CS 4758 Robot Learning: Beer Pong Butler

Kimberly Sheriff - kgs45
Brian Toth - bdt25

Abstract— **This paper describes the application of the PR2 robot as a "Beer Pong Butler", which will identify a ping pong ball in a cup, pick up that cup, and move it to another location. We utilize Hough circle detection using OpenCV and an SVM to detect a ball in a cup. We then use motion planning to perform the task given the location of the cup containing a ball.**

I. Introduction

Beer pong, also known as Beirut, is a game typically played at college parties, which involves two teams of two players each throwing ping pong balls across a table with the goal of getting a ball into a cup of (root)beer at the other end. Figure 1, shows the typical set up for a beer pong game. For our application, we will assume a game played with six cups on each side, which will be empty for our purposes.

When a ball is successfully thrown into a cup, that cup must be removed from the game. The beer pong butler will identify the cup that contains a ball and move that cup away from play.

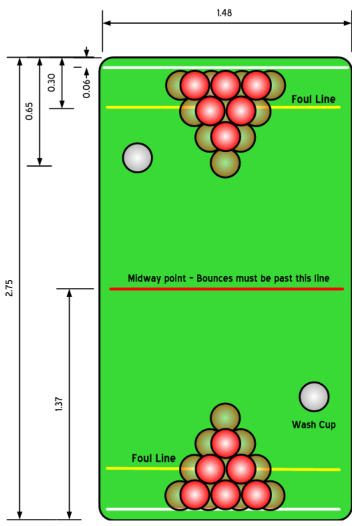

Fig. 1. Typical beer pong setup.

A. Related Work

Willow Garage has implemented a PR2 which responds to a "Beer Me" app that allows users to request a beer be brought to them from the fridge. The PR2 navigates to the fridge, uses handle recognition to open the door, and object recognition to determine which beers are in a rack in the fridge. The fridge is modelled using live perception and data from several sources according to a representative from Willow Garage. The PR2 uses motion planning, presumably some type of inverse kinematic solver, to determine the path of the arm to the beer to be picked up.The PR2 is also capable of opening the beer bottle. This algorithm provides the makings of a full service party bot, and our application attempts to provide additional functionality.

Solems Vision Blog explains how to use Hough transforms and OpenCV to detect lines and circles on an image of a guage. HoughCircles is a documented function which takes the image, the detection method to use, the inverse ratio of the accumulation resolution to the image resolution, minimum distance between the centers of the detected circles, two threshold values, and the minimum and maximum radii. These parameters must be tweaked for different applications.

SVMs are becoming widely used for image classification. David et al. use SVM on genetic syndrome diagnosis which requires image classification. The results shown in Table 6 of their report shows a comparison between the error rates of the SVM to other machine learning algorithms including 7-nearest neighbor, neural network, and naive Bayes. The results show that SVM has the lowest error rate of the compared algorithms. Anthony et al. use SVM for land cover mapping. Their results show high accuracy using both one-versus-one and one-versus-all approaches, and they conclude that the choice between the two methods for image classification is simply personal preference.

From a high level, our approach is to detect the remaining cups, determine which cup has a ball in it, find the X,Y, and Z coordinates of that cup in the base_link_frame, and remove that cup from the formation. This involves learning what a cup looks like and planning motion to a specified point.

II. Perception

The most complicated and novel part of our project is detecting which cup contains a ball. Although this sounds simple at first, it rapidly becomes complicated when the possibilities of changing environments, lighting, cups, and viewing angles are considered.

A. Dataset

To assemble our dataset, we took pictures of formations of cups using the Kinect. A cup formation is created by placing six cups in a pyramid shapes and then removing between zero and five cups. To make sure that our data was faithful to the data that would be gathered by the PR2, we placed the Kinect at a height and angle consistent with the PR2's technical documents. To simulate real-world conditions, we made small adjustments to the height, angle, and position of the Kinect while capturing data, all while varying the lighting conditions. In some cup formations, one cup holds a ball; in others no cups contain balls. In total 120 pictures were taken. Each of these pictures was labelled with a coordinate pair corresponding to the approximate center of the cup which contained the ball (if no ball was present, the pair (-1,-1) was used). The center of the cup was used, because this would be the ideal point for the robot hand to enter the cup.

