COMS 6731 Humanoid Robotics
Final Report
May 2018

"Go Fetch" Robot

Team 6
Kathleen Lee (kl2926)
Seungwook Han (sh3264)
Vladislav Scherbich (vss2113)

Abstract

The automation of picking and placing tasks has become increasingly important for the integration of robots in households and in industrial settings. With the development of an efficient algorithm for object recognition and grasping, robots can fulfill a broad range of tasks that require identifying and delivering an item from point A to point B. Particularly for companies like Amazon, a streamlined process of finding and moving objects in a warehouse is crucial. Picking and placing objects in warehouses is often repetitive and difficult for humans. Therefore, a robot replacement for the task significantly reduces the costs and time. These robot "pickers" can follow textual or verbal commands to "pick" objects off the shelves and "place" them in plastic bins that are then sent to "packers" who package the items in a box to send them to customers.

In an attempt to solve this issue, we used the Fetch Robot, a mobile humanoid robot, which has a 7 DOF arm capable of lifting 6 kilogram worth of objects to simulate a pick and place task on objects on top of a table. We can break down our approach into three main processes:

1. Use speech recognition algorithm built from PocketSphinx [1] to parse our speech command to the robot.
2. Use object recognition, which uses the Iterative Closest Point (ICP) algorithm to align point clouds of objects to find a match, to identify an object of interest.

3. Perform grasping using the Moveit interface.

Our end goal was to deliver the whole task sequence from processing a speech command, performing object recognition to identify object of interest, and then grasping the object and performing delivery.

Related Work

The main paper in which we based our implementation was "Robotic Grasping of Novel Objects using Vision" [2] by Saxena, Driemeyer, and Ng at the Computer Science Department in Stanford University. This paper tackles the problem of grasping novel objects with a probabilistic-model based algorithm that identifies a few grasping points from 2D images and predicts 3D grasping points through projections. The algorithm was trained with supervised learning on synthetic images labeled with their respective grasping points. At first, with the same dataset available online, we intended to implement this algorithm as the basis of our grasping mechanism and extend the task for the robot by adding the first layer of object recognition. However, since we were provided with a database of meshes of objects available at the Columbia Robotics lab, we concluded it was unnecessary to predict 3D grasping point using 2D images and rather convert the meshes to point clouds and use another algorithm to identify a grasping point. On top of such grasping mechanism, we aimed to deliver the whole task sequence from speech input processing, object recognition, and object grasping to object delivery.

The second paper which we based our implementation was "Towards Perceptual Shared Autonomy for Robotic Mobile Manipulation" [3] by Pitzer, Styer, Bersch, et al. This paper uses interactive object selection using graph-cut segmentation to identify desired object within cluster of objects. Our main interest in this paper was its use of the ROS tabletop object detection for the PR2, a 7 DOF humanoid robot, to grasp the object. [4] The tabletop detection first estimates the table by finding the dominant plane in the point cloud using RANSAC, then clusters all points about the table to identify individual objects, and then applies simple iterative fitting technique to match the clusters to a model of the desired object in a database. If a good fit is found, a pre-calculated grasp planning is executed. Since we were given a database of meshes, we believed we could easily convert the meshes to point cloud and use the tabletop package to find a good fit to carry out a pre-planned grasp.

Progress

Object Recognition:

Approach 1: We first attempted a vision-driven method for object recognition using the ROS tabletop object detection. The recognition required a connection to a database of known models with pre-calculated grasp executions, namely the household_objects_database, to perform recognition. Thus, we had to download the database backup file and restore it using PostgreSQL server. Instead of saving it to our local database server we saved the household_objects_database on Google Cloud to allow access to all our team members. In order to implement tabletop object recognition, we needed a table with objects on top of it where each object had a minimum distance between them of 3cm and were in an “upright” position. We created two world files, one in which there was a single object on top of the table for simple testing, and then another world file which included five objects on top of the table, both satisfying the conditions to successfully execute the tabletop object recognition. While building the pr2_object_manipulation package, we noticed that we were not able to build it because it was no longer supported. Furthermore, while fixing the dependencies necessary to build the package, the PR2 robot available at the robotic lab broke down. We decided we needed to convert to the Fetch Robot and to find a new approach for object recognition.

