An Algorithm for Transferring Parallel-Jaw Grasps Between 3D Mesh Subsegments

Matthew Matl[1,2], Jeff Mahler [1,2], Ken Goldberg [1,2,3]

***Abstract*— In this paper, we present an algorithm that improves the rate of successful grasp transfer between 3D mesh models by breaking each mesh into functional subsegments and transferring grasps between similar subsegments rather than between full models. This algorithm combines prior research on grasp transfer with mesh segmentation techniques from computer graphics to successfully transfer contact points more often while potentially preserving task-specific knowledge across transfers. The algorithm extracts subsegments from each mesh model with a customized segmentation algorithm designed for speed and then groups similar subsegments with D2 shape descriptors and Gaussian mixture models (GMMs). Grasps are then transferred by aligning similar subsegments with Super4PCS, a global point cloud registration algorithm. We experimentally evaluated this algorithm against a non-segmenting baseline on over 20,000 grasp transfers across a set of 80 objects and found that the segmentation-based algorithm improved the success rate for finding a transferred grasp from 82% to 98%. Additionally, grasps transferred with our algorithm were only 8.7% less robust on average than the original grasps without any local re-planning.**

I. INTRODUCTION

Grasp synthesis is the process of selecting a suitable grasp configuration for a rigid object, and it is an active research topic in robotics [1], [2]. Many current methods use physics-based robustness metrics such as Ferrari and Canny's ε-metric [3] and force closure [2] to predict whether a grasp will succeed or fail. These analytic methods can be used to rapidly evaluate thousands of grasp candidates [4], but they are sensitive to errors in modeling and do not encode task-specific information about grasps. Other approaches use human demonstrations to learn high-quality grasp configurations [5], [6]. Human-selected grasps often carry useful functional information – for example, grasps on the handle of a hammer allow users to hit a nail – but grasp demonstrations are expensive to collect, which limits the scalability of these types of planners.

One method for improving both of these techniques is grasp transfer – the process of transforming a grasp on one object to an equivalent grasp on similar objects. For example, a grasp on the handle of one hammer could be transferred to grasps on the handles of other hammers. Grasp transfer can address the scalability issues of human grasp demonstrations by generalizing a single demonstration across entire classes of objects. Furthermore, transferring analytically-generated grasps across a dataset and preserving links between related grasps could allow knowledge gained about a single grasp – for example, affordance labeling for a task – to be propagated to all similar grasps.

[1]The AUTOLAB at UC Berkeley; automation.berkeley.edu
[2]EECS, UC Berkeley; {mmatl, jmahler, goldberg} @eecs.berkeley.edu
[3]IEOR, UC Berkeley; goldberg@eecs.berkeley.edu

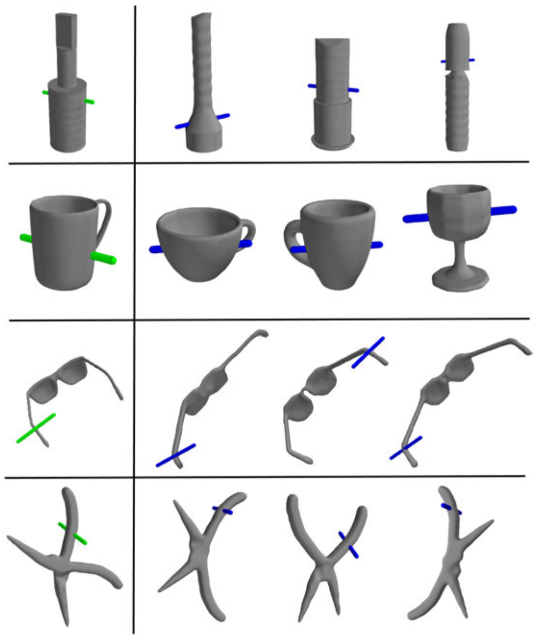
Fig. 1. Transfers of sampled grasps in the bearing, cup, glasses, and plier datasets. **(Left)** The original grasp configuration. **(Right)** Grasps transferred from the original. Articulated objects like pliers and glasses are problematic for prior grasp transfer algorithms since the full meshes are difficult to align, but our algorithm succeeds by transferring grasps between subsegments of the 3D meshes.

Prior grasp transfer methods have generally focused on warping contact points from a source object to a target object [7], [8], [9], [10], [11]. While the exact technique for warping the contacts varies, nearly all of the prior transfer methods begin by attempting to align the full geometries of the source and target objects. However, aligning full object geometries is often difficult, especially for articulated objects. For example, it would be impossible to rigidly align two pliers whose jaws are opened to different angles.

In this paper, we propose an algorithm that addresses this issue by exploiting the observation that similar objects are often composed of a common set of *subsegments* – smaller components that usually have an associated functionality. For example, all eyeglasses are composed of two lenses and two temple arms, and all pliers have two jaws, two handles, and a hinge. Similar subsegments are likely to have less geometric variation than full object models, and prior research suggests that they often carry useful implications about grasp affor-

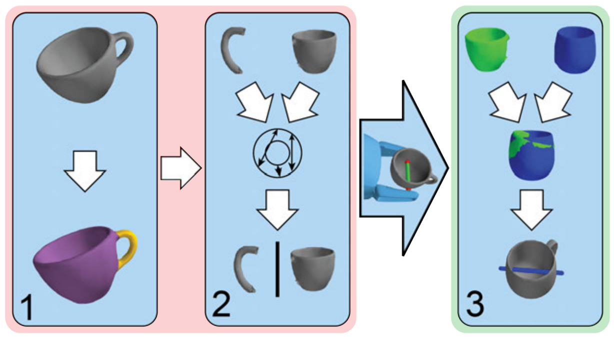

Fig. 2. Overall grasp transfer algorithm. **(1)** Each 3D mesh is decomposed into subsegments. **(2)** D2 shape descriptors are computed for each subsegment and a GMM is used to cluster similar subsegments. **(3)** A given grasp is transferred from one object to another by re-scaling the relevant subsegments and aligning them with Super4PCS.

dances [12]. Therefore, we propose a method for identifying these similar subsegments and transferring grasps between them rather than between full object models.

Our algorithm decomposes a set of related triangular mesh models into subsegments using a customized mesh segmentation algorithm that has been optimized for speed. It then clusters geometrically-similar subsegments into groups using D2 shape descriptors and Gaussian Mixture Models (GMMs). Finally, it transfers grasp contact points between similar subsegments with a warping method based on eigenvalue scaling and rigid registration. A diagram of this process is shown in Figure 2.

