GreenBot: A Robotic Manipulation Approach to Identifying, Picking, and Placing Recyclable Waste

Jared Boyer - Fall 2022

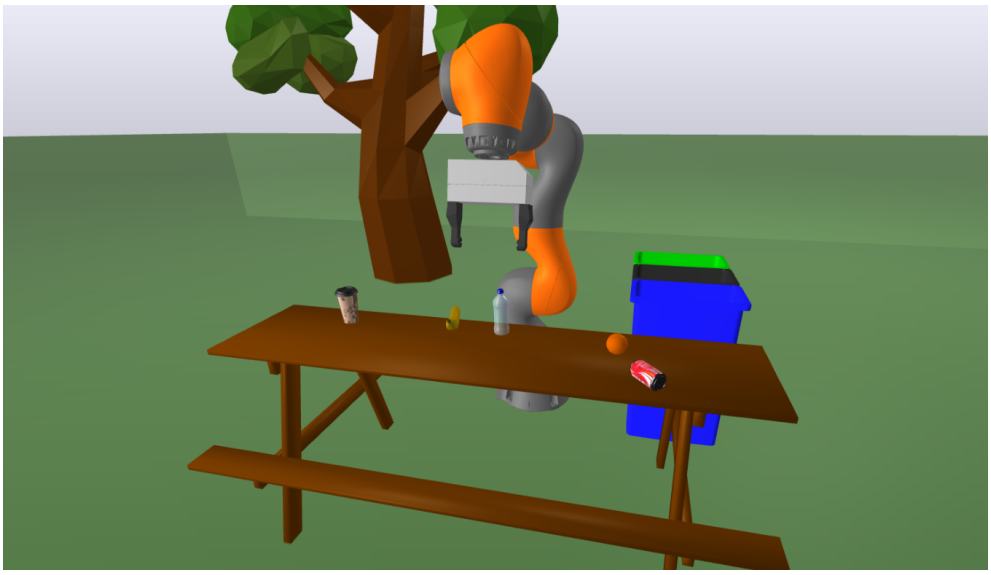

Figure 1: Visual rendering of the GreenBot simulation environment

1 Abstract

This work introduces a fully autonomous robotic manipulation system capable of sorting different kinds of waste and placing objects in their respective bins. It features deep learning for object segmentation, optimal grasp generation from point cloud analysis, and a differential inverse kinematics controller. The tests highlighted in the results section show promising performance, while future work, limitations, and 'sim2real' gaps are touched upon in the discussion section. Source code for the project can be found in the Robotic Recycling repository on github.

2 Motivation

The mis-classification of waste items is a common culprit of low recycling rates, a leading cause for waste mismanagement, carbon emissions, and ocean pollution. Human error at the source of waste disposal sets an innate limit to the amount of proper waste management, creating the need for off-site waste organization facilities. However, sifting through streams of household waste to find recyclable items is a difficult, dangerous, and infeasible job for humans to do. Introducing autonomous systems has the potential to increase waste classification accuracy, improve recycling rates, and shift the quality of proper waste management worldwide.

3 Technical Approach

3.1 Configuration

GreenBot is implemented in Drake, leveraging a large backbone of manipulation tools present in Russ Tedrake's Manipulation repository and online textbook. This work simulates a 7-link Kuka iiwa and a Schunk WSG 50 gripper as the main manipulation station; both of these models are native to Drake. A set of supplemental 3D models were compiled in order to simulate a realistic recycling application, including waste bins, common household trash objects, and a picking table. Visual meshes were obtained through various online sources, but collision geometries were made using an auxiliary convex mesh decomposition tool.

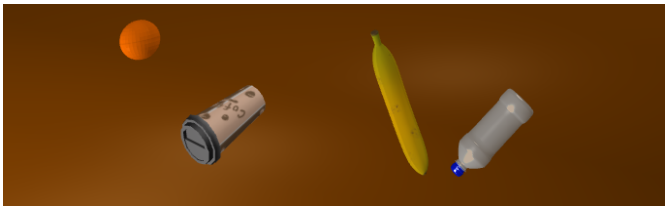

Figure 2: Example waste items that GreenBot can sort into either landfill, recycling, or compost

