Semantic Annotation of 3D Surface Meshes based on Feature Characterization

Marco Attene, Francesco Robbiano, Michela Spagnuolo and Bianca Falcidieno

Istituto di Matematica Applicata e Tecnologie Informatiche - Genova, CNR, Italy

Abstract. In this paper we describe the main aspects of a system to perform non-trivial segmentations of 3D surface meshes and to annotate the detected parts through concepts expressed by an ontology. Each part is connected to an instance in a knowledge base to ease the retrieval process in a semantics-based context. Through an intuitive interface, users create such instances by simply selecting proper classes in the ontology; attributes and relations with other instances can be computed automatically based on a customizable analysis of the underlying topology and geometry of the parts.

1 Introduction

Digital 3D shapes have a fundamental role in important and diverse areas such as product modeling, medicine, virtual reality and simulation, and their impact on forthcoming multimedia-enabled systems is foreseen to grow significantly. In the latest years we have assisted to an impressive growth of online repositories of 3D shapes [1–4] which reveals the importance of making these resources more accessible and easy to share and retrieve.

The Stanford repository is one of the earliest and widely used [4] and it maintains simple records of 3D shape models: *core* data – the geometry – plus a brief textual description. More recently, the increase of the number of shapes called for more intelligent searching methods; among them we can cite the first significant results proposed by the Princeton Shape Benchmark [3], which supports searching by geometric similarity starting from queries defined by sketching or by example. A different perspective is adopted by the AIM@SHAPE Shape Repository [1], which is based on a formal organization of 3D models enriched with metadata that make it possible to search for content also in terms of knowledge *about* the represented 3D shapes. Most repositories attempt to ease the retrieval process by associating each shape to a coarse category (e.g. vehicles, aircraft, humans). Note that, since such association is typically a manual operation, a finer subdivision would be a too demanding and subjective task. Nonetheless, the ever-growing size of 3D repositories is making the retrieval hard even in a single bounded category. Thus, characterizing shapes of a given domain is becoming more and more important, and specific annotation approaches that require minimal human intervention must be devised.

To achieve this goal, the structural subdivision of an object into subparts, or *segments*, has proven to be a key issue. At a cognitive level, in fact, the comprehension of an object is often achieved by understanding its subparts [5, 6]. For instance, if we consider an object which has a *human face* it is likely that the object is a *human being* (the presence of a subpart influences the interpretation of the whole object), and if we want to retrieve a human which is *strong* we can search for humans whose arms'*volume* is big (here the quantitative characterization of a part influences the qualitative interpretation of the whole). Furthermore, a proper subdivision of the shape would allow users to access directly subparts: users will be able not only to search, but also to retrieve *legs*, *noses* and *heads* even if the repository was originally intended for whole human body models. Such a possibility is extremely important in modern design applications: creating original shapes from scratch, in fact, is a time-consuming operation and it requires specific expertise, hence re-using parts of 3D shapes is recognized to be a critical issue.

Clearly, the retrieval of 3D objects within a repository can be significantly improved by annotating each shape not only as a whole, but also in terms of its meaningful subparts, their attributes and their relations. The possibility to semantically annotate shape parts may have a relevant impact in several domains. An application that we consider particularly important, for example, is the creation of *avatars* in emerging *MMORPGs* (Massive Multiplayer Online Role-Playing Games) and in online virtual worlds such as Second Life [7]. Currently the avatar design is done from scratch or through the personalization of a predefined amount of parameters (e.g. shape of the body, skin, hair, eyes, clothes). Producing an original avatar through this approach, however, is becoming more and more difficult due to the exponential *demographic* growth in the aforementioned virtual worlds (currently, in Second Life there are more than 7 million residents, each one with his/her avatar). In our view, the possibility to browse and search huge repositories of virtual characters and their parts would end up in a potentially infinite number of avatars, obtained by combination and further personalization of the different parts. Annotated parts might also be exploited by proper interpretation rules to allow their retrieval based on high-level characterizations (e.g. retrieve "heads of Caucasian adult male" or "heads of children", or "heads of black women").

In general, both the extraction and the annotation of the subparts are characterized by an inherent context dependance: the kind of geometric analysis used to detect the segments, as well as the *interpretation* of the segments, can significantly vary in different contexts. A nearly-cylindrical object can be annotated as a *finger* in the domain of *human bodies*, as a *piston* in the domain of *car engines*, and may be not detected at all in another domain in which this kind of features is not interesting. Thus, it is important to devise annotation approaches which are both general (i.e. they must not depend on any particular context) and personalizable (i.e. they must easily adapt to any particular context).

