CS449/549: Learning for Robotics
Bilkent University
Fall 2024
Due date: 26 November 2024 23.59

Homework 3

1 Antipodal Grasp Detection Using Point Clouds

In robotics, the ability to correctly grasp and manipulate objects is fundamental to a wide range of applications. The challenge of finding effective grasp candidates has been extensively studied, leading to the development of numerous grasping methods. In recent years, deep learning models have become a primary approach for generating grasp candidates in robotic manipulation. These models can identify potential grasp points by learning from large datasets, often providing robust results even in complex and unstructured environments. While deep learning has proven effective for grasp detection, graspable regions can also be accurately identified using geometric techniques directly from point clouds without the need for additional object information or pre-training. These geometric approaches not only perform well in practice but can also be used in simulations to generate data for training deep learning models.
In this homework, you will work within a bin environment containing two bins and three objects, each with varying levels of geometric complexity (Fig. 1). Your task is to pick up these objects from one bin and place them into another. Additionally, we encourage you to create additional objects and attempt the task with them as well. You are expected to implement a basic perception module to capture point clouds of the objects, apply the antipodal grasp detection algorithm, and use the detected grasp points as constraints in KOMO to solve the pick-and-place problem.

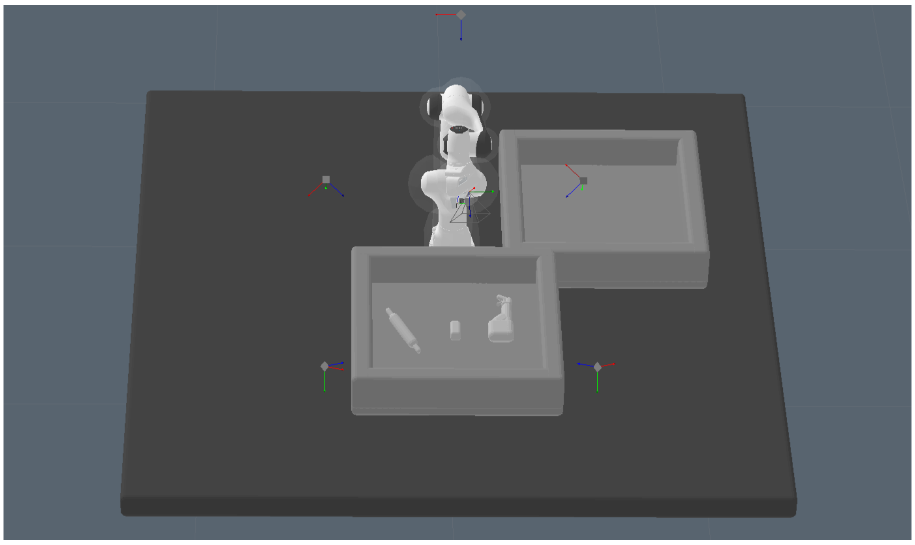
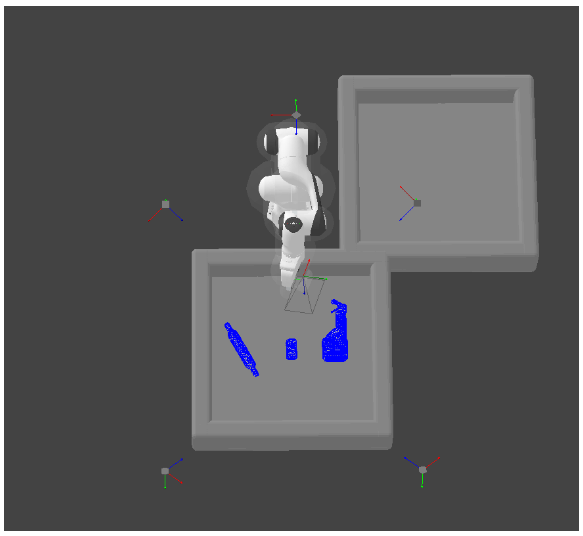


Figure 1: Bin environment setup for the homework, featuring two bins and three objects with different levels of geometric complexity.


(a) Represents the view after adding cropped point clouds into the simulation. As can be seen, only the objects are covered with blue points.