Approach 2: Our next approach was to utilize the cluster_segmentation package provided by CRLab that launches a ROS node for point cloud cluster based segmentation of cluttered objects on a table. The ROS node publishes clusters of point clouds for the individual objects on the table as a ROS topic called pcl_clusters. The original cluster_segmentation package did not use the Fetch Robot so we had to manually add robot in the launch file and change the frame from “world” to “base_link” since there is no “world” frame for the Fetch robot. To perform object recognition, we created a new ROS node that subscribes to the pcl_clusters topic and then runs the iterative closest point algorithm to find the object of interest. We provide a .PLY mesh file and convert this mesh file into point cloud using PCL functions. We assign this point cloud as the target object of interest. We then iterate through each cluster of point clouds, assigning each point cloud as the source, and then run the icp algorithm to align the two point clouds. We then call the icp.getFitnessScore(), which returns the sum of squared distances from the source to the target, which means the smaller the score, the closer the match between the source and the target. We find the minimum score from the result and save the corresponding point cloud to find the pose of the object. Since we have the point cloud of the object of interest, we find the centroid of the point cloud and its coordinates for the pose. Our ROS node then publishes the centroid coordinates and orientation as a geometry_msgs::PoseStamped to a ROS topic and publishes a marker in frame “base_link” to visualize the pose in Rviz.

Grasping:

Approach 1: We started out by trying to adapt the Dex-Net 2.0 package developed in Berkeley University. [5] It provided a Python API to manage HDF5 databases of 3D object models, grasps and various metrics. It seemed to be widely used by the open source community, and there was good documentation for it use. However, it’s installation instructions seemed a bit lacking, and the problem was further compounded by outdated packages that needed continuous patching. It eventually felt like falling into a rabbit hole of fixing dependencies unrelated to the core package, and was distracting from the goal of grasp planning. This approach was eventually abandoned.

http://aimlrobots.blogspot.com/2017/11/dexnet-installation-anaconda.html

Approach 2: Our next step was pivoting to the use of GraspIt! grasping framework developed by the Columbia University Robotics Lab. [6] It's installation procedure was much better defined and did not display any of the dependency issues encountered with Dex-Net. We felt more comfortable with using the library, as one of the homework assignments familalized us with GraspIt! usage. It was quite straight-forward to import the Fetch gripper and an object into the graspit_commander interface and plan grasps on the object. Once we had a list of GraspIt! grasp plans, the next step step was executing them in MoveIt! We made use of the CURPP library provided by David Watkins on GitHub. However, we were ultimately unable to execute grasps in MoveIt using this approach. Every grasp plan generated in GraspIt! failed to execute in MoveIt.

Approach 3: After the previous two approaches did not pan out, we began using the MoveIt framework directly to execute arm movements. We achieved intermittent results, where the Fetch arm could be moved to some Poses, but (mostly) not others. Grasping was implemented by adding a Close and Open gripper position.

Speech Recognition:

For speech recognition, we created a language model consisting of the names of the objects that we want to be able to identify and grasp and used a modified version of the Pocketsphinx as a ROS node. For the purpose of simplifying the complexity of the speech recognition feature, we executed the speech recognition feature in key word detection mode. Although we could not optimize the thresholds for each of the words in our language model for maximum accuracy, the speech recognition feature returned the correct object label about 80% of the times - assuming low levels of noise from the microphone.

<u>Challenges</u>

General installation challenges:

All our team member faced difficulties when initially getting used to the ROS environment. Whenever we installed a new package to integrate into our project we faced missing dependency issues, which required a constant search for a solution online. Furthermore, our initial approach to use the PR2 robot for our project led to many installation and dependency challenges because many of its packages were outdated and no longer supported.

