Grasping Objects Using High-Density, High-Accuracy Point Cloud Data

Alex Krimkevich
alexkrim@stanford.edu
December 12, 2008

***Abstract*— Grasping is an important manipulator function that forms the basis of a robot's interaction with its environment. Interesting algorithms cast the question as a learning problem [1], [2]. However, these efforts have been somewhat hamstrung by the lack of reliable and detailed depth data. The new Borg laser scanner for the STAIR platform remedies this problem, providing dense, highly accurate depth data by active triangluation. Given this data, we seek to discover whether local point cloud geometry suffices to determine the optimal grasp position for a robotic gripper, where we assume a priori knowledge of the actual grasping points. We present algorithms trained using regression-based methods on planar and ellipsoid approximations of the data. We find that, given some assumptions about our features, we can predict quite successfully the best arm configuration when grasping an object.**

I. INTRODUCTION

We present brief background information on the robotic manipulator, training platform, and grasping point classifier.

A. Robot and Training Platform

The STAIR1 platform is built around the Katana arm, a 5 degree-of-freedom robotic manipulator. Motion planning for the arm is accomplished using the Motion Planning Kit, or MPK. MPK combines Adaptive-SBL, a probabilistic roadmap planner, and an Open Inventor-based renderer to compute and visualize collision-free paths in configuration space for arbitrary robots. As a preliminary step to the research contained in this paper, we made several bug fixes and modifications to MPK. Most importantly, we gave the visualization machinery the ability to extract relevant configuration information about the arm. We also added the ability to overlay features and labels on a scene. Together, these allowed us to perform training completely in simulation. Finally, we created an improved model of the STAIR1 platform and obtained an accurate calibration to transform Borg laser scans into MPK scenes.

B. Grasping Point Classifier

The current grasping point classifier is a linear SVM that uses 405 exclusively depth-based features. The training algorithm first obtains a 640 x 480 pixel depth map from the Borg scanner. It divides the map into 6 x 6 pixel regions and convolves each with Laws' texture masks. These allow extraction of edge features, which are combined with similarly obtained features from neighboring patches to better understand local shape information. This is paired with an appropriate label for the patch in order to generate a training example. In order to execute a grasp given a test depth map of a scene, features are extracted from each 6 x 6 patch, as described above. Each patch and its features are then passed to the classifier, and several of the top patches are chosen as candidate grasp points. See Fig. 1.

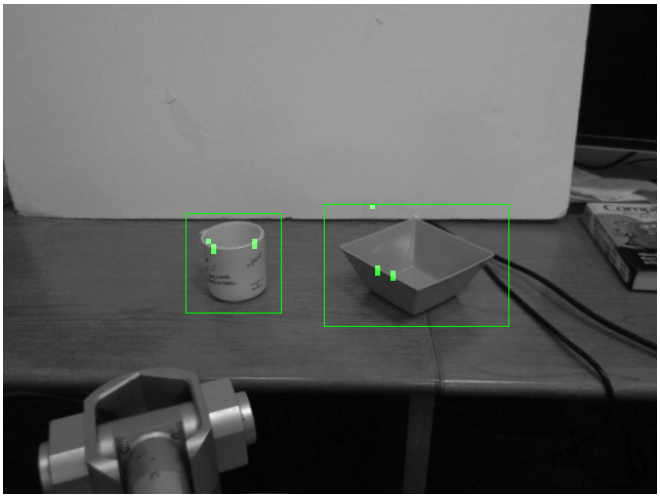

Fig. 1. Grasp points (in green) in sample scene.

C. Paper Structure

We first discuss the intuition and basic approach to the problem. Next, we go into detail about the methods used. Finally, we discuss our results, conclusion, and future work.

II. APPROACH

Given a grasp point, we define its neighborhood to be its k nearest neighbors. This definition is rather limited–consider, for example, implications in a cluttered environment where different objects are very near to each other. However, in simple scenes, this serves as a good heuristic for identifying regions of a surface containing our grasp point.

A. Local Plane Approximation

Once we have determined the neighborhood of a grasp point, we can find a local plane approximation to the surface in that neighborhood. To do this, we use principal component analysis [3]. For grasping point p_i, denote its neighborhood $Nbhd(p_i)$, and let μ_i be the centroid of $Nbhd(p_i)$. Then the covariance matrix is given as follows:

$$Cov(Nbhd(p_i)) = \sum_{x \in Nbhd(p_i)} (x - \mu_i)(x - \mu_i)^T$$

Note that because the points in $Nbhd(p_i)$ already have an appropriate scale (determined by their spatial layout), we do not rescale the points to have unit variance.

