MINI PROJECT 2: GRASPING

Due: Tuesday, October 14 11:59pm EST

The objective of MP2 is to study grasping algorithms for table-top objects. In this mini project, we will use MuJoCo to simulate known objects and generate their point clouds. These point clouds will then be used for grasp planning. Your task is to implement an antipodal grasping algorithm to identify the best grasp poses that can successfully pick up and hold the object.

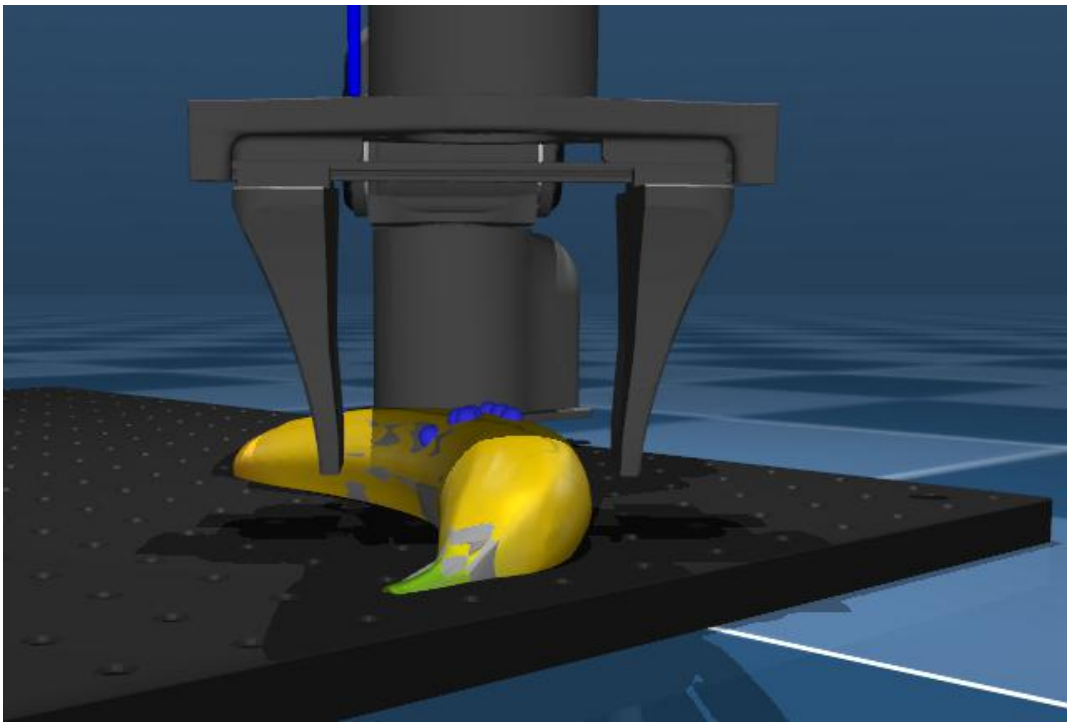

Figure 1. MuJoCo simulation of a banana for table-top grasping. The blue lines inside the banana represent a subset of pairs of antipodal points. Based on these pairs, you will infer the robot's grasping pose.

A. Point Cloud Filtering (1 pts)

As the first step, your objective is to filter noisy point clouds containing outliers. Obtaining clean point clouds is essential before conducting grasp planning. You may use a statistical outlier removal method for this purpose.

Your task is to complete `filter_outliers` method in `Grasping` class in `Grasping.py`. And you can test the results via function *same_points_unordered()* in *grasping_test.py*.

B. Compute the Point Cloud Normals (2 pts)

The grasp selection strategy we develop will rely on the local geometry (normal directions) of the scene. Learning how to estimate these quantities from point clouds is an excellent exercise in point cloud processing and is representative of many similar algorithms. Your task is to compute the surface normal for each point in the given point cloud.For specific details on nearest neighbor, please refer to lecture notes and Section 5.5.2 in these notes.