Grasp Planning:

DexNet: Because of missing and broken dependencies that required manual patching, we were initially bogged down in the installation process. Additionally, the lack of documentation for this package led us to abandon this approach and resort to the Graspit! + Moveit approach.

Graspit!: Although our generated GraspIt! messages have been translated into MoveIt grasps, they could not be successfully executed in MoveIt. It was not immediately clear what the reason was, but most likely it was connected to object position and orientation translation between GraspIt! and MoveIt.

Moveit: Even though we were successful in planning valid movements in RViz, there were limited valid points in space where the Fetch arm could be moved programmatically. In an attempt to solve this problem, we not only moved the location of the object but also cleared the planning scene of any objects that may collide with the movement of the arm. Because there was no simple method as to exploring and projecting the valid goal pose of the arm and the state space of the location of the object is continuous, we had difficulties in solving this last problem of the task.

Object Recognition:

Our initial method of using ROS tabletop object detection was not possible because the package was outdated and no longer supported. We had to completely convert our object recognition method to one that was suitable to the Fetch robot. For our object recognition using the icp algorithm, we initially got the transformation matrix after icp performed alignment. We planned to convert the Eigen:: Matrix4f transformation matrix to an geometry_msgs::Pose using eigen conversions to find the pose. However, we encountered many issues during the conversion process that we decided to find another approach. Furthermore, we encountered issues when inserting the Fetch robot into the launch file for cluster_segmentation. The Fetch robot would spawn in gazebo but some of its joints would not link and the camera was not recognizing the objects. We later realized this error was because we were still using the "world" frame that the node originally published its topic to.

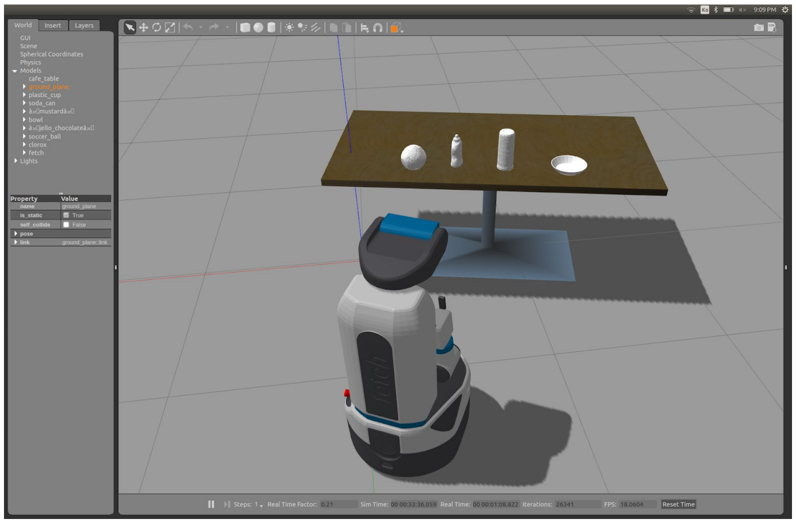
Integration Challenges:

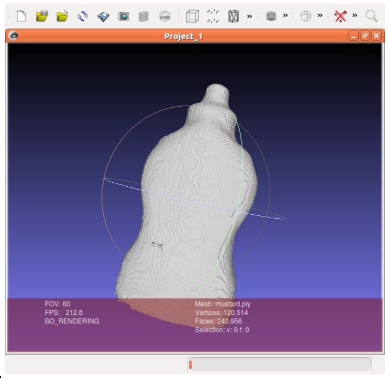
Since each of our group members worked on different aspects of the project, namely speech recognition, grasping, and object recognition, and worked on different branches on github, we had issues when building each other's branches because we all used different packages. We resolved this issue by using GitMan, a "dependency manager" using Git. After we provided the source and branch versions for our packages on GitMan we created a completely new branch with all the necessary packages so we could integrate all three aspects of the project.

