An SVM Learning Approach to Robotic Grasping

Raphael Pelossof, Andrew Miller, Peter Allen, Tony Jebara
Department of Computer Science, Columbia University, New York, New York, 10027
Email: rp2056@columbia.edu, {amiller, allen, jebara}@cs.columbia.edu

***Abstract*— Finding appropriate stable grasps for a hand (either robotic or human) on an arbitrary object has proved to be a challenging and difficult problem. The space of grasping parameters coupled with the degrees-of-freedom and geometry of the object to be grasped creates a high-dimensional, non- smooth manifold. Traditional search methods applied to this manifold are typically not powerful enough to find appropriate stable grasping solutions, let alone optimal grasps. We address this issue in this paper, which attempts to find optimal grasps of objects using a grasping simulator. Our unique approach to the problem involves a combination of numerical methods to recover parts of the grasp quality surface with any robotic hand, and contemporary machine learning methods to interpolate that surface, in order to find the optimal grasp.**

I. Introduction

The grasp planning problem is extremely difficult because of the number of degrees of freedom of a robotic hand. For example, the Barrett hand (a 3-fingered robotic hand) we use has 10 degrees of freedom, 6 for orientation in space, and 4 for finger manipulation. This number of DOF's creates a large search space of hand configurations. We note that humans have a much larger space, since the number of DOF's of the human hand far exceeds most robotic hands. However, humans are capable of learning grasps through experience. The goal of our research is to apply new machine learning techniques to this problem and effectively have robots learn how to grasp arbitrary objects.

We are using supervised training to learn what is a good grasp for a robotic hand. This requires a method that allows us to try a large number of grasps of an object and report a metric on the quality of each grasp in the training set. Using this training set, we can then generate basis functions that can effectively both predict the quality of an arbitrary new set of grasping parameters and also use these basis functions to find an optimal set of grasping parameters for an object.

To build up our training data, we are using a robotic grasping simulator called GraspIt! developed by Miller and Allen [1]. This simulator, described in section III, takes as input a robotic hand model, object description, and a set of grasping parameters and calculates the grasp quality for that grasp. The simulator is fast, contains full collision detection, and allows a choice of material properties for the robot hand and object which can affect grasp quality. By sub-sampling the grasping parameter space using this simulator, we can efficiently create large training sets with grasp metrics. The simulator also allows us to visualize each grasp as needed.

In previous work [2] GraspIt! was used to plan grasps for objects using a heuristic based approach that attempted to sub-sample a reduced parameter space and choose the best grasp for an object. While effective in that context, it is clear that this approach is limited. We believe a learning approach is needed to extend robotic grasping to arbitrary objects. We are encouraged by results in machine learning, and in particular Support Vector Machines (SVM's), which have shown great promise in difficult learning problems.

Using this method we can approximate the grasp quality measure for a new set of grasping parameters, select an optimal grasp from the space of grasping parameters of an object, and hopefully extend these results to different objects. These parameters correspond to the degrees of freedom of an actual hand, rather than the placement of point contacts on the object surface. It is also important to note that our method is not dependent on a single type of robotic hand or class of objects. It provides a robust system for testing different robotic hands, and analyzing the quality space that they span.

We briefly mention other related research. Several authors have used learning for visually guided grasping [3], [4], [5]. Wheeler et al. have developed a learning system for high level grasping [6], and Oztop and Arbib used Hebbian learning to grasp unknown objects [7]. Rezzoug and Groce [8] use multi-stage neural networks to learn grasping postures.

II. Object Modeling

First presented by Barr [9], the superquadric model can create a wide range of smooth objects, with a smooth transition between them. The need for a small fixed number of parameters to describe this family of objects makes it perfect for using with learning algorithms that expect a fixed length feature vector. Research in the past has produced results in superquadric recovery from images [10] and touch [11]. This will enable our system to connect to a vision system and model objects in the image as composite superquadrics.

Our initial results use only undeformed superellipsoids, however, in future work we intend to apply our method to superparaboloids and supertoroids with varying deformation and scaling. The 3D superellipsoid is the spherical product of two 2D curves [9].

The values ϵ_1 and ϵ_2 affect the roundness of the shape in the horizontal and vertical directions respectively, and their values range between 0 and 2. Because the system ultimately should be able to find grasps for any superellipsoid, it must be trained on a variety of different examples. In order to have a manageable number of grasps that must be evaluated when generating training data, we have chosen 9 representative models that span the space of superellipsoids by choosing ϵ_1

and ϵ_2 to be one of 0.3, 1.0, or 1.7. Because the collision detection system within GraspIt! can only handle polygonal models, we approximate each of the superellipsoids with a triangular mesh. To create a mesh, we sample $r(\eta, \omega)$ using a 25x25 matrix of evenly distributed values of η and ω, and triangulate the resulting points. The overall mesh is then scaled to fit within a 180mm cube, which would fit within a grasp of the hand we are using for our tests.

