Grasping with Intent to Place

CS 6780 Report

Stephen Moseson
(scm42@cornell.edu), MEng CS

Andrew Perrault
(arp86@cornell.edu), Senior CS

Abstract

We consider the problem of how to grasp an object given that we know its desired placement location. Instead of making path planning an intrinsic part of the grasp point evaluation algorithm, we attempt to evaluate the “goodness” of certain grasps by calculating metrics on the grasps and object point clouds themselves. This work aims to capture human intuitions about good grasps for certain placements and uses machine learning as a method of refining them.

Introduction

Grasping objects from stereo vision can be formulated as a supervised learning problem by learning the weights of 2D features on the disparity image. Using a maximum submatrix algorithm to find a rectangle that represents an oriented grasp point allows for an algorithm that runs in $O(n^3)$, which is quite fast in practice and effective if there is a good grasping point distinguishable from the perspective of the camera. If not, the algorithm could be extended by taking pictures from multiple views and comparing the scores of the rectangles.

However, from a personal robotics standpoint, there is more to consider when picking up an object than whether the robot could grip it at that point. Most of the time, the purpose of picking up an object is to put it down somewhere, often in a known terminal orientation. As humans, we naturally know how to do this without carefully planning the object’s movement. We consider features of the object -- how much clearance we will have, how stable the grasp is, whether the object will slip out of our hands when we are manipulating it, but this not very difficult in most cases. For example, for a household robot, it will be important when placing things on shelves for there to be a clear method for removing the hand after making the placement. One simple rule to guarantee this is to have the robot always grasp the object such that its hand is between the robot and the object in the final configuration. We propose a machine learning algorithm that captures the spirit of such rules.

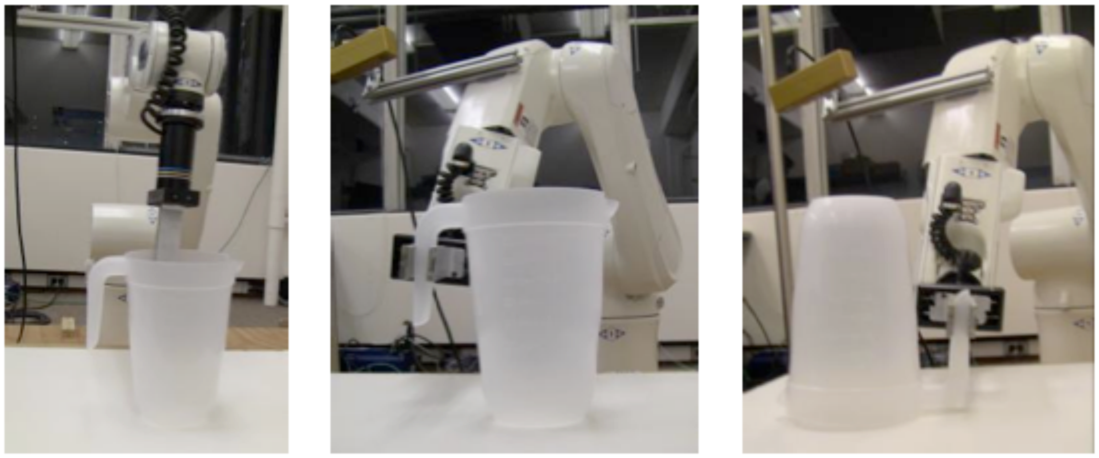

Figure 1: If the intent is to flip the cup over, the grasp on the left will result in failure due to the gripper colliding with the ground. A better grasp is shown in the middle picture where the transformed position (right) is also a good grasp.

Problem Motivation

One motivation is to improve current grasps by removing bad grasps. To do this we combine 3D point cloud data in both the initial and final position of the object to calculate our features. This differs from current grasping algorithms because it uses the point cloud of the object in the final orientation as well.

The same intuitions about grasping objects with limited information will be applied to tasks in personal robotics, to allow quick calculations to be done in cases where many alternatives are possible. As the grasping problem approaches a solution, placing will increase in importance.

Problem Specification

Given a stereo image, point cloud, rigid-body transformation from object initial to final position and a ranked list of grasps, re-rank the grasps using SVM-learned feature weights.