1.1 Overview and contributions

In this paper we tackle the aforementioned part-based annotation problem by presenting a flexible and modular system called the *ShapeAnnotator*. We use surface meshes to represent 3D shapes, and ontologies to describe the annotation domains. By exploiting results from Computer Graphics, Vision and Knowledge Technologies, the ShapeAnnotator makes it possible to load a 3D surface mesh and a domain ontology, to define the meaningful shape parts, to annotate them properly and to save the result in a knowledge base. The ShapeAnnotator makes use of the following solutions:

– A *multi-segmentation* framework to specify complex and heterogeneous surface segments (the $features$);
– A set of functionalities called *segmentmeters* to calculate geometric and topological characterizations of the segments;
– An ontology module which allows the user to browse the domain ontology and to create instances *describing* the features;
– A mechanism to *teach* the system how instance properties can be computed automatically based on segmentmeters.

2 Related Work

To our knowledge, existing literature does not deal with any framework to support feature-based annotation processes for 3D shapes. The two main tasks addressed in our work are the *segmentation* and the *annotation* of 3D shapes. A lot of research that deals with the integration of these two aspects is related to traditional visual media, images and videos in particular. A variety of techniques has been developed in image processing for content analysis and segmentation, and the annotation techniques available are mainly based on the computation of low-level features (e.g. texture, color, edges) that are used to detect so-called regions-of-interest [8–10]. For video sequences, keyframe detection and temporal segmentation are widely used [11].

Although the general framework of content analysis and annotation developed for images or videos may be adapted to 3D shapes, the two domains are substantially different. For images and videos, indeed, the objective is to identify relevant objects in a scene, with the inherent complexity derived by occlusions and intersections that may occur. In the case of 3D, information about the object and its features is complete, and the level of annotation can be therefore more accurate. A peculiarity of 3D shapes is that geometric measures of the single segments, such as bounding box length or best-fitting sphere radius, are not biased by perspective, occlusions or flattening effects. Therefore these measures can be directly mapped to cognitive properties (roundness, compactness, ...).

2.1 Keyword-based and Ontology-based Annotation

Generally speaking, the purpose of annotation is to create correspondences between objects, or segments, and conceptual tags. Once an object and/or its parts are annotated, they can easily match textual searches. Stated differently, advanced and semantics-based annotation mechanisms support content-based retrieval within the framework of standard textual search engines.

The two main types of textual annotation are *keyword*-driven and *ontology*-driven. In the first case users are free to tag the considered resources with any keyword they can think of, while in the second case they are tied to a precise conceptualisation. The trade-off is between flexibility and meaningfulness. In fact, in the case of free keyword annotation users are not forced to follow any formalized scheme, but the provided tags have a meaning just for themselves: since no shared conceptualisation is taken into account, the association of the tag to a precise semantic interpretation can be only accidentally achieved. Well-known examples of this kind of annotation for 2D images are *FLICKR* [12] and *RIYA* [13].

In the *ontology*-driven annotation, the tags are defined by an ontology. An ontology is a formal, explicit specification of a shared conceptualization of a domain of knowledge, and expresses the structuring and modeling of that particular domain [14, 15]. Since the conceptualisation is shared, there is no freedom in the selection of tag names, but this is rewarded with a common understanding of the given tags eligible for selection. Moreover, the shared conceptualisation can also be processed by computer applications, opening up challenging opportunities for further enriching search results with inference mechanisms [16]. In M-*OntoMat-Annotizer* [17] the user is allowed to highlight segments (i.e. regions) of an image and to browse specific domain ontologies in order to annotate parts of the former with specific instances of the latter. Similarly, *Photostuff* [18] provides users the ability to annotate regions of images with respect to an ontology and publish the automatically generated metadata to the Web.

The work presented in this paper falls in the class of the ontology-driven annotation approaches. Specifically, we tackle the problem of annotating shapes belonging to a specific category which is described by an ontology (i.e. human bodies, cars, pieces of furniture, ...). Each of these ontologies should conceptualize shape features characterizing the category, their attributes and their relations. In a *human body*, for example, *head*, *arms* and *legs* are relevant concepts, relations such as *arm* `is_a` *limb* hold, and attributes such as the *size* of the *head* are helpful to infer higher-level semantics (e.g. ethnic group, gender, age-range).

2.2 Segmentation of polygonal meshes

Having an ontology that describes a given class of shapes, the optimal solution for annotating 3D models would be to use a shape segmentation algorithm able to automatically detect all the features conceptualized by the ontology. This approach is far from being feasible, as existing segmentation algorithms hardly target semantic features and usually follow a pure geometric approach. Recent surveys of these methods can be found [19] and [20].

Much of the works tackling the segmentation problem with the objective of *understanding* a shape [21] are inspired by studies on human perception. For example there are theories that received a large consensus [22, 5] and that indicate how shapes are recognized and mentally coded in terms of relevant parts and their spatial configuration, or structure.