III. Generating Training Data

To learn how to grasp these superquadrics requires a method of generating example grasps with their associated quality. For this we made use of GraspIt!, an interactive simulation, planning, analysis, and visualization system for robotic grasping (see figure 1(a). It can import a wide variety of different hand and robot designs, and a world populated with objects, all of which can be manipulated with in a virtual 3D workspace. A custom collision detection and contact determination system prevents bodies from passing through each other and can find and mark contact locations. The grasp analysis system can evaluate grasps formed with the hand using a variety of different quality measures, and the results of this analysis can be visualized by showing the weak point of a grasp or presenting projections of the 6D grasp wrench space. A dynamics engine can compute contact and friction forces over time, and allows for the evaluation of user written robot control algorithms [12].

For our experiments we used a model of the Barrett hand, but this method could be applied to other hands as well. It is an eight-axis, three-fingered mechanical hand with each finger having two joints. One finger (often called the thumb) is stationary and the other two can spread synchronously up to 180 degrees about the palm. Although there are eight axes, the hand is controlled by four motors. Each of the three fingers has one actuated proximal link, and a coupled distal link that moves at a fixed rate with the proximal link. A novel clutch mechanism allows the distal link to continue to move if the proximal link's motion is obstructed (referred to as *breakaway*). An additional motor controls the synchronous spread of the two fingers about the palm.

After the Barrett hand and a polygonalized superquadric model have been loaded into the workspace, the grasp tester can read a file of grasp starting positions and test the quality of each one, recording the results. To perform a grasp, the hand is first placed at the starting position, and the fingers are spread to the designated angle. Next, the hand is moved along the grasp approach direction until it is prevented from moving further by a contact. Then the fingers are closed around the object until contacts or joint limits prevent further motion. If at least one finger is in contact with the object at this point, the grasp is evaluated as described below. Figure 1(a) shows an example grasp evaluated within the GraspIt! system.

A. Grasp Evaluation

The tester can be used with any form of grasp evaluation that results in a scalar value. Since our aim is to find stable grasps for pick and place operations, we are using a quality metric that determines the magnitude of the largest worst-case disturbance wrench that can be resisted by a grasp of unit strength. This measure has been proposed in several forms, but it is best described by Ferrari and Canny [13].

The process involves approximating the contact friction cones as a convex sum of a finite number of force vectors around the boundary of the cone, computing the associated object wrench for each force vector, and then finding the convex hull of this set of wrenches. If we assume that each of the contact cones has unit height, then the convex hull corresponds to the L_1 grasp wrench space described by Ferrari and Canny. This space represents the space of wrenches that can be applied by the grasp given that the sum total of the contact normal forces is one. If the origin is not contained within this space, the grasp does not have force-closure, meaning there exists some set of disturbance wrenches that cannot be resisted by the grasp. In this case the quality of the grasp is 0. Otherwise, the quality of the grasp is equal to the distance from the origin to the closest facet of the convex hull. The wrench in this direction is the most difficult for the grasp to apply. It is important to note that the amount of friction that can be supported by the contacts greatly affects this quality measure. GraspIt! allows each body to have an associated material type and determines the coefficient of friction for each contact based on a lookup table of material types. In our examples, the palm and inner links of the Barrett hand are plastic, the outer links are rubber, and the superquadrics are metal. The coefficient of friction between plastic and metal surfaces is defined as 0.2, and between rubber and metal it is 1.0.

B. Spanning the Set of Possible Grasps

Our supervised learning requires a set of grasps that can potentially span the space of grasp parameters. Choosing a good parameterization is important to effectively sub-sample the space, and to include both good and bad grasps. If all 10 hand parameters (4 for internal motors and 6 for the relative pose of the wrist) are allowed to vary freely, only a very small sub-space of these values would result in worthwhile grasps. Many of them would have the fingers penetrating the object or not able to touch it at all. We have identified a minimal set of 4 parameters that effectively span the set of possible grasps and provide a large enough ratio of good to bad grasps to make learning possible. First we assume the palm should always be parallel to the surface of the object, hence the hand approach vector used by the tester will always be normal to the object surface at each grasp starting position. Since the tester moves the hand along the approach vector until contact occurs, we do not need a parameter to specify the distance of the palm from the object surface. We also do not need to specify the finger joint angles, since the fingers will start in a fully open position and will be closed around the object by the tester. This leaves us with two parameters, η and ω, to specify the starting position of the palm, one parameter to specify the roll angle of the hand about the approach vector (also referred to as thumb