50 pictures were reserved for testing and 70 were used for training. The 70 training examples were sometimes further split into training and validation groups if the particular algorithm being tested required such a split. Before splitting the data, each picture was manually processed to isolate the cup formation.

B. *Tabletop Detection*

In order to segment the tabletop from the objects on the table, we used an existing table segmentation package. For example, we segment the hands and keyboard from the table in Figure 2. Although we were successful in doing so, the resulting pointcloud was not dense enough to actually determine which cup contained the ball as shown by Figure 3. To solve this, we overlay the segmented pointcloud data on the RGB image in order to isolate the portion of the RGB image that contains the cup formation, Figure 4. This allows us to use the superior pixel density of the RGB image to determine which cup contains the ball but provides an easy way to discard most of the image.

Fig. 2. RGB image taken with Kinect

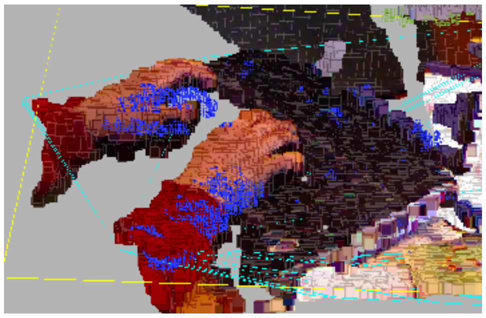
Fig. 3. Point cloud data

The purpose of detecting the tabletop and removing it from consideration is to isolate the pixels which form the triangle of cups. Although the image segmentation is not fine-grained enough to detect individual cups, the task of picking out cups

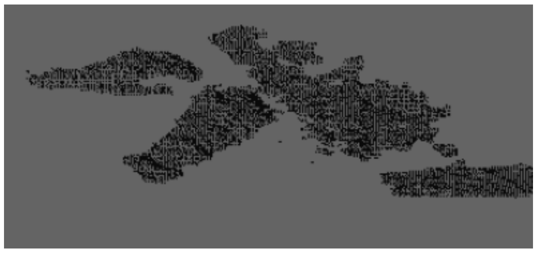
Fig. 4. Point cloud data translated to RGB image

from the RGB image is greatly simplified when the image can be cropped to a ”blob” which closely follows the cups.

Shown here (clockwise from the right) are the registered pointcloud, the RGB image, and one result of tabletop segmentation projected onto a two dimensional image. Tabletop segmentation returns significant clusters of points which appear above the tabletop surface. In this case, one such cluster is the hands and keyboard of the person working at the computer. The cluster is not yet properly aligned with the RBG image because we have not yet determined which method of coordinate transform is necessary to transform from the Kinect’s 3D coordinate frame to its 2D coordinate frame. Methods attempted include using ROS’s tf, projecting the 3D coordinates on to the plane of the observer and rotating that plane, converting the pointcloud to an image directly using ROS’s CloudToImage, and simply removing the Z coordinate of each point (most successful, pictured).

C. *Hough Circles*

The second step in our filtering pipeline is the detection of individual cups. Although table segmentation provides a blob which contains the cup formation, this blob also includes many unnecessary pixels which can be confusing to the ball detection algorithm. In order to further narrow our search, we use OpenCV’s Hough circles to exploit the geometry of the cups.

Unfortunately, slightly varying the parameters for the Hough circles function results in the generation of wildly different circles. Specifically the *minDist* (minimum distance between the centers of detected circles), *minRadius* (minimum radius of the detected circles), and *maxRadius* (maximum radius of the detected circles) are difficult to tune by hand, because they are so dependent on the exact position and angle of the Kinect sensor. Attempts to make these parameters simple functions of the size of the image failed to capture the relatively small variances in our dataset. Even when tuned correctly, the circle detector frequently recognizes the shadow of a cup as a circle and ignores furthest cup because the aperture of that cup appears to be an oval from the camera’s perspective.

It is at this point that we decided to apply machine learning in order to discriminate between the circles which represented actual cups and those which were merely random coincidences. The Hough Circle step of the pipeline takes a brute force approach to the parameter-tuning issue and simply varies each parameter slightly in an empirically determined range. This results in many accurate circles, but also many misses.

