Voluminious Object Recognition & Spatial Reallocation Into a Containing Medium (VORSRICM)

Lisa Fawcett, Michael Nazario, and John Sivak

Abstract— **Placing objects in the fridge, like many tasks, is simple for humans and complex for robots. A necessary component of loading a refrigerator is to stack any items such that they are available and in the most space-efficient locations. We consider optimal packing as a refrigerator that has the smallest amount of unoccupied space between items and has the most items in view. Because of this, rotating objects and placing them on top of one another becomes important, meaning the *stackability* of each object's sides is crucial for packing a fridge.**

Assuming a simple rectangular prism model for each object, we propose a learning approach to determine the *stackability* of each object's faces. Training a support vector machine on point cloud data of food and kitchen related objects, we determine which sides are suitable for objects to be placed on top of. This information then is incorporated into our bin-packing algorithm, allowing the PR2 to stack items with less initial information about its environment.

I. INTRODUCTION

Often containers can only store all desired items if packed efficiently. In some cases, packing all items is acceptable, but in a situation such as a refrigerator, food will spoil if it is not packed. On the other hand, if objects are packed too compactly, fragile items could be crushed. A personal robot should be able to make a compromise between these two extremes and create an acceptable solution that gets the job done.

Our proposal is to create an algorithm that would allow a personal robot to make such decisions. In order to do this, we need to determine what a good packing is, while ensuring that objects are not crushed through a concept of *stackability*, or how the object can be stacked and stacked upon. Using a simple rectangular prism model for objects, each object's face is considered stackable if its face is stable enough to set on a flat surface and place an object on top of. A support vector machine classifies the point cloud data from a PR2 to determine stackability. We use simple food and kitchen-related object models as our training set. Once the object's stackability has been determined, we run a greedy packing algorithm which places the objects from largest to smallest and from the back of the fridge to the front, while guaranteeing that all stacking constraints are satisfied. After all of the objects' locations have been decided upon, we use a motion planning algorithm integrating the ROS Pick and Place package in order to get the PR2 to move the objects from the table to the fridge.

Our algorithm has a 79.81% accuracy of classifying the stackability of sides of unknown objects.

Lisa Fawcett, Michael Nazario, and John Sivak are with the School of Computer Science, Cornell University. {lcf38,mgn29,js2728}@cornell.edu

II. RELATED WORK

Previous work has been done in placing unknown objects by Jiang et al., 2012. Our approach is different as we are not allowing for objects to be placed inside one another in our model. However, we use features similar to theirs in order to classify whether a side is stackable or not in our rectangular prism model. In addition, we are assuming the environment is a perfectly flat surface whereas they assumed nothing about the flatness of the environment.[1]

III. APPROACH

Our approach involves three main sections: the determination of the size and stackability of objects, the packing algorithm and visualizing the result, and controlling the PR2 to take action based on the algorithm's output.

A. Background

Since the entire task of discovering an object, classifying it, and moving it to a location in the refrigerator is an extremely complex problem, we make the following assumptions in our model to simplify the problem:

- All objects are placed on a flat surface within reach of the PR2.
- Objects are in known locations and orientations.
- Objects are easily approximated by rectangular prisms.
- The bottom surface of an object is *stackable*.
- Environment is known. Locations of refrigerator and tables are known.

B. Object Bounding

Because we represent objects as rectangular prisms, we need to determine a reasonable bounding box for each perceived object. Therefore, we created a simple algorithm using Point Cloud Library (PCL) to achieve this goal.

First, since noise is common in stereo point cloud data, we clean up our data with a pass-through filter and statistical outlier remover. Using a voxel grid, we simplify our point cloud data so that we can use a RANSAC algorithm to determine the largest plane in the data. We then remove the plane from the data and use the plane's representation to constrain our search of bounding boxes to only those which lie normal to the plane as one of the box's dimensions. Finally, we rotate the box using a small increment in angle and keep the tightest bounding box around the object. We have determined that this will give a reasonable bounding box for an object, since we guarantee at least one of the faces is stackable and therefore placeable in the fridge. Figure 1 is

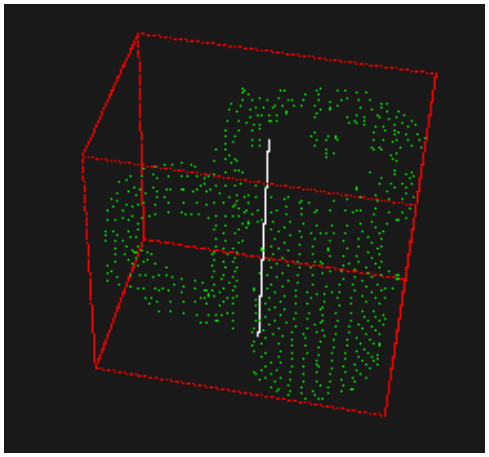


Fig. 1. Bounding box for a point cloud of a coffee cup

an example of a bounding box which is tightly bound around the point cloud of a coffee cup.