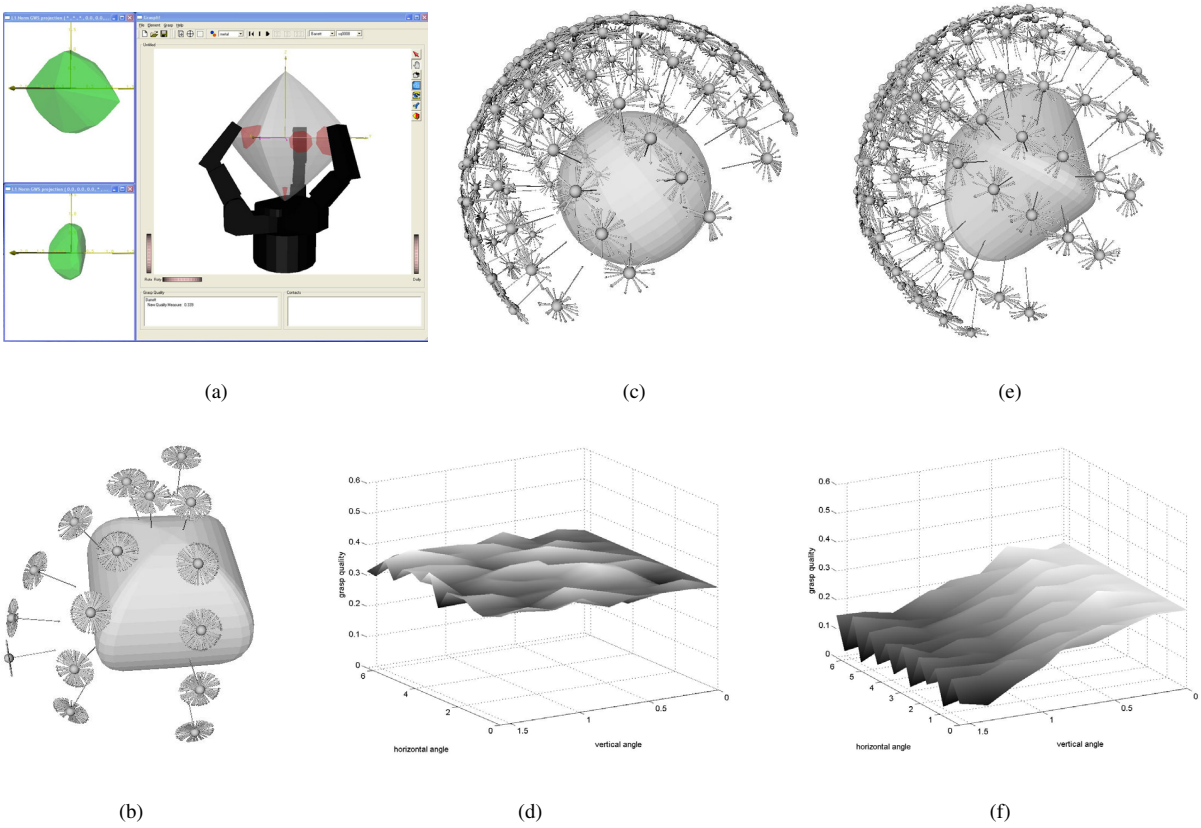


Fig. 1. (a) The GraspIt! simulator allows us to import a robot hand model (here a Barrett hand) and a object model (here a superellipsoid with $\epsilon_1 = \epsilon_2 = 1.8$) and evaluate a large number of grasps of the object. This image shows one successful grasp of this object. The grasp has a quality of 0.339, and the two windows on the left show projections of the grasp wrench space. The upper window shows the space of forces that can be applied to the object without a net torque, and the lower window shows the space of torques that can be applied without applying a net force. (b) For each superquadric in the training set we generate 1,600 grasp starting poses. These cover 16 positions over 1/8th of the total surface area, with 100 random combinations of different thumb orientations and finger spread angles. Here the long vector from each point denotes the grasp approach direction and the collection of short vectors shows the various thumb orientations. The spread angle is not shown. (c-d) Graphs of the average grasp quality of 25 grasps at each of 16x9 points over half of a polygonalized sphere ($\epsilon_1 = 1, \epsilon_2 = 1$), and (e-f) over a polygonalized rounded diamond shape ($\epsilon_1 = 1, \epsilon_2 = 0.2$).

orientation), and one parameter to specify the spread angle of the fingers. Because of the symmetric nature of superellipsoids we only have to vary η and ω between 0 and $\pi/2$, or 1/8th of the total surface. In our initial tests we are using a regular sampling of 3 values for η and 6 values for ω. This results in 16 unique starting positions since 3 positions coincide when $\omega = \pi/2$. To choose values for the remaining 2 parameters we employ a Monte-Carlo approach, and randomly choose roll angles between 0 and 360 degrees and spread angles between 0 and 90 degrees. Spread angles past 90 degrees rarely result in force closure grasps. Figure 1(b) shows an example superquadric (with $\epsilon_1 = 0.2$ and $\epsilon_2 = 1.8$) and the set of grasp starting positions that were generated. For each grasp starting position, the tester moves the hand, closes the fingers, evaluates the grasp, and records the quality.