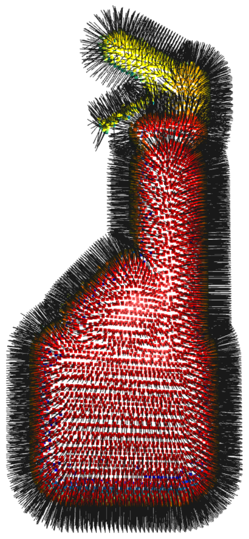

(b) Represents the point cloud of one object after estimating its normals. This image is taken after the segmentation of the point clouds.

Figure 2: Some of the key steps in the perception module include point cloud cropping and normal estimation.

1.1 Perception Module

At the beginning of the project, you will implement a basic perception module to capture point clouds of the objects. You can use functions from the robotics library to obtain point clouds; ext_rendering.ipynb tutorial can be utilized for guidance on interacting with cameras to capture these point clouds. **Ensure you are using a tutorial version compatible with your robotics library version.**
In the setup, four cameras monitor the first bin. To obtain clean point clouds of the objects along with their normals, you need to complete a few steps. Capture point clouds from each camera, but remember that these point clouds are initially in the camera frame, so you must transform them into the world frame before merging them. Additionally, the cameras currently capture points from the entire environment. You will need to crop these point clouds to include only the area inside the bin. You can define a bounding box around the objects and retain only the points inside it. You can check the point cloud by adding it to the simulation (tested for version 0.1.10). To do this, first add a frame as a child of the world frame, such as: **C.addFrame("pcl", "world")**. Then, you can add the point cloud with the following function: **f.setPointCloud(points = pcl, colors = color)**. After cropping, your point clouds should look like Fig. 2a.
You should also estimate the normals of the points to use in the antipodal grasp algorithm. To find the normals, you can use the Open3D library. Ensure that the normals of the point clouds are correctly aligned. An example point cloud with normals can be seen in Fig. 2b.
Finally, you should separate the objects from the overall point cloud to obtain each object individually. For this homework, a simple segmentation algorithm, such as K-means, is sufficient for isolating the objects. You can use the Scikit-learn library for this.

1.2 Antipodal Grasp Detection Algorithm

Antipodal grasp detection is a technique that identifies pairs of contact points on an object's surface where opposing forces can securely hold the object. It leverages spatial and orientation information in point clouds, such as surface normals, to find stable grasp points. The algorithm takes a 3D point cloud of the object, where each point represents a location on the object's surface, and each point has a surface normal vector indicating the direction perpendicular to the surface at that point. The algorithm begins by selecting pairs of points on opposite sides of the object that are a suitable distance apart.
For each pair, it then checks the orientation of their normals. For an antipodal grasp, the normal vectors at each point should ideally point in opposite or near-opposite directions, with an angle close to 180° as a preferred criterion. After identifying pairs that meet these criteria, the algorithm can select the best candidate based on additional metrics, such as grasp stability, distance between points, and ease of access for the gripper. For more details about the algorithm, you can refer to Russ Tedrake's course notes
In this part, you will implement a pick-and-place heuristic that uses the antipodal grasp detection algorithm to determine where to grasp the object. Your robot should grasp the object from the identified contact points, pick it up, and place it in the second bin. The exact placement position is not important as long as the object ends up in the second bin. You should use **BotOp** to implement the task. **It is forbidden to grasp the objects in any way other than through force and gripper operations.** One approach to solving this problem is to add the identified contact points as constraints in KOMO and use the gripper to pick up the objects.

1.3 Additional Objects

For bonus points on this homework, we highly recommend that you try generating your own shapes. Creating one or two new shapes is sufficient; then, add these objects to the ones we provided and solve the task using your own objects as well. Ideally, these shapes should represent real-world objects. We created the existing objects using the **ssBox** shape. You can also use different ssBox configurations and combine them to create your own objects.

Submission

Please follow this instruction to submit your work.

- The coding will be done using the RAI (robotics) package introduced in the tutorials.
- You should upload all the Python files (.py and .ipynb), along with any '.g' files you created.
- You should include recordings demonstrating your solutions in the simulation.
- At the beginning of your code, please specify the version of the RAI robotics library you are using.
- You should submit a single ".zip" file, which involves all of the files in the working form. **Please ensure that all the files run smoothly without any need for modification when executed.**

- For your ."zip" files, please use this name convention "StudentNumber_HW3.zip"