Grasp Planning via Decomposition Trees

Corey Goldfeder, Peter K. Allen, Claire Lackner, Raphael Pelossof

Abstract— **Planning realizable and stable grasps on 3D objects is crucial for many robotics applications, but grasp planners often ignore the relative sizes of the robotic hand and the object being grasped or do not account for physical joint and positioning limitations. We present a grasp planner that can consider the full range of parameters of a real hand and an arbitrary object, including physical and material properties as well as environmental obstacles and forces, and produce an output grasp that can be immediately executed. We do this by decomposing a 3D model into a superquadric 'decomposition tree' which we use to prune the intractably large space of possible grasps into a subspace that is likely to contain many good grasps. This subspace can be sampled and evaluated in GraspIt!, our 3D grasping simulator, to find a set of highly stable grasps, all of which are physically realizable. We show grasp results on various models using a Barrett hand.**

I. Introduction

GRASPING is something that humans can do with ease but robots can currently do only under carefully constrained conditions. Choosing an appropriate grasp for a given object requires knowledge of the object's geometry as well as knowledge of the geometry and kinematics of the robotic hand executing the grasp, but it is hard to analytically solve for all of these variables at once. Instead, some researchers [1][2] have embraced the idea that the best way to find a real world usable grasp for a given hand and object is to dynamically simulate the hand executing a set of likely candidate grasps and choose the most successful candidate.

The key advantage of grasp planning via simulation is the ability to weed out grasps that might seem analytically promising but would fail if executed in reality, and conversely to find good grasps that might appear analytically weak but in fact succeed due to the quirks of the hand hardware. Simulation based planning also allows additional variables such as friction, environmental obstacles, or external forces to be taken into consideration when selecting a grasp. Unfortunately, the space of grasps with a dexterous hand is too large to sample in its entirety, making brute-force simulation of all possible grasps infeasible. In this paper we show how to prune the search space of candidate grasps via multilevel decomposition of the target object. Unlike many other works on grasp planning our method does *not* attempt to find a set of good grasps analytically. Instead, we attempt to define a small search space which is likely to contain *many* good grasps. The grasps lying in this smaller space can then be sampled and evaluated in simulation to find stable output grasps. Our goal is to preserve the significant advantages of simulation-based grasp selection while significantly decreasing the number of costly simulations.

The core intuition underlying our approach is that most objects can be decomposed into component parts and that a good grasp on part of an object or on several parts at once is likely to be a good grasp on the entire object. This idea is sometimes referred to as 'grasping by parts.' To find such grasps we decompose the object into a binary tree with the initial model at the root and each branch representing the division of a model into two parts. We note that the smaller a part is relative to the hand, the less important its detailed shape is for grasp planning, and so we represent each component as a superquadric, which is a simple closed surface described below. Our complete object representation, which we refer to as a 'decomposition tree,' is a multilevel tree of superquadrics created using an automatic decomposition of the initial model. Although some object parts might be poorly represented by superquadrics, we will show how superquadrics can be used not only for modeling but also to drive decomposition, increasing the likelihood that the decomposition tree will be useful.

A. Previous Work

The use of superquadrics as an intermediate representation for a grasp planner is not new in and of itself. Katsoulas and Jaklic used individual superquadrics to model deformable sacks to be picked up by a vacuum gripper [3], and Salganicoff, Ungar and Bajcsy used active learning to find good grips for superquadrics in the plane [4]. Our problem is more general: we use complex models requiring multiple superquadrics, and we do not restrict position and orientation in any way. We also wish to plan grasps for multi-fingered dexterous hands, not just simple grippers. Boughorbel et al. suggested the use of multiple-superquadric models for grasping [5], but limited their discussion to a structured planar environment and did not show how to obtain a grasp from this primitive representation.

There is a good deal of literature on model-based analytical grasp planning, and we direct the reader to recent surveys such as [6] and [7]. Analytical grasping approaches suffer from the inability of a simplified model to encapsulate all of the factors that affect grasp quality. It is beyond the scope of this paper to discuss analytical approaches in any greater depth. Liu, Lam, and Ding demonstrated how to

Manuscript received September 15, 2006. This work was supported by NSF Grant No. IIS-03-12693 and by a National Defense Science and Engineering Graduate Fellowship.