As a test of our data generation, we created 2 sets of 3,600 grasps each for two superquadrics, one a sphere ($\epsilon_1 = 1, \epsilon_2 = 1$) and the other an rounded diamond ($\epsilon_1 = 1, \epsilon_2 = 0.2$). We sampled the grasp position parameters such that we had 16 values of η from 0 to 2π, and 9 values of ω from 0 to $\pi/2$, and at each position we chose 25 combinations of random hand roll and finger spread angles. To graph the results we averaged the quality of the 25 grasps at each position, and show this average over the 16x9 position samples. As expected the average quality of the grasps of the sphere was uniform, and the quality of the grasps of the rounded diamond was nearly uniform at the top where it resembles the sphere (see figure 1(c)). However, grasps near the equator of the diamond resulted in 8 roughly symmetrical peaks in quality due to the 4 symmetrical edges of the object.

IV. SVM Learning from Simulated Data

A. Feature Vectors and Training Data

Following the detailed explanation of the grasping system, we now discuss the machine learning algorithms used to build a regression mapping between object shape, grasp parameters and grasp quality. Once trained, this regression mapping can be used very efficiently to estimate the grasping parameters

ϵ_1	ϵ_2	U_x	U_y	U_z	$\cos\theta$	$\sin\theta$	$\cos\rho$	$\sin\rho$

Fig. 2. Input feature vector X

that obtain highest grasp quality for a new query set of superquadric shape parameters. Our learned regression mapping accepts a fixed length input vector that contains the shape and grasping parameters and returns a single scalar which estimates the grasp quality. If only provided with a subset of the input vector (i.e. the shape parameters), the regression algorithm will perform an efficient search for the optimal setting of the missing input vector values (i.e. the grasp parameters) that maximize grasp quality. While it is possible to consider input vectors that are not of fixed length (via more advanced kernel methods), in this setting we have only considered the simple vector-input scalar-output regression case. For effective learning, our input feature-vectors should have some clustering in the high dimensional space, or produce a smooth manifold.

The input feature vectors that we are currently dealing with are as follows. The superquadric shape parameters ϵ_1 and ϵ_2, a unit vector that points to the intersection between the superquadric and the surface normal, the hand roll angle θ, and the finger spread angle ρ. Figure 2 shows the entries in each of our nine-dimensional input feature vectors. We use Cartesian coordinates to represent the two angles to avoid the discontinuity in the spherical representation at the angles of 0 and 2π, which can confuse the SVM regression during learning. We generate input vectors $X_1, \ldots, X_T$ by sampling many shape and grasp poses. Each X_t of these input vectors is provided to GraspIt! for simulation and generates a single scalar quantity y_t, which is the grasp quality. Thus, our training data set is a set of T pairs of vector and scalar data points $(X_1, y_1), \ldots, (X_T, y_T)$. From this training dataset, we will learn a function $f(X)$ which will be used to estimate y using support vector machine regression.

B. SVM Regression Training

Support vector machines have recently become popular learning tools for performing classification [14], [15] and regression [16], [17]. Typically, SVM classification is used to build a function f(X) that predicts binary y values. SVM regression on the other hand generates functions whose outputs are scalars. However, unlike least-square or empirical methods, SVM regression maintains the same motivation as SVM classification: minimizing a bound on the expected error for future test data, inheriting interesting generalization properties and sparsity. Consider the simplest case of linear SVM regression, where $f(X)$ is given by the following:

$$f(X) = W^T X + b. \tag{1}$$

Here, the function involves taking the inner product of the input vector X with a model parameter W and adding the scalar bias b. The SVM regression optimization problem finds a linear function that predicts outputs almost equal to y with no more than epsilon error (above or below) while simultaneously minimizing the norm of W to encourage better generalization. This finds the lowest norm or flattest linear function that approximates the data within an epsilon-tube of sensitivity. In other words, we minimize the following constrained cost function:

$$\min_{W,b} \tfrac{1}{2} W^T W \; subject\, to \; \begin{cases} y_t - \langle W, X_t \rangle - b \leq \varepsilon \\ \langle W, X_t \rangle + b - y_t \leq \varepsilon \end{cases} \quad \forall t. \tag{2}$$

