A Benchmark for 3D Mesh Segmentation

Xiaobai Chen, Aleksey Golovinskiy, Thomas Funkhouser
Princeton University

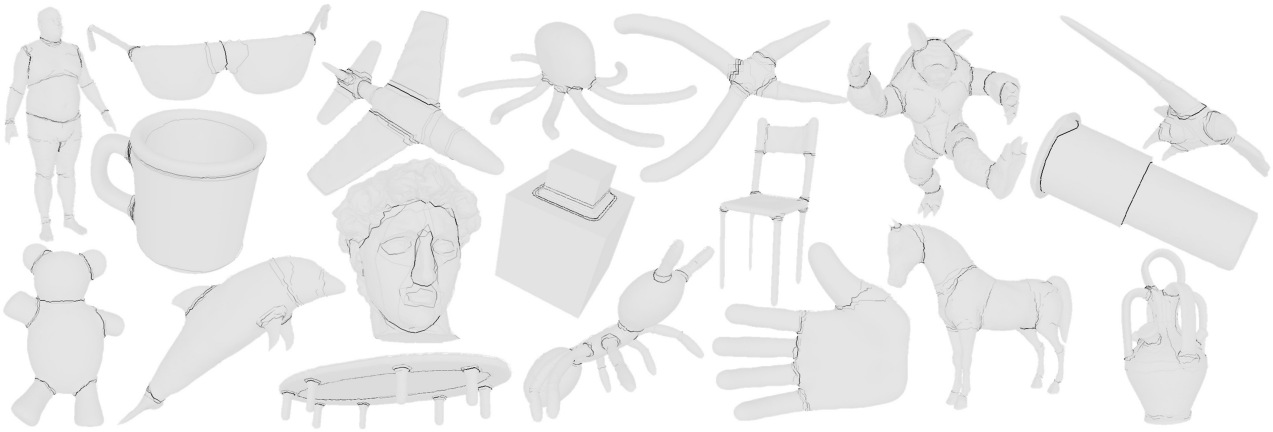


Figure 1: *Composite images of segment boundaries selected by different people (the darker the seam the more people have chosen a cut along that edge). One example is shown for each of the 19 object categories considered in this study.*

Abstract

This paper describes a benchmark for evaluation of 3D mesh segmentation algorithms. The benchmark comprises a data set with 4,300 manually generated segmentations for 380 surface meshes of 19 different object categories, and it includes software for analyzing 11 geometric properties of segmentations and producing 4 quantitative metrics for comparison of segmentations. The paper investigates the design decisions made in building the benchmark, analyzes properties of human-generated and computer-generated segmentations, and provides quantitative comparisons of 7 recently published mesh segmentation algorithms. Our results suggest that people are remarkably consistent in the way that they segment most 3D surface meshes, that no one automatic segmentation algorithm is better than the others for all types of objects, and that algorithms based on non-local shape features seem to produce segmentations that most closely resemble ones made by humans.

Keywords: 3D mesh segmentation, 3D mesh analysis

1 Introduction

Automatic segmentation of 3D surface meshes into functional parts is a fundamental problem in computer graphics. A part decomposition not only provides semantic information about the underlying object, but also can be used to guide several types of mesh processing algorithms, including skeleton extraction [Biasotti et al. 2003; Katz and Tal 2003], modeling [Funkhouser et al. 2004], morphing [Zöckler et al. 2000; Gregory et al. 1999], shape-based retrieval [Zuckerberger 2002], and texture mapping [Lévy et al. 2002]. All of these applications benefit from mesh segmentations that match human intuition.

While many automatic mesh segmentation algorithms have been developed over the last several years, there has been little work on quantitative evaluation of how well they perform. The main problem has been the lack of a "gold standard." There has been no benchmark set of 3D models and no agreed-upon metrics of success. Rather, most new algorithms are tested on their own set of 3D meshes, and results are presented only as images with different segments shown in different colors. In a few cases, more than one algorithm have been compared on the same set of meshes (e.g., [Attene et al. 2006a]), but the evaluation is qualitative (images shown side-by-side).

In this paper, we present a benchmark for quantitative evaluation of mesh segmentation algorithms. To build this benchmark, we recruited eighty people to manually segment surface meshes into functional parts, yielding an average of 11 human-generated segmentations for each of 380 meshes. This data set provides a sampled distribution over "how humans decompose each mesh into functional parts," which we treat as a probabilistic "ground truth" (Figure 1). Given this data set, we analyze properties of the human-generated segmentations to learn about what they have in common with each other (and with computer-generated segmentations); and we compute metrics that measure how well the human-generated segmentations match computer-generated ones for the same mesh, which provide a basis for quantitative comparisons of mesh segmentation algorithms.

The contributions of the paper are five-fold. First, we describe a procedure for collecting a distribution of mesh segmentations from people around the world. Second, we investigate four metrics for comparing mesh segmentations, adapting ideas from computer vision to the domain of 3D meshes. Third, we perform quantitative comparison of seven automatic mesh segmentation algorithms ac-

cording to these metrics. Fourth, we analyze properties of human-generated and computer-generated mesh segmentations, making observations about how they are similar and different with respect to one another. Finally, we provide a publicly available data set and software suite for future analysis and comparison of mesh segmentations (http://segeval.cs.princeton.edu).

2 Related Work

Segmentation is a classical problem in processing of images, video, audio, surfaces, and other types of multimedia data. Accordingly, a large number of methods have been proposed for both computing and evaluating segmentations. As methods have matured for each data type, segmentation benchmarks have been proposed for evaluation and comparison of methods [Over et al. 2004]. Examples include The Berkeley Segmentation Benchmark for images [Martin et al. 2001], The TREC Data Set for video [Smeaton et al. 2004], The IBNC Corpus for audio [Federico et al. 2000], and so on. However, to our knowledge, no analogous segmentation benchmarks have been developed for 3D surface meshes.

Over the last decade, many 3D mesh segmentation algorithms have been proposed, including ones based on K-means [Shlafman et al. 2002], graph cuts [Golovinskiy and Funkhouser 2008; Katz and Tal 2003], hierarchical clustering [Garland et al. 2001; Gelfand and Guibas 2004; Golovinskiy and Funkhouser 2008], primitive fitting [Attene et al. 2006b], random walks [Lai et al. 2008], core extraction [Katz et al. 2005], tubular multi-scale analysis [Mortara et al. 2004a], spectral clustering [Liu and Zhang 2004], critical point analysis [Lin et al. 2007] and so on (recent surveys can be found in [Agathos et al. 2007] and [Shamir 2008]). However, most of these methods have been evaluated only by visual inspection of results, and rarely are comparisons provided for more than one algorithm in the same study.

Perhaps the state-of-the-art in 3D mesh segmentation evaluation is the study of Attene et al. [Attene et al. 2006a]. They compared five segmentation algorithms [Attene et al. 2006b; Katz et al. 2005; Katz and Tal 2003; Mortara et al. 2004a; Mortara et al. 2004b] on eleven 3D surface meshes. Evaluation and comparison was performed by showing side-by-side images of segmented meshes produced by different algorithms with segments shown in different colors. Since no "ground truth" was available, readers were asked to evaluate results visually, and discussion focused on qualitative characteristics of algorithms (e.g., segmentation type, suitable inputs, boundary smoothness, whether the method is hierarchical, sensitivity to pose, computational complexity, and control parameters).

The focus of our work is to provide quantitative metrics for evaluation of mesh segmentation algorithms. We would like to not only show images of example segmentations, but also provide statistical measures of "how close they are to what a human would do" for a large set of 3D models. In this sense, we follow the work of LabelMe [Russell et al. 2008], The Berkeley Segmentation Benchmark [Martin et al. 2001], and other image databases that have built sets of human-generated segmentations and then used them for evaluation of automatic algorithms [Correia and Pereira 2000; Everingham et al. 2002; Huang and Dom 1995; Unnikrishnan et al. 2005; Zhang 1996; Zhang et al. 2008]. We leverage the ideas of this previous work in computer vision, using related metrics to evaluate the similarities in overall structure and boundaries of automatically-generated segmentations with respect to human-generated ones.

Concurrent with our work is a project by [Benhabiles et al. 2009], which has similar goals and methodology. Our study includes more human subjects, segmentation algorithms, mesh models, object categories, evaluation metrics, and analysis of human-generated segmentations.

3 Benchmark Design

The focus of this paper is to investigate the design of a benchmark for 3D mesh segmentation algorithms. In building such a benchmark, several issues have to be addressed, including "which 3D models to study?," "how to acquire unbiased segmentations from humans?," "how to acquire data from a large variety of people?," "how to measure the similarity between two segmentations?," and "how to present aggregate statistics when comparing the results of different algorithms?." These topics are discussed in the following five subsections.

