Grasp Transfer by Parts

Eli Bronstein* Amit Talreja*

***Abstract*—Grasping is a subfield of robotics research that focuses on enabling robots to manipulate objects. Grasping is hard because of the large space of possible grasps that must be considered. We sought to decrease the complexity of planning grasps on objects by 1) breaking an object into component parts, and 2) transferring known-good grasps to these parts from parts of previously seen objects, with the novel consideration that the two objects need not be from the same class. While this system produced some successful transferred grasps, the overall quality of transferred grasps was lower than grasps produced by direct computation. Further information can be found at https://sites.google.com/berkeley.edu/grasptransferbyparts/home**

I. Introduction

Grasping by parts is a method of grasping that is based on ideas from the human visual system. In (Biederman), Biederman proposes that humans recognize objects by visually decomposing them into component "blocks cylinders, wedges, and cones" and other objects, and that humans may not recognize objects as a whole because there is too much variation and it is simpler to break them into common components. Similarly, grasping by parts separates an object into its components so that grasps can be planned on the components instead of the full objects. The theory is that some parts of an object are more natural to grasp, such as the handle on a cup, so planning grasps on these semantically meaningful components might result in more successful grasps than simply planning over the whole object and hoping a grasp is successful. These parts that are more amenable to being grasped are called *affordances*, and have been extensively explored in human factors literature, such as (Gaver).

Grasp transfer aims to reduce the time spent computing grasps by taking known-good grasps from one object and warping or transforming them to apply to other, similar objects. The hope is that this transfer process is simpler and cheaper than computing grasps on the new object directly.

Our method combines the ideas of grasping by parts and grasp transfer. We take a group of objects, segment them into functional parts, and plan grasps on each part. Given a new object we segment it and then try to find parts in the database that are similar to the new parts. Once similar parts are identified, we transfer the precomputed grasps to the new parts and evaluate their efficacy.

Our method achieves varying results on different objects. For each object some percentage of the transferred grasps are invalid because they do not make contact with the surface of the queried object. We find that the percentage of valid transferred grasps varies from 21 to 70 percent depending on the object. For the transfer grasps that are valid, we find that their quality decreases by 0.16 in grasp quality on average.

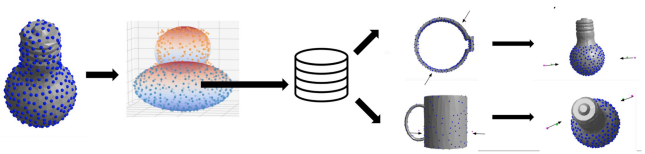


Fig. 1: Overview of the grasp transfer by by parts process. Given a query object, we segment it into parts using superquadric decomposition trees. For a specific query part, we find the most similar parts of other objects from a database that contains individual parts with associated precomputed grasps. Next, the precomputed grasps on the database parts are transferred to the query part and processed to lie directly on the query part's surface.

II. Related Work

A. Grasping By Parts

Prior work on grasping by parts has focused on different ways to segment an object to find the semantically relevant components. The central idea in all cases is to plan on a part of the object because it reduces complexity as compared to planning on the whole object.

(Goldfeder et. al) use a similar superquadric decomposition tree method to ours as a way to reduce the search space for grasping. They recognize that superquadrics are very efficient to sample and plan grasps on, so they plan grasps at each level of the superquadric decomposition tree (effectively on each part) and then simulate the grasps on the actual 3D models to see which are feasible.

(Kai et. al) believe that representing an object by shape primitives is a fruitful avenue of research to cut down the search space of grasps but they want to answer the question "how rudimentary can a model of a thing be in order to be handled succesfully and efficiently?" They use a tree decomposition process to fit rectangular bounding boxes to parts of objects as a tradeoff between accuracy and efficiency of the representation.

Although a slightly older paper, (Miller et. al) is interesting because it seems to be the precursor for a lot of the more modern grasping by parts work. The authors manually decompose objects into their component shapes (for example, they show a coffee cup manually modeled as a cylinder with a box for the handle), and then manually define the ways in which each shape primitive can be grasped.

B. Grasp Transfer

As stated earlier, grasp transfer is the idea of taking a good grasp on one object and finding a transformation such that it can be used as a good grasp on a similar object to avoid having

to go through the expensive process of planning a grasp on the new object.

