Robot Grasping

Hee-Tae Jung

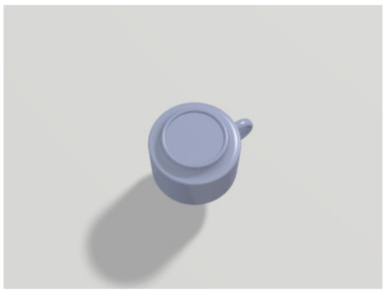
December 14, 2007

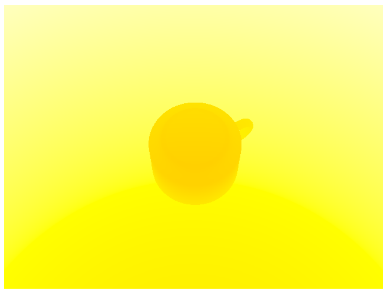
1. Introduction

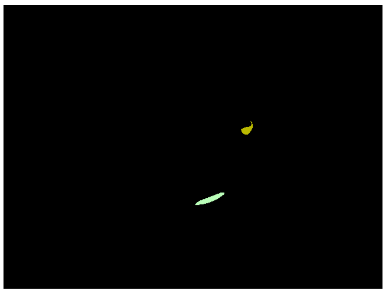
Robots have been widely used in various industries to perform complicated tasks which require fast and accurate performance; such as assembling the parts of vehicles and welding them together. However, those tasks can be done only when the robots are hard-coded to do so and even a very simple task such as grasping a newly seen object is still a great challenge to a robot. Various approaches have been made and most of the previous works have been heavily relying on the 3-d model of an object to successfully grasp it. However, building the 3-d model of a newly seen object itself is a very challenging task. To address this difficulty, [1] proposes the learning algorithm that does not require building the 3-d model of an object. Rather, they utilize the features extracted from 2-d images to predict possible grasping points of the object. They demonstrate that a robot can successfully grasp the object that it never saw before, but use only the image data of an object leading to limited performance. Hereby, I'm proposing to use the depth map of an object in addition to the image to improve performance.

2. Training

In this project, I use logistic regression and synthetically generated data to train the algorithm[1]. This set of data includes the synthetically generated images and corresponding depth maps and correct labels; which are the ideal grasping points. Since the set of test data is in 144-by-176 resolution, the synthetic data are resized down to the same resolution and divided into 4-by-4 patches. Hence, I end up having (144 / 4) * (176 / 4) = 1584 patches from each image file. I have 11000 pairs of object images, and the size of training data becomes too large when I utilize all those patches for training. Also, each set of label, the rightmost image of the three below, has only a handful of positive labels; non-black represents positive labels and black represents negative labels.


<image>


<depth map>


<label>

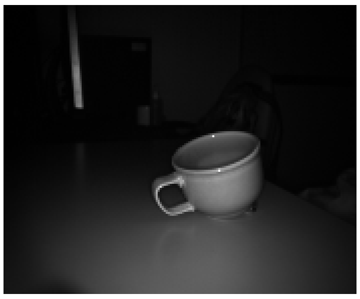
[1] http://ai.stanford.edu/~asaxena/learninggrasp/data.html

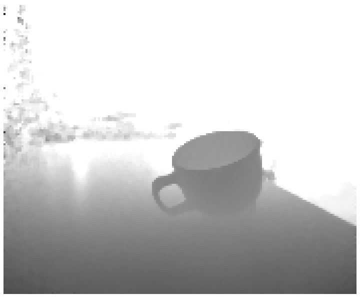
Thus, I use all the positive labels, the same number of randomly chosen negative labels and the corresponding features to reduce the size of training data and to balance the number of positive labels and the number of negative labels.