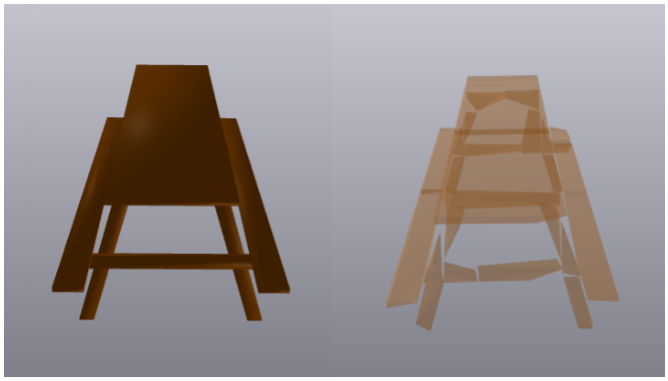

Figure 3: Decomposing a complex mesh into convex subregions for effective collision geometry

3.2 Perception

GreenBot's perception system ingests the RGBD output of two cameras, each placed at opposite ends of the picking table. Before a picking action is planned, RGBD images are passed through a Mask R-CNN model to generate masked and labeled depth images of the observed trash items.

The technique for training this Mask R-CNN model was adapted from the approach we've used in class. Starting from a model pre-trained on the COCO data set, the network head was re-trained to instead predict object segmentations for each of the prescribed waste items. Due to limited disk space, low processing power, and a crunch on time, this network head could only be trained for one epoch on the available development machine.

Training data were generated by randomly sampling configurations of waste items. To create a given sample, the four waste items are first loaded into the simulation and randomly oriented over the picking table until a configuration of no collision has been reached. Afterward, RGB images and labeled masks are saved from each camera, along with a respective .json file containing the mappings from label index to object type. This was repeated to generate a set of 500 images.

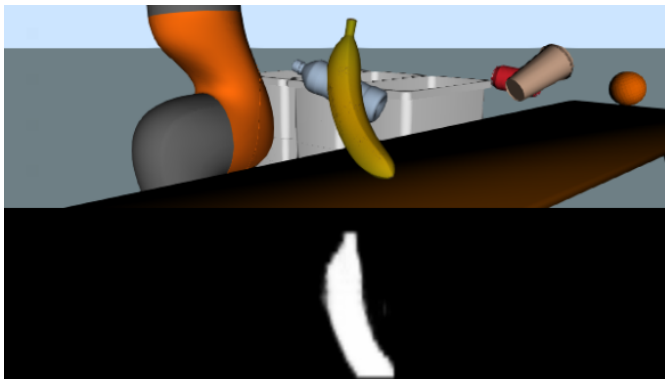

Figure 4: Test image and corresponding "Banana" segmentation from Mask R-CNN

After this Mask R-CNN model is queried by camera RGB outputs, the resulting object-segmented depth images are mapped to labeled point clouds. The highest confidence prediction is chosen and the corresponding point cloud is subsequently handed off to a module responsible for grasp pose generation.

3.3 Grasping

After identifying a labeled point cloud, GreenBot samples approximately 100 candidate grasp poses and selects the grasp that maximizes the following function.

$$\sum_{i \in C} (N_i \cdot g_x)^2 \tag{1}$$

Where N_i is the normal vector corresponding to the ith element in the point cloud, C, and g_x is the gripper x-axis. GreenBot also imposes the constraint that the gripper is not in collision with the picking table.

Grasp candidates are generated via antipodal sampling. This is accomplished by sampling a random point within the point cloud and then calculating, flipping, and projecting the local normal vector into the XY plane. The gripper pose is then created such that the gripper X-axis is aligned with the projected normal, the gripper Y-axis is aligned with the world -Z, and the measured point cloud fits within the gripper fingers.

Figure 5: Example grasp using anti-podal sampling

3.4 Planning & Control

Throughout operation, GreenBot uses a pseudo-inverse controller. This controller ingests a desired spatial velocity and calculates the necessary joint velocities using the Jacobian pseudo-inverse. In order to follow a desired trajectory, piece-wise translations (rotations) are linearly (spherically) interpolated, and this trajectory is differentiated to get a time-series of desired spatial velocities.

One limitation to this approach is that position error is not actually regulated; velocity errors will be integrated into position error that grows over time. In order to compensate for this, GreenBot uses an extra piece of logic that checks if the final position is within an epsilon threshold

of its intended location. If it isn't, a miniature trajectory is planned to close the gap.