In another large class of methods, the focus is mainly on the detection of geometrically well-defined features. These segmentations do not produce natural features but patches useful for some tasks that require completely different and possibly non-intuitive schemes (e.g., approximation, remeshing, parameterization) [23–25]. Generally speaking, this approach is feasible when the features have some formal structure that can be associated to a mathematical formulation. In natural domains, for example human body models, there is no clue on how to define relevant features, and only few methods in the literature tackled a semantics-oriented segmentation in these kind of domains [26].

3 Multi-segmentation and Part-based Annotation

In the case of 3D shapes, the identification of relevant features is substantially different from the corresponding 2D case. For 2D images, segmentation algorithms are not always considered critical to define features for annotation; on a flat image, in fact, useful features may be even sketched by hand [17]. In contrast, a 3D shape may be very complex and drawing the boundary of a feature might become a rather time-consuming task, involving not only the drawing stage, but also rotating the scene, translating and zooming in and out to show the portions of the surface to draw on. Moreover, while on a 2D image a closed curve defines an inner area, in 3D this is not always true. Hence, for the 3D case using segmentation algorithms to support feature detection is considered a mandatory step.

Nevertheless, the huge amount of different and specialized works on mesh segmentation indicates that satisfactory results are missing. The majority of the methods used in computer graphics are not devised for detecting specific features within a specific context, as for example is the case of form-feature recognition in product modeling and manufacturing. The shape classes handled in the generic segmentation contexts are broadly varying: from virtual humans to scanned artefacts, from highly complex free-form shapes to very smooth and feature-less objects. Moreover, it is not easy to formally define the meaningful features of complex shapes in a non-engineering context and therefore the capability of segmentation methods to detect those features can only be assessed in a qualitative manner [20].

Hence, our proposition is that, due to intrinsic limitations, no single algorithm can be used to provide rich segmentations, even within a single domain. This motivates the introduction of a theoretical framework for working with multi-segmentations, that allow for a much more flexible support for semantic segmentation. The intuition behind multi-segmentation is that a meaningful shape segmentation is obtained by using in parallel a set of segmentation algorithms and by selecting and refining the detected segments.

Most segmentation algorithms proposed in the literature [20] strive to subdivide the surface into non-overlapping patches forming an exhaustive partitioning of the whole model (Figure 1(b), (c) and (d)). Our proposition is that even this assumption is too restrictive: following the claim that the segmentation has to reflect the cognitive attitude of the user, the detected parts do not necessarily have to constitute a partition of the model, as some segments may overlap, and some surface parts may not belong to any significative segment at all. Therefore, it is often possible to design a proper technique for the identification of a particular class of features [27, 23] and, if there is the need to identify features of different classes, it is possible to use different segmentation algorithms and take the features from all of their results. In some cases, moreover, there is an intrinsic *fuzziness* in the definition of the boundaries of a feature (i.e., in a human body model the neck may be considered part of both the head and the torso). This is another reason to avoid the restriction of using a sharp partitioning of the whole to identify all the relevant segments.

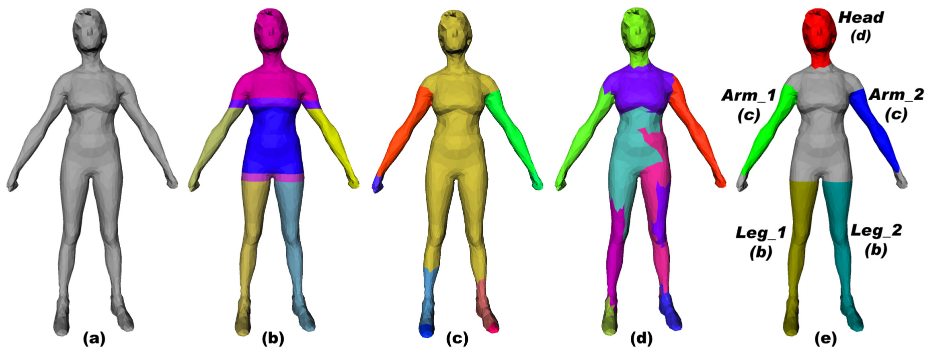


Fig. 1. An original mesh (a) has been partitioned using different segmentation algorithms: [28] in (b), [27] in (c) and [23] in (d). Only the most relevant features taken from (b), (c) and (d) have been selected and annotated in (e).

Due to these observations, we introduce the concept of *multi-segmentation* of a 3D surface represented by a triangle mesh, and say that in a multi-segmented mesh, the results of several segmentation approaches may overlap (e.g. {(b),(c),(d)} in Figure 1). When a multi-segmented mesh is interpreted within a specific context, some of the segments can be considered particularly *meaningful*. Such meaningful segments (i.e. the features) can be annotated by specific conceptual tags describing their meaning within the context. In this paper, we refer to an *annotated mesh* as to a multi-segmented mesh in which some of the segments have been annotated (e.g. Figure 1 (e)).