We evaluated our algorithm against a non-segmenting baseline that transfers grasps between full object meshes instead of between subsegments. Each method was used to attempt approximately 20,000 parallel-jaw grasp transfers on 80 objects from the Princeton Segmentation Benchmark [13]. We found that our algorithm improved grasp transfer success rates from 82% to 98% over the baseline, where a transfer attempt was successful if the algorithm was able to align the source and target objects closely enough to find corresponding grasp contact points on the target. Furthermore, grasps transferred with our algorithm experienced a mean loss in robustness of only 8.67% on average when compared to the original grasp – a loss which could likely be recovered with a local grasp re-planner.

II. RELATED WORK

In this paper, we draw from prior research on grasp transfer, mesh segmentation, and data-driven grasp synthesis.

A. Grasp Transfer

Prior approaches to grasp transfer have focused on point cloud feature-matching, nonlinear warping, and grasp moduli spaces as techniques for mapping a grasp from a source object to a target.

In the realm of point-cloud feature matching, Detry et al. [7], [8] transferred grasps from source point clouds to a target by volumetrically segmenting each source point cloud in the vicinity of the source grasp to produce a library of part candidates. Then, their algorithm iteratively aligned each part candidate with the target point cloud, selected the best match, and directly transferred grasp contact points by transforming the source grasp into the target's coordinates. In a similar vein, Herzog et al. [14] computed heightmap descriptors from the point clouds of objects grasped during kinesthetic demonstrations. When presented with a new scene, their algorithm finds the closest heightmap in the scene to each labeled descriptor and transfers the grasp from the best labeled descriptor into the scene.

Taking a different approach, Stouraitis, Hillenbrand, and Roa [9], [10] demonstrated a method for warping the surface geometry of a source object onto a target object by rigidly aligning the two objects, assigning correspondences between nearby sampled surface points, and interpolating those correspondences to form a continuous mapping. Their algorithm then transfers grasp contact points using the computed mapping and locally re-plans transferred grasps with perturbation sampling to achieve force closure.

In the realm of topology, Pokorny et al. [11] used spectral analysis of point clouds and grasp moduli spaces to represent objects and grasps in a common space. The authors demonstrated a gradient-based optimization technique that iteratively warps the shape of a source object toward a target object in order to achieve direct grasp transfer. Transferred grasps were shown to maintain robustness levels when compared against the original grasps.

Our algorithm draws on the "parts" library ideas of [7], [8], [14], as we group together similar object subsegments for grasp transfer. Additionally, we draw on the ideas of [9], [10] as our algorithm transfers grasp contact points with rigid alignment and contact point warping on object subsegments.

B. Mesh Segmentation

In the field of mesh segmentation, Katz et al. proposed an algorithm for iteratively decomposing a mesh into segments by clustering nearby faces with k-means, computing a "fuzzy" region of uncertainty between each pair of adjacent segments, and then computing a boundary between the segments in the fuzzy region using minimum cuts on the dual graph of the mesh [15]. Golovinskiy and Funkhouser used randomized hierarchical clustering algorithms on mesh faces to rapidly generate random mesh segmentations [16]. Most recently, Huang et al. produced a method that collects a large library of segment candidates for each object and optimizes a linear program that prioritizes choosing similarly-shaped segments for all objects [17]. Shape similarity in their algorithm is measured with the D2 shape descriptor [18], and the algorithm achieves the highest marks to date on the canonical Princeton Segmentation Benchmark [13].

Our mesh segmentation algorithm is based on Huang et al.'s work [17], but we make several simplifications to achieve higher speed at the cost of some loss in quality. Additionally, we use the D2 shape descriptor [18] as our primary measure of shape similarity and use meshes from [13] for experimental evaluation of our algorithm.

C. *Data Driven Grasp Synthesis*

A full survey of data-driven grasp synthesis techniques can be found in [1]. Much work has been done on grasping objects by viewing them as a decomposition of components [19]. For example, Miller et al. computed rough hand formations for different components of objects by fitting primitive shapes such as cylinders and cubes to each component [20]. Additionally, El-Khoury et al. demonstrated a method for identifying and grasping the handles of objects by using super-quadric approximations for components of the object [21].

More recently, Mahler et al. implemented Dex-Net 1.0, a system that samples parallel-jaw grasps on 3D mesh models and evaluates their robustness through Monte Carlo integration of physics-based metrics over uncertainty in object pose, gripper pose, and friction. [4]. This work drew on prior research in physics-based grasp analysis, including GraspIt! [22] and other similar methods.

In our work, we use Dex-Net 1.0 directly to evaluate our grasp transfer algorithm. We sample thousands of transfers and evaluate the robustness of each grasp using a metric from Dex-Net that estimates the probability of achieving force closure for a given grasp.

III. PROBLEM STATEMENT

Given a set of similar 3D meshes, our algorithm transfers grasps between them by **(1)** decomposing each mesh into subsegments, **(2)** partitioning subsegments uniquely into clusters based on shape and **(3)** transferring grasps on one subsegment to all other subsegments in its cluster (see Figure 2). This section formally defines each of these phases as a distinct problem, and the succeeding sections describe the algorithms used to solve each problem in detail.

Let $\mathcal{M} = \{M_1, M_2, \dots, M_n\}$ be a set of related polygonal meshes with vertices in $\mathbb{R}^3$. We assume that $\forall i \in \{1, \dots, n\}$, M_i is an orientable manifold with strictly triangular faces and that the faces of M_i form a single connected component.

A. *Mesh Segmentation*

Let M be a triangular mesh as described above. Let $\mathcal{F} = \{f_1, \dots, f_n\}$ be the set of M's faces, where f_i is a triangle embedded in $\mathbb{R}^3$. Now, let $G = (\mathcal{V}, \mathcal{E})$ be the undirected *dual graph* of M. Each face $f_i \in \mathcal{F}$ dualizes to a unique vertex $v_i \in \mathcal{V}$. Additionally, each edge shared by two triangular faces f_i and f_j in M dualizes to an edge $e_{ij} = (v_i, v_j) \in \mathcal{E}$.

