Eurographics Italian Chapter Conference (2010)
E. Puppo, A. Brogni, and L. De Floriani (Editors)

Manual Segmentation and Semantic-based Hierarchical Tagging of 3D models

Laura Papaleo[1,2] and Leila De Floriani[1]

[1]Department of Computer Science, University of Genova, Italy
[2]IT Department, Province of Genova, Italy

Abstract

Today 3D objects have become widely available in different application domains, thus it is becoming fundamental to use, integrate and develop techniques for extracting and maintaining their implicit knowledge. These techniques should be encapsulated in intelligent systems able to semantically annotate the 3D models, thus improving their usability and indexing, especially in innovative web cooperative environments. In our work, we are moving in this direction, by defining and developing data structures, methods and interfaces for structuring and semantically annotating 3D complex models (and scenes), even changing over time, according to ontology-driven metadata. In this paper, we focus on tools and methods for manually segmenting manifold 3D models and on the underline structural representation that we build and manipulate. We present also an interface from which the user can inspect and browse the segmentation, describing also the first prototype of an annotation tool which allows a hierarchical semantic-driven tagging of the segmented model.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computer Graphics—Computational Geometry and Object Modeling Curve, surface, solid, and object representations; [H.5.1]Multimedia Information Systems;

1. Introduction

Nowadays, multidimensional data (pictures, audio and, also, 3D shapes) are available within digital libraries and they are used and shared by wide communities. Among them, the importance of 3D object models is increasing: they are playing a preeminent role in application domains such as manufacturing, science, 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. Geometric meshes, as for example triangle meshes, provide unstructured descriptions of object shapes which, in general, are not sufficient for reasoning on them. The knowledge embedded into these digital representations can be better organized by using different levels of abstractions namely, *geometry*, *structure* and *semantics* [aim07]. 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 [DHP*07,ARSF07,HG10, MSS10]. Also, with the advent of the Semantic Web [BLHL01] and moving toward the *Web 3.0* [LH07, Hen08], 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. The Semantic Web envisages technologies, which can make possible the generation of "intelligent" multimedia content. We can define an intelligent multimedia content as a content which "*knows about*" its own meaning in order that automated processes can "*know what to do*" with it. The

DOI: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2010/025-032

Semantic Web proposes annotating document content using semantic information. The result is multimedia content with machine interpretable mark-up that provide the source material with which agents and Semantic Web services operate, thus creating annotations with well-defined semantics.

In this context, we designed (and we are developing) a framework for inspecting complex (manifold and non-manifold) 3D shapes (and scenes) - even changing over time - and for structuring and annotating them using ontology-driven metadata. The contribution of this paper is related to the manual segmentation module as to the semantic annotator of our Semantic Web framework. In particular, we improved our previous manual segmentation tool presented in [DPC08] with a more powerful segmentation method with the main goal of decomposing the input 3D model into *meaningful* parts. A model can be in this way segmented 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 system has been developed in Java and supports the visualization and editing of 3D surface models described in X3D [x3d08]. Our interface allows the global visualization and inspection of the segmentation graph, rendered as an hyperbolic graph. For the semantic annotation, we developed 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, thus preparing the basis of a semantic-based modeling environment.

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

2. Related Work

Different manual or user-guided segmentation techniques have been proposed in the literature and they are rooted in the expert perception of the object [GSL*99, ZSH00, WPH*04, FKS*04, SBSCO06, ZLZG06, WPP*07, Bro08, LHMR08]. 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 (see [Sha08] for a complete survey), which have been usually 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 [GSL*99, ZSH00, WPH*04, FKS*04, SBSCO06] 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 [ZLZG06, WPP*07, Bro08, LHMR08].