From our 70-image training set, we generated approximately 13,000 individual circle images by running the Hough circle function with various parameters. Because this was far too many images for us to manually label, we selected 500 random images and labelled only these. Circles were labeled as not a cup, Figure 5, a cup, Figure 6, or a cup containing a ball, Figure 7.

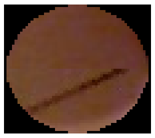
Fig. 5. Individual circle without a cup

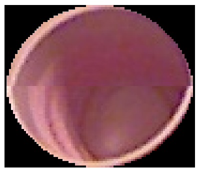
Fig. 6. Individual circle with a cup

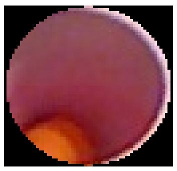
Fig. 7. Individual circle with a ball

We considered three distinct algorithms to determine which cup (if any) contains a ball. Although none of the algorithms performed incredibly well, the ”direct ball detection” algorithm provided the best results on the testing set.

Our first approach was to determine the images that were in fact images of cups, and then find which of those contained the ball.

D. Cup Detection

We considered a number of alternatives for using machine learning to bolster cup perception. Although OpenCVs Hough Circles do a good job of delimiting cups when properly tuned, as shown in Figure 9, they can perform erratically when the parameters are a bit off. Slight changes in environment, lighting, height, or angle can result in incorrect or incomplete labelling of cups as seen in Figure 8.

Our first idea was to implement a combined HMM and SVM to determine whether we had properly identified the cups. The problem naturally lends itself to an HMM, because there are a small number of discrete states (configurations of the cups), and the game naturally transitions from one state to another. Even better, the transition probabilities are

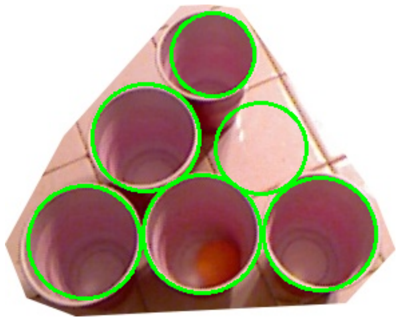
Fig. 8. Good Hough Circles

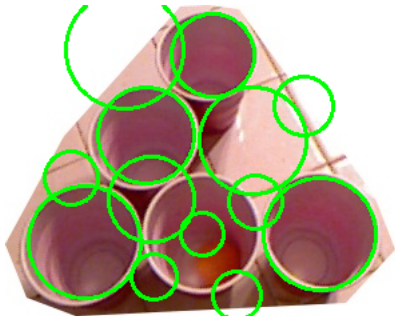
Fig. 9. Bad Hough Circles

certainly not uniform and can be determined experimentally simply by throwing balls into cups. Although we were very excited about this approach, it was abandoned due to time constraints. Because each unique configuration would require a custom SVM to be trained, there would have to be 6+6+15+15+20+1+1= 64 different SVMs. Gathering sufficient data to train and validate this many SVMs would not be practical.

1) SVM-based Cup Detection: The simplest cup classifier simply looks to determine, for each drawn circle, if that circle represents a valid cup. Therefore this classifier considers only one cup at a time. The features (all drawn from the input image) are the width, height, number of pixels, the average red value of the pixels, the average green value of the pixels, and the average blue value of the pixels. These features represent a simple intuitive grouping of the parameters used to describe a picture.

We used SVMLight to train a classifier using this information. First the training data (the 500 labelled circle images) was normalized so that the classifier would not be inherently biased. An extreme example of this would be a training set composed of 90% positive and 10% negative examples; the classifier could simply label everything as positive and achieve 90% accuracy! Our labelled images contained 203 negative examples and 297 positive examples, so we used 203 of each. These 406 images were shuffled and split into five groups. Using these groups we performed five-fold cross-validation in order to determine the best trade-off between training error and margin (c value) for the classifier as shown in Figure 10. With the appropriate c value selected, the final classifier was trained using all 406 examples.

