Assignment 6

Assignment Objective

The objective of this assignment is to implement model-free antipodal grasping purely geometrically. The central idea is to compute proposal grasps given a point cloud and clear the clutter off the area in front of the robot depicted below.

Instructions

In this assignment, we will ask you to fill out missing pieces of code in a Python script. You will submit the completed code to autograder. Your code is run to verify the correctness of your work and assign you a score out of 100.

- Download the assignment 6 files from the Files menu of Canvas.
- For each "Part" of the HW assignment, fill in the missing code as described in the problem statement. This assignment has 1 part.
- Submit the completed `assignment_6.py` to autograder.
- The assignment is due on December 12th, 2024.
- You will need `numpy`, `open3d`, and `pybullet` libraries to visualize the point clouds and simulate the robot.

Part 1 – Compute Antipodal Grasp (100 points)

Consider the scene depicted in Fig. 1. The robot (a Kuka LBR iiwa) is tasked with removing the clutter in front of it. There are two cameras (not depicted in this scene) that produce a point cloud of the scene. The robot must use the point cloud to compute antipodal grasps and execute them to remove the blocks.

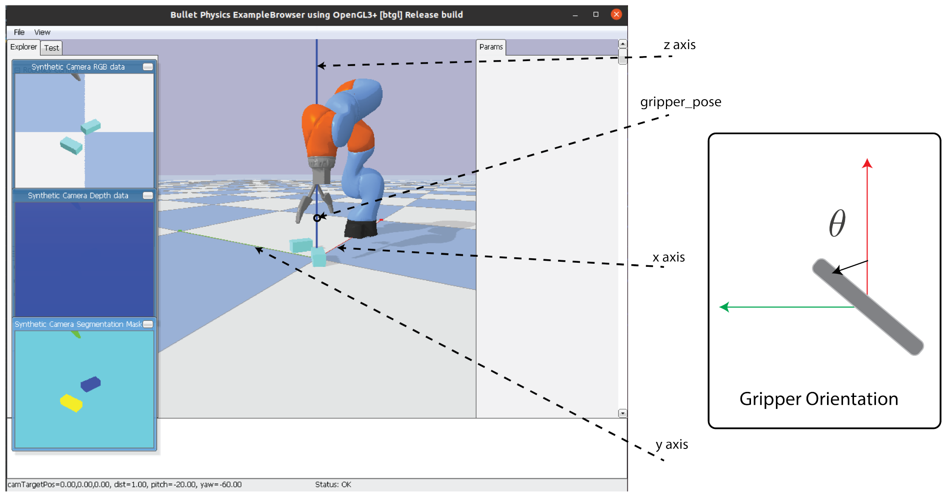
Bullet Physics ExampleBrowser using OpenGL3+ [btgl] Release build
File
View
Explorer
Test
Synthetic Camera RGB data
Synthetic Camera Depth data
Synthetic Camera Segmentation Mask
Params
camTargetPos=0.00,0.00,0.00, dist=1.00, pitch=-20.00, yaw=-60.00
Status: OK
z axis
gripper_pose
x axis
y axis
θ
Gripper Orientation

Figure 1. The robot and scene. The robot is tasked to remove the clutter in front of it using a simple pick-and-place algorithm. The world reference frame is visualized with the red, green, and blue lines. The gripper pose is also visualized, this is the desired pose we calculate and command we send to the robot. On the right, the gripper orientation is visualized. The gripper angle is measured with respect to the positive x axis.

Let's talk about implementation details. Open `assignment_6.py` and take a look at the `main()` function. The first thing to note is that you are not going to need to change this function, we're only going to explain what it does. The first command `world = helper.World()` initializes the robot and its environment. Next, we jump into a for loop that does the following:

1. Generates a point cloud from the 2 cameras in the scene. The combined point cloud is registered to the world frame and contained in `pcd`. This means that every point in the point cloud has a coordinate value of (x, y, z) in the world frame. An important note: the point cloud is already cropped such that the robot and floor are removed. You do not need to do anything extra. The point cloud contains only information about clutter.

2. The `helper.check_pc(pcd)` function checks to see if there are still objects in the scene to pick.

3. The `helper.draw_pc(pcd)` function visualizes the point cloud you collected from the scene. In the `open3d` window that pops up, you can hit `n` to visualise the normals, just as in Fig. 2.

4. The `get_antipodal(pcd)` function is the core of this assignment and what you'll work on. It returns a desired gripper pose the robot should place the gripper in.

5. The `world.grasp`, `world.drop_in_bin`, and `world.home_arm` commands will execute the grasp, followed by moving to a drop position to release the object, followed by homing the arm into its initial configuration.

