# 3D-mesh segmentation: automatic evaluation and a new learning-based method

Halim Benhabiles



Numéro d'ordre: 40589


UNIVERSITÉ LILLE 1 SCIENCES ET TECHNOLOGIES

LABORATOIRE D'INFORMATIQUE FONDAMENTALE DE LILLE

ECOLE DOCTORALE SCIENCES POUR L'INGÉNIEUR UNIVERSITÉ LILLE


# THÈSE

Présentée en vu d'obtenir le grade de Docteur,
spécialité Informatique

par


Halim Benhabiles


# 3D-MESH SEGMENTATION: AUTOMATIC EVALUATION AND A NEW LEARNING-BASED METHOD

Thèse soutenue le 18 Octobre 2011 devant le jury composé de :

| | | | |
|---|---|---|---|
| M. | PHILIPPE PREUX | Professeur, Université Lille 3 | (Président) |
| M. | CHRISTOPHE ROSENBERGER | Professeur, ENSICAEN | (Rapporteur) |
| Mme | MICHELA SPAGNUOLO | Senior Researcher, IMATI-GE / CNR | (Rapporteur) |
| M. | FLORENT DUPONT | Professeur, Université Lyon 1 | (Examinateur) |
| M. | FRANÇOIS GOULETTE | Maître-Assistant (HDR), MINES ParisTech | (Examinateur) |
| M. | MOHAMED DAOUDI | Professeur, TELECOM Lille 1 | (Directeur) |
| M. | GUILLAUME LAVOUÉ | Maître de Conférences, INSA-Lyon | (Co-encadrant) |
| M. | JEAN-PHILIPPE VANDEBORRE | Maître de Conférences, TELECOM Lille 1 | (Co-encadrant) |

*This manuscript is dedicated*
*to all those who contributed, from near or far, to its elaboration.*

# Acknowledgements

First of all I would like to thank my different supervisors, Pr. Mohamed Daoudi, Dr. Jean-Philippe Vandeborre, and Dr. Guillaume Lavoué, with whom I really appreciated to work this last three years and I discovered the scientific research world, in particular that of the 3D-multimedia field. I also thank them for their trust, daily encouragements, and their useful advises.

Special thank goes to the members of committee, in particular, Mme Michela Spagnuolo (senior researcher, IMATI-GE / CNR) and M. Christophe Rosenberger (professeur, ENSICAEN) for accepting to review my thesis and for their interesting comments. I also thank the rest of the members; Florent Dupont (professeur, Université Lyon 1) M. François Goulette (Maître-Assistant (HDR), MINES ParisTech), and M. Philippe Preux (professeur, Université Lille 3). All these people made me the honor of being present the day of my oral presentation despite their busy schedules.

I also thank the different members of the MIIRE and FOX teams who were more than colleagues! I thank them for the very good atmosphere of work, for their help on a technical level, and for all the foot-ball games that we organized together!

I thank my former supervisors Mme and M. Baba Ali (professeurs USTHB), thanks to whom I made a first step in the scientific research world during the preparation of my engineer degree.

I would like to express my gratitude to my parents who always were present to encourage me and support me.

Special thank goes to my maternal aunt who lives in France, her husband, and their children (my cousins) for their warm welcome and their help.

Finally, I thank people at TELECOM Lille 1, especially the technical

SIIR service for their kindness, and all the contributors to the JMesh project for the quality of their algorithms.

# Author's publications

## International publications

### Journals

- **Halim Benhabiles**, Guillaume Lavoué, Jean-Philippe Vandeborre and Mohamed Daoudi,
  "*Learning boundary edges for 3D-mesh segmentation*",
  **Computer Graphics Forum** - Eurographics Association - Ed. Blackwell, DOI: 10.1111/j.1467 − 8659.2011.01967.x, published on-line, 2011.

- **Halim Benhabiles**, Jean-Philippe Vandeborre, Guillaume Lavoué and Mohamed Daoudi,
  "*A comparative study of existing metrics for 3D-mesh segmentation evaluation*",
  **The Visual Computer** - International Journal of Computer Graphics, Springer Editions, volume 26, number 12, pp. 1451-1466, December 2010.

### Conferences

- **Halim Benhabiles**, Guillaume Lavoué, Jean-Philippe Vandeborre and Mohamed Daoudi,
  "*A subjective experiment for 3D-mesh segmentation evaluation*",
  **IEEE International Workshop on Multimedia Signal Processing (MMSP) 2010**, pp. 356-360.

- **Halim Benhabiles**, Jean-Philippe Vandeborre, Guillaume Lavoué and Mohamed Daoudi,
  "*A framework for the objective evaluation of segmentation algorithms using*

*a ground-truth of human segmented 3D-models*",
**IEEE International Conference on Shape Modeling and Applications (SMI) 2009**, pp. 36-43.

**Software**

- **Halim Benhabiles** and Souhaib Fatan,
  "*LIFL/LIRIS Benchmark*"
  An automatic tool to evaluate mesh segmentation algorithms (paper:"*A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models*"), implementation available on-line.
  `http://www-rech.telecom-lille1.eu/3dsegbenchmark/`.

## Local publications

**Conferences**

- **Halim Benhabiles**, Guillaume Lavoué, Jean-Philippe Vandeborre, Mohamed Daoudi,
  "*Une expérimentation subjective pour l'évaluation de segmentations de maillages 3D*",
  **Compression et Représentation des Signaux Audiovisuels (CORESA) 2010**, pp. 43-47.

- **Halim Benhabiles**, Jean-Philippe Vandeborre, Guillaume Lavoué, Mohamed Daoudi,
  "*Une collection de modèles 3D avec vérité-terrain pour l'évaluation objective des algorithmes de segmentation*",
  **Compression et Représentation des Signaux Audiovisuels (CORESA) 2009**, pp. 104-109.

- **Halim Benhabiles**, Guillaume Lavoué, Jean-Philippe Vandeborre,
  "*Une nouvelle métrique de similarité pour l'évaluation de segmentations de maillages 3D en utilisant un corpus de vérités-terrains*",
  **Journées de l'Association Française d'Informatique Graphique (AFIG) 2009**, pp. 65-72.

# Contents

# List of Figures

# 1 Introduction

With the recent technological developments concerning three-dimensional images (3D scanners, 3D graphic accelerated hardware, and so on; see figure 1.1), the creation and storage of three-dimensional models have become a reality. The usage range of these three-dimensional models is wide: cultural heritage, medical and surgical simulation, CAD design, video games, multimedia applications, etc (see figure 1.2).


Figure 1.1 – *3D scanner (©Faro) and GeForce 3D accelerated graphic card (©NVIDIA).*

Consequently to the growing usage of three-dimensional models, the scientific community produces a lot of works about the processing of these 3D-data for various computer graphic applications such as modeling, indexing, watermarking and compression.

The three-dimensional models are generally represented as meshes of polygons (generally triangles). This kind of representation has the advantage of being perfectly adapted to 3D display with the help of modern

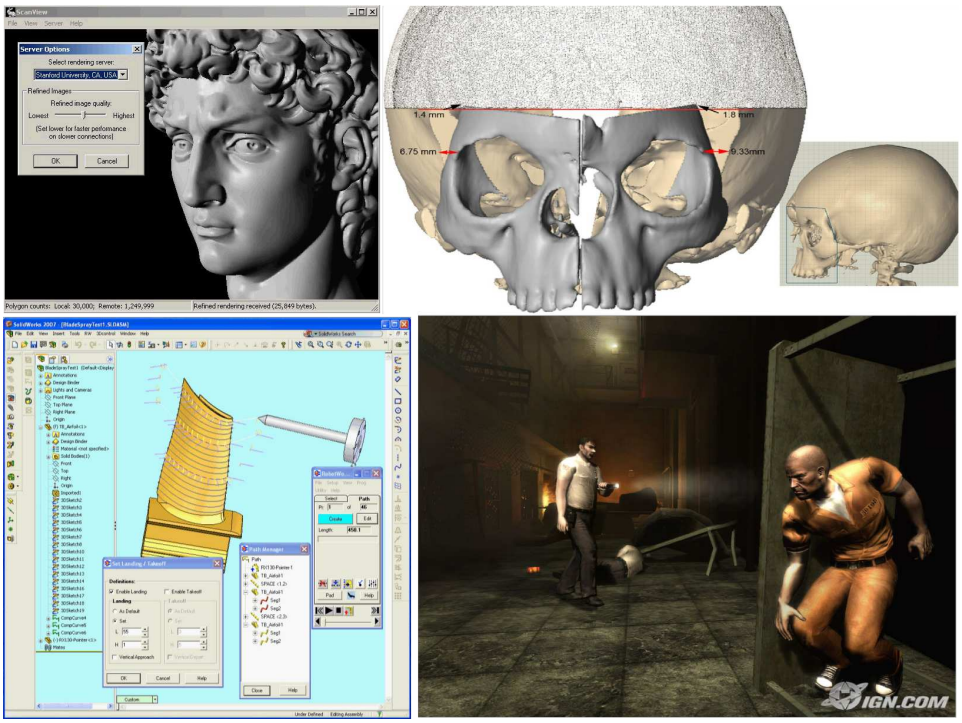



Figure 1.2 – *From left to right and top to bottom: Remote visualization of stored 3D models (©the Leland Stanford Junior University), life-changing facial reconstruction for young Child (©Sensable), solid modelling 3D-CAD design (©Compucraft, Ltd), splinter cell 3D-video game (©UBISOFT).*

3D accelerated hardware. But the main drawback of this format is the lack of a structure or a hierarchical description that could be very useful for the applications cited above. Hence, the automatic segmentation of 3D-meshes is very often a necessary pre-processing tool for these applications. Mesh segmentation consists in subdividing a polygonal surface into patches of uniform properties either from a strictly geometrical point of view or from a perceptual / semantic point of view.

Many systems were and are still currently developed for the segmentation of bidimensional data (images or videos). However these solutions are not really effective or not easily adaptable to intrinsically three-dimensional data. Indeed, the segmentation algorithms proposed so far in the literature for three-dimensional meshes require geometrical and/or topological descriptors which characterize the shape parts either from a geometric point of view or from a semantic point of view. Defining such descriptors is not an obvious task, and existing ones are still suffering from many limitations such as sensitivity to geometric noise, sensitivity

to topological changes, etc. Moreover, one could easily notice that, contrary to the 2D-data domain, there is neither protocol, nor standard data collection that allow the researchers to evaluate and compare existing and new 3D segmentation methods. In this context, and within the framework of the MADRAS project [1], the objectives of this thesis are to address two main problems namely the quantitative evaluation of mesh segmentation algorithms and learning mesh segmentation by exploiting the human factor.

## 1.1 Contributions

In this thesis we bring three different contributions: the first two ones are related to the problem of quantitative evaluation of mesh segmentation, and the last one is related to the problem of learning mesh segmentation.

**A benchmark for 3D-mesh segmentation evaluation.** We propose a benchmark for 3D-mesh segmentation evaluation which includes a ground-truth corpus and a set of similarity metrics. The corpus is composed of a set of 3D-models grouped in different classes and associated with several manual segmentations produced by human observers. The metrics allow to measure the similarity between the reference segmentations from the corpus and that obtained by an algorithm (on the same models). The quality of segmentations obtained by automatic algorithms is then evaluated automatically in a quantitative way thanks to the metrics, and on an objective basis thanks to the ground-truth corpus. Besides, we propose a thorough study and comparisons of existing metrics addressing the assessment problem of mesh segmentation together with a new measure of segmentation similarity that allows to quantify the consistency between multiple segmentations of a model. We show that this new metric outperforms existing ones in terms of properties and discriminative power.