After a suitable grasp pose has been calculated, a piece-wise pose trajectory is made from the starting pose to the desired grasp pose, and the gripper fingers are then closed. A trajectory is then planned to the object's assigned waste bin, which is a manually defined correspondence. After opening the gripper fingers and dropping the item, a trajectory is planned back to the initial pose. This process is repeated until there are no more items on the table.

4 Results

4.1 Success Rates

To evaluate GreenBot's performance, 100 test cases were ran. In each test, all four waste items were randomly placed on the table and GreenBot was given 60 seconds for sorting. Unfortunately, 44 of these tests experienced multi-body dynamics solving issues, which was a common occurrence during development. Of the 56 remaining valid tests, all items were successfully placed 67% of the time, and at least three items were successfully placed 87.5% of the time. The results are tabulated below.

# of items placed	0	1	2	3	4	Total
# of tests	1	1	5	11	38	56

4.2 Failure Modes

A limitation of GreenBot is that trash is assumed to sparsely populate the environment. Under this assumption, we can confidently plan piece-wise pose trajectories to and from the picking table and waste bins without worrying about colliding with other trash objects. However, since initial configurations of trash objects are sampled randomly, this sparsity assumption isn't always true.

In the case of trajectory planning from the picking table to waste bins, this issue can be circumvented. By planning trajectories that employ a "pre-picking" and "post-picking" pose, located just above the desired object, GreenBot avoids planning trajectories through the potentially hazardous picking area.

However, this sparsity issue also arises in grasp pose generation, where it has a more problematic effect. For example, when two trash objects are initialized close to one another, the grasp pose generator may select a pose that is within collision. This is possible because the grasp generator checks if pose candidates are in collision with sedentary objects in the environment (picking table, waste bins), but not with the variable-pose trash objects, which are not currently accounted for in the grasp generator's internal system model. Figure 6 details an example case, where the gripper collides with a coffee cup on its way to pick up an orange.

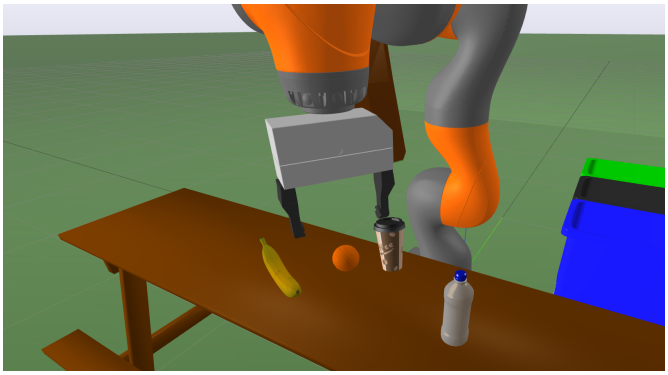

Figure 6: Incorrect grasp generation leads to collision

Future iterations of GreenBot can remedy this item sparsity issue by either opting for a more robust planning technique, focusing on collision avoidance, or by simply including trash objects into its internal system model.

5 Discussion

In addition to the limitations mentioned previously, GreenBot makes a few major assumptions to simplify the problem at hand. For example, in order to figure out which waste item belongs in which waste bin, GreenBot uses a manually defined set of correspondences from object type to waste type. Since real-world applications won't have access to an exhaustive set of these correspondences for all possible items, this is likely an unrealistic assumption. Moving forward, this system can benefit from more robust learning techniques that can not only determine the object class of a given item, but also infer the type of waste it is.

Additionally, the candidate objects used in this simulation (plastic bottles, cups, produce) are all pliable objects in the real world. Safe grasping of these objects would require force control as well as knowledge of the structural integrity of each item. Rather than develop a sophisticated force controller and dive deep into simulating soft-body interactions, the simulated GreenBot instead assumes all objects are perfectly rigid. Including a safe, force-controlled gripping mechanism is an essential step forward for making GreenBot viable in the real world.

There is also a lot of potential for future *Robotic Manipulation* final projects to build off of GreenBot. One possible adaptation that is of high interest is instead using a mobile manipulator for automatic recycling. Especially in light of the popular demos from Everyday Robots, mobile manipulation systems have infinitely more utility than stationary robots for real-word robotic recycling.

In general, I hope that the environment, 3D modeling, and baseline system that I have created can be a useful starting point for other projects to take inspiration from.