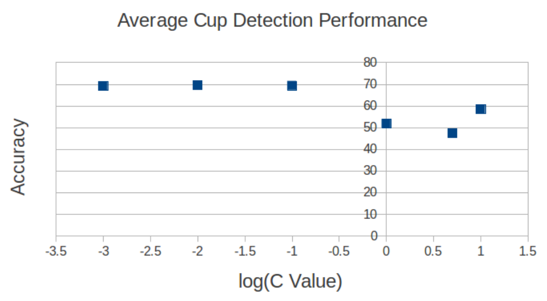

Fig. 10. Cup Detection Performance

This classifier determines which circles (generated by the Hough circle function) qualify as cups, but it does not solve the problem of determining which cup contains the ball. This is performed by a simple scan of the cups for orange pixels. This process introduced another parameter to tune; the cutoff for the minimum number of orange pixels that could qualify as a cup (recall that some configurations of balls contain one ball and others contain none). To select the proper value for this parameter, the training set (70 images of between one and six cups, sometimes containing a ball) was evaluated using a number of different parameter values. Each image in the training set was manually labelled with a point indicating the approximate center of the cup containing the ball (for cup formations not containing a ball (-1,-1) was used). For each of these images, the Hough circles function was used to generate a number of circle images. Then the classifier was applied to find the circles which corresponded to cups. The number of orange pixels in each of the cups was counted, and the cup which contained the largest number of orange pixels was selected. If the number of pixels present was greater than the threshold, the classifier predicted a ball at that location. If not, then the classifier predicted no ball in the formation. After repeating this process for each parameter value, the value which generated the greatest accuracy was selected (see figure 11).

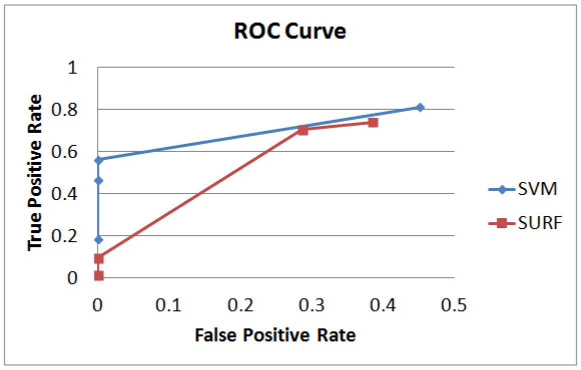

Fig. 11. ROC Curve

2) SURF-based Cup Detection: The second classifier also attempts to identify individual cups, but took a very different approach to the process. SURF (Speed Up Robust Features) was used to categorize circles as either cups or not. Because SURF works by identifying keypoints in an image, computing the descriptors for those keypoints, and then comparing those descriptors to the descriptors calculate from the keypoints of another image, it is most similar to KNN. In order to leverage the previously-labelled dataset, we created a classifier that takes a circle image and computes its similarity to a collection of known cup images using SURF. The smallest 'distance' found this way is chosen to represent the image. As before, another parameter is introduced: the largest acceptable 'distance' which still qualifies a circle image as a cup. Once again, this parameter was varied to determine the best value to use. In order to avoid contaminating the results, five-fold cross-validation was used. The 297 positive examples were split into five groups. For each of these five groups, the other four groups were used as the database of known cup images. The leftover group was combined with an equal number of randomly selected negative images (approximately 60 of them) and run through the classifier. As above, the accuracy of the classifier was determined for various threshold values, and the results were averaged across all five splits. The final classifier used the best threshold value, as well as a known cup database consisting of all 297 labelled positive examples.

As with the previous classifier, the second classifier only handles the task of task of determining which circles contain cups. The same approach described two paragraphs above was used to translate this into a classifier that can determine which cups contains a ball (see figure 11).

3) Ball Detection in Unfiltered Circles: Since both of the above methods simply add a layer of abstraction on top of simple ball detection, it made sense to try the ball detection on its own to make sure that the more complicated methods actually add value. In the usual way, we used the Hough circle function to generate cup images and tried a number of parameter values to determine which worked best on the training set. We found that this method was less effective than the two above methods, confirming our intuition that some circles contain orange pixels that were not part of a ball in a cup.