The eigenvectors of $Cov(Nbhd(p_i))$ sorted in order of decreasing eigenvalue yield the principal components of the points. In particular, the eigenvector associated with the smallest eigenvalue is the normal to the least-squares plane approximation of $Nbhd(p_i)$. The Katana manipulator has a parallel-plate gripper that works best for grasping narrow objects. Therefore, it seems quite natural that for objects with local planar geometry we would want our gripper to be parallel to this normal vector. Similarly, to enable this parallel grasp we must approach the object perpendicular to the normal vector. This suggests that the first and second principal components influence the angle of approach to the object.

B. Local Ellipsoid Approximation

For block-shaped or curved objects, the local plane approximation may not properly capture the true local geometry of the neighborhood of our grasping point. An ellipsoid might be a better fit in these kinds of situations. To find this ellipsoid, we construct the covariance matrix as above and find its eigenvectors. These eigenvectors serve as the principal axes of the ellipsoid. The lengths of the axes are given by the corresponding eigenvalues. We use a parametric representation of the ellipsoid that is helpful in generating features:

$$\begin{aligned} v_1^{(i)} &= \lambda_1^{(i)} cos(\theta) cos(\phi) \\ v_2^{(i)} &= \lambda_2^{(i)} cos(\theta) sin(\phi) \\ v_3^{(i)} &= \lambda_3^{(i)} sin(\phi) \end{aligned}$$

for $\theta \in [0, 2\pi)$ $\phi \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Here, $\lambda_1^{(i)} \geq \lambda_2^{(i)} \geq \lambda_3^{(i)}$ are our eigenvalues found above. Worth noting is that the resulting vector $[v_1^{(i)}\ v_2^{(i)}\ v_3^{(i)}]^T$ is in the basis defined by the eigenvectors, so we perform a change of basis to standard coordinates. Also worth noting is that the parameterization is not one-to-one at the poles, so these points should be treated separately.

III. SUPERVISED LEARNING

A. Labels

To perform supervised learning, the label $y^{(i)}$ for a single training example consists of the vector in R^3 between the fingertips when the arm is in the optimal grasping position about a grasping point. The Katana wrist is capable of full rotation, so $y^{(i)}$ is equivalent to -$y^{(i)}$ for our purposes. This label is often sufficient to fully describe the configuration of the Katana arm. Specifically, let P_i be the vertical plane containing the origin and the grasp point p_i. Then provided that the projection of $y^{(i)}$ onto P_i is non-zero, the label will have a unique corresponding approach position: the normal to the label lying in P_i. Knowing the wrist and approach angles, we can easily perform inverse kinematics to find an appropriate position for the arm.

B. Features

In our experiments, we used two sets of features. We now describe both.

1) Planar Features: When considering the planar approximation of a surface, we used the three principal components as features. The motivation for this is that the Katana arm performs best when grasping narrow objects, so we want our prediction to be as near to parallel with the local plane normal as possible. Our desire for a parallel grip then dictates that we approach perpendicular to the plane normal, so ideally this approach angle is a linear combination of the first and second principal components.

However, there are a several more issues to consider. First, the workspace of the Katana arm limits the reachable configurations. Thus, for example, it is not possible to achieve a vertical grasp for objects near the boundary of the workspace, or approximately 0.6 m from the origin. Therefore, we would like our predictions to somehow reflect these sort of constraints. Also, the labels are obtained in a simulated environment where precision control of the gripper is very difficult. As a result, the labels contain a fair amount of noise, and we also want our predictions to filter human error in the training examples. Thus, there is a strong learning component to making our predictions using local planar features.

