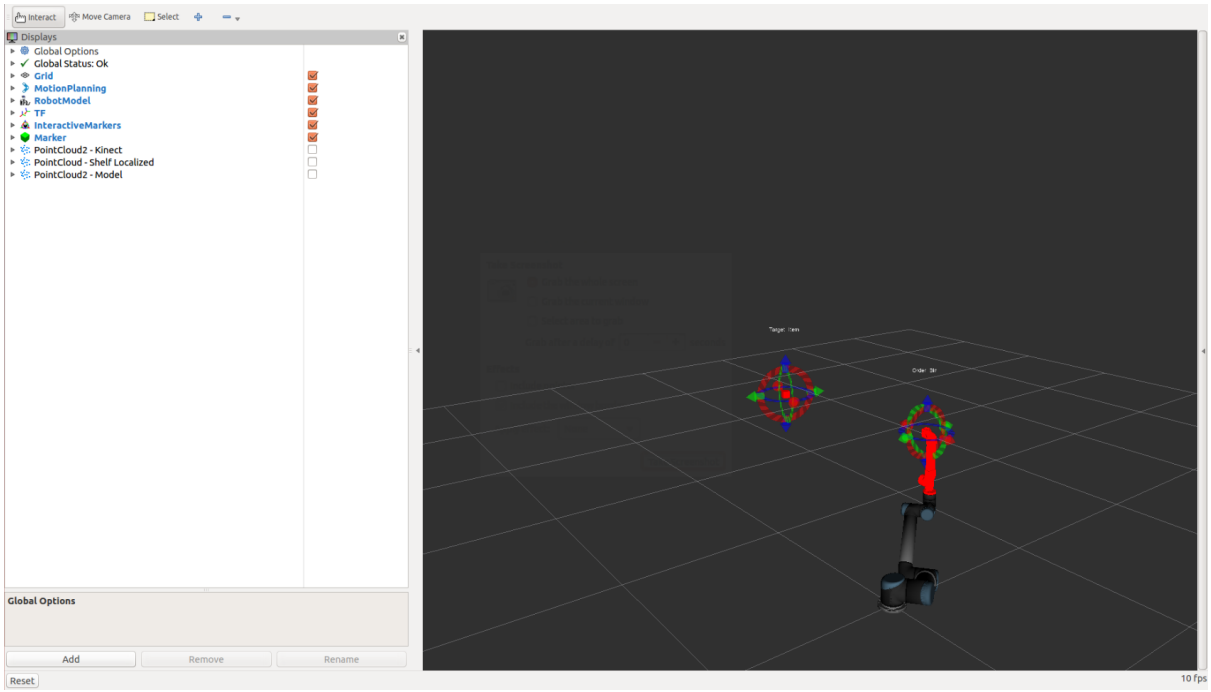

Figure 1: UR5 with interactive markers for target and order bin in Rviz

After this, I worked on defining the static coordinate transforms for our new 9-bin shelf. Previously we had a 12-bin shelf model, but for our preliminary testing now, we have generated a 9-bin shelf with cardboards and to match that in simulation, I defined the static coordinate transforms with dimensions that match with the real shelf model. After this, Rick, Alex and myself worked together to integrate localization and other changes within our state controller and completed the transition from PR2 to UR5. We are able to get the complete functionality, as we had till FVE with PR2, with UR5 now.