Approach

Machine Learning algorithm

Compute various features from the initial point cloud and terminal point cloud, calculated from the input rigid-body transformation. Use outputs of the grasping algorithm as training data, ranked Good, OK and Bad, and use SVM Rank to learn weights that favor good grasps.

Algorithm

We have chosen to use SVM-rank, implemented by Thorsten Joachims, as the learning algorithm. Our problem has a small search space as only 100 or so grasps are output by the front-end machine vision algorithm. The following features will be used when determining which grasp is best.

Features

One of the challenges of choosing features is to not overlap too much with the ones used in the original grasping algorithm. The role of the algorithm is not to second guess the chosen grasp points, but rather to evaluate them along a different metric. Since we are taking a reduced subset of points, we have time to evaluate more computationally intensive features, although to take too much time reduces the competitive advantage of the machine learning approach vs. motion planning.

1. Original score
Output of the original grasping algorithm

2. Obstacle proximity
Original and transformed
We would like to make predictions on a path existing between the initial and final configurations without explicitly calculating that path. One of the indicators that likely predicts this is the distance between the grasp point and any known obstacles in the initial and final configurations.

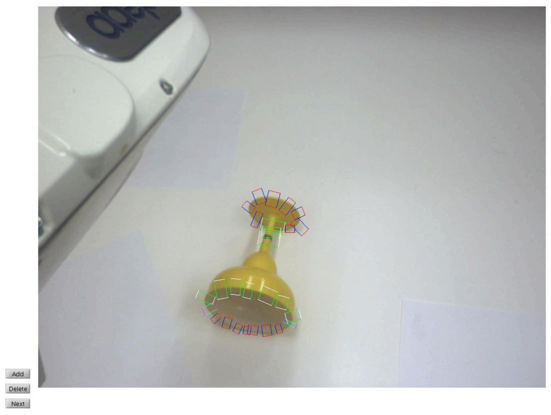

Figure 2: labeling data in MATLAB

3. Distance from object centroid (dc)
Since the first stage algorithm we are using does not use the point cloud, this algorithm will take into account the smaller amount of effort required to grasp an object near its center of mass. Another possible optimization at this stage would be concatenating multiple point clouds taken from different angles to have a more complete view of the object.

4. Object length
We derive the following features from the above: minimum obstacle distance (dmin), maximum obstacle distance (dmax), dc, dmin/length, dmax/length, dc/length. Evaluating these features takes around a second per pose-transformation pairing.

Transformation

Computing the features in the object's placing position requires transforming the original point cloud to this new position. We define an object transformation by separating it into a translation in 3 dimensions, and a rotation specified with the yaw, pitch, and roll the objects undergoes in the transformation. With these six values along with the object's original position, we compute a transformation matrix that transforms any point in the original coordinates to the objects transformed placing position. By multiplying each point in the original point cloud by this transformation we obtain a point cloud of the object in the desired placing position.

Object positions can be defined arbitrarily, but the point cloud rotation will occur about this point. Thus, we choose points in the center of objects so as to simplify the transformations.

After a grasp is determined the placing orientation of the arm must be calculated. This is done with the same transformation matrix used to transform the point cloud. By multiplying the single grasping point by the matrix we obtain the arm's placing position in 3D space. Computing the yaw pitch and roll for the arm in this placing position is done by post-multiplying the original grasping rotation matrix with the transformation matrix the object undergoes. The arm's placing yaw pitch and roll is then computed from this matrix.

Data

Training

Training was created by manually labeling grasps as either bad, ok or good. We trained on two objects in three different poses each with two to five transformations per pose for a total of 3,200 training rectangles. We measured precision by counting what percentage of the top, top three and top five grasps, after re-ranking, were labeled with good in the training data. Using 50-50 cross validation gave our algorithm a score of 90% on all three measures. It is difficult to expand the set of training objects because the objects must have several stable orientations and also more than one grasping position. Rod-like objects satisfy neither of these criteria. Plates have two stable orientations, but they can only be grasped one way.