In (Hillenbrand et. al) the authors present a method for transferring grasps between objects of the same functional category. They achieve this by warping the source object into the target object and then using the warp information to find new vertices for the grasp on the target object. They then use local grasp re-planning to make sure that the grasp is feasible on the new object.

Unlike the previous work, (Kopicki et. al) does not explicitly require that objects be in the same category for the grasp to transfer. The authors describe a method that uses probability density functions to model the finger contacts and the hand configurations. On new objects grasps are chosen to maximise the product of these densities, thus "transferring" the information learned from prior grasp situations.

C. Part-based Grasp Transfer

We also found prior work that combined the ideas of grasp transfer and grasping by parts.

(Aleotti and Caselli, 2012) use Reeb graphs to segment objects. A Reeb graph is a topological construct that "tracks the topology changes in the level-sets of [a] scalar function" on the object. In this way it generates a canonical representation of an object that can then have its parts annotated by performing a graph matching with Reeb graphs from previously seen objects. This is possible because the authors assume that "Objects to be grasped are assumed to belong to a set of known classes of objects, where all objects of the same class approximately share the same topology". Grasps can then be planned on the parts of the object that are annotated as affordances. Although this is not strictly transfer grasping by parts because only the labels are transferred and not the grasps, it does involve both prior knowledge from other objects and the idea of grasping by parts.

(Aleotti and Caselli, 2011), also by the same authors, continues the idea of using Reeb graphs for segmentation but uses them more explicitly for retrieval by adding the additional step of using human-demonstrated grasps. Grasps are demonstrated by a human in a virtual reality system, and the Reeb graph allows for similar parts to be identified so that grasps from the human can be transferred to new objects.

(Matl et. al) is a true method of transfer grasping by parts. In this work a mesh segmentation algorithm is used to directly compute components of an object from the mesh without using an intermediate form like a Reeb Graph or superquadrics. Once the segments are obtained they are matched to similar segments using D2 shape descriptors and Gaussian Mixture Models. Grasps are transferred by aligning the similar segments using a point cloud registration algorithm to recover the homogeneous transformation between them. This work is closest in structure to the method we implemented, replacing the particular ways of doing segmentation, matching, and transfer with our use of superquadrics.

(Detry et. al) presents a method of grasping by parts that aims to learn useful part prototypes by using the shape of the parts and the grasp examples on the provided objects. This technique is meant to increase the generalizability of grasp transfer by being more flexible in part selection than other existing segmentation algorithms.

Finally, (Vahrenkamp et. al) uses mesh segmentation to get the components of an object before planning grasps on each part. When presented with a new object it is again segmented and a corresponding graph from the database of prior grasps is retrieved. The paper is extrmeley light on actual details but the system described seems similar to that of (Matl et al.).

III. Methods

A. Goals

The goal of this work is to use a grasping-by-parts approach to transfer precomputed grasps from a database of object affordances to a new object. To evaluate the success of our method, we focus on the performance of its two main components: object segmentation and grasp transfer. First, to assess the performance of the object segmentation step, we qualitatively observe whether the resulting object parts are useful affordances for grasping and how similar the segmentation is to one performed by a human. Second, we compare the force closure grasp qualities of the transferred grasps with the qualities of grasps on the same object parts computed in a standard way (i.e. by sampling many grasps and selecting the ones with the highest quality).

B. Object Segmentation via Superquadric Decomposition Trees

Superquadrics are a broad family of parametric shapes including cubes, cylinders, superellipsoids, octahedra, and spindles, among others (Solina and Bajcsy). They have been widely used as primitive shapes in computer graphics. For the purposes of this work, we restrict ourselves to convex superellipsoids, as these are successful enough at approximating many common objects. A superquadric is defined by a total of 11 parameters: two shape parameters η and ω, three scale parameters a_1, a_2, and a_3, three Euler angles for rotation, and and three Cartesian coordinates for translation. We refer to these parameters as Λ.

The surface of a superquadric with zero rotation and translation is defined by the following vector (Solina and Bajcsy):

$$\boldsymbol{x}(\eta,\omega) = \begin{bmatrix} a_1 \text{sgn}(\cos(\omega))\cos^{\epsilon_1}(\eta)|\cos(\omega)|^{\epsilon_2} \\ a_2 \text{sgn}(\sin(\omega))\cos^{\epsilon_1}(\eta)|\sin(\omega)|^{\epsilon_2} \\ a_3 \text{sgn}(\sin(\eta))|\sin(\eta)|^{\epsilon_2} \end{bmatrix}$$

where $-\pi/2 \leq \eta \leq \pi/2$ and $-\pi \leq \omega \leq \pi$.