We define a *segment* $\mathcal{S}$ of mesh M to be a subset of the vertices of its dual graph (i.e. $\mathcal{S} \subseteq \mathcal{V}$). Each segment dualizes to a set of faces that forms the segment's 3D mesh. A set of segments $\mathcal{T} = \{\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_k\}$ forms a valid *segmentation* [15] of M if and only if $\mathcal{T}$ is an exact cover of $\mathcal{V}$ and each segment $\mathcal{S}_i \in \mathcal{T}$ forms a connected component of G. Given M, a mesh segmentation algorithm seeks to compute an optimal $\mathcal{T}$ with respect to some cost function. This forms phase 1 of our algorithm as seen in Figure 2.

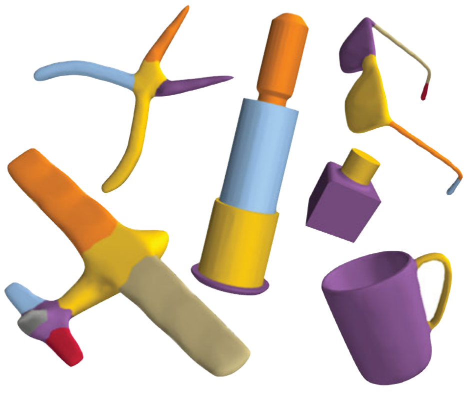
Fig. 3. Sample final segmentations computed by our algorithm. Categories listed counterclockwise from top left: plier, bearing, glasses, mech, cup, airplane.

B. *Segment Clustering*

Given a set of segments $\mathcal{I}$, we define a *cluster* $\mathcal{C}$ of $\mathcal{I}$ to be a subset of the segments in $\mathcal{I}$ (i.e. $\mathcal{C} \subseteq \mathcal{I}$). A set of clusters $\mathcal{K} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_k\}$ defines a valid *clustering* of $\mathcal{I}$ if and only if $\mathcal{K}$ is an exact cover of $\mathcal{I}$. Given $\mathcal{I}$, a segment clustering algorithm seeks to find an optimal $\mathcal{K}$ with respect to some cost function. This forms phase 2 of our algorithm as seen in Figure 2.

C. *Grasp Transfer*

Now, we formalize the grasp transfer problem. Let $\mathbf{g} = (\mathbf{x}, \mathbf{v})$ be a parallel-jaw grasp parameterized by the centroid $\mathbf{x} \in \mathbb{R}^3$ of the gripper's jaws and an axis $\mathbf{v} \in \mathbb{S}^2$ that points from one jaw to the other. We assume a robustness metric $\mathcal{R}$ that measures the quality of a grasp $\mathbf{g}$ on mesh M. Given a grasp $\mathbf{g}_i$ on a source mesh M_i, our overall goal is to find a corresponding grasp $\mathbf{g}_j$ on each target mesh $\{M_j \in \mathcal{M} \mid j \neq i\}$ that maximizes $\mathcal{R}(\mathbf{g}_j, M_j)$ under the constraint that $\mathbf{g}_j$ should be similar $\mathbf{g}_i$. This forms phase 3 of our algorithm as seen in Figure 2.

Here, we define similarity with respect to a set of segmentations $\mathcal{W} = \{\mathcal{T}_1, \dots, \mathcal{T}_n\}$ of the meshes in $\mathcal{M}$ and a clustering $\mathcal{K}$ on all segments in $\mathcal{I} = \cup_{i=1}^{n} \mathcal{T}_i$. Let $\mathcal{S}_i \in \mathcal{T}_i$ be the segment of mesh M_i that grasp $\mathbf{g}_i$ intersects, and define $\mathcal{S}_j$ analogously for grasp $\mathbf{g}_j$ on mesh M_j. Now, $\mathbf{g}_i$ and $\mathbf{g}_j$ are similar if and only if $\mathcal{S}_i$ and $\mathcal{S}_j$ are in the same cluster $\mathcal{C} \in \mathcal{K}$.

In order for a grasp transfer algorithm defined in this way to work well, we need a mesh segmentation algorithm that produces functionally relevant segments for each mesh and a clustering algorithm that places similar segments in the same cluster. Additionally, we need some way to compute the optimal positioning of $\mathbf{g}_j$ on the segments of M_j that are similar to $\mathcal{S}_i$. Our solutions to each these problems are provided in the following sections.

IV. MESH SEGMENTATION ALGORITHM

During the first part of our grasp transfer algorithm, we produce a segmentation $\mathcal{T}_i$ for each mesh $M_i \in \mathcal{M}$ such that the segments chosen for each mesh are functionally relevant.

A fair amount of research has been done on this topic, but no open-source implementations of state-of-the-art methods are available, so we implemented an algorithm that draws inspiration from Huang et al.'s joint shape segmentation algorithm [17]. Their algorithm works in two stages. First, it creates a large set of candidate segments $\mathcal{I}_i$ for each mesh in $\mathcal{M}$ using randomized cuts [16]. Then, it formulates and optimizes a linear program to select a valid segmentation from each candidate set. The program's objective has a local component that selects segments with high repetition counts in each $\mathcal{I}_i$ and a global component that selects groups of similarly-shaped segments across all meshes in the dataset.

Their algorithm produces very high-quality segmentations, but it takes between 20 and 120 minutes to segment sets of 20 meshes from the Princeton benchmark. Our algorithm seeks to reduce this run time while still producing quality segmentations by making a few key simplifications. Instead of randomly generating hundreds of segment candidates for each mesh, we deterministically generate around twenty candidates using hierarchical clustering [16] and fuzzy cuts [15]. Then, we replace the repetition count term in the linear program's objective with a weight that favors segments with concave boundary edges. Most techniques for mesh segmentation use concave edges to find good boundaries between segments [16], and the underlying algorithms that Huang et al. used to randomly generate their segment candidates are no exception. Therefore, this substitution serves as a reasonable approximation, as a more concave boundary should correspond to a higher repetition count.

With the optimizations described above, we were able to compute segmentations for any set of 20 meshes from the Princeton benchmark in under 10 minutes. Our segmentations are not as consistent as Huang et al.'s, but the decreased running time could be crucial when segmenting a large dataset of models.

Like Huang et al.'s algorithm, ours proceeds in two phases:

1) Generate a set of segment candidates for each mesh.
2) Formulate and optimize a linear program to select a final segmentation from the candidate segments for each mesh.

The details of these two phases are provided below.