Since the constraints may not always be satisfied (i.e. if some y values have large noise or are outliers and can never fit the epsilon-tube), we introduce T total ξ slack variables and T total ξ^* slack variables (a pair of for each data point) which allow the program to permit some violations of the epsilon tube. We end up with following relaxed regression which accommodates the epsilon-sensitive tube while paying a linear loss for violations when the regression predicts values outside the tube:

$$\begin{aligned} &\min_{W,b,\vec{\xi},\vec{\xi}^*} \tfrac{1}{2} \|W\|^2 + C \sum\nolimits_t \xi_t + \xi_t^* \;\; , \\ &\forall t \begin{cases} y_t - \langle W, X_t \rangle - b \leq \varepsilon + \xi_t \\ \langle W, X_t \rangle + b - y_t \leq \varepsilon + \xi_t^* \\ \xi_t, \xi_t^* \geq 0 \end{cases} \end{aligned} \tag{3}$$

In the above, C is a scalar regularization parameter that penalizes the amount of slack used (smaller values of C yield more outlier rejection). This problem is now readily solvable via any general quadratic program (QP) framework (e.g. in Matlab) which recovers the $D = 9$ dimensional W vector, the scalar bias b as well as the $2T$ slack ξ and ξ^* variables. Instead, however, it is traditional to rewrite the above constrained optimization problem using Lagrange multipliers α and α^* (a pair of alphas for each data point) as follows:

$$\begin{aligned} L = \min_{W,\vec{\xi},\vec{\xi}^*} \; & \tfrac{1}{2} \|W\|^2 - \sum\nolimits_t \alpha_t^* \left(\varepsilon + \xi_t^* + y_t - \langle W, X_t \rangle - b\right) \\ & - \sum\nolimits_t \alpha_t \left(\varepsilon + \xi_t - y_t + \langle W, X_t \rangle + b\right) + C \sum\nolimits_t (\xi_t + \xi_t^*) \\ & - \sum\nolimits_t (\eta_t \xi_t + \eta_t^* \xi_t^*) \quad subject\, to \; \xi_t \geq 0 \;, \; \xi_t^* \geq 0 \; \forall t \end{aligned} \tag{4}$$

Our previous constraints are captured via the α and α^* Lagrange multipliers. Typically, however, we solve the *dual* version of the above *primal* problem. The dual maximization problem is more efficient to solve and readily accommodates non-linear regression:

$$\begin{aligned} D = \min_{\vec{\alpha},\vec{\alpha}^*} \; & -\tfrac{1}{2} \sum\nolimits_{t,t'} (\alpha - \alpha_t^*)(\alpha_{t'} - \alpha_{t'}^*) \langle X_t, X_{t'} \rangle \\ & -\epsilon \sum\nolimits_t (\alpha_t - \alpha_t^*) + \sum\nolimits_t y_t (\alpha_t - \alpha_t^*) \\ & subject\, to \;\; \sum\nolimits_t (\alpha_t - \alpha_t^*) = 0 \quad \alpha_t, alpha_t^* \in [0, c] \end{aligned} \tag{5}$$

The above can be solved using QP, recovering the $2T$ total α and α^* Lagrange multipliers. For efficiency, we used a method similar to [18]. Reconstructing the regression function from the above α and α^* Lagrange multipliers:

$$f(X) = \sum\nolimits_t (\alpha_t - \alpha_t^*) \langle X_t, X \rangle + b. \tag{6}$$

An interesting result is that only a few α and α^* Lagrange multipliers

will be non-zero, these correspond to the points in the regression that are on the boundary of the epsilon tube (called support vectors) as well as outlier points outside of the tube. Points inside the tube obtain zero α and α^* values. This sparsity yields good generalization to new testing conditions as well as efficient computation of the regression function $f(X)$. The bias value b is computed from the Karush-Kuhn Tucker conditions by noting that, for the support vectors, when α_t lies in $(0, C)$ we must have $f(X_t) = y_t - \epsilon$ and for support vectors when α_t^* lies in $(0, C)$ we must have $f(X_t) = y_t + \epsilon$. We can readily accommodate nonlinear regression by replacing *all* the inner product symbols in the dual formulation and in $f(X)$ by kernel evaluations as follows:

$$\langle X, X' \rangle = X^T X' \rightarrow k\left(X, X'\right), \tag{7}$$

where $k(X, X')$ is any function that satisfies Mercer's condition (i.e. positive-definiteness) and produces a scalar quantity output. In our experiments, we used the radial basis function (RBF) kernel which is known to handle various nonlinear problems well:

$$k\left(X, X'\right) = \exp\left(-\sigma \left\|X - X'\right\|^2\right). \tag{8}$$

We manually selected appropriate values of ϵ and σ and C during training to ensure good generalization performance on test data. In our experiments, values of $\epsilon = 0.02$ and $\sigma = 0.1$ and $C = 600$ performed reasonably well and provided an $f(X)$ function with good cross-validation accuracy. Ultimately, we have the following learned function involving RBFs:

$$f\left(X\right) = \sum\nolimits_t \left(\alpha_t - \alpha_t^*\right) \exp\left(-\sigma \left\|X_t - X\right\|^2\right) + b. \tag{9}$$

