Consistent Segmentation of 3D Models

Aleksey Golovinskiy and Thomas Funkhouser

Princeton University

Abstract

This paper proposes a method to segment a set of models consistently. The method simultaneously segments models and creates correspondences between segments. First, a graph is constructed whose nodes represent the faces of every mesh, and whose edges connect adjacent faces within a mesh and corresponding faces in different meshes. Second, a consistent segmentation is created by clustering this graph, allowing for outlier segments that are not present in every mesh. The method is demonstrated for several classes of objects and used for two applications: symmetric segmentation and segmentation transfer.

Key words: Mesh segmentation, Mesh analysis

1. Introduction

The goal of our paper is to develop a method that can produce a consistent segmentation of a set of meshes (Figure 1b). Such a segmentation is useful for a number of applications. Parts can be labeled and put into a knowledge ontology [24], they can be interchanged as part of a modeling tool [15], and they can be put into a searchable database [27]. In addition, we can expand lexical databases such as Wordnet [6] from having part-of relationships ("a seat is a part of a chair") to having possibly probabilistic spatial relationships ("some chairs have armrests, which are oblong, in front of backs, and above seats"). All these applications have at their heart the problem of this paper: consistently decomposing a set of 3D models into parts.

Many methods have been proposed to segment an individual mesh into parts. While these methods have produced segmentations of increasing quality, consistently segmenting a set of meshes remains challenging. Some mesh segmentation methods use heuristics designed to remain consistent between meshes (such as the Shape Diameter Function [29]). However, segmenting meshes individually ignores important cues available from processing a whole class of objects simultaneously. While several methods [27, 15] have been proposed to segment multiple objects, compared to our technique they have shortcomings, such as assuming that each mesh can be segmented individually, or that each mesh has the same number of segments.

Our approach extends the idea of single-mesh segmentations to a segmentation of multiple meshes: we simultaneously segment models and create correspondences between segments. Specifically, we first build a graph whose nodes represent faces of all the models in the set, and whose edges represent links between adjacent faces within a mesh, and between corresponding faces of different meshes. We then cluster the graph, creating a segmentation in which adjacent faces of the same model and corresponding faces between different models are encouraged to belong to the same segment.

Our approach has several advantages. First, segmenting a set of objects in a class simultaneously can produce not only more consistent results across the class, but also better individual segmentations than segmenting each object separately (as demonstrated in Figure 1). This is because a set of objects helps to identify salient segments that are shared across the set, and because those models that have more obvious segmentation cues help to segment more difficult models. Second, modeling tools are often used to create an object via a hierarchy that is then saved in VRML and other formats, and this hierarchy can be used as "prior" segmentation to give cues to the desired segments. Our approach combines these prior segmentations with other traditional segmentation cues, such as connected mesh components, concavities, and short boundaries between components. Third, by posing the problem as that of clustering a graph representing all the meshes, our method allows for outlier segments in the resulting segmentation, such as detection of armrests only on those chairs that have them. Finally, our method handles models with disconnected components, which is not the case for many other segmentation algorithms.

We demonstrate the effectiveness of our method on several object classes. We then suggest two new applications of our method: 1) creation of a symmetry-respecting segmentation of a single model, and 2) transfer of segmentations between a set of models of the same class.

2. Previous Work

We review previous work in several related categories: segmentation of individual meshes, segmentation of sets of objects, consistent parameterization, and semantic labeling.

Segmentation. Many mesh segmentation methods exist in the graphics literature that aim to decompose a mesh into functional parts; a recent survey can be found in [28] and [1], and comparisons between several algorithms appear in [3]. These

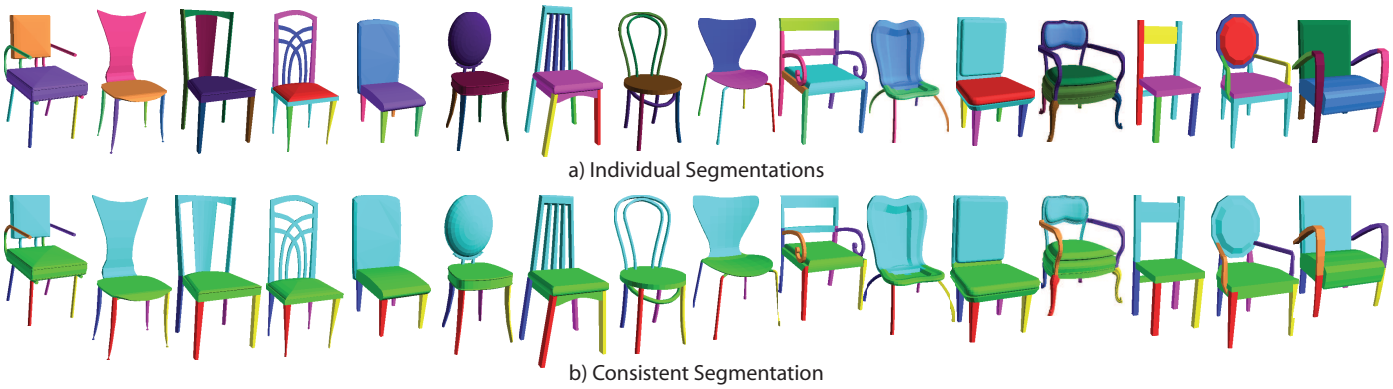

Figure 1: Individual segmentations of a set of chairs are shown in (a). Instead, our method creates a consistent segmentation of a set of meshes (b), that allows outlier segments, such as armrest segments in those chairs that have armrests. Consistent segmentations not only bring similar object parts into correspondence, but create better part decompositions for each mesh than individual segmentations do. Note that in many individual segmentation, the backs of chairs are either broken into several components, or aggregated with the rear legs.

approaches aim to create segments that are well-formed according to some pre-defined low-level criteria: the segments are convex, boundaries lie along concavities, etc.. They use techniques such as K-means [30], graph cuts [12], hierarchical clustering [7, 8, 11], random walks [16], core extraction [13], tubular primitive extraction [21], spectral clustering [19], and critical point analysis [18]. While some research tries to use segmentation criteria that are consistent between meshes in a class (such as shape diameter function [29]) or between articulated versions of a model (such as diffusion distance [5]), it is difficult in general to run segmentation methods independently on a set of meshes and obtain results with corresponding segments (Figure 1a). We expand such segmentation techniques to simultaneously segment a set of meshes.