A. Segment Candidate Generation

This section describes our method for generating a set of segment candidates $\mathcal{I}_i$ for each mesh $M_i \in \mathcal{M}$. First, we quickly partition M_i into a segmentation $\mathcal{T}_i$ of k initial segments with hierarchical clustering. Hierarchical clustering works quickly, but the segment boundaries it produces are not well aligned with the concave edges of M_i. In order to rectify this, we re-compute the boundaries between the initial k segments of $\mathcal{T}_i$ using a variation of the fuzzy cuts algorithm of [15]. See Figure 4 for an example of the results of an initial hierarchical clustering run followed by boundary improvement with fuzzy cuts.

Once the boundaries between the initial k segments have been refined, we initialize $\mathcal{I}_i$ with those segments and continue to use hierarchical clustering to reduce the number of segments in $\mathcal{T}_i$ to 2. At each iteration, two existing segments are merged to produce a new one, and we add the newly-created segment to $\mathcal{I}_i$. Once this run is completed, $\mathcal{I}_i$ holds $2k-2$ candidate segments for mesh M_i.

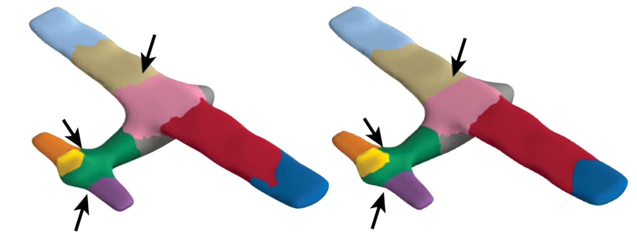
Fig. 4. Example of hierarchical segmentation **(left)** followed by fuzzy cuts **(right)** on an airplane mesh. Note how the segment boundaries move to smoothly follow concavities in the mesh after the fuzzy cuts stage.

The details of the hierarchical clustering and fuzzy cuts algorithms are described below.

1) Hierarchical Clustering: Hierarchical clustering is a greedy algorithm that segments a mesh M with n faces in $\mathcal{O}(n \log n)$ time. It takes as parameters α, which affects how strongly the concavities of the mesh are weighted, and k, a target number of segments.

Let M be a triangular mesh with dual graph $G = (\mathcal{V}, \mathcal{E})$. Then, let

$$c(v_i, v_j) = \ell_{ij} \min\Big((\theta/\pi)^{\alpha}, 1\Big) \tag{1}$$

be the edge cut cost between adjacent vertices $v_i, v_j \in \mathcal{V}$. Here, ℓ_{ij} is the edge length between the dual faces f_i and f_j of v_i and v_j and θ is the external dihedral angle between f_i and f_j. This formulation assigns low cut costs to concave edges and cut costs of 1 to convex or flat edges.

In addition, let

$$c(\mathcal{S}) = \sum_{v_i \in \mathcal{S}} \Bigg(\sum_{v_j \in \mathcal{N}(v_i) | v_j \notin \mathcal{S}} c(v_i, v_j) \Bigg) \tag{2}$$

$$\bar{c}(\mathcal{S}) = \frac{c(\mathcal{S})}{A(\mathcal{S})} \tag{3}$$

be the cut cost and area-normalized cut cost of a segment $\mathcal{S}$, respectively, where $A(\mathcal{S})$ is total surface area of the dual mesh of $\mathcal{S}$ and $\mathcal{N}(v_i) \subseteq \mathcal{V}$ is the set of vertex v_i's neighbors in G.

Hierarchical clustering works by initially assigning a unique segment to each vertex of G and then iteratively merging two adjacent segments, $\mathcal{S}_i^*$ and $\mathcal{S}_j^*$, which are chosen such that

$$\mathcal{S}_i^*, \mathcal{S}_j^* = \underset{\mathcal{S}_i, \mathcal{S}_j}{\operatorname{argmax}} \Big(\bar{c}(\mathcal{S}_i) + \bar{c}(\mathcal{S}_j) - \bar{c}(\mathcal{S}_i \cup \mathcal{S}_j) \Big) \tag{4}$$

This continues until only k segments remain.

2) Edge Refinement with Fuzzy Cuts: Given two adjacent segments $\mathcal{S}_1$ and $\mathcal{S}_2$ on a mesh M, this algorithm, which is a variant of fuzzy cuts [15], re-aligns the boundary between the segments to fit closely to concave features of the mesh M. It takes a parameter $\beta \in [0, 1]$, which controls the size of the fuzzy region in which we search for a new boundary.

Let $G = (\mathcal{V}, \mathcal{E})$ be the dual graph of M and let $\mathcal{B} \subseteq \{\mathcal{S}_1 \cup \mathcal{S}_2\}$ be the vertices on the boundary between segments $\mathcal{S}_1$ and $\mathcal{S}_2$. More formally,

$$\begin{aligned}\mathcal{B} = &\{v_i \in \mathcal{S}_1 \mid \mathcal{N}(v_i) \cap \mathcal{S}_2 \neq \emptyset\} \cup \\ &\{v_j \in \mathcal{S}_2 \mid \mathcal{N}(v_j) \cap \mathcal{S}_1 \neq \emptyset\}\end{aligned} \tag{5}$$

We first extract a "fuzzy" region in the vicinity of $\mathcal{B}$ in which we search for a new boundary between $\mathcal{S}_1$ and $\mathcal{S}_2$. Let $G_1(\mathcal{S}_1, \mathcal{E}_1)$ and $G_2(\mathcal{S}_2, \mathcal{E}_2)$ be the dual graphs of $\mathcal{S}_1$ and $\mathcal{S}_2$, respectively, with edge weights defined by Equation 1. For graph $G_i \in \{G_1, G_2\}$, we add an extra synthetic vertex b_i that represents the boundary itself. Each vertex in $\mathcal{B} \cap S_i$ is connected to b_i with an edge of zero weight. Then, we run Dijkstra's algorithm on G_i from b_i to determine the cut cost distance of each vertex $v_j \in \mathcal{S}_i$ from the boundary.

Let $d(v_j)$ be the distance of vertex v_j from b_i, and let $d_i = \max_{j \in \mathcal{S}_i} d(v_j)$. Then, the subset of $\mathcal{S}_i$ in the fuzzy region can be defined as

$$\mathcal{F}_i = \Big\{ v_j \in \mathcal{S}_i \mid d(v_j) < \beta d_i \Big\} \tag{6}$$

where $\beta \in [0, 1]$ is a parameter that determines the size of the fuzzy region.