Your task is to complete `estimate_normals` method in the `Grasping` class. And its input is the neighbour_index computed using the provided function `nearest_neighbor`. And you can visualize the normals via function *draw_pointclouds_and_normals_in_viewer()* in *grasping_test.py.*

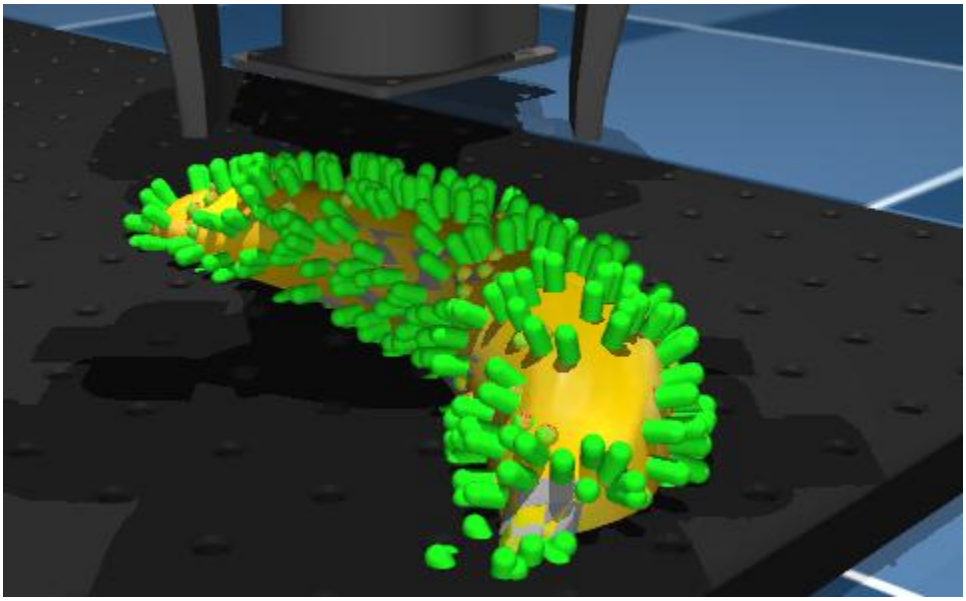

Figure 2. Visualization of the point cloud normals. Some normals may be flipped (but still optimal solutions), causing them to point inward toward the banana.

C. Compute the Grasping Score (3 pts)

Based on the point cloud normals, you may design your own heuristic method to identify pairs of antipodal points. Since different students may propose different heuristics (via the function `find_antipodal_pairs`), this part will not be graded directly, but rather implicitly through the quality of the final grasping outcome. For each antipodal pair, there exists a possible grasp pose. It is therefore useful to evaluate and score these pairs, then generate grasp poses for the top candidates. One way to score antipodal pairs is by measuring how colinear the local normals are

with the grasp axis (i.e., the line connecting the two antipodal points). Since normals may be flipped, be sure to account for this when evaluating alignment.

Your task is to complete `evaluate_grasp_score` method in the `Grasping` class.