Segmentation of Sets of Objects. Several computer graphics papers have been written that involve segmentation of sets of objects. Part Analogies [27], for example, segments each model into parts, and then creates a distance measure between parts that takes into account both local shape signatures, and the context of the parts within a hierarchical decomposition. The main output of this is a catalog of parts with inter-part distances, which can then be used to create a consistent segmentation. We differ from this approach in that instead of first creating independent segmentations of the models and then finding correspondences between these segments, we create segments and correspondences between them simultaneously. Segmenting models and finding segment correspondences simultaneously improves segmentation quality because (i) a set of objects helps to identify salient segments that are shared across the set, and (ii) those models that have more obvious segmentation cues help to segment more difficult models.

Another related work is Shuffler [15], a modeling tool that allows the user to swap a segment in one mesh for a segment in another. Shuffler creates a mutually consistent segmentation between a pair of meshes, in which the segments are in one-to-one correspondence. We differ from their approach in two ways. First, while Shuffler's method could in principle be extended to segmenting a set of objects, their work demonstrates only pairwise segmentations. Second, we do not assume that all models contain all parts but instead allow outlier segments, making the problem more difficult, but allowing richer decompositions.

Some papers in the computer vision literature aim to obtain a segmentation of the same object or similar objects in different images: [17] creates an implicit shape model for segmenting a class of objects from examples, [25] uses a generative graphical model to improve segmentation by using two images rather than one, and [31] uses spectral clustering to find correspondences and matching segments. The last method constructs a Joint-Image Graph that encodes intra-image similarities as well as inter-image correspondence, which is similar to the graph we construct. Our method extends this idea to the 3D model domain, where the challenges lie not in disconnecting foreground objects from a cluttered background, but in using geometry to create precise segmentations.

Consistent Parameterization. Another approach to creating a consistent segmentation may be to create a continuous mapping between the objects, segment one of them, and transfer the segmentation to the others. A large body of work exists that deals with consistently parameterizing a set of meshes. Some papers offer general methods for establishing a mapping between surfaces (e.g. [22, 14, 26]) while others focus on consistent parameterization for a particular class of objects such as faces [4] and bodies [2]. Once such a mapping is created, surface attributes such as texture coordinates or displacement maps can be transferred. Unfortunately, many real-world object classes have a large degree of intra-class variation and inconsistent mesh quality and topology, making it difficult or impossible to find a single consistent parameterization across the entire class, unless the meshes are very similar (such as faces [4] or human bodies [2]). For example, there is no continuous global mapping from a chair with armrests to one without them. In this paper, we propose a method to find consistent segmentations without first finding a continuous parameterization.

Semantic Labeling. Several papers address the problem of assigning semantic labels to meshes. ShapeAnnotator [24] de-

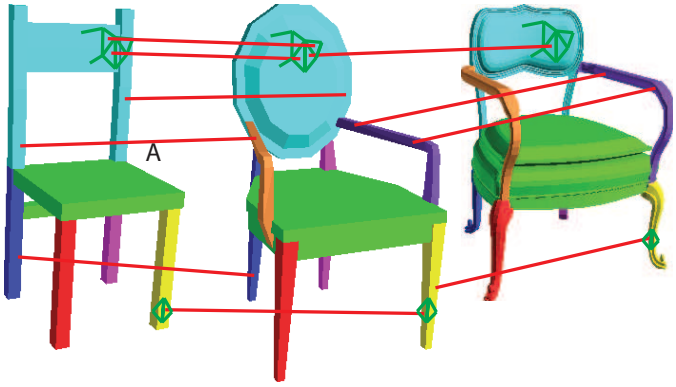

Figure 2: Schematic of our method. We create a graph whose nodes are mesh faces, and whose edges connect adjacent faces within a mesh (green lines) and corresponding faces in different meshes (red line). Some of the correspondences are incorrect (such as the one labeled 'A'). We find a clustering of this graph (with clusters represented by the colored rendering) that represents a consistent segmentation, allowing for outliers such as armrests.

scribes an effort to put 3D models into a knowledge base, and consistently annotate their parts. They do not provide an automatic algorithm to do so, instead offering a tool that presents the output of several automatic segmentation algorithms such as those described above, and allows the user to select sections from each output to form segments and annotate them. Such efforts would benefit from a tool such as ours that creates a consistent segmentation of meshes automatically.

3. Method

Our algorithm takes as input a set of meshes, and produces as output a *consistent segmentation*, which assigns every face of every model to a segment, so that, for example, all mesh faces on seats of chairs belong to the same segment. This algorithm proceeds in two main steps, as follows.

First, we create a graph that contains as nodes the faces of the models. To this graph, we add two types of edges. Adjacency edges connect neighboring faces within a mesh, and are responsible for intra-mesh segmentation. Their weights use cues such as disconnected components and concavities to estimate whether they cross a segment boundary. The adjacency edges are similar to edges used in other segmentation algorithms and encourage short segmentation boundaries along concave edges (and, in our case, between disconnected components).

The other edges in our graph are correspondence edges. These edges are created between closest point pairs of globally aligned models. Ideally, correspondence edges link surfaces of corresponding parts of different models. These edges are responsible for inter-mesh segmentation; that is, they encourage corresponding parts of different meshes to aggregate into the same segment.