The most simple cut-based manual segmentation method is the one who allows the user to iteratively select vertices, edges or faces on the input model, drawing in this way the cut directly on the model. This approach, is effective but very time-consuming. Additionally, in case of complex shapes, the selection of a vertex can be almost impossible due to occlusions. Another idea is to use specific primitives in order to isolate portions of the input model: in this case, the cutting path is defined by the set of model components which intersect the boundary of the chosen primitive. The most simple primitive is the plane: drawing a line in the scene, a plane is computed which cuts the input model into two pieces. In this case, the main problem is that the chosen plane depends on the view point and the cutting can produce undesirable results. In [WPH*04] different primitives are used, such as cylinders or cubes. In [BS01] a cut-based manual segmentation method is proposed, in which a cut can be directly drawn on a triangle mesh, thus simulating the use of a *scissor*. Some other methods allow the user to select a sequence of vertices on the surface mesh and cut along the shortest paths between them [GSL*99, ZSH00]. However, these methods present the necessity to rotate the mesh during the selection. To solve this problem, there exist different approaches which use a sort of automatic completion of the cutting path. In [SBSCO06], for example, the user paints a stroke near the desired position of the cut. The positions of the mouse are projected onto the surface and a set of faces are created to represent the stroke. The method computes automatically the invisible part of the cut, by adjacency using Dijkstra's algorithm. A similar approach, from which we took inspiration, is the one presented in [FKS*04], where the so-called *intelligent scissoring* of 3D meshes is described (see Section 5 for more details).

Following a complementary idea, the *region-based* methods let the user choose the *meaningful* portions of the model by drawing scribbles or points on them and they compute automatically the cut using different heuristics. In this case, the user, instead of concentrating his attention on the cut, is more focused on the regions he wants to identify. In [ZLZG06], taking inspiration from a image segmentation method presented in [BJ01], a 3D model region-based manual segmentation method is presented. Here, the user selects with a *foreground* scribble the region to be cut, and with the *background* scribble the remaining part of the model. Once the scribbles have been drawn, the method automatically computes the two associated regions. The method works only for triangle meshes and it allows to draw only two scribbles at a time. In [WPP*07] an extension of the method in [ZLZG06] is presented, in which the user can define multiple scribbles. An improvement in terms of time efficiency of the method in [ZLZG06] has been presented recently in [Bro08]. Here, the entire segmentation process is performed using a hierarchical acceleration procedure which works on the structural graph representing the model, using an octree to compress the quantity of information. Another recent method, presented in [LHMR08], is a manual approach which aims at segmenting triangle meshes by taking inspiration from the image segmentation method presented in [Gra06] and by using a method, initially developed for mesh noise removal, presented in [SRML07]. In [LHMR08] the user selects n faces, $s_1, \ldots, s_n$ (called *seeds*), where n will be the number of regions the user wants to identify. The method assigns a probability value to each edge of each face in the model and proceeds in order to assign a label to each face. In particular, a face f will belong to the region denoted by the seed s_i, if the *random path* starting from f will have higher probability to reach s_i than any other seed chosen by the user. By using different probability distributions, the method can be used to segment different kinds of models. Obviously the main problem in this case, is to find, every time, the optimal probability distribution.

3. The Structural Representation for Semantic Annotation

As we said before, we are working in the direction to develop and implement data structure, methods and interfaces to deal with complex (manifold and non-manifold) 3D shapes and scenes, even changing over time. 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 (Figure 1). 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* ((Figure 2). 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.

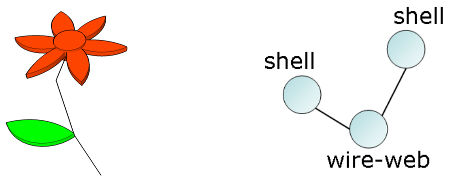


Figure 1: *The two-level decomposition graph. A non-manifold model representing a flower and the relative decomposition graph. Three component are identified - two shells (the flower and the leaf, respectively) and a wireweb (the stalk).*

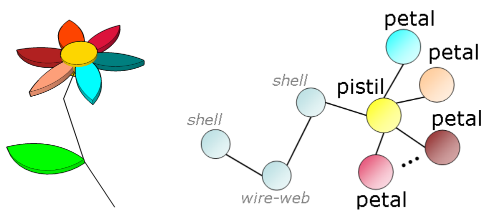


Figure 2: *The two-level decomposition graph. The non-manifold model representing the same flower of Figure 1 further decomposed and the relative segmentation graph.*

The two-level segmentation graph is also the *core* of a complex framework, we designed for bridging Semantic Web technologies and Shape Structuring and Analysis [DHP*07]. In this work, we present two modules of the framework namely the *Manual Segmentation* module (Section 5) and an initial prototype of the *Semantic Annotator* (Section 6) for the manifold parts. Thus, we will focus 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 performing the manual segmentation.