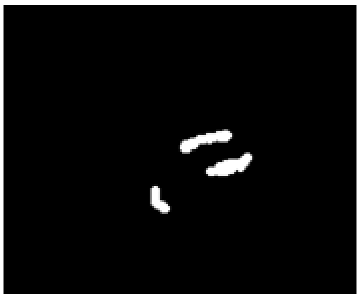
I come up with four base feature sets. Each image file is divided into 4-by-4 pixel patches, and the first set consists of these 16 pixel values. The second set consists of the edge, texture and color features extracted from each patch. When extracting the features I use the same filters that Ashutosh used in [1]. In depth map information, two same shapes in different distances end up being considered to be different due to their different offsets. Hence, the 16 depth values of each patch are transformed by using $\mathrm{Z} - \min(\min(\mathrm{Z}))$[2], where Z is a 4-by-4 patch matrix. This becomes my third set. The fourth set consists of the edge, texture and color features extracted from each depth map patch by using the same filters; applying the same filters to depth maps is somewhat naïve approach. I try different combinations of these four feature sets, and the results are given in the next section.

3. Testing

I test on two different sets of data; the first being 1% held-out synthetic data, and the second being the real data taken by SwissRanger[3]. Intensity information is used instead of image information because fully calibrated images from a stereo camera, BumbleBee2, are not available at this time. Before using intensity and depth map data, I changed the scale of intensity and depth map data to make them consistent with that of synthetic data by using $\frac{256*(\mathrm{Z}-\min(\min(\mathrm{Z})))}{(\max(max(Z))-\min(\min(Z)))}$, where Z is a 4-by-4 patch matrix.


<intensity>


<depth map>


<label>

Features are extracted in the same manner as before. Ideal grasping points are labeled by hands. Among the different combinations of the feature sets I tried, the following five combinations are

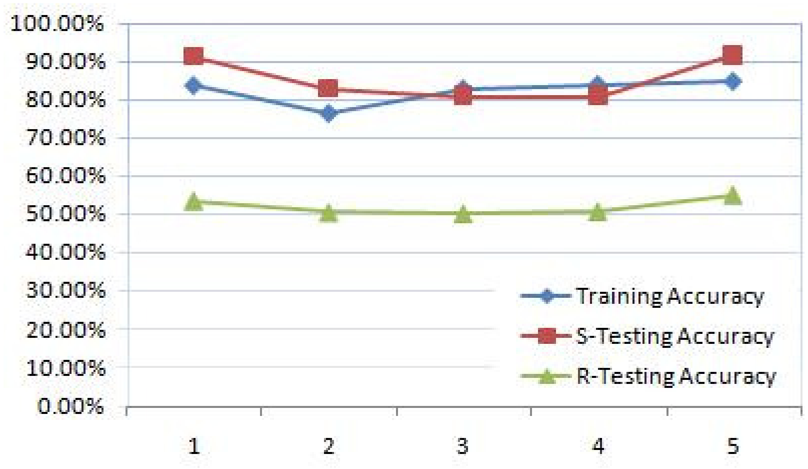
[2] I tried three different options: 1) not using transformation, 2) applying $\mathrm{Z} - \min(\min(\mathrm{Z}))$ and finally 3) using $\frac{256*(\mathrm{Z}-\min(\min(\mathrm{Z})))}{(\max(max(Z))-\min(\min(Z)))}$. I performed training and testing with these three cases, and end up getting the best result when using $\mathrm{Z} - \min(\min(\mathrm{Z}))$, and it has been used for the rest of the project.

[3] http://www.mesa-imaging.ch/

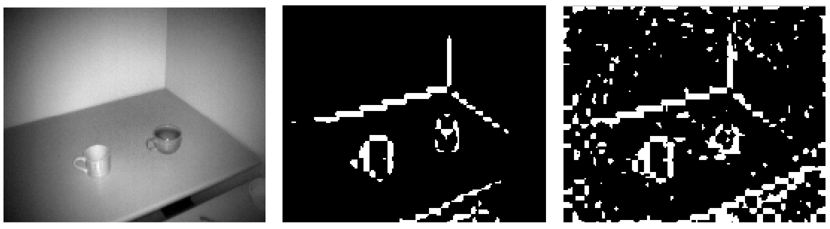
displayed below.

1. 1st feature set + 2nd feature set (only relying on images)
2. 3rd feature set + 4th feature set (only relying on depth maps)
3. 1st feature set + 3rd feature set
4. 1st feature set + 2nd feature set + 3rd feature set
5. 1st feature set + 2nd feature set + 3rd feature set + 4th feature set

The combination 1 and the combination 5 show relatively high accuracies as can be seen in the graph below.