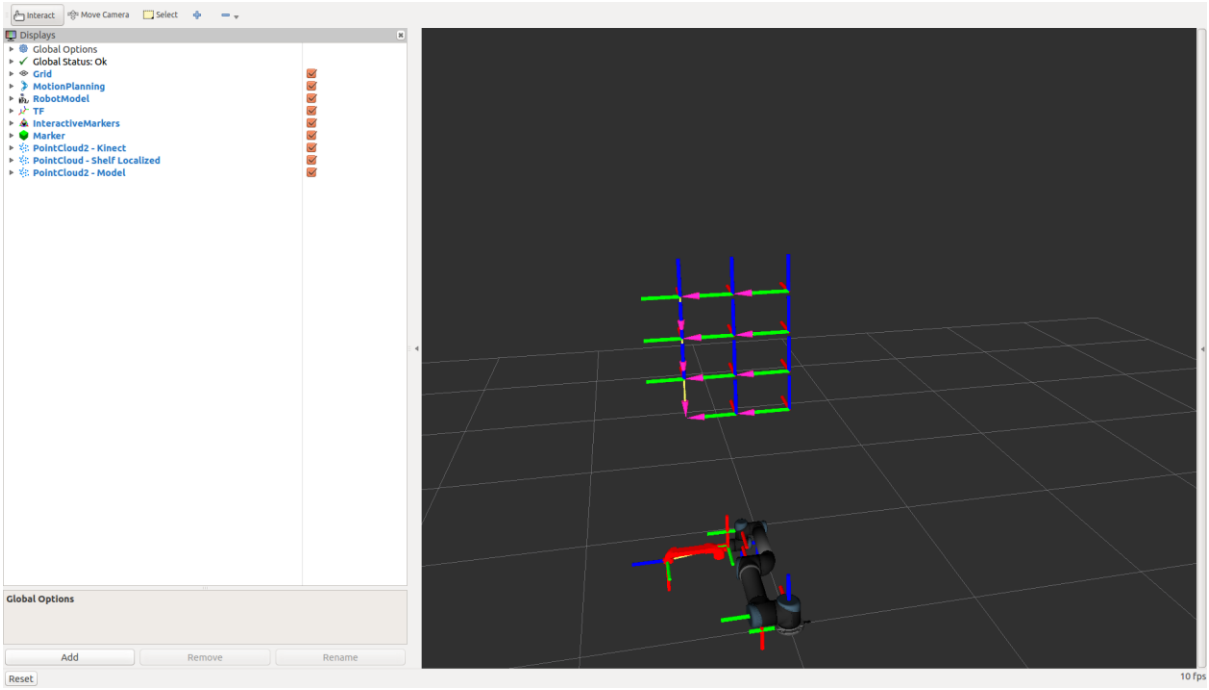

Figure 2: UR5 and coordinate frames for 9-bin shelf

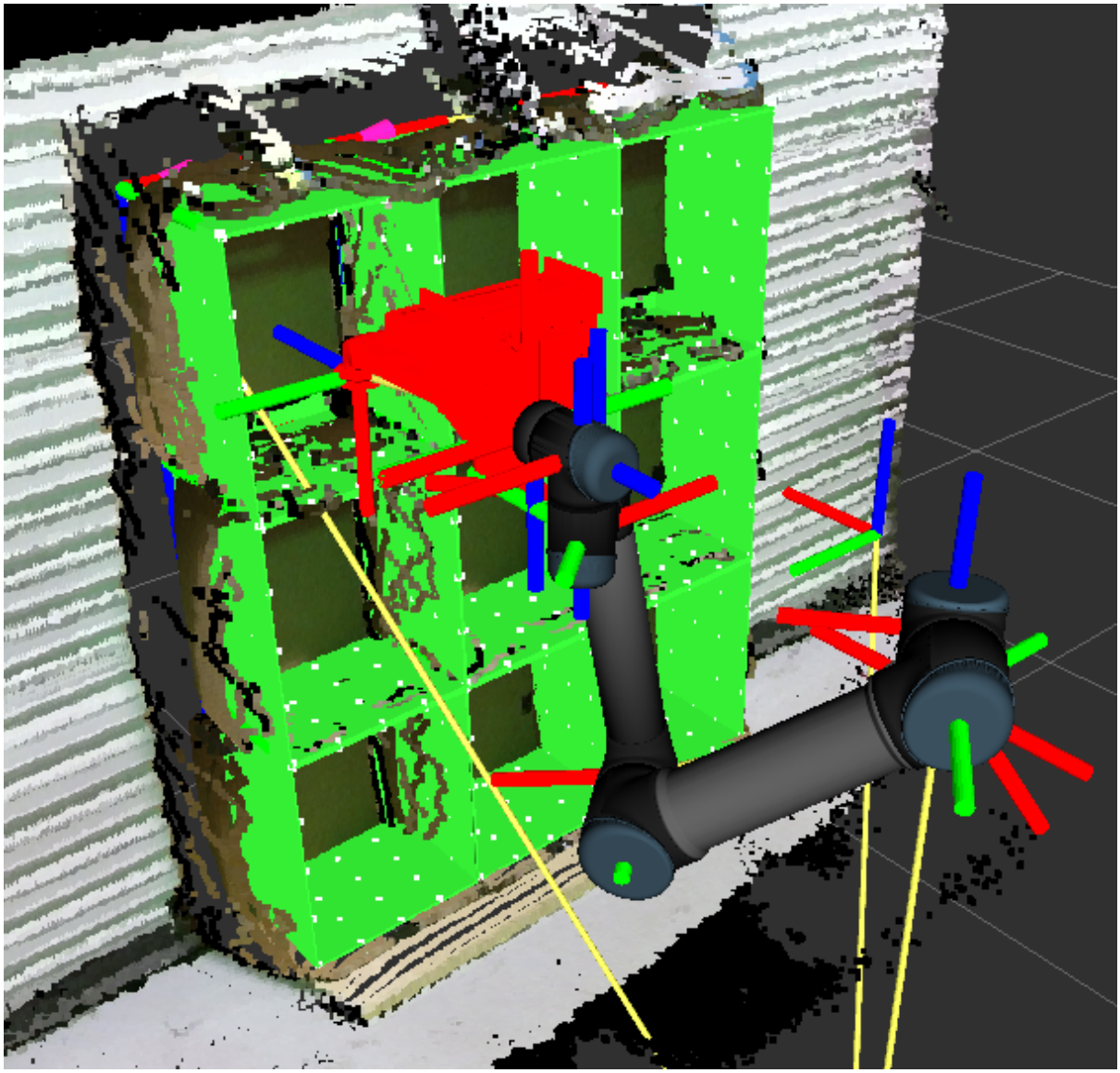

Figure 3: Complete integrated system with UR5 with custom end-effector, 9-bin shelf collision mesh, real-time shelf stream from Kinect2 and localization

