Recycler Bot
CS 229 Project Report

Jiahui Shi
Advisors: Morgan Quigley and Alan Asbeck

***Abstract*— We consider the application of using Personal Robot 2 (PR2) to sort out different categories of recyclable trash on a conveyer belt, currently including glass bottles, plastic bottles and cans. The robot will be able to find and recognize these items and send them to the right recycle bins. We will describe the algorithms which incorporates visual and tactile perception, show the current results and briefly talk about some ideas we would like to explore in the future.**

I. Introduction

Collecting and sorting recyclable trash is a trivial and tedious task for humans. There are lots of recyclable items left on messy tables or placed in wrong recycle bins and wasted. In this project we will let PR2 take over the job of recycling trash. There are a few scenarios we considered to work on. One is to let PR2 move around in a building, collect recyclable items and send them to recycle bins. Another scenario is to look into a recycle bin and pick out the misclassified items. The third scenario is to stand in front of a conveyer belt with trash on it, then pick out and sort the items of different categories. This project tries to achieve the third scenario, while solving this problem will greatly help solve the other two as well. The robot uses stereo cameras to find recyclable objects on a conveyer belt, use the pressure sensor array on its gripper to sense the materials, classify them into different categories, then grasp and throw them into proper recycle bins.

This project involves visual perception, tactile perception and grasping. Currently, we are working on three categories of objects: cans, glass bottles and plastic bottles. We use vision information to find objects and further incorporate tactile information for classification. Now the bottles and cans are described as cylinders with and without bottle caps. We fit the 3D point clouds retrieved from the stereo cameras to cylinder models. The features include the height and radius of the cylinder and the existence of bottle caps. The tactile features we used are the relationships between the fingertip movement distance, the total force and the force variance on the pressure sensor.

II. Data from sensors

There are two pairs of stereo cameras installed on PR2's head with two different baselines as shown in figure 1. There is also a projector which can project texture in front in order to find the depth of textureless regions. We used both the narrow stereo camera and the wide stereo camera with the texture projector turned on. The narrow stereo camera retrieves more accurate 3D point clouds in the projected area while the wide stereo camera finds larger but more noisy point clouds. We only need the wide camera to find bottle caps since they are missing in the point cloud from the narrow camera.

Fig. 1. Two pairs of stereo cameras

Fig. 2. Fingertip sensor

The tactile information is sensed by the two pressure sensitive fingertips on PR2's left gripper as shown in figure 2. The pressure sensor array on each fingertip comprises 22 pressure elements as illustrated in figure 3: a 3×5 array on the front, one on the back and 3 on each edge. We can also know the distance between two fingertips from the joint encoder on the gripper.

III. Object detection from stereo vision

This part introduces how we detect objects from the 3D point cloud. The general idea is to first remove the background, then find big clusters in the remaining point cloud, fit the clusters to cylinders, find bottle caps above the cylinders and apply some shape features to classify the objects.

A. Background removal

In order to isolate the point clouds of target objects from the background, we first detect the conveyer belt and