2) Ellipsoid Features: For the ellipsoid approximation, we obtained 26 normals to the local ellipse and used these as features. Using the Cartesian representation of an ellipse, we take the gradient to find an equation for the normal at arbitrary x, y, z:

$$\begin{aligned} f(x,y,z) &= \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1 \\ \nabla f(x,y,z) &= \langle \frac{2x}{a^2}, \frac{2y}{b^2}, \frac{2z}{c^2} \rangle \\ &= \langle \frac{2cos(\theta)cos(\phi)}{\lambda_1}, \frac{2cos(\theta)sin(\phi)}{\lambda_2}, \frac{2sin(\phi)}{\lambda_3} \rangle \end{aligned}$$

where the final equality used our parametrization shown previously. We take normals for θ on $[0, 2\pi)$ at $\frac{\pi}{4}$ intervals and for ϕ on $(-\frac{\pi}{2}, \frac{\pi}{2})$ at $\frac{\pi}{4}$ intervals as well. We include

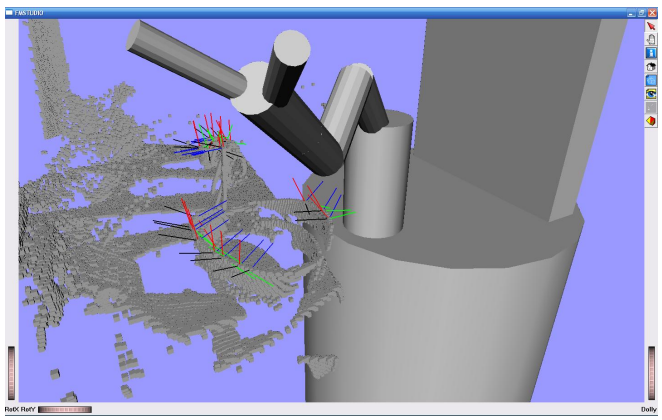

Fig. 2. Labels (in black) and features (red, green, and blue for strongest, middle, and weakest components, respectively) in a sample scene.

$[0\ 0\ -\lambda_1]^T$ and $[0\ 0\ \lambda_1]^T$ separately. These correspond to $\phi = -\frac{\pi}{2}$ and $\phi = \frac{\pi}{2}$.

Our hope in using the ellipse features is that we can obtain a finer, more granular representation of curved surfaces, like mugs. We may also be able to better account for workspace constraints and training noise as described in our discussion of planar features.

C. Training

We tried linear regression and locally weighted linear regression for each set of features used. We trained our algorithms on approximately 80 grasps on cups, plates, and blocks at a variety of distances from the base of the robotic arm. We discuss our methods in detail.

1) Linear Regression: Let $X^{(i)} \in R^{3\times n}$ be our i^{th} training example, where n is the number of features, and $\theta \in R^n$ be our parameter vector. For linear regression, we seek to minimize the following cost function:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{m}(X^{(i)}\theta - y^{(i)})^T(X^{(i)}\theta - y^{(i)})$$

Given a training set of local planar examples, we can write a closed form expression for θ similar to the normal equations.

$$\theta = (\sum_{i=1}^{m}(X^{(i)})^T X^{(i)})^{-1}(\sum_{i=1}^{m}(X^{(i)})^T y^{(i)})$$

For a training set of local ellipsoid examples, the matrix $\sum_{i=1}^{m}(X^{(i)})^T X^{(i)}$ is generally singular, so we cannot use the equation above. Instead, we find the gradient of the cost function and perform batch gradient descent.

$$\nabla J(\theta) = \sum_{i=1}^{m}(X^{(i)})^T X^{(i)}\theta - (X^{(i)})^T y^{(i)}$$

$$\frac{\partial}{\partial\theta_j}J(\theta) = \sum_{i=1}^{m}(X_j^{(i)})^T(X^{(i)}\theta - y^{(i)})$$

$$\theta_j := \theta_j - \alpha * \sum_{i=1}^{m}(X_j^{(i)})^T(X^{(i)}\theta - y^{(i)})$$

We found that α = 0.00001 works fairly well for this problem. Empirically, the ellipsoids tend to have one very large axis, and for larger values of α gradient descent actually diverges.

2) Locally Weighted Linear Regression: Initially, we used the following standard weights:

$$w^{(i)} = exp(\frac{\|p_i - p_{query}\|^2}{2\tau^2})$$

with τ = 0.8.