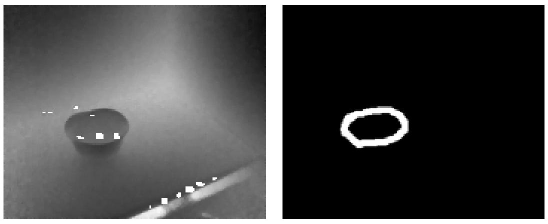


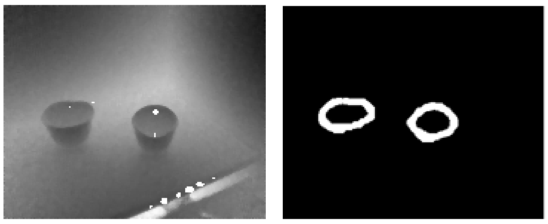
When tested on the 1% held-out synthetic data, the combination 1 gives about 87.92% true positive predictions[4] and the combination 5 gives 89.19%. However, when tested on the real data taken by SwissRanger, the combination 1 gives about 10.58%, while the combination 5 gives about 20.33%. At the first glance one may think the testing accuracies on the real data are way too low, but the results are quite reasonable considering that the real data are noisier than the synthetic data; the real images have a table and walls. The predictions from the combination 1 and the combination 5 are visualized as below.

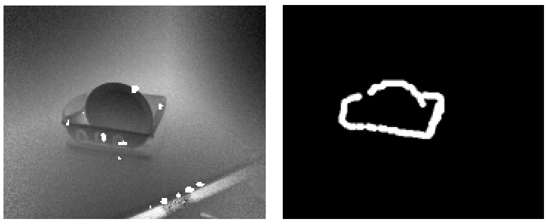

Two Mugs Combination 1 Combination 5

4 $\frac{\text{\# of true positive prediction}}{(\text{\# of true positive prediction} + \text{\# of false negative prediction})}$

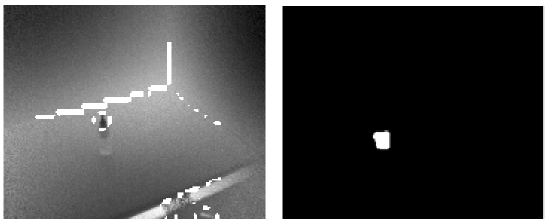
In general, the combination 1 gives clearer results visually but the predictions were not that accurate. On the other hand, the combination 5 gives too many false positive predictions, which is 9.85%; while the combination 1 gives only 3.26% of false positive predictions.
With these two results, I combined the two predictions to boost the accuracy. First, each positive prediction of the combination 1 propagates to its neighboring patch results; meaning, a patch becomes positive if its neighboring patches are predicted to be positive. Then the result from the combination 5 is combined. Different weights are tried, and higher true positive prediction rate can be achieved by giving more weights to the combination 5; all the way up to 28.71%. However, by giving the same weights to the combination 1 and the combination 5, more straightforward results can be achieved sacrificing the true positive prediction rate; which decreases down to only 11.81%. The sample images with the predicted grasping points and the corresponding labels are provided below; white dots on the objects are predicted candidate grasping points.


Bowl


Two Bowls



Plate


Cylindrical Object

4. Future Direction

This project can be further improved in several ways. First, intensity information should be replaced with real 2-d images so that more correct texture and color features from the images can be extracted. Second, smarter features should be used for depth maps to better represent the 3-d characteristics of an object. Third, real data can be used even for training. And the last, SVM can be used instead of logistic regression. During this project, SVM has been tried but it couldn't handle the same amount of training data that logistic regression did; I ended up having either 'out of memory' error or 'over parameterized' warning. Hence, the first three should be resolved before trying the last one.

5. Acknowledgment

I would like to thank Ashutosh Saxena and Andrew Y. Ng. for the helpful discussion throughout the project.

6. Reference

[1] Robotic Grasping of Novel Objects using Vision, Ashutosh Saxena, Justin Driemeyer, Andrew Y. Ng. *International Journal of Robotics Research (IJRR)*, 2007.