The resulting graph is sketched in Figure 2, with adjacency edges shown in green and correspondences edges in red. Note that the correspondence edges are not always correct: the correspondence edge labeled 'A' associates the armrest of one chair with the back of another.

Given this graph, we create a consistent segmentation by clustering its nodes into disjoint sets. This is done with a greedy hierarchical algorithm that seeks to minimize an error function similar to the normalized cut cost. The resulting clusters aggregate mesh parts such that each part (i) is weakly connected to the rest of the mesh and (ii) has more inter-mesh correspondences to the other parts in the cluster than to parts outside of the cluster. In Figure 2, these clusters are represented by mesh face colors.

Our approach has the following advantages. It balances the two smoothness constraints desirable in a consistent segmentation: adjacent faces should belong to the same segment, and corresponding faces from different meshes should belong to the same segment. Because these are not hard constraints, inevitable wrong correspondences can be corrected by strong adjacency cues, and poor adjacency cues can be corrected by consistent correspondences. For example, some of the faces of a chair's armrest may be initially erroneously found to correspond to faces of another chair's back (correspondence labeled 'A' in Figure 2). However, strong adjacency cues would prevent a small part of an armrest from being cut from the rest of the armrest and being associated with the "back" segment. Conversely, a chair may have no adjacency cues to suggest that its back should be separated from its rear legs (such as the first chair of Figure 2). But, consistent correspondences between the faces of this chair's and other chairs' rear legs would suggest that they belong in a separate segment. Finally, posing the consistent segmentation problem as a clustering problem allows for outlying segments: it permits identifying parts such as armrests only in those chairs that have them.

4. Implementation

The following two subsections describe in greater detail how we build the graph for our method, and how we segment the graph.

4.1. Graph Construction

The first step is to construct a graph, sketched in Figure 2, whose nodes represent mesh faces, and whose edges encode two kinds of weighted constraints: adjacency constraints, that suggest that adjacent faces belong in the same segment, and correspondence constraints, that suggest that faces from different meshes that are in correspondence belong in the same part.

Adjacency Edges. Adjacency edges represent traditional segmentation cues of single-mesh segmentations: segment boundaries should be small, should lie along concavities, and should separate disconnected components. In addition, if the mesh contains a part hierarchy (as meshes created with modeling tools often do), the output should aim to respect this hierarchy.

Adjacency edges are formed between every pair of adjacent faces. Weights are chosen such that the cost of a graph cut corresponds to the perimeter of the enclosed segment, weighed by the concavity of the edges, as in [9]. Namely, if θ is the exterior dihedral angle of a mesh edge and l is its length, the

weight of the corresponding graph edge is $\min((\theta/\pi)^{\alpha} l, l)$, with α chosen to be 10. Finding segments of small cut cost in this graph encourages segment boundaries to be small, and to lie along concavities.

These edges are sufficient to form a graph within connected components of the mesh, but 3D models are often composed of many disconnected components (in the examples in this paper, the number of disconnected components ranges from 1 to 70). More adjacency edges need to be added to encourage segment boundaries along disconnected components, while allowing disconnected components to join into the same segment when necessary. The details of these additional edges are described below for completeness, but the key idea is illustrated in Figure 3, where disconnected components are represented by rectangles, and the new adjacency edges by green lines. These edges are created such that those connected components that are closer to each other (Figure 3a), and that are relatively close along a larger surface area (Figure 3b), are more strongly connected in the graph.

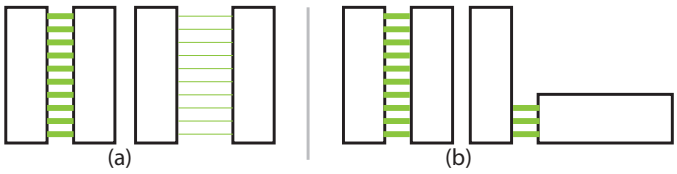

Figure 3: Disconnected components are drawn as rectangles, which we connect with (green) edges. We want components that are closer (a), and have more surface area in proximity (b) to be more strongly connected.

In greater detail, if D is the diameter of the bounding box of the mesh, we consider all pairs of disconnected components whose closest distance d is less than $.1D$. We would like to add edges to the graph (corresponding to the green lines in the schematic of Figure 3) whose lengths are close to d. We let $d_{eps} = .005D$ be a measure of a "small" length, and we consider $d' = 1.2(d + d_{eps})$. We sample points on both components, and for each point whose closest point on the other component is closer than d', we add an edge between the faces containing the points. The weight of that edge is a decreasing function of the distance times a constant; namely, we set it to $k_{cc}\overline{C}/(1+d'')$, where k_{cc} is a constant that controls how important aggregating faces within a connected component is versus aggregating connected components, $\overline{C}$ is the average adjacency edge cost of the mesh, and d'' is the distance between the pair of points relative to a small mesh distance (that is, divided by $.005D$). For setting $k_c c$, no heuristic is foolproof, but we find it helpful to set k_{cc} low enough so that each connected component is fully aggregated before disconnected components are combined; after a few experiments, we set it to $k_{cc} = .01$.

Finally, we incorporate cues from a pre-existing part hierarchy, if one is available. We do this by considering the leaf nodes of the hierarchy as a "prior" segmentation, and encouraging segment boundaries along the boundaries of this "prior" segmentation. Specifically, we lower the cost of edges that cut across prior segmentation boundaries by a factor k_{prior} (we use $k_{prior} = 10$).