4 The Shape Annotator

Having established *what* is an annotated mesh, it remains to explain *how* to produce it out of an existing triangle mesh. In principle, an expert in a particular domain should be able to identify significant features and to assign them a specific meaning. As an example, an engineer should be able to look at a surface mesh representing an engine and identify which parts have a specific mechanical functionality. Unfortunately, to the best of our knowledge, today there is no practical way to transform such expertise into usable content to be coupled with the plain geometric information.

To bridge this gap, we defined an annotation pipeline and developed a prototype graphical tool called the *ShapeAnnotator*. This tool has been specifically designed to assist an expert user in the task of annotating a surface mesh with semantics belonging to a domain of expertise.

After loading a model and a domain ontology, the first step of the annotation pipeline is the feature indentification, i.e. the execution of segmentation algorithms to build the multi-segmented mesh. Once done, from the resulting multi-segmented mesh interesting features can be interactively selected. Each interesting feature can then be annotated by creating an instance of a concept described in the ontology. Optionally, the system may be also programmed to automatically compute attributes and relations among the instances to significantly enrich the resulting knowledge base.

4.1 Feature identification

In order to identify surface features, the ShapeAnnotator provides a set of segmentation algorithms. Our prototype has a plugin-based architecture so that it is possible to import proper segmentation algorithms according to the requirements of the specific class of features. In the current implementation, we have chosen a number of algorithms that cover quite a wide range of feature types. In particular, it is possible to capture:

– *Planar features* through a clustering based on a variational shape approximation via best-fitting planes [24];
– *Generalized tubular features* with arbitrary section computed by the *Plumber* algorithm introduced in [27];
– *Primitive shapes* such as planes, spheres and cylinders through a hierarchical segmentation based on fitting primitives [23];
– *Protrusions* extracted through shape decomposition based on Morse analysis using the height function, the distance from the barycenter and the integral geodesics [28].

Using these tools, it is possible to roughly capture features and also to refine them through morphological operators. Up to now the following operators are available:

– Growing a segment by adding a strip of triangles to its boundary;
– Shrinking a segment by removing a strip of triangles from its boundary;
– Merging two segments.

It is also possible to remove a segment or to add a new segment from scratch (i.e., a single triangle), and edit it through the above operators.

These functionalities make it possible to refine raw segmentations and properly define useful non-trivial features within few mouse clicks, as shown in Figure 2. Further examples are shown in Figure 6.

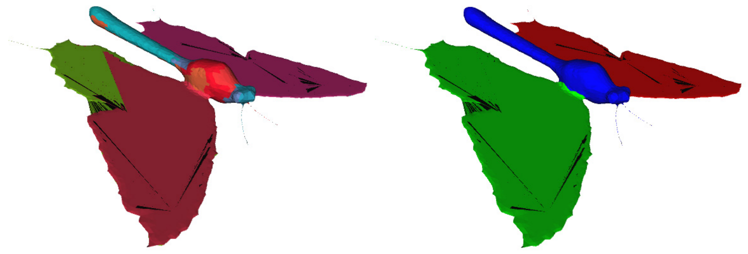


Fig. 2. Definition of non-trivial features starting from a raw segmentation. On the left, the HFP algorithm [23] could not capture the features due to degenerate mesh elements. On the right the unprecise features computed have been edited to obtain a better segmentation.

4.2 Manual Annotation

To annotate the features, the user may select proper conceptual tags within a domain of expertise formalized as an OWL [29] ontology. This choice offers a number of advantages, including the non-negligible fact that OWL is supported by popular ontology editors [30]. Strictly speaking, for the current functionalities of the ShapeAnnotator, a simpler language could be sufficient, as long as the user is prompted with the chance of selecting among relevant concepts; the choice of OWL, however, has been driven by the potential evolution of the ShapeAnnotator, which is foreseen to become more *intelligent* in the sense of providing inference functionalities (see Section 5).

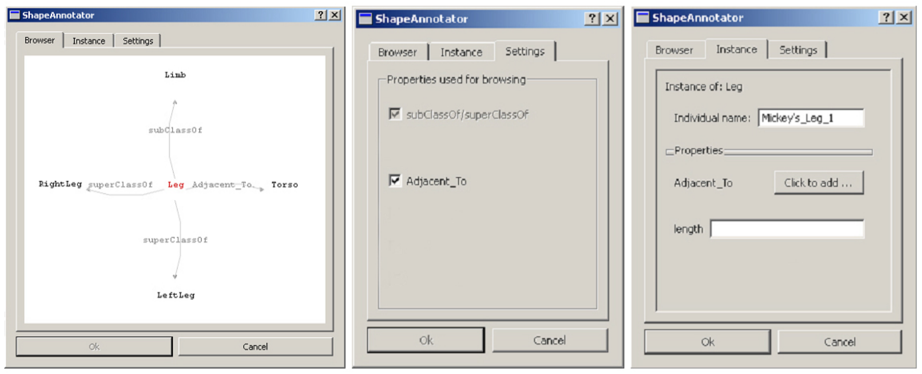
Non trivial ontologies may be huge [31], and effective browsing facilities are fundamental to reduce the time spent to seek the proper concept to instantiate. In our approach, the ontology is depicted as a graph in which nodes are classes and arcs are relations between them (see Figure 3, left).