E. Direct Ball Detection

1) SVM-based Ball Detection: After attempting to first find the circles that contain cups and then determine from among those the cup which contains the ball, we realized that it might be more effective to simply use a classifier to find the ball directly. After all, the Hough circles already did much of the work in separating the cups from the rest of the image, and adding the additional requirement that an image has to contain a ball to be a positive example creates new features. Specifically we added the number of orange pixels in the image as a feature in the classifier. As detailed in the SVM-based cup detection section above, five-fold cross validation was used to determine the best c value.

Much like with the SVM-based cup detection above, we then had to determine a threshold for ball detection. Once again, we followed the methodology described in the second paragraph of that section. This time the threshold was placed on the output of the classifier generated by the SVM instead of the count of orange pixels. The threshold which produced the highest accuracy was selected for this classifier.

2) *SURF-based Ball Detection:* We also attempted to perform direct ball detection using SURF. However, this failed due to the incredibly small size of the ball images (typically 34x34 pixels). OpenCV's SURF function simply could not find keypoints on these images.

F. Results

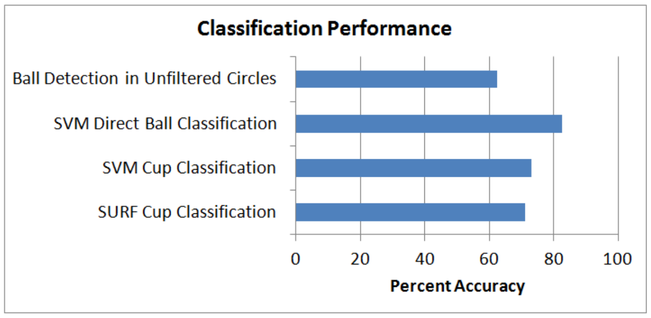

Fig. 12. Results for classification methods

Our ultimate evaluation metric was, of course, performance on the 50 reserved cup formation images. Although none of our classifiers performed exceptionally, direct classification of cups with balls using an SVM was clearly the best method. Upon reflection, this is the approach we should have taken initially; by starting with a simpler and better classifier we could have avoided much unnecessary work. Furthermore, the addition of more features to this classifier could only improve its performance.

Our attempts to isolate individual cups were not totally futile. It is quite clear that even methods which are only moderately successful at distinguishing cups from non-cups (the SVM and SURF based classifications) still manage to weed out some non-cups that confuse the ball detection formula when run on unfiltered circles. Although the SVM direct ball classifier already incorporates all of the features from the SVM cup classifier, it may be possible to improve it by incorporating the SURF-based classifier.

III. Motion Planning

The overall goal of the motion planning is to remove the cup, which was determined to contain a ball, from the cup arrangement. To accomplish this goal, a model of the situation was created in gazebo and inverse kinematics were used to plan the right arm's path. The motion plan assumes that the PR2 is positioned in front of the cup arrangement as shown in Figure 13 with its arms positioned straight in front of the PR2 body and the grippers pointed upwards at an angle. The motion plan uses only the right arm of the PR2 to complete the goals.

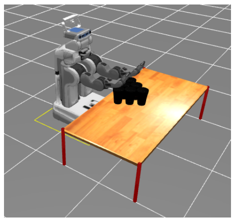

Fig. 13. PR2 completing motion task in an empty world.

A. Gazebo Models

The beer pong scene was modelled in Gazebo using two different object types. The first was the default table object. The second was a plastic cup object to represent the commonly used red plastic SOLO cups used in beer pong. The plastic cup model was sourced from the Google 3D Warehouse. An xml file was created to allow the object to be spawned in the Gazebo world. The cup object had to be scaled in order to fit the scope of the gazebo world. The cup was scaled to 4% of it's original size to be the approximate size of a real cup with respect to the PR2 and the table in the Gazebo world. Originally the collision map was set to be the same size as the visual cup in gazebo. However, as will be described in the inverse kinematics section below, that size collision map was not ideal. The collision map for the object was shrunk to 3% of it's original size compared to the visual cup which was scaled to 4% of it's original size. By making the collision map smaller, the gripper was able to move closer to the cup object and inverse kinematics was able to successfully be solved.