[1] MADRAS (3D Models And Dynamic models Representation And Segmentation) is a French research project sponsored by ANR (The French National Research Agency) – ref. ANR-07-MDCO-015 – `http://www-rech.telecom-lille1.eu/madras/`.

**A subjective experiment.** We present a subjective quality assessment experiment for 3D-mesh segmentation. To this end, we carefully designed a protocol with respect to several factors namely the rendering conditions, the possible interactions, the rating range, and the number of human subjects. To carry out the subjective experiment, 50 human observers have rated a set of 250 segmentation results issued from various algorithms. The obtained *Mean Opinion Scores*, which represent the human subjects' point of view toward the quality of each segmentation, have then been used to evaluate the quality of automatic segmentation algorithms and to validate the relevance of our benchmark in term of quality of the ground-truth corpus and of discriminative power of the new metric.

**A new 3D-mesh segmentation algorithm with machine learning techniques.** We propose a new fully automatic 3D-mesh segmentation algorithm based on boundary edge learning. Our algorithm is carried out using two main steps: an off-line step in which an objective boundary edge function is learned from a set of human segmented training meshes, and an on-line step in which the learned function is used to segment any input 3D-mesh. The edge function is determined based on multiple geometric feature calculations and Adaboost learning, and then is used in a processing pipeline (on-line step) to produce smooth closed boundaries. The processing pipeline includes the following stages: thinning, closing contour, and snake movement. The battery of experiments conducted using different benchmarks (benchmark[2] from Chen *et al.* (CGF09) and our own benchmark) demonstrates the performance superiority of our algorithm over the state-of-the-art of mesh segmentation algorithms. We propose an application of our segmentation algorithm for kinematic skeleton extraction of dynamic 3D-meshes and we show that the early obtained results are promising.

## 1.2 Outline

The manuscript of this thesis is organized as follows:

[2] http://segeval.cs.princeton.edu/

Chapter 2 reviews the state-of-the-art of 3D-mesh segmentation algorithms and their evaluation methods. The chapter points out on one hand the fact that plenty of mesh segmentation algorithms were and are still developed due to the importance of this processing task for mesh understanding and analysis. On the other hand it points out the need of automatic tools to evaluate/compare the segmentation quality of existing and new mesh segmentation algorithms.

Chapter 3 presents our benchmark dedicated to the evaluation of mesh segmentation algorithms. The chapter describes the different steps and protocol followed to create the ground-truth corpus. It presents also an extensive experimental comparison between our new proposed metric and existing ones. The experiments include subjective tests that allow to validate the discriminative power of the new metric.

Chapter 4 presents our new segmentation algorithm based on a learning approach together with extensive experiments that demonstrate its relevance. It presents also an application of this new segmentation algorithm which consists of extracting kinematic skeletons for dynamic meshes.

Chapter 5 concludes the manuscript and provides a discussion about future work.

# 2 State-of-the-art of 3D-mesh segmentation and evaluation

Contents

This chapter reviews the state-of-the-art of 3D-mesh segmentation algorithms and their evaluation methods. First, we define the mesh segmentation problem and classify the different existing techniques of the state-of-the-art. We briefly discuss about the advantages and drawbacks of each technique. Then, we present the different types of segmentations while listing out for each one of them some application examples. Finally, we emphasize the need of automatic tools for the quality assessment of mesh segmentation. We present the different existing evaluation methods in the field of 2D-image since the most significant proposed works for the 3D-mesh segmentation evaluation are based on the same methodology as that proposed in the 2D-image domain. Then we summarize the few existing works in the literature for mesh segmentation evaluation.

## 2.1 Introduction to polygon mesh

The passage from the reality to the computer modeling, begins as in any domain activity, with an idea of the designer, and then is translated to a computer representation. This computer representation is called *model*, and the translation of the idea to the computer representation is called *modeling*.

The model allows to understand or to visualize the structure or behavior of the entity. It allows also to experiment manipulations and observe their effects.

In the different fields of study, there are several models: mathematical models for meteorology for instance, chemical models such as the Bohr atom model, behavioral models in psychology, etc.

Since this thesis is related to the computer graphics field, we are interested in 3D modeling based on geometric models. A geometric model describes the shape of a 3D space object that it represents.

There exist several tools to represent these shapes, for instance: surface or curve modeling (NURBS, B-spline, etc.), constructive solid geometry, voxels, etc. These tools are generally used to create primitive shapes which are gathered to obtain the final model. Basically, the model created is converted to a polygon mesh in order to be efficiently displayed by the computer.

Nowadays the most popular way to represent a geometric model is polygon meshes.

A polygon mesh is a collection of connected polygons (i.e. bounded planar surfaces of different sizes) which allows to approximate the surface of any 3D object. As shown in figure 2.1 the polygons can be triangles, quadrangles, or any kind of polygons. With such a mesh, it is possible to obtain an approximation of any surface. Thus, we can consider that the transformation of any surface to a polygon mesh is a discretization more or less fine of this surface.

The main reasons that make polygon meshes very useful is the fact that 3D specialized graphic cards are designed to optimize their display. Indeed, to obtain a rendering of such a mesh, only an algorithm more

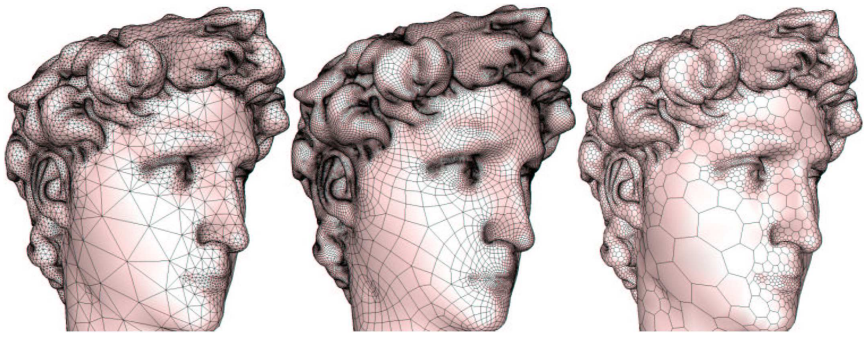


Figure 2.1 – *David model presented by a triangle, quadrangle, and polygon mesh (ADIV03).*

or less evolved (Wireframe, Gouraud shading, Phong shading, etc.) is used to draw the polygons one by one. These algorithms are integrated in the graphics processing units. Moreover, even current three dimensional scanners use techniques which provide a 3D-model of the object scanned under the form of a polygon mesh. Finally, this format is universal since all three dimensional representations can be converted to polygon meshes.

As part of our thesis, we restrict our discussion to the processing of a specific kind of polygon meshes, namely closed two-manifold triangle meshes.

In what follow we present through a set of definitions the notions of 3D triangle meshes, two-manifold meshes, and closed meshes.

**Definition 2.1** *(3D triangle mesh) A three-dimensional triangle mesh $M$ is defined as a tuple $\{V, E, F\}$ of vertices $V$, edges $E$, and triangles (or facets) $F$:*

$$V = \{v_i | v_i \in \mathbb{R}^3, 1 \leq i \leq m\} \tag{2.1}$$

$$E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V, i \neq j\} \tag{2.2}$$

$$F = \{f_{ijk} = (v_i, v_j, v_k) | v_i, v_j, v_k \in V, i \neq j, i \neq k, j \neq k\} \tag{2.3}$$

**Definition 2.2** *(Two-manifold mesh) A three-dimensional triangle mesh $M$ is two-manifold if every vertex $v$ of $M$ has a neighborhood homeomorphic to a disk or a half disk (see figure 2.2).*

**Definition 2.3** *(Closed two-manifold mesh) A closed two-manifold mesh $M$ is a two-manifold mesh which does not contain any boundary edges. A boundary edge $e$ is an edge which has only one adjacent facet $f$.*

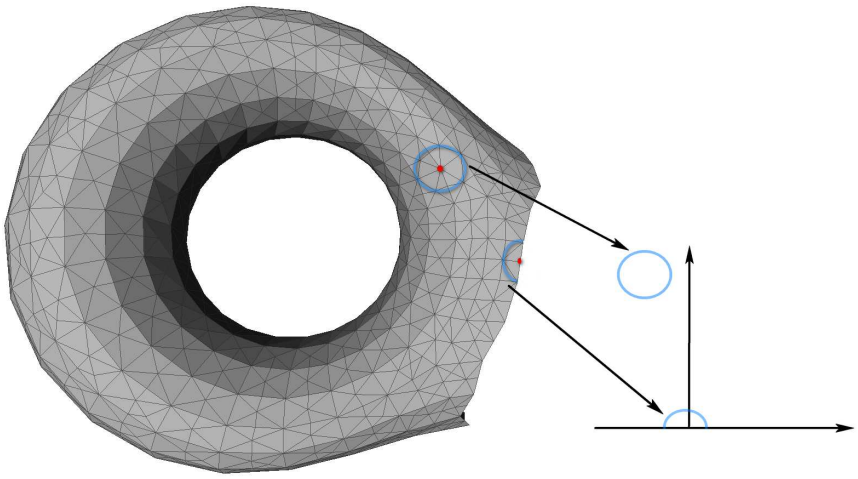


Figure 2.2 – *Example of non-closed two-manifold 3D-mesh.*

As we have pointed out, polygon mesh particularly triangle mesh is an excellent format for 3D display since it can be directly processed by the specialized devices. However, the main drawback of a polygon (or triangle) mesh is the lack of structure or hierarchical description. Indeed, if the object is composed of a set of significant parts such as head, arms, legs of a human model, its representation by a polygon mesh will lead to the loss of its semantic (or the composition information). Consequently, it will not be possible to deal with the different object parts. From this statement, it appears the need of 3D-mesh segmentation.

## 2.2 Mesh segmentation problem

Mesh segmentation consists in decomposing a polygonal surface into different regions (i.e. connected set of vertices or facets) of uniform properties, either from a *geometric* point of view or from a *semantic* point of view. It is a critical step toward content analysis and mesh understanding.

A formal definition of mesh segmentation is the following:

**Definition 2.4** *(Mesh segmentation S) Given a mesh M, and the set of mesh elements R which corresponds to V, E, or F. The segmentation S of M is the set of sub-meshes* $S = \{M_0, ..., M_{k-1}\}$ *induced by the partitioning of R into k disjoint sub-sets* ($R = \{R_0, ..., R_{k-1}\}$).

**Definition 2.5** *(Sub-mesh) Let us consider* $R'$ *as a sub-set of R* ($R' \subset R$). *Thus, we can create a*

*sub-mesh $M' \subset M$ by choosing all vertices which are included in $R'$ like $V'$, and then define $M' = \{V', E', F'\}$, where:*

$$E' = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V', i \neq j\} \quad (2.4)$$

$$F' = \{f_{ijk} = (v_i, v_j, v_k) | v_i, v_j, v_k \in V', i \neq j, i \neq k, j \neq k\} \quad (2.5)$$

Shamir (Sha08) proposed to define the mesh segmentation problem as an optimization problem.

**Definition 2.6** *(Mesh segmentation as an optimization problem) Given a mesh $M$, and the set of elements $R \in \{V, E, F\}$, find a disjoint partitioning of $R$ into $R_0$,..., $R_{k-1}$ such that the criterion function $J = J(R_0, ..., R_{k-1})$ be minimized (or maximized) under a set of constraints $C$.*

The mesh segmentation process uses both criterion and constraint sets for partitioning. These two sets are generally related to the end-application. Shamir (Sha08), however, proposed to separate them from the objective of the segmentation process. The set of constraints includes conditions on the sub-sets $R_i$, such as a restriction on their minimum/maximum size, and on sub-meshes $M_i$ with respect to their geometry or topology. The criteria that decide which elements have to belong to the same segment include the attributes of these elements such as planarity and normal directions (AFS06), geodesic distances (KT03), dihedral angles (ZTS02), etc.