All authors with Columbia University, NY, NY 10027 USA.
Emails: {coreyg, allen, pelossof, cnl2007} @cs.columbia.edu

heuristically find force closed contact points on a discretely sampled surface [8], an approach superficially similar to ours, but their method does not account for scale and assumes only point contacts. Morales, Sanz, del Pobil, and Fagg demonstrated a vision based grasp planner that accounts for hand geometry but limited themselves to 2D grasping [2]. In our own previous work we implemented simulation-based grasp planning for an arbitrary 3-D mesh [1][9], making use of hand geometry and fully accounting for kinematics, scale, environmental obstacles, friction, and planar or even deformable contacts [10] as opposed to the more usual point contacts found in much of the grasping literature. However, our earlier planner required a manually constructed primitive decomposition of the object. This work represents significant progress, as we have removed the need for a manual decomposition, introduced a multilevel representation that helps us find many good grasps rather than a single grasp, and broadened the grasp search space by planning on superquadrics rather than more basic primitives.

II. SUPERQUADRICS

A superquadric [11] is a surface defined as the spherical product of two Lamé curves. Following the computer vision convention, we are using the word 'superquadric' informally, as we are only interested in convex superellipsoids, where the shape parameters are restricted to the domain [0, 2]. An arbitrary superquadric in space is fully specified by its two shape parameters ε and η, three scale coefficients, three Euler angles for orientation, and three Cartesian coordinates for position. This combination of representational power and conciseness means that superquadrics are an excellent way of approximating 3-D volumes for a variety of tasks. For grasp planning, superquadrics have the advantage of encapsulating not only the approximate volume of the original object but also approximate surface normals. Other volumetric simplifications such as voxels or oriented bounded boxes capture little or no normal information.

To approximate a 3D object with a superquadric we sample the object's surface and produce a point cloud. We hen use a nonlinear fitting technique such as Levenberg-Marquardt to find a superquadric whose surface approximates the point cloud in a least squares sense. This is hindered somewhat by the fact that there is no known analytical method to calculate the exact distance between a superquadric and a point. Instead, we can make use of the superquadric "inside-outside" function:

$$F(x,y,z)=\left(\left(\frac{x}{a_1}\right)^{\frac{2}{\eta}}+\left(\frac{y}{a_2}\right)^{\frac{2}{\eta}}\right)^{\frac{\eta}{\varepsilon}}+\left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon}}-1$$

which is positive outside the superquadric and negative inside. This function does not minimize the error of fit perfectly because it does not vary linearly with distance from the surface. A somewhat better choice is the distance from a point to a superquadric along the vector between the point and the superquadric's centroid. This is known as the radial Euclidean distance, and we use it in our fitter.

A. Fitting Multiple Superquadrics

While fitting a single superquadric is useful, we want an automatic way of approximating a model with a number of superquadrics. There are several published algorithms for breaking up a complex point set into smaller superquadric-representable pieces. Jaklic, Leonardis, and Solina demonstrated a "recover and select" region growing algorithm for fitting superquadrics to 2.5-D range data [11][12], which consists of placing a grid of "seed" superquadrics throughout the point data, growing them all at once, and selecting the best fits from the results. This approach works best with regularly spaced range data and less well with unstructured point clouds. It produces no topological information or relationships between the final superquadrics. Chevalier, Jaillet and Baskurt presented a "split-merge" approach better suited for unstructured clouds [13]. In the split stage of their algorithm a single superquadric is fit to the entire point cloud. If the error-of-fit is above a threshold the point cloud is split along a central plane orthogonal to the principal axis, and the algorithm is called recursively on the resultant clouds. In the merge stage neighboring superquadrics are considered and the neighbors whose union can be best fit as a single superquadric are merged, until the total error is below a threshold. This algorithm produces not only an approximation of the cloud but also a topologically meaningful decomposition tree showing which superquadrics were merged. An alternative method of choosing the splitting plane, taking into account both the distribution of the fitting error and concavities in the underlying 3-D data, was proposed by Zha, Hoshide, and Hasegawa [14] but we found that simple split planes of [13] resulted in excellent decompositions. An analysis of the time complexity of decomposition can be found in the Appendix.