The cup objects were spawned into the gazebo world using the following coordinates as seen in Figure 14:

Cup 0 (0.7, 0.15, 0.52)
Cup 1 (0.7, 0.0, 0.52)
Cup 2 (0.7, -0.15, 0.52)
Cup 3 (0.83, 0.075, 0.52)
Cup 4 (0.83, -0.075, 0.52)
Cup 5 (0.96, 0.0, 0.52)

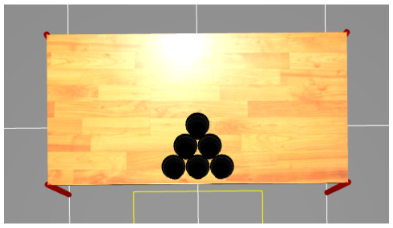

Fig. 14. Cup arrangement in gazebo

These coordinates were chosen because they mimic the real set up of a six cup game, the cup edges do not touch, and the cups do not collide with the table object. By spawning the cups such that they do not touch any other object, the cups are prevented from flying off in a collision.

B. Gripping

Several gripping techniques were explored in the search for the optimal approach. The two main techniques considered were: gripping one side of the cup and expanding the gripper inside the cup. In the first method, the grip would be positioned such that it is point down vertically at the edge of the cup. The gripper would also be positioned with one side of the gripper on the inside and outside of the plane made by the side of the cup. The gripper would be opened and moved down so that the grippers are on either side of the cup. The gripper would then close and raise the cup away from the arrangement. This method would require the gripper to be positioned fairly accurately above the center of an edge. Furthermore, this method would be difficult to use in a ten cup set up because it would be hard to get to the edges of the middle cup.

The second method, the gripper would be moved so that it is was pointing vertically over the center of the cup. The gripper would then be lowered into the cup and opened such that it applies a constant pressure to the interior sides of the cup. The gripper would be opened a set distance which would be determined through trial and error using the real PR2. The cup would then be lifted up away from the arrangement with the gripper held open to secure the cup. This method would work for either inner or outer cups as well as cups of different shapes and sizes. Additionally, this method would still work if the gripper was not placed exactly at the center of the cup. The second method was chosen as the final gripping method.

C. Inverse Kinematics

The overall goal of the inverse kinematics is to move the right arm of the PR2 to complete the following tasks:

1) move the gripper such that it points down over the center of the cup
2) move the gripper down into the cup
3) spread the gripper to secure the cup
4) move the gripper up and away to remove the cup from the arrangement

The x,y,z coordinates of the cup we want to move are taken from perception and transformed into the "base_link" coordinate system from the Kinect coordinate system using the tf package.

The first step was to insure that the PR2 could successfully complete the motion without violating joint limits in an empty world as seen in Figure 15.

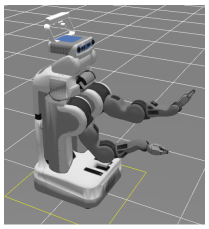

Fig. 15. PR2 completing motion task in an empty world.

After the same movements were confirmed to possible with the table in gazebo, cup 1 was spawned into gazebo and the inverse kinematics was again repeated. However, with the cup in the gazebo, the inverse kinematics failed. The issue was with the collision map of the cup. The collision map was shrunk to be slightly smaller than the visual cup. By shrinking the collision map, the gripper could move closer to the cup and the inverse kinematics were successful. The collision map size was tuned to be small enough to allow inverse kinematics to solve but large enough such that the gripper would not go straight through the visual cup. When the collision map was too small, the arm went straight through the cups and collided with the table as shown in Figure 16.

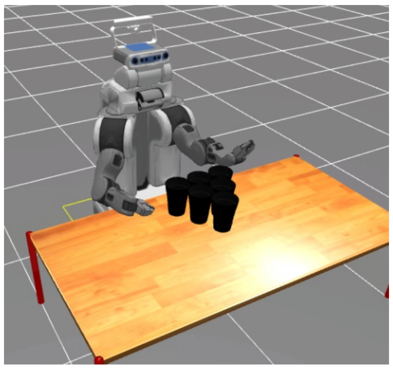

Fig. 16. Effects of too small of a collision map

Finally, the motion plan was executed in simulation with the full cup arrangement. We were not able to use the real PR2 because it was not operational. The gripping motion was not able to be performed in gazebo because the physics of gazebo is not ideal. However, we are confident that the designed gripping method we decided on would work if we were able to calibrate it on the real PR2.