3.1 Selecting a 3D Model Data Set

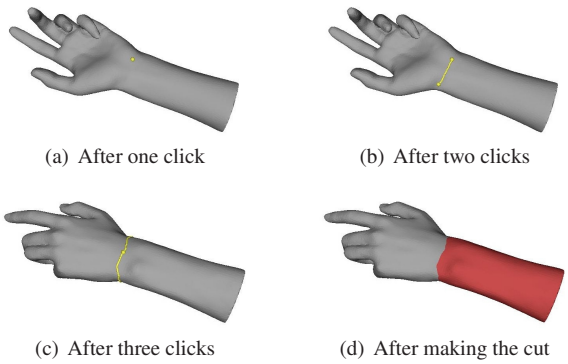
The first issue is to select a set of 3D models to include in the segmentation benchmark. In making our selection, we consider four main criteria: 1) the data set should cover a broad set of object categories (e.g., furniture, animals, tools, etc.); 2) models should exhibit shape variability within each category (e.g., articulated figures in different poses); 3) models should be neither too simple, nor too complex, to segment; and, 4) models should be represented in a form that most segmentation algorithms are able to take as input. This last criterion suggests that the models should be represented by manifold triangle meshes with only one (or a small number of) connected component(s).

While several data sets fit these criteria, and any of them could have been selected for our study, we chose to work with the 3D meshes from the Watertight Track of the 2007 SHREC Shape-based Retrieval Contest provided generously by Daniela Giorgi [Giorgi et al. 2007]. This data set contains 400 models spread evenly amongst 20 object categories, including human bodies, chairs, mechanical CAD parts, hands, fish, pliers, etc. (one example from most categories is shown in Figure 1). It contains two categories where the same object appears in different poses (armadillo and pliers), another seven categories where different articulated objects of the same type appear in different poses (human, ant, octopus, teddy bear, hand, bird, four leg), and three categories where objects have non-zero genus (cup, vase, chair). Several object categories are fairly easy to segment (teddy, bearing), while others are very challenging (e.g., buste). Most importantly, every object is represented by a watertight triangle mesh with a single connected component, the representation supported most commonly by currently available segmentation algorithms.

While this data set provides a good starting point for a benchmark, it also has some drawbacks. First, every model was remeshed during the process of making it watertight (e.g., the original polygonal model was voxelized, and then an isosurface was extracted). As a result, the sampling of vertices in the mesh is more uniform and noisier than is typical in computer graphics models. Second, it contains a category called Spring, for which segmentation does not really make sense (each object is a single coil of wire), and thus we excluded it from our study. Over time, we expect the benchmark to include other data sets with different advantages and drawbacks that can be used to test different aspects of algorithms. For example, we have produced a segmented version of the Princeton Shape Benchmark [Shilane et al. 2004]. However, many of its models contain mesh degeneracies, and thus it is not suitable for this study.

3.2 Designing an Interactive Segmentation Tool

The second issue is to design a system for collecting example segmentations from humans. Since the choice of segments/boundaries is subjective, and there are always variances among people's opinions, we needed a mechanism with which we can acquire segmentations from many different people for each polygonal model in the



(a) After one click (b) After two clicks

(c) After three clicks (d) After making the cut

Figure 2: *Using the Interactive Segmentation Tool.*

data set. The tool used to acquire segmentations should be easy to learn, to enable collection of data from untrained subjects; it should be quick to use (a couple of minutes per model), since thousands of examples are needed to cover the 380 models each with multiple examples; and, most importantly, it should not bias people towards particular types of segmentations – i.e., the results should reflect "what segmentation a person wants," not "what segmentation is possible with the tool."

There are many possible approaches to interactive mesh segmentation. For example, Shape Annotator [Robbiano et al. 2007] supports an interactive interface for combining segments generated by different automatic segmentation algorithms, and thus provides a solution that is easy to learn and quick to use. However, since the results are limited to contain only segments generated by automatic algorithms, they are highly constrained and biased towards those algorithms. Alternatively, several system are available that support intelligent scissoring of 3D meshes, using interfaces based on either strokes along cuts [Funkhouser et al. 2004; Lee et al. 2004] or sketches within segments [Wu et al. 2007]. However, then the results would be biased towards the cuts favored by the system's "intelligent" scissoring error function.

Our approach is to develop a simple interactive tool with which people can select points along segment boundaries (cuts) with a simple point and click interface, while the system connects them with shortest paths [Gregory et al. 1999] (Figure 2). The user builds and adjusts the current cut incrementally by clicking on vertices through which the cut should traverse. For each click, the system updates the (approximately) shortest closed path through the selected vertices along edges of the mesh (shown in yellow in Figure 2). Vertices can be inserted into the middle of the cut or removed from it, and cuts can be defined with multiple contours (e.g., to cut off the handle of a cup). Whenever the user is satisfied with the current cut, he/she hits a key to partition the mesh, after which the faces on either side of the cut are shown in different colors. The segmentation process proceeds hierarchically as each cut makes a binary split.

This approach strikes a balance between user-control and ease-of-use. While the system provides some assistance to the user when defining cuts (it connects selected vertices with shortest paths along edges), it is significantly less than the assistance provided by intelligent scissoring programs (which also encourage cuts to follow concave seams as defined by an error function) and significantly more than a brute-force interface where the user selects every vertex along every cut. In the former case, the system is very easy-to-use, but the results are biased. In the latter case, the system is impractically tedious to use, but gives the user absolute control over the placement of cuts. Our approach is somewhere in the middle. Specifying a desired cut often takes as few as three clicks, but can take more than ten for complicated cuts. Yet, we expect that connecting the user's input with shortest paths does not unfairly bias the results towards any particular segmentation algorithm.

3.3 Collecting Segmentations from Many People

The third issue is to design a procedure to acquire segmentations from multiple people for each 3D model in the data set.

The most obvious way to address this issue is to pay a group of people to come to our lab and run our interactive segmentation program under close supervision. However, this procedure is very costly, in both time and money. So, instead we recruited people to segment models through Amazon's Mechanical Turk [Amazon 2008]. The Mechanical Turk is an on-line platform that matches workers from around the world with paid tasks. Requesters (us) can post tasks (hits) to a web site (www.mturk.com), where workers can select them from a list, follow the instructions, and submit the results online. The big advantage of this approach is that it taps the inexpensive and under-utilized resource of people with idle time around the world. The challenge is that we must provide instructions and quality assurance procedures that minimize the risk that acquired data is biased or poor quality.

To minimize bias, we distribute meshes to workers in random order, and we log the IP of people who access our task and combine it with the worker information from the Mechanical Turk to guarantee that each worker segments each model at most once. We also provide very little information about how models should be segmented. Quoting from the web page, we tell the workers: "Our Motivation is to create a 3D mesh segmentation benchmark that helps researchers to better understand mesh segmentation algorithms. Our task is to use an interactive program to manually segment the 3D polygonal model into functional parts. We evaluate manually segmented 3D polygonal models by comparison to segmentations provided by experts. They should cut the surface along natural seams into functional parts with as much as detail as possible."

We also have a simple set of instructions to describe how to use the interactive segmentation tool: "An interactive tool for segmenting 3D meshes will start as soon as you accept our task. The program provides several commands (e.g., 'a') for creating and editing a path through vertices of the mesh and a command ('c') for cutting the mesh into two segments by separating along the current path. Each 'cutting path' partitions a single segment into two, and thus these simple commands can be applied in sequence to decompose a mesh into multiple segments. The segmentation result will be saved when the program exits (hit the ESCAPE key to quit the program). Full list of commands are listed in the tab called 'Tool Command Handbook.'"

Using this procedure, we received 4,365 submissions during a one month time period, of which 4,300 were accepted and 65 were rejected (for having no cuts at all). In looking at the results, we find that 25 are over-segmented in comparison to the others, and 353 have at least one cut that seems to be an outlier. However, we do not exclude this data from our analysis so as to avoid bias in the results. As shown in Figure 3, we accepted an average of 11 segmentations for each model. According to Amazon's Mechanical Turk statistics, the accepted submissions came from more than 80 participants, and each took an average of 3 minutes of worker time.