## 2.3 Mesh segmentation techniques

There are different ways to classify segmentation algorithms. In this section, we propose to classify them according to their characteristics and how much user intervention they need. Note that this classification is not exhaustive; we just give an overview of the most popular techniques and briefly discuss their advantages and drawbacks. Interested readers are referred to different surveys proposed in the literature (Sha08, APP*07, AKM*06) for more details.

### 2.3.1 Region growing

Region growing is the most intuitive method to segment a mesh (LDB05b, ZH04, ZPK*02, RB02, ZTS02, MW99). As described in algorithm 1, it starts by selecting a seed element (a vertex), and then the growing is realized by adding successively compatible elements (vertices which satisfy a given criterion). This leads to create a region (or segment). The growing process is repeated with a new seed element each time the previous growing is interrupted. The algorithm stops when all the seed elements are visited.

**Algorithm 1**: Region growing

1: *Initialize a queue Q*
2: *Select seed elements s and insert them into Q*
3: **while** *Q is not empty* **do**
4: *Get the next seed element* $s_i$ *from Q*
5: *Define a new region* $R_i$
6: *Add* $s_i$ *to* $R_i$
7: *Add all compatible elements with* $s_i$ *to* $R_i$
8: **end while**

The main difference between the region growing algorithms lies in the choice of the criterion which decides whether an element can be added to a given region.

Zeckerberger *et al.* (ZTS02) have proposed two segmentation methods of which one of them is based on a region growing algorithm. The algorithm starts by computing the dual graph of the input mesh. Each node of the graph represents a facet of the mesh, and the arcs which connect the nodes represent the adjacency relation between the facets. Then, the algorithm randomly selects a node (seed element) and goes through the graph while collecting facets which form a convex patch (or convex region). The process of computing a new patch is launched each time the previous one is interrupted because of the violation of the convexity.

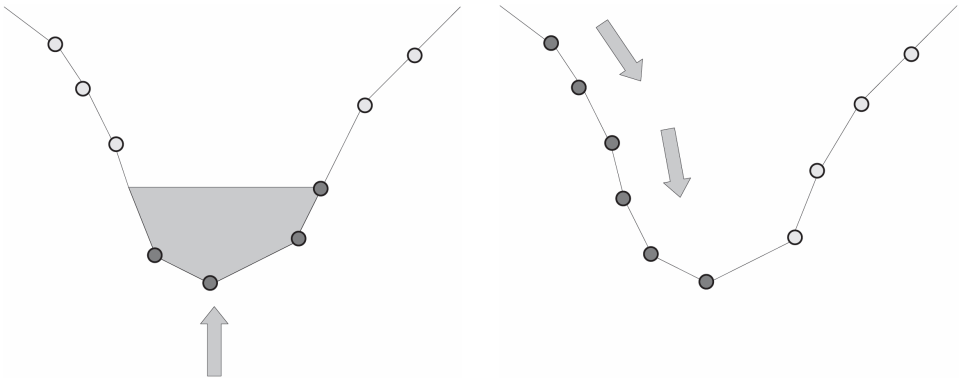
The segmentation method proposed by Lavoué *et al.* (LDB05b) is also based on a region growing algorithm. The curvature is first calculated for all vertices of the mesh, and classified into several clusters. A region

growing mechanism then extracts connected regions (associated with similar curvature), starting from several seed-facets.

One of the drawbacks of region growing techniques is the dependency on the initial seeds. Indeed, a bad choice of seed elements can lead to a bad segmentation. Another drawback is the over-segmentation (multiple small patches) induced by an important number of seed elements.

### 2.3.2 Watershed

Watershed is a popular method used to segment 2D-images. The fundamental principle of this method is segmenting regions into catchment basins. Many existing works in the literature have adapted this technique for 3D-mesh segmentation (CG06, YGZS05, AGC*05, PKA03, ZTS02, MW99). Based on a prior defined hight function over the vertices of the mesh, the local minima are detected (vertices which do not have lower level neighbors). Then catchment basins are associated with these minima based on one of the two following strategies: the first one (ascendant approach, see figure 2.3(a)) is to progressively *flood* the minima until the neighbor basins touch each other. The second one (descendant approach, see figure 2.3(b)) is to stream a drop along the steepest gradient until reaching a minimum and then labeling the path traversed by the drop. However, in both strategies, a special case should be considered. It corresponds to flat regions (or *plateaus*) which consist of a set of connected vertices (not only one vertex) which does not have lower level neighbors.



(a) *Ascendant approach* (b) *Descendant approach*

Figure 2.3 – *Watershed strategies.*

Algorithm 2 summarizes the different steps of the watershed segmentation process. The algorithm uses a hierarchical queue (HQ) to optimize the *flood* simulation process in term of computation time. Each queue is associated to a level of the height function which can be the curvature for instance. The priority of processing queues corresponds to the lowest level of the height function.

**Algorithm 2**: Watershed

```
1: Compute the height function for each vertex of the input mesh (curvature for
   instance)
2: Find flat regions (plateaus)
3: Find local minima and assign each one a label
4: Insert the minima in the HQ according to the level of their height function
5: while HQ is not empty do
6:    Get the next vertex x from HQ
7:    Get the vertex x neighbors which are not labeled yet
8:    for each non labeled neighbor do
9:       Assign the neighbor the same label as x
10:      if the neighbor does not belong to plateaus then
11:         Insert the neighbor in the HQ according to its height function level
12:      end if
13:   end for
14: end while
```

Zeckerberger *et al.* (ZTS02) proposed a segmentation method based on a watershed algorithm. They defined a simple height function $h$ computed for each edge of the mesh. $h = 1 - \cos(\alpha)$, where $\alpha$ is the dihedral angle of the edge (angle between the normals of the two facets sharing this edge). Then each facet is associated with the edge that has the lowest height. Thus, all adjacent planar facets will be clustered into the same segment. Figure 2.4 illustrates the result of segmentation, using the watershed and based on the height function defined above, for a steps model. Note that the facets that lie on the same plan belong to the same patch, and thus each step is decomposed into two segments.

Chen and Georganas (CG06) proposed a watershed method which use

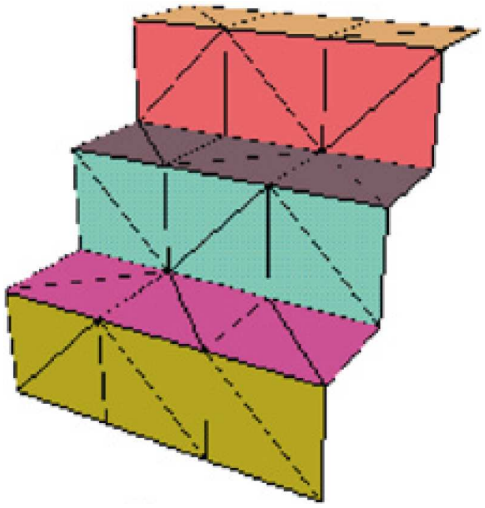


Figure 2.4 – *Segmentation result based on the watershed algorithm from Zeckerberger* et al. *(ZTS02).*

the Gaussian curvature together with the concaveness information. The specificity of their method is its ability to segment highly detailed meshes (high number of vertices) thanks to the use of what they defined as extended multi-ring neighborhood. This latter one consists of considering only the $i^{th}$ level of vertices connected to a given vertex while excluding the vertices of inferior levels $(i-1)$. The algorithm extracts features based on Gaussian curvature and concaveness estimation, then applies an ascendant watershed approach to segment the mesh into meaningful parts.

Watershed algorithms can be seen as *multiple* region growing algorithms of which the seed elements correspond to local minima. Consequently, the quality of the segmentation depends highly on the height function definition. The algorithms suffer also from over-segmentation due to the noise on the polygonal surface and its complexity. The classical solution to solve this problem is to merge segments in order to eliminate non-significant ones. Chen and Georganas (CG06) have integrated this latter solution in their algorithm and considered to this end two constraints: the size of regions (or segments), and the length of boundaries.

### 2.3.3 Hierarchical clustering

In hierarchical clustering, each segment is initially represented by a unique mesh element (a facet for instance). Each pair of adjacent segments is as-

signed a merging score based on a given criterion (AFS06, GG04, GWH01, She01). A merging process is then realized according to the increasing order of the scores. Algorithm 3 describes the general schema of the hierarchical clustering.

**Algorithm 3**: Hierarchical segmentation

```
1: Initialize a queue Q
2: Insert all possible pairs of adjacent regions in Q according to the increasing order of
   merging scores
3: while Q is not empty do
4:    Get the next pair (u, v) from Q
5:    if the pair (u, v) can be merged then
6:       Merge (u, v) in w
7:       Insert w in Q
8:       Update Q
9:    end if
10: end while
```

Sheffer (She01) proposed a clustering algorithm based on compatibility criteria. The algorithm is carried out through two stages: adjacency graph computation, and a graph contraction algorithm. In the first stage each cluster, which is initially represented by one facet, is mapped to a graph node, and the adjacency between two clusters is mapped to a graph arc. In the second stage, each arc is assigned a weight that corresponds to the improvement in the shape properties when merging the two clusters connected by the given arc. The weight is a combination of a set of geometric indices which are the following:

- Uncontractable arcs used to detect whether two clusters cannot be merged. This happens when the edge is non-manifold.
- Boundary preservation used to preserve clusters with sharp boundaries and do not merge them. The computation of this index is based on dihedral angle of edges.
- Region size used to control the size of clusters and decide if they have to undergo a merging operation or not. The computation of

this index is based on the area of the cluster, and the length of its perimeter.

- Boundary shape used to join clusters with longer shared boundaries of which the angle is obtuse.
- Region curvature change used to maintain the compatibility of curvature when merging two clusters.

Attene *et al.* (AFS06) proposed a segmentation algorithm based on fitting primitives. The set of primitives includes plan, sphere, and cylinder. The algorithm generates a binary tree of segments where each segment is fitted to one of the employed primitives. To this end, all possible pairs of adjacent segments are considered (initially each pair of segment is represented by two adjacent facets), and the pairs that are fitted well with one of the defined primitives, form a new segment. The authors proposed to use the $L^2$ distance to approximate the fitting error between segments and primitives. The distance allows to select the primitive that covers as well as possible the segment (see figure 2.5).

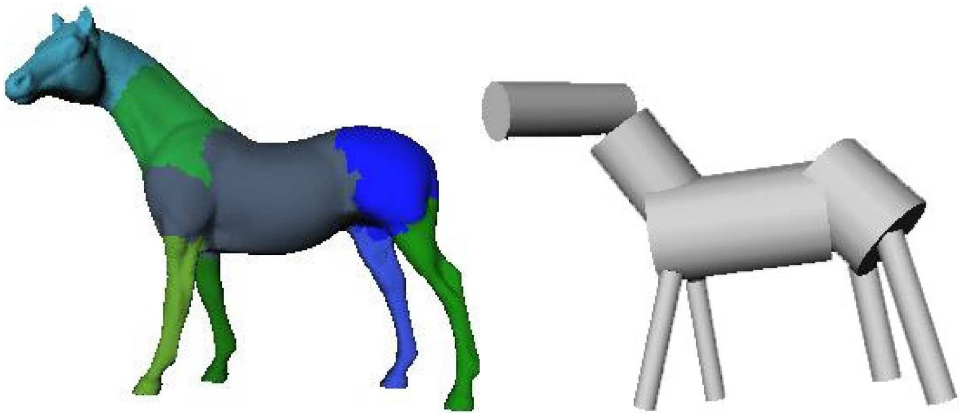


Figure 2.5 – *Horse segmentation based on fitting segments with cylinders obtained by the algorithm from Attene* et al. *(AFS06).*

Hierarchical clustering allows to simplify the segments, computed for a given model, thanks to its hierarchy. Consequently, this kind of algorithm can be used as a post processing step for region growing and watershed algorithms which are suffering from over-segmentation. However, the main difficulty in this kind of algorithm is the definition of a relevant merging criterion that allows to produce a significant segmentation.

