Feroze Naina M Dheen Mohamed Ismail

Team D - HARP

Teammates – Alex Brinkman, Rick Shanor, Abhishek Bhatia, Lekha Mohan

ILR02

Oct. 23, 2015

Individual Progress

This week, I worked on creating the basic ROS nodes framework, setting up the computer and drivers, simulating the Kiva shelf and PR2 platform on Gazebo, conducting a demo session on git version control system and creating a git workflow for the team.

I created the basic ROS 'skeleton' code which has a state controller node, image publisher node and a service node for feature detection. These nodes were created according to the software specification document which clearly defines the nodes, their functions, inputs and outputs. This was done so that Rick and Lekha could plug-in their computer vision code directly and run it as ROS nodes.

I learned to use Gazebo – ROS simulator package. The Gazebo package allows us to simulate the physics and dynamics of bodies and visualize it. Gazebo can also simulate changes in lighting and noise in odometry and perception data. The purpose of using Gazebo is to test our perception algorithms for robustness and also analyze how the target items would behave when picked up from the shelf bin. Figure 1 (below) shows the Gazebo simulation with the PR2 model. The Kiva shelf model was obtained from the Amazon Picking Challenge website. The white lines spreading out from the top of the PR2 robot indicate the field of vision of the perception sensors mounted on the pan-tilt head.

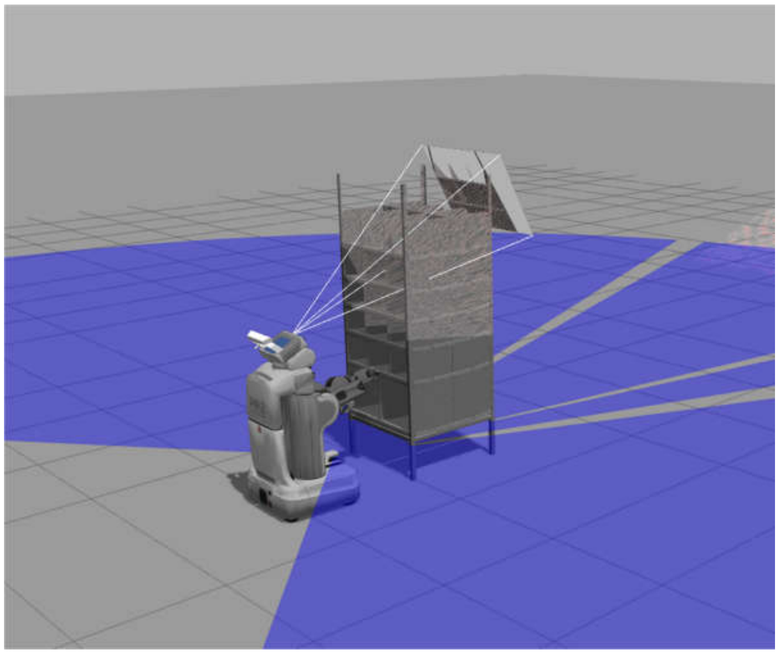

Fig 1: Gazebo simulation

I conducted a demo session on git version control system to help my teammates get comfortable with it. I created a private repository on github and demonstrated how to clone repositories, modify and commit code into branches and push it to github. I also finalized our git workflow - the master branch would contain only the 'production' code – code which has been documented and tested. If a new feature is being developed, it should be created on a separate branch. All the members should commit their code everyday even if it is still a work in progress. This would help us keep track of what each one is working on. Once a feature is completed and tested, I would review the code, document it and merge it with the master branch.

Challenges

Initially, we had planned on developing the entire code-base in Python. However, most of the examples and documentation for perception and navigation libraries and algorithms were written in C++. So we've decided to switch to C++. Having worked mostly with Python, I spent the week getting familiar with C++ programming.

My main challenge was using the object oriented programming paradigm (classes and objects) in C++ and ROS. If the ROS subscriber callback functions are defined as class methods, the C++ declaration syntax is different. I also had to learn about how multi-threading works in C++ and ROS.

Team Work

Bhatia and Lekha analyzed the demo videos of competitors from last year's Amazon Picking Challenge and learned about the game strategies and techniques being used. Rick worked on 2D item recognition, SIFT feature detection, homography estimation and 3D feature matching. He also developed the CAD model for the suction gripper arm.

Alex built a prototype of the suction system using a vacuum pump, a suction cup, pressure sensor and Arduino. He created a ROS node running on the Arduino to publish the pressure values as a ROS topic. He also created the software specification document which contains detailed information on what ROS nodes are required, their functions, the ROS topics they subscribe and publish to and which services they call. This was done so that the team can work on programming the sub-systems independently and later integrate it into ROS easily. The work breakdown structure and milestones schedule documents were also prepared.

Plans

For the project review #2, I will get the PR2 arm to move to the desired position and orientation by specifying the joint angles and visualize it in rviz. I will also extract the point cloud data from Kinect2 and publish it as a ROS topic. I will be learning how to operate PR2 robot platform this week.

Alex had found the SMACH high-level executive ROS library which is a state controller written in Python specifically for the PR2 platform. We've decided to use this instead of writing our own state controller as it is well documented and has been widely used.

An issue we have come across is that the PR2 platform runs ROS Groovy on Ubuntu 12.04 platform. However, the Kinect2 was released in 2014 and would not have driver support in Ubuntu 12.04 (released in 2012). I will verify if ROS nodes from different versions can communicate with each other. If that doesn't work, we will try compiling the Kinect2 drivers in Ubuntu 12.04.

Another major issue is when Rick tested the PR2 and shelf models, he found that even with the robot's arm and torso fully extended, the top bins could not be reached. We have thought of two ways of solving this - construct a platform to boost the height of the robot's base or provide an additional DOF and extend the suction gripper arm.