Characterization of 3D Shape Parts for Semantic Annotation

Marco Attene [*] Francesco Robbiano Michela Spagnuolo
Bianca Falcidieno

IMATI-GE / CNR, Via De Marini 6, 16149, Genova, Italy

Abstract

3D content stored in big databases or shared on the Internet is a precious resource for several applications, but unfortunately it risks to be underexploited due to the difficulty of retrieving it efficiently. In this paper we describe a system called the "ShapeAnnotator" through which it is possible to perform non-trivial segmentations of 3D surface meshes and annotate the detected parts through concepts expressed by an ontology. Each part is connected to an instance that can be stored in a knowledge base to ease the retrieval process based on semantics. 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. We show how our part-based annotation framework can be used in two scenarios, namely for the creation of avatars in emerging Internet-based virtual worlds, and for product design in e-manufacturing.

Key words: Surface mesh, Ontology, Segmentation

1 Introduction

Digital models of 3D objects are nowadays recognized as the upcoming wave of digital content shared, distributed and even created over the Internet. Besides the availability of broad-band communication networks, indeed, recent technological advances made available cost-effective scanning devices that could

[*] Tel.: +39 010 6475691; Fax.: +39 010 6475660
Email addresses: `attene@ge.imati.cnr.it` (Marco Attene), `robbiano@ge.imati.cnr.it` (Francesco Robbiano), `spagnuolo@ge.imati.cnr.it` (Michela Spagnuolo), `falcidieno@ge.imati.cnr.it` (Bianca Falcidieno).

not be imagined a decade ago: it is now possible to acquire 3D data of physical objects, and even whole cities, and produce digital models of their geometry that can be easily shared and transmitted. Not by chance, in the last few years we have assisted to an impressive growth of online repositories of 3D shapes [31,29,39,28] which reveals the importance of making these resources more accessible and easy to share and retrieve. At the same time, 3D social networking and 3D mapping applications (e.g., Second Life, GoogleEarth) are getting such a success to induce leading analysts to predict that a dramatic shift is taking place in the way people see and navigate the Internet [23].

Besides the impact on entertainment, the ease of producing and collecting 3D data in digital form has caused a gradual shift of paradigm in various applied and scientific domains: from physical prototypes and experience to virtual prototypes and simulation. This shift has an enormous impact on a number of industrial and scientific sectors, such as Design and Manufacturing, Serious Gaming and Simulation, Cultural Heritage and Archaeology, Medical Applications, Bioinformatics and Pharmaceutical Sciences.

Key to an effective sharing of 3D media is the possibility to annotate 3D models with information about their semantics, or any other knowledge item, useful to characterize the objects and the context in which they are used. In the last years, research in multimedia and knowledge technologies demonstrated the potential of semantic annotations to support advanced sharing and interactions with multimedia, for example for search, reuse and repurposing of multimedia resources. The power of annotation, indeed, relies in the possibility to create correspondences between objects, or parts of them, and conceptual tags: once the media and/or their 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.

Even if with a slightly different flavour, the importance of annotating 3D data has been largely addressed in industrial manufacturing, where interoperability becomes a crucial issue and different solutions have been proposed to face it efficiently. For example, the introduction of the so-called Group Technology [35] in industrial product modelling allowed to model and encode classes of objects that exhibit design and manufacturing similarities, with a favorable impact on design and production practices. This approach to design, however, did not provide means to explicitly support the annotation and indexing of ”pieces” of the objects at hand. Also, in industrial manufacturing, knowledge technologies have been proposed as an efficient methodology to formalize and share semantics in different contexts of the product development process. Product Lifecycle Management, or PLM, systems are the most common tool used in this domain, and such formalizations are mainly document-oriented and provide a low-level description of the product data with no special customized view. Also, there are examples of ontologies for the formalization of CAD/PDM/PLM knowledge [17,18].

Besides industrial manufacturing, other domains have addressed the problem of defining appropriate knowledge management approaches to code and document 3D data: in spatial data handling, medical applications or bioinformatics we can find many *de facto* standards for sharing data, and numerous repositories of 3D shapes are now online and call for efficient retrieval tools.

Existing 3D repositories, however, are based on metadata that describe the object as a whole, possibly indicating the membership to some semantic class (e.g. vehicles, aircraft, humans). Note that, since this association is typically done manually, a finer classification would be very demanding. 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. Moreover, to the extent of our knowledge, there is currently no repository offering the possibility to browse or search model collections according to annotations which refer to specific parts of the objects. At the same time, the problem of automatic classification of 3D objects is still an open research challenge, and the solutions offered by the computer graphics community for shape analysis are decoupled from any actual semantic context, providing segmentations that are in most of the cases meaningful only in a geometric sense.