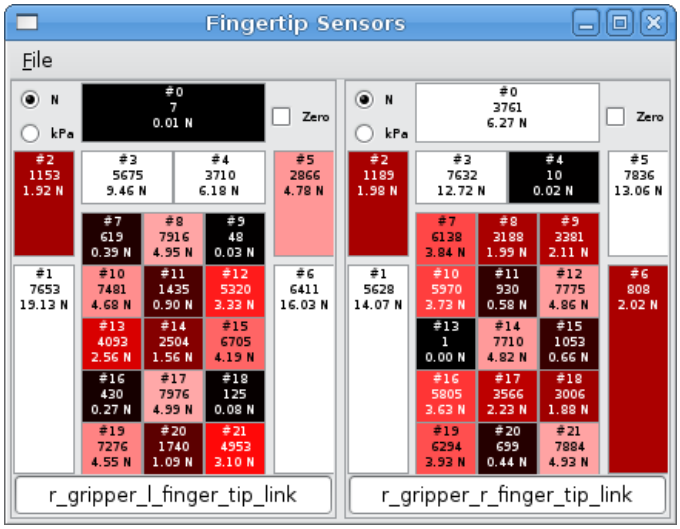

Fig. 3. Output from fingertip sensors

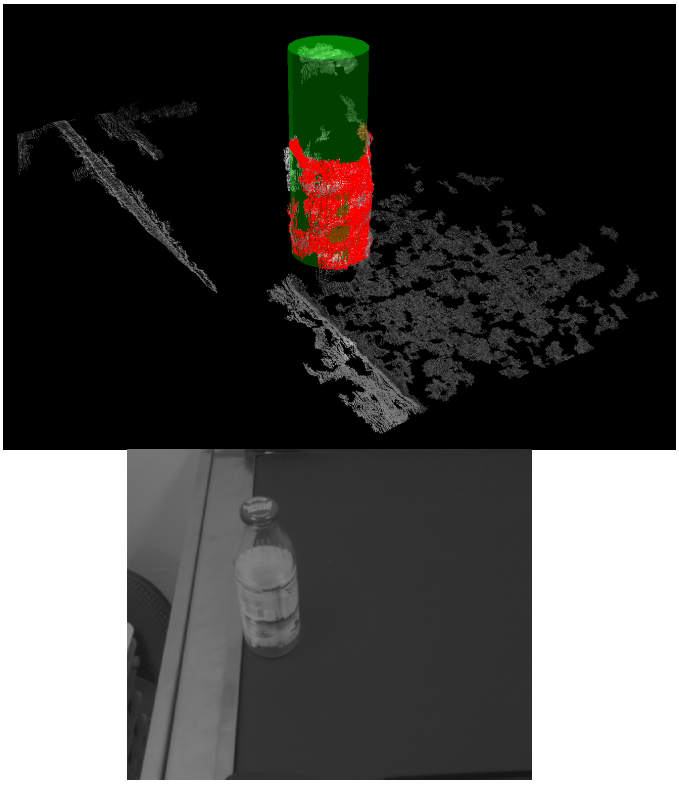

Fig. 4. Bottle detection from a 3D point cloud

remove all the points except for those above the scope of the conveyer belt. We first discretize the space into a 3D grid. The cells with a significant number of points are marked as occupied. Then we build a histogram according to the z axis and find the z value with the largest number of occupied cells. This value is regarded as the height of the conveyer belt plane. In order to get rid of the points higher than the plane but not directly above the conveyer belt, we find the x-y scope of the conveyer belt plane, leave a small margin and only keep the points within this scope.

B. Clustering using ANN (Approximate Nearest Neighbor)

After removing the background, we get point clouds of the surfaces of cans and bottles, which are noisy and in the shape of multiple cylinders. As plastic and glass bottles are partially transparent, we only have the point clouds of their labels. The clusters and corresponding cylinders are found iteratively one by one in a greedy manner.

In each iteration, we apply ANN (Approximate Nearest Neighbor) to find the biggest cluster in the point cloud. Here we use STANN (Simple, Thread-safe Approximate Nearest Neighbor) C++ Library[1]. This library provides the algorithm to find the k nearest neighbor points of a query point. By iteratively finding the nearest neighbor points of each of the k neighbor points, we get a cluster. k is set to be 5. After finding all the clusters, we pick the largest one. The search stops when the largest cluster is too small.

C. Cylinder fitting using RANSAC

Within the largest cluster we found, we use RANSAC (RANdom SAmple Consensus) to fit a cylinder model. Since currently we only work on the standing up cylinders, we can project the 3D points in to the x-y plane and fit the 2D points to a circle. For each iteration, we pick three points randomly and find the circle defined by these points. By allowing a tolerant margin around the circle boundary, the number of cluster points within the margin is counted as the consensus. Repeat this process until the maximum iteration is reached or a circle with a large enough consensus is found. Then the optimal circle for the all the consensus points is found by least-square fitting. Thus, we get the radius and position of a cylinder. The points belong to this cylinder will be removed from the point cloud for the next cylinder search iteration.

D. Bottle cap detection

The existence of a bottle cap is an intuitive feature to distinguish between cans and bottles. Since the point clouds of caps sometimes do not appear in the narrow stereo point cloud, here we use the wide stereo point cloud though it is more noisy. The cap is found by locating a small cluster above each cylinder found in last step. The height of a bottle is the distance from the top of its cap to the conveyer belt.

The result of object detection from stereo vision is shown in figure 4. The grey points is the original point cloud from the narrow stereo camera. The red points is the largest cluster. The cylinder which bounds the bottle cap indicates the position and size of the bottle and its green color means it is classified as a glass bottle from its appearance.