B. Decomposition Trees

Although the merge step automatically stops when the total error of fit crosses a threshold, the final number of superquadrics is not always optimal for grasp planning. Small superquadrics that represent individually graspable features but contribute little to the total error can be mistakenly merged into larger components. Conversely, more components beyond a certain point do not significantly improve grasp planning. To ensure that the tree contains enough detail for grasping while avoiding unnecessarily fine decomposition, we force the merge state to terminate at a predetermined number of superquadrics, which we refer to as n. These n components form the leaves of our tree. We then continue merging. Each merge step joins exactly two superquadrics, and thus the process of merging the n leaf nodes into a single root node creates a binary tree from the bottom up, with $2n$-1 nodes. Fig. 1 shows a teddy bear and its decomposition tree. Descending the tree, each superquadric represents a simpler piece of the object. Higher

levels of the tree give gross shape and position information, while lower levels give more precise structure.

An appropriate choice of n should depend on the dexterity of the robotic hand. In our experiments we used the Barrett hand, which has a hard palm surrounded by three multi-jointed fingers, two of which can synchronously swivel up to 180 degrees around the palm. For this hand we found n=6 to be well suited for grasp planning.

III. Grasp Planning

As described above, the space of possible grasps is too large to search exhaustively. Our contribution is to reduce the search space drastically by planning on the simplified superquadric tree rather than the actual object. The reason this works is that while we *plan* grasps only on the superquadric approximations, we *simulate* these planned grasps on the original 3D model. The decomposition trees are used only to delineate a grasp subspace for the simulator to explore. Formally, given a 3D model and its superquadric decomposition tree, we define the grasp subspace G as the union of stable grasps for the $2n$-1 superquadric nodes in the tree. At first glance G does not appear to be substantially smaller than the unrestricted grasp space, but superquadrics, being convex and symmetric, are easy to plan grasps for and thus G is easy to sample for simulation candidates.

To sample G, we need to sample the stable grasps of a single superquadric, for which we use a heuristic. We place approach points on the superquadric's surface at approximately 1 inch intervals, a spacing that provides a good balance between undersampling G and unnecessarily raising the cost of simulation. For each point we start the hand a small distance from the superquadric and approach palm first along the surface normal, closing the fingers upon contact. This is similar to the heuristics of [1]. We use a tripod grip with the angles between the three fingers of the Barrett hand fixed at 60 degrees, but it is possible to sample the space of finger positions as well. For each target point we tried 3 grasp orientations, with the Barrett hand's "thumb" aligned with the cross product of the approach vector and each of the principal axes of the superquadric. Not all grasps produced in this fashion are stable on the superquadric, and therefore not all of the grasps lie within G. Nevertheless, with a sufficient sampling density of approach points it can be assumed that the grasps which *are* stable represent a good sampling of G and thus we are at worst sampling a superset of G. Fig. 2 shows the sampling of G produced by this method for the teddy bear object.

A. GraspIt!

We next simulate the candidate grasps on the input model and sort the results by grasp quality. For this we use GraspIt! [9], a grasp planning and analysis tool developed by our group. GraspIt! incorporates a full dynamics simulator and can import models of many hands, including those with deformable surfaces such as a human hand [10]. We can specify material properties for both the hand and an imported

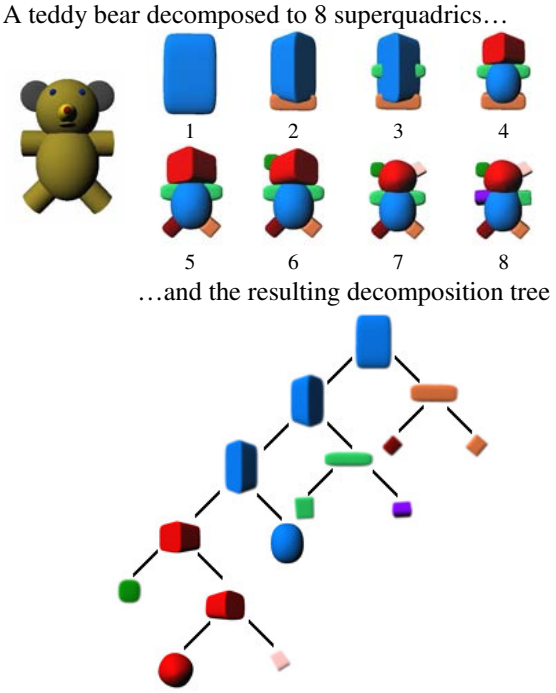

Fig. 1. A decomposition tree, with n=8. The leaf nodes are merged pairwise to form a binary tree from the bottom up. Note to readers: figures should be viewed in color.