Now, we extract the new boundary by running a minimum cuts algorithm in the fuzzy region $\mathcal{F} = \{\mathcal{F}_1 \cup \mathcal{F}_2\}$. Let G_f be the dual graph of $\mathcal{F}$. We create two new synthetic vertices, s_1 and s_2, and connect each vertex $\{v_j \in \mathcal{F}_i \mid \mathcal{N}(v_j) \cap (\mathcal{S}_i \setminus \mathcal{F}_i) \neq 0\}$ to s_i with an edge of infinite weight. Then, we compute the new boundary between $\mathcal{S}_1$ and $\mathcal{S}_2$ by computing an s-t mincut between s_1 and s_2 with [23], which runs in $\mathcal{O}(f^2 \log f)$ time, where f is the number of vertices in $\mathcal{F}$.

B. Segment Selection

Once an initial set of segment candidates is chosen for each mesh, we formulate a linear program that selects a valid segmentation $\mathcal{T}_i \subset \mathcal{I}_i$ from each candidate set. Segment candidates in $\mathcal{I}_i$ are weighted to favor segments with concave boundaries that are similar in shape to other segments in the dataset, and constraints are formulated to ensure that the resulting segmentation $\mathcal{T}_i$ is valid on M_i. The linear program is then optimized with standard solvers to obtain a final segmentation for each mesh in $\mathcal{M}$. Sample outputs from this stage of the algorithm are shown in Figures 3 and 5.

1) Shape Descriptors: In order to determine how similar two segments are in shape, we use the D2 shape descriptor [18]. The descriptor is a histogram of Euclidean distances between pairs of points that are uniformly sampled from the surface of a mesh, and it has been shown to provide good discriminative power in practice. Huang et al. [17] suggested computing three D2 descriptors for different scalings of the three principle eigenvalues of each mesh in order to factor out variations in scale between segments. The three cases are as follows:

1) Isotropic scaling of the first eigenvalue to 1.
2) Anisotropic scaling of the first two eigenvalues to 1.
3) Anisotropic scaling of all three eigenvalues to 1.

Here, we use the same distance metric as Huang et al. [17]:

$$d(\mathcal{S}, \mathcal{S}') = \left(\min_i ||\mathbf{d}_i(\mathcal{S}) - \mathbf{d}_i(\mathcal{S}')||^2 + \lambda ||\Lambda(\mathcal{S}) - \Lambda(\mathcal{S}')||^2 \right)^{\frac{1}{2}}$$

where $\mathcal{S}$ and $\mathcal{S}'$ are segments, $\mathbf{d}_i(\mathcal{S})$ is the D2 descriptor computed from case $i \in \{1, 2, 3\}$ as described above, $\Lambda(\mathcal{S})$ are the original principal eigenvalues of $\mathcal{S}$'s dual mesh, and λ is a constant that controls the penalty for differences in initial scale between the two segment meshes. All D2 descriptors were computed with 10,000 samples and 128 histogram bins, which we found to provide sufficient discriminative power.

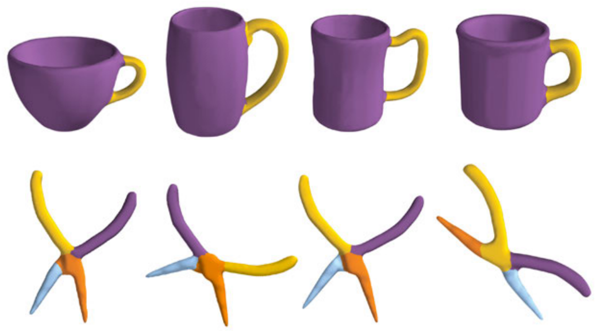

Fig. 5. Sample final segmentations for four cups and four pliers. The coloration of the cups indicates classification as well as segmentation – all handles are part of one class and all cup bodies are part of another. The colors for the pliers do not represent segment classification – the jaws and handles are sorted into two categories, and the remaining mixtures captured odd segments from a few meshes with poor face topologies.

2) Linear Program Formulation: Given a set of segment candidates $\mathcal{I}_i$ for each mesh $M_i \in \mathcal{M}$, we formulate and optimize a linear program to select a valid segmentation $\mathcal{T}_i \subset \mathcal{I}_i$ for each mesh.

For each candidate segment $\mathcal{S}_i \in \mathcal{I}_i$, we compute a weight that accounts for both the segment's local properties and its similarity to other segments in the dataset. Let $w(\mathcal{S}_i)$ be defined as

$$w(\mathcal{S}_i) = \mathcal{H}(\mathcal{S}_i) \sqrt{\frac{A(\mathcal{S}_i)}{A(M_i)}} \left(\sum_{k=1}^{n} \mathcal{D}(\mathcal{S}_i, \mathcal{S}_k^*) \mathcal{H}(\mathcal{S}_k^*) \right) \tag{7}$$

Here, $A(\mathcal{S}_i)$ is the surface area of segment $\mathcal{S}_i$'s dual mesh and $A(M_i)$ is the surface area of mesh M_i. $\mathcal{H}(\mathcal{S})$ is a weight that favors segments with concave boundaries, and it is defined as

$$\mathcal{H}(\mathcal{S}) = \exp\left(-\gamma \frac{c(\mathcal{S})}{p(\mathcal{S})\eta_r^2} \right) \tag{8}$$

where $c(\mathcal{S})$ is the cut cost for segment $\mathcal{S}$ as defined in Equation 2, $p(\mathcal{S})$ is the perimeter of $\mathcal{S}$, and $\eta_r = \gamma(\mu_r - \sigma_r)$, where μ_r is the mean perimeter-normalized cut cost across all segment candidates, σ_r is the standard deviation in perimeter-normalized cut cost, and γ is a parameter that modifies the strength of the penalty against segments with non-concave boundaries.

$\mathcal{D}(\mathcal{S},\mathcal{S}')$ is a weight that favors segments that match closely in shape, and it is defined as

$$\mathcal{D}(\mathcal{S},\mathcal{S}') = \exp\left(-\frac{d^2(\mathcal{S},\mathcal{S}')}{2\sigma_d^2}\right) \tag{9}$$

where σ_d is the median distance between the most similar segments across all shapes. In this formulation,

$$\mathcal{S}_k^* = \underset{\mathcal{S}_k \in \mathcal{I}_k}{\operatorname{argmin}}\, d(\mathcal{S}_i, \mathcal{S}_k) \tag{10}$$

is the segment from mesh M_k closest to $\mathcal{S}_i$ in shape.

