Semantic-Based Segmentation and Annotation of 3D Models

Laura Papaleo* and Leila De Floriani

Department of Information and Computer Science, University of Genova
Via Dodecaneso 35, 16100 Genova, Italy
{deflo,papaleo}@disi.unige.it
http://www.disi.unige.it

Abstract. 3D objects have become widely available and used in different application domains. Thus, it is becoming fundamental to use, integrate and develop techniques for extracting and maintaining their embedded knowledge. These techniques should be encapsulated in portable and intelligent systems able to semantically annotate the 3D object models in order to improve their usability and indexing, especially in innovative web cooperative environments. Lately, we are moving in this direction, with the definition and development of data structures, methods and interfaces for structuring and semantically annotating 3D complex models (and scenes) - even changing in time - according to ontology-driven metadata and following ontology-driven processes. Here, we concentrate on the tools for segmenting manifold 3D models and on the underline structural representation that we build and manipulate. We also describe the first prototype of an annotation tool which allows a hierarchical semantic-driven tagging of the segmented model and provides an interface from which the user can inspect and browse the entire segmentation graph.

Keywords: Shape segmentation and Structuring, Semantic-driven Annotation.

1 Introduction

In the last years, the amount of digital audio-visual information is huge and rapidly increasing. This data is available within digital libraries in a number of different formats (pictures, audio and, also, 3D shapes). Among them, the importance of 3D object models is augmenting: they are playing a preeminent role in application domains such as manufacturing, science and edu-entertainment and many more. Moreover, all these application domains are recently opening their activities to the web thanks also to the development of collaborative environments. In this context, efficient and effective methods to manage these data are becoming crucial. The most common and simple representation for a 3D object consists of a mesh of triangles discretizing the object boundary. Geometric meshes, as triangle meshes, provide unstructured descriptions of object shapes which, in general, are not sufficient for reasoning on them. The

* Corresponding author.

knowledge embedded into these digital representations can be better organized by using different levels of abstractions namely, *geometry*, *structure* and *semantics* [1]. At the geometric level, topological and geometric information are explicit but no further information is encoded in the model. At the structural level, meaningful parts of the shape are described together with their connections. Finally, the semantic level associates semantics to lower levels: the association can be done manually or through an automatic semantic annotation process. In order to reason and understand a given 3D model, all the information identifiable at the three different levels must be extracted and kept.

At the state-of-the art, there is a strong request of annotation tools capable of extracting semantics from complex 3D shapes and of enhancing digital representations with context-dependent metadata. Also, moving toward the Web 3.0, Internet is becoming a universal medium for data, information, and knowledge. To be really accessible and usable, multimedia data (as 3D shapes) must be represented accordingly. In this context, we designed (and we are developing) a system, called `be-SMART`[1], for inspecting complex (manifold and non-manifold) 3D shapes (and scenes) - even changing in time - and for structuring and annotating them using ontology-driven metadata. The idea behind our work is very interesting and promising, and different other systems with basically the same goal are growing in the last years. An example is the work presented in [6] where the concept of multi-segmentation is introduced in order to semantically annotate 3D meshes. In our case, the system has been designed to address not only 3D meshes but 3D scenes, even changing in time, and the choice of the X3D encoding for models and scenes makes the system particularly suited for web applications.

The contribution of this paper is basically related to the segmentation modules of our system as to the first prototype of the module for the semantic annotation. In particular, we developed different automatic and manual segmentation techniques with the mail goal of decomposing the input 3D model into *meaningful* parts. A model can be segmented automatically, according to geometric criteria, and manually, according to the user perception. We keep the semantic-based decomposition into a structural representation, in the form of a *segmentation graph*. In this graph, the nodes identify the portions of the model and the arcs the connections among these portions as the information of the shared boundaries. Our interface, developed in Java for portability issues, supports the use of X3D language [2] for the model encoding and allows the global visualization and inspection of the segmentation graph. We can also tag each portion of the model using a hierarchical semantic-based tagging procedure by which we are able to maintain the history of the segmentation. Once semantically annotated, the portions of the model can be saved all together or separately, preparing the basis of a semantic-based modeling-by-example environment.