Correspondence Edges. Correspondence edges link corresponding faces from different meshes using a set of weighted point correspondence pairs. While our algorithm accepts any scheme that creates weighted correspondences between surface points of different models, to avoid obscuring our main algorithm, we only use the simplest such scheme: closest points in the alignment that minimizes the Root Mean Square Distance between a pair of meshes within the space of similarity transforms. In particular, we align the pair of models with PCA, and starting with the 24 possible axis orientations, run the ICP algorithm, returning the alignment with the lowest RMSD. We find that this method rarely fails to find the best similarity transform relating a pair of models, and failure to find high quality correspondences instead usually indicates the need to use non-uniform scale or a non-linear alignment.

To form correspondence edges, we first align each mesh to every other mesh. We then sample the surface of the "from" mesh, and for each point, find the closest compatible point on the "to" mesh. Compatibility is determined by whether the dot product of the normals is greater than a threshold (we use .3). If this closest compatible point is sufficiently close (within 20% of the diameter of the bounding box of the model), we form a correspondence edge between the face node containing the "from" point, and the face node containing the "to" point. The result is that the cut cost of the correspondence edges in a clustering represents the surface area of each mesh of those points on the mesh that are in different segments from their corresponding points on other meshes.

4.2. Clustering

Given the graph constructed above, we need to define an error function and a procedure to minimize it that results in a segmentation of the graph, grouping together adjacent and corresponding faces of the input meshes. A number of segmentation schemes can be adapted to this problem; we use as our basis the hierarchical clustering scheme of [9], in which each face starts in a separate segment, and segments are greedily merged in the order that minimizes a normalized cut cost of the graph.

To measure adjacency error E_{adj}, we use the same area-weighted normalized cut cost as in [9], which is the sum of the adjacency edge cut cost of each segment normalized by the area of the segment. For correspondence error E_{corr}, we use the normalized cut error of the correspondence edges, which is the sum of the correspondence edge cut cost of each segment normalized by the total cost of the correspondence edges attached to nodes in the segment. These errors represent quantities of different dimensionalities, so we form our segmentation error as the weighted product of the two:

$$E_{seg} = \left(E_{adj}\right)^{\alpha} (E_{corr})^{1-\alpha}$$

where α controls the trade-off between merging segments within a mesh (towards $\alpha = 1$) and merging segments in close correspondence between different meshes (towards $\alpha = 0$).

Similarly to [9], our method of (approximately) minimizing the above error is hierarchical clustering: we start with each face in its own segment, and greedily merge segments in the

order that yields the least error. However, there are two difficulties with this approach directly applied to the above error. The first is that the adjacency error encodes several cues (concavity, disconnected components, pre-existing segmentation) that vary widely from mesh to mesh and are difficult to normalize among meshes. This may result, for example, in one mesh having the same error with two segments as another mesh may have with twenty. The second problem is that merging in order of E_{seg} is relatively slow, since the contribution of each segment to the error is complicated, whereas merging in order of E_{adj} can be done relatively quickly with a priority queue since each potential segment merge affects the error as only a function of the two segments involved.

Due to these two reasons, we split our method into two steps. First, we oversegment each mesh independently to some preset number of segments, which is assumed to be an oversegmentation of the final result, by merging to minimize only E_{adj}. This initial oversegmentation is similar to the idea of superpixels [23] in computer vision. Note that while this stage is executed independently for every model, we do not try to segment the model to its final decomposition into natural parts; we merely aim for an oversegmentation, which is a much easier problem. Following this stage, we perform a slower segmentation of the entire graph representing all the models in the class by, at each iteration, considering all potential segment merges and computing the full E_{seg} for each (we use $\alpha = .01$ in this step). Note that while inter-mesh segmentation only begins at the second stage, intra-mesh segmentation continues.

Finally, to compensate somewhat for the greedy nature of the algorithm and to allow the correction of a poor choice of segments to merge, we perform border refinement every k_{br} steps during the second segmentation stage (we use $k_{br} = 12$). In this step, we check over all graph nodes adjacent to each segment boundary, and expand the segment to contain an adjacent node if that expansion decreases the error. In all, two parameters are manually chosen for each consistent segmentation: the number of parts to oversegment each model to in the first phase, and the final number of segments in the consistent segmentation.

4.3. Running Time

Our algorithm has three main time-intensive stages. First, the graph is constructed; in particular, adjacency edges between disconnected components and correspondence edges are created. If there are m meshes, with n vertices, and d disconnected components per mesh, with the aid of a kd-tree the time complexity is $O(md^2n \log n)$ and $O(m^2n \log n)$ for disconnected component and correspondence edges respectively. This stage took about 20 seconds for the chair example shown in Figure 1 on a 2GHz PC (the chairs class has 16 chairs, with a number of faces ranging from 168 to 9272, with an average 2296 faces per chair, and a number of connected components ranging from 1 to 16, with an average of 8). Second, each mesh is independently oversegmented; the running time for this oversegmentation is $O(mn \log n)$, and it took about a further 5 seconds for the chair example. For classes with many models, the most time-consuming stage is the final stage, in which we consider potential merges between segments of the oversegmented meshes at every iteration. The complexity of this stage is independent of mesh resolution. If the oversegmentation creates k segments per mesh, there are $O(mk)$ merges to perform, and a potential $O(m^2k)$ merges to check at every iteration, so the complexity is $O(m^3k^2)$. This stage takes about 3 minutes for the chair example (which was oversegmented to $k = 13$ segments). In all, the algorithm takes several minutes for the most complicated experiments in this paper. Many optimizations can be made to improve this. In particular, it is likely that not all pairs of meshes need to be considered for correspondence, and a constant number of corresponding meshes may be sufficient for each mesh.