To our knowledge, the first attempt of adopting knowledge technologies in 3D shape modeling was pursued by the AIM@SHAPE project [32]. In AIM@SHAPE, 3D content is organized by making a distinction between knowledge pertaining to a purely geometric level (e.g. type of representation, number of vertices, genus), to a structural description of the objects' shape (e.g. skeletons, segmentations), and finally to a semantic annotation of objects and objects' parts. The structural subdivision of an object into subparts, or *segments*, has proven to be a key issue to devise effective shape annotations as it provides a way to identify relevant parts of an object geometry in an automatic manner. Also at a cognitive level, in fact, the comprehension of an object is often achieved by understanding its subparts [20,32]. 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).

1.1 Overview and contributions

In this paper we discuss a flexible and modular system for part-based annotation of 3D objects, called the *ShapeAnnotator*. The paper integrates and extends the work presented in [4,5]. In our settings, 3D shapes are represented by surface meshes while annotation domains are formalized by ontologies: these are mainly implementation choices, while the whole framework has a larger applicability and is independent of the specific representation used both for the geometry and knowledge.

The novelty of the ShapeAnnotator relies on the concurrent use of a variety of shape segmentation tools to offer a rich set of operators by which the user can interact with the objects' shape and easily select the part he/she wishes to link to relevant concepts expressed by the ontology. The ShapeAnnotator acts therefore at two levels: it helps the user in the identification of relevant parts, or features, in the model, and it provides the software environment to annotate the parts with concepts that express their semantics. Moreover, since the formalization of a concept may also involve the specification of metric parameters of the part (e.g. dimensions, length), the annotation step implements also a number of automatic services for the computation of these quantitative properties.

To summarize, in the paper we will describe how the ShapeAnnotator makes use of the following solutions:

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

The use of the ShapeAnnotator will be discussed in two scenarios of application: virtual character design and reverse engineering. In the first scenario, we will make use of an ontology that specifies the structuring of a human body into its main component, while in the second scenario we will show how a feature ontology can be used to attach information about relevant manufacturing parts to an object represented as a plain surface mesh.

2 Related Work

The two main tasks addressed by our work are related to 3D feature extraction and indexing, which are supported by *segmentation* and *annotation* respectively.

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 available annotation techniques are mainly based on the computation of low-level features (e.g. texture, color, edges) that are used to detect the so-called regions-of-interest [16,40,36]. For video sequences, keyframe detection and temporal segmentation are widely used [9].

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 annotation can be therefore more accurate. For this reason, geometric measures, such as bounding box length or best-fitting sphere radius of relevant parts, are not biased by perspective, occlusions or flattening effects.

In the following, we will briefly review relevant work in the area of annotation and segmentation, pointing out the issues that are more relevant to the features of the ShapeAnnotator. We also mention the work presented in [8] as an interesting example of how semantic annotations could be embedded in an MPEG-7 framework. The ShapeAnnotator nicely complements the work in [8] by providing the tools to actually interact with the geometric representations of object and realize the annotation in practice.

2.1 Keyword-based and Ontology-based Annotation

The purpose of an annotation process is to create correspondences between objects, or segments, and conceptual tags. 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* [24] and *RIYA* [26].

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 [13,14]. 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 the further enrichment of search results with inference mechanisms [12]. In M-*OntoMat-Annotizer* [37] 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* [25] 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 3D shapes

Given an ontology that describes a specific 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 [38], while a comparison of the segmentation results is addressed in [3].

Much of the works tackling the segmentation problem with the objective of *understanding* a shape [1] are inspired by studies on human perception, which loosely couple semantics to geometry. For example there are theories that received a large consensus [7,20] 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 tasks that require different and possibly non-intuitive schemes (e.g., approximation, remeshing, parameterization) [2,11,41]. 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 [21].

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 [37]. 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 given 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 [3].

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

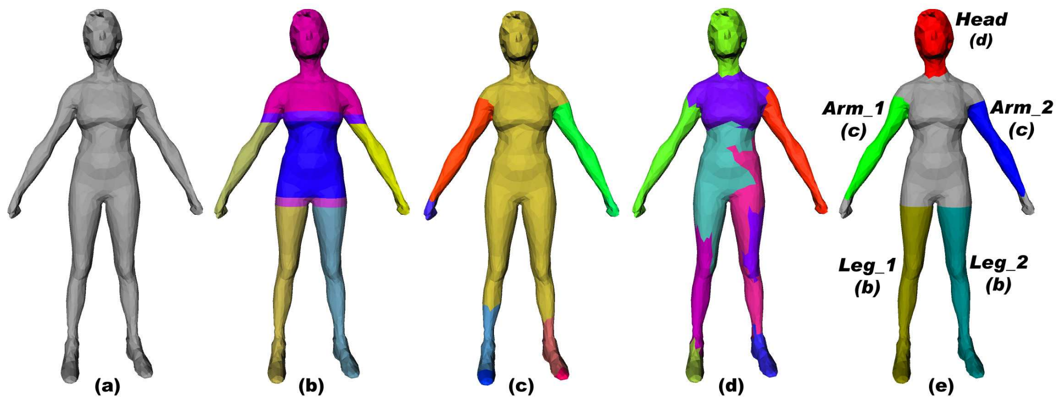


Fig. 1. An original mesh (a) has been partitioned using different segmentation algorithms: [6] in (b), [22] in (c) and [2] in (d). Only the most relevant features taken from (b), (c) and (d) have been selected and annotated in (e).