The reminder of this paper is organized as follows. In Section 2 we present the related work on shape segmentation, in Section 3 we introduce the underlying structural representation we use for semantic annotation. Section 4 is dedicated to the presentation of our automatic segmentation tool, while Section 5 presents our tool for manual segmenting a 3D model by painting strokes on it. Section 6 will focus on the first prototype of our annotation tool. In Section 7 some concluding remarks are drawn.

[1] BEyond Shape Modeling for understAnding Real world represenTations.

2 Related Work

In this work, we basically focus on the segmentation modules of our system, thus, we present here related works on shape segmentation, outlining which methods we used and the extensions and adaptations we performed.

Mesh segmentation is, historically, one of the most ubiquitous technical problem in 3D model understanding. The problem of partitioning a 3D model into *meaningful pieces* has proved to be as difficult as other computational problems that attempt to mimic the capabilities of human perception. Shape understanding and semantic-based object representations rely on the segmentation of 3D meshes that represent objects. Shape segmentation techniques borrow from related fields, such as image segmentation, unsupervised machine learning and others. A complete survey of existing techniques can be found in [4]. The segmentation problem can be formulated as an *optimization problem*. Given a mesh M and the set of elements $S = \{V, E, F\}$, vertices, edges and faces of M respectively, the problem of segmenting M is equivalent to the problem of finding a disjoint partitioning S into $S_0, \ldots, S_{k-1}$ such that a given *criterion function* $J = J(S_0, \ldots, S_{k-1})$ is minimized (or maximized) under a set of specific constraints C. Shamir [4] provides a classification of the existing segmentation techniques into *part-type segmentations*, which segment an object into volumetric parts [18,24], and *surface-type segmentations*, which partition the boundary of the object into meaningful patches [20,7,10,23,5].

Also, different manual or user-guided segmentation techniques have been proposed in the literature [14,26,21,13,19,25,22,8,17] and they are rooted in the expert perception of the object. In this sense, the regions of interest are intrinsically guided by the semantic the user *recognizes* in the object model. In opposition to the automatic segmentation techniques, developed for a specific application context (e.g. CAD/CAM or biomedicine), manual segmentation methods are general purpose. We can divide the existing manual segmentation techniques into two main categories: *cut-based* methods and *region-based* methods. Algorithms belonging to the first category [26,21,13,19] allow the user to draw the cutting path on the model while those which belong to the second category let the user select the interesting regions, and they automatically compute the right position of the cut [25,22,8,17]. In [13], a so-called intelligent scissoring of 3D meshes has been presented in which the user paints *strokes* on the mesh surface to specify where cuts should be made.

For the automatic segmentation procedure, we used the method in [10] and a modification of the clustering method presented in [20] for the automatic region merging (Section 4). The manual procedure (Section 5), instead, has been implemented as an extension of the method described in [13] in order to deal also with non-triangular and not connected meshes, using in toto the expressiveness power of X3D encoding.

3 Underlying Structural Representation for Semantic Annotation

As we said before, our main objective is to develop a general framework for structuring and semantically annotating these complex shapes for improving their usability, indexing and retrieval. Semantic annotation can be performed if a structural representation underlying the input model (or scene) is used. In our case, the structural description has been defined as a *two-level graph* representation able to capture information in case of both manifold and non-manifold conditions. The first level, in case of non-manifold models, is a description of the decomposition of the shape into manifold components. Their structural interconnection is represented as a graph, that we call the *decomposition graph*, described as a hypergraph $H = (N,A)$ in which the nodes correspond to the components in the decomposition, while the hyperarcs catch the structure of the connectivity among the components or are self-loops. In this latter case, they represent the *non-manifold structure* of a single component and correspond to the duplicated non-manifold vertices and edges in the component. In the two-level graph representation, each manifold component (if further segmented) is structured in a *segmentation graph*. In this graph - used in the tools presented in this paper - the nodes identify the portions of the model and the arcs the connections among these portions. Formally, in the segmentation graph G, a node n_i represents a patch C of the input shape and an arc $a = (n_i, n_j)$ represents the adjacency relationship between the patches C_i and C_j and, thus, their shared boundary.

