Robust Grasp Prediction on Baxter

Richard Doan, CJ Geering, Andrew Reardon, and Jeff Mahler
Email: {rdoan, cjgeering, areardon11, jmahler}@berkeley.edu
University of California, Berkeley
Berkeley, CA 94720

***Abstract*—Combining the techniques and theory learned in robotics and machine learning, this project attempts to score the predicted quality of grasps. By scoring the predictions, the hope is to better understand how labels generated in simulation perform on real world objects.**

I. Introduction

The goal of this project is to test the performance of a model that was trained using simulated data on real world objects.

A. Motivation

The motivation for this project is to understand how well simulation data can predict grasp quality given noisy point cloud data. By pursuing this motivation, we combine topics such as perception, robotic grasping, and machine learning.

B. Approach

The approach taken was to first analyze which components could be tackled independently. We noticed that the learning portion could be done completely on a local machine, so there was no need to utilize the Baxter machine. By noticing this, we could get the perception and actuation set up in parallel with the machine learning model generation. This proved to be key in finishing the project on time. From there, we scoured the Internet for documentation on the Xbox Kinects and ROS packages that would prove to be useful.

II. Related Works

Bootstrapping simulated data for real world grasps is also the focus of "Leveraging Big Data for Grasp Planning" [2]. Their approach involves crowd-sourcing to gather the labels and assumes that humans are optimal at choosing grasps. While [2] uses humans to label generated data, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours" [5] uses Baxter to label its own data. The data they generate is from the real world as opposed to computer generated objects. In doing so, [5] created a massive database of real-world objects and grasps.

In a way, this paper seems to extend "Using Geometry to Detect Grasps in 3D Point Clouds" [4], since we use two Xbox Kinects to only use the geometry of the object as a feature. [4] uses a combination of computer vision techniques and geometry to plan grasps. However, what our paper ultimately tests is how well a model trained using Dex-Net [3] chooses grasps. [3] provides a massive database of object models which can be used as a basis for learning how to plan optimal and robust grasps. Using [3] and, "Planning Optimal Grasps" [1], this paper demonstrates a way to find candidate grasps and rank them based off the metric devised in [1].

III. Methods

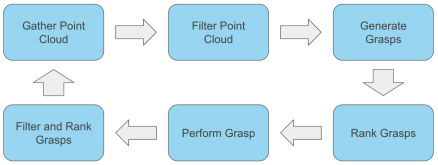

Fig. 1. The experimental cycle discussed below.

A. High-Level Summary

In order to test how well simulated data can be used to predict real world grasps, we must first create a representation of a real world object. To create the representation, we use two Xbox Kinects, $X1$ and $X2$, placed on opposite sides of an object. We then merge the point clouds by appending the list of points generated by $X2$ to the list of points generated by $X1$. After, we filter the point clouds to only include points within a certain region.

Once this filtering is done, what we are left with is a point cloud of our object. From here, we then sample 30 point pairs on our object, which act as the two contact points for a grasp. We then featurize the grasp so that we are able to feed it into our learned model.

For our learned model, we train two neural networks. We train one to classify whether or not a grasp is in force closure, and then we train the other neural network to regress on the Ferrari-Canny metric. This two step process helps us filter out grasps that will not work, and once we have a few candidate grasps, we then the Ferrari-Canny metric to rank the grasps and then perform the highest rank grasp.

After the grasp has been chosen, it is performed 10 times. From those repetitions, we compute the success rate of that particular grasp and see how well our model predicted our success rate.

B. Kinect Setup

Here are a few notes about kinect setup. Because of the high bandwidth outputted by a kinect, a single usb controller cannot handle processing two kinects at a time. Therefore, the

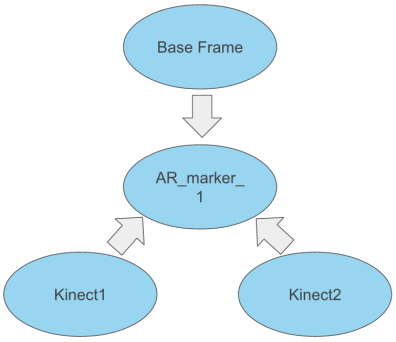

Fig. 2. The TF tree that is set up for gathering and merging point clouds.

solution is to kinect the two kinects into usb ports which use a different usb controller.

Once the setup is complete, we use *ar_track_alvar* to create a transform frame tree so that we know the location of the kinects in Baxter's base frame. Now the point cloud can be rendered and merged in Baxter's base frame.