From this bounding box, we extract point cloud segments which can be treated as the faces of the object. Any points which are within a threshold distance from the bounding box's edges are treated as part of that face. As a result, we get a snapshot of that side's shape and size. Figure 2 displays a visualization of these divisions. We treat any points within $\frac{1}{4}$ of the dimension's distance from a face as part of that face.

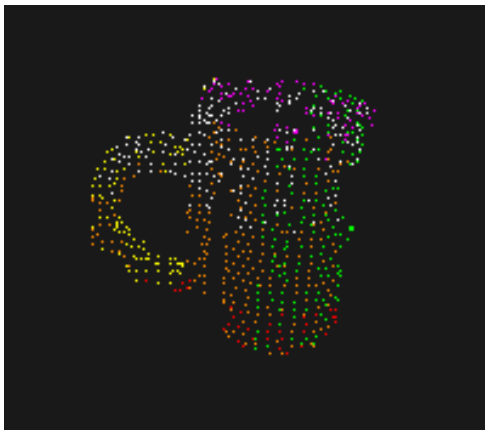


Fig. 2. Point cloud segmented into faces. Each color represents a face which that point belongs to.

C. Stacking Classification

Since the sides of our objects are of unknown stackability, we need to determine whether or not the sides are stackable. In order to do this, we are using an implementation of a support vector machine, SVM^{light}[2], to classify our point cloud data's stackability. The training data is manually classified by a human in order to guarantee better accuracy in stackability recognition.

Since we use a simple rectangular prism model, we are assuming stackable objects are in general flat for their entire side of a box. Our features are therefore mainly related to the curvature of the surface. Working off of Jiang's features[1], we use a histogram of the height of the face to capture the general shape of the object along with a measure of the curvature of the surface directly. In addition to these features, we use simpler features such as the orientation of the face with respect to the table. In total, we are using 49 histogram features, 14 curvature features, and 4 simple features.

D. Bin-Packing and Visualization

Since we know what objects the robot is working with, as well as knowledge about the environment (object locations, fridge location, etc.), the first step is determining how to put the given set of objects in the fridge. In order to establish where the PR2 will place the objects we created a bin-packing algorithm that takes in rectangular prisms representing the bounding boxes of each object and their stackability properties, as well as the dimensions of the fridge, and returns the coordinates and rotations for each object to be put in the fridge. The algorithm takes a greedy approach to the problem by stacking the largest items first, and stacking from back to front and left to right.

The algorithm works by partitioning the floor of the refrigerator into cm-by-cm squares, each with an associated height. As objects are placed in the refrigerator, the height of each square is updated (at 1 cm resolution) based on the object placed on it, allowing the algorithm to take into account previously placed objects. If a square is not stackable, that location is treated as an invalid location for higher stacking; however, if a stacking orientation would place an object over the location, lower unstackable heights will not block the object from being placed.

We also created a visualization to go along with the packing algorithm, allowing the user to see how the objects are being physically placed. The screenshots in Figure 3 show the result of one run of the algorithm using a test set of objects we created. Note how the objects are stacked vertically as much as possible, which is an artifact of our test input. Also, there is some acceptable overhang between objects as long as it is not too great, shown in greater detail in Figure 4, which allows for superior packing results.

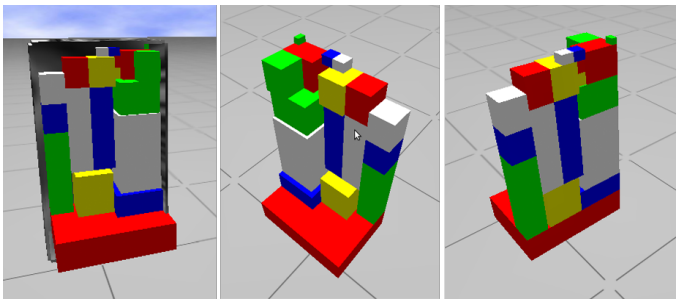


Fig. 3. (left) The objects stacked in a simple refrigerator model. (center) The stacking result without the refrigerator. (right) The back of the stack is flush with the refrigerator, wasting no space.