The two-level segmentation graph is also the *core* of a complex framework, called `be-SMART`, we designed for bridging Semantic Web technologies and shape analysis [11]. `be-SMART` has the objective to semantically annotate complex 3D shapes (even non-manifold meshes) using domain-specific ontologies and following domain-specific ontology-driven processes. Here, we present our first results regarding three main modules of `be-SMART`, namely the *Manual and Automatic Segmentation* modules (Section 4 and Section 5) for the manifold parts and a first prototype of the *Semantic Annotator* (Section 6) which adds semantic information to the recognized parts directly in the X3D model subparts. Thus, we will focus only on the updates and operations on the *segmentation graph G*, described above. In particular, in our implementation, G is represented using a standard adjacency list with additional information (also on the arcs) necessary for the clustering and the manual segmentation methods.

4 Combining Automatic Segmentation Methods

In this section we describe how we automatically segment a manifold 3D model. Specifically, we combine a segmentation and a clustering technique, namely the Variational Shape Approximation (VSA) presented in [10] and a clustering method inspired by [20]. In general, the system has been designed to the modular, thus allowing to import new automatic segmentation techniques, once implemented.

Given a manifold component M, the VSA algorithm segments M according to planarity criteria. Let R be a partition of M (initially the set of triangles in M). The idea is that every region $R_i \in R$ can be represented by a pair $P_i = (X_i, N_i)$, where X_i is the average center of R_i and N_i is the average normal. P_i is called a *shape proxy* of the region R_i. Thus, for any given partition of a surface in k regions, there is a set $P = \{P_i\}, i = 1 \ldots k$, of shape proxies that approximate the whole geometry. Partition R defines a *dual metamesh* [10] of the original: every shape proxy is a meta-face and the connectivity of the regions R_i defines the topology of the new mesh. The VSA algorithm first creates the set of seeds for activating a region-growing process. Then, from every seed s_i, a region R_i

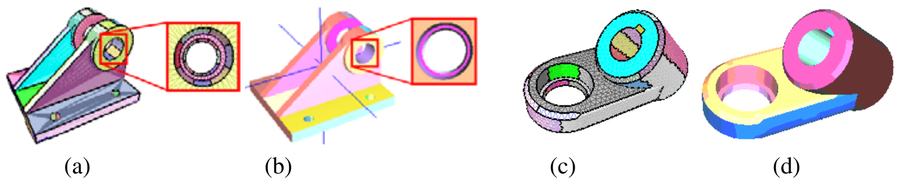


Fig. 1. (a)-(c) Segmentations of CAD models using the VSA method. Flat regions are segmented into single regions and cylindrical surfaces are segmented with many striped regions. (b)-(d) Refined segmentations via the clustering algorithm: cylindrical surfaces are partitioned with fewer regions.