However, two considerations motivated us to try another set of weights. First, we did not have a very large dataset and were reluctant to throw large parts of it away. Second, we noted that we could take advantage of the workspace constraints of the arm when setting the weights. Beyond a certain fundamental distance of about 0.35 m, it is impossible for the arm to perform a vertical grasp. Conversely, within 0.35 m, the arm cannot perform horizontal grasps. Therefore, it seems that grasps on either side of that boundary should be similar to each other. This leads us to try the following weights:

$$w^{(i)} = \frac{1}{1 + exp(k \times sign(0.35 - \|p_{query}\|)(\|p_i\| - 0.35))}$$

These favor points on the same side of the 0.35 m boundary. k is a parameter that controls how quickly training points on the opposite side of the 0.35 m boundary decay. Results for both weights are shown below.

D. Modified PCA

In order to get good performance from the algorithm, it was necessary to do slight post-processing of the principal components. The reason for this is that the principal component directions are somewhat arbitrary. From the point of view of PCA, there is no difference between the positive and the negative directions. Unfortunately, this also means that there is no consistency to directions for points from different surfaces. This means that training on one object may lead to poor results when testing on another. Therefore, we make the assumption that the predictiction can always be represented by a linear combination of the features having all nonnegative coefficients. Note that this only requires reversing principal components that form an angle with their label exceeding 90^o. This would give a consistent orientation to all of our principal compoents. To implement this assumption, we compared the raw principal components to their labels and changed their directions as needed. This was also done for training examples that were held out for testing, a decision that will be discussed later.

IV. RESULTS

20-fold cross validation was used to test our algorithm. Results follow for both standard and locally weighted linear regression, as well as for both types of weights and for modified PCA (see above). We use LR to denote linear regression and LWR to denote locally weighted linear regression.

TABLE I

RESULTS, ERROR IN DEGREES

	Planar	Ellipsoid
LR	52.8^o	56.7^o
LR (modified PCA)	16.3^o	43.9^o
LWR (modified PCA, standard weights)	15.7^o	6.1^o
LWR (modified PCA, special weights)	16.3^o	6.1^o

V. CONCLUSION

Results obtained without using modified PCA are fairly poor. The reason for this is the variability of principle component directions, and training with the larger number of ellipsoid features seems even more vulnerable to the problem. The issue is discussed in more detail above. Once we take the step of modifying PCA, though, results for the planar case improve considerably. Interestingly, though they do improve for the ellipsoid case, they still remain quite bad. This finding is difficult to interpret. It is possible that

the ellipsoid method is more sensitive to noise, which would also explain the difference in results between the planar and ellipsoid models when doing standard linear regression. More work is required here.

In the planar case, the improvement due to locally weighted linear regression was marginal. For the ellipsoid approximation, on the other hand, it resulted in a nearly three-fold improvement in accuracy over the planar. This suggests that the ellipsoid is very good at capturing geometric information for similar groups of training examples, so given a large, broad training set, it seems likely that it could be successfully used to predict optimal grasp angles for novel objects. We also note here that our special weights performed no better than the standard ones, which means our intuition for using them was not necessarily correct.

Finally, recall that the principal component directions are also modified for the test examples. This step seems somewhat suspect, but the improved results indicate that the assumption behind it is indeed valid. Therefore, the principal components, when correctly interpreted, do account for some of the structure of the problem.

VI. FUTURE WORK

In the future, we would like to remove the need to orient the principal components for training and test data. [3] presents a possible algorithm that could be adapted for this purpose. Essentially, one picks a normal vector as a reference direction and sets nearby normal vectors to point to the same direction. This approach does not require knowledge of the labels, so it could solve our problem if we extend it to the other two principal components. We would also like to perform real grasping experiments using our algorithm. This is a necessary step to make sure that our results hold outside of simulation. Finally, it would be interesting to try new sets of features to see what kind of results they give. There may be other geometric ideas that yield better predictions.

VII. ACKNOWLEDGMENTS

The author gratefully acknowledges the help of Quoc V. Le in providing calibrations between the Borg laser and MPK (along with patient guidance), and Hee-Tae Jung in explaining the mechanics of the grasping point classifier. Additionally, this project would not be possible without the CS229 professor and staff, so a great thank you to them as well.

REFERENCES

[1] L. Wong. Learning to Select a Good Grasp. *CS229 class project report*, 2007.

[2] A. Saxena, M. S. Chung, A. Ng. Robotic Grasping of Novel Objects using Vision. *International Journal of Robotics Research*, vol. 27, no.2, pp. 157-173, Feb. 2008.

[3] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *In Proceedings of the 19th Annual Conference on Computer Graphics and interactive Techniques*, pp. 71-78, 1992.