3.4 Computing Evaluation Metrics

The next design decision is to develop a set of metrics that can be used to evaluate how well computer-generated segmentations match the human-generated ones. Here, we follow prior work in computer vision, adapting four metrics that have previously been

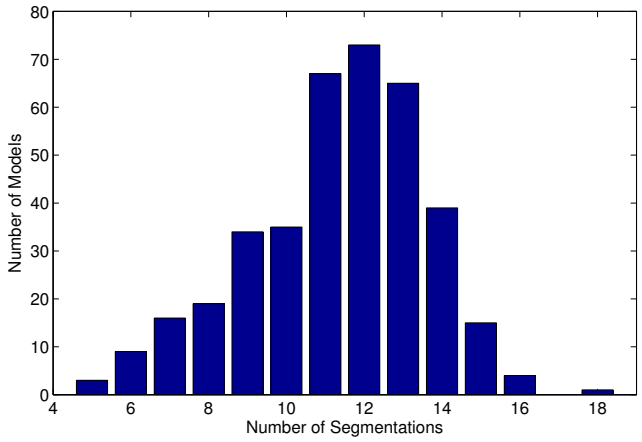


Figure 3: *Histogram of the number of segmentations per model.*

used to evaluate image segmentations. Generally speaking, the first one is boundary-based (i.e., measures how close segment boundaries are to one another), and the last three are region-based (i.e., measure the consistency of segment interiors). Among those, one explicitly considers segment correspondences in its error calculations, and another accounts for hierarchical nesting. Rather than selecting one of these metrics, we present results for all four, as each may provide different information that could be valuable in our study.

3.4.1 Cut Discrepancy

The first metric, *Cut Discrepancy*, sums the distances from points along the cuts in the computed segmentation to the closest cuts in the ground truth segmentation, and vice-versa. Intuitively, it is a boundary-based method that measures the distances between cuts [Huang and Dom 1995].

Assuming C_1 and C_2 are sets of all points on the segment boundaries of segmentations S_1 and S_2, respectively, and $d_G(p_1, p_2)$ measures the geodesic distance between two points on a mesh, then the geodesic distance from a point $p_1 \in C_1$ to a set of cuts C_2 is defined as:

$$d_G(p_1, C_2) = \min\{d_G(p_1, p_2), \forall p_2 \in C_2\}$$

and the Directional Cut Discrepancy, $\mathrm{DCD}(S_1 \Rightarrow S_2)$, of S_1 with respect to S_2 is defined as the mean of the distribution of $d_G(p_1, C_2)$ for all points $p_1 \in C_1$:

$$\mathrm{DCD}(S_1 \Rightarrow S_2) = mean\{d_G(p_1, C_2), \forall p_1 \in C_1\}$$

We define the Cut Discrepancy, $\mathrm{CD}(S_1, S_2)$, to be the mean of the directional functions in both directions, divided by the average Euclidean distance from a point on the surface to centroid of the mesh (avgRadius) in order to ensure symmetry of the metric and to avoid effects due to scale:

$$\mathrm{CD}(S_1, S_2) = \frac{\mathrm{DCD}(S_1 \Rightarrow S_2) + \mathrm{DCD}(S_2 \Rightarrow S_1)}{\mathrm{avgRadius}}$$

The advantage of the Cut Discrepancy metric is that it provides a simple, intuitive measure of how well boundaries align. The disadvantage is that it is sensitive to segmentation granularity. In particular, it is undefined when either model has zero cuts, and it decreases to zero as more cuts are added to the ground truth segmentation.

3.4.2 Hamming Distance

A second metric uses *Hamming Distance* to measure the overall region-based difference between two segmentation results [Huang and Dom 1995]. Given two mesh segmentation $S_1 = \{S_1^1, S_1^2, \ldots, S_1^m\}$ and $S_2 = \{S_2^1, S_2^2, \ldots, S_2^n\}$ with m and n segments, respectively, the *Directional Hamming Distance* is defined as

$$D_H(S_1 \Rightarrow S_2) = \sum_i \|S_2^i \backslash S_1^{i_t}\|$$

where "\" is the set difference operator, $\|x\|$ is a measure for set x (e.g., the size of set x, or the total area of all faces in a face set), and $i_t = \max_k \|S_2^i \cap S_1^k\|$. The general idea is to find a best corresponding segment in S_1 for each segment in S_2, and sum up the set difference. Since there are usually few segments per model, we use a brute force $O(N + mn)$ algorithm to find the correspondences.

If S_2 is regarded as the ground truth, then Directional Hamming Distance can be used to define the missing rate R_m and false alarm rate R_f as follows:

$$R_m(S_1, S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{\|S\|}$$
$$R_f(S_1, S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{\|S\|}$$

where $\|S\|$ is the total surface area of the polygonal model. The *Hamming Distance* is simply defined as the average of missing rate and false alarm rate:

$$\mathrm{HD}(S_1, S_2) = \frac{1}{2}(R_m(S_1, S_2) + R_f(S_1, S_2))$$

Since $R_m(S_1, S_2) = R_f(S_2, S_1)$, the Hamming Distance is symmetric. Its main advantage and disadvantage is that it relies upon finding correspondences between segments. This process provides a more meaningful evaluation metric when correspondences are "correct," but adds noise to the metric when they are not. It is also somewhat sensitive to differences in segmentation granularity.

3.4.3 Rand Index

A third metric measures the likelihood that a pair of faces are either in the same segment in two segmentations, or in different segments in both segmentations [Rand 1971].

If we denote S_1 and S_2 as two segmentations, s_i^1 and s_i^2 as the segment IDs of face i in S_1 and S_2, and N as the number of faces in the polygonal mesh, $C_{ij} = 1$ iff $s_i^1 = s_j^1$, and $P_{ij} = 1$ iff $s_i^2 = s_j^2$, then we can define *Rand Index* as:

$$\mathrm{RI}(S_1, S_2) = \binom{2}{N}^{-1} \sum_{i,j,i<j} [C_{ij}P_{ij} + (1 - C_{ij})(1 - P_{ij})]$$

$C_{ij}P_{ij} = 1$ indicates that face i and j have the same id in both S_1 and S_2. $(1 - C_{ij})(1 - P_{ij}) = 1$ indicates that face i and j have different IDs in both S_1 and S_2. Thus $\mathrm{RI}(S_1, S_2)$ tells the proportion of face pairs that agree or disagree jointly on their segment group identities in segmentation S_1 and S_2. As a slight departure from standard practice, we report $1 - \mathrm{RI}(S_1, S_2)$ in our benchmark to be consistent with the our other metrics that report dissimilarities rather than similarities (the lower the number, the better the segmentation result).

The main advantage of this metric is that it models area overlaps of segments without having to find segment correspondences.

3.4.4 Consistency Error

The fourth metric, *Consistency Error*, tries to account for nested, hierarchical similarities and differences in segmentations [Martin et al. 2001]. Based on the theory that human's perceptual organization imposes a hierarchical tree structure on objects, Martin et al. proposed a region-based consistency error metric that does not penalize differences in hierarchical granularity [Martin et al. 2001].

Denoting S_1 and S_2 as two segmentation results for a model, t_i as a mesh face, "\" as the set difference operator, and $\|x\|$ as a measure for set x (as in Section 3.4.2), $R(S, f_i)$ as the segment (a set of connected faces) in segmentation S that contains face f_i, and n as the number of faces in the polygonal model, the local refinement error is defined as:

$$E(S_1, S_2, f_i) = \frac{\|R(S_1, f_i) \backslash R(S_2, f_i)\|}{\|R(S_1, f_i)\|}$$

Given the refinement error for each face, two metrics are defined for the entire 3D mesh, *Global Consistency Error* (GCE) and *Local Consistency Error* (LCE), as follows:

$$GCE(S_1, S_2) = \frac{1}{n} \min\{\sum_i E(S_1, S_2, f_i), \sum_i E(S_2, S_1, f_i)\}$$

$$LCE(S_1, S_2) = \frac{1}{n} \sum_i \min\{E(S_1, S_2, f_i), E(S_2, S_1, f_i)\}$$

Both GCE and LCE are symmetric. The difference between them is that GCE forces all local refinements to be in the same direction, while LCE allows refinement in different directions in different parts of the 3D model. As a result, $GCE(S_1, S_2) \geq LCE(S_1, S_2)$. The advantage of these metrics are that they account for nested, hierarchical differences in segmentations. The disadvantage is that they tend to provide better scores when two models have different numbers of segmentations, and they can actually be misleading when either model is grossly under- or over-segmented with respect to the other. For example, the errors will always be zero if one of the meshes is not segmented at all (all faces in one segment) or if every face is in a different segment, since then one segmentation will always be a nested refinement of the other.

3.5 Comparing Segmentation Algorithms

The final design decision in defining the benchmark is to describe a protocol with which segmentation algorithms should be executed to produce segmentations suitable for fair comparisons.

The first issue is how to set parameters. Almost every algorithm provides tunable parameters (e.g., weights on error terms, spacing between seeds, thresholds on segment size, etc.), which sometimes affect the output segmentations dramatically. However, it would be unrealistic to search for an optimal set of parameter settings for each model; and, even if we could do that, the process would be unfair to the algorithms with fewer parameters. Hence, we utilize a single set of parameter settings for all runs of each algorithm, limiting our evaluation to the parameter settings recommended by the authors.