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 [3] 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 [22,2] 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.

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 identification, 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

Hereafter, we refer to a *segment* as to a connected set of triangles produced by a segmentation algorithm, and to a $feature$ as to a part of the shape that is worth an annotation; specifically, a feature can be either a connected component of the mesh, a part of a connected component, a set of connected components or even the whole input 3D model. Note that if the input is a scene made of several components, it is reasonable to expect that most connected components correspond to actual features to be annotated. Also, some modelers offer the possibility to create "concise" or "iconic" meshes by using non-manifold configurations of the faces; it is possible, for example, to replace the wings of an aircraft (which are solid, though relatively thin) with sheets of triangles with zero-thickness. In this kind of models, the decomposition in manifold components often corresponds to the identification of regions which are actually meaningful; in the example of the aircraft, such a decomposition would lead to one fuselage and two wings.

During the loading stage, the ShapeAnnotator creates a "default" segmentation by grouping together triangles forming connected components. If the triangles of one of such components do not constitute a combinatorial manifold, the loader automatically runs a conversion algorithm [15] that properly duplicates singular elements so that the component is replaced by a set of connected combinatorial manifolds with the same geometry. Eventually, each of these connected manifolds becomes a single segment of the default segmentation.

In order to identify surface features which are subparts of manifold components, the ShapeAnnotator provides a set of mesh 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 [11];
- *Generalized tubular features* with arbitrary section computed by the *Plumber* algorithm introduced in [22];
- *Primitive shapes* such as planes, spheres and cylinders through a hierarchical segmentation based on fitting primitives [2];
- *Protrusions* extracted through shape decomposition based on Morse analysis using the height function, the distance from the barycenter and the integral geodesics [6].

Also, it is possible to specify curves as sets of connected mesh edges; such curves can be used as auxiliary tools to define surface features. In their turn, the curves can be either drawn manually by the user, or automatically detected by the system (e.g. as sharp creases of the mesh).

The first step is to use some of these tools and roughly capture some features of the object. Then it is possible to refine them through morphological operators. Up to now we set up some operators which act on a feature, determining the growth (or the shrinkage) of it by adding (or removing) a strip of triangles to (from) its boundary. But, since the above operators act blindly only on the geometry of the feature, we devised a set of more meaningful and useful operators, that take into account the presence of other influencing nearby features. These operators make it possible for instance to:

- Merge two surface features;
- Grow or shrink a feature until its boundary is shared with another feature, or coincides with a curve;
- Split a feature in two using a curve as the new common boundary.

It is also possible to remove a feature or to add a new one from scratch (i.e.,

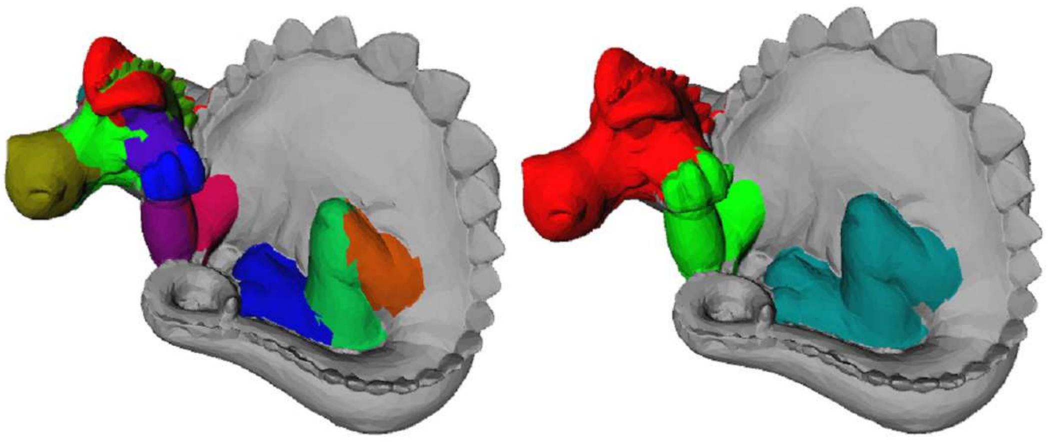


Fig. 2. Definition of non-trivial features starting from a raw segmentation. On the left, the HFP algorithm [2] could not capture the features properly. On the right the unprecise features computed have been edited to obtain a more useful segmentation.

a single triangle), and edit it through the above operators.