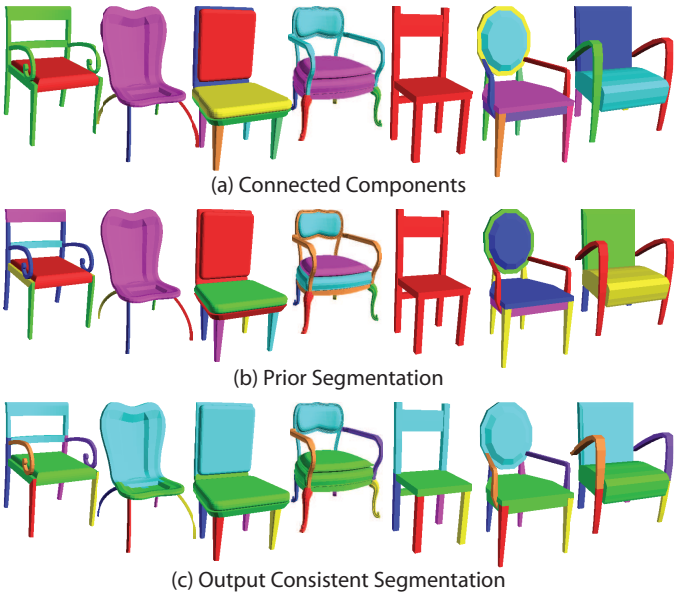

Figure 4: For the right-most chairs of Figure 1, we show the connected components (a) and prior segmentation (b) that serve as cues for segmentation. These cues are helpful, but inconsistent between different models. We replicate the consistent segmentation results for these models in (c).

5. Results

In this section, we discuss some results of our method, and suggest several applications. We discuss three scenarios: the standard use of our algorithm to make consistent segmentations, the creation of a symmetric segmentation of a single mesh, and segmentation transfer between meshes of the same class.

5.1. Consistent Segmentations

The basic application of our method is to take as input a set of models and create a consistent segmentation. These models may be pulled automatically from the Internet, categorized automatically or semi-automatically, and then processed with our method to create a database of segments which may then be used in modeling or animation. We demonstrate the utility of our method for this application with models from the Viewpoint database, a commercial database containing household objects and furniture.

Figure 1b shows the results of segmenting a set of chairs. To give an example of the adjacency cues for this set, the connected components and prior segmentations are shown for several of the chairs in Figure 4. Note that while these are helpful cues, they are inconsistent between different meshes. There are instances where final part boundaries did not lie along boundaries in connected components or prior segmentation, and most connected components and prior segments do not correspond to the final parts. These cues had to be combined to form the segmentations: some chairs relied more heavily on connected components, others on prior segmentations, and yet others used cues from the underlying geometry, such as concavity (the back and seat of several chairs would have been one segment if only prior segmentation or connected components were used). Despite these challenges, our method was able to create a consistent segmentation, and also find the outliers: armrests were found in those chairs that had them, enriching the resulting segmentations. Finally, Figure 1a shows the chairs segmented individually to the same number of segments achieved in the consistent segmentation. This illustrates how segmenting the chairs in a group enhances individual segmentations: many of the cues to segment back legs from the backs of chairs come from correspondences to other chairs.

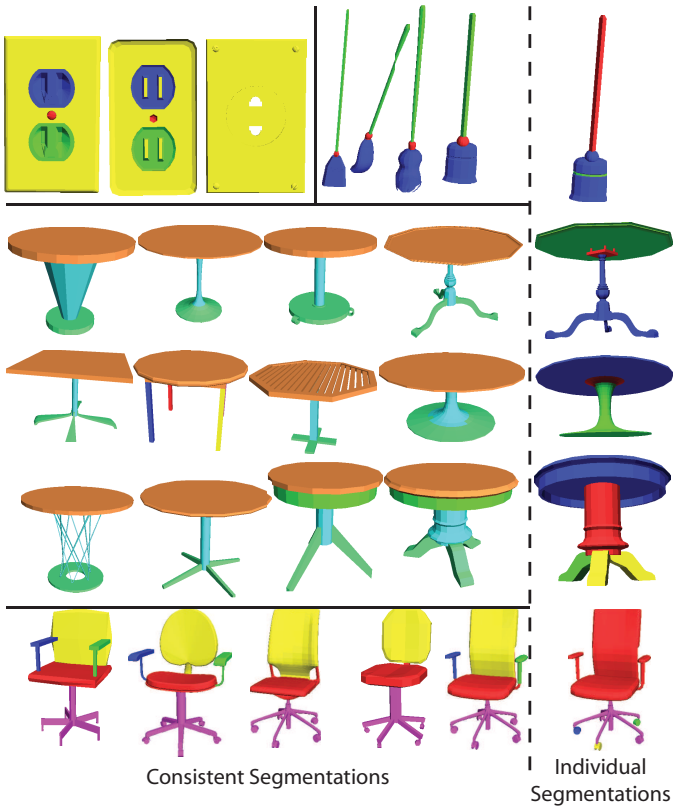

Figure 5: Results of consistent segmentations of several classes are shown to the left of the dotted line. Note that in many cases, the ability of our method to find outlier segments is helpful. To the right of the dotted lines are individual segmentations of the last models in the row. These individual segmentations fail to identify important segments that can be identified by comparing to other models in the class.

We show similar results for several other object classes in Figure 5. To the left of the dotted line, we show the consistent segmentations of several sets of objects: outlets, brooms, tables, and swivel chairs. To save space, we do not show the connected component and prior segmentation cues, but, similarly to the example in Figure 1, these cues are inconsistent between models, and the results used a combination of connected component, prior segmentation, and geometry cues. Most of these classes show examples where the possibility of outlier segments is helpful. The plugs and middle screw of outlets, the cylindrical elements around table tops and the legs of the four-legged table, and the armrests of swivel chairs would not have been identified if every object were required to have every part. Finally, to the right of the dotted lines are those models found immediately to the left of the dotted line segmented individually (to the same number of segments). These examples demonstrate how segmentations of sets of objects enhance the individual segmentations. For tables, for example, comparison to other tables makes clearer that the leg stands are a more important segment than a frequently disconnected component joining the stand to the top.