Another major update in APC rules this year is that the number of items in each shelf bin could be upto 10 and items could be partially occluded. This could be problematic for our system as our current object recognition algorithm that is based on ICP is not supposed to work well with highly cluttered shelves. To mitigate this risk, we started exploring other alternatives for object recognition. Rick found out two existing frameworks for object recognition 'Simtrack' and 'Object Recognition Kitchen'. Rick started exploring 'Simtrack' and I started with exploring 'ORK'. Object Recognition Kitchen (ORK) is a project originated at Willow Garage for object recognition. There is currently no unique method to perform object recognition. Objects can be textured, non-textured, transparent, articulated, etc. For this reason, the Object Recognition Kitchen was designed to easily develop and run simultaneously several object recognition techniques [3].

ORK framework is based on ROS Indigo and takes 3D models of the objects as inputs and carries out object recognition task. ORK also provides an option to build our own database to store 3D models and reuse them in any point of time. But the only drawback with ORK is that

it is based on Kinect1. All the topics that it subscribes to are published by Kinect1 framework. Kinect2 publishes different topics. Hence, to make it work, we have to change either the ORK source code or develop a wrapper around the ORK packages, that'll subscribe to Kinect2 topics/messages and publishes Kinect1 topics/messages which will be further subscribed by ORK. But before I make that modification, I wanted to ensure that the results from this framework by using Kinect1 are good enough such that ORK presents itself as a backup incase ICP performs bad with cluttered shelves. I started playing around with tabletop detection and object recognition tutorials. Currently, I am working to generate a database with 3D models of the objects from the APC item dictionary and use them for object recognition.

II. Challenges

I faced a lot of challenges while working for this progress review. The first was while modifying the TFUtil file to make the transform package work for ROS Indigo with UR5. I could not find much support in terms of the documentation online that could help me resolve this issue. I took some help from my teammate Alex and also Andrew Dornbush (SBPL lab). Andrew suggested moving over to rewriting the TFUtil again in 'C++' than 'Python'. Finally, Alex suggested a 'tf2_ros' tutorial that helped me resolve the issue.

Later, I faced issues while trying to setup the Object Recognition Kitchen. I faced issues with the openni driver for Kinect1. The ROS package for openni was not recognizing the Kinect1. I later figured out that this issue was possibly because of incompatibility between openni driver and ROS Indigo. The support for openni drivers with latest versions of ROS had been deprecated. To resolve this issue, I used the freenect_launch driver for Kinect1.

III. Teamwork

This week's progress review was crucial for the whole team as we wanted to change our system from PR2 based to UR5 as soon as possible. As always, we had divided this transition into various tasks and worked together to complete this before the progress review. We all updated our systems from Ubuntu 12.04 based to Ubuntu 14.04 and ROS Groovy based to ROS Indigo.

Alex: Alex took the charge for modifying our state controller to have UR5 as the platform and get us to the same state as we were during the Fall Validation Experiment with UR5. Besides, he also worked on redesigning the suction gripper design considering UR5.

Feroze: Feroze mostly worked on making the collision model work as part of our state controller. He first worked on testing this collision framework with PR2 and then added the same for UR5. Finally, he worked with Alex to integrate this functionality as part of our state controller.

Lekha: Lekha worked on setting up input handling block and integrated it with our state controller. Later, she worked on Grasp Planning, to determine safe grasp points for the suction based end-effector to grasp the items off the shelf, safely.

Rick: Rick primarily focussed on integrating the complete vision pipeline as part of our state controller within ROS. Besides, he worked on setting up the localization for our system.

Abhishek: I worked on fixing the issue with transforms that helped us enable the interactive markers, then worked on defining the static coordinate transforms for our new 9 bin shelf and further worked on Object Recognition Kitchen framework.

IV. Future Plans

My major targets for the next Performance Review are to get some results from the Object Recognition Kitchen framework with Kinect-1, verify its compatibility with Kinect-2 and confirm if we can use it as part of our system. Besides, I plan to work on defining an approach for the stowage task. We also plan to work on setting up UR5, incase we get the arm early next week. And finally, work with the team to generate a suitable video required for official APC 2016 registration.

IV. References

1) http://amazonpickingchallenge.org/
2) http://wiki.ros.org/tf
3) http://wg-perception.github.io/object_recognition_core/