In a more general view, the entire two-level graph representation represents the basis for the semantic annotation which will be performed attaching specific information to the nodes of the graph (geometric components of the model). The semantic meaning will be added by following conceptual schema previously defined in domain-specific ontologies. The chosen domains will guide the annotation, thus allowing a multi-level semantic tagging where each geometric element of the input model will possibly have different meanings, depending on the annotation procedure.

4. Previous Approach and its Limitations

In [DPC08], a Java3D framework is presented, able to combine automatic segmentation and merging techniques for the analysis of manifold objects represented by triangle meshes in X3D format. Two well known partitioning techniques [CSAD04, She01] have been used adapting them to the developed framework. In the work it has been proved that, by combining different automatic techniques good quality segmentations can be obtained, in specific application domains. However, while automatic segmentation techniques 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 an open issue. We believe, in fact, 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 direction, in [DPC08], the framework has been extended with a simple manual editing functionality. Basically, a basilar cut-based manual segmentation method has been implemented, which allows the user to split a selected region of the model into two new regions by manually drawing a cutting path on the model. This path is built by iteratively selecting adjacent vertices on the input model. This functionality has been proved to be a good support, in combination with the automatic techniques implemented. However, it is quite complex to be used. For example, in case of regions with complex shape, it forces the user to rotate the selected region in order to reach all the necessary vertices. This paper presents our results in the design and implementation of an advanced cut-based manual segmentation tool which overcomes the above limitations (see Section 5).

Also, in [DPC08] an initial prototype of a tool for inspecting both the segmented model and the produced segmentation graph has been described. With this tool, the user can navigate the graph discovering geometrical properties of each region of the model. Once a region has been selected, it is possible to visualize a graphical representation of the corresponding node in the segmentation graph. The main limitation is that the user cannot have a *global* vision of the segmentation graph, but only a focus on the selected portion (graph node) and all the adjacent regions (graph nodes). This limits, in some sense, the possibility to be a real support in the reasoning. In this work, we present a new segmentation graph visualization and inspection tool (Section 6).

5. Manual Segmentation by Strokes Painting

We developed a cut-based manual segmentation tool which enables 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.

We took inspiration from the *intelligent scissoring* operation presented in [FKS*04]. The general idea of the original method 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. Cuts are considered 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 using the Dijkstra's algorithm with a cost function which depends also on geometric properties of each edge involved (e.g., dihedral angles).

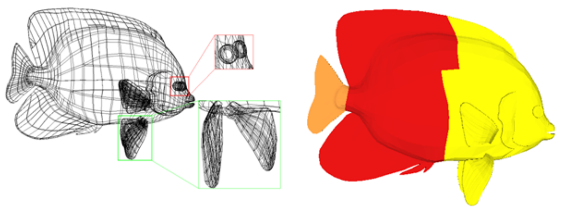


Figure 3: *A 3D model representing a fish. As shown in the wireframe visualization, the model is not triangular. Moreover it is composed by different connected components. On the right the segmentation produced with our tool is depicted.*

In our case, since X3D allows to define object models using vertices and not edges, the direct 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. In this way, we do not have to compute intersections (solving linear systems) but just Euclidean distances. Moreover, we have been able to extend the procedure to surface meshes which are not necessarily triangular, using all the faces types defined by the X3D standard, as in the case of the 3D model showed in Figure 3. Figure 4 shows a 3D model, describing a duck, and the relative cut. On the left, the set of circles identify the stroke on the model, while on the right, the two created regions are showed.

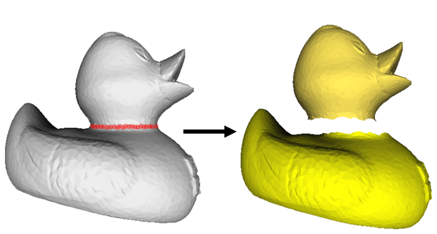


Figure 4: *A X3D 3D model representing a duck. On the left - before the cut - the set of red circles are the input for computing the visible part of the cut. On the right - after the cut - the two new regions are showed.*

Furthermore, we have 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 a final vertex, as the case showed in Figure 5. In this case, taking the center of the circle, the nearest (in terms of euclidean distance) visible vertex point of the model will be chosen.