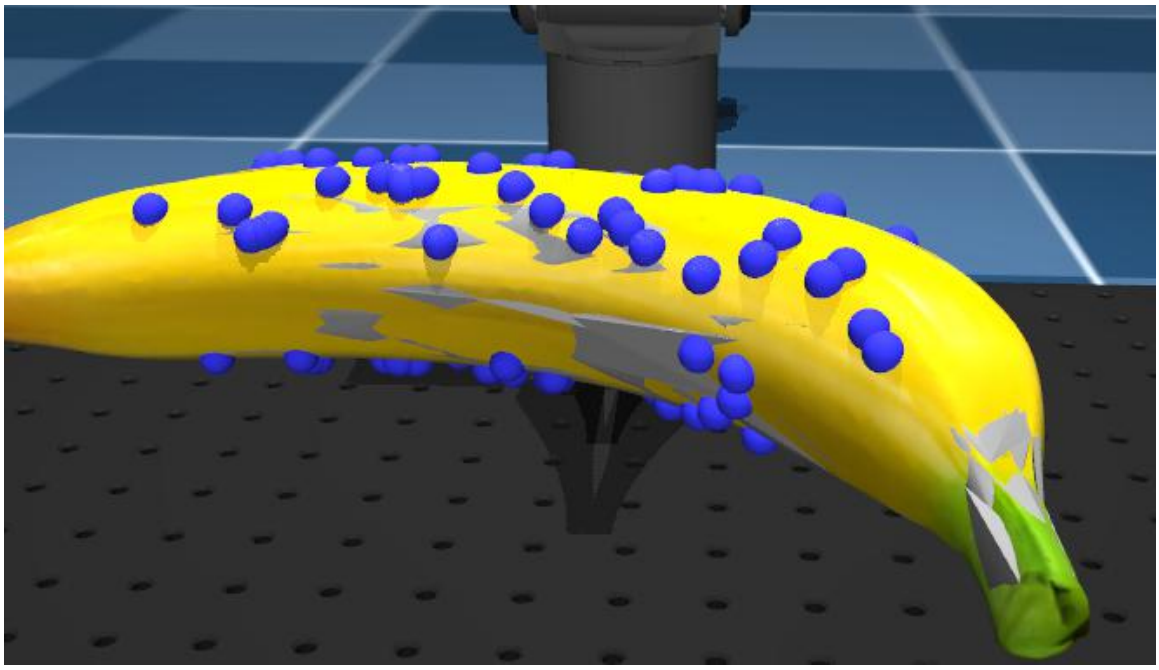

Figure 3. Visualization of the antipodal points, with each blue line connecting a pair.

D. Compute the Grasping Pose (3 pts)

From the antipodal points, you will determine the corresponding grasp poses. To simplify the problem, we assume that all grasps are perfectly table-top grasps, meaning the gripper axis is always aligned with the $-Z$ axis. In other words, the robot end effector only rotates within the horizontal plane. Under this assumption, each antipodal point pair corresponds to a unique grasp pose.

Your task is to complete `compute_grasp_poses` method in the `Grasping` class. And you can visualize the grasp poses via function *draw_grasp_origins_in_viewer()* in *grasping_test.py*.

E. Testing the final grasping outcome in MuJoCo (6ps):

Linux:

Download and unzip `MP2.zip` from Canvas and place it under

`CS4803ARM_Fall2025/user_data`

Note: Any files placed in this directory will be accessible from the Docker container

Change the directory

```
cd CS4803ARM_Fall2025/docker
```

Important: You will not see GUI if you don't run this command before running the shell script.

```
xhost +
```

Run the convenience script to run the container

```
./docker_run.sh
```

or

```
./docker_run_gpu.sh
```

(If your computer has an Nvidia GPU and you have installed Nvidia container toolkit. MuJoCo will be much smoother and faster.)

Change the directory to MP2 inside the Docker container.

```
cd user_data/MP2
```

Run the MuJoCo simulation.

```
python3 grasping_test.py
```

Mac:

Download and unzip MP2.zip from Canvas and place it under

```
CS4803ARM_Fall2025/user_data
```

Change the current directory to the MP2 folder

```
cd CS4803ARM_Fall2025/user_data/MP2
```

Run the MuJoCo simulation.

```
mjpython grasping_test.py
```

However, we have observed some problematic contact behaviors in the current simulation environment, so Yunhai and Soobum are still debugging the MuJoCo simulator. As a result, you may encounter unexpected issues when grasping the banana. We will update the simulation code soon and post an announcement on Piazza. Thank you for your understanding!

ALso, later this week, we will announce the grading criteria for each section (e.g., through an autograder or manual inspection), and may also test the grasping on the physical robots.