### 2.3.4 Iterative clustering

Finding an optimal segmentation can be formulated as an iterative research process of the best partitioning of $k$ segments, where $k$ is the number of segments which is defined a priori (see algorithm 4). This process corresponds to solve the $k$-means clustering problem associated to the Lloyd's quantification algorithm (Llo82). The iterative process begins with $k$ clusters, each one having a representative centroid, and adds each mesh element (vertex for instance) to the closest cluster. To this end, a distance between the vertex and each class centroid is measured based on a given criterion (SSCO08, KJS07, LZHM06, YLW06, JKS05, SS05, WK05, PC04, CSAD04, KT03, STK02). Once all vertices are assigned to the different clusters, the centroids of these clusters are updated. The iterative process is repeated until the centroids stop changing.

**Algorithm 4**: Iterative clustering

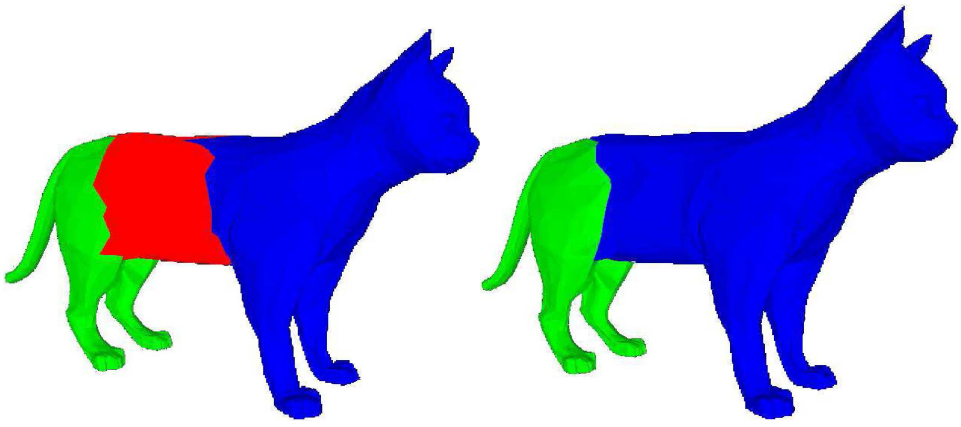
1: *Initialize k clusters*
2: **repeat**
3: **for** *each element (vertex or facet) of the mesh* **do**
4: *assign it to the closest cluster*
5: **end for**
6: *Update the centroids of clusters*
7: **until** *the centroids of clusters stop changing*

Katz and Tal (KT03) proposed an iterative segmentation algorithm using fuzzy clustering and cuts. The algorithm proceeds from coarse to fine thanks to a binary tree of which each node corresponds to a segment. For each node of the tree (initially only the root node which corresponds to the whole mesh), the algorithm computes a $k$-decomposition through the following steps:

- Compute a geodesic and angular distances for each pair of adjacent facets of the mesh.
- Compute an initial $k$-decomposition and assign each facet the probability of belonging to each segment.

- Compute a fuzzy decomposition by refining the probability values.
- Construct the exact boundaries between the segments, and thus transform the fuzzy decomposition into the final one.

The probability that a facet belongs to a segment depends on its distance with respect to all facets of this segment. In the binary case (2-decomposition) for instance, the two facets that are the farthest away from each other in term of geodesic and angular distance are selected to represent the initial two segments. Then, the probability of belonging to each segment is computed for the remaining facets of the mesh. This generates a fuzzy decomposition (see figure 2.6(a)) since for some facets of the mesh, the probability of belonging to each segment is identical. The fuzzy decomposition is then converted into the final one by computing the exact boundary (see figure 2.6(b)). This is done using a graph cut algorithm which traverses the fuzzy patch (red facets in figure 2.6(a)). The binary decomposition is recursively repeated on each segment until a given condition is no longer satisfied (for example a threshold on the distance between the representatives of segments).



(a) *Fuzzy decomposition (fuzzy facets are represented by red color)*

(b) *Final decomposition*

Figure 2.6 – *Binary decomposition, for the cat model, obtained by the algorithm from Katz and Tal (KT03).*

Lai *et al.* (LZHM06) proposed a clustering-based iterative segmentation algorithm dedicated for large models with high connectivity. They first generate a mesh hierarchy suitable for segmentation using a feature

sensitive remeshing algorithm. Then, they apply a clustering algorithm to segment the mesh. To increase the robustness of their method against geometric variations on the regions of the mesh, they introduced a metric which allows to compute efficiently the distance between facets for clustering. The metric definition includes geodesic distance, integral invariants related to averaged normal curvature (MHYS04), and statistical measures of these invariants characterizing local properties such as geometric texture.

Shapira *et al.* (SSCO08) have also proposed an iterative segmentation algorithm based on a volume shape function called the shape-diameter function (SDF). The SDF expresses a measure of the diameter of the object's volume in the neighborhood of each point on the surface. Globally the algorithm is composed of two steps. The first step uses a soft-clustering of the mesh elements (facets) to $k$ clusters based on their SDF values, and the second step finds the partitioning using $k$-way graph-cut to smooth the boundaries between segments.

The major drawback in iterative clustering algorithms is the convergence of the iterative process. To face this latter problem, a particular attention has to be paid regarding the way how to compute the representative centroids. An other drawback is the choice of the initial representatives of classes which may affect also the algorithm convergence and the final segmentation result. However, generally the existing algorithms in the state-of-the art give hand to users to choose the initial representatives.

### 2.3.5 Spectral segmentation

Spectral segmentation has seen an important amount of work (LZ07, LZ04, FBCM04, BH03, NJW01, ZHD*01, PF98) over the past decade and it is mainly based on spectral graph theory (Spi07, Chu97). We suppose that the input mesh is represented as a graph $G$ of which $A$ and $D$ are their respective adjacency and degree matrices. $A$ is a binary matrix such that $A_{ij} = 1$ if the $i^{th}$ and $j^{th}$ elements (vertices for instance) are adjacent or 0 otherwise. $D$ is a diagonal matrix where $D_{ii} = d_i$ is the degree (or

valence) of the $i^{th}$ vertex (it represents the number of neighbor vertices). The Laplacian $L$ of the graph $G$ corresponds to the matrix $L = D - A$.

Let $\{\xi_0, \xi_1, ..., \xi_{n-1}\}$ be the Laplacian eigenvectors associated with the eigenvalues $\{\lambda_0, \lambda_1, ..., \lambda_{n-1}\}$. The graph $G$ can be embedded into the space $R^k$ by using the first $k$ eigenvectors. Thus, a vertex $v_i$ of $G$ will be positioned at a point of coordinates $\{\xi_0(i), \xi_1(i), ..., \xi_k(i)\} \in R^k$. As raised by Gotsman (Goto3), the embedding allows to convert the combinatorial graph partitioning problem into a geometric space partitioning problem. To make clear his statement, he gave an example which is illustrated in figure 2.7. The figure shows on left side the graph $G$, on right side the spectral embedding and the resulting partitioning. The embedding is carried out as follow: find the direction $s$ of the largest spread of the vertices in $R^2$, and the 1 hyperplane in $R^1$ (straight line) normal to $s$ which partitions $R^2$ into two half-spaces. The partitioning of the graph $G$ (or graph cut) consists of the edges that straddle the hyperplane.

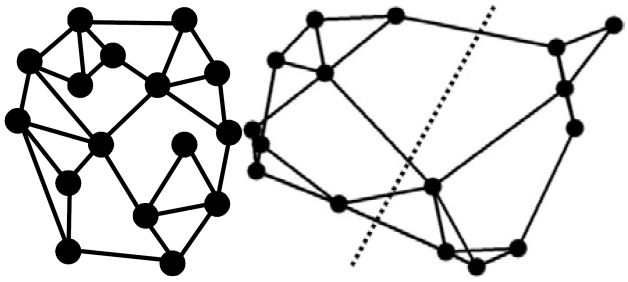


Figure 2.7 – *On left the graph G, on right the spectral embedding together with the resulting partitioning (*2 *parts separated by dashed line) (Goto3).*

Liu *et al.* (LZ04) have proposed a segmentation algorithm that makes use of spectral space partitioning. To this end, they defined a symmetric affinity matrix $W \in R^{n \times n}$ such that for each $i, j$, $0 \leq W_{ij} \leq 1$ encodes the probability that two facets $i$ and $j$ can be grouped in the same segment. The defined matrix is inspired from the one proposed by Katz and Tal (KT03) which is based on geodesic and angular distances. This latter one allows to avoid grouping facets which are separated by concave regions. The $W$ matrix can be seen as a graph adjacency matrix $A$. The spectral analysis of the $W$ matrix (using its first $k$ largest eigenvectors and

computing an embedding) creates a partitioning which induces a segmentation of the mesh.

Liu *et al.* (LZ07) have proposed an other algorithm which improves the results of the previous one (LZ04). Besides the spectral analysis, they made use of contour analysis. The algorithm consists of the following steps:

- Projecting the 3D-mesh into a 2D-plane.
- Extracting and analyzing the contour on the 2D-plane.
- Detecting the spectral cuts.

The 2D-projection is accomplished using the Laplacian $L$. To this end, the first 3 eigenvectors are computed (the first eigenvector is a constant so it is omitted). However, the projection using the original version of $L$ matrix does not allow to capture the geometric segmentation information since this latter matrix is based only on the connectivity of the mesh. To overcome this drawback, Liu *et al.* (LZ07) proposed a new Laplacian matrix $NL = D - W$, with $W$ an adjacency matrix based on principal curvatures of the vertices.

After 2D-projection, the contour is extracted by rendering the 2D-shape in black against a white background and tracing the boundary of the resulting binary image. If the contour is judged "segmentable" (it satisfies a convexity criterion), two sample points located on two different parts of the contour are computed. From these two points, which correspond to two facets on the mesh, a linear sequence of facets is derived using Nyström approximation (FBCM04). Then a 1D embedding is computed and used to perform a linear search over the sequence of facets. Each bisection of the sequence corresponds to a cut in the mesh, resulting in two parts. Figure 2.8 illustrates the different steps of the algorithm.

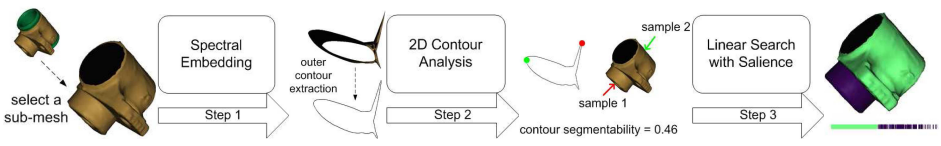



Figure 2.8 – *An iteration of the segmentation algorithm from Liu* et al. *(LZ07).*

Obviously, spectral analysis based segmentation methods depend strongly on the definition of the Laplacian matrix. This latter matrix has to be defined in such way to capture the geometric information. This allows to obtain a relevant segmentation.

### 2.3.6 Skeleton extraction based segmentation

Some segmentation algorithms make use of the shape skeleton to deduce the different segments (ATC*08, RT07, TVD07, XWS03, LTTH01). First of all, an approximate skeleton of the input mesh is computed using Reeb graph for instance (Tie08), then each critical node of the skeleton will correspond to a segment.