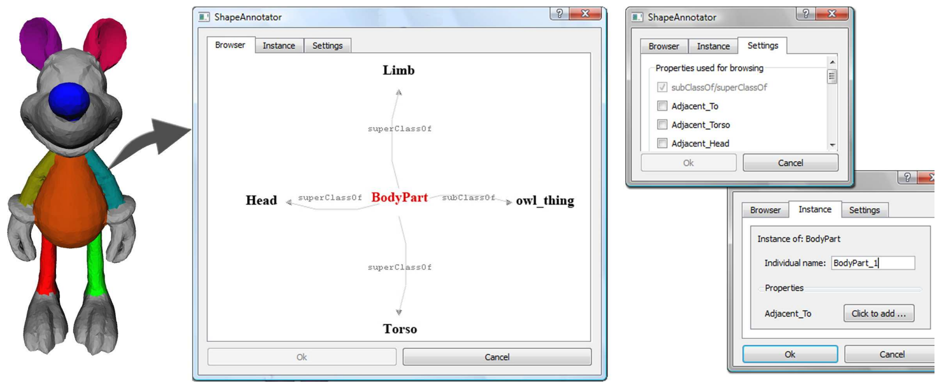
These operations make it possible to refine raw segmentations and properly define useful non-trivial features within a few mouse clicks, as shown in Figure 2. Further examples are shown in Figures 6 and 7.

4.2 Manual Annotation

To annotate the features, the user may select proper conceptual tags within a domain of expertise formalized as an OWL [33] ontology. 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 6), and by the fact that OWL is supported by popular ontology editors [34].

Non trivial ontologies may be huge [30], 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 ShapeAnno-



[✓] subClassOf/superClassOf
[ ] Adjacent_To
[ ] Adjacent_Torso
[ ] Adjacent_Head

Instance of: BodyPart
Individual name: BodyPart_1
Adjacent_To

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

tator provides the possibility to create an instance, which means providing a URI (Universal Resource Identifier) 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 further modified in order to make it possible to assert relations between instances of the knowledge base (i.e., `myHead isAdjacentTo myNeck`).

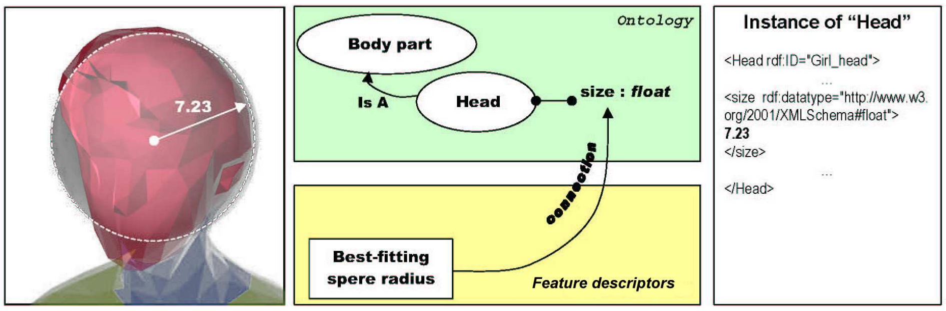
4.3 Automatic Annotation

Currently, our system requires the user to manually select the concepts to instantiate; for attributes and relations between instances, however, there is the 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, ...). Each of these measures is focused on a specific aspect of the features, either intrinsic or in their relationship with other connected features, and for this reason we call them *feature descriptors*.

Currently, they belong to the following two groups:

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

Some of these descriptors are immediately connectable with important prop-



```
<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 feature descriptor.

erties of the feature (e.g. area, enclosed volume), but we also allow their combination within formulae, in order to obtain higher-level descriptors. Some examples can be $\texttt{volume}^2/\texttt{area}^3$, which measures the compactness of a feature and is invariant to its uniform scaling, or $\texttt{genus} > 0$, which can be connected with the presence of holes. Anyhow, since feature descriptors are calculated independently of any ontology, the user may define their *interpretation* within each specific domain of annotation. The user may establish a set of *connections* between topological relations and conceptual descriptors (e.g. "feature adjacency" $\leftrightarrow$ `is_connected_to`) and between geometric descriptors and class attributes (e.g. "radius of best-fitting cylinder" $\leftrightarrow$ `through_hole` :: `radius`, $\texttt{genus} > 0 \leftrightarrow$ `pierced`).

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 features 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 in Figure 4.

Since we believe that modularity is crucial to provide a flexible annotation framework, we made our system able to load additional descriptors implemented externally as plug-ins, just as we have done for the segmentation algorithms, and to combine the descriptors available off the shelf in order to produce higher-level ones, through the help of a small editor in which the user can write formulae and save them for later use.

At this point, the connections can be established through a proper dialog in which all the feature descriptors are available in a drop-down menu; when one of them is chosen, 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 picked; this avoids, for example, the connection of a property with more than one descriptor, or between non-compatible descriptors and ontology properties (e.g. "feature 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.

The formulae used to combine simple descriptors into higher-level ones will be extended in the future to include conceptual characterizations in their bodies. For instance the formula "is-human AND `height` < `120`" could be used to create a (boolean) feature descriptor, connectable for instance with "is-children", allowing the construction of semantics on top of other forms of semantics in an automatic and layered way through the use of simple yet powerful forms of inference.

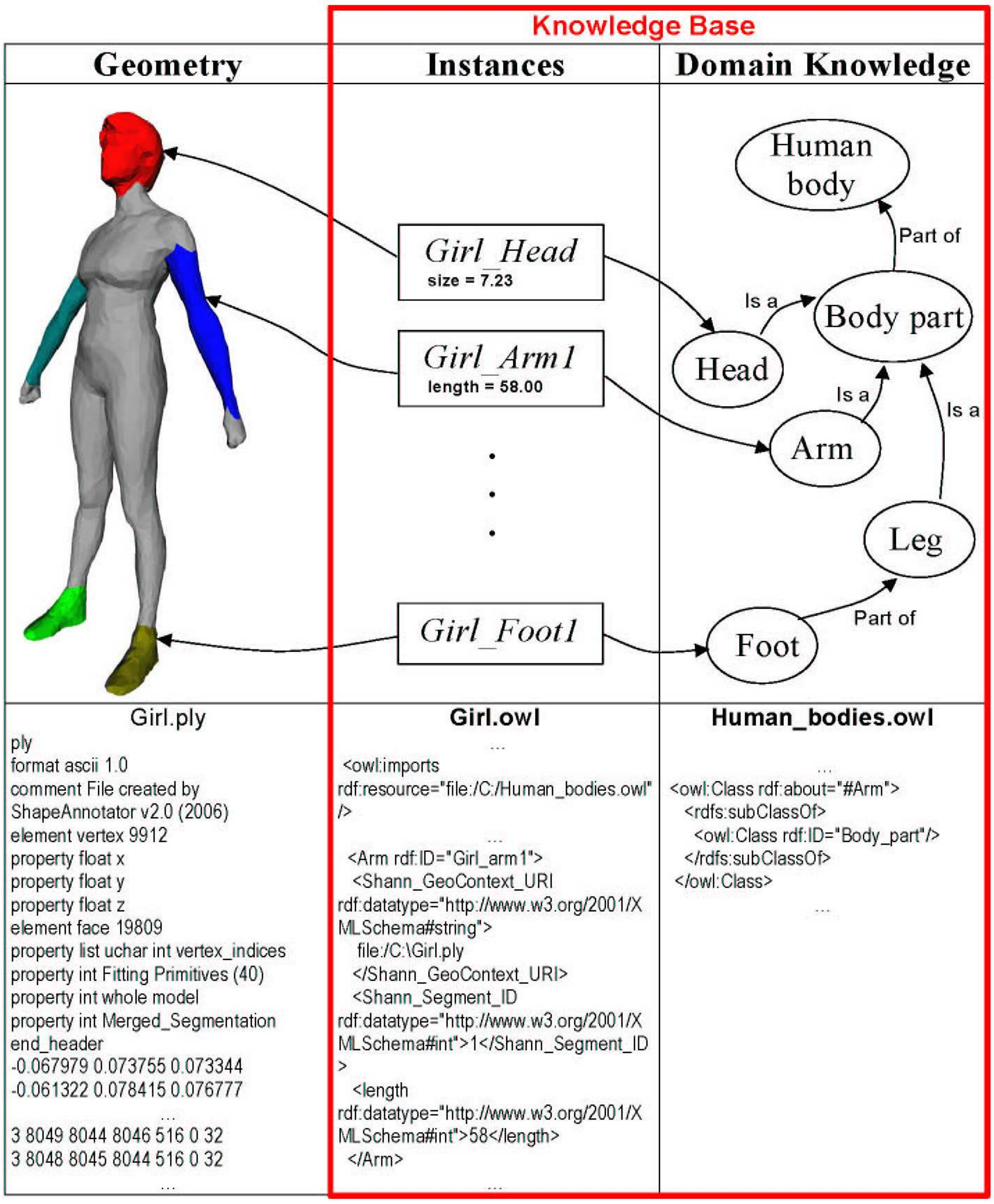
4.4 Encoding of the Annotated Features

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 [28]. 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



Girl.ply

```
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
...
```

Girl.owl

```
...
<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>
...
```

Human_bodies.owl

```
...
<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.

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 Application Scenarios

We present two scenarios that describe how the part-based annotation framework might be used.