Our objective function $\mathcal{U}(\{\mathcal{T}_1,\dots\mathcal{T}_n\})$ is given by

$$\mathcal{U}(\{\mathcal{T}_1,\dots\mathcal{T}_n\}) = \sum_{i=1}^{n}\sum_{\mathcal{S}_j\in\mathcal{T}_i} w(S_j) \tag{11}$$

Maximizing this objective is equivalent to finding a segmentation $\mathcal{T}_i^*$ for each mesh M_i such that

$$\mathcal{T}_i^* = \underset{\mathcal{T}_i}{\operatorname{argmax}} \sum_{S_j\in\mathcal{T}_i} w(S_j) \tag{12}$$

Let $\mathbf{w}_i$ be a column weight vector for $\mathcal{I}_i$ whose entries are the individual segment weights $w(\mathcal{S}_j), \mathcal{S}_j \in \mathcal{I}_i$, and let $\mathbf{x}_i$ be a column vector of the same dimension, where each entry $x_{ij} \in \{0,1\}$ is an indicator for whether segment $\mathcal{S}_j$ should be included in $\mathcal{T}_i^*$. Finding $\mathcal{T}_i^*$ then reduces to solving an integer program that maximizes $\mathbf{c}_i^\intercal\mathbf{x}_i$. Since integer programming is difficult, we follow Huang et al.'s example and convert the problem into a linear program by allowing each indicator x_{ij} to take on values $\in [0,1]$ [17]. This gives us the following linear program:

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}_i^\intercal\mathbf{x}_i \\ \text{subject to} \quad & \mathbf{A}_i\mathbf{x}_i = 1 \\ & \mathbf{0} \le \mathbf{x}_i \le \mathbf{1} \end{aligned}$$

In this formulation, the constraint $\mathbf{A}_i\mathbf{x}_i = 1$ is designed to ensure that the selected segments form an exact cover of M_i's dual vertices $\mathcal{V}_i$.

Once the linear program is optimized, we iteratively round the entries of $\mathbf{x}_i$. At each iteration, we snap the highest remaining indicator value to 1 and set the indicators for every segment that overlaps the most recently rounded one to 0. Once all indicators have been rounded fully, each $\mathbf{x}_i$ uniquely determines $\mathcal{T}_i^*$:

$$\mathcal{S}_j \in \mathcal{T}_i^* \iff x_{ij} = 1 \tag{13}$$

V. SEGMENT CLUSTERING

After mesh segmentation, we need to form a clustering of our dataset's segments such that similarly-shaped segments end up in the same cluster. Given the set of meshes $M = \{M_1,\dots,M_n\}$ and their corresponding segmentations $\{\mathcal{T}_1,\dots,\mathcal{T}_n\}$, we find a clustering $\mathcal{K} = \{\mathcal{C}_1,\dots,\mathcal{C}_g\}$ for $\mathcal{I} = \cup_{i=1}^n \mathcal{T}_i$ that places segments with similar shapes together by using a Gaussian Mixture Model (GMM) over the vertices of each segment's D2 descriptors.

First, we form a vector $\mathbf{v} \in \mathbb{R}^{384}$ for each segment $\mathcal{S}$ from the normalized histogram bins of $\mathcal{S}$'s three D2 descriptors. We make the assumption that each vector $\mathbf{v}$ can be modeled as a sample of a 384-dimensional Gaussian random variable, where the mean and covariance matrix for the random variable is determined by the cluster to which the segment belongs. Given the number of clusters, g, for the dataset, we assign each segment $\mathcal{S}$ to a cluster $\mathcal{C}_i$ by fitting a 384-dimensional Gaussian Mixture Model to the set of all vectors $\mathbf{v}$ in our dataset.

Determining the best g is an unsolved problem, so we set it equal to the mean number of segments per object in the dataset as an approximation. This can produce too many clusters for objects such as pliers, which have two identical handles and two identical jaws per mesh. However, we prefer to err on the side of over-estimating g, as under-estimating g will result in clusters with very different shapes grouped together and lead to poor grasp transfers.

See the cups in Figure 5 for an example of segment classification in action.

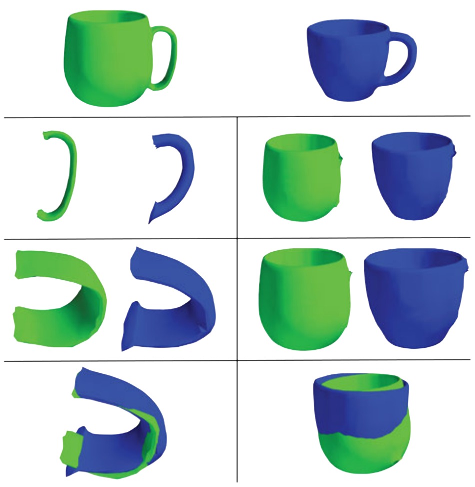
Fig. 6. Example of segmentation and alignment with Super4PCS. We start with two meshes **(top row)** and then segment them and cluster their segments **(second row)**. Each pair of similar segments is scaled anisotropically **(third row)** and aligned with Super4PCS **(bottom row)** to achieve grasp transfer.

VI. GRASP TRANSFER

Now, given the segment clustering $\mathcal{K}$ for our dataset and a parallel jaw grasp $\mathbf{g}_i$ on mesh M_i that intersects segment $\mathcal{S}_i$, we attempt to synthesize a grasp $\mathbf{g}_j$ on every other mesh $M_j \in \mathcal{M}$ such that $\mathbf{g}_j$ maximizes a robustness metric $\mathcal{R}$ under the constraint that $\mathbf{g}_j$ intersects some segment $\mathcal{S}_j$ of M_j that is in the same cluster as $\mathcal{S}_i$.

However, finding a grasp that maximizes a robustness metric over an entire mesh is difficult and usually involves extensive sampling. Therefore, as a metric-agnostic first-order approximation, we attempt to directly transfer $\mathbf{g}_i$ to $\mathcal{S}_j$ by aligning the dual meshes of $\mathcal{S}_i$ and $\mathcal{S}_j$ after an anisotropic scaling (see Figure 6 for a visualization of this process).

Let M and M' be the dual meshes of segments $\mathcal{S}_i$ and $\mathcal{S}_j$, respectively. First, we compute the contact points $\mathbf{c}_1$ and $\mathbf{c}_2$ of $\mathbf{g}_i$ on M and parameterize them by their barycentric coordinates relative to the triangular faces f_1 and f_2 they lie in. Then, we anisotropically scale all three principal eigenvalues of M and M' to one, producing new meshes $\bar{M}$ and $\bar{M}'$. This factors out scale variations in all three dimensions, which makes it easier to directly transfer grasps between segments that have similar shapes but different principal eigenvalues.