5.1.1. Symmetric Segmentations

It is desirable that the segmentations of symmetric or nearly symmetric objects be symmetric. However, most segmentation methods do not not produce symmetric results when run on nearly symmetric models. Figure 6a, for example, shows the results of the hierarchical segmentation algorithm we use in our clustering phase on two nearly symmetric models. Such segmentations of nearly symmetric objects are often not symmetric for two reasons. First, segments are treated one at a time in many algorithms, and an asymmetry forms when, for example, one human arm is processed before the other–the segmentation errors become different after the merge and the algorithm may proceed differently when it gets to the other arm. Second, slight asymmetries in triangulation and geometry result in asymmetries in energies. To overcome such deficiencies, we take a "symmetry-aware" processing approach as proposed in [10].

Figure 6: Using intra-model constraints to create a symmetric segmentation (b) of models that would have otherwise been segmented asymmetrically (a).

Our method places intra-model correspondence constraints between symmetrically corresponding points instead of be-

tween points on different meshes. To get these constraints, we can use a method such as [10], which finds surface mappings between corresponding parts of nearly symmetric objects. For the two examples shown here, the symmetry is so close to perfect that it is sufficient to create closest-point correspondences to the reflection of the model. In Figure 6b, we show segmentations of the models created with our method, with constraints added from a global reflective symmetry. Note that not only does this result in correspondences between symmetric segments, but also produces better segmentations: the asymmetric cuts above the chest of the human model, and across the face of the dog are eliminated. While we demonstrate results for the simple case of one strong global symmetry, since our method takes as input any weighted point correspondences, it can also be used for multiple, local, and partial symmetries.

5.1.2. Segmentation Transfer

In many scenarios, it may be helpful to start with example segmentations of objects and transfer the segmentations to previously unsegmented meshes. These example segmentations may come in the form of labels from a user interface, previously existing examples, or the result of other algorithms that may be tuned to find specific object parts. Here, we explore three such scenarios: transferring segmentations from a single example to a set, using a set of segmented examples to segment a new model, and taking advantage of partially segmented data to complete the segmentations. Our algorithm can handle any mixture of these constraints.

Consider the three biplanes in Figure 7. They have no prior segmentation, and their connected components are shown in Figure 7a on the left. Note that here, connected components mostly represent an oversegmentation; however, for example, in the third plane, the top and bottom wings as well as their vertical supports are all one connected component, where we would like to create four segments (blue arrow). We can run our method, as usual, to create a consistent segmentation (right column of (a)). This segmentation may not match the desired one; for example, it may be better for the horizontal tail stabilizers to be represented as two segments, and to separate the vertical tail stabilizer from the body (red arrow). Below, we describe three scenarios in which additional constraints (in the form of "known" segments) may be used to improve the situation and to transfer segmentations from examples. The only change needed to our method to reflect these constraints is to start with the faces merged into those segments that are "known", and to prevent two "known" segments from merging.

In the first scenario, we can use a segmented example mesh to label a set of other meshes in the class. This segmented example may be generated in real time with an interface, or it may come from an existing repository or other external sources. In our example, such a scenario is represented in Figure 7b on the left. The first plane is segmented, and used as a template for the other two planes. The result is shown on the right of Figure 7b. Compared to the baseline consistent segmentation result, the horizontal tail stabilizers are recognized as two segments, and the vertical stabilizer is separated from the body.

In the second scenario, there exists a fully segmented set of models belonging to some class. We would like to augment this set with a new, unsegmented model. This scenario is represented in Figure 7c on the left: the first two planes are segmented, and the third is not. Using these two models, we segment the third plane; the segmentation is shown on the right of Figure 7c. Compared to the baseline consistent segmentation case, the stabilizers are segmented correctly following the examples. Compared to using one segmented example, since more precise correspondences are available, the propeller segment is smaller and better resembles the example propeller segments, although it also includes a small part of the plane that is not the propeller (green arrow).

In the third scenario, the input is a partially segmented set of meshes. Such a partially labeled set can come from several sources. One possibility is to use the labels created by users during modeling, and attached to object parts. While these labels are not consistent, it is conceive able that parsing them and using a lexical database such as Wordnet [6] will create proper associations. Alternatively, one may use automatic detectors that are trained to find specific object parts. Note that in this scenario, these partial segmentations offer much more information than the "prior" segmentation shown, for example, in Figure 4b: for this application, we assume that those segment boundaries that are provided are correct, and that segments of the same type provided in different meshes are associated with each other. We simulate this situation by manually segmenting the biplanes of Figure 7, and randomly removing half of the segments. The input is shown on the left in Figure 7d, and the output shown on the right. Note that, again, compared to the unconstrained segmentation, the rear stabilizers are found as desired.

6. Conclusion

We present a method that takes as input a set of meshes, and produces a consistent segmentation from several inconsistent cues. The method is demonstrated on several examples, and two more specific applications are suggested: symmetric segmentations of a single mesh and segmentation transfer between meshes belonging to the same class.

As this is a prototype system, there are several limitations. First, our method uses parameters to balance the relative importance of various segmentation cues. These parameters could, in principle, have been learned from example segmentations. Instead, we set them experimentally to favor aggregating adjacent, convex faces, followed by adjacent, concave faces, then close disconnected segments, and then distant disconnected segments, with faces within "prior" segments aggregating before faces across prior segmentation boundaries. We did not fine-tune these parameter values - most of them are our initial choices, so the values are reasonable, but by no means optimal.