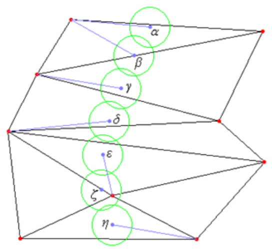


Figure 5: *A case in which the given stroke cannot find an initial vertex inside itself (circle η). In this case the nearest vertex is the red one on the left, which is the nearest (in terms of euclidean distance) visible vertex point.*

For automatic computation of the cut in the not visible part of the model, we have 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. The entire approach can be summarized as follows. Starting for the X3D input model, the user determines the best view for the cut. The system computes the visibility for each vertex in the model, the user draws the stroke on the model (visible part) and the system computes the visible and not visible cutting paths. At the end of the process, it creates the two new regions and update the segmentation graph accordingly. At the structural level, the system performs an update operation on the *segmentation graph* G (see 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.

In Figure 6 we show some examples of segmentations obtained with our manual segmentation tool. In Figure 7 two special situations in which a complete cut cannot be found are depicted. In these cases, those cuts which do not (potentially) divide the models into two separate portions cannot be used, since multiple strokes are necessary for dividing the models. We are working in order to support this type of functionality.

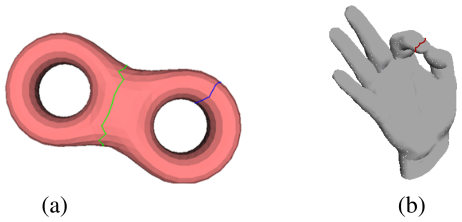


Figure 7: *Two cases in which our tool cannot produce feasible results. The blue stroke in (a) and the red one in (b).*

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 section, we described the updating operations we implemented in case of the manual segmentation method. 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 8 shows the overall interface. It has been implemented in Java extending the Xj3D browser [xj308] 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 (Figure 8, on the right): 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 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: $ancestor_1 : ancestor_2 : \dots ancestor_n : RegionName$.
In this way, we are able to trace the entire segmentation process and we obtain two interesting results. On the one hand,

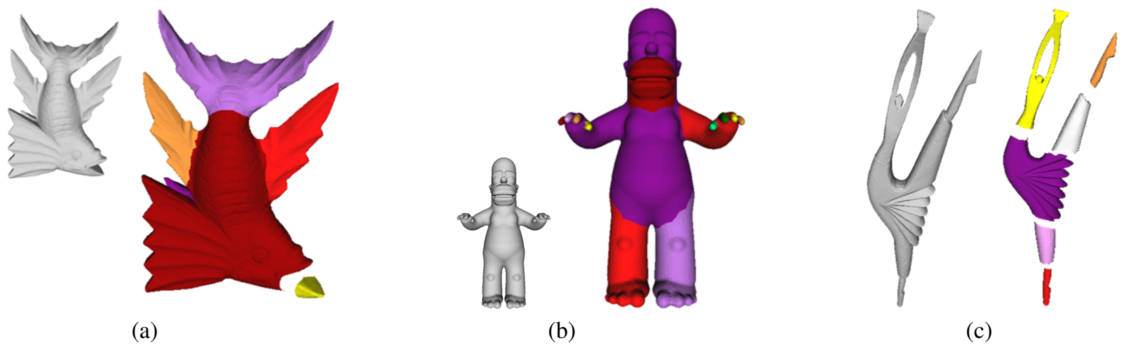
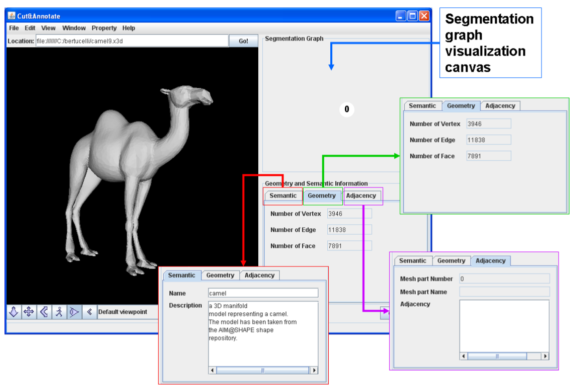