Browsing the ontology consists of moving along paths in the graph, which means jumping from a concept to another across relations. The navigation may be customized by the user and, while the simplest way of browsing is across relations of type `subClassOf` or `superClassOf`, it is possible to select any combination of properties that will be shown by the browser (see Figure 3, middle). Once a proper concept has been identified, the ShapeAnnotator provides the possibility to create an instance, which means providing a URI (i.e., a unique name) and setting the value of the properties (attributes and relations) defined in the ontology for the class being instantiated (see Figure 3, right).

Each instance may be modified in a second step in order to make it possible to define non-trivial relations between instances of the knowledge base (i.e., `myHead isAdjacentTo myNeck`).

4.3 Automatic Annotation

Currently, our system requires the user to manually select the concepts to instantiate; for attributes and relations between instances, however, there is the



☑ subClassOf/superClassOf
☑ Adjacent_To

Instance of: Leg
Individual name: Mickey's_Leg_1
Adjacent_To
length

Fig. 3. The ontology browser, the selection of navigation settings and the creation of an instance.

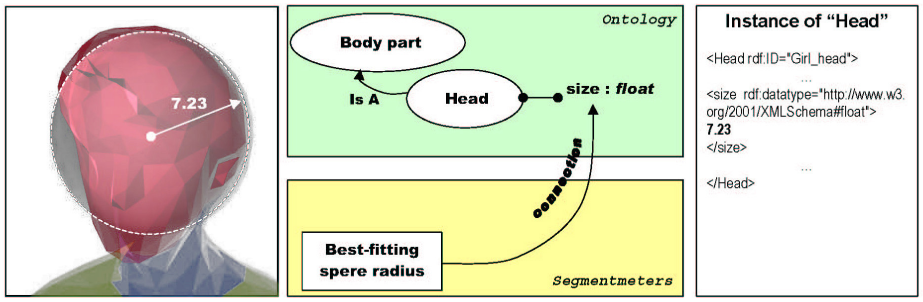
possibility to *tell* the ShapeAnnotator how these properties can be calculated without the user intervention. The ShapeAnnotator, in fact, comes with a set of functionalities to measure geometric aspects of shape parts (i.e. bounding box length, radius of best-fitting cylinder, ...) and to establish topological relations among the parts (i.e. adjacency, containment, overlap, ...). We call these functionalities *segmentmeters*.

Though segmentmeters work independently of any ontology, the user may define their *interpretation* within each specific domain of annotation. Namely, the user may establish a set of *connections* between topological relations and conceptual relations (e.g. ”segment adjacency” $\leftrightarrow$ `is_connected_to`) and between calculated values and class attributes (e.g. ”radius of best-fitting cylinder” $\leftrightarrow$ `through_hole` :: `radius`).

After having established such connections, the instance properties are transparently computed by the system. For example, when annotating a reverse engineered mechanical model, a part may be manually annotated as a `Through_hole`, while its parameter `radius` is automatically computed by the ShapeAnnotator as the radius of the cylinder that best fits the geometry of the part; if two adjacent segments are manually annotated as instances of the class `stiffener`, the relation `is_adjacent_to` is automatically set to conceptually link the two instances. An example of connection is shown if Figure 4.

Furthermore, there is the possibility to combine some segmentmeters within formulae to be connected to specific attributes. When annotating a human body model, for example, the user may want to tell the ShapeAnnotator something like ”for each instance of the class `head` set the attribute `size` with the length of the great circle of the segment’s best-fitting sphere”. This will be accomplished by connecting the formula $2 * \pi * best\_fitting\_sphere\_radius$ to the attribute `head` :: `size`.

Since we believe that modularity is crucial to provide a flexible annotation framework, we made our system able to load additional segmentmeters imple-



```
<Head rdf:ID="Girl_head">
             ...
<size  rdf:datatype="http://www.w3.
org/2001/XMLSchema#float">
7.23
</size>
             ...
</Head>
```

Fig. 4. The attribute *size* of the instance *Girl_head* is automatically set to the value 7.23 because this is the value computed by the connected segmentmeter.

mented externally as plug-ins, just as we have done for the segmentation algorithms.