C. Featurization

Here describes the way we featurize a grasp that is then used to predict the quality of that grasp.

We first randomly sample two points from our filtered point cloud. Given contact points c_1 and c_2, we create a 15×15 projection window centered around each contact and approximate the distances to the surface from the projection window.

In order to do the approximation, we project every point in our point cloud onto the plane p_1 defined by c_1 and $n_1 = c_2 - c_1$, and the plane p_2 defined by c_2 and $n_2 = c_1 - c_2$. After the projection is applied, we then toss out the points that are 5cm away from the center of the respective planes. We then discretize the plane into a 15×15 grid and then bin the points into the grid. Once the points are binned, we compute the residual of the projected points and assign that grid section the absolute lowest residual value. Once we have the grid of absolute minimum residuals, we call that our projection window and flatten it to use as our feature vector.

We also include the magnitude of the moment arms as a feature, which we append to the end of the feature vector generated by the projection window.

IV. Results

Our experiment consisted of trying out grasps on the following objects: a water bottle, tape roll, and full Pepto-Bismol bottle. Below are the success rates of executing a grasp 10 times on an object. It seems as though Baxter successfully picks up the given objects with a decent success rate with the bottles, but has a hard time picking up the roll of tape. The difficulty with the tape is probably due to the fact that the tape rolls away when Baxter contacts it.

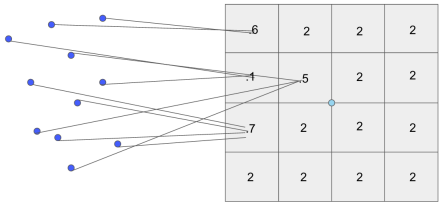

Fig. 3. An example of the projection window that we use for training. If no points are mapped to a grid cell, then it is automatically set to a relatively value. The light blue dot is a contact point the grid is centered at and the indigo points are the other points generated from the filtered point cloud.

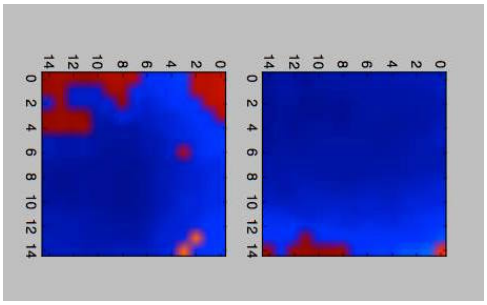

Fig. 4. Here's a picture of what the featurization looks like when plotted as a heat map.

TABLE I
TABLE OF RESULTS

Object	Success Rate
Water bottle 1	.80
Tape Roll 1	.60
Pepto-Bismol Bottle 1	.90

V. Conclusion

Given data from Dex-Net, we were able to determine which grasps would be successful using a trained neural network. For these particular objects, we are able to pick them up with pretty high success rates. However, that might be because the objects we tested were pretty easy for parallel jaws to pick up from many places.

A few things we would do differently are: using a chessboard instead of AR tags, trying out a different neural network architecture, and using the reflex hand to model an anthropomorphic grasp. Using a chessboard could eliminate the misalignment due to the low resolution of the AR tag, but then we could also use newer Kinects with a better resolution camera. In addition to different perception mechanisms, we could try a different neural network architecture to see what other results could arise. Lastly, using the Reflex hand could be interesting since we could test how adding that extra contact point stabilizes a grasp, making it more robust. Given our frustration with MoveIt, we might try to implement our own motion planner using Paden-Kahan.

ACKNOWLEDGMENT

The authors would like to thank Jeff Mahler for helping us through the more complex theory and math encompassed by our project. Not much could be done without his insight.

REFERENCES

[1] C. Ferrari and J. Canny., "Planning Optimal Grasps", *IEEE Int. Conf. Robotics and Automation (ICRA)*, 1992.

[2] D. Kappler et al., "Leveraging Big Data for Grasp Planning", *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015.

[3] J. Mahler et al., "Dex-Net 1.0: A Cloud-Based Network of 3D Objects for Robust Grasp Planning Using a Multi-Armed Bandit Model with Correlated Rewards", *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015.

[4] A. ten Pas and R. Platt, "Using Geometry to Detect Grasps in 3D Point Clouds", 2015.
everaging Big Data for Grasp Planning, Kappler et al., 2015

[5] L. Pinto and A. Gupta, "Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours", in *CoRR abs/1509.06825*, 2015.