<u>Results</u>

Object Recognition

We were successfully able to perform object recognition using the ROS node we created using the icp algorithm to identify the object of interest given a mesh file of that object. In the following images of the simulation, we provided a mesh of a French Classic Mustard to the ROS Node. We were ultimately able to successfully identify the mustard on the table (the second object from the left). The marker topic, in the form of a green sphere, is exactly positioned in the middle of the mustard.

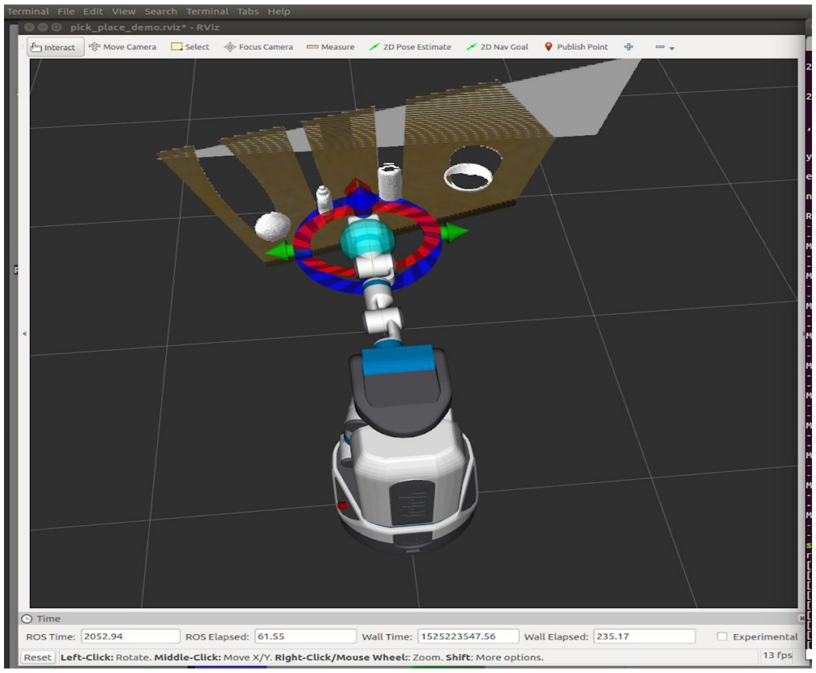




FOV: 60
FPS: 212.8
BO_RENDERING

Mesh: mustard.ply
Vertices: 120,514
Faces: 240,956
Selection: v: 0 f: 0

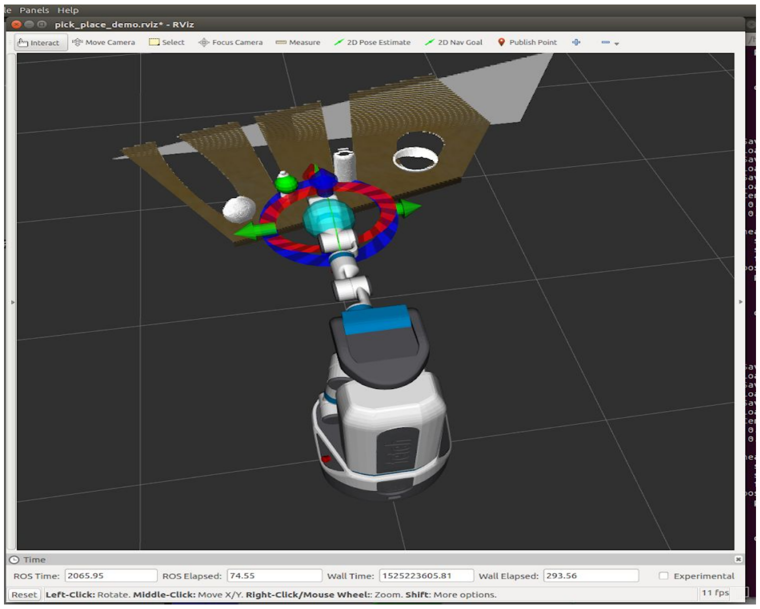
<u>Gazebo View of Objects on Table</u> <u>Mustard Mesh</u>