Second, as implemented, our method is limited to those cases in which satisfactory correspondences may be created through a global similarity alignment. These correspondences do not need to be perfect (in none of our examples were the correspondences perfect), since segmentation cues can compensate for

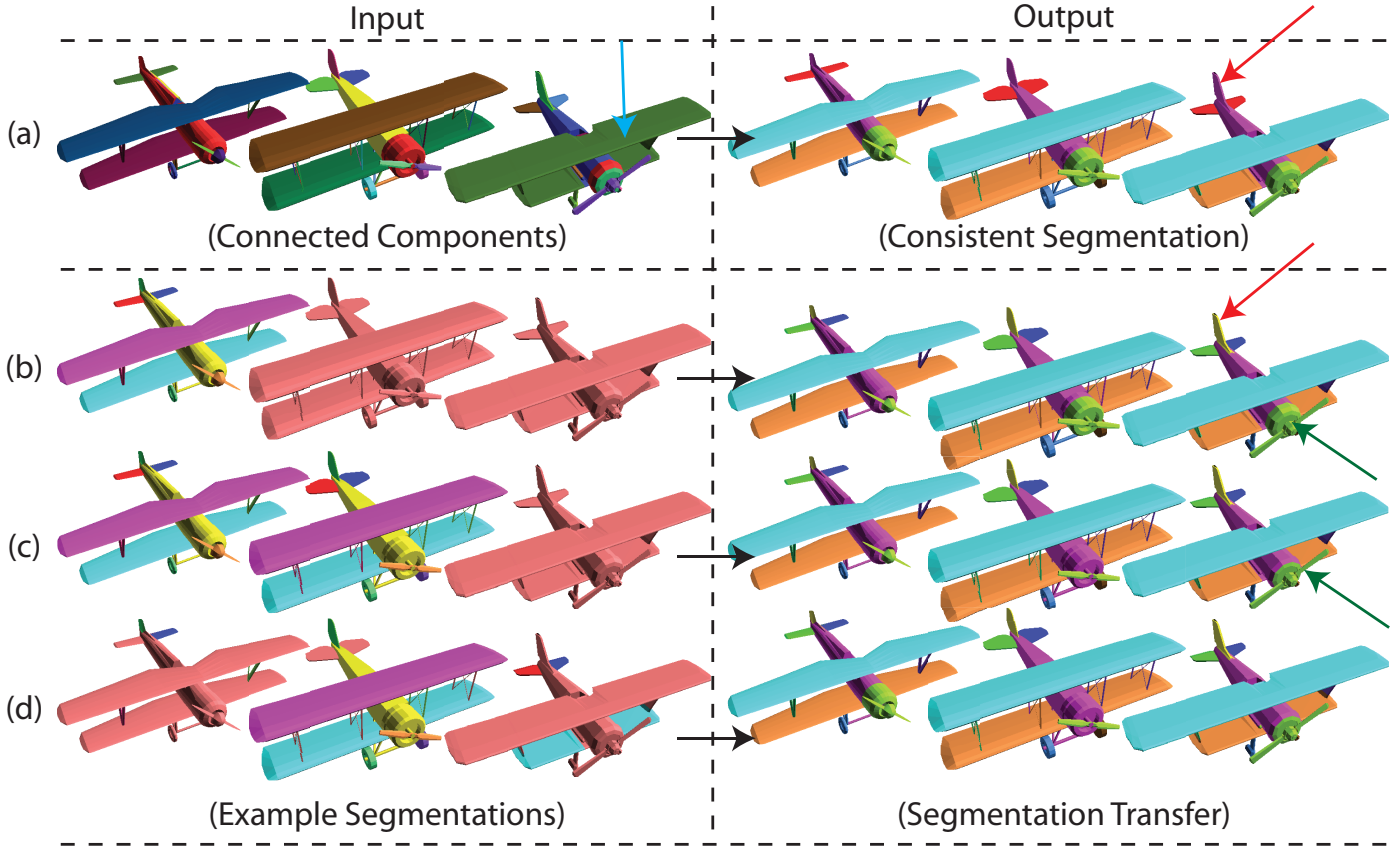

Figure 7: Our method can transfer segmentations from example meshes. Consider the three biplanes, whose connected components are shown on the left side of (a). We can run our method, as usual, to create a consistent segmentation (right column of (a)). This segmentation may not match the desired one; for example, it may be desirable to separate the three stabilizers in the back of the plane (red arrow). We can use the first plane as an example to segment the others (b). This segments the stabilizers correctly, but may miss other desired details. For example, the propeller segment may be too large (green arrow). In (c), we use the example segmentations of the first two planes to segment the third. Compared to (b), this creates more precise restrictions, and creates a smaller propeller segment. Our method can also handle constraints in which segments are partially given, and the remainder is required to be filled in. In (c), we randomly choose half of the segments to be known, and use these as examples to generate the segmentation in the right column.

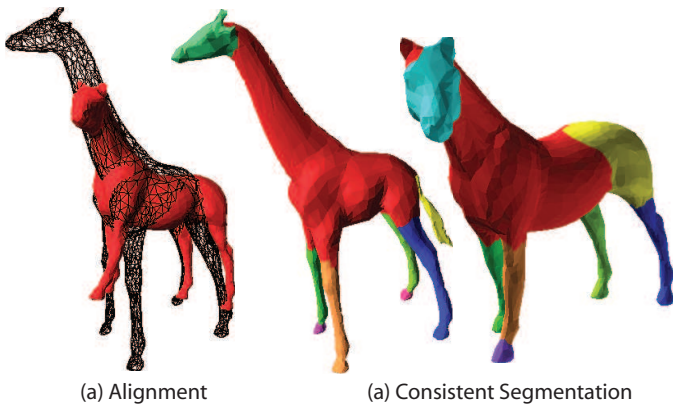

Figure 8: An example of how a poor alignment (a) hampers the resulting consistent segmentation (b). Note that due to the poor alignment, the heads of the animals are not found to correspond, the hoofs are sometimes labeled as outlying segments, and the back of the horse is found to correspond to the tail of the giraffe.

errors in correspondences. However, better alignments produce more consistent segmentations. Figure 8 shows an example of how a poor alignment can hamper the resulting segmentation. Note that because of the poor alignment, the heads of the animals are in separate segments, some of the hoofs are outlying segments, and the back of the horse is merged with the tail of the giraffe. To improve the results of such cases, it may help to consider non-rigid alignments, or part-based correspondence methods.