The one exception is for algorithms that require the target number of segments as an input parameter ($numSegments$). The problem is to select a value for this parameter that does not provide an unfair advantage or disadvantage for these algorithms with respect to algorithms that do not take such a parameter (i.e., ones that predict the number of segments automatically). One extreme solution would be to run each algorithm once for each model with the target number of segments set to some arbitrary value, say 7. Of course, this would unfairly penalize the algorithms that rely upon this parameter, since typical users would probably have a better way to select the target number of segments based on visual inspection of the model or its object category. Another extreme solution would be to run each algorithm repeatedly, once for each human-generated segmentation H to which it is being compared, each time setting *numSegments* to match the number of segments in H exactly. Of course, this would unfairly benefit the algorithms that take *numSegments* as a parameter – they would be given an oracle to predict the number of segments created by every individual person. Our solution is somewhere in the middle. We imagine that a typical user can look at a model and guess the number of target segments approximately. So, modeling a scenario where a user looks at a model and then runs the segmentation algorithm, we choose a separate value for *numSegments* for each model, setting it to be the mode of the number of segments appearing in segmentations created by people for that model in the benchmark data set (this design decision is evaluated in Section 4.6).

The second issue is remeshing. Some segmentation algorithms change the topology of the mesh. Yet, most of our evaluation metrics rely upon meshes having the same topology. To overcome this problem, we provide a step where the segment labels of the modified mesh are mapped back to the input mesh (this mapping employs an area-weighted voting scheme to assign a face to a segment if it is split into two in the output, and it eliminates segments with fewer than 3 faces or less than 0.01% of the total surface area). This process avoids penalizing algorithms for changing the topology of the mesh.

The final issue we must address is how to aggregate evaluation metrics over multiple segmentations for the same model and multiple models for the same object category. For now, we simply report averages, both per category and over the entire data set for each algorithm (averages are computed first within each model, then those results are averaged within each object category, and then those results are averaged across the entire database to avoid biasing the results towards models and/or categories with more segmentations in the benchmark data set). However, more sophisticated aggregate statistics could be computed in the future.

4 Results

In order to investigate the utility of the proposed benchmark for evaluating segmentations, we performed six experiments. The first asks "How consistently do different people segment the same object?" The second two study how effectively the benchmark can be used to compare the performance of different segmentation algorithms, asking questions like: "Which algorithms produce outputs that most closely mimic human segmentations?," and "Does any algorithm provide the best results for every type of object?" The next two investigate properties of human- and computer-generated segmentations, asking "which geometric properties are consistent in all segmentations by people" and "how do the properties of automatic segmentations compare to humans'?." The final study evaluates how our method for selecting the number of target segments for each model affects the comparison of algorithms.

While the results of these studies may be interesting in their own right, our main goal is to investigate whether the data set and evaluation procedures described in the preceding sections are useful for understanding similarities and differences between segmentations. It is our intention that other analyses and comparisons will be performed in the future as the benchmark grows as a public-domain resource for the research community.

4.1 Consistency of Human-Generated Segmentations

The benchmark provides segmentations made manually by multiple people for each of 380 models from a variety of different object categories, and thus it provides an opportunity to study how consistently different people segment the same objects. This question is not only important for its obvious implications in perceptual psychology, but also for its implications on our benchmark. Implicit in our choice to use human-generated segmentations as "ground truth" for our benchmark are the assumptions that people tend to segment objects in the same ways and that automatic algorithms ought to mimic them. We can validate the first of these assumptions with visual and statistical analysis of the benchmark data set.

Figures 1 and 4 show visualizations of the segmentations collected for a sampling of objects in the benchmark data set. These images show composites of cuts made by different people, and so darker lines represent edges where people placed segmentation boundaries more consistently. For example, in the image of the human body on the left of Figure 4, it can be seen that people cut very consistently on edges that separate the arms from the body, but less consistently on edges that separate the chest from the stomach. In general, by visual inspection, it seems that people select the same types of functional parts, possibly cutting on edges that are parallel and slightly offset from one another (e.g., at the junctions of the fingers and the palm of the hands), and possibly decomposing objects with different levels of granularity (e.g., only some people cut at the knuckles in the two images of hands in Figure 4). Although a few exceptions exist (e.g., in the eyeglasses shown in the top-left image of Figure 4, people agree that the two lenses should be separated, but they disagree whether the bridge between two lenses should be segmented as an independent part), we conclude that people are remarkably consistent in their decompositions into parts.

This conclusion can be validated quantitatively using the evaluation metrics described in Section 3.4. For each segmentation in the benchmark, we can "hold it out" and then ask how well it matches the segmentations made by other people for the same model according to the four evaluation metrics. The results of this analysis are shown in Figure 5 – the bar labeled "Human" on the left side of each of the four plots shows the averages of the human segmentations evaluated with respect to all the other human segmentations. While it is difficult to make conclusions from the absolute heights of the bars in these plots, the bars labeled "Human" are clearly lower (better) than the corresponding bars for computer-generated segmentations, suggesting that people are more consistent with each other than algorithms are with people.

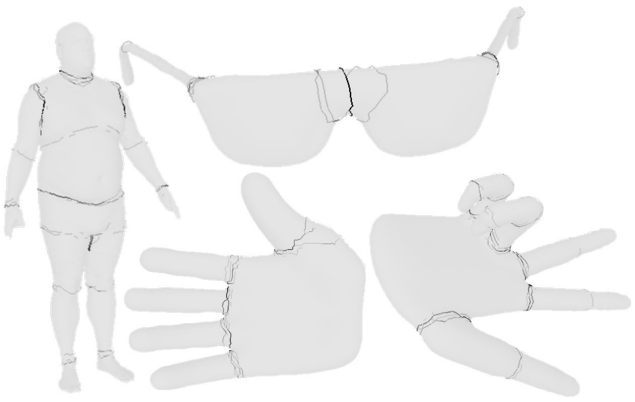


Figure 4: *Composited images of segmentation boundaries selected by multiple people for four example models (darker lines appear where a larger fraction of people placed a segmentation boundary).*

4.2 Comparison of Segmentation Algorithms

The main purpose of the benchmark is to evaluate and compare the results of automatic segmentation algorithms. To test its utility for this task, we followed the procedure described in Section 3.5 to compare seven algorithms recently published in the literature (listed below). In every case, except one (K-Means), the source code and/or executable was provided directly by the original inventors, and so each algorithm was tested with its original implementation and with its default parameters. The following paragraphs provide a brief description of each algorithm, but please refer to the cited papers for details:

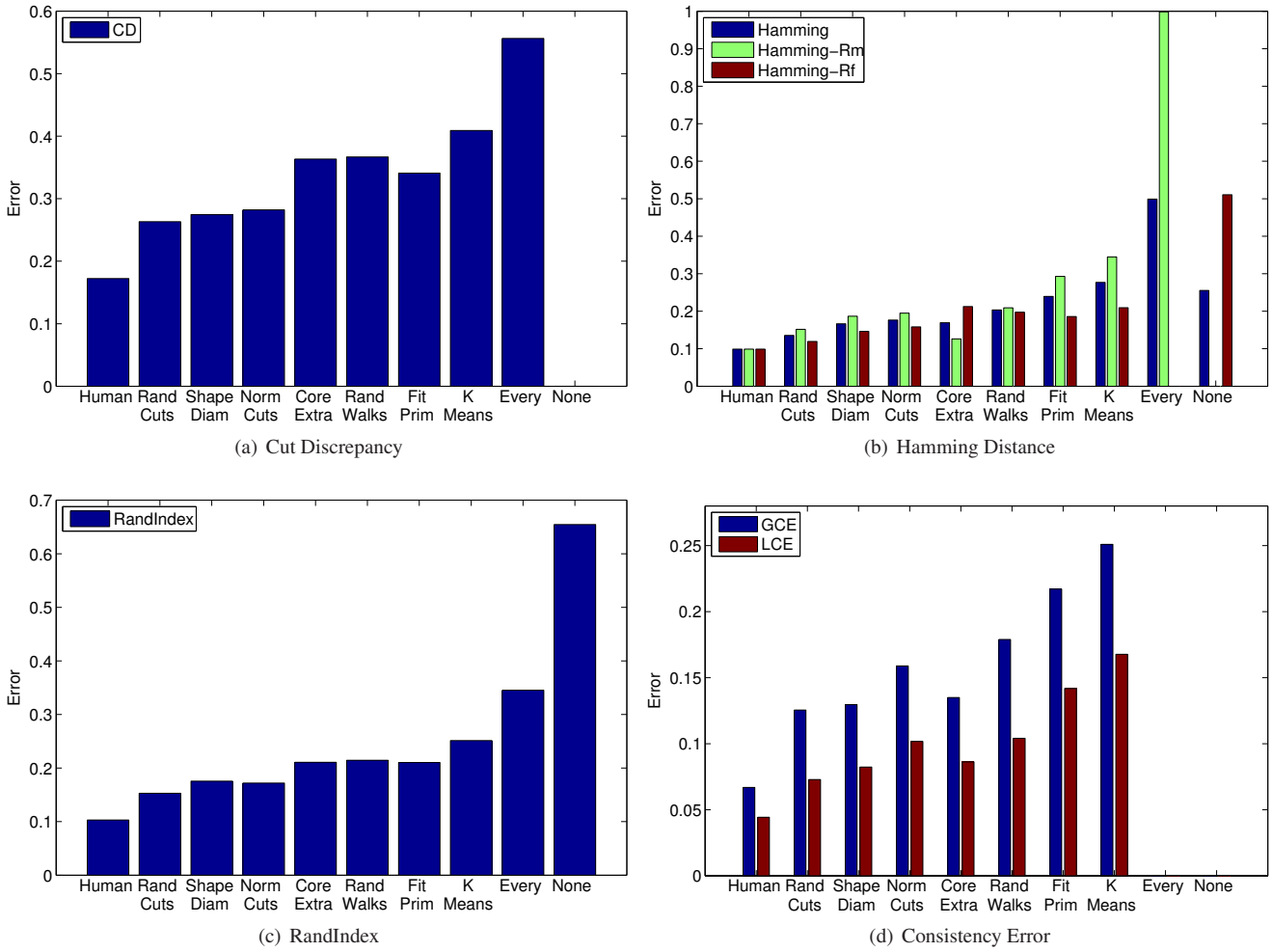
- **K-Means:** Shlafman et al. [Shlafman et al. 2002] describe an algorithm based on K-means clustering of faces. Given a user-specified number of segments, k, the algorithm first selects a set of k seed faces to represent clusters by continually selecting the further face from any previously selected. Then, it iterates between: 1) assigning all faces to the cluster with the closest representative seed, and 2) adjusting the seed of each cluster to lie at the center of the faces assigned to it. This iteration continues until the assignment of faces to clusters converges. In our implementation, which differs from the original by Shlafman et al., we compute distances on the dual graph of the mesh with a penalty related to the dihedral angle of each traversed edge using the method in [Funkhouser et al. 2004].

- **Random walks:** Lai et al. [Lai et al. 2008] describe a procedure that proceeds in two phases. During the first phase, an over-segmentation is computed by assigning each face F to the segmented associated with the seed face that has highest probability of reaching F by a random walk on the dual graph of the mesh. During the second phase, segments are merged hierarchically in an order based on the relative lengths of the intersections and total perimeters of adjacent segments. The hierarchical clustering terminates when a user-specified number of segments has been reached.

- **Fitting Primitives:** Attene et al. [Attene et al. 2006b] propose a hierarchical clustering algorithm based on fitting primitives. The algorithm starts with every face in a separate segment. At each iteration, the best fitting geometric primitive (plane, cylinder, or sphere) is computed to approximate the faces in every pair of adjacent segments, and the best fitting pair is chosen for merger. The algorithm proceeds bottom-up, merging segments until a user-specified number of segments has been reached.

- **Normalized cuts:** Golovinskiy et al. [Golovinskiy and Funkhouser 2008] describe a hierarchical clustering algorithm in which every face of the mesh starts in its own segment, and segments are hierarchically merged in an order determined by the area-normalized cut cost: the sum of each segment's perimeter (weighed by concavity) divided by its area. The clustering terminates when a user-specified number of segments has been reached. This algorithm encourages segments to have small boundaries along concave seams while maintaining segments with roughly similar areas.

- **Randomized cuts:** Golovinskiy et al. [Golovinskiy and Funkhouser 2008] also propose a hierarchical decomposition procedure that uses a set of randomized minimum cuts to guide placement of segmentation boundaries. They first decimate the mesh (to 2,000 triangles in this study), and then proceed top-down hierarchically, starting with all faces in a single segment and iteratively making binary splits. For each split, they compute a set of randomized cuts for each segment, and



(a) Cut Discrepancy (b) Hamming Distance

(c) RandIndex (d) Consistency Error

Figure 5: *Comparison of segmentation algorithms with four evaluation metrics.*

then they identify for each segment which cut is most consistent with others in the randomized set. Amongst this set of candidate cuts, they choose the one that results in the minimal normalized cut cost. The algorithm terminates when a user-specified number of segments has been reached.

- **Core extraction:** Katz et al. [Katz et al. 2005] propose a hierarchical decomposition algorithm that performs four main steps at each stage: transformation of the mesh vertices into a pose insensitive representation using multidimensional scaling, extraction of prominent feature points, extraction of core components using spherical mirroring, and refinement of boundaries to follow the natural seams of the mesh. The algorithm partitions segments hierarchically, stopping automatically when the current segment S^i has no feature points or when the fraction of vertices contained in the convex hull is above a threshold.

- **Shape Diameter Function:** Shapira et al. [Shapira et al. 2008] describe an algorithm based on the "Shape Diameter Function" (SDF), a measure of the diameter of an object's volume in the neighborhood of a point on the surface. The SDF is computed for the centroid of every face, and then the segmentation proceeds in two steps. First, a Gaussian Mixture Model is used to fit k Gaussians to the histogram of all SDF values in order to produce a vector of length k for each face indicating its probability to be assigned to each of the SDF clusters. Second, the alpha expansion graph-cut algorithm is used to refine the segmentation to minimize an energy function that combines the probabilistic vectors from step one along with boundary smoothness and concaveness. The algorithm proceeds hierarchically for a given number of "partitioning candidates," which determines the output number of segments. We follow the authors' advice, setting the number of partitioning candidates to 5, and let the algorithm determine the number of segments automatically.

Figure 5 shows evaluations of these seven algorithms according to the proposed benchmark. Each of the four bar charts shows a different evaluation metric computed for all seven algorithms and averaged across the entire data set as described in Section 3.5. Added to each chart is a bar representing the human-generated segmentations evaluated with respect to one other (labeled "Human," on the left) and two bars representing trivial segmentation algorithms ("None" = all faces in one segment, and "Every"= every face in a separate segment). These three bars are added to the plots to provide sanity checks and to establish a working range for the values of each evaluation metric. In all cases, lower bars represent better results.

Segmentation Algorithm	Avg Compute Time (s)	Rand Index
Human	-	0.103
Randomized Cuts	83.8	0.152
Shape Diameter	8.9	0.175
Normalized Cuts	49.4	0.172
Core Extraction	19.5	0.210
Random Walks	1.4	0.214
Fitting Primitives	4.6	0.210
K-Means	2.5	0.251
None	0	0.345
Every	0	0.654

Table 1: *Analysis of trade-off between compute time and segmentation quality (compute time is in seconds measured on a 2Ghz PC).*

From this data, we see that all four evaluation metrics are remarkably consistent with one another. Although one of the metrics focuses on boundary errors (Cut Discrepancy), and the other three focus on region dissimilarities (Hamming Distance, Rand Index, and Consistency Error), all variants of all four metrics suggest almost the same relative performance of the seven algorithms.

Examining the average timing and performance results in Table 1, we see that the set of algorithms that perform best overall according to the Rand Index metric (Randomized cuts, Shape Diameter Function, Normalized cuts, and Core Extraction) are the ones that take the most time to compute.[1] This result is not surprising. They are the algorithms that consider both boundary and region properties of segments and utilize non-local shape properties to guide segmentation. For example, the algorithms based on Core Extraction and the Shape Diameter Function both consider volumetric properties of the object in the vicinity of a point, and the Randomized Cut and Normalized Cut algorithms both evaluate expensive objective functions that take into account large-scale properties of segment boundaries and areas. These four algorithms should probably be used for off-line computations, while the others are nearly fast enough for interactive applications.

4.3 Comparisons Within Object Categories

The results shown in the previous subsection provide only averages over all models in the benchmark, and thus paint broad strokes. Perhaps more important than ranking algorithmic performance overall is to understand for which types of objects each algorithm performs best. Then, users can leverage the results of the benchmark to select the algorithm most appropriate for specific types of objects.