E. Object Grasping and Placement

We implemented basic pick-and-place capability into our custom ROS package using existing ROS packages and

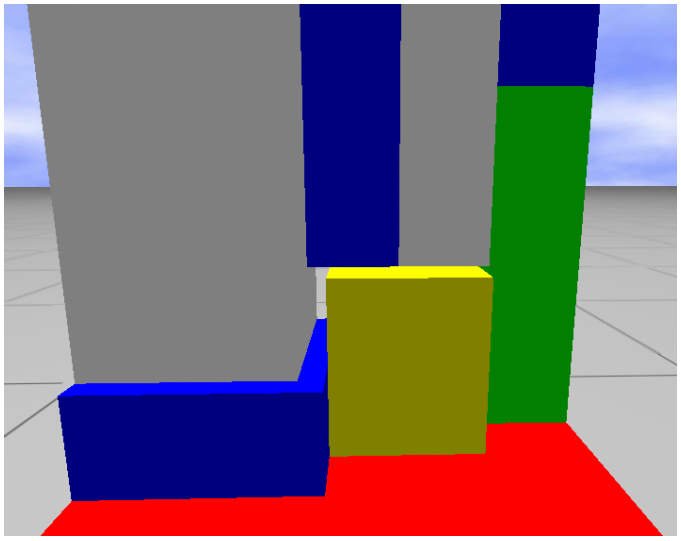


Fig. 4. The algorithm allows some overhang and gaps between objects if it provides for better packing.

examples, most notably the Pick and Place package for the PR2. This allows us to locate objects on a flat surface, have the robot turn to inspect the object, find a point to grasp the object from (if one exists), and plan the motion to grasp the object. Combining this with the ability to move the arm, rotate the base, rotate the end effector, and open / close the gripper (all granted from other ROS packages); we have all of the tools to relocate objects to the fridge. Figure 5 below shows a sequence of screenshots displaying the sequence of actions the robot takes when picking up an object.

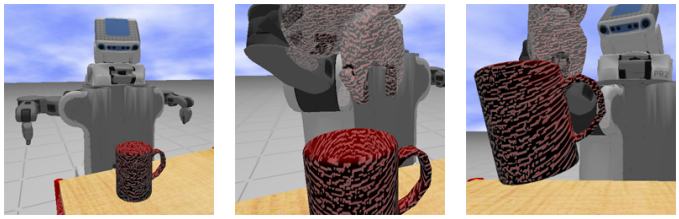


Fig. 5. The PR2 locates and picks up most objects on its own.

With all of these system components in place, piecing them together resulted in a straightforward system that goes through each object starting with the bottom-most, scans the area where it should be, locates and grasps the object, then turns to the refrigerator and places the object as specified. The resulting program can robustly find and grasp objects on the table, but has some trouble placing them. Due to the nature of the Pick and Place package, we could not use the place functionality to put items in the refrigerator. This resulted in hardcoding Cartesian movements to reach the destination for each object. The largest potential for error using this method is crashing into other already-placed objects, because they are ignored while the arm translates. Another shortcoming of the program is that it is incapable of placing objects in certain rotations. For example, in Figure 4 the PR2 is grasping the can from its sides. If the packing algorithm were to place the can on its side, the PR2 would be incapable of doing so, because its gripper would be in the way.