In our prototype the connections can be established through a proper dialog in which all the segmentmeters are shown as buttons. Currently, they belong to the following two groups:

– **Topological relations between segments** consisting of *adjacency*, *overlap*, *disjointness* and *containment*;
– **Geometric aspects of a segment** consisting of *oriented bounding box length*, *width* and *height*, *best-fitting sphere radius*, *best-fitting cylinder radius*.

By clicking on a segmentmeter button, a list of properties defined in the domain ontology is shown and the user may select some of them to establish connections. The list of properties shown is filtered so that only admissible connections can be selected; this avoids, for example, the connection of a property with more than one segmentmeter, or between non-compatible segmentmeters and ontology properties (e.g. "segment adjacency" ↔ `through_hole` :: `radius`).

The connections can be established either before the creation of instances or afterwards. In the former case, for each newly created instance the properties are computed on the fly based on the existing connections; in the latter case, the values of the properties of the existing instances are (re)computed according to the newly established connections.

To allow their easy reuse when annotating several models in the same domain, the connections can also be saved as an XML-based file and loaded later on.

4.4 Resulting Knowledge Base

The result of the annotation process is a set of instances that, together with the domain ontology, form a knowledge base. Each instance is defined by its URI, its type (i.e., the class it belongs to) and some attribute values and relations that might have been specified/computed. In its current version, the ShapeAnnotator saves the multi-segmented mesh along with the selected, and possibly edited features as a single PLY file. The instances are saved as a separate OWL file that imports the domain ontology. Additionally, the OWL file contains the definition of two extra properties:

- `ShannGeoContextURI`, whose value is the URI of the multi-segmented mesh (typically the path to the PLY file saved by the ShapeAnnotator);
- `ShannSegmentID`, whose value is an index that specifies a segment in the multi-segmented mesh.

All the instances produced during the annotation pipeline are automatically assigned values for the above two properties, so that the link between semantics and geometry is maintained within the resulting knowledge base (see Figure 5).

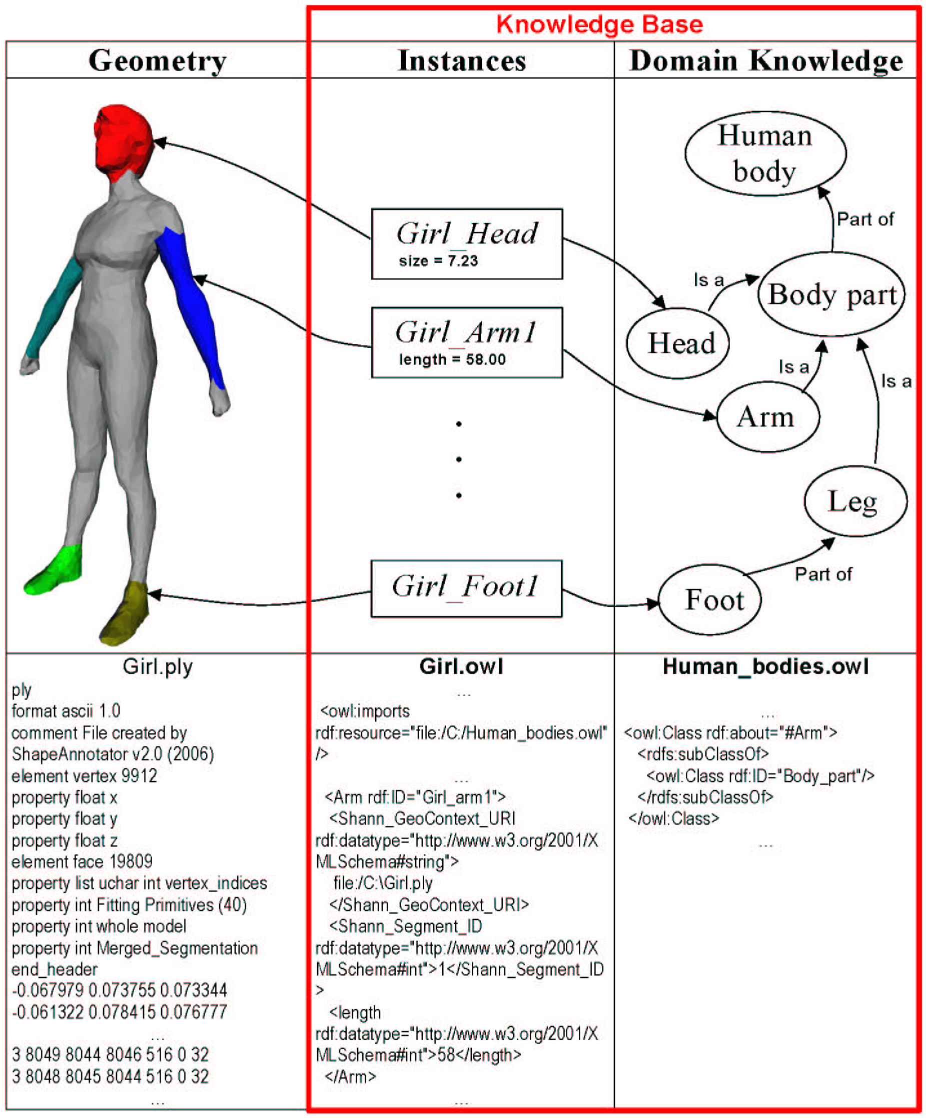
Note that the OWL files produced by the annotation of several models in the same domain can constitute a unified knowledge base; this would contain all the instances describing the models in terms of their meaningful parts, allowing unprecedented levels of granularity for query formulation. Once an instance has been located, for example, it is possible to retrieve the geometry of the corresponding part and, possibly, to extract it without the need to download and process the whole model.