Table 2 shows a comparison of the performance of the seven algorithms for each object category. The entries of this table contain the rank of the algorithm according the Rand Index evaluation metric averaged over models of each object category: 1 is the best (shown in red) and 7 is the worst. From these results, we see that no one segmentation algorithm is best for every category. Interestingly, it seems that some algorithms do not necessarily work best (relative to others) on the types of objects for which they were designed. For example, the method designed to fit geometric primitives is among the best for Humans, but not for the Mech and Bearing classes, which are CAD Models comprised of geometric primitives. These results suggest that either people do not segment objects in the expected way, or that there are other cues that reveal the part structures of these objects (e.g., concave seams).

[1] It is important to note that the Shape Diameter Function and Core Extraction algorithms automatically predict the number of segments to output, while the other algorithms require *numSegments* as an input parameter. So, these two algorithms are solving a more difficult problem than the others. The reader should keep this in mind when comparing results. This issue is investigated further in Section 4.6

Object Category	Rand Cuts	Shape Diam	Norm Cuts	Core Extra	Rand Walks	Fit Prim	K-Median
Human	1	5	2	7	6	3	4
Cup	1	5	2	3	4	6	7
Glasses	1	4	2	6	7	5	3
Airplane	2	1	4	7	6	3	5
Ant	2	1	3	4	5	6	7
Chair	4	2	1	5	3	6	7
Octopus	4	1	3	2	5	7	6
Table	7	4	1	5	2	3	6
Teddy	1	2	4	3	5	6	7
Hand	1	7	3	4	5	6	2
Plier	2	7	4	1	5	3	6
Fish	3	1	5	2	4	7	6
Bird	1	2	4	3	7	6	5
Armadillo	3	1	5	7	4	2	6
Bust	1	3	6	5	2	4	7
Mech	4	3	1	6	2	5	7
Bearing	2	1	3	7	5	4	6
Vase	1	4	3	2	5	6	7
FourLeg	2	1	6	4	7	3	5
Overall	1	3	2	5	6	4	7

Table 2: *Comparison of segmentation algorithms for each object category. Entries represent the rank of the algorithm according to the Rand Index evaluation metric (1 is the best, and 7 is the worst).*

4.4 Properties of Human-Generated Segmentations

The human-generated segmentations acquired for the benchmark are useful not only for comparisons, but also for studying how people decompose objects into parts. Several psychologists have put forth theories to explain how people perceive objects as collections of parts (e.g., the Recognition by Components theory of Biederman [Biederman 1987], the Minima Rule hypothesis of Hoffman and Richards [Hoffman et al. 1984; Hoffman and Singh 1997], etc.), and several algorithms incorporate assumptions about what type of segmentations people prefer (e.g., cuts along concave seams [Katz and Tal 2003], segments of nearly equal area [Shlafman et al. 2002], convex parts [Chazelle et al. 1997], etc.). We conjecture that our benchmark data set could be used to test these theories and assumptions.

As an example, the *left column* of Figure 6 shows five histograms of properties collected from the 4,300 human-generated segmentations of the benchmark set: a) minimum curvature at vertices on (blue) and off (green) segmentation boundaries; b) convexity, as measured by the average distance from a segment's surface to its convex hull [Kraevoy et al. 2007] normalized by the average radius of the model; c) the length of a cut between segments normalized by the average of all cuts for its mesh; d) the area of a segment's surface normalized by the average segment area for its mesh; and e) the number of parts in each segmentation. We also have collected statistics for dihedral angles, maximum curvature, Gaussian curvature, mean curvature, minimum curvature derivative, and compactness [Kraevoy et al. 2007], but do not show them in Figure 6 due to space limitations.

These statistics seem to support many of the shape segmentation theories and assumptions that are prevalent in the literature. For example, segment boundaries are indeed more likely along concavities in our data set – i.e., at vertices with negative minimum curvature (Figure 6a). Also, segment surfaces tend to be convex – i.e., having relatively small distances between the surface of a segment and its convex hull (Figure 6b).

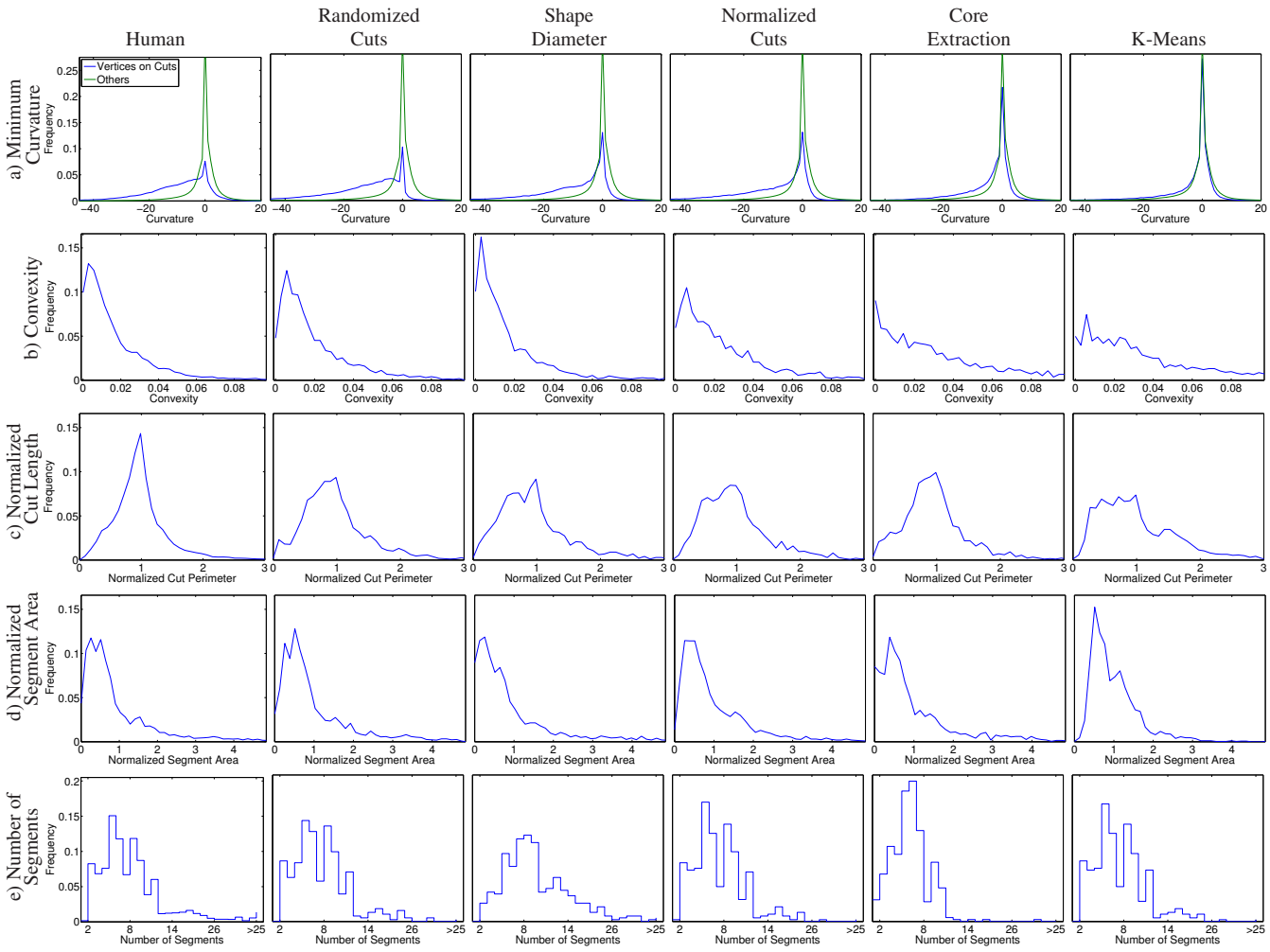


Figure 6: *Properties of human-generated segmentations (left column) compared with computer-generated segmentations (from second column to sixth) for five algorithms (properties of Fitting Primitives and Random Walks not shown due to space limitations).*

However, some of the results are a bit surprising. For example, it seems that cuts between segments within the same mesh tend to have approximately the same lengths – i.e., the Normalized Cut Length (length of a cut divided by the average for the mesh) is often close to one (Figure 6c). It seems that this property (or related ones, like symmetry) could be included in a segmentation error function, but we are not aware of any prior segmentation algorithm that does so. In contrast, it seems that not all segments have the same area – i.e., the Normalized Segment Area (area of a segment divided by the average for the mesh) is usually not equal to one (Figure 6d). Rather, it seems that each mesh has a small number of large segments (where the Normalized Segment Area is much greater than 1) and a larger number of small segments (where the Normalized Segment Area is less than 1), probably corresponding to the "body" and "appendages" of an object. This observation suggests that algorithms that explicitly aim for equal segment areas (e.g., K-Means) are less likely to mimic what people do. Of course, these statistics and observations are specific to our data set, but we conjecture that they are representative because the benchmark includes a wide variety of object types.