The PR2 first moves the gripper over the cup which will be moved as seen in Figure 17. Next the gripper would be spread to secure the cup in the real world scenario.

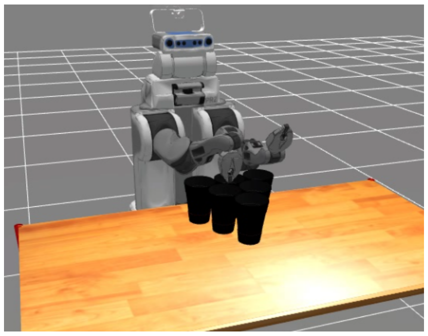

Fig. 17. PR2 gripper over cup 1

Finally, the PR2 moves the gripper away from the robot while maintaining control of the cup shown in Figure 18.

D. Other Considerations

We began by working with pick and place, but this algorithm does not work for our application because we are not picking up a stand alone object.

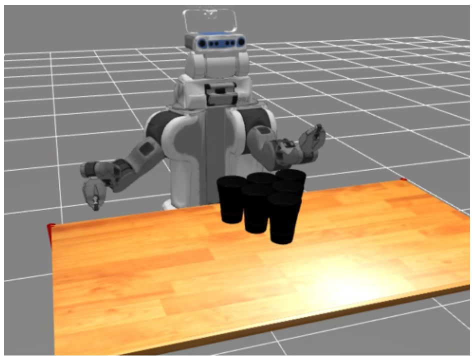

Fig. 18. PR2 gripper moved away from arrangement

IV. Conclusion

The problem of identifying a ball in a formation of cups proved to be surprisingly difficult. Small changes in angle, lighting, and arrangement confounded naive approaches. Furthermore, the presence of random objects made the challenge even greater. Even after isolating the cup formation using tabletop segmentation and singling out likely areas using a Hough circle function, our best classifier only achieved 83% accuracy on our test set. The classifiers we tried, in order of accuracy on the test set, were ball detection on unfiltered circles, SURF cup classification, SVM cup classification, and SVM direct ball classification. Our hope is that a real robot would be able to adjust its position and the angle of the Kinect slightly in order to gather multiple images and make a more informed decision about the presence of a ball.

The motion planning was mostly successful given that it was entirely run in the simulator. The inverse kinematics solver was sufficient to move the gripper to the designated cup in the correct orientation and then away from the cup arrangement. However, without the real robot we were unable to complete the gripping action. We are confident however that with the real robot the gripping method selected would be successful.

References

[1] ”Beer Me, Robot.” Willow Garage. N.p., Web. 05 Apr. 2013. http://www.willowgarage.com/blog/2010/07/06/beer-me-robot

[2] ”Feature Detection.” Feature Detection OpenCV 2.4.4.0 Documentation. N.p., n.d. Web. 05 Apr. 2013. http://opencv.willowgarage.com/documentation/cpp/imgproc_feature_detection.html

[3] ”Wiki.” Tabletop_object_detector. N.p., n.d. Web. 15 May 2013. http://www.ros.org/wiki/tabletop_object_detector

[4] Anthony, Gregg, and Tshilidzi. Image Classification Using SVMs: One-against-One Vs One-against-All. Tech. N.p.: n.p., n.d. Print.

[5] David, and Lerner. Support Vector Machine-based Image Classication for Genetic Syndrome Diagnosis. Tech. N.p.: n.p., n.d. Print.

[6] ”Solem’s Vision Blog.” Reading Gauges. N.p., n.d. Web. 15 May 2013. http://www.janeriksolem.net/2012/08/reading-gauges-detecting-lines-and.html

[7] ”Feature Detection.” Feature Detection Opencv V2.1 Documentation. Thorsten Joachims, n.d. Web. 15 May 2013. http://svmlight.joachims.org/

[8] ”Features2D + Homography to find a known object.” Web. 15 May 2013 http://docs.opencv.org/doc/tutorials/features2d/feature_homography/feature_homography.html#feature-homography