ROS Time: 2052.94 ROS Elapsed: 61.55 Wall Time: 1525223547.56 Wall Elapsed: 235.17

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

<u>Running Segmentation and Point Cloud Topic Shown in Rviz</u>



ROS Time: 2065.95 ROS Elapsed: 74.55 Wall Time: 1525223605.81 Wall Elapsed: 293.56

Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.

<u>Running ICP Object Recognition and Marker Topic Shown in Rviz</u>

Grasping

Ultimately, we were not able to execute any successful grasps on an object using the MoveIt framework. We published the object poses obtained from our object recognition algorithm to the “pose_stamp” topic and subscribed to those messages in a consumer. When we attempted to move the Fetch arm to or near those poses, every time MoveIt failed to execute those arm movements due to either PLANNING_FAILED or INVALID_MOTION_PLAN errors. To correct these errors, we tried raising the Fetch base higher from the ground, and placing the arm into a more favorable position as the start. Our hypothesis was that the default tucked-in arm pose and lowered body stance was somehow causing path planning failures. It turned out that this did not solve the problem outright. Another theory we were going to test was that the robot was too far away from the object and hence could not physically extend its arm there. By moving the robot closer to the table, and running the subscriber code again, the arm moved to the object right away; however, it did get stuck when striking the table and was not able to recover. We believe that by implementing the following improvements, we’ll eventually be able to move the arm into the desired position relative to the object and successfully execute a grasp: (1) move Fetch close enough to the table such that it's able to reach the object without hitting the table and (2) raising its base and placing the arm in an extended position above the table surface.

Conclusion

Through this project, we accomplished to fulfill a few sub-features of the pick and place task. However, if we had more time, we would have explored several other possible approaches to complete our pick and place project:

1) We realized later on that manipulating the Fetch arm directly through motion planning in Rviz was easier to implement. If we had a successful motion planning trajectory for a certain object, we could have saved this trajectory for the object and execute it whenever the robot identifies this specific object of interest.
2) We could have developed speech recognition so that it understands speech input in the form of sentences and identify the object name within the sentence rather than performing the key word search.
3) Given the time, we wanted to integrate speech recognition with object recognition so that we can provide the name of the mesh file via speech command rather than manually providing the name of the mesh file.

The problem of grasping is an open-ended and complex questions that requires the algorithm to adjust its approach based on the shape of the object and the shape of the gripper. Even for humans, the solution for grasping is not uniform. Although we were not able to delve into this question further, a machine-learning based approach for this task could have resulted in intriguing observations and solutions.

Division of Labor

- Kathleen Lee: Object recognition algorithm development, ROS integration, research, testing
- Seungwook Han: Speech recognition, object recognition algorithm development, research, testing
- Vlad Scherbich: Grasp planning, ROS integration, research, testing

References

[1] https://medium.com/@PankajB96/pocketsphinx-in-ros-demo-1-0-74b2dfc5ebca

[2] Saxena, Ashutosh, et al. "Robotic Grasping of Novel Objects Using Vision," *The International Journal of Robotics Research,* Vol 27, Issue 2, 2008, pp. 157-173. doi.org/10.1177/0278364907087172

[3] B. Pitzer, M. Styer, C. Bersch, C. DuHadway and J. Becker, "Towards perceptual shared autonomy for robotic mobile manipulation," *2011 IEEE International Conference on Robotics and Automation*, Shanghai, 2011, pp. 6245-6251. doi: 10.1109/ICRA.2011.5980259

[4] http://wiki.ros.org/tabletop_object_detector

[5] https://berkeleyautomation.github.io/dex-net/

[6] http://graspit-simulator.github.io/