Parameters ϵ and η can be eliminated from the superquadric parametric equation to obtain the superquadric inside-outside function:

$$F(\Lambda, x, y, z) = ((\frac{x}{a_1})^{\frac{2}{\epsilon_2}} + (\frac{y}{a_2})^{\frac{2}{\epsilon_2}})^{\frac{\epsilon_2}{\epsilon_1}} + (\frac{z}{a_3})^{\frac{2}{\epsilon_1}}$$

(Goldfeder et al.). F takes on negative values for points inside of the superquadric, positive values for points outside of the superquadric, and zero on the surface of the superquadric.

1) Fitting a Superquadric: The process of fitting a superquadric to a set of points involves minimizing the error between the surface of the superquadric and the locations of the points. Simply minimizing the sum of the values of the inside-outside function for each point does not succeed because it does not vary linearly with the distance from a point to the superquadric surface. Computing the exact distance from a point to the surface of a superquadric is difficult and computationally expensive, so various approximations have been used for this distance.

We use an approximation similar to the one presented by Solina and Bajcsy (Solina and Bajcsy),

$$G(\Lambda, D) = \sqrt{a_1 a_2 a_3} \sum_{i=1}^{N} (F^{\epsilon_1}(\Lambda, x_i, y_i, z_i) - 1)^2$$

where D is the point cloud of size N.The $\sqrt{a_1 a_2 a_3}$ term of the object function seeks to find the smallest superquadric describing the given points, and the $F^{\epsilon_1} - 1$ terms seeks to minimize the distance between the points and the surface of the superquadric. Raising F to the power of ϵ_1 is done in order to prevent the explosion of the value of F when ϵ_1 takes on a small value. To fit the superquadric to a set of points, we minimize G over the set of points. Although other works have minimized the objective function using the Levenberg-Marquardt method for nonlinear least squares optimization, we had more success in using L-BFGS-B. As suggested by (Solina and Bajcsy), we found that good initial estimates superquadric parameters were necessary to result in well-fit superquadrics.

We estimated the translation of the superquadric as the mean of the points and the shape parameters as $\epsilon_1, \epsilon_2 = [1, 1]$, which represents a sphere or ellipsoid. To estimate the rotation, we used the method described in (Solina and Bajcsy) to compute the inertial axes of the point cloud. Specifically, we computed the matrix of central moments which is defined with respect to the center of gravity $(\bar{x}, \bar{y}, \bar{z})$ (Solina and Bajcsy). The inertial axes are then defined as the eigenvectors of the matrix of central moments. The object coordinate system is oriented such that the z axis lies along the longest side of elongated objects or along the shorted side for flatter objects. In other words, if $\vec{e}_1, \vec{e}_2, \vec{e}_3$ are the eigenvectors of M and $\lambda_1 < \lambda_2 < \lambda_3$ are the associated eigenvalues, then we assign the z axis of the object coordinate system in the following way: if $|\lambda_1 - \lambda_2| < |\lambda_2 - \lambda_3|$, then we set $z = \vec{e}_3$. Otherwise, $z = \vec{e}_1$.

Once we estimate the object coordinate system with the inertial axes, we use the Euler angles of this coordinate system as the initial estimate for the optimization. Finally, the shape parameters of the superquadric are estimated as the x, y, and z parameters of the bounding box of the unrotated version of the superquadric.

We also set constraints on the superquadric parameters to improve how well the general shape of the superquadric approximates that of the point cloud. We constrained $\epsilon_1, \epsilon_2 \in [0.1, 1]$ because when $\epsilon_1, \epsilon_2 < 0.1$ the objective function becomes unstable, and when $\epsilon_1, \epsilon_2 > 2$, concavities are introduced in the superquadric shape. We also constrained $_1$, a_2, and a_3 to be within 20% of the estimated a_1, a_2, and a_3 values.