5 Conclusions and Future Research

This paper tackles the problem of providing useful semantic annotations to 3D shapes. We have discussed the key aspects of the subject, and shown that simple keywords attached to a whole shape do not provide enough information to answer complex queries. Thus, we have illustrated how to decompose the shape into interesting features within the multi-segmentation framework, and introduced the annotation pipeline to attach a formal semantics to the features and the whole shape. We have pointed out that the process is unfeasible using only state-of-the-art approaches. Conversely, we have described our novel ShapeAnnotator tool that makes it possible to annotate 3D shapes through few mouse clicks using the pipeline proposed. The introduction of segmentmeters along with their context-based interpretation represents a first step towards automatic annotation methods for the 3D domain.

In future developments, we plan to treat also lower-dimensional features (i.e. curves and points) with a twofold benefit: they will be eligible for annotation, just as the other segments, and they will be useful to edit other segments (e.g. a curve may be used to split a segment into two subsegments).

In its current version, the ShapeAnnotator has minimal inference capabilities which have been implemented just to provide a flexible browsing of the ontology. This means that input ontologies are assumed to be mostly asserted; if not, the user can use an offline reasoner to produce the inferred parts. Future developments are targeted to this aspect, and internal inference capabilities are foreseen. Besides simple deductions on the input ontology, inference will also be used to (partially) automate the whole annotation pipeline. Although the process can be completely automated in rather few domains, in many others the user might be required to contribute only to disambiguate few situations.



```
ply
format ascii 1.0
comment File created by
ShapeAnnotator v2.0 (2006)
element vertex 9912
property float x
property float y
property float z
element face 19809
property list uchar int vertex_indices
property int Fitting Primitives (40)
property int whole model
property int Merged_Segmentation
end_header
-0.067979 0.073755 0.073344
-0.061322 0.078415 0.076777
...
3 8049 8044 8046 516 0 32
3 8048 8045 8044 516 0 32
...
```

```
...
<owl:imports
rdf:resource="file:/C:/Human_bodies.owl"
/>
...
 <Arm rdf:ID="Girl_arm1">
  <Shann_GeoContext_URI
rdf:datatype="http://www.w3.org/2001/X
MLSchema#string">
    file:/C:\Girl.ply
  </Shann_GeoContext_URI>
  <Shann_Segment_ID
rdf:datatype="http://www.w3.org/2001/X
MLSchema#int">1</Shann_Segment_ID
>
  <length
rdf:datatype="http://www.w3.org/2001/X
MLSchema#int">58</length>
 </Arm>
...
```

```
...
<owl:Class rdf:about="#Arm">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Body_part"/>
  </rdfs:subClassOf>
</owl:Class>
...
```

Fig. 5. The instances along with their specific relations represent a formal bridge between geometry and semantics.

6 Acknowledgements

This work is supported by the AIM@SHAPE Network of Excellence (EC FP6 IST Contract N. 506766). Thanks are due to all the members of the Shape Modeling Group of the CNR-IMATI-GE, and to the people that, through discussions and suggestions, made it possible to develop the Shape Annotator. *This work* *was conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine.*

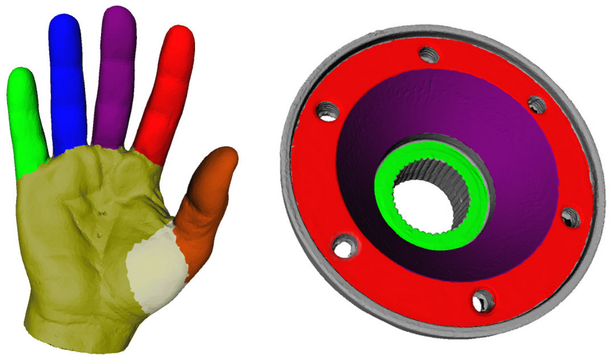


Fig. 6. Examples of segmentation of a natural and an artificial shape. The bright color on the hand surface indicates that the corresponding part is an overlap of segments (in this case the thumb and the palm).