is grown on the basis of specific conditions. In the process, the algorithm uses a priority queue Q, where the triangle T_i priority is equal to a specific distortion error E_{T_i} When Q is empty, the partition R is created. Successively, a fitting procedure is applied: for every region $R_i \in R$ the corresponding proxies P_i are updated in order to minimize the error $E(R_i, P_i)$. The clustering algorithm we have developed (fully described in [12]), extending and adapting the idea in [20] allows the refinement of the obtained segmentation by merging parts according to specific geometric conditions. It uses the *segmentation graph* G (Section 3): first, it considers every face of M as a patch C_i and then, at every step, it merges pairs of patches (C_i, C_j) creating a new patch C_{ij}. The choice of the patches to be merged is done according to *weights* assigned to the boundary of each patch. These weights are computed by combining geometric properties of the edges and faces sharing the boundaries. For example, the algorithm prevents the union of patches with *sharp corners* between them and avoids the existence of small patches. The clustering algorithm is applied by simply performing a *sequence of contractions* of the arcs in the graph G. The merging of a pair of patches C_i and C_j is done by contracting the corresponding arc $a = (n_i, n_j)$ in the graph G. This operation removes arc (n_i, n_j) from G, merges nodes n_i, n_j into a new node n_{ij} and transform the arcs incident in n_i and in n_j into arcs incident in node n_{ij}, inheriting the conditions and weights of the arcs and nodes involved. The result is a segmentation graph with the nodes corresponding to larger patches (shape regions). Figure 1(a) and (c) show two segmentations obtained by the VSA method for CAD (manifold) models. Flat surfaces are segmented as single regions, while cylindrical surfaces are segmented with many striped regions. Figure 1 (b) and (d) show the refined segmentations obtained by the application of our clustering approach: cylindrical surfaces are partitioned into fewer regions. Our tool, developed in Java, allows the user to perform a given number of step of the clustering algorithm and to assign specific weights to each geometric constraints to be considered in the computation of the weight W_a assigned to each border a of the segmentation. This is done via a user-friendly and portable interface.

5 Manual Segmentation by Strokes Painting

While automatic segmentation techniques - as those presented in the previous section - can be fast and precise, in some cases, the intervention of the user is fundamental, since modeling the human perception into an automatic system is still a open issue. For this reason, we believe that a combination of automatic and manual segmentations can be a real support to the user, when he points to semantically annotate a 3D model according to different criteria. These criteria can be *objective* (thus, based on specific, computable measures) and *subjective*, namely dependent on the user personal perception. The main goal is that the system can be able to associate different metadata to each patch of the model, allowing a hierarchical semantic organization of the identified portions.

In this sense, taking inspiration from the intelligent scissoring operation presented in [13], we developed a manual segmentation tool which enable the user to paint strokes on the mesh surface. By painting these strokes on the *visible* part of the model, the user specifies where cuts should be made and the system automatically computes the entire cut, splitting the selected portion of the model into two pieces and updating the *segmentation graph* accordingly. The general idea of the original method in [13] is the following: the user draws a stroke on the surface model. It has a specified width (r) representing a region of uncertainty within which the method should construct the cut, considering cuts along edges of the mesh. The stroke starts in a region (in which the algorithm selects an *initial* vertex a) and ends in another region (in which a *final* vertex b is selected). Successively, two edges paths (minimum paths) are computed: one front-side, involving edges touched by the stroke, and the other back-side. This last is the minimum path formed by edges *not visible* from the view point connecting the initial and the final vertices. Each minimum path is computed by the use of the Dijkstra's algorithm with a cost function which depends also on geometric properties of each edge involved (e.g. dihedral angles).

In our case, since X3D allows to define shapes using vertices and not edges, the use of an implementation of the original approach would have been computationally expensive. Every time the user would draw the stroke, the system should compute the intersection between a circle of radius r and all the edges present in the scene and projected on the viewplane. We decided to apply the algorithm on the vertices of the model. This change does not modify the general methodology, but it allows us to reduce the number of operations to be performed. With our choice, in fact, we do not have to compute intersections (solving linear systems) but only euclidean distances. Additionally, we have been able to extend the procedure to surface meshes which are not represented only by triangles, using in this way all the faces types defined by the X3D standard. Furthermore, we extended the method to special cases: we can treat the case in which, given a stroke (as a set of circles) there is no vertex inside it connecting the initial and final vertex of the stroke: each time we cannot find a connection inside the stroke, we search for the nearest point in the surface model and we let the path passing from it. This procedure solves also the case in which, given the stroke, the system cannot find an initial and/or final vertex. For automatic computation of the cut in the not visible part of the model, we improved the original method restricting the search of the connections to a subset of vertices. For doing this we compute the visibility of each vertex before performing the cut.