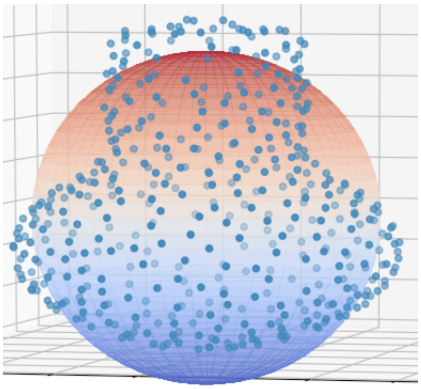


Fig. 2: Example of a superquadric fit to points on the surface of a lightbulb object.

This prevented the issue of very large superquadrics being used to approximate points simply because the points were on the superquadric surface, while much of the surface was not near the points. Finally, the Euler angles were naturally constrained to be in $[0, 2\pi)$ and no constraints were placed on the translation.

2) Fitting Multiple Superquadrics: To best approximate and segment an object, we wish to fit multiple superquadrics to the object's surface. Various methods of segmenting a point cloud into multiple superquadrics have been presented in previous works (Goldfeder et al.).

In this work, we use the split-merge approach to fitting multiple superquadric presented by Chevalier, Jaillet, and Baskurt (Chevalier et al.). This approach consists of two primary stages: the split stage and the merge stage. In the split stage, one superquadric is first fit to the entire point cloud. The point cloud is then split along the plane orthogonal to the primary inertial axis of the superquadric, resulting in two new point subsets. The fitting and splitting process is then recursively applied to the point subsets until a specified terminating condition, which could be a maximum recursion depth or a maximum error of fit threshold. The split stage results in a full binary tree of point subsets and their corresponding superquadrics because each subset is split in half for each iteration of the algorithm. We refer to number of leaves in the split tree as n. In the merge stage, the point subsets are merged two at a time from the bottom up to best describe the affordances of the object. Specifically, we attempt to merge one of the subsets S_1 with each candidate subset S_2 by fitting a superquadric to the points in $S_1 \cup S_2$. The set of candidate subsets is the set of all other subsets resulting from the split stage. Although this requires fitting a large number of superquadrics, attempting merges with every other subset ensures that segmentation result is independent of the order in which the point subsets are considered. The two subsets S_1 and S_2^* are merged if they result in the smallest error of fit compared to the other subset pairs, and if the size of the superquadric approximating the points $S_1 \cup S_2^*$ is smaller than the sum of the sizes of the superquadrics approximating S_1 and S_2^*. The result of the merge stage is a binary tree with n leaves originating from the split tree and $2n-1$ nodes. Higher

levels in the merge tree contain more coarse approximations of the point cloud, while lower levels describe finer details and smaller parts. The advantage of this approach is that it results in a topologically meaningful representation of the different parts of the object, and that it provides varying degrees of segmentation. For a particular object, we only select the top k merge stages, where k is determined empirically, to avoid segmenting the object too finely into semantically meaningless components.

C. Creating a Database of Parts

In order to transfer grasps to a new object, there must be a database of objects with precomputed grasps that are to be transferred. In fact, we require a database of object parts because we aim to transfer grasps from one part to another similar part. Given a set of objects, we segmented each of these objects using the superquadric decomposition tree approach and saved the resulting set of parts as a "flat" database of parts.

D. Computing Grasps

We used a sampling-based approach to find valid grasps. We first sampled points on the surface of the object to use as vertices. We then used rejection sampling to repeatedly pick two points from the set of vertices and throw out any that were not antipodal or were not in force closure. After finding a valid grasp we assigned it to a part of the object by the following method: we computed which part each of the vertices was in by finding the face that it contacted and determining which part each face belonged to. If both vertices belonged to the same part then the grasp was assigned to the part. We continued sampling grasps until we reached a threshold for the number of grasps assigned to each part. This procedure helped us avoid generating grasps that were invalid on the object but valid on the part because we sampled from grasps on the whole object and then assigned them to the particular part they were associated with.

To evaluate the grasps we created a quality metric based on force closure. Instead of finding a binary measure of whether a grasp is in force closure or not we generalize to a non-binary measure in the range of 0-1, where higher grasp qualities are associated with smaller angles between the normals of the two vertices, indicating a more antipodal grasp.

E. Part Similarity

After fitting superquadrics to the query object to obtain its component parts the central problem that remains is matching one of the query parts to a part in the database so that a grasp can be transferred from the database to the new part. This requires a method to measure the similarity between two superquadrics. Although the simplest method may seem to be a direct comparison between the parameters of the superquadrics, discussed earlier, this is not a fruitful comparison because a linear change in parameters (especially the shape parameters and) results in a nonlinear change in the shape of the superquadric.