References

1. [online]: AIM@SHAPE shape repository. http://shapes.aimatshape.net/ (2004)
2. [online]: 3D CAD browser. http://www.3dcadbrowser.com/ (2001)
3. Shilane, P., Min, P., Kazhdan, M., Funkhouser, T.: The princeton shape benchmark. In: SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04), Washington, DC, USA, IEEE Computer Society (2004) 167–178
4. [online]: The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3dscanrep
5. Marr, D.: Vision - A computational investigation into the human representation and processing of visual information. W. H. Freeman, San Francisco (1982)
6. [online]: Network of excellence AIM@SHAPE, EU FP6 IST NoE 506766. http://www.aimatshape.net (2004)
7. [online]: Secondlife. http://www.secondlife.com
8. Hermes, T., Miene, A., Kreyenhop, P.: On textures: A sketch of a texture-based image segmentation approach. In Decker, R., Gaul, W., eds.: Classification and Information Processing at the Turn of the Millenium. (10 - 12 March 2000) 210–218 Proc. 23rd Annual Conference Gesellschaft für Klassifikation e.V. 1999.
9. Xiaohan, Y., Ylä-Jääski, J., Baozong, Y.: A new algorithm for texture segmentation based on edge detection. Pattern Recognition **24**(11) (1991) 1105–1112
10. Pavlidis, T., Liow, Y.T.: Integrating region growing and edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **12**(3) (1990) 225–233
11. Browne, P., Smeaton., A.F.: Video retrieval using dialogue, keyframe similarity and video objects. In: ICIP 2005 - International Conference on Image Processing. (2005)
12. [online]: Flickr. http://www.flickr.com/

13. [online]: Riya. http://www.riya.com/
14. Gruber, T.R.: Towards principles for the design of ontologies used for knowledge sharing. In Guarino, N., Poli, R., eds.: Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands, Kluwer Academic Publishers (1993)
15. Guarino, N.: Formal ontology and information systems. In Guarino, N., ed.: Formal Ontology in Information Systems. IOS Press, Amsterdam (1998) 3–18
16. C.Saathoff: Constraint reasoning for region-based image labelling. In: 3rd IEE International Conference of Visual Information Engineering, VIE 2006, Special Session on Semantic Multimedia. (2006)
17. Petridis, K., Anastasopoulos, D., Saathoff, C., Timmermann, N., Kompatsiaris, I., Staab, S.: M-OntoMat-Annotizer: Image annotation. linking ontologies and multimedia low-level features. In: KES 2006 - 10th Intnl. Conf. on Knowledge Based, Intelligent Information and Engineering Systems. (oct 2006)
18. [online]: Photostuff. http://www.mindswap.org/2003/photostuff/
19. Shamir, A.: Segmentation and shape extraction of 3D boundary meshes. In: Eurographics 2006 - State of the Art Reports. (2006) 137–149
20. Attene, M., Katz, S., Mortara, M., Patanè, G., Spagnuolo, M., Tal, A.: Mesh segmentation - a comparative study. In: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06), Washington, DC, USA, IEEE Computer Society (2006) 7
21. Attene, M., Biasotti, S., Mortara, M., Patane, G., Falcidieno, B.: Computational methods for understanding 3D shapes. Computers&Graphics **30**(3) (2006) 323–333
22. Biederman, I.: Recognition-by-Components: A theory of human image understanding. Phicological Review **94** (1987) 115–147
23. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. The Visual Computer **22**(3) (2006) 181–193
24. Cohen-Steiner, D., Alliez, P., Desbrun, M.: Variational shape approximation. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, New York, NY, USA, ACM Press (2004) 905–914
25. Zhang, E., Mischaikow, K., Turk, G.: Feature-based surface parameterization and texture mapping. ACM Transactions on Graphics **24**(1) (2005) 1–27
26. Mortara, M., Patané, G., Spagnuolo, M.: From geometric to semantic human body models. Computers & Graphics **30**(2) (2006) 185–196
27. Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Plumber: A method for a multi-scale decomposition of 3D shapes into tubular primitives and bodies. In: SM '04: Proceedings of the ACM Symposium on Solid Modeling, New York, NY, USA, ACM Press (2004) 339–344
28. Biasotti, S.: Computational Topology methods for Shape Modelling Applications. PhD thesis, University of Genoa, Italy (2004)
29. [online]: OWL web ontology language guide. http://www.w3.org/tr/2004/rec-owl-guide-20040210/ (Feb 2004) W3C Recommendation.
30. [online]: The Protégé ontology editor. http://protege.stanford.edu (2006)
31. [online]: The NCI cancer ontology. http://www.mindswap.org/2003/cancerontology (2003)