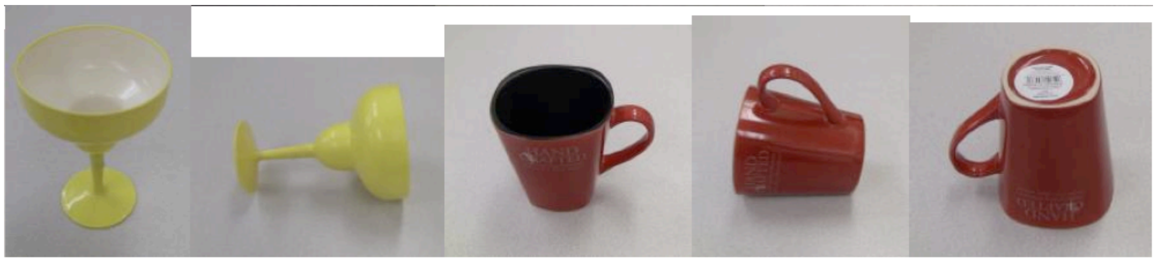

Figure 3: Some poses and transformations for the objects we used to train on.

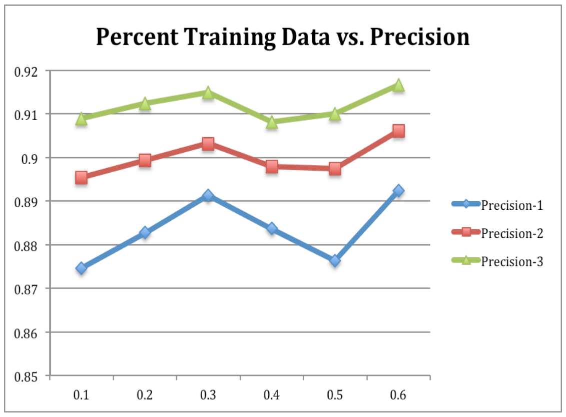

Figure 4: Training curve for our dataset, with 10%-60% used for training. Most of the information in our dataset is contained in less than 10% of the examples. Figure 5 shows the results of omitting certain features. Omitting any single feature had minimal effect, but by omitting the two features that make use of the minimum obstacle distance, precision is reduced to 50%.