Figure 6: *(a) A 3D model representing a fish and the related segmentation. (b) a 3D model representing Homer Simpson and the related segmentation. (c) a 3D model representing a dancer and the related segmentation.*


Cut&Annotate
File Edit View Window Property Help
Go!
Segmentation Graph
Segmentation graph visualization canvas
0
Geometry and Semantic Information
Semantic Geometry Adjacency
Default viewpoint

Location: file:///////C:/bertucelli/camel9.x3d

Number of Vertex 3946
Number of Edge 11838
Number of Face 7891

Name camel
Description a 3D manifold model representing a camel. The model has been taken from the AIM@SHAPE shape repository.

Mesh part Number 0
Mesh part Name
Adjacency

Figure 8: *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.*

using our tags - organized hierarchically - we are able to remerge 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. Figure 9 shows a 3D segmented model representing a dinosaur and the name given to a specific region, following our hierarchical tagging.

For what concern the visualization of the segmentation graph, we decided to visualize it globally as an hyperbolic graph (Figure 10), where every node has a specific color (the same of the associated region) and a number, which links the node to the related portion of the model. The user can browse the graph very intuitively. By clicking on a node *C* in the graph, the associated region in the Xj3D canvas will be highlighted and *C* will become the new center for the visualization of the hyperbolic graph (Figure 10). 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.

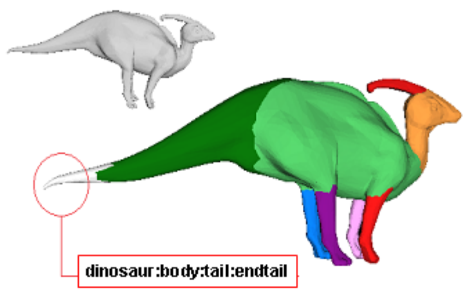


Figure 9: *A 3D segmented model representing a dinosaur and the name given to a specific region, following our hierarchical tagging. In particular, the white region has been named* `endtail`*. The name associated to the region maintains the textual information of its ancestors, namely* `tail`*,* `body` *and* `dinosaur`*.*

Figure 11 shows two examples of 3D models, their segmentations obtained via our tool and the associated segmentation graph, where for simplicity only the real name of the portion has been written (without hierarchy). Also, for each model, a specific portion has been selected and the Figure highlights the relative hierarchical tag. Our tool has also some other interesting functionalities. First of all, it allows the visualization of the model also in wire-frame. Second, the segmentation can be rendered in an *expanded* or *not expanded* way, as shown in the examples depicted in Figure 6. Ultimately, 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 the model, saving it separately with all the associated semantic information. In Table 1 a portion of an X3D file describing a segmented region is shown. The code highlights the X3D tags used to encode the semantic information.

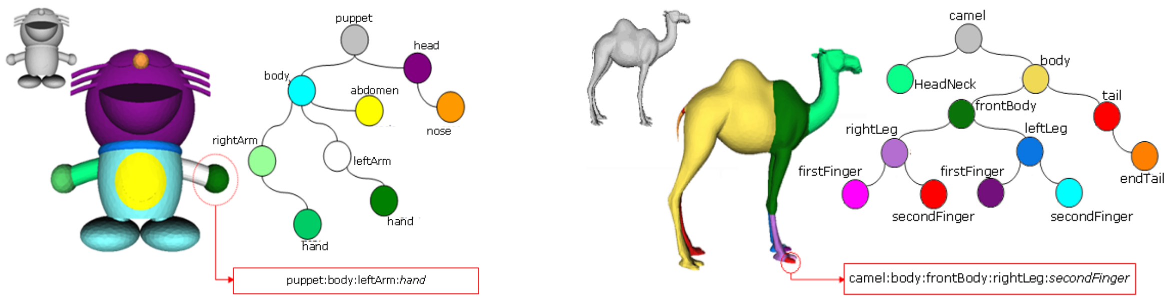


Figure 11: *(a) a 3D model representing a puppet, its segmentation and the relative segmentation graph. A portion representing the left hand is selected and the relative hierarchical tag is shown. (b) a 3D model representing a camel. as for the puppet, the segmentation is depicted and the focus in on a specific region. In this case the second finger of the frontal right leg.*