E. Comparison with object detection in 2D images

We also considered to detect the bottles and cans directly from a 2D image in order to find them beyond the scope of stereo vision. We tried the code of Felzenszwalb's Discriminatively Trained Deformable Part Models [1] with the

[1]Available online: http://sites.google.com/a/compgeom.com/stann/

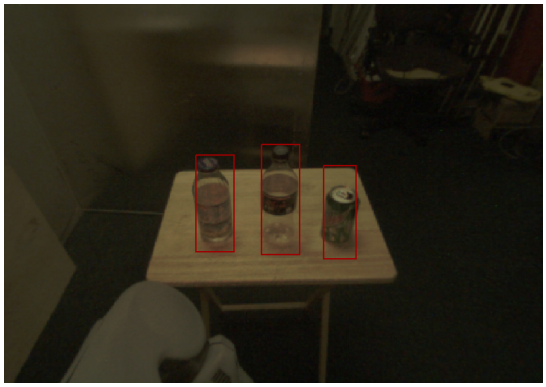

Fig. 5. Bottle detection from a 2D color image

result shown in Figure 5. After running for about half a minute on Matlab, the three standing up bottles and cans were successfully detected. But when they are placed in another orientation, for example, laying down, the algorithm failed to find them. It might be possible to slightly modify the algorithm to adapt to different orientations, but we will leave this problem to the future and focus on stereo vision at this moment.

Our algorithm on 3D point cloud is more efficient. In our final experiment, we actually only need to find one largest cluster and the corresponding most confident cylinder for each grasping. The vision processing can take less than one second though it depends on the amount and quality of the 3D points.

IV. Tactile perception

It is very hard to distinguish between plastic bottles and glass bottles only from the visual feedback. Here we utilized the tactile perception. We let PR2 use its left gripper to squeeze an item and sense the material. After setting the maximum effort of the gripper, the fingers will gradually close and squeeze the item. The feedback from the fingertip pressure sensors is a 22 dimensional array. Here we only consider the 3×5 array on the front of fingers. We observed the total force on all sensors, the variance of force on different positions and the gripper movement distance. Their relationships are illustrated in figure 6 and 6. From figure 6, the plot of total force versus gripper movement distance, we can find that the can has multiple sudden big changes of shape, the plastic bottle changes more smoothly, and for the glass bottle, the gripper stopped moving at a certain point since it cannot squeeze it. Figure 6 illustrates the relationship between the total force and the variance. The force variance on cans is smaller than that on plastic bottles because the touching area of a can gets flatter than that of a plastic bottle. There can be used as features for classification.

V. Grasping

After successfully finding an object from vision, the robot will grasp and send it to a proper place. In order to avoid touching other bottles, the gripper first moves above the found object. Then it moves down and close the gripper. At the same time, it squeezes the object and do the tactile classification. We used the robot arm motion planning algorithms provided in ROS [2]. Now the robot is able to grasp an object using either the left or the right arm. We will design a planning algorithm for two arm grasping in the future.

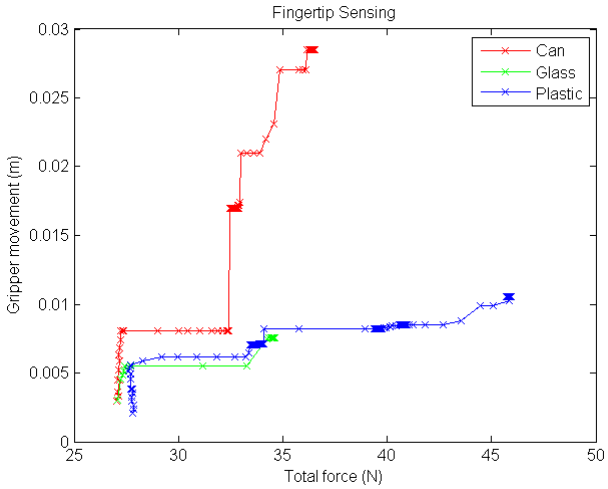

Fig. 6. Total force versus gripper movement distance

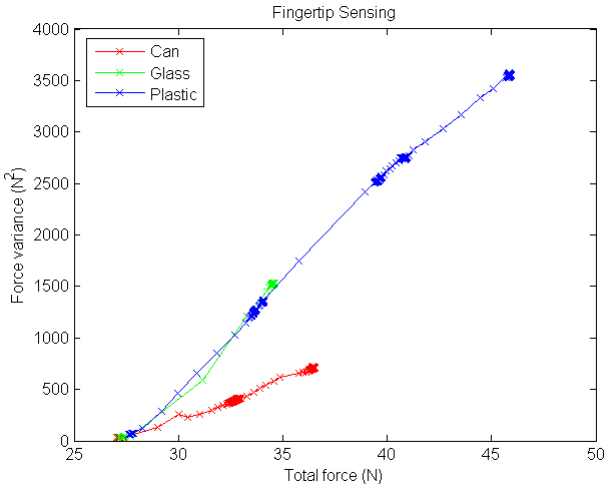

Fig. 7. Total force versus force variance

VI. Object classification

To classify the detected object, we use the tactile features as described in the previous section, the height and radius of the cylinder, as well as the existence of a bottle cap. The result is shown in figure 8. The upper picture is from the left lens of the narrow stereo camera. The lower picture is the result shown in the 3D point cloud, where we use red cylinders to mark cans, green for glass bottles and blue for plastic bottles.