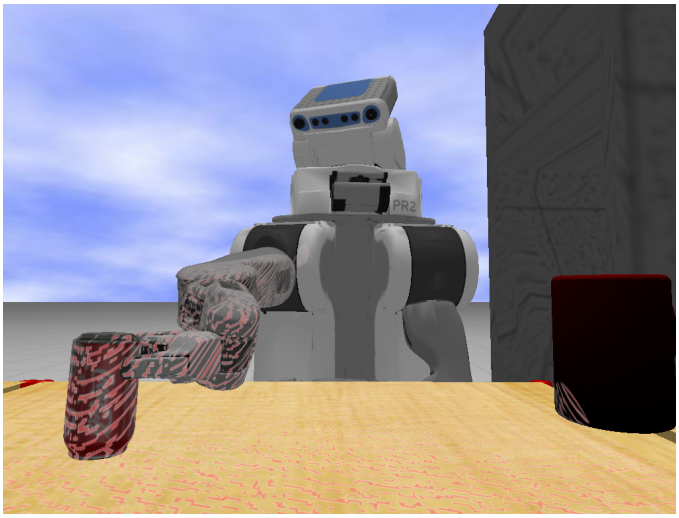


Fig. 6. The PR2 can pick up differently shaped objects such as the soda can

There are also some more general problems with the program. First among these is the long time it takes some of the features to run (such as constrained arm path planning). Such wait times often varies from one run to the next and could result in the PR2 taking as long as 15 minutes to pick up and place a single object in the refrigerator. Another issue is that objects could fall out of the PR2's grasp fairly easily if they were shaken around enough, and even when they did not fall out, they could be repositioned which changed its orientation when placed. Fixing these issues would be an important step in refining the project and getting the system ready for operation on a real PR2.

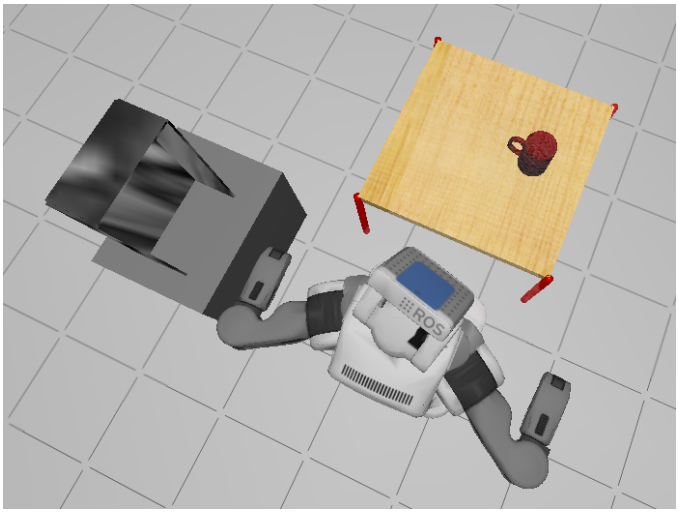


Fig. 7. A small test environment, containing a single surface, a cup, and the refrigerator.

F. Future Work

While we have implemented a system which defines the bounding box and stackability of objects as well as places them in the simulator, there is more work which could be done to further this work. Further testing and adaptation of these algorithms to a physical PR2 should be completed to further verify the results of this project.

In addition, integration of Alejandro Perez's "Interleaved Planning for Placement on the PR2" project would allow the process a more fine-tuned navigation in an enclosed space such as the refrigerator. Integration with Daniel Jeng's "Multi-object 3D Placing Constraints" would also possibly improve the compression of stacking items in the fridge by solving constraints over the items placed in the fridge rather than greedily filling it.

This process could also be adapted to work in other environments to pack objects such as a bookshelf or a warehouse. For more complicated situations and objects, both arms could be used to manipulate items when moving them into and out of the refrigerator.

IV. EXPERIMENTAL SETUP

A. Overview

For testing our packing and stackable face recognition systems, we have been using models of common household items that are often found in refrigerators. This data set includes items such as milk jugs, juice containers, and tupperware, which are all items we anticipate encountering during actual operation on the PR2. For testing our entire system from start to finish, we created a simulated environment in Gazebo, which runs everything using a simulated PR2. The environment is similar to the one we anticipate working with in the real world, although there will be differences in localization. Figure 8 shows our full test environment to check operation on multiple objects.