The `main()` function will run as is; however, the robot does not do anything interesting other than to reach down at the origin and try to blindly grasp.

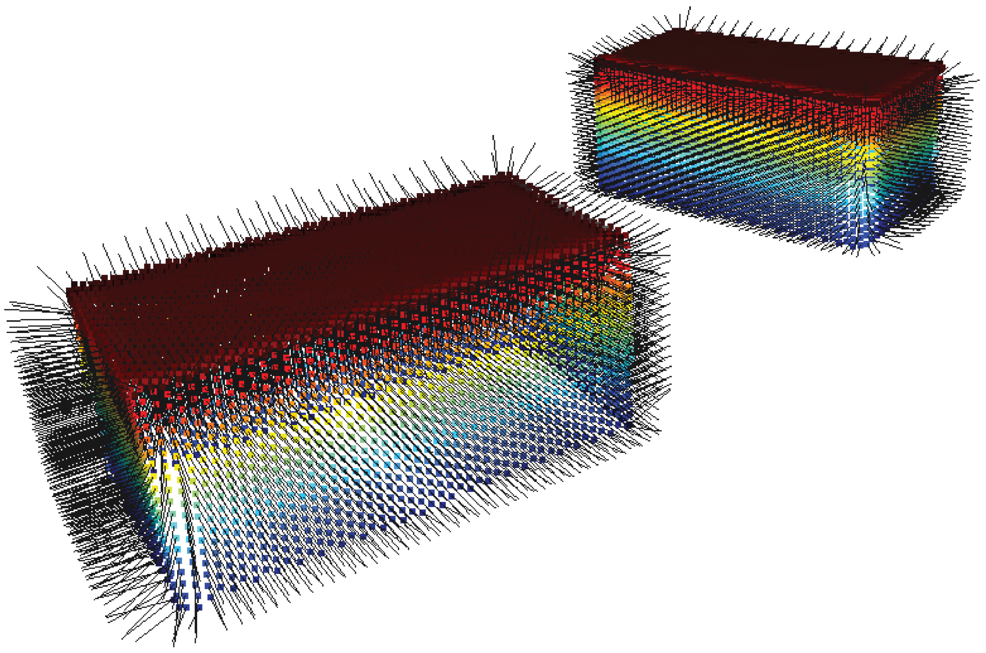

Figure 2. The fused point cloud of the objects, registered to the world reference frame and the corresponding normals.

Next, let's consider the `get_antipodal(pcd)` function. The first two commands convert `pcd` to two `numpy` arrays: the first for the points and the second for their normals. You are to complete this function to return the target gripper pose as a `(4, )` `numpy` array. The first 3 entries are the (x, y, z) values of the gripper pose in meters in the world reference frame. The gripper pose location is visualized in Fig. 1. The last entry is the orientation of the gripper, the angle of the gripper with respect to the x axis in radians.

The visualization in Fig. 1 corresponds to $\theta = 0$ where the gripper fingers are in line with the x axis (the red line).

Recall from lecture that the idea behind finding an antipodal grasp is to find 2 points $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ that have the following properties:

- $||\boldsymbol{p}_1 - \boldsymbol{p}_2|| \leq w$ where w is the gripper width.
- The corresponding normal vectors $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ point in opposite directions.
- The normal vectors are co-linear – they lie along the same line.

The latter two constraints will not hold perfectly, you'll have to find points that approximately satisfy them. In the lectures, we discussed a relatively complete formulation for computing antipodal grasps. In this particular case, we don't need to resort to that level of complexity in reasoning. The objects are polygonal with parallel sides. Exploit this fact to search for antipodal grasps with the simplest algorithm you can come up with.

To help you along your quest, here are a few additional notes to take into consideration:

- The object poses are not random here, they always spawn in the same location. However, don't get too comfortable. During testing, we'll move them so be robust!
- There will only ever be 2 objects in the scene during testing. We won't add more.
- The objects will always be polygonal with parallel sides. Exploit this domain knowledge to simplify your reasoning.
- The gripper width is at most $w_{max} = 0.15$ meters.
- The `main` function is designed to end once there are no more perceived objects in front of the robot.

Grading Details: Your code will be tested on 55 different configurations. They include the default configuration (the configuration we provide you) which is worth 70% of the final score. The remaining 54 tests contribute equally to the remaining 30%. You will have 5 grasp attempts to clean each scene. The grading has a timeout so if your grasp computation takes more than 2 minutes in our computer (it is a powerful machine) it is killed and you lose one of the 5 attempts. Therefore, we recommend that you make sure that your code is efficient.