5.1 Virtual character design

In several application domains, ranging from simulation to entertainment and gaming, the problem of designing virtual characters arises. Whereas in the case of simulation the virtual characters, along with their motion capabilities, are often modeled after real humans [19], in the case of entertainment and gaming avatars most often come from imaginative inspiration. The design is done most of the times from scratch or through the personalization of a given set of parameters. This is the current status for the avatar creation in many *MMORPGs* and in online virtual worlds such as Second Life [27], where avatars can be created constructively, by selecting from predefined sets of values the shape of the body, the skin, the hair, the eyes, the clothes, and so on. Thus, the actual creative freedom is limited to the selection and combination of predefined attributes and parameters. As an example, we consider the path of a user interested in harvesting and selecting digital models of human *heads* having specific high-level characteristics (e.g. large, narrow, belonging to a male, belonging to a given ethnic group, with a long nose, with distant eyes), starting from a repository containing virtual humans. Currently (s)he should browse the repository, calculate the parameters (s)he is interested in (e.g. distance between eyes, length of nose), download each interesting model, and perform editing operations in order to get the parts of the models corresponding to *heads*. All of these steps should be performed manually. Thanks to the ShapeAnnotator the resources in the repository will be annotated (the parameters will be calculated automatically), and a dedicated knowledge base will contain direct instances of *heads*, corresponding to portions of the original models in the repository. Each *head* will be also conceptualized via attributes and relations, and so the proper selection could be performed independently of the specific geometry. Moreover, when some interpretation rules will be coded, as it is foreseen in the ShapeAnnotator, the values of the parameters could be connected with some higher level characterization. Thus, the user will be able to search the repository directly for "heads of Caucasian adult male" or "heads of children", or "heads of black women". The virtual character creation effort could then be significantly eased.

5.2 E-manufacturing and product (re)design

In e-manufacturing, designers and engineers may collaborate remotely using Internet technologies to devise new products. As creating models from scratch is rather costly and time-consuming, modern design frameworks strive to foster the reuse of existing resources. Thus, to design original 3D product models, shape parts may be retrieved from distributed databases, adapted and virtually combined and assessed.

With current technologies there are two main problems:

- Metadata in state-of-the-art repositories are typically related to the whole shape. Thus, retrieving suitable parts that may belong to composite models in the repository is an extremely difficult task.
- Even if one assumes that a model can be retrieved, adapting it to the needs of the new product may be another hard task. In particular, if the model is reverse engineered, its parts are neither explicit nor annotated, and turning the polygon mesh into a feature-based model to be edited becomes a long and tedious operation.

Through the ShapeAnnotator, each model in the repository can be *abstracted* and represented as a combination of its meaningful features. Each such feature is described by an instance in the knowledge base in which attributes and relations with other features are explicit. Thus, all the relevant parts of all the models in the repository can be easily indexed and effectively retrieved. Moreover, once a reverse engineered polygon mesh has been retrieved, its features are explicit and their parameters properly instantiated, making the translation into an editable feature-based model much easier. A detailed application of our system in such a scenario is described in [10].

6 Conclusions

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 a few mouse clicks using the pipeline proposed. The introduction of feature descriptors along with their context-based interpretation represents a first step towards automatic annotation methods for the 3D domain.

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.

7 Acknowledgements

This work is supported by CNR research activity ICT-P03, by the EU-FP7 project FOCUS-K3D (Contract N. ICT-2007.4.2) and by the FIRB - SHALOM project (International Collaboration Italy-Israel, Contract N. RBIN04HWR8). 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.*

References

[1] M. Attene, S. Biasotti, M. Mortara, G. Patane, B. Falcidieno, Computational methods for understanding 3D shapes, Computers&Graphics 30 (3) (2006) 323–333.

[2] M. Attene, B. Falcidieno, M. Spagnuolo, Hierarchical mesh segmentation based on fitting primitives, The Visual Computer 22 (3) (2006) 181–193.

[3] M. Attene, S. Katz, M. Mortara, G. Patanè, M. Spagnuolo, A. Tal, Mesh segmentation - a comparative study, in: SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06), IEEE Computer Society, Washington, DC, USA, 2006, p. 7.

[4] M. Attene, F. Robbiano, M. Spagnuolo, B. Falcidieno, Part-based annotation of virtual 3d shapes, in: Proceedings of Cyberworlds' 07, spec. sess. on NASAGEM workshop, IEEE Computer Society Press, 2007, pp. 427–436.