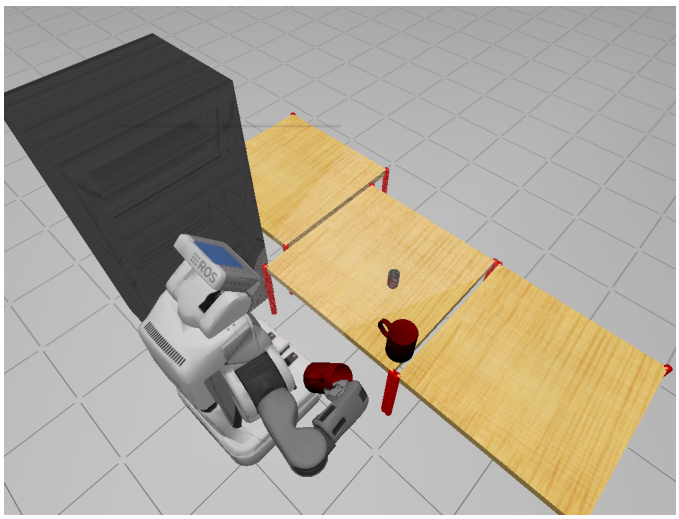
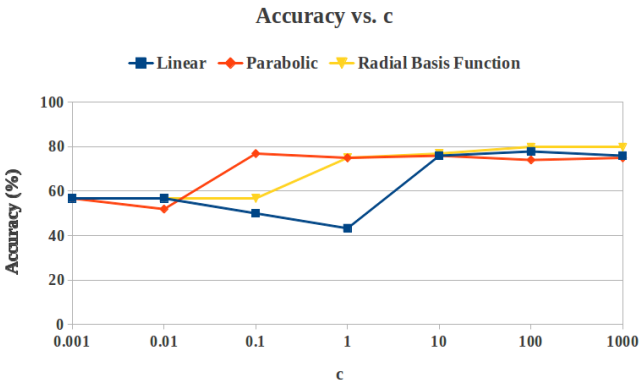


Fig. 8. Full size test environment containing multiple coffee cups to place in the refrigerator.

Our main task was to determine the effectiveness of the features chosen for the SVM. In order to do this, we needed a dataset which was large enough to accurately be able to test our features. Since our algorithm contains 67 features, we want a dataset which was at least on the order of the number of features. We gathered a total of 104 datapoints in order to test the program. We would have preferred more data; however, gathering data in Gazebo was a very lengthy process and good point cloud data for our test models was hard to come by. We used leave-one-out cross validation to confirm the accuracy of our features with our small dataset. Also, we tested how the robot performs the entire process by testing in a sample environment such as the one in Figure 8.

B. Results

Our features performed decently on determining the stackability of a face for our dataset. Our best results came from using the radial basis function kernel with a high c within the range of 100 to 1000 with an accuracy of about 80% as seen in Table 1. In general, the radial basis function outperformed the linear and parabolic kernels.



c	Linear	Parabolic	Radial Basis Function
0.001	56.73%	56.73%	56.73 %
0.01	56.73%	51.92%	56.73 %
0.1	50.00%	76.92%	56.73 %
1	43.27%	75.00%	75.00 %
10	75.96%	75.96%	76.92 %
100	77.88%	74.04%	79.81 %
1000	75.96%	75.00%	79.81 %

TABLE I
ACCURACY OF EACH KERNEL FUNCTION VS. C

Our data is somewhat consistent across c values; however, noise is still present in the data. There are many sources of noise which may have affected the data:

- A human must classify the stackability of a surface
- The dataset is somewhat small
- The stackability of a surface depends entirely upon its bounding box which may confuse a human classifier

When the simulated robot placed objects into the refrigerator, we had a somewhat low rate of success of placing objects in their correct locations. Most of these issues were based around issues in the PR2 navigation stack as the robot would need to take an extremely long time to pick, plan, and place

an object. While the robot was planning or moving, often the navigation stack would fail or crash without warning, leaving the robot to freeze in place.

C. Future Experiments

A more extensive dataset should be collected in order to ensure the accuracy of the chosen features. From there, the algorithm should be tested on completely unknown objects. Once the algorithm is verified in the simulator, it should be tested on the actual robot using known locations for the objects to place in the fridge. However, the stackability of the objects must remain unknown to test the accuracy of the chosen features on real noisy point cloud data. Improved methods of navigation should also be tested when the navigation into the refrigerator is improved.

V. CONCLUSION

In general, our algorithm will allow a PR2 to place an unknown object in a known location into a known refrigerator. Our accuracy of classifying stackable faces is approximately 80% which is reasonable for the loose definition of stackability. Our simulation is also capable of placing up to two items in the refrigerator. From these two basic metrics of our algorithm, it is feasible to be able to place a small number of items into the refrigerator without trouble.

Future research needs to be done into improved navigation in order for the PR2 to perform at its best in a constrained environment. Alejandro Perez's "Interleaved Planning for Placement on the PR2" project would help significantly in improving this algorithm's success rate at placing objects into the refrigerator.

ACKNOWLEDGMENTS

We would like to thank Yun Jiang and Ashutosh Saxena for the help with the organization and execution of this project.

REFERENCES

[1] Jiang, Yun, et al. Learning to Place New Objects. To appear in International Journal of Robotics Research (IJRR), 2012.

[2] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.