In the segmentation algorithm proposed by Tierny *et al.* (TVD07), the skeleton is used to delimit the object core and to identify the junction surfaces. Thus, the resulting segmentation is a coarse one which is refined following a hierarchical schema based on the topology of the model. The authors developed their own algorithm that allows to compute an enhanced topological skeleton (see figure 2.16(a)). This latter algorithm seeks to follow both of topological and geometrical variations of mesh contours computed using a geodesic mapping function.

Once the skeleton is computed, each node will refer to a segment. This gives an over-segmentation, as shown in figure 2.16(b), which is refined by merging segments. To this end, the skeleton nodes are classified, according to their degrees, into three categories: extreme with 1-degree (green nodes in figure 2.16(a)), tubular with 2-degree (blue nodes in figure 2.16(a)), and junction with at least 3-degree (red nodes in figure 2.16(a)). Thus the object core is defined by merging the segments which correspond to junction nodes. Then, each component adjacent to the object core, undergoes a merging to obtain a hierarchical segmentation (see figure 2.16(c,d)).

Similarly to the work cited above, Au *et al.* (ATC*08) proposed to use a skeleton for mesh segmentation. The main difference between the two segmentation algorithms relies on the skeleton computation. The one proposed by Au *et al.* (ATC*08) is based on mesh contraction. To this end,

(a) *Topological skeleton* (b) *Over-segmentation* (c) *Fine segmentation* (d) *Coarse segmentation*

Figure 2.9 – *Skeleton extraction based segmentation result, of the hand model, obtained by the algorithm from Tierny* et al. *(TVD07).*

a Laplacian smoothing is applied on the mesh to iteratively contract its geometry. The contraction allows to remove details from the input mesh, and leads to a zero-volume mesh which is converted to a 1D-curve skeleton. The conversion is carried out by removing all the collapsed facets from the zero-volume mesh. To remove collapsed facets a sequence of edge-collapse operations is applied using a coast function.

The quality of segmentation produced by this kind of algorithms depends strongly on the computed skeleton. This latter one has to reflect both of topological and geometrical variations of the shape in order to reveal the different object parts (or segments). Computing a skeleton with such property is not an obvious task, especially on noisy surfaces.

### 2.3.7 Interactive methods

This kind of methods requires the interaction of the user (FLL11, WPP*07, JLCW06, LLS*04). In the method proposed by Wu *et al.* (WPP*07), the user draws, on the 3D-mesh, a set of sketches which correspond to future segments (see figure 2.10). Each set of vertices traversed by a sketch is assigned a unique label. Then a merging algorithm is applied to generate the final segments. To this end, a queue is filled, initially with the unlabeled

vertices which are directly connected to labeled ones (direct neighbors). During the insertion, an angular distance is computed between each unlabeled vertex and its direct labeled neighbors. An iterative process is then applied in which the queue vertex that has the minimum distance is merged with the closest segment (set of labeled vertices) and is replaced in the queue by its direct unlabeled neighbors. The process stops when the queue is empty.

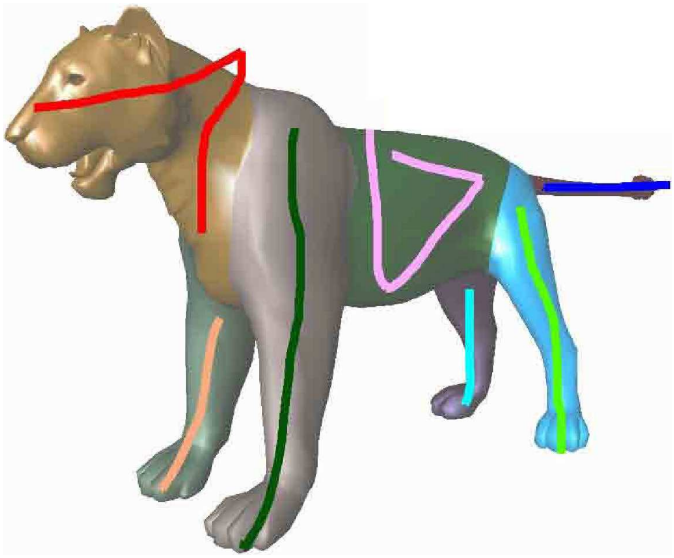


Figure 2.10 – *Interactive segmentation result based on drawing sketches (WPP*07).*

Generally, the main difference between the proposed methods relies on the definition of the criterion (angular distance in (WPP*07), Gaussian mixture and shape diameter function in (FLL11)) that decides how to merge the mesh elements (vertices for instance). The drawback of this kind of methods is the user interaction amount induced by the manual specification of sketches.

### 2.3.8 Learning segmentation

Lastly mesh segmentation has seen the use of advanced techniques based on learning thanks to the recent creation of ground-truth segmentation databases (CGF09). These databases have given to the computer graphics community the opportunity to quantitatively analyze and learn mesh segmentation. A very recent work based on learning approach has been proposed by Kalogerakis *et al.* (KHS10). It allows to simultaneously seg-

ment and label the input mesh, and is expressed as an optimization problem. The problem consists in optimizing a Conditional Random Field (CRF) of which an objective function is learned, using a set of geometric features, from a collection of labeled training meshes. The algorithm has demonstrated its efficiency through the improvement of the results over the state-of-the-art of mesh segmentation. According to our knowledge, only this latter work has been proposed that involves learning for 3D-mesh segmentation. In chapter 4, we present our own segmentation technique based on learning.

### 2.3.9 Other segmentation techniques

Some other segmentation techniques exist in the literature, that cannot be classified into one of the previously described classes. In this section we present some of them.

**Statistical methods.** Lai *et al.* (LHMR08) proposed an algorithm based on such approach. Their algorithm is inspired from random walk methods used for image segmentation. The algorithm is carried out through three steps. Firstly, seed facets are selected either manually or automatically and assigned each one a label. These facets represent initial elements used to compute the final segments. Secondly, a probability is associated with each non-seed facet. It corresponds to the likeliness that a random walk moves across a facet to another one. Thirdly, each non-seed facet $f_i$ is assigned the same label as the one of a seed facet $f_s$, if a random walk starting from $f_i$ has the highest probability to reach $f_s$. Golovinskiy and Funkhouser (GF08) proposed also a statistical segmentation algorithm. It consists of applying, on the input mesh, different segmentation algorithms with different settings for several times. Then a partition function is defined on each edge. This function measures the frequency that an edge belongs to a segment boundary in the mesh regarding the set of generated segmentations. A segmentation is then computed based on this partition function.

**Symmetry computation based methods.** Objects surrounding us (scanned objects and human-made objects) present a certain degree of symmetry, with respect to their shape, which is more or less important. Consequently, some existing algorithms in the literature (SKS06, PSG*06, MGP06) seek to exploit such information (symmetry) for segmenting the object into symmetric parts. The algorithm proposed by Simari *et al.* (SKS06) provides a hierarchical segmentation thanks to a recursive process that computes a set of symmetry planes. More precisely, at a given iteration a part of the mesh (initially the whole input mesh) is divided into two half parts that are separated by the detected symmetry plane. The same process is repeated on the resulting local parts until there is no longer symmetry plane detected. To compute a symmetry plane, the authors use an iterative re-weighted least squares (IRLS) algorithm. Basically, the main difference in this family of algorithms is the method used to compute the symmetry plane. For instance, in the segmentation algorithm proposed by Podolak *et al.* (PSG*06) they use Monte Carlo integration to compute this plane, while in the algorithm proposed by Mitra *et al.* (MGP06) they use a method based on matching local shape descriptors.

**Feature point extraction.** Katz *et al.* (KLT05) proposed a segmentation algorithm based on such approach. First, the mesh is transformed into a canonical mesh, using multidimensional scaling, where the Euclidean distance between its vertices is similar to geodesic distance. This latter transformation leads to a pose-invariant representation. Then a set of feature points that correspond to the prominent points on the canonical mesh are extracted, and used to guide the segmentation. Finally the core component of the original mesh is extracted by applying a spherical mirroring operation on the canonical mesh, and the remaining segments (each segment is represented by at least one feature point) are determined by "removing" the core component.

## 2.4 Discussion on existing segmentation techniques

Mesh segmentation is a wide research area. We conducted in the previous section a study of the different segmentation algorithms proposed in the literature and tried to classify them according to their characteristics. Table 2.1, on the next page, summarizes the different segmentation techniques while providing their principles together with their advantages and drawbacks. However, it is clear that comparing the different families of segmentation algorithms is not obvious since each one of them seek to produce a specific segmentation which depends on the end-application. Moreover, even algorithms issued from the same family do not necessarily produce the same type of segmentation. In what follow we describe the different types of segmentation and give some application examples.

| **Segmentation technique** | **Principle** | **Advantages** | **Drawbacks** |
|---|---|---|---|
| Region growing | Select one seed element each time then apply growing | Simple to implement<br>Not time consuming | Dependence on seed elements<br>Over-segmentation |
| Watershed | Select multiple seed elements (local minima) then apply growing | Simple to implement<br>Not time consuming | Dependence on seed elements<br>Over-segmentation |
| Hierarchical | Merging mesh elements based on a criterion | Coarse to fine segmentation | Difficult to define a relevant merging criterion |
| Iterative clustering | Solve *k*-means problem | Coarse to fine segmentation | Convergence of iterative process<br>Dependence on initial representatives |
| Spectral partitioning | Spectral embedding of the mesh | Converting graph partitioning problem into geometric space partitioning problem | Dependence on the definition of Laplacian matrix |
| Skeleton extraction | Use critical nodes of the skeleton to define segments | Follow topological & geometrical variations<br>Coarse to fine segmentation | Time consuming<br>Dependence on computed skeleton |
| Interactive methods | Draw sketches by user which will correspond to segments | Semantic segmentation guided by user | Not fully automatic |
| Learning | Use ground-truth databases to learn segmentation | Semantic segmentations similar to those produced by humans | Applied only on the same category of learned models |
| Other techniques | Random walk, Symmetry reflection, etc. | Coarse to fine segmentation | Sensitivity to noise and pose-variation |

Table 2.1 – *Summary of 3D-mesh segmentation techniques.*

## 2.5 Mesh segmentation types and their applications

According to recent states-of-the-art (Shao8, APP*07, AKM*06), mesh segmentation techniques can be classified into two categories: surface-type (or *geometric*) methods and part-type (or *semantic*) methods (see figure 2.11). In the first case, the algorithms are based on low level geometric information (e.g. curvature (LDB05b)) in order to define segments (i.e. regions) with respect to geometric homogeneity, while in the latter case, the algorithms aim at distinguishing segments that correspond to relevant features of the shape, such as in the recent work proposed by Kalogerakis *et al.* (KHS10).

Figure 2.11 – *Example of surface-type segmentation on the left, and part-type segmentation on the right (Shao8).*

Both categories of 3D-mesh segmentation techniques are a fundamental process in many applications. In what follow we summarize some of them.

### 2.5.1 Applications based on surface-type segmentation

We list here some applications that use surface-based segmentation, and give, as examples, some existing works in the literature.

**Mesh compression.** 3D-compression techniques are used for reducing the delays in transmitting triangle meshes over the Internet, and to make their storage feasible. Interested readers are referred to the survey proposed by Alliez and Gostman (AG05). Among the different compression methods some of them are based on spectral approach (KG00). This is achieved by projecting the mesh geometry onto an orthonormal basis derived from the mesh topology. To reduce complexity, the mesh is partitioned into a number of balanced sub-meshes with minimal interaction, each of which are compressed independently. Lavoué *et al.* (LDB05a) have proposed a framework based on subdivision surface approximation for polygon mesh compression and coding. Their method involves the segmentation of the target 3D object into surface patches of which boundaries are extracted. They approximate then the surface patches, put them together, and encode the mesh information. Qin *et al.* (QXP*06) addressed the problem of photo-realistic rendering using a parallel architecture and proposed a mesh compression scheme called PRMC (Parallel Rendering based Mesh Compression). The segmentation allows them to obtain a set of sub-meshes which are compressed and sent to multiple rendering servers in order to compute the different parts of the scene.