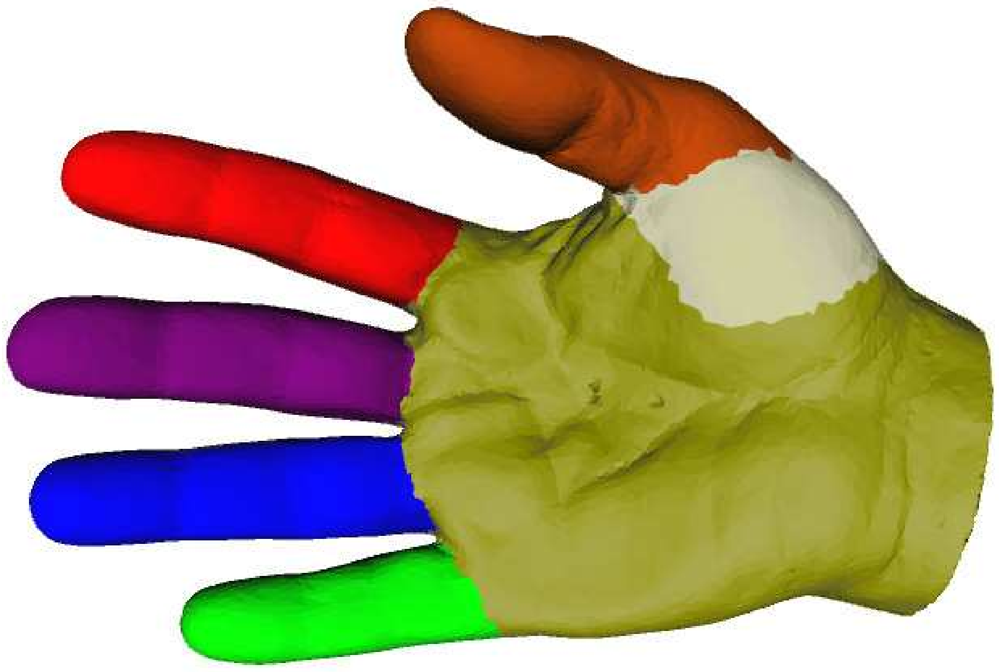


Fig. 6. Example of segmentation of a natural shape. The bright color on the hand surface indicates that the corresponding part is an overlap of features (in this case the thumb and the palm). Model courtesy of AIM@SHAPE Shape Repository.

[5] M. Attene, F. Robbiano, M. Spagnuolo, B. Falcidieno, Semantic annotation of 3d surface meshes based on feature characterization, in: Lecture Notes in Computer Science Vol. 4816 (SAMT'07 Procs.), 2007, pp. 126–139.

[6] S. Biasotti, Computational topology methods for shape modelling applications, Ph.D. thesis, University of Genoa, Italy (2004).

[7] I. Biederman, Recognition-by-Components: A theory of human image understanding., Phicological Review 94 (1987) 115–147.

[8] I. M. Bilasco, J. Gensel, M. Villanova-Oliver, H. Martin, An mpeg-7 framework enhancing the reuse of 3d models, in: Web3D '06: Proceedings of the eleventh international conference on 3D web technology, ACM, New York, NY, USA, 2006, pp. 65–74.

[9] P. Browne, A. F. Smeaton., Video retrieval using dialogue, keyframe similarity and video objects, in: ICIP 2005 - International Conference on Image Processing, 2005.

[10] C. Catalano, M. Attene, F. Robbiano, M. Spagnuolo, B. Falcidieno, Shape knowledge annotation for virtual product sharing and reuse, in: In Proceedings of the ASME Conference on Engineering Systems Design and Analysis, ESDA08, 2008, pp. 367–375.

[11] D. Cohen-Steiner, P. Alliez, M. Desbrun, Variational shape approximation, in:

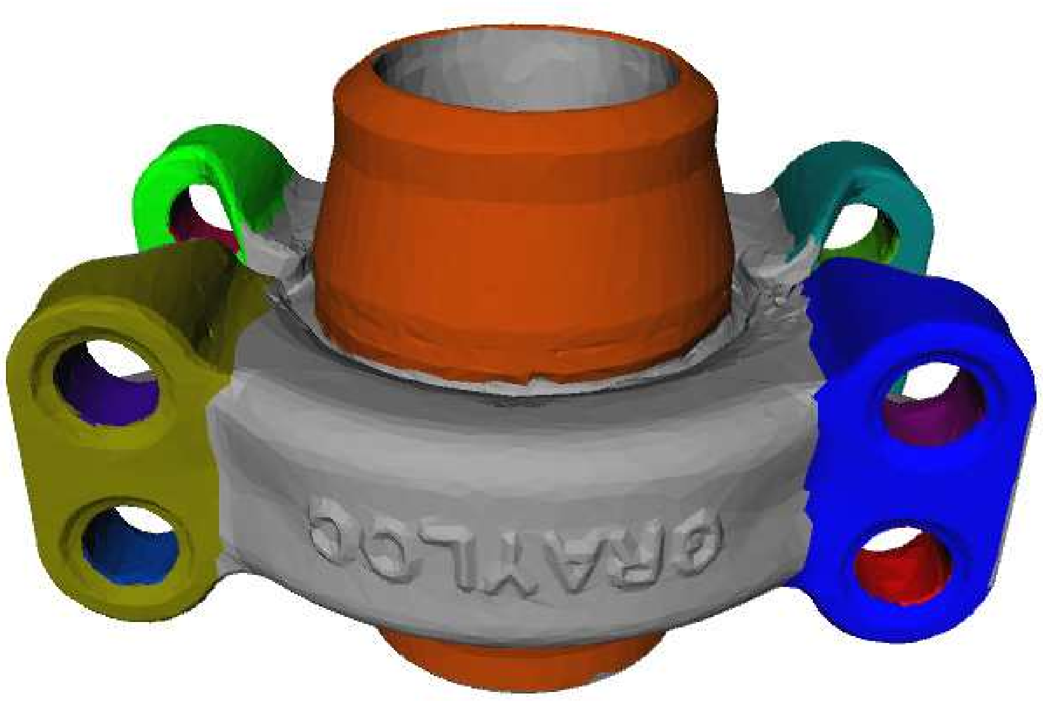


Fig. 7. Example of segmentation of an artificial shape. Model courtesy of AIM@SHAPE Shape Repository.

SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, ACM Press, New York, NY, USA, 2004, pp. 905–914.

[12] 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.

[13] T. R. Gruber, Towards principles for the design of ontologies used for knowledge sharing, in: N. Guarino, R. Poli (eds.), Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, Deventer, The Netherlands, 1993.

[14] N. Guarino, Formal ontology and information systems, in: N. Guarino (ed.), Formal Ontology in Information Systems, IOS Press, Amsterdam, 1998, pp. 3–18.

[15] A. Guéziec, G. Taubin, F. Lazarus, B. Horn, Cutting and Stitching: Converting Sets of Polygons to Manifold Surfaces, IEEE Transactions on Visualization and Computer Graphics (2001) 136–151.

[16] T. Hermes, A. Miene, P. Kreyenhop, On textures: A sketch of a texture-based image segmentation approach, in: R. Decker, W. Gaul (eds.), Classification and Information Processing at the Turn of the Millenium, 2000, pp. 210–218, proc. 23rd Annual Conference Gesellschaft für Klassifikation e.V. 1999.

[17] I. Horvath, J. Vergeest, G. Kuczogi, Development and application of design concept ontologies for contextual conceptualization, in: ASME Design Engineering Technical Conference, 1998.

[18] H.-B. Jun, D. Kiritsis, P. Xirouchakis, Product lifecycle information modelling with rdf, in: PLM05, 2005, pp. 44–54.

[19] P. Kalra, N. Magnenat-Thalmann, L. Moccozet, G. Sannier, A. Aubel, Real-time animation of realistic virtual humans, IEEE Computer Graphics and Applications 18 (5) (2006) 42–55.

[20] D. Marr, Vision - A computational investigation into the human representation and processing of visual information, W. H. Freeman, San Francisco, 1982.

[21] M. Mortara, G. Patané, M. Spagnuolo, From geometric to semantic human body models, Computers & Graphics 30 (2) (2006) 185–196.

[22] M. Mortara, G. Patanè, M. Spagnuolo, B. Falcidieno, J. Rossignac, 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, ACM Press, New York, NY, USA, 2004, pp. 339–344.

[23] S. Nichols, Building the 3d internet, itnews, http://www.itnews.com.au/news/newsstory.aspx?story=42704 (2006).

[24] [online], Flickr. http://www.flickr.com/.

[25] [online], Photostuff. http://www.mindswap.org/2003/photostuff/.

[26] [online], Riya. http://www.riya.com/.

[27] [online], Secondlife. http://www.secondlife.com.

[28] [online], The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3dscanrep.

[29] [online], 3D CAD browser. http://www.3dcadbrowser.com/ (2001).

[30] [online], The NCI cancer ontology. http://www.mindswap.org/2003/cancerontology (2003).

[31] [online], AIM@SHAPE shape repository. http://shapes.aimatshape.net/ (2004).

[32] [online], Network of excellence AIM@SHAPE, EU FP6 IST NoE 506766. http://www.aimatshape.net (2004).

[33] [online], OWL web ontology language guide. http://www.w3.org/tr/2004/rec-owl-guide-20040210/, w3C Recommendation (Feb 2004).

[34] [online], The Protégé ontology editor. http://protege.stanford.edu (2006).

[35] H. Opitz, A Classification System to Describe Workpieces, Pergamon Press, NY, 1970.

[36] T. Pavlidis, Y.-T. Liow, Integrating region growing and edge detection, IEEE Trans. Pattern Anal. Mach. Intell. 12 (3) (1990) 225–233.

[37] K. Petridis, D. Anastasopoulos, C. Saathoff, N. Timmermann, I. Kompatsiaris, S. Staab, 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, 2006.

[38] A. Shamir, Segmentation and shape extraction of 3D boundary meshes, in: Eurographics 2006 - State of the Art Reports, 2006, pp. 137–149.

[39] P. Shilane, P. Min, M. Kazhdan, T. Funkhouser, The princeton shape benchmark, in: SMI '04: Proceedings of the Shape Modeling International 2004 (SMI'04), IEEE Computer Society, Washington, DC, USA, 2004, pp. 167–178.

[40] Y. Xiaohan, J. Ylä-Jääski, Y. Baozong, A new algorithm for texture segmentation based on edge detection, Pattern Recognition 24 (11) (1991) 1105–1112.

[41] E. Zhang, K. Mischaikow, G. Turk, Feature-based surface parameterization and texture mapping, ACM Transactions on Graphics 24 (1) (2005) 1–27.