Third, we consider only low-level cues: adjacency and point correspondences. Better segmentations may be achieved by expanding the alignment error function to include shape signatures and contextual cues. The heuristics used in Part Analogies [27] and Shuffler [15], for example, include such terms.

Finally, our method takes as manual input the number of desired segments. Ideally, this number would be determined automatically: perhaps studying the behavior of our error function as segments are merged will suggest heuristics to automate the final number of clusters (e.g., as in [12]).

7. Acknowledgments

We would like to thank Aim@SHAPE and Viewpoint for 3D models. We acknowledge NSF (CNFS-0406415, IIS-0612231,

and CCF-0702672) and Google for providing funding to support this project.

References

[1] Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N.,and Azariadis, P. 2007.3D mesh segmentation methodologies for CAD applications. Computer-Aided Design & Applications 4, 6, 827-841.

[2] Allen, B., Curless, B., and Popović, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, ACM, New York, NY, USA, 587-594.

[3] 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.

[4] Blanz, V., and Vetter, T. 1999. A morphable model for the synthesis of 3d faces. In SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 187-194.

[5] De Goes, F., Goldenstein, S., and Velho, L. 2008. A hierarchical segmentation of articulated bodies. Computer Graphics Forum (Special Issue of Symposium on Geometry Processing) 27, 5, 1349-1356.

[6] Fellbaum, C., Ed. 1998. WordNet: An Electronic Lexical Database (Language, Speech, and Communication). The MIT Press, May.

[7] Garland, M., Willmott, A., and Heckbert, P. 2001. Hierarchical face clustering on polygonal surfaces. In ACM Symposium on Interactive 3D Graphics, 49-58.

[8] Gelfand, N., and Guibas, L. 2004. Shape segmentation using local slippage analysis. In Symposium on Geometry Processing, 214-223.

[9] Golovinskiy, A., and Funkhouser, T. 2008. Randomized cuts for 3D mesh analysis. ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) 27, 5 (Dec.).

[10] Golovinskiy, A., Podolak, J., and Funkhouser, T. 2007. Symmetry-aware mesh processing. Princeton University TR- 782-07 (Apr.).

[11] Inoue, K., Takayuki, I., Atsushi, Y., Tomotake, F., and Kenji, S. 2001. Face clustering of a large-scale cad model for surface mesh generation. Computer-Aided Design 33, 251-261.

[12] Katz, S., and Tal, A. 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. In SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, ACM, New York, NY, USA, 954-961.

[13] Katz, S., Leifman, G., and Tal, A. 2005. Mesh segmentation using feature point and core extraction. The Visual Computer (Pacific Graphics) 21, 8-10 (October), 649-658

[14] Kraevoy, V., and Sheffer, A. 2004. Cross-parameterization and compatible remeshing of 3d models. ACM Trans. Graph. 23, 3, 861-869.

[15] Kraevoy, V., Julius, D., and Sheffer, A. 2007. Shuffler: Modeling with interchangeable parts. The Visual Computer.

[16] Lai Y-K, Hu S-Mu, Martin R R, and Rosin P L. 2009. Rapid and effective segmentation of 3D models using random walks. Computer Aided Geometric Design.

[17] Leibe, B., Leonardis, A., and Schiele, B. 2004. Combined object categorization and segmentation with an implicit shape model. In In ECCV workshop on statistical learning in computer vision, 17-32.

[18] Lin, H., Liao, H., and Lin, J. 2004. Visual salience-guided mesh decomposition. In IEEE Int. Workshop on Multimedia Signal Processing, 331-334.

[19] Liu, R., and Zhang, H. 2004. Segmentation of 3d meshes through spectral clustering. In Proceedings of the 12th Pacific Conference on Computer Graphics and Applications.

[20] Min, P., Kazhdan, M., and Funkhouser, T. 2004. A comparison of text and shape matching for retrieval of online 3D models. In European Conference on Digital Libraries.

[21] Mortara, M., Patan, 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.

[22] Praun, E., Sweldens, W., and Schröder, P. 2001. Consistent mesh parameterizations. In Proceedings of ACM SIGGRAPH 2001, 179-184.

[23] Ren, X., and Malik, J. 2003. Learning a classification model for segmentation. In In Proc. 9th Int. Conf. Computer Vision, 10-17.

[24] Robbiano, F., Attene, M., Spagnuolo, M., and Falcidieno, B. 2007. Part-based annotation of virtual 3d shapes. cw 0, 427-436.

[25] Rother, C., Minka, T., Blake, A., and Kolmogorov, V. 2006. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, Washington, DC, USA, 993-1000.

[26] Schreiner, J., Asirvatham, A., Praun, E., and Hoppe, H. 2004. Inter-surface mapping. ACM Trans. Graph. 23, 3, 870- 877.

[27] Shalom, S., Shapira, L., Shamir, A., and Cohen-OR, D. 2008. Part analogies in sets of objects. Eurographics Workshop on 3D Object Retrieval.

[28] Shamir, A. 2006. Segmentation and shape extraction of 3d boundary meshes (state-of-the-art report). In Eurographics, 137-149.

[29] 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.

[30] Shlafman, S., Tal, A., and Katz, S. 2002. Metamorphosis of polyhedral surfaces using decomposition. In Eurographics 2002, 219-228.

[31] Toshev, A., Shi, J., and Daniilidis, K. 2007. Image matching via saliency region correspondences. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 0, 1-8.