Next, we align $\bar{M}$ and $\bar{M}'$ with Super4PCS, a global point cloud alignment algorithm [24]. Minimal parameter tuning was needed, and the algorithm finds optimal alignments for any pair of meshes in a few seconds. Then, we find the transformed contact points $\bar{\mathbf{c}}_1$ and $\bar{\mathbf{c}}_2$ on $\bar{M}$ by re-applying their original barycentric coordinates to the transformed triangles $\bar{f}_1$ and $\bar{f}_2$.

The line $\overline{\bar{\mathbf{c}}_1\bar{\mathbf{c}}_2}$ is then intersected with $\bar{M}'$ to find transferred contact points $\bar{\mathbf{c}}'_1$ and $\bar{\mathbf{c}}'_2$. If they don't exist, we report a transfer failure for $\mathbf{g}_i$ on segment $\mathcal{S}_j$ and exit. Otherwise, the contact points are again parameterized by their barycentric coordinates relative to the contact triangular faces $\bar{f}'_1$ and $\bar{f}'_2$ and are transferred back to M' by re-applying their barycentric coordinates to the original triangles f'_1 and f'_2 on M'.

This process extracts contact points $\mathbf{c}'_1$ and $\mathbf{c}'_2$ on M', and the transformed grasp $\mathbf{g}_j = (\mathbf{x}_j, \mathbf{v}_j)$ can be parameterized such that $\mathbf{x}_j$ lies halfway between $\mathbf{c}'_1$ and $\mathbf{c}'_2$ and $\mathbf{v}_j$ points from $\mathbf{c}'_1$ to $\mathbf{c}'_2$. This completes the initial grasp transfer. If further refinement is required, a local grasp re-planner could be used to improve the transferred grasp as in [14], [10].

VII. EXPERIMENTS

In order to test our segmentation-based algorithm's effectiveness, we compared it against a baseline grasp transfer algorithm that uses the same eigenvalue scaling and point cloud alignment techniques as our segmentation-based algorithm, but simply transfers grasps between full 3D mesh models instead of mesh subsegments. Both algorithms were evaluated on four datasets (the cups, pliers, bearings, and glasses datasets) of twenty meshes each from [13]. We chose five meshes at random from each dataset and used Dex-Net [4] to sample fifty parallel-jaw grasp configurations on each selected mesh.

For the baseline algorithm, we attempted to transfer each grasp to every other object in the same dataset. For our segmentation-based algorithm, we first segmented each mesh and clustered similar segments within each dataset. Then, we attempted to transfer each grasp on a segment to all segments in its cluster. For all experiments, we set k to 10 (safely above the maximum number of subsegments we expected per object), α to 10 as advised by [16], λ to 0.1 as advised by [17], and γ to $\log 4$ and β to 0.25 based on experimental testing. The number of segment clusters, g, was set to the mean number of segments per object in the dataset.

The baseline algorithm produced a total of 15559 successful transfers from its 1000 initial grasps and failed to transfer 3441 grasps, for a total successful transfer rate of 82%. Our segmentation-based algorithm improved these numbers substantially, producing a total of 20194 successful transfers from the 1000 initial grasps and failing to find a transfer in 401 cases, for a total successful transfer rate of 98%. Some example grasp transfers for our algorithm are shown in Figure 1.

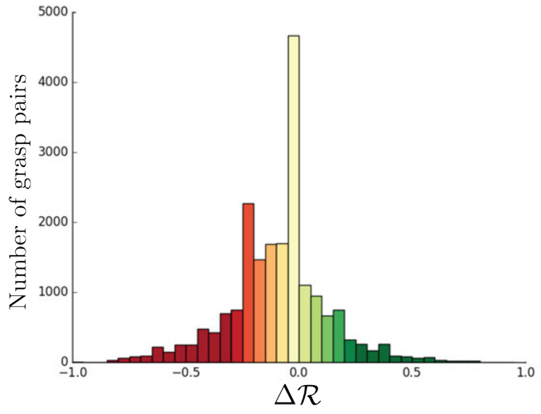

Fig. 7. Histogram of differences in robustness between original grasps and transferred grasps for our primary algorithm. For 20194 pairs of grasps, the mean $\Delta\mathcal{R}$ was -0.087 with a standard deviation of 0.217.

As expected, the non-segmenting baseline is often unable to align all components of full object models – especially for articulated objects like pliers. When the grasped component of the source object doesn't overlap the aligned target object, the grasp fails to transfer, as the algorithm cannot find contact points on the target object that match the source grasp. For example, this occurs for pliers opened to different angles whose handles cannot be simultaneously aligned. However, our segmentation-based algorithm is able to avoid most of these cases, as the object subsegments are much closer in overall shape than the full object models. For example, plier handle meshes can always be aligned closely, regardless of the relative configurations of the source and target pliers.

In addition to transfer success rates, we also evaluated the change in robustness caused by transferring a grasp. We approximated the robustness $\mathcal{R}(\mathbf{g})$ of a grasp $\mathbf{g}$ by estimating the probability of achieving force closure. We borrow from Mahler et al. [4] and assume Gaussian distributions on object pose, gripper pose, and friction coefficient, with translational variances of $\sigma = 0.005\,\mathrm{m}$, rotational variances of $\sigma = 0.1\,\mathrm{rad}$, and frictional variance of $\sigma = 0.1$, with a mean friction coefficient of 0.5. $\mathcal{R}(\mathbf{g})$ is computed by Monte Carlo integration of the force closure properties for 50 perturbed grasps sampled near $\mathbf{g}$.

For each transfer of a grasp $\mathbf{g}$ to a grasp $\mathbf{g}'$, we computed the change in robustness $\Delta\mathcal{R} = \mathcal{R}(\mathbf{g}) - \mathcal{R}(\mathbf{g}')$. A histogram of these differences for our segmentation-based algorithm is shown in Figure 7. The distribution of $\Delta\mathcal{R}$ was approximately normal for both our segmentation-based algorithm and the baseline, with a large spike of grasp pairs near $\Delta\mathcal{R} = 0$ appearing in both. The sample mean for the baseline was $\mu_{\Delta\mathcal{R}} = -0.102$ with a standard deviation of $\sigma_{\Delta\mathcal{R}} = 0.240$, while the statistics for our algorithm were $\mu_{\Delta\mathcal{R}} = -0.087$ and $\sigma_{\Delta\mathcal{R}} = 0.217$. Thus, we can say with 95% confidence that, on average, our algorithm increases ΔR by 0.015 ± 0.005 over the baseline.