3D object and model frictional and gravitational forces. The simulated hand can also be mounted on a robotic arm or mobile platform, and both hand and object can be placed in a larger 3D environment that might include obstacles or a surface that the object rests on. Built in collision detection detects and discards grasps that require the hand to travel through an obstacle or through part of the object. Grasp simulation is very fast, which makes it possible to simulate large numbers of grasps efficiently.

We are interested in finding grasps that are not merely force closed, but that are closed to *reasonably large* forces and torques. The set of wrenches that a grasp can resist depends both on the grip strength and on the contacts and friction between the hand and the object. The relationship between grip strength and resistible wrenches is linear, and so we measure quality in terms of a unit strength grip. Grasp quality, as described by Ferrari and Canny [15], is taken as

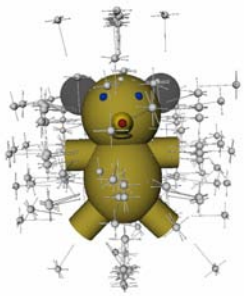

Fig. 2. Approach vectors and hand orientations representing a sampling of the grasp subspace G for a complex object composed of many superquadrics.

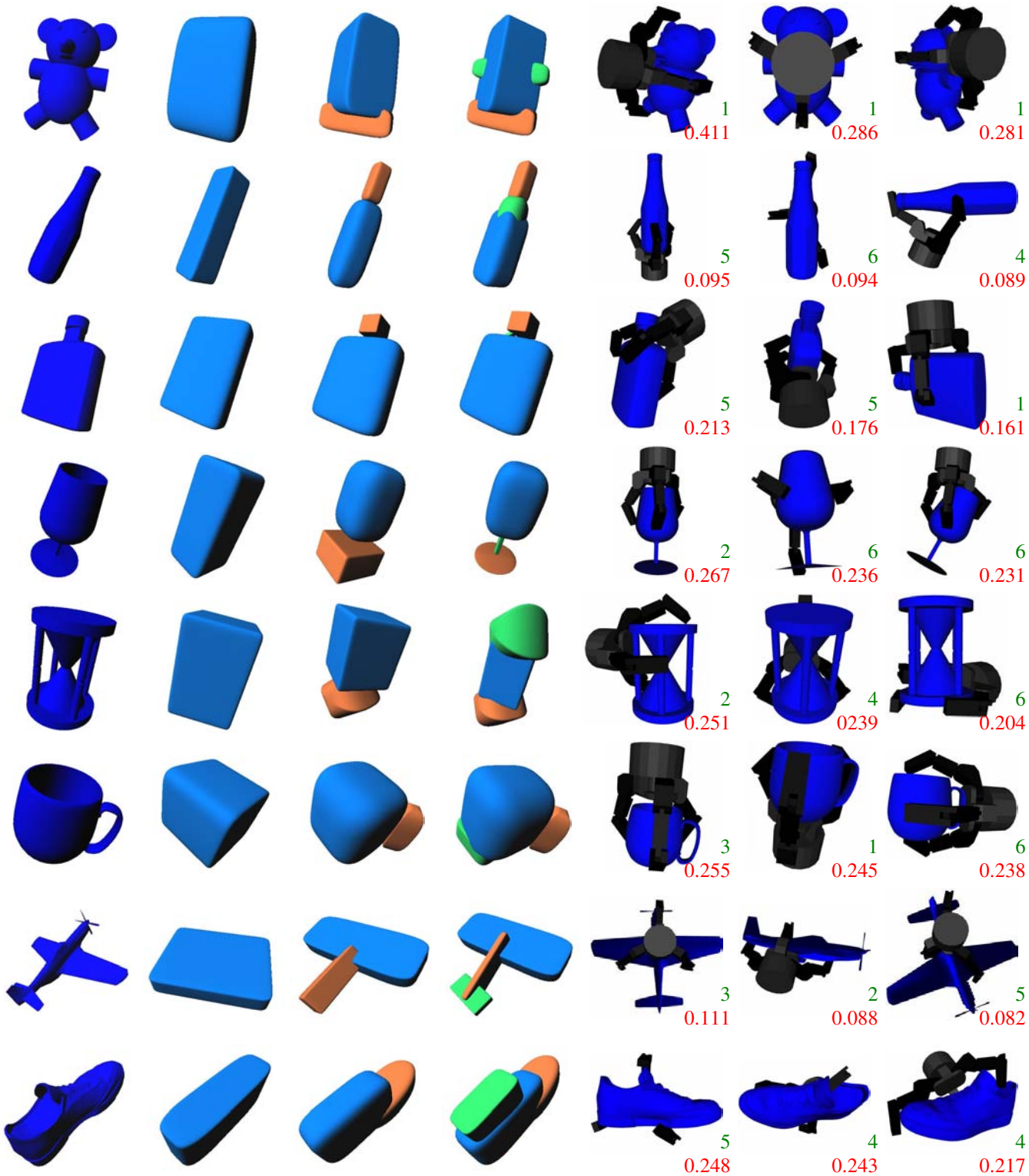

Fig. 3. Some of the test objects. Left to right, the original mesh, the first 3 levels of automatic decomposition, and the 3 best distinguishable grasps planned on a tree with 6 levels of decompositions. The numbers below each grasp are the level of decomposition where the grasp was found and the grasp quality

the magnitude of the largest worst-case disturbance wrench that can be resisted by this grasp with a unit strength grip. A quality of 0 means that there is some wrench the grasp cannot resist, while 1 means that the grasp can exactly resist any force or torque up to the maximum grip force the hand is capable of providing. Experimentally we have determined that a quality of 0.1 or greater corresponds to grasps that a human would consider "stable." Even though *any* grasp with a quality greater than 0 is force closed, we are interested only in these "stable" grasps because we aim to produce realistic grasps that will actually be useable in real world conditions.

B. Using Decomposition Trees

Like our earlier planner [1], our planner takes a grasping-by-parts approach to grasping complex objects. Grasp planning on individual object components in this fashion can often miss good grasps that take into account more than one object segment at a time. Our new object representation was created to overcome this limitation. We use the planner to create candidate grasps from each level of the decomposition tree, rather than only planning on the finest level of decomposition. This is not terribly expensive as there are only a total of $2n$-1 superquadrics. In general, candidates generated from a component high in the tree take into account many subcomponents, but only approximately, while candidates generated from a lower level component will make use of local features but possess less information about the overall shape of the object. Using the entire tree thus protects us from taking either too local or too global a view of the object. Additionally, we can often avoid the extra simulation costs by using a greedy strategy of planning first on the highest level components and only considering finer components if no candidate meets a given quality threshold.