4.5 Properties of Computer-Generated Segmentations

We can also study properties of computer-generated segmentations to better understand how algorithms decompose meshes and how they compare to humans'. Towards this end, we show histograms of the five properties discussed in the previous section for segmentations computed by the Randomized Cuts, Shape Diameter Function, Normalized Cuts, Core Extraction, and K-Means algorithms in Figure 6 (from left to right starting in the second column). By comparing these histograms to the ones in the far left column (labeled "Human"), we aim to understand not only which computer-generated segmentations most closely resemble Humans', but also how are they similar and how are they different.

For example, looking at the histograms in the rightmost column of Figure 6 ("K-Means") and comparing them to the ones in the leftmost column ("Human"), we see that very few of the histograms are similar. It seems that the K-Means algorithm does *not* preferentially cut along vertices with negative minimum curvature (there is little separation between the blue and green lines in the top right histogram), but does preferentially decompose each mesh into segments with nearly equal areas (there is a large peak at '1' in the rightmost histogram of the bottom row). These properties do not match the Humans', which may explain why the evaluation metrics for K-Means were the poorest in Figure 5.

In contrast, comparing the first and second columns, we see that the histograms representing Randomized Cuts segmentations (second column) very closely resemble those of the Humans' (first column) – i.e., they often have the same shapes, maxima, and inflection points. These results corroborate the evaluation metrics presented in Section 4.2 – i.e., since many properties of the Randomized Cuts segmentations match the Humans', it is not surprising that they also produce the best evaluation metrics. Generally speaking, the histograms further to the left in Figure 6 resemble the Humans' better than those that are further to the right, and similarly are ranked better by most evaluation metrics.

4.6 Sensitivity to Number of Segments

As a final experiment, we investigate how sensitive our evaluations are to the number of segments produced by the algorithms in our study. As discussed in Section 3.5, several algorithms (Randomized Cuts, Normalized Cuts, Fitting Primitives, Random Walks, and K-Means) require a parameter that determines the number of target segments produced by the algorithm (*numSegments*). We set that parameter based on the most frequent number of segments found in human-generated segmentations of the same model in the benchmark (the mode for each model). This choice was made to ensure that the algorithms are being tested on typical input – i.e., it makes sense to evaluate how well an algorithm segments a coffee mug into two parts, but probably not seventeen. On the other hand, some algorithms do not take *numSegments* as a parameter and instead determine the number of segments to output automatically, and thus are at a disadvantage if the other algorithms are given a good estimate for the number of parts (note the differences for those two algorithms in the number of parts per model in the bottom row of Figure 6). To investigate the practical implications of this choice, we tested six alternative ways to set *numSegments* and study their effects on the relative performance of all seven segmentation algorithms:

1. **By Dataset:** *numSegments* could be set to the same number for all runs of every algorithm – i.e., to the average number of segments found amongst all examples in the benchmark data set (which is 7).

2. **By Category:** *numSegments* could be set separately for every *object category* – i.e., to the mode of the number of segments observed for that category in the benchmark data set.

3. **By Model:** *numSegments* could be set separately for every *model* – i.e., to the mode of the number of segments observed for that model in the benchmark data set.

4. **By Segmentation:** *numSegments* could be set separately for each human-generated segmentation. That is, every algorithm could be run once for every unique number of segments found in the benchmark, and then evaluations could be performed only between segmentations with exactly the same numbers of segments. This approach implies that the algorithm has an oracle that predicts exactly how many segments each person will produce, even if different people produce different numbers of segments – and thus provides a lower-bound on the evaluation metrics with respect to different selections of *numSegments*.

5. **By Shape Diameter Function:** *numSegments* could be set separately for every model according to the number of segments predicted by the Shape Diameter Function (SDF) algorithm. This choice allows investigation of how the relative performance of the SDF algorithm is affected by its choice in the number of segments – i.e., it puts the other algorithms on nearly equal footing.

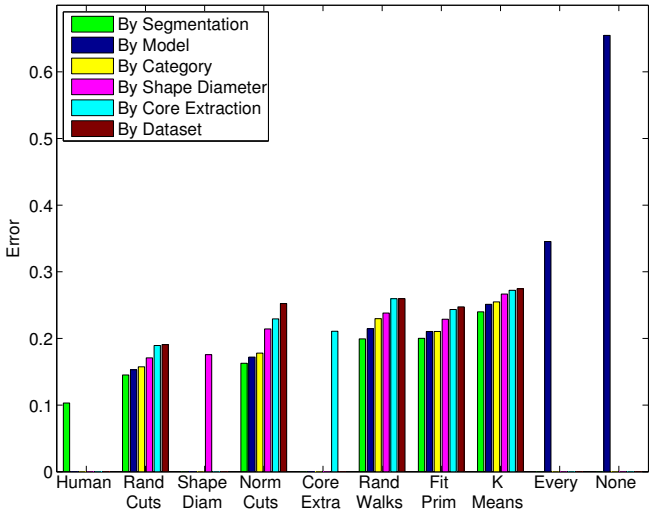


Figure 7: *Plot of Rand Index for six different methods of setting the target number of segments for algorithms that take it as input.*

6. **By Core Extraction:** *numSegments* could be set separately for every model according to the number of segments predicted by the Core Extraction algorithm. This choice allows investigation of how the relative performance of the Core Extraction algorithm is affected by its choice in the number of segments.

We implemented all six alternatives and ran an experiment to test how they affect the benchmark results. A plot of Rand Index for this study is shown in Figure 7. This plot is similar to the one in the bottom-left of Figure 5 (in fact, the dark blue bars are the same), except that each algorithm that takes *numSegments* as an input parameter is represented by six bars in this plot, one for each alternative for setting *numSegments*.

Looking at this plot, we observe that the comparisons of algorithms that take *numSegments* as input are largely insensitive to the method chosen to set *numSegments*. The main difference is that the algorithms that do not take *numSegments* appear relatively better or worse depending on how intelligently *numSegments* is set for the other algorithms. On the one hand, if the number of segments is chosen "By Dataset" (all models are decomposed into 7 segments), then the Shape Diameter Function algorithm (which chooses the number of segments adaptively) performs best. On the other hand, if the number of segments is chosen "By Segmentation" (green bars), "By Model" (dark blue bars), or "By Category" (yellow bars), then the Randomized Cuts algorithm performs best. Otherwise, we observe that the relative performance of algorithms remains remarkably consistent across different ways of controlling the number of segments.

Interestingly, when *numSegments* is chosen to match the number of segments predicted by the Shape Diameter Function algorithm (magenta bars), the performance of that algorithm is almost equal to the Randomized Cuts'. The same applies for the Core Extraction algorithm when *numSegments* is chosen to match its prediction (cyan bars). This result suggests that the performance difference between those two algorithms and the others may largely be due to differences in the numbers of segments generated, as opposed to differences in the placement of segments. However, this conjecture must be verified with an implementation for those two algorithms that allows setting *numSegments* (no such parameter is available currently).

Even more interesting is that fact that there is a significant difference between the performance of the best algorithms and the performance of the humans, even when the algorithms are given an oracle to set *numSegments* to match each human's exactly ("By Segmentation") – i.e., the green bar for "Human" on the far left is much smaller than the green bar for any algorithm. This is in contrast to the relatively small differences in performance between setting *numSegments* "By Segmentation" vs. "By Model" vs. "By Category." This result suggests that the main differences between humans and algorithms are not due to differences in the number of segments produced by the algorithms.

5 Conclusion

This paper describes a benchmark for mesh segmentation algorithms. The benchmark contains 4,300 manual segmentations over 380 polygonal models of 19 object categories. Initial tests have been performed to study the consistency of human-generated segmentations and to compare seven different automatic segmentation algorithms recently proposed in the literature. The results suggest that segmentations generated by different people are indeed quite similar to one another and that segments boundaries do indeed tend to lie on concave seams and valleys, as predicted by the minima rule. We find that the benchmark data set is able to differentiate algorithms, but we do not find one algorithm that is best for all object categories. Generally speaking, it seems that algorithms that utilize non-local shape properties tend to out-perform the others. Since the benchmark data set and source code is publicly available (http://segeval.cs.princeton.edu/), we expect that these results will be amended and augmented by researchers as more data becomes available, more tests are performed, and more algorithms are developed in the future.