Overall, our algorithm only loses around 8.7% in robustness during grasp transfer, which could partially be accounted for by local differences in mesh topology between the source and target segments. Additionally, the segmentation-based transfer algorithm ignores some global information about the target mesh that was used to compute the initial robustness metric, such as the relative locations of the target's center of mass and the transferred grasp. A grasp re-planner that looks at the target mesh's global topology and adjusts the final configuration of the transferred grasp could likely be used as in [9] to recover these small losses in robustness.

Additionally, while the baseline seems to preserve grasp robustness nearly as well as our algorithm, it fails to find a transferred grasp at all during 18% of attempts. These failed transfers cannot be re-optimized at all with a local planner, which weakens the baseline relative to our segmentation-based algorithm. Finally, the baseline does not guarantee that transferred grasps will be functionally similar at all to their original grasps, while our segmentation-based algorithm at least ensures that transferred grasps are placed on similarly-shaped subsegments across each dataset.

VIII. DISCUSSION AND FUTURE WORK

We have shown that our algorithm successfully transfers grasps between similar subsegments of 3D mesh models with only a modest loss of robustness. Additionally, we showed that transferring grasps between subsegments significantly improves transfer success rates over a baseline algorithm that transfers grasps between full mesh models.

In order to further improve our algorithm, we will examine other strategies for transferring grasps between segments and optimizing the final configuration of each grasp. For example, we could adopt the nonlinear warping and local replanning strategies of [9], [10]. In future work, we will also integrate our transfer pipeline with human-labeled meshes and methods such as PointNet [25] that can semantically label object segments, with the goal of moving towards a robust, scalable system for planning functional, task-specific grasps on a large database of objects.

ACKNOWLEDGEMENTS

This research was performed at the AUTOLAB at UC Berkeley in affiliation with the Berkeley AI Research (BAIR) Lab, the Real-Time Intelligent Secure Execution (RISE) Lab, and the CITRIS "People and Robots" (CPAR) Initiative. The authors were supported in part by the U.S. National Science Foundation under NRI Award IIS-1227536: Multilateral Manipulation by Human-Robot Collaborative Systems and the Berkeley Deep Drive (BDD) Program, the Department of Defense (DoD) through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program, and by donations from Siemens, Google, Cisco, Autodesk, and IBM. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Sponsors. In addition, we thank Carolyn Chen, Jacky Liang, Menglong Guo, and Caleb Chuck for their helpful feedback.

REFERENCES

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesisa survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[2] B. Siciliano and O. Khatib, *Springer handbook of robotics*. Springer, 2016.

[3] C. Ferrari and J. Canny, "Planning optimal grasps," in *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pp. 2290–2295, IEEE, 1992.

[4] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards,"

[5] R. Balasubramanian, L. Xu, P. D. Brook, J. R. Smith, and Y. Matsuoka, "Physical human interactive guidance: Identifying grasping principles from human-planned grasps," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 899–910, 2012.

[6] J. Weisz and P. K. Allen, "Pose error robust grasping from contact wrench space metrics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 557–562, IEEE, 2012.

[7] R. Detry, C. H. Ek, M. Madry, J. Piater, and D. Kragic, "Generalizing grasps across partly similar objects," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3791–3797, IEEE, 2012.

[8] R. Detry, C. H. Ek, M. Madry, and D. Kragic, "Learning a dictionary of prototypical grasp-predicting parts from grasping experience," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 601–608, IEEE, 2013.

[9] U. Hillenbrand and M. A. Roa, "Transferring functional grasps through contact warping and local replanning," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 2963–2970, IEEE, 2012.

[10] T. Stouraitis, U. Hillenbrand, and M. A. Roa, "Functional power grasps transferred through warping and replanning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 4933–4940, IEEE, 2015.

[11] F. T. Pokorny, Y. Bekiroglu, and D. Kragic, "Grasp moduli spaces and spherical harmonics," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 389–396, IEEE, 2014.

[12] J. Aleotti and S. Caselli, "A 3d shape segmentation approach for robot grasping by parts," *Robotics and Autonomous Systems*, vol. 60, no. 3, pp. 358–366, 2012.

[13] X. Chen, A. Golovinskiy, and T. Funkhouser, "A benchmark for 3d mesh segmentation," in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 73, ACM, 2009.

[14] A. Herzog, P. Pastor, M. Kalakrishnan, L. Righetti, J. Bohg, T. Asfour, and S. Schaal, "Learning of grasp selection based on shape-templates," *Autonomous Robots*, vol. 36, no. 1-2, pp. 51–65, 2014.

[15] S. Katz and A. Tal, *Hierarchical mesh decomposition using fuzzy clustering and cuts*, vol. 22. ACM, 2003.

[16] A. Golovinskiy and T. Funkhouser, "Randomized cuts for 3d mesh analysis," in *ACM transactions on graphics (TOG)*, vol. 27, p. 145, ACM, 2008.

[17] Q. Huang, V. Koltun, and L. Guibas, "Joint shape segmentation with linear programming," in *ACM transactions on graphics (TOG)*, vol. 30, p. 125, ACM, 2011.

[18] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Shape distributions," *ACM Transactions on Graphics (TOG)*, vol. 21, no. 4, pp. 807–832, 2002.

[19] C. Goldfeder, P. K. Allen, C. Lackner, and R. Pelossof, "Grasp planning via decomposition trees," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 4679–4684, IEEE, 2007.

[20] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, vol. 2, pp. 1824–1829, IEEE, 2003.

[21] S. El-Khoury and A. Sahbani, "Handling objects by their handles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, no. EPFL-TALK-168926, 2008.

[22] A. T. Miller and P. K. Allen, "Graspit! a versatile simulator for robotic grasping," *IEEE Robotics & Automation Magazine*, vol. 11, no. 4, pp. 110–122, 2004.

[23] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM (JACM)*, vol. 35, no. 4, pp. 921–940, 1988.

[24] N. Mellado, D. Aiger, and N. J. Mitra, "Super 4pcs fast global pointcloud registration via smart indexing," in *Computer Graphics Forum*, vol. 33, pp. 205–215, Wiley Online Library, 2014.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *arXiv preprint arXiv:1612.00593*, 2016.