Using superquadrics implies that we might have difficulty with objects that cannot be well approximated by convex primitives. In the limit, as the tree height grows very large, difficult portions of the shape are represented by small, thin, superquadrics which locally approximate patches of the surface. With a sufficiently large tree height these small surface elements can adequately represent any object. In practice, though, our method performs well for complicated objects even using a small tree capped at n, because we use the superquadrics only to set the initial position and orientation of the hand, and then allow GraspIt! to find the actual points of contact on the original object. Even a few superquadrics generally contain enough shape information to predict good initial positions.

IV. Experimental Results

To test our method we planned grasps on a variety of objects. We used the Barrett hand, but our general approach can easily be adapted to any hand, particularly those already in the library included with GraspIt!. We set the material of the fingertips to be rubber to match the rubber-tipped fingers of our real Barrett hand, and the object material to be plastic, for a coefficient of friction of 1.0.

Eight of our example objects, along with a portion of their decomposition trees are shown in Fig. 3. Superquadrics at higher levels of the tree will often be similar but not identical to ones at the lower levels, and so our method tends to generate many grasps that are visually indistinguishable but numerically slightly different. For each object we show the 3 best visually distinguishable grasps computed by our method. The number below each grasp represents the Ferrari and Canny grasp quality as discussed in Section III.A. Our planner found stable grasps for all of the test objects. Some of the grasps in Fig. 3, such as those for the ketchup bottle, do not appear to be force closed, but the frictional forces provided by the large contacts between the fingers and the object account for their stability. For many objects, including the bear, the hourglass, and the flask, even the single root superquadric encapsulates enough shape information for the planner to find a stable grasp, which allowing the planner to return a strong grasp very quickly. Some of the objects, however, were not adequately grasped until the planner reached the 3rd level. Even though we were able to find stable grasps high in the trees, most of the *best* grasps came from lower levels of the trees, confirming our expectation that planning on successively finer components is likely to find better grasps than simply returning the first stable grasp.