C. SVM Regression for Grasp Evaluation

Given our learned $f(X)$ function (from many object shapes and grasp pose vectors), we can now efficiently compute the grasp quality. However, we wish to predict good grasps from only shape parameters. This is also readily feasible with our regression function as well. We simply have only a sub-component of the X vector corresponding to the shape information while the rest of the vector is missing. Thus, we can estimate the remaining components of X by maximizing the function $f(X)$ to obtain the best possible grasp according to our SVM. In other words, consider splitting the input vector into two components $X = [X^a X^b]$, we find the best setting of X^b as follows:

$$X^b = \arg\max_{X^b} f\left(\left[X^a \ X^b\right]\right) \tag{10}$$

$$= \arg\max_{X^b} \sum\nolimits_t \left(\left(\alpha_t - \alpha_t^*\right) \exp\left(-\sigma \left\|X_t^a - X^a\right\|^2 - \sigma \left\|X_t^b - X^b\right\|^2\right)\right) \tag{11}$$

One particularly interesting aspect of the above equation is that it involves reweighting the RBFs in $f(X)$ by the affinity between the current X^a and the X_t^a support vectors in our training set which seem to act like prototypes of shape vectors.

For a given X^a containing just an object's shape parameters the maximization proceeds as follows. We first evaluate all X^b values that were in our support vectors. The highest scoring one of these is then used to seed a gradient ascent technique. We then merely update the current estimate of the X^b with gradient ascent on $f(X)$ as given by the following update rule:

$$X^b \leftarrow X^b + \delta \sum\nolimits_t \left(\left(\alpha_t - \alpha_t^*\right) \exp\left(-\sigma \left\|X_t^a - X^a\right\|^2 - \sigma \left\|X_t^b - X^b\right\|^2\right) \left[X_t^b - X^b\right]\right) \tag{12}$$

We use a small δ value and repeat until convergence. This results in an estimated grasp X^b for a novel object in less than 1 second on a regular Pentium IV machine providing a very fast initial guess for the robotic hand's grasp parameters even if the object shape subvector X^a is a novel configuration not seen in our training dataset.

V. Results

For each of the 9 superquadrics in our training set, we generated 1,600 grasps, which consisted of 100 random roll and spread angle combinations for each of the 16 grasp starting positions. This gave us a total of 14,400 grasps, which were evaluated by GraspIt! over the course of approximately 4 hours on a Pentium IV 2.3 GHz Windows machine. It is important to generate good training sets comprised of a large number of stable grasps. The fact that only slightly more than 3,000 of the total 14,440 grasps had zero quality (they were not force closure grasps), appears to validate our approach of uniformly sampling some parameters and randomly sampling others.

For each superquadric, that data was sorted, and all but the best 150 grasps were eliminated. Of these 150 (for each object), 135 were randomly chosen to be part of the training set for the SVM, and the remaining 15 were put in the test set. The results of the SVM training are shown in figures 3(a) and 3(b). The accuracy of the SVM regression on testing indicates that the learned $f(X)$ function can be used to rank the vectors of grasping parameters by their estimated grasp qualities. Although the regression does not always favor the simulated best grasp, it typically chooses grasps that still perform well.

To test the noise in the GraspIt! simulation (arising from polygonal shape approximations), we had GraspIt! perform 160 grasps with a fixed spread angle of 60 degrees at different locations on an approximated sphere. On a true sphere the quality would be uniform, but as the histogram in figure 3(c) shows, there is a Gaussian noise distribution with a standard deviation of 0.0192. Interestingly, the standard deviation of the error between the predicted quality and the simulated quality (during testing) is 0.0412. Thus the learned model has roughly twice the noise of full simulation.

We then evaluated the regression function for its ability to predict grasping parameters when queried with only shape parameters. The procedure in the previous section (see eq. 11) describes the arg maximization of quality to produce grasp parameters. In figures 4(a)- 4(c), we show predicted grasps from the SVM for both a shape previously seen in the

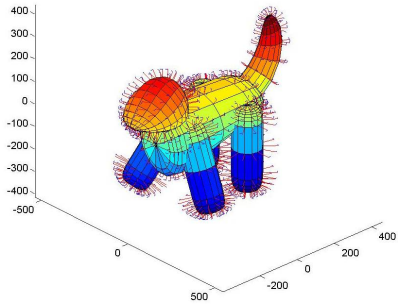


Fig. 5. Multiple deformed superquadrics can be combined to form more interesting shapes to grasp.

training data as well as a novel shapes. The SVM-predicted grasps appear qualitatively reasonable, and when simulated in GraspIt!, they subsequently yielded high grasp qualities.

VI. Discussion and Future Work

Our modeling and simulation method had opened a door to a variety of machine learning algorithms for grasp quality regression. It appears that the data does form a consistent mapping in high dimensional space, and therefore we were able to get good results. Nevertheless it's possible that with different kernels one could improve on our results.