We implemented the method of comparison described in (Chen et al.) : a Monte Carlo approximation of the difference in volume between two superquadrics. The procedure described consists of removing the translation and rotation parameters from the objects, sampling points uniformly in a cube encompassing both of them, and then calculating the fraction of points which fall inside of one superquadric and outside of the other one, using the inside-out function. We modified the procedure described in the paper slightly by normalizing the scale parameters of the superquadrics: for each superquadric we divided the scale parameters by their largest value so that the scales of the superquadrics were roughly comparable.

We conducted a first-order test of this similarity method by constructing two artificial superquadrics and verifying that the similarity metric showed they were becoming less similar as we manually varied the parameters of each superquadric.

F. Grasp Transfer

Once the most similar superquadric to a given query part is found, we must transfer the grasp from the database to the new part. To do this we find the homogeneous transformation between the position and rotation of the database superquadric and the position and rotation of the query superquadric. We then apply this transformation to the vertices of the grasp from the database to find the coordinates of the grasp in the frame of the new object. As it is likely that the new grasp is not on the surface of the new object, we perform a ray-tracing procedure to find the contact points of the new grasp: we draw an imaginary line between the two vertices and calculate where it intersects the surface of the part to find the new vertices for the transferred grasp. If the line does not intersect the mesh then it does not count as a successful transfer.

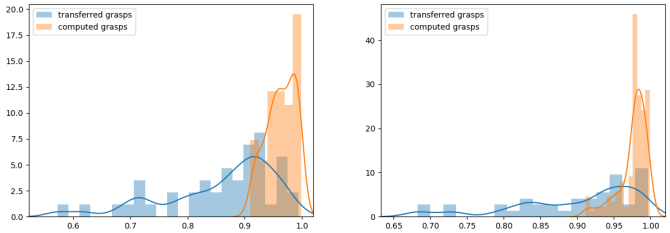
IV. Results

A. Experimental Setup

For our experiments, we created a dataset of parts as described in sec. III-C. To create this database, we first selected a number of objects from the 3DNet dataset (Wohlkinger et al.) and from the Thingiverse dataset referenced in (Danielczuk et al.) that have similar affordances. For example, we selected a screwdriver and a bottle due to their similar cylindrical sections, as well as two rings and a mug due to their similar circular cross sections. The objects included in the database were: a lightbulb, a screwdriver, a mug, two different rings, a bottle, and a door handle extender. For each grasp transfer experiment, we selected one query object from the database and segmented it with superquadrics. For each of the resulting query parts, we found the most similar parts in the database, ensuring the exclude parts from the query object itself. Finally, we transferred the precomputed grasps from each of the database parts to the associated similar query object part. For the grasps, we assumed a parallel jaw gripper with two contact points.

Object	P(Valid Grasp)	Transfer Grasp Quality Mean	Valid Transfer Grasp Quality Mean	Precomputed Grasp Quality Mean
bottle	**0.70**	**0.603**	0.858	0.963
door handle extender	0.24	0.213	0.875	0.961
lightbulb	0.47	0.419	**0.899**	0.976
mug	0.76	0.506	0.670	0.966
ring 1	0.18	0.133	0.756	0.985
ring 2	0.21	0.132	0.640	0.976
screwdriver	0.42	0.368	0.872	0.963

Table 1: Force closure quality of transferred grasps on various objects. The right 4 columns indicate the proportion of valid transferred grasps (i.e. on the surface of the object), the mean quality of all of the transferred grasps, the mean quality of the valid transferred grasps, and the mean quality of all of the precomputed grasps.



Bottle Lightbulb

Fig. 3: Comparison of the force closure quality of valid transferred grasps and precomputed grasps for the bottle and lightbulb objects.

B. Object Segmentation

The object segmentation process via superquadric decomposition trees worked well for most of the objects. The results were particularly good for the lightbulb, the screwdriver, and the bottle. The lightbulb was segmented into the glass bulb and the metal base, and the screwdriver was segmented into the handle, the shank, and the component joining the handle and the shank, and the bottle was segmented into the body and the neck. Not much more semantic segmentation was possible for the rings since they can be easily approximated by cylinders, so the segmentation process simply split them into horizontal levels. Unfortunately, the segmentation of the mug and the door handle extended did not yield semantically meaningfully components, likely due to the fact that the superquadrics were only split across their primary intertial axes in the split stage rather than in more complex ways.