**Texture mapping.** Texture mapping allows to glue an image to a 3D object (polygon mesh) in order to enrich its photo-realistic rendering and to reduce its complexity in term of size (see figure 2.12). Sander *et al.* (SSGH01) have proposed a texture mapping method for progressive meshes. Given an arbitrary mesh, they construct a progressive mesh (PM) such that all meshes in the PM sequence share a common texture parametrization. The method begins by partitioning the mesh into charts (surface patches) using planarity and compactness heuristics. Next, it simplifies the mesh while respecting the chart boundaries. Finally, the charts are packed into a texture atlas. Sander *et al.* (SWS*03) have proposed an algorithm which partitions a mesh into rectangular charts while preserving a one-to-one texel correspondence across chart boundaries. This mapping permits any computation on the mesh surface which is typically

carried out on a regular grid, and prevents seams by ensuring resolution continuity along the boundary.

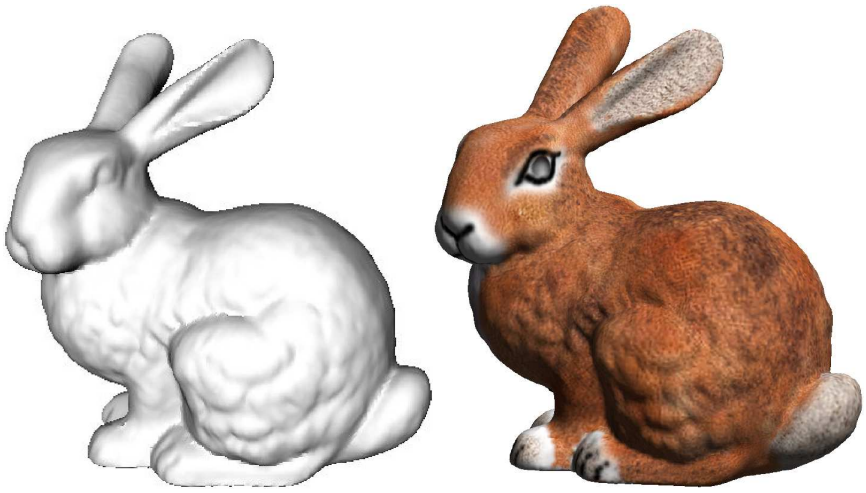


Figure 2.12 – *Example of texture mapping for 3D-mesh (LPRM02).*

**Watermarking.** 3D-mesh watermarking aims to preserve the author's copyright of 3D models. For example, a message which allows to identify the object owner can be dissimulated by slightly modifying the position of certain vertices. Watermarking methods that use the global information of the mesh fail to face attacks which consist to cut the mesh in order to keep a part of it. To address this drawback, Randão *et al.* (RaAMC07) have proposed to firstly segment the input mesh into a set of parts and then to watermark each one of them. In Wang *et al.* (WLDB11) the mesh is normalized to a canonical and robust spatial pose by using its global volume moments. Then, the normalized mesh is segmented into patches and the watermark is embedded into some selected candidate patches.

### 2.5.2 Applications based on part-type segmentation

By partitioning an object into meaningful parts (part-based segmentation), many analysis and modeling tasks could be enhanced. For instance, partial match queries can be formulated, annotation of parts in objects can be utilized, modeling-by-parts applications could be supported, object skeleton extraction is facilitated, etc.

**Partial matching.** Partial matching systems aim at helping human users browsing large collections of 3D shapes in an interactive and intuitive way. These systems are expected to retrieve objects that have similar subparts even if they visually differ globally (see figure 2.13). Mademlis *et al.* (MDA*08) have proposed a framework used for partial and global 3D-object retrieval. First, the object is decomposed into meaningful parts and an attributed graph is constructed based on the connectivity of the parts. Then a 3D shape descriptor is computed and associated to the corresponding graph nodes. Finally, the matching process, based on attributed graph matching, is performed. Shapira *et al.* (SSS*10) have addressed the problem of finding analogies between parts of 3D objects. In their approach, all objects are hierarchically segmented into meaningful parts, and each part is given a local signature. To find corresponding parts in other objects they use a context enhanced part-in-whole matching based on bipartite graph matching algorithm.

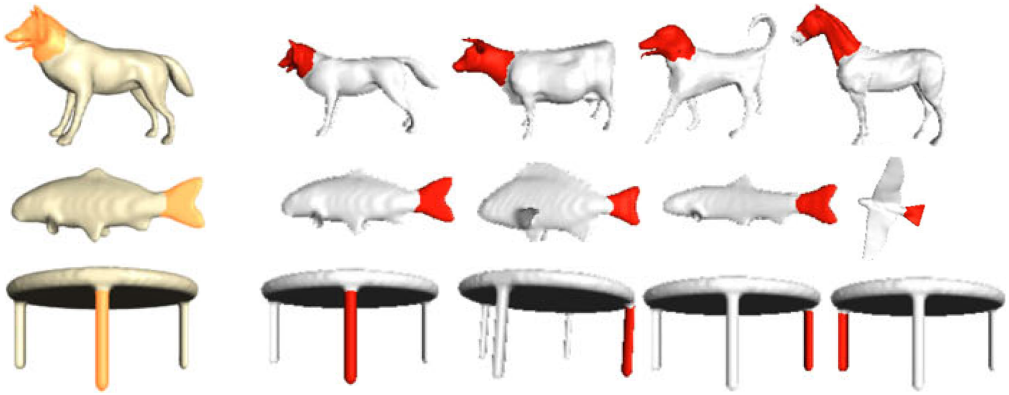


Figure 2.13 – *Example of partial matching. On the left the query part and on the right the results (SSS*10).*

**Semantic annotation.** Semantic annotation of a 3D object consists in annotating this object in terms of its meaningful subparts, their attributes and their relations (see figure 2.14). The possibility to semantically annotate shape parts may have a relevant impact in several domains such as the creation of avatars in emerging MMORPGs (Massive Multiplayer Online Role-Playing Games) and in on-line virtual worlds. Attene *et al.* (ARMS09) have proposed a system called the "ShapeAnnotator" through which an input 3D-object is first segmented into meaningful parts, then each de-

tected part is annotated through concepts expressed by an ontology. Each part is connected to an instance that can be stored in a knowledge base. Through an intuitive interface, users create such instances by simply selecting proper classes in the ontology.

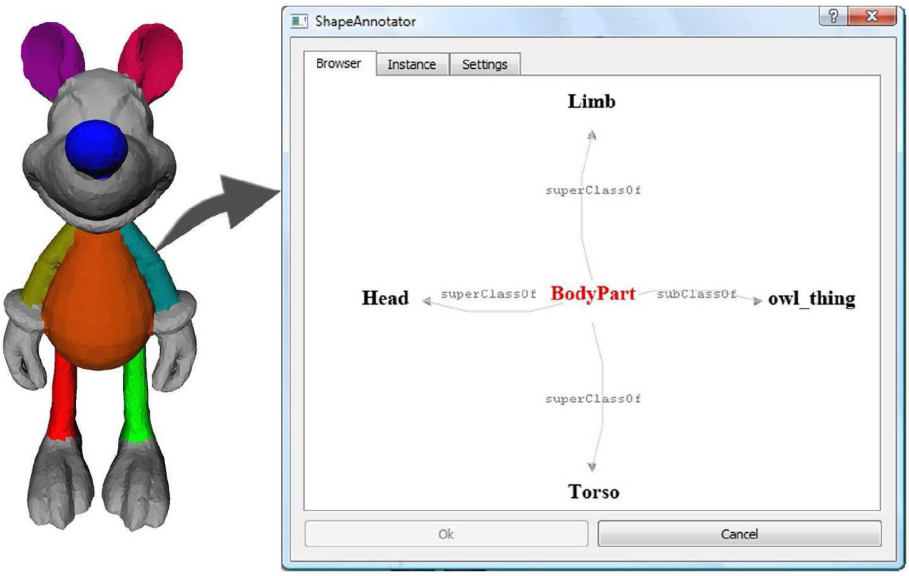



Figure 2.14 – *Example of semantic annotation based on ontology (ARMS09).*

**Modeling by example.** Modeling by example enables a novice user to search into a large database of 3D meshes to find parts of interest, select the desired parts of several meshes, and compose them together in different ways to form new objects (see figure 2.15). Funkhouser *et al.* (FKS*04) have proposed a data-driven synthesis approach for such application. The models of a database are segmented into meaningful parts, and using a shape-based search algorithm, 3D-models with parts matching the input query are found. The user can then perform editing operations in which parts are cut out from the retrieved models and composited into a new model.

**Skeleton extraction.** A skeleton is an object that represents the shape of its target object with a lower dimension (see figure 2.16). Because a skeleton is simpler than the original object, many operations, e.g., shape deformation, can be performed more efficiently on the skeleton than on the full object. In the segmentation techniques section2.3, we have seen

Figure 2.15 – *Modeling a new chair composed from the circled parts of the others (FKS*04).*

that some segmentation algorithms make use of the shape skeleton to deduce the different segments. In what follow we describe the reverse application in which the segmentation is used to deduce the shape skeleton. Lien *et al.* (LKA06) have proposed an iterative approach that simultaneously generates a hierarchical shape decomposition (or segmentation) and a corresponding set of multi-resolution skeletons. The skeleton of a model is extracted from the components of its decomposition. Mortara *et al.* (MPS06) have introduced a framework for the automatic extraction and annotation of anthropometric features from human body models. Based on a meaningful segmentation, a semantic model is built as an annotated shape-graph where each node corresponds to a relevant feature represented by its centerline skeleton and a set of cross-sections.

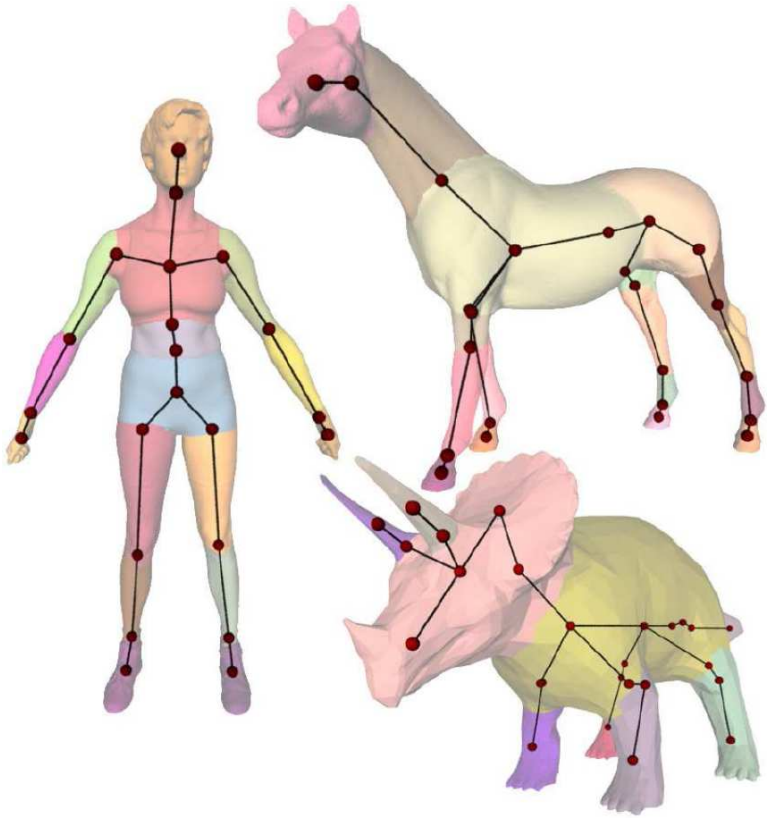


Figure 2.16 – *Extracted skeletons of some 3D-models (LKA06).*

## 2.6 2D-image *VS.* 3D-mesh segmentation evaluation

As we have seen in section 2.3, many 3D-mesh segmentation methods have been developed over the past decade which makes it hard to compare their different results, or even different tunings of the same method. Indeed, having tools which allow to automatically evaluate the quality of the segmentation in an application-independent way is important:

- To select among different segmentation algorithms the best one without any a priori knowledge;
- To rank new segmentation algorithms or existing ones based on a solid experimental and comparative study;
- To analyze the drawbacks of proposed algorithms and thus improve their quality.