Acknowledgments

This project would not have been possible without the help of several people and organizations who provided the data, code, and funding. We especially thank Daniela Giorgi and AIM@SHAPE for providing the meshes from the Watertight Track of SHREC 2007 that form the basis of the benchmark. We would also like to thank Sagi Katz, Ayellet Tal, Yukun Lai, Shi-Min Hu, Ralph Martin, Lior Shapira, Ariel Shamir, Daniel Cohen-Or, Vladislav Kraevoy, Alla Sheffer, Marco Attene, and Michela Spagnuolo, who provided source code and/or executables for segmentation algorithms. We are also thankful to Jia Deng and Fei-Fei Li for their assistance with the Mechanical Turk, to Szymon Rusinkiewicz for providing the trimesh2 library used for computing surface properties, and to Tushar Gupta and Jeehyung Lee for their efforts building prototype versions of the benchmark data set. Finally, we thank the NSF (CNFS-0406415, IIS-0612231, and CCF-0702672) and Google for funding.

References

AGATHOS, A., PRATIKAKIS, I., PERANTONIS, S., SAPIDIS, N., AND AZARIADIS, P. 2007. 3d mesh segmentation methodologies for cad applications. 827–841.

AMAZON, 2008. Mechanical turk. www.mturk.com.

ATTENE, M., KATZ, S., MORTARA, M., PATANE, G., SPAGNUOLO, M., AND TAL, A. 2006. Mesh segmentation - a comparative study. In *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006*, IEEE Computer Society, Washington, DC, USA, 7.

ATTENE, M., FALCIDIENO, B., AND SPAGNUOLO, M. 2006. Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput. 22*, 3, 181–193.

BENHABILES, H., VANDEBORRE, J., LAVOUE, G., AND DAOUDI, M. 2009. A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3d-models. In *Shape Modeling International*.

BIASOTTI, S., MARINI, S., MORTARA, M., AND PATANÈ, G. 2003. An overview on properties and efficacy of topological skeletons in shape modelling. In *Shape Modeling International*, 245–256, 297.

BIEDERMAN, I. 1987. Recognition-by-components: A theory of human image understanding. *Psychological Review 94*, 2, 115–147.

CHAZELLE, B., DOBKIN, D., SHOURHURA, N., AND TAL, A. 1997. Strategies for polyhedral surface decomposition: An experimental study. *Computational Geometry: Theory and Applications 7*, 4-5, 327–342.

CORREIA, P., AND PEREIRA, F. 2000. Objective evaluation of relative segmentation quality. In *ICIP00*, Vol I: 308–311.

EVERINGHAM, M., MULLER, H., AND THOMAS, B. T. 2002. Evaluating image segmentation algorithms using the pareto front. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, Springer-Verlag, London, UK, 34–48.

FEDERICO, M., GIORDANI, D., AND COLETTI, P. 2000. Development and evaluation of an Italian broadcast news corpus. In *Second International Conference on Language Resources and Evaluation (LREC)*, 921–924.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Trans. Graph. 23*, 3, 652–663.

GARLAND, M., WILLMOTT, A., AND HECKBERT, P. S. 2001. Hierarchical face clustering on polygonal surfaces. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, ACM, New York, NY, USA, 49–58.

GELFAND, N., AND GUIBAS, L. J. 2004. Shape segmentation using local slippage analysis. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, ACM, New York, NY, USA, 214–223.

GIORGI, D., BIASOTTI, S., AND PARABOSCHI, L., 2007. SHREC:SHape REtrieval Contest: Watertight models track, http://watertight.ge.imati.cnr.it/.

GOLOVINSKIY, A., AND FUNKHOUSER, T. 2008. Randomized cuts for 3D mesh analysis. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 27*, 5 (Dec.).

GREGORY, A. D., STATE, A., LIN, M. C., MANOCHA, D., AND LIVINGSTON, M. A. 1999. Interactive surface decomposition for polyhedral morphing. *The Visual Computer 15*, 9, 453–470.

HOFFMAN, D. D., AND SINGH, M. 1997. Salience of visual parts. *Cognition 63*, 29–78.

HOFFMAN, D. D., RICHARDS, W., PENTL, A., RUBIN, J., AND SCHEUHAMMER, J. 1984. Parts of recognition. *Cognition 18*, 65–96.

HUANG, Q., AND DOM, B. 1995. Quantitative methods of evaluating image segmentation. In *ICIP '95: Proceedings of the 1995*

International Conference on Image Processing (Vol. 3)-Volume 3, IEEE Computer Society, Washington, DC, USA, 3053.

KATZ, S., AND TAL, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22*, 3 (July), 954–961.

KATZ, S., LEIFMAN, G., AND TAL, A. 2005. Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10, 649–658.

KRAEVOY, V., JULIUS, D., AND SHEFFER, A. 2007. Shuffler: Modeling with interchangeable parts. In *Pacific Graphics*.

LAI, Y.-K., HU, S.-M., MARTIN, R. R., AND ROSIN, P. L. 2008. Fast mesh segmentation using random walks. In *Symposium on Solid and Physical Modeling*, 183–191.

LEE, Y., LEE, S., SHAMIR, A., COHEN-OR, D., AND SEIDEL, H.-P. 2004. Intelligent mesh scissoring using 3D snakes. In *12th Pacific Conference on Computer Graphics and Applications (PG)*.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 21*, 3, 362–371.

LIN, H.-Y. S., LIAO, H.-Y. M., AND LIN, J.-C. 2007. Visual salience-guided mesh decomposition. *IEEE Transactions on Multimedia 9*, 1, 46–57.

LIU, R., AND ZHANG, H. 2004. Segmentation of 3d meshes through spectral clustering. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, IEEE Computer Society, Washington, DC, USA, 298–305.

MARTIN, D., FOWLKES, C., TAL, D., AND MALIK, J. 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *in Proc. 8th Intl Conf. Computer Vision*, 416–423.

MORTARA, M., PATANE, G., SPAGNUOLO, M., FALCIDIENO, B., AND ROSSIGNAC, J. 2004. Blowing bubbles for the multi-scale analysis and decomposition of triangle meshes. *Algorithmica (Special Issues on Shape Algorithms) 38*, 2, 227–248.

MORTARA, M., PATANÈ, G., SPAGNUOLO, M., FALCIDIENO, B., AND ROSSIGNAC, J. 2004. Plumber: a method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In *ACM symposium on Solid modeling and applications*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 339–344.

OVER, P., LEUNG, C., IP, H., AND GRUBINGER, M. 2004. Multimedia retrieval benchmarks. *IEEE MultiMedia 11*, 2, 80–84.

RAND, W. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association 66*, 846–850.

ROBBIANO, F., ATTENE, M., SPAGNUOLO, M., AND FALCIDIENO, B. 2007. Part-based annotation of virtual 3d shapes. In *CW '07: Proceedings of the 2007 International Conference on Cyberworlds*, IEEE Computer Society, Washington, DC, USA, 427–436.

RUSSELL, B. C., TORRALBA, A., MURPHY, K. P., AND FREEMAN, W. T. 2008. Labelme: A database and web-based tool for image annotation. *Int. J. Comput. Vision 77*, 1-3, 157–173.

SHAMIR, A. 2008. A survey on mesh segmentation techniques. *Computer Graphics Forum 28*, 6, 1539–1556.

SHAPIRA, L., SHAMIR, A., AND COHEN-OR, D. 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput. 24*, 4, 249–259.

SHILANE, P., MIN, P., KAZHDAN, M., AND FUNKHOUSER, T. 2004. The princeton shape benchmark. In *Shape Modeling International*, IEEE Computer Society, Washington, DC, USA, 167–178.

SHLAFMAN, S., TAL, A., AND KATZ, S., 2002. Metamorphosis of polyhedral surfaces using decomposition.

SMEATON, A. F., KRAAIJ, W., AND OVER, P. 2004. TREC video retrieval evaluation: a case study and status report. In *Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval (RIAO)*.

UNNIKRISHNAN, R., PANTOFARU, C., AND HEBERT, M. 2005. A measure for objective evaluation of image segmentation algorithms. In *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05), Workshop on Empirical Evaluation Methods in Computer Vision*, vol. 3, 34 – 41.

WU, H.-Y., PAN, C., PAN, J., YANG, Q., AND MA, S. 2007. A sketch-based interactive framework for real-time mesh segmentation. In *Computer Graphics International*.

ZHANG, H., FRITTS, J. E., AND GOLDMAN, S. A. 2008. Image segmentation evaluation: A survey of unsupervised methods. *Comput. Vis. Image Underst. 110*, 2, 260–280.

ZHANG, Y. 1996. A survey on evaluation methods for image segmentation. *PR 29*, 8 (August), 1335–1346.

ZÖCKLER, M., STALLING, D., AND HEGE, H.-C. 2000. Fast and intuitive generation of geometric shape transitions. *The Visual Computer 16(5)*, 241–253.

ZUCKERBERGER, E. 2002. Polyhedral surface decomposition with applications. *Computers and Graphics 26*, 5 (October), 733–743.