C. Quality of Transferred Grasps

To evaluate the quality of the transferred grasps, we use the force closure grasp quality metric, which we also used to precompute grasps on the parts in the database. Table 1 presents the quality of the transferred grasps for each object. Many of the transferred grasps were invalid, meaning that if they were to be executed by a robot with a parallel jaw gripper, no contact would be made with the object. As can be seen from the first column, the majority of transferred grasps were invalid for most of the objects. We set the quality of these invalid grasps to 0, which resulted in quite small mean grasp qualities for each object. However, we observe that even though the valid transferred grasps are not as good as the precomputed grasps, they are of decent quality for most of the objects and are likely to be successful.

Figure 3 shows the grasp quality distributions of the valid transferred grasps and the precomputed grasps for the bottle and the lightbulb objects. The transferred grasps have a wider variety of qualities, ranging from about 0.6 to 1, while the precomputed grasps are above 0.9 in quality. However, there is a sufficient number of transferred grasps of comparable quality to the precomputed grasps, so by executing the best transferred grasps, a robot would likely succeed in grasping the query object.

V. Conclusion

A. Summary

In summary, we attempted to determine the feasibility of transfer grasping by parts as a method to cut down the complexity of computing grasps on parts of new objects by referencing grasps on parts of previously seen objects.

To do this we curated a database of objects with similar parts, implemented a segmentation procedure based on superquadric decomposition trees, segmented our objects using this procedure, computed grasps on the parts, and then tried to transfer the computed grasps to new objects whose superquadrics were similar to those of the superquadrics in our database.

The overall results from this process were underwhelming: the transferred grasps were invalid (did not intersect the surface of the new part) between 30 and 82 percent of the time (depending on the query object), and the quality of the transferred grasps was much lower than those produced by direct computation. While we did not measure the exact time difference between the transfer process and computing grasps directly, we can qualitatively say that the transfer process was also slower than computing grasps on the new part directly.

B. Challenges

We encountered several challenges in the process of fitting superquadrics to point clouds. In the case of fitting one superquadric, we found it was crucial to properly estimate the initial parameters for L-BFGS-B to converge to a good solution. These estimates became particularly important once nonzero rotation and translation were introduced to the point clouds. In addition, we found that it was necessary to constrain the parameters for the fitting process to be successful for most point clouds. As explained in (Solina and Bajcsy), constraining the shape parameters ϵ_1 and ϵ_2 was particularly important because the superquadric inside-outside function can become numerically unstable outside of these bounds. We decided to additionally constrain the scale parameters a_1, a_2, and a_3 because without these constraints, we obtained superquadrics whose surface approximated the points well but whose shape did not match the expected underlying shape of the point cloud. Constraining the scale parameters to be close to the estimated scale parameters improved this issue. In addition,

speed was initially an issue when fitting superquadrics, particularly for point clouds that were not easily approximated by a superquadric. We were able to increase the speed of the fitting process by providing L-BFGS-B the Jacobian of the error of fit with respect to the parameters rather than analytically computing the Jacobian.

Moreover, in the merge stage of the fitting multiple superquadrics process, it was unclear which neighbors of the current point subset to consider for merging. Considering too few would lead to poor segmentation results, while considering too many was significantly more computationally expensive. Ideally, one would intelligently select the neighbor point subsets based on their topological relationship in the split tree, their Euclidean distance, or some other metric.

Finally, we also had a difficult time finding an accurate method to compare the superquadrics so that a similar part could be reliably retrieved from the database. We curated the dataset to include visually similar parts, but when we implemented the similarity metric the parts that it returned as being most similar did not fit with our visual intuitions. For example, when we tested the lower spherical part of the light bulb object, one of the most similar parts it returned was a non-circular section of the door handle extender object. We were not able to find much literature beyond (Chen et. al), and our modifications to it (like normalizing the scale parameters) did not seem to lead to a metric that matched all of our visual intuition.