Several of the grasps in the figure do not correspond to the intuitively best grasps that a human would use on the same objects. Humans choose grasps based on many criteria other than grasp quality, including comfort, convenience, and object-specific rules (such as a cup full of liquid never being turned upside down). The grasps in Fig. 3, on the other hand, were chosen as 'best' using grasp quality as the sole criterion. Most of the intuitive grasps for the test objects are in fact generated by our technique, but have lower grasp quality values than the grasps shown. Adding additional constraints to filter the returned grasps would allow these 'natural' grasps to be brought to the fore.

To demonstrate the advantages of a multilevel representation, we compared our planner to a hypothetical planner that uses only a single level of decomposition. Rather than predetermining the number of superquadrics for this experiment, we ran the planner n times on each object, with 1 to n superquadrics, and chose the level which contained the best grasp as the "single-level" representation. This means that the hypothetical planner had *a priori* knowledge of the level containing the best grasp, which a real planner would not be aware of. Even with this extra help, the multilevel planner outperformed the single level planner at a running time cost of only a constant factor of 2, as shown in Fig. 4 for 3 objects. The way we chose the single level means that the best grasp from both methods is by definition the same, but the quality of the additional grasps obtained from any single level decreases much faster than the quality of the grasps from the full tree. In many applications more than one good grasp might be needed for a given object, since the best grasp may be occluded or may violate an application-specific constraint (such as a grasp that turns a

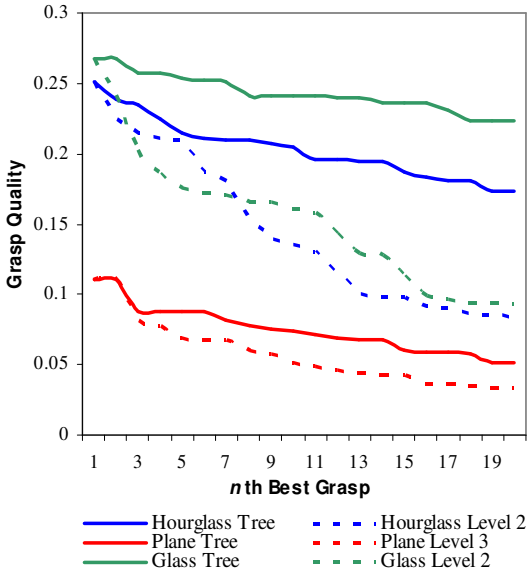

Fig. 4. Grasp qualities using decomposition trees vs. a single level of decomposition, for three objects.

mug full of hot coffee upside down). Decomposition trees have the useful property of providing better secondary grasps than a single level of decomposition would, and as mentioned above, a single level planner is unlikely to find even the best initial grasp every time without *a priori* knowledge of the optimal number of components.

To isolate the improvement over our previous work due to automatic decomposition, we compared our planner to our earlier manual decomposition planner [1] on the same input models. For the automatic decomposition planner we used n=6, and for the manual decomposition planner we used the decompositions used in [1]. A formal comparison is difficult, as the earlier planner's results are very dependent on the specific human input decomposition, but in testing we found our new planner to produce grasps with quality similar to and often much greater than those from the earlier planner.

V. Future Work

Our sampling heuristic generates a large number of candidate grasps, not all of which lie strictly within G. If we could reliably determine the stable grasps for a single superquadric we would have fewer candidates to evaluate. In previous work [16] we used GraspIt! to train a Support Vector Machine for superquadrics so that gradient ascent can quickly find stable grasps. We are currently working to integrate this machine learning approach into our system. A challenge is using the SVM to find many good grasps without having them cluster very closely in parameter space.