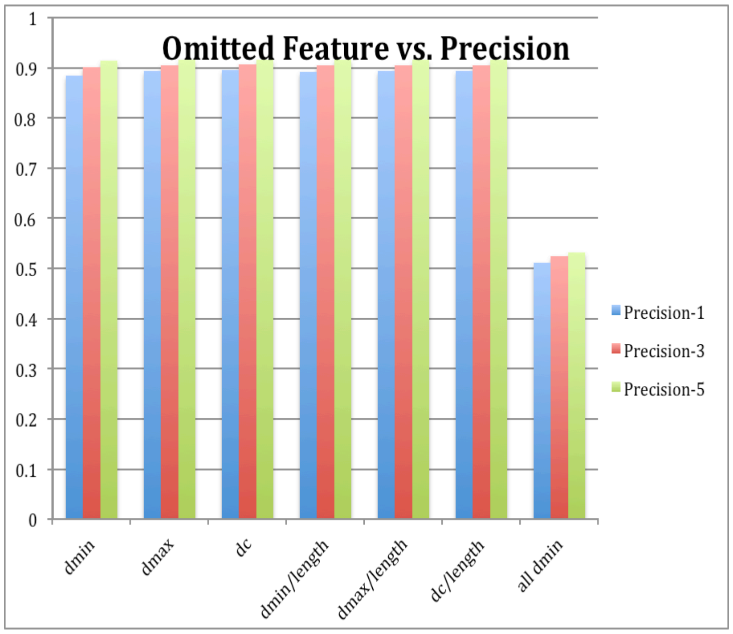

Figure 5: effects of omitting features on precision

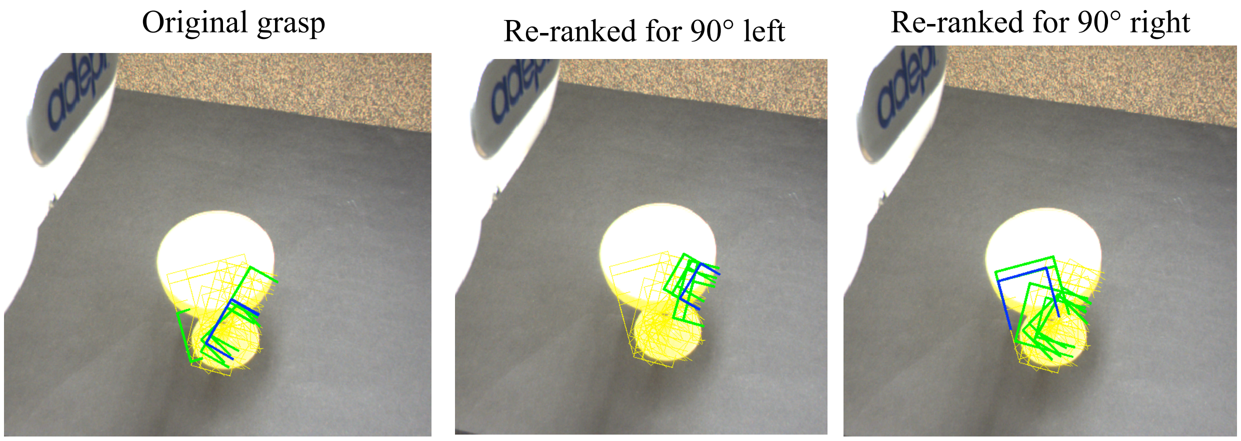

Figure 6: placing a martini glass (from left): a) original grasps b) putting down to the left c) putting down to the right. The top grasp is blue, the next 10 grasps are green, and the remaining 100 grasps are yellow.

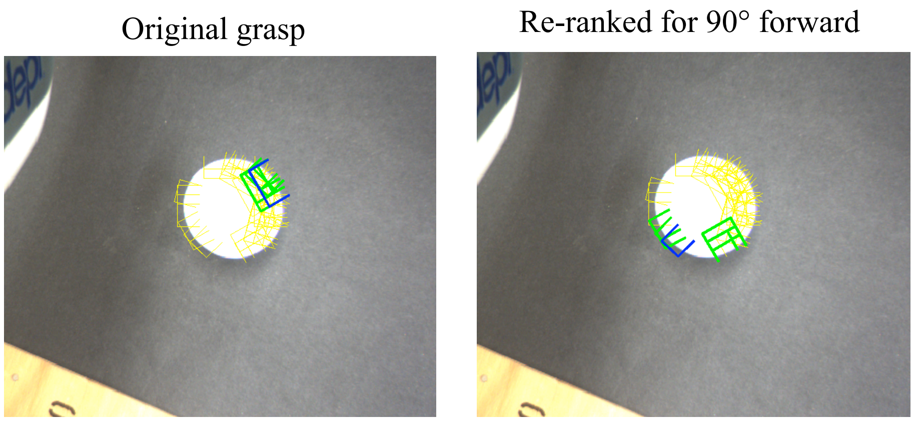

Figure 7: When viewing a Styrofoam cup from above there the first stage algorithm returns many good grasps all along the top rim. When given a transformation that turns the cup forward onto its top side, our algorithm chooses the grasps that are as far away from this side as possible, and chooses grasps on the bottom rim of the cup as seen in the picture.

Experiments and Results

We performed testing on the Adept Viper arm, using Marcus Lim's 3D grasping algorithm as our first stage algorithm. We tested the robot on a martini glass, a mug and a cup without a handle. We tested 2-5 transformations per object and evaluated our chosen grasp at five stages. We checked 1) if the grasp was spatially well-positioned, i.e. it wasn't to close to an obstacle in the initial or final configuration, 2) if it could be grasped by a human, 3) if we would have labeled it as good, 4) if the robot could grasp the object and 5) if the robot could place the object.

Criteria	***Score***
Spatially well-positioned	100%
Human graspable	82%
Would be manually labeled as good or ok	82%
Robot grasping success	82%
Robot placing success (if grasp was successful)	72%

We could improve the grasping part of the algorithm by taking into account the score given to the rectangles by the first stage algorithm. Assuming that a higher score correlates to more grasp-ability, taking this score into account would increase our grasping success rate. To improve placing, we need to estimate the shape of the object and the obstacles in the scene and use motion planning to plan a placing path. We also need to take into account how the configuration of an object changes when it is picked up.

References

Quoc V. Le, David Kamm, Arda F. Kara, Andrew Y. Ng, "Learning to grasp objects with multiple contact points", in *ICRA*, 2010.

Yun Jiang, Stephen Moseson and Ashutosh Saxena, "Learning to Grasp: What Should We Actually Learn?", submitted *ICRA*, 2011.