C. Future Work

With more time there are a number of research directions that we would wish to explore, starting with having a larger database of objects. We intentionally limited the number of objects in the database because we wanted to be able to iterate quickly on the segmentation and query stages of the procedure, both of which would take much longer with a larger database. While we attempted to choose objects that contained some similar parts, it is possible that the size of the database in our project was too small to contain sufficient objects such that good transfer between similar parts was possible.

Beyond this change, we want to explore other ways of computing similarity between the superquadrics, or going to the underlying mesh segments and using them for computing similarity. Although the volume difference computation method that we are currently using for computing superquadric similarity works in that it produces lower similarity values for superquadrics that we visually classify as not being similar, we think there are more precise methods of finding similar parts to a given part. This is partly because the same superquadric can be used to represent a wide variety of underlying points clouds, and so computing similarity between superquadrics may not be the best way to represent if the two segments are actually similar.

Continuing with this idea of replacing the use of superquadrics in parts of our procedure, a third direction of exploration that we are curious about is the transfer of the grasp from the database part to the query part. Instead of computing the transform between the two superquadrics, we are curious to see the effect of using a procedure similar to (Matl et. al), in which a registration process based on Super4PCS (Mellado et. al) is used to align the actual point clouds corresponding to the two parts. This would hopefully eliminate the problem of the transferred grasp not contacting the surface of the new object and lead to an increase in the quality of the transferred grasp.

REFERENCES

[1] Biederman, Irving. "Recognition-by-components: a theory of human image understanding." Psychological review 94.2 (1987): 115.

[2] Gaver, William W. "Technology affordances." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, 1991.

[3] Goldfeder, Corey, et al. "Grasp planning via decomposition trees." (2007): 4679-4684.

[4] Huebner, Kai, Steffen Ruthotto, and Danica Kragic. "Minimum volume bounding box decomposition for shape approximation in robot grasping." 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008.

[5] Miller, Andrew T., et al. "Automatic grasp planning using shape primitives." (2003): 1824-1829.

[6] Kopicki, Marek, et al. "Learning dexterous grasps that generalise to novel objects by combining hand and contact models." 2014 IEEE international conference on robotics and automation (ICRA). IEEE, 2014.

[7] Aleotti, Jacopo, and Stefano Caselli. "A 3D shape segmentation approach for robot grasping by parts." Robotics and Autonomous Systems 60.3 (2012): 358-366.

[8] Aleotti, Jacopo, and Stefano Caselli. "Part-based robot grasp planning from human demonstration." 2011 IEEE International Conference on Robotics and Automation. IEEE, 2011.

[9] Matl, Matthew, Jeff Mahler, and Ken Goldberg. "An algorithm for transferring parallel-jaw grasps between 3D mesh subsegments." 2017 13th IEEE Conference on Automation Science and Engineering (CASE). IEEE, 2017.

[10] Detry, Renaud, et al. "Generalizing grasps across partly similar objects." 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012.

[11] Vahrenkamp, Nikolaus, et al. "Part-based grasp planning for familiar objects." 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). IEEE, 2016.

[12] Hillenbrand, Ulrich, and Maximo A. Roa. "Transferring functional grasps through contact warping and local replanning." 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012.

[13] Solina, Franc, and Ruzena Bajcsy. "Recovery of parametric models from range images: The case for superquadrics with global deformations." IEEE transactions on pattern analysis and machine intelligence 12.2 (1990): 131-147.

[14] Mellado, Nicolas, Dror Aiger, and Niloy J. Mitra. "Super 4pcs fast global pointcloud registration via smart indexing." Computer Graphics Forum. Vol. 33. No. 5. 2014.

[15] Chen, L-H., Y-T. Liu, and H-Y. Liao. "Similarity measure for superquadrics." IEE Proceedings-Vision, Image and Signal Processing 144.4 (1997): 237-243.

[16] Chevalier, Laurent, Fabrice Jaillet, and Atilla Baskurt. "Segmentation and superquadric modeling of 3D objects." (2003).

[17] Walter Wohlkinger, Aitor Aldoma Buchaca, Radu Rusu, Markus Vincze. 3DNet: Large-Scale Object Class Recognition from CAD Models. In IEEE International Conference on Robotics and Automation (ICRA), 2012.

[18] Danielczuk, Michael, et al. "Segmenting unknown 3D objects from real depth images using mask R-CNN trained on synthetic point clouds." arXiv preprint arXiv:1809.05825 (2018).