Fig. 8. Classification result in the 3D point cloud

VII. Results

The classification works for almost 100% accuracy. The major cause of failure is the occasional inaccuracy and damping of arm motion, which may push over the bottles. Figure 9 shows a sequence of pictures of how PR2 works. It can successfully sort out a variety of bottles as shown in figure 10. A 2-minute demonstration video is available online at http://www.youtube.com/watch?v=dok2WppuJiA .

VIII. Future work

We will apply more robust algorithms in order to deal with more complicated situations, including objects in arbitrary orientations, occluding each other and finally piled up. In such complicated cases, we will try incorporating 2D color images with 3D point clouds for the segmentation and object detection. We also want to include more categories of recyclable items in the future, such as paper boxes and bottles. More advanced algorithms for tactile sensing will help distinguish between different materials.

IX. Acknowledgement

The project has been made under the guidance of Morgan Quigley and Alan Asbeck. It would not have been possible to implement this idea without their sound ideas and direction.

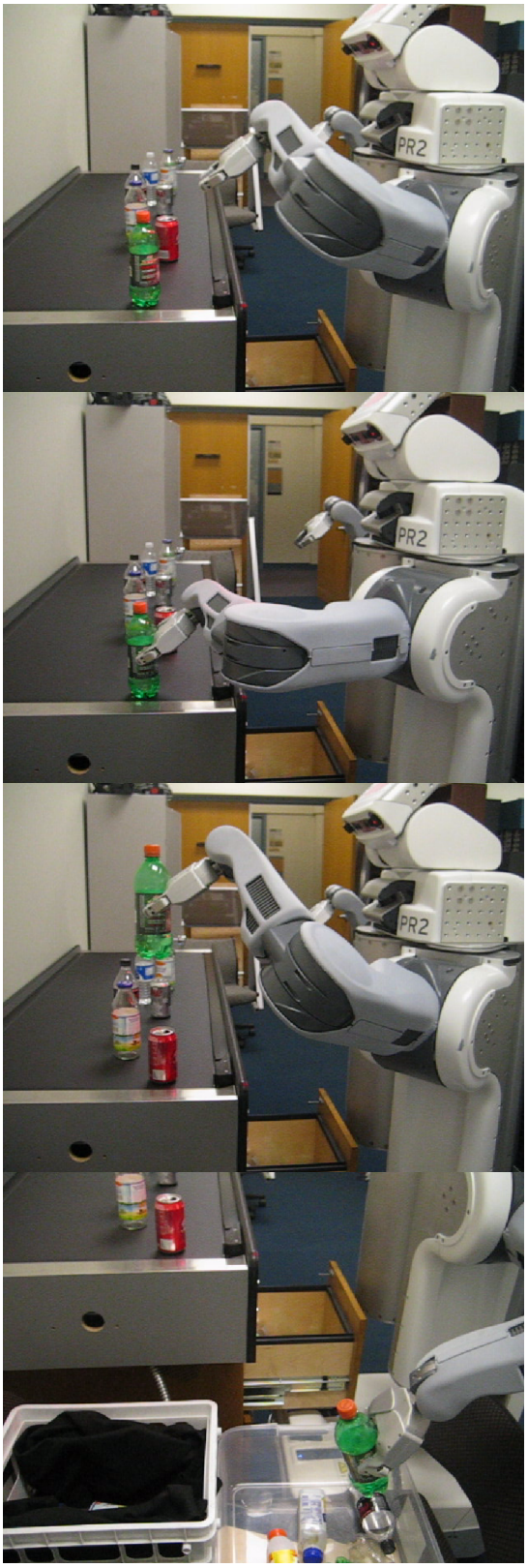

Fig. 9. PR2 working in front of a conveyer belt.

Fig. 10. Result of sorting a variety of bottles.

REFERENCES

[1] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, Discriminatively Trained Deformable Part Models, Release 4, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 9, September 2010.

[2] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, ROS: an open-source Robot Operating System, in Proc. Open-Source Software workshop of the International Conference on Robotics and Automation (ICRA), 2009.