However, providing an objective and quantitative evaluation method for segmentation quality is not obvious. On one hand, each person has

his/her own criteria for a good segmentation. On the other hand, the segmentation is usually application-dependent. However, in most applications the difference between a good and bad segmentation is clear. Consequently, it is mandatory to design automatic tools to capture such difference.

In what follow, we provide a review of the state-of-the-art of 2D-image and 3D-mesh segmentation evaluation. Indeed, the most significant works for the 3D-mesh segmentation evaluation (CGF09) (including our work which is presented in the next chapter) are based on the same methodology as that proposed in the 2D-image domain (MFTM01).

### 2.6.1 2D-image segmentation evaluation

Several advanced works exist for the quality assessment of *2D-image* segmentation. Zhang et al. (ZFG08) offer a study on the different methods proposed for this task. According to them, the different methods can be classified into five groups:

**Analytical methods.** They directly treat the segmentation algorithms themselves by taking into account principles, requirements, utilities, complexity, etc. of algorithms. Using analytical methods to evaluate segmentation algorithm avoids a concrete implementation of the algorithms. However, the real quality of these algorithms cannot be obtained by a simple analytical study.

**Subjective methods.** They evaluate the segmentation algorithms in a subjective way in which the segmentation results are judged by a human operator. Therefore, the evaluation scores can vary significantly from one human evaluator to another since they do not have necessarily the same standards for assessing the quality of a segmentation. Furthermore, the results can depend on the order in which the human operator observes them. To minimize bias, such a method requires a large set of objects and a large group of humans. Unfortunately, this kind of methods cannot be integrated in an automatic system.

**System level evaluation methods.** This kind of methods indicates if the characteristics of the results obtained by a segmentation algorithm are suited for the over-all system which uses this segmentation algorithm. However, this evaluation method is indirect. If the process which follows the segmentation generates better results, it does not necessarily mean that the segmentation results were superior, and vice-versa.

**Empirical goodness or unsupervised methods.** They evaluate the performance of the algorithms by judging the quality of the segmented images themselves. To achieve this task, a set of quality criteria has to be defined. These criteria are established according to human intuition about what conditions should be satisfied by an ideal segmentation. However it seems difficult to establish quantitatively the quality of a segmentation only by using such a priori criteria.

**Empirical discrepancy or supervised methods.** A set of reference images presenting the ideal segmentation is first of all built. This set of images, which can be manually segmented by experts of the domain, constitutes a ground-truth. The purpose is to measure the discrepancy between these reference segmentations and that obtained by an algorithm to evaluate. So, these methods try to determine how far a segmented image obtained by an algorithm is from one or several manually segmented images. A large discrepancy involves a large segmentation error and thus this indicates a low performance of the considered segmentation algorithm.

Although some unsupervised methods exist (CEL*04, CERL06), the empirical discrepancy methods are the most popular for 2D-image segmentation evaluation (MFTM01, UPH07). Indeed they seem to be the most suited for a quantitative evaluation as the measures of quality can be numerically computed, and for an objective evaluation thanks to the ground-truth.

Martin et al. (MFTM01) have proposed such a method to evaluate image segmentation algorithms. They built a public corpus containing ground-truth segmentations produced by human volunteers for images of

a wide variety of natural scenes. They also defined a measure of segmentation similarity based on the computation of refinement error of a pixel between two segments (i.e. regions) containing this pixel.

### 2.6.2 3D-mesh segmentation evaluation

In the 3D-domain, there exist some works proposing the segmentation quality assessment in a specific context. In the MRI (Magnetic Resonance Imaging) field for example, Gerig et al. (GJC01) propose a tool that quantifies the segmentation quality of 3D-images (volumetric images) including different shape distance metrics such as maximum Hausdorff distance, and mean/median absolute distance between object surfaces. For texture mapping, Sander et al. (SSGH01) introduce a metric based on the texture stretch induced by the parametrization of the segmented regions and allowing the evaluation of the segmentation quality.

More recently, Attene et al. (AKM*06) have proposed some criteria like the aspect of the boundaries (smoothness, length), the hierarchical / multi-scale properties, the robustness, the complexity and the number of parameters. However these criteria rather judge some technical points than the *real* quality of the techniques themselves, they rather fall in the empirical goodness methods. As raised by the authors, the main problem is that the objective quality of a segmentation of a given model is quite difficult to define, since it depends on the viewer's point of view and knowledge.

Berretti et al. (BDBP09) have presented some experimental results which are based on ground-truth segmentations to evaluate and validate their own segmentation algorithm. However, the ground-truth segmentations are not available on-line and according to the authors they contain very simple 3D-models (surfaces of revolution, vases, etc.).

One of the goals of this thesis is the evaluation of the quality of mesh segmentation algorithms (empirical discrepancy). In the next chapter, we present our evaluation method which is based on a benchmark. The benchmark includes a ground-truth corpus of human segmented 3D-models and a set of similarity metrics. The evaluation of a segmentation

algorithm is realized by quantifying the consistency between the reference segmentations of the ground-truth corpus and those obtained by this algorithm on the same models using the set of similarity metrics. A concurrent work that addresses the same task and is based on the same methodology (a benchmark including a ground-truth corpus and a set of similarity metrics) has been simultaneously proposed by Chen *et al.* (CGF09). The difference between the two benchmarks lies in the protocol followed to create the ground-truth corpus and the choice of the similarity metrics (more details are provided in the next chapter).

## 2.7 Conclusion

In this chapter we reviewed the state-of-the-art of 3D-mesh segmentation algorithms and the evaluation methods proposed to assess their quality.

In the first part, that concerns 3D-mesh segmentation algorithms, we have seen that plenty of algorithms have been developed over the past decade. We decided to classify them according to their characteristics and how much user intervention they need, so we defined 8 groups which are the most popular in the literature, namely region growing, watershed, hierarchical clustering, iterative clustering, spectral segmentation, skeleton extraction based segmentation, interactive methods and learning methods. We added an other group that we called "other techniques" in which we pointed out some non common algorithms in the literature (random walk, symmetry detection, etc.). We briefly discussed about the advantages and drawbacks of each group. Globally, we have seen that except interactive and learning methods, the others require geometrical and/or topological descriptors that provide relevant information about the semantic of the shape and its geometry. Defining such descriptors is not an obvious task. Interactive algorithms aim at improving the segmentation quality with the help of the user. Unfortunately, this kind of algorithms cannot be integrated in a fully automatic system since they require user interactions. We have also pointed out the appearance of a new algorithm category in the literature that involves learning for 3D-mesh segmentation. This kind of algorithms seeks to fix all the previous drawbacks by exploiting the

information provided by ground-truth segmentation databases. Finally, we have shown the importance of 3D-mesh segmentation in providing a structural description of 3D-meshes and listed out several applications that use this latter process.

In the second part we have raised the need of automatic tools to evaluate and compare segmentation algorithms. Although development of mesh segmentation algorithms has drawn extensive and consistent attention, relatively little research has been done on segmentation evaluation. We have shown that evaluation tools are necessary since they allow for instance to rank the different algorithms and to select the best one without any prior knowledge. We have pointed out the fact that existing works for the quality assessment of 3D-mesh segmentation are inspired from 2D-image field. Consequently, we have included the classification of the 2D-image segmentation evaluation methods which is available in the literature. Then we have reviewed existing methods in the 3D field. We have seen that these methods fall either in empirical goodness methods or empirical discrepancy methods.

In the following chapter, we present in detail the contributions of this thesis regarding to the evaluation of 3D-mesh segmentation algorithms.

# 3 A benchmark for 3D-mesh segmentation evaluation

Contents

This chapter presents a benchmark addressing the quality assessment problem of mesh segmentation. The benchmark includes a ground-truth segmentation corpus and a new reliable similarity metric named the 3D Normalized Probabilistic Rand Index (3D-NPRI). The chapter also presents an extensive experimental comparison of existing similarity metrics in the literature and the new one. The new metric is shown to outperform the others in terms of properties and discriminative power. The experimental comparison includes a subjective experiment with human observers. Finally the 3D-NPRI is applied to evaluate six recent segmentation algorithms using our corpus and the Chen's *et al.* (CGF09) corpus.

## 3.1 Motivation

Our motivation to propose a benchmark for 3D-mesh segmentation evaluation is justified by the fact that before starting this thesis, no automatic tool had been proposed to evaluate segmentation algorithms (particularly part-type ones) in a general purpose context. In the previous chapter, we showed that the evaluation is an important task not only for researchers to compare a new algorithm to those already existing, but also for users so as to choose an algorithm and fix its parameters depending on the problem to solve.

Although for surface-type segmentation algorithms some evaluation tools exist depending on the end application as texture mapping (SSGH01) or medical imaging (GJC01), the question of the evaluation of part-type segmentation algorithms remains critical. Whereas compression or recognition algorithms are quite easy to evaluate thanks to compression ratio or misclassification probability, this task is far more difficult to handle for semantic segmentation. Typically researchers exhibit some results for several models and just point out why their results look "good". Moreover many authors argue that a segmentation quality is theoretically impossible to evaluate objectively because it depends only on the desired application. Indeed the desired task is of course of importance. For instance a structural recognition application does not need the same segmentation than a mesh texture mapping task. However for many applications (see previous chapter) researchers aim to obtain semantic decomposition (or part-type segmentation). Thus our objective is rather to focus on the evaluation of such semantic (i.e. part based) algorithms.

To this end, we propose a benchmark for segmentation evaluation which includes a ground-truth corpus and a set of similarity metrics. The quality of segmentations obtained by automatic algorithms is then evaluated in a quantitative way thanks to the metrics, and on an objective basis thanks to the ground-truth corpus (see figure 3.1). More specifically, the evaluation is carried out by measuring the similarity between the reference segmentations from the corpus and that obtained by the automatic algorithm (on the same models). The corpus is composed of a set of 3D-models

grouped in different classes and associated with several manual segmentations produced by human observers. Of course, the ground-truth can depend also on the application.