At structural level, our manual segmentation method performs an update operation on the *segmentation graph* G (Section 3), involving the region (node) C the user has cut. The graph is modified by creating two new nodes C_1 and C_2 and by eliminating the node C. The arcs having in C an extreme are updated accordingly. We will show in the

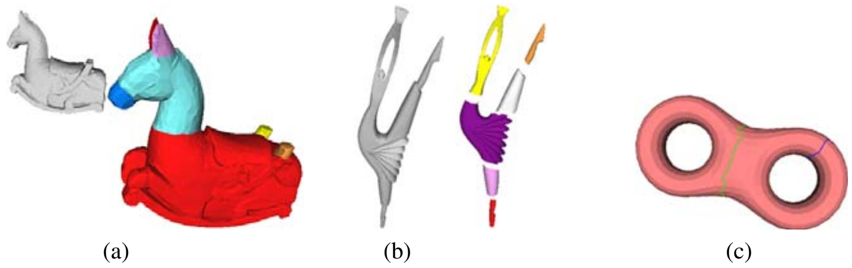


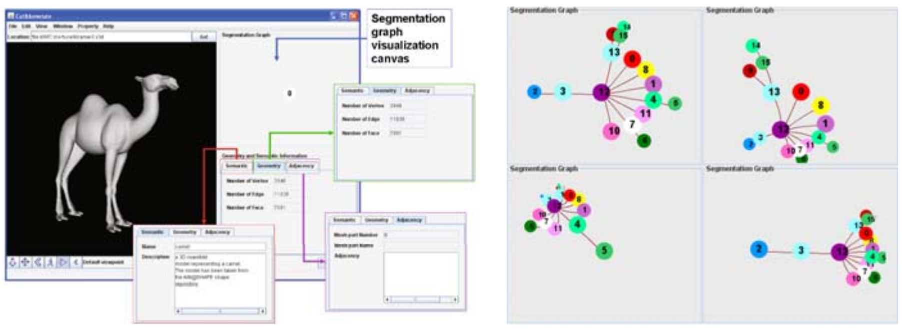
Fig. 2. (a)-(b) Segmentations obtained with our manual tool. (c) a case in which our tool cannot produce feasible results (blue stroke).

next section how the hierarchical semantic tagging, allows to maintain the history of the segmentation. In Figure 2 (a) and (b) we show two examples of segmentation obtained with our manual segmentation tool, while Figure 2 (c) shows a special case in which a complete cut cannot be found. In this case, the blue cut does not divide the model into two separate portions and multiple strokes should be used. We are working in order to support this type of functionality.

6 Global Inspection and Semantic-Based Hierarchical Tagging

As we mentioned in Section 3, the underlying structural representation for the segmentation procedures, performed on manifold models (or portions of models), is a graph that we called the *segmentation graph*. In the previous sections, we described also the updating operations we implemented in case of both the automatic and manual segmentation methods. Here, we present the interface we developed for inspecting and browsing the segmentation graph which is the basis for the semantic-based hierarchical tagging. Figure 3-(a) shows the overall interface. It has been implemented in Java extending the Xj3D browser [3] devoted to the visualization of X3D models. On the left, we have the canvas for model visualization and inspection and, on the right, we have placed the entire segmentation graph and the fields for semantic annotation. In particular, three different working tabs have been designed: for each node C of the segmentation graph G, one tab collects the geometrical information, automatically extracted (*geometry*); another tab (*adjacency*) describes the adjacency information, again automatically extracted. The last working tab (*semantic*) is, instead, devoted to the user-defined semantic annotation.

We have implemented a simple semantic tagging which allows the user to add information to a region. Every segmented region (node) C will contain in the name also the names of its ancestors with the following syntax: $ancestor_1 : ancestor_2 : \ldots ancestor_n :$ *RegionName*. In this way, we are able to trace the entire segmentation process and we obtain two interesting results. On the one hand, using our tags - organized hierarchically - we are able to re-merge the segmented regions simply by checking the names of the regions and by merging their faces and vertices. On the other hand, looking at the name of a given region C we can access immediately to its history.



(a) (b)

Fig. 3. (a) the overall interface for the manual segmentation. On the right the model, on the left the layer for visualizing the segmentation graph as for the semantic annotation. (b) an example of the global visualization and browsing on a complex segmentation graph. All the nodes are visible and the focus is on a specific node.

For the visualization of the segmentation graph, we decided to show it globally (Figure 3-(b)), where every node has a specific color (the same of the associated region) and a number. The user can browse the graph very intuitively. By clicking on a node C, the associated region in the Xj3D canvas will be highlighted and C will become the new visualization center of the hyperbolic graph. Also, the user can drag a node and the graph will change its shape accordingly. All the nodes will be always visible, but the focus will be on the active node and the nearest ones. Finally, when a node C is selected, all the associated information are shown in the working tabs described before.

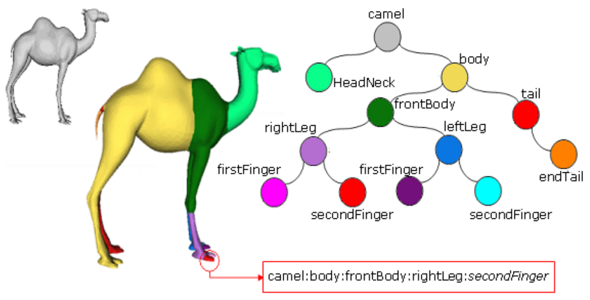


Fig. 4. A 3D segmented model representing a camel and the name given to a specific region, following our hierarchical tagging. In this case the second finger of the frontal right leg.

Figure 4 shows an examples of 3D model, its segmentations obtained via our tool and the associated segmentation graph. The interface has also some other functionalities. The segmentation can be rendered in an *expanded* or *not expanded* way, as shown in the examples depicted in Figure 2(a) and (b). Also, the user can save the entire segmented and annotated model in X3D format (using the tag `Metadata` and `MetadataSet`) or can select a portion of it, and saves it separately.

7 Concluding Remarks

In this paper, we presented different segmentation methods we have implemented in order to recognize meaningful portions of a 3D model. We focused also on how we maintain the decomposition into a structural representation and the updating operations we perform on it. Our interface allows the user to inspect both the segmented model and the segmentation graph. This has been done in order to support the user in the understanding of the overall model and to guide the user in the semantic annotation of each portion. We presented also our semantic-based hierarchical tagging by which we are able to maintain the history of the segmentation procedure. Once semantically annotated, the portions of the model can be saved all together or separately, preparing the basis of a semantic-based modeling-by-example environment. This work is part of a more complex Semantic Web system for inspecting complex 3D shape and for structuring and annotating them according to ontology-driven metadata.

Our future directions are multiple. For the automatic segmentation module, we are developing different other procedures, in order to cover several types of geometric constraints. The manual segmentation framework will be extended in order to solve the problem we showed in Section 5. Additionally, we are planning to implement innovative methods as those presented in [16,15,9] where the strokes are painted on the surface to identify meaningful portions (not the cut), and the system will automatically compute the right cut (according to the minimal rule).

The semantic annotator we presented is the basis of the annotation using ontological schema. We are actually working in order to interface our system with pre-defined ontologies. In this case, the semantics associated to the portions of the model will be saved also in separated RDF files, thus allowing to experiment the complete power of Semantic Web technologies.

Acknowledgments. This work has been partially supported by Project FIRB *SHALOM* funded by the Italian Ministry of Education, Research and University n.$RBIN04HWR8$. The authors thank Matteo Bertucelli and Nicoló Carissimi for the support in implementation. Some shapes in the paper are from the AIM@SHAPE shape repository [1].

References

1. The European Network of Excellence AIM@SHAPE - contract number 506766 (2004-2007), www.aimatshape.net
2. The web3d consortium x3d working group (2008), http://www.web3d.org/x3d
3. The xj3d project (2008), http://www.xj3d.org
4. Shamir, A.: A survey on mesh segmentation techniques. In: Computer Graphics Forum (2008)
5. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. The Visual Computer 22(3), 181–193 (2006)
6. Attene, M., Robbiano, F., Spagnuolo, M., Falcidieno, B.: Semantic annotation of 3d surface meshes based on feature characterization. In: Falcidieno, B., Spagnuolo, M., Avrithis, Y., Kompatsiaris, I., Buitelaar, P. (eds.) SAMT 2007. LNCS, vol. 4816, pp. 126–139. Springer, Heidelberg (2007)

7. Boier-Martin, I.M.: Domain decomposition for multiresolution analysis. In: Proc. of the EG Symposium on Geometry Processing, pp. 31–40. Eurographics Association (2003)
8. Brown, S.W.: Interactive part selection for mesh and point models using hierarchical graph-cut partitioning. Brigham Young University, PhD Thesis (2008)
9. Brown, S.W.: Interactive part selection for mesh and point models using hierarchical graph-cut partitioning. Brigham Young University, PhD Thesis (2008)
10. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. ACM Trans. Graph. 23(3), 905–914 (2004)
11. De Floriani, L., Hui, A., Papaleo, L., Huang, M., Hendler, J.A.: A semantic web environment for digital shapes understanding. In: Falcidieno, B., Spagnuolo, M., Avrithis, Y., Kompatsiaris, I., Buitelaar, P. (eds.) SAMT 2007. LNCS, vol. 4816, pp. 226–239. Springer, Heidelberg (2007)
12. De Floriani, L., Papaleo, L., Carissimi, N.: A java3d framework for inspecting and segmenting 3d models. In: Proc. of the International Conference on 3D Web Technology (Web3D), pp. 67–74. ACM, New York (2008)
13. Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., Dobkin, D.: Modeling by example. ACM Transactions on Graphics 23(3), 652–663 (2004)
14. Gregory, A.D., State, A., Lin, M.C., Manocha, D., Livingston, M.A.: Interactive surface decomposition for polyhedral morphing. The Visual Computer 15(9), 453–470 (1999)
15. Huaiyu, C., Jia, Q., Songde: A sketch-based interactive framework for real-time mesh segmentation. In: Proc. of the Computer Graphics International (2007)
16. Zhongping, J., Ligang, L., Zhonggui, C., Guojin, W.: Easy mesh cutting. Computer Graphics Forum 25(3), 283–292 (2006)
17. Lai, Y.-K., Hu, S.-M., Martin, R.R., Rosin, P.L.: Fast mesh segmentation using random walks. In: SPM 2008: Proceedings of the 2008 ACM symposium on Solid and physical modeling, pp. 183–191. ACM, New York (2008)
18. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. In: ACM SIGGRAPH Proceedings. ACM, New York (2003)
19. Sharf, A., Blumenkrants, M., Shamir, A., Cohen-Or, D.: Snappaste: an interactive technique for easy mesh composition. The Visual Computer 22(9-11), 835–844 (2006)
20. Sheffer, A.: Model simplification for meshing using face clustering. Computer-Aided Design 33(13), 925–934 (2001)
21. Weyrich, T., Pauly, M., Heinzle, S., Keiser, R., Scandella, S., Gross, M.: Post-processing of scanned 3d surface data. In: Symposium on Point-Based Graphics, pp. 85–94 (2004)
22. Wu, H.-Y., Pan, C., Pan, J., Yang, Q., Ma, S.: A sketch-based interactive framework for real-time mesh segmentation. In: Proceedings of the Computer Graphics International, CGI (2007)
23. Wu, J., Kobbelt, L.: Structure recovery via hybrid variational surface approximation. Computer Graphics Forum 24(3), 277–284 (2005)
24. Lee, Y., Lee, S., Shamir, A., Cohen-Or, D., Seidel, H.P.: Mesh scissoring with minima rule and part salience. Comput. Aided Geom. Des. 22(5), 444–465 (2005)
25. Zhongping, J., Ligang, L., Zhonggui, C., Guojin, W.: Easy mesh cutting. Computer Graphics Forum 25(3), 283–292 (2006)
26. Zöckler, M., Stalling, D., Hege, H.-C.: Fast and intuitive generation of geometric shape transitions. The Visual Computer 16(5), 241–253 (2000)