In this paper, we have shown a machine learning approach to learning robotic grasps. Our results show promise in applying new machine learning techniques such as SVM's to learn optimal grasps of objects as well as transfer grasping knowledge to new kinds of objects. In our experiments, we were able to constrain the parameter space by choosing simple object models and also using the somewhat limited Barrett robotic hand. To extend these results to more complex hand designs and more complex objects, we are working on two fronts.

First, to reduce the dimensionality of the hand parameter space, we can make use of other sensors such as vision [19]. Our experiments are essentially "blind" in that there is no other sensing other than contacts of the fingers and hand. If we include simple vision processing, we can use knowledge about an object's silhouette or contour that can initially constrain the grasp parameter space, particularly with respect to approach vector of the object. This reduces the space of learned grasps that must be spanned by a learning system, and allows us to infer correct grasps. In a sense, we can create a coarse/fine grasping system in which vision creates an index into a smaller set of learned grasps that can be approximated by methods such as SVM's.

Second, our initial superquadric models are clearly not able to model the full range of possible objects to be grasped. However, these models can be combined into larger sets of composite models that are composed of multiple superquadric shapes. Figure 5 shows a complex object assembled from a number of superquadrics that include bending and tapering parameters.

One potential approach to solving this multi-superquadric grasping problem involves using the SVM regression $f(X)$ function for single superquadrics in a cascade. We compute an $F(X)$ for multi-superquadrics from re-instantiations of $f(X)$ that are appropriately rotated and translated given the pose of each superquadric. This maps a single $f(X)$ to multiple $f_i(X)$ functions. We can then devise a combination scheme which updates $F(X)$ which is initially set to zero by cascading multiple $f_i(X)$ functions. For instance, we may consider computing $F(X)$ as the minimum of all the qualities of the individual superquadrics: $F(X) = \min_i f_i(X)$. Alternatively, we may use SVM regression to learn a composition function, which iteratively assembles two superquadrics at a time to produce a multi-superquadric grasp quality regression as follows: $F(X) = g(f_i(X), f_j(X))$. Here $g()$ function has two-dimensional inputs and one dimensional output.

Another promising direction is to consider kernels on more general representations of 3D objects. For instance, a kernel function was recently proposed between two vectors-sets (as opposed to two vectors in our formulation) [20]. These vector-sets need not be of the same size and the original work demonstrates that these can be used to recognize point-clouds of various shapes such as digits. Such vector-sets are useful for representing 3D objects as a collection of 3D vertices which is a very general representation which may be more promising than superquadrics since it does not involve modularly piecing together multiple shapes. Rather, it simply involves computing a metric (or kernel affinity) between shape prototypes.

Of course, the results of this planning are not useful unless they can be applied to an actual manipulation system. We have shown that we can use a real-time model based vision system to register the poses of simulated objects in the environment with the poses of the actual objects they represent. Then grasps planned within GraspIt! can be carried out on the actual robot [21]. We would like to expand this vision system to automatically estimate object shape parameters using established techniques for fitting superquadrics [10].

Acknowledgment

This work was supported in part by an NSF-ITR award IIS-03-12693.

References

[1] A. Miller and P. Allen, "GraspIt!: A versatile simulator for grasping analysis," in *Proc. of the ASME Dynamic Systems and Control Division*, vol. 2, Orlando, FL, 2000, pp. 1251–1258.

[2] A. Miller, S. Knoop, H. Christensen, and P. Allen, "Automatic grasp planning using shape primitives," in *Proc. of the 2003 IEEE Intl. Conf. on Robotics and Automation*, 2003, pp. 1824–1829.

[3] B. Rossler, J. Zhang, and A. Knoll, "Visual guided grasping of aggregates using self-valuing learning," in *Proc. of 2002 IEEE Intl. Conf. on Robotics and Automation*, 2002, pp. 3912–3917.

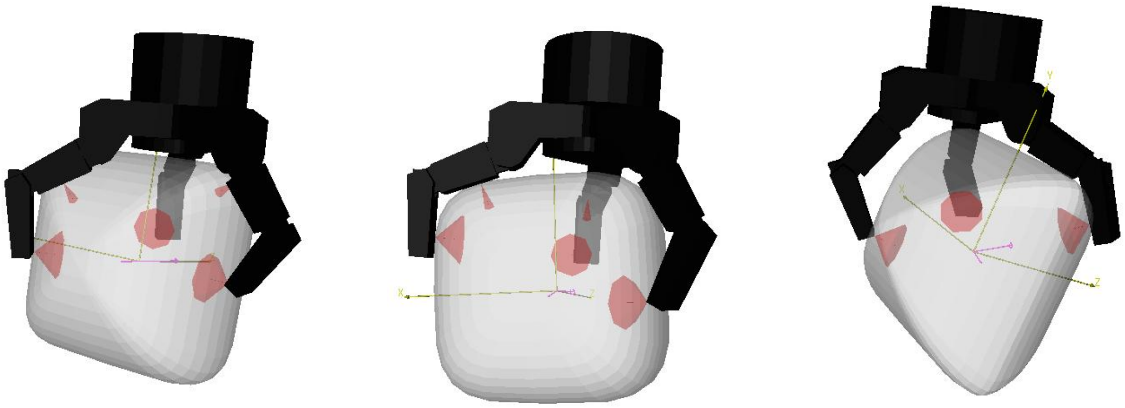
[4] J. Piater, "Learning visual features to recommend grasp configurations." Univeristy of Massachusetts Amherst, Dept. of Computer Science, Tech. Rep. 2000-40, 2000.