Appendix

Our decomposition is based on [13], and although that work did not include a complexity analysis we offer one here. Fitting a superquadric to a point cloud using Levenberg-Marquardt has a time complexity of $O(n^3)$ in the number of points, and so the split stage is worst case $O(n^4)$, where each split outputs a single point and a cloud of n-1 points. In the best case, where each split evenly divides the cloud, the fitting of the initial superquadric dominates the function and the time complexity is $\Omega(n^3)$. In practice, since the splitting plane is always chosen to include the centroid of the cloud, the splits are roughly equal and performance is close to $\Omega(n^3)$. Additionally, while this analysis assumed that splitting continues until each point cloud is reduced to a single point, a reasonable error threshold will usually cause the split stage to terminate quite early.

In [13] the merge stage is $O(n^5)$ because its definition of neighboring superquadric can sometimes force it to fit a merged superquadric to every possible pair. We modified the algorithm to consider at most k nearest neighbors resulting in an $O(n^4)$ algorithm that produces results very similar to the $O(n^5)$ version. In this work we chose k to be 7.

Planning grasps on the decomposition tree is done in a greedy fashion, as described in Section III.B. In the worst case when we need to plan grasps on all $2n$-1 nodes, the simulation time is still only $O(n)$. In practice, unless the density of the sampling heuristic is set extremely high, the cost of decomposition far exceeds the cost of simulation.

References

[1] A. Miller, S. Knoop, P. Allen, H. Christensen, "Automatic Grasp Planning Using Shape Primitives," *IEEE ICRA*, pp. 1824-29, Sep. 2003.

[2] A. Morales, P. Sanz, A. del Pobil, A. Fagg, "Vision-Based Three Finger Grasp Synthesis Constrained by Hand Geometry," *Robotics and Autonomous Systems*, 54(6):496–512, 2006

[3] D. Katsoulas, A. Jaklic, "Fast Recovery of Piled Deformable Objects Using Superquadrics," *DAGM,* Sep. 2002.

[4] M. Salganicoff, L. Ungar, R. Bajcsy, "Active Learning for Vision-Based Robot Grasping," *Machine Learning*, Volume 23, May 1996.

[5] F. Boughorbel, Y. Zhang, S. Kang, U. Chidambaram, B. Abidi, A. Koschan, M. Abidi, "Laser Ranging and Video Imaging for Bin Picking," *Assembly Automation*, Vol. 23, pp. 53-59, Mar. 2003

[6] A. Bicchi "Hands for Dexterous Manipulation and Robust Grasping; A difficult road toward simplicity," *IEEE T-RA*, 16(6):652–62, 2000

[7] A. Bicchi, V. Kumar, "Robotic Grasping and Contact: A Review," *IEEE ICRA*, pp 348-53, 2000

[8] Y. Liu, M. Lam, D. Ding, "A Complete and Efficient Algorithm for Searching 3-D Form-Closure Grasps in the Discrete Domain," *IEEE T-RA*, 20(5):805–16, 2004.

[9] A. Miller, P. Allen, "Graspit!: A Versatile Simulator for Robotic Grasping," *IEEE Robotics and Automation Magazine*, V. 11, No.4, Dec. 2004, pp. 110-22.

[10] M. Ciocarlie, A. Miller, P. Allen, "Grasp Analysis Using Deformable Fingers," *IEEE IROS,*, Aug. 2005.

[11] A. Jaklic, A. Leonardis, F. Solina, *Segmentation and Recovery of Superquadrics,* Comp. Imaging and Vision 20, 2000.

[12] A. Jaklic, A. Leonardis, F. Solina, "Superquadrics for Segmenting and Modeling Range Data," *IEEE PAMI*, 19(11):1289–95, Nov. 1997.

[13] L. Chevalier, F. Jaillet, Atilla Baskurt, "Segmentation and Superquadric Modeling of 3D Objects," WSCG, Feb. 2003.

[14] H. Zha, T. Hoshide, T. Hasegawa. "A Recursive Fitting-and-Splitting Algorithm for 3-D Object Modeling Based on Superquadrics." *Third Asian Conf. on Comp. Vision,* Jan. 1998.

[15] C. Ferrari, J. Canny, "Planning Optimal Grasps," *IEEE ICRA*, pp 2290-95, 1992.

[16] R. Pelossof, A. Miller, P. Allen, T. Jebara, "An SVM Learning Approach to Robotic Grasping," *IEEE ICRA*, pp. 3212-18, Apr. 2004.