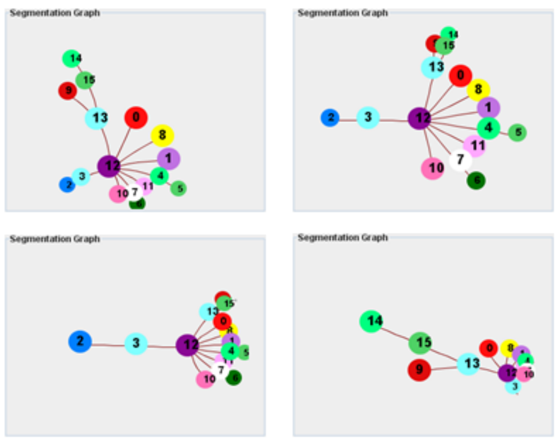


Figure 10: *An example of the global visualization and browsing on a complex segmentation graph. All the nodes are visible and the focus is on a specific node.*

Table 1: *A portion of an X3D file describing a segmented region: the code highlights the X3D tags used to encode the semantic information*

```
<Shape>
 <MetadataSet name="Semantic">
  <MetadataString name="name" value="horse::body::leftleg"/>
  <MetadataString name="description" value="..."/>
 </MetadataSet>
 <MetadataSet name="Geometry">
  <MetadataInteger name="vertex" value="539"/>
  <MetadataInteger name="edge" value="1593"/>
  <MetadataInteger name="face" value="1054"/>
 </MetadataSet>
 <MetadataSet name="Adjacency">
  <MetadataInteger name="meshNumber" value="1"/>
  <MetadataString name="adjacency" value="5 "/>
 </MetadataSet>
    [...]
 <IndexedFaceSet coordIndex='1 2 3 -1 6 [...] 3 -1'>
 <Coordinate point='40.0289, [...] -47.0883'/>
 </IndexedFaceSet>
</Shape>
```

7. Concluding Remarks

In this paper, we presented a cut-based manual segmentation tool we have implemented in order to recognize meaningful portions of a 3D model easily. We focused also on how we maintain the decomposition into a structural representation, called the segmentation graph, and the updating operations we perform on it. Additionally, the interface we have designed 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.

There are several interesting future directions for the work presented here. For the manual segmentation we are working in order to solve the problem we showed in Section 5. We are also extending the method to allow a local remeshing of the model, if the user specifies exactly where the cut should pass. Additionally, we are planning to implement innovative region-based manual segmentation methods as those presented in [WPP*07, Bro08] 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). In this way, the user will have the possibility to choose among different editors. For the interface, we are working in order to allow multiple selection, so that it will be possible to save separately a portion of the model made up of different segmented regions, if necessary. Finally, the semantic annotator we have presented in Section 6 is the basis for annotation. In our implementation the user can define specific tags, in some well-defined applications, instead, pre-defined thesauri or dictionaries could be used (e.g. medicine, bio-informatics and so on). We are working in order to support RDF/RDFS integration in the system. Thus we will be able to annotate according to specific ontological schema in OWL [W3C10a] or using SKOS [W3C10b]. In this case, the semantics associated to the portions of the model will be saved also in separate RDF files, thus allowing experimenting the complete power of Semantic Web technologies.

Acknowledgments.

The authors would like to thank Matteo Bertucelli for the implementation of the segmentation method and the interface. Some shapes used are taken from the AIM@SHAPE shape repository [aim07] and converted in X3D.

References

[aim07] The European Network of Excellence AIM@SHAPE - contract number 506766. www.aimatshape.net, 2004-2007.

[ARSF07] ATTENE M., ROBBIANO F., SPAGNUOLO M., FALCIDIENO B.: Semantic annotation of 3d surface meshes based on feature characterization. In Falcidieno et al. [FSA*07].

[BJ01] BOYKOV Y. Y., JOLLY M. P.: Interactive graph cuts for optimal boundary &amp; region segmentation of objects in n-d images. vol. 1, pp. 105–112.

[BLHL01] BERNERS-LEE T., HENDLER J., LASSILA O.: The semantic web. *Scientific American 284*, 5 (2001), 34–43.