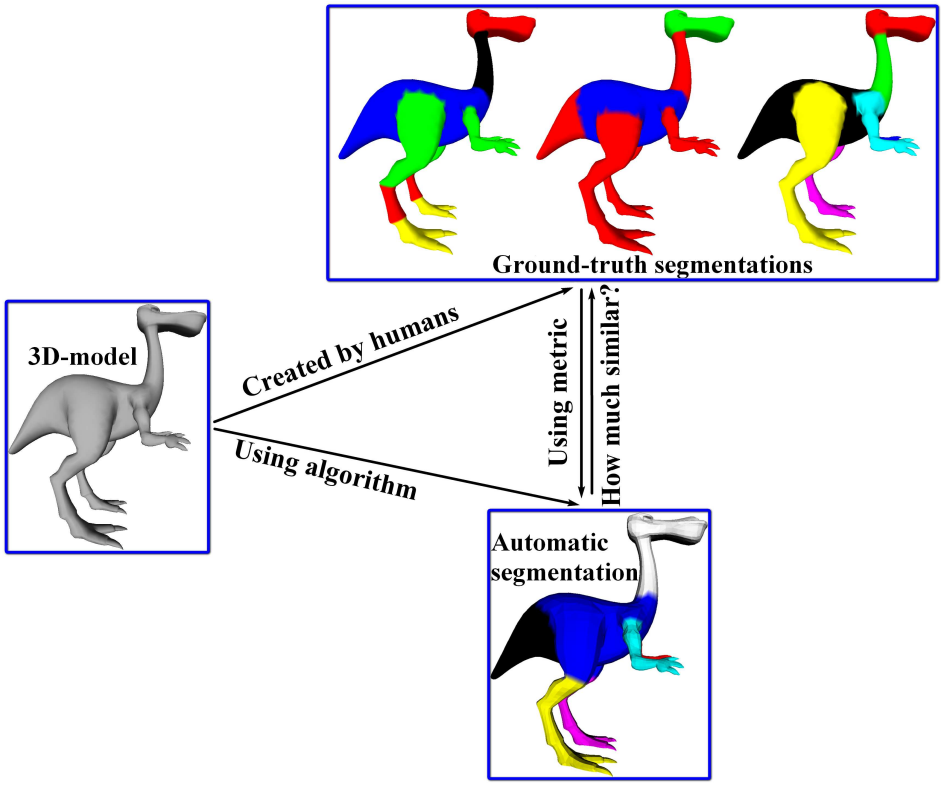



Figure 3.1 – *Overview of benchmark-based mesh segmentation evaluation methods.*

As a service to the scientific community, we have made the benchmark publicly available under the form of an on-line automatic tool[1] (see figure 3.2).

In order to quantify the efficiency of our benchmark (metrics and ground-truths), we include in this chapter a detailed experimental comparison between the different similarity metrics while studying their properties. Moreover, we propose a subjective experiment in which a set of volunteers rate a set of automatic segmentations including ground-truths of our benchmark, and then these rates are correlated with the values produced by the metrics for the same segmentations. The subjective experiment allows, from one side to check whether the ground-truths have the best rates given by the volunteers, with comparison to the remaining

[1] http://www-rech.telecom-lille1.eu/3dsegbenchmark/


Figure 3.2 – *A snapshot of our automatic tool for mesh segmentation evaluation.*

segmentations, and thus validate the quality of our ground-truths. From the other side, it allows to find the metric of which the values are the best correlated with the rates given by the volunteers, and thus validate the discriminative power of this latter metric.

## 3.2 Ground-truth corpus

We detail now how we created our ground-truth corpus while emphasizing the dataset construction, the tool used for manual segmentation and the segmentation protocol followed.

### 3.2.1 Dataset construction

The corpus contains twenty-eight 3D-models (as triangle meshes) grouped in five classes, namely animal, furniture, hand, human and bust. Note that

it was crucial for our corpus, to present a high variety of models so as to be able to study properly the different segmentation algorithm's behaviors while avoiding to privilege certain algorithms over others. We have then conducted a large campaign of manual segmentation with human subjects. Each 3D-model of the corpus is associated with 4 manual segmentations which give a total of 112 ground-truth segmentations done by 36 volunteers. Figure 3.3 illustrates the models of the corpus with one manual segmentation per model. We have selected a small number of varied models with respect to a set of properties. All the selected models are manifold, connected, and do not have intersecting faces. Hence they are supported as an input by any segmentation algorithm. The models come from the GAMMA[2] database from INRIA, from the Princeton Shape Benchmark[3] (SMKF04), and from the AIM@SHAPE[4] repository, which are public 3D-model databases.

### 3.2.2 Tool for manual segmentation

In order to easily collect manual segmentations from a wide range of people, we have used the MeshLab[5] application; this software allows the processing of 3D-meshes, providing a set of tools for editing, filtering, inspecting, rendering and converting them. In particular it allows an explicit vertex-per-vertex segmentation of models using colors.

Indeed, a virtual brush allows a human observer to colorize each vertex of the mesh to segment. Each segment (a set of connected vertices) is then distinguished from others by its associated color.

Using this application, anyone can segment models without having any prior skills in computer graphics. Figure 3.4 illustrates the coloring process using the MeshLab application. Moreover to accelerate the coloring process (which could be fastidious for complex models) and to make it easier, we have developed a color propagation algorithm that allows the user to only indicate the different boundaries between the different

[2] http://www-roc.inria.fr/gamma/gamma.php
[3] http://shape.cs.princeton.edu/benchmark/
[4] http://shapes.aimatshape.net/
[5] http://meshlab.sourceforge.net/

Figure 3.3 – *Models of our corpus associated with one ground-truth.*

segments; the whole segments are then automatically filled by colors (see figure 3.5). Basically with this tool, between 5 and 15 minutes are necessary for an observer to segment a 3D-model depending on its complexity.

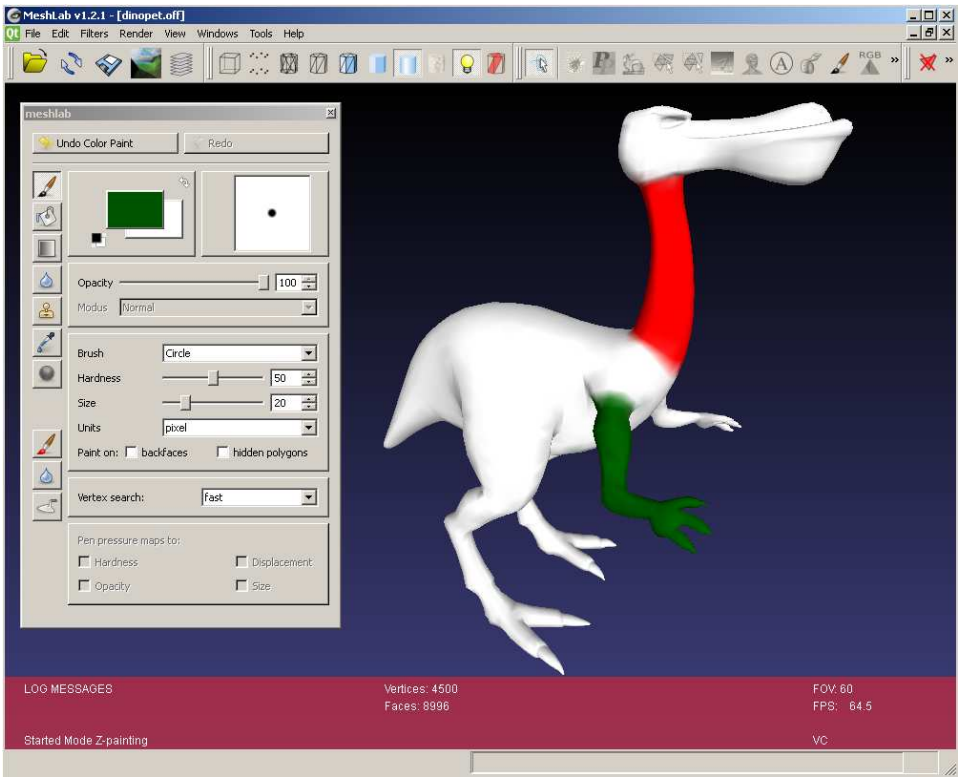



Figure 3.4 – *Vertex coloring process using MeshLab.*

### 3.2.3 Segmentation protocol

To obtain precise manual segmentations, we have assisted the 28 volunteers (staff members and PhD students from University of Lille and Insa-Lyon) in tracing the vertex-boundaries through the different models. Note that the volunteers have freely segmented the models and no condition was imposed on the manner with which they have segmented them. The models were randomly assigned to each volunteer with a bias towards models that had been already segmented several times.

Having more than one segmentation per mesh is very important since two observers do not necessarily share the same opinion on the segmentation of a model. This is due to the lack of rules that define how to decompose an object into sub-objects. Consequently, two observers may segment the same model differently for the following reasons:

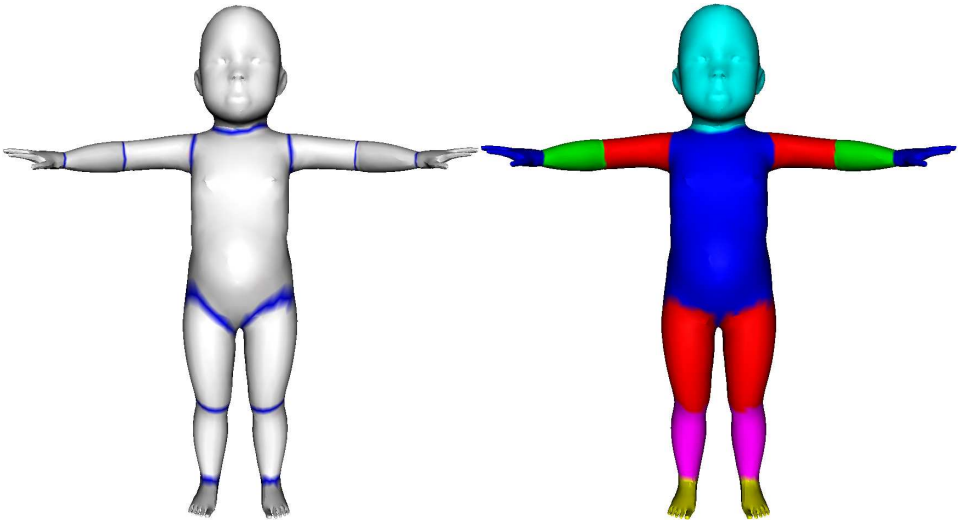


Figure 3.5 – *Automatic propagation of colors on the baby model. The user just need to color the boundaries of the regions that he wants to separate (left), our algorithm then automatically complete the coloring (right).*

- **Attention**. Observers may differently pay more attention to some parts of the object and may therefore over-segment these parts of interest, and under-segment the parts to which they did not pay attention.

- **Refinement**. Two observers may segment a given object identically, except that one observer may divide the object segments into smaller sub-segments than the other observer did. In other term, observers can segment at different granularities.

### 3.2.4 Consistency of ground-truth segmentations

We explained in the previous section that the segmentations produced by different humans for a given 3D-object are not necessarily identical. But are they consistent? The positive answer to this question is important, since it affects directly the utility of our benchmark. Our choice to use human-made segmentations as "ground truth" for our benchmark is justified by the following assumptions:

- Although people do not share the same attention and refinement degree toward an object and its segmentation, they tend to segment it in the same way in term of significance of its parts.

- Automatic algorithms, particularly part-type ones, seek to imitate human segmentations.

A direct manner to assert the first assumption is to visually analyze our corpus ground-truths. Figure 3.6 illustrates multiple manual segmentations done by different persons for some 3D-models of our corpus. One can notice that people select the same types of functional parts, however, as we raised in the previous section, the difference lies rather in the level of refinement and slightly on the position of the boundaries. For instance, we can see that the second and the third segmentations, from left to right, of the baby model are mutual refinements of each other regarding the head and legs. We can say that the humans are consistent in their segmentations.

This statement was also validated quantitatively using a subjective experiment described in section 3.5.

### 3.2.5 Our corpus *VS.* Princeton corpus

Chen *et al.* (CGF09) proposed another corpus that seems complementary to ours: they present more objects (380 3D-models of the Watertight Track of the 2007 SHREC Shape-based Retrieval Contest (GBP07)) when we selected a small representative set (it allows to rapidly evaluate a segmentation algorithm without running it on 380 objects). They chose to use the web application Amazon's Mechanical Turk[6] to collect the manual – i.e. ground-truth – segmentations without any supervision when we chose to supervise our volunteers to obtain more precise manual segmentations. Finally, their ground-truths present facet-based segmentations whereas ours contain vertex-based