[5] I. Kamon, T. Flash, and S. Edelman, "Learning visually guided grasping: a test case in sensorimotor learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 3, pp. 266–276, May 1998.

[6] D. Wheeler, A. Fagg, and R. Grupen, "Learning prospective pick and place behavior," in *Proc. of the IEEE/RSJ Intl. Conf. on Development and Learning*, 2002.

(a) (b) (c)

Fig. 3. (Best viewed in color) (a) For each of the 9 tested superquadrics we chosen the 150 best grasps and trained the SVM on 90% of this data (135 randomly selected grasps for each superquadric), and the quality of these grasps is shown in blue. Overlaid on top of that is the expected grasp quality for each of the training grasps from the SVM regression. (b) The remaining 10% of our generated data is used for testing purposes. These are grasps that the SVM has been not trained on. (c) The distribution of grasp quality for fixed spread angle grasps of an approximated sphere demonstrates the noise inherent in the data generation due to polygonalization.



(a) (b) (c)

Fig. 4. (a) The optimal grasp of the SVM regression function for a superquadric shape previously seen in the training set ($\epsilon_1 = 1, \epsilon_2 = 0.3$). The quality of this grasp as tested in GraspIt! is 0.402. (b-c) The optimal grasp of the SVM regression function for novel superquadrics ($\epsilon_1 = 0.6, \epsilon_2 = 0.5$) and ($\epsilon_1 = 1.3, \epsilon_2 = 0.2$). The quality of these grasps as tested in GraspIt! is 0.315 and 0.102 respectively.

[7] E. Oztop and M. Arbib, "A biologically inspired learning to grasp system," in *Proc. of the 23rd Annual Intl. Conf. of the IEEE Engineering in Medicine and Biology Society*, 2001, pp. 857–860.

[8] N. Rezzoug and P. Gorce, "A multistage neural network architecture to learn hand grasping posture," in *Proc of the 2002 IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2002, pp. 1705–1710.

[9] A. Barr, "Superquadrics and angle-preserving transformations," *IEEE Computer Graphics and Applications*, vol. 1, no. 1, pp. 11–23, 1981.

[10] A. Pentland, "Recognition by parts," in *Proc. First Intl. Conf. on Computer Vision*, 1987, pp. 612–620.

[11] P. Allen and P. Michelman, "Acquisition and interpretation of 3-D sensor data from touch." *IEEE Transactions on Robotics and Automation*, pp. 397–404, August 1990.

[12] A. Miller and H. Christensen, "Implementation of multi-rigid-body dynamics within a robotic grasping simulator," in *Proc. of the 2003 IEEE Intl. Conf. on Robotics and Automation*, 2003, pp. 2262–2268.

[13] C. Ferrari and J. Canny, "Planning optimal grasps," in *Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation*, 1992, pp. 2290–2295.

[14] C. Burges, *A tutorial on support vector machines for pattern recognition*. Kluwer Academic Publishers, 1998.

[15] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[16] A. Smola and B. Schölkopf, "A tutorial on support vector regression," Royal Holloway College, Univerisity of London, Tech. Rep. NC-TR-98-030, 1998.

[17] S. Gunn, *Support Vector Machines for Classification and Regression*. University of Southampton, 1998.

[18] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning*, B. Schölkopf, C. Burges, and A. Smola, Eds. MIT Press, 1999, pp. 185–208.

[19] M. Salganicoff, L. Ungar, and R. Bajcsy, "Active learning for vision-based robot grasping," in *Recent Advances in Robot Learning*, J. Franklin, T. Mitchell, and S. Thrun, Eds. Kluwer, 1996.

[20] R. Kondor and T. Jebara, "A kernel between sets of vectors," in *Proc. of the Intl. Conf. on Machine Learning, ICML*, 2003.

[21] D. Kragić, A. Miller, and P. Allen, "Real-time tracking meets online grasp planning," in *Proc. of the 2001 IEEE Intl. Conf. on Robotics and Automation*, 2001, pp. 2460–2465.