[Bro08] BROWN S. W.: Interactive part selection for mesh and point models using hierarchical graph-cut partitioning. In *Brigham Young University, PhD Thesis* (2008).

[BS01] BRUYNS C., SENGER S.: Interactive cutting of 3d surface meshes. *Computers & Graphics 25*, 4 (2001), 635–642.

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Trans. Graph. 23*, 3 (2004), 905–914.

[DHP*07] DE FLORIANI L., HUI A., PAPALEO L., HUANG M., HENDLER J. A.: A semantic web environment for digital shapes understanding. In Falcidieno et al. [FSA*07], pp. 226–239.

[DPC08] DE FLORIANI L., PAPALEO L., CARISSIMI N.: A java3d framework for inspecting and segmenting 3d models. In *Web3D* (2008), Fellner D. W., Collins S., (Eds.), ACM, pp. 67–74.

[FKS*04] 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 (2004), 652–663.

[FSA*07] FALCIDIENO B., SPAGNUOLO M., AVRITHIS Y. S., KOMPATSIARIS I., BUITELAAR P. (Eds.):. *Semantic Multimedia, Second International Conference on Semantic and Digital Media Technologies, SAMT 2007, Genoa, Italy, December 5-7, 2007, Proceedings* (2007), vol. 4816 of *Lecture Notes in Computer Science*, Springer.

[Gra06] GRADY L.: Random walks for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 28*, 11 (2006), 1768–1783.

[GSL*99] GREGORY A. D., STATE A., LIN M. C., MANOCHA D., LIVINGSTON M. A.: Interactive surface decomposition for polyhedral morphing. *The Visual Computer 15*, 9 (1999), 453–470.

[Hen08] HENDLER J.: Linked data, web 3.0 and the semantic web. *International Conference on Semantics, Knowledge and Grid 1* (2008), 10–20.

[HG10] HUNTER J., GERBER A.: Harvesting community annotations on 3d models of museum artefacts to enhance knowledge, discovery and re-use. *Journal of Cultural Heritage 11*, 1 (2010), 81 – 90.

[LH07] LASSILA O., HENDLER J.: Embracing "web 3.0". *IEEE Internet Computing 11*, 3 (2007), 90–93.

[LHMR08] LAI Y.-K., HU S.-M., MARTIN R. R., ROSIN P. L.: Fast mesh segmentation using random walks. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2008), ACM, pp. 183–191.

[MSS10] MORONI D., SALVETTI M., SALVETTI O.: Shape analysis, semantic annotation and context modelling for the retrieval of 3d anatomical structures. *Pattern Recognition and Image Analysis 20* (2010), 86–93. 10.1134/S1054661810010098.

[SBSCO06] SHARF A., BLUMENKRANTS M., SHAMIR A., COHEN-OR D.: Snappaste: an interactive technique for easy mesh composition. *The Visual Computer 22*, 9-11 (2006), 835–844.

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Computer Graphics Forum* (2008).

[She01] SHEFFER A.: Model simplification for meshing using face clustering. *Computer-Aided Design 33*, 13 (Nov. 2001), 925–934.

[SRML07] SUN X., ROSIN P. L., MARTIN R. R., LANGBEIN F. C.: Random walks for mesh denoising. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling* (New York, NY, USA, 2007), ACM, pp. 11–22.

[W3C10a] W3C: Owl web ontology language overview. www.w3.org/TR/owl-features/, 2010.

[W3C10b] W3C: Skos simple knowledge organization system. www.w3.org/2004/02/skos/, 2010.

[WPH*04] 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* (2004), pp. 85–94.

[WPP*07] 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).

[x3d08] The Web3D consortium x3d working group. www.web3d.org/x3d, 2008.

[xj308] The xj3d project. www.xj3d.org, 2008.

[ZLZG06] ZHONGPING J., LIGANG L., ZHONGGUI C., GUOJIN W.: Easy mesh cutting. *Computer Graphics Forum 25*, 3 (2006), 283–292.

[ZSH00] ZÖCKLER M., STALLING D., HEGE H.-C.: Fast and intuitive generation of geometric shape transitions. *The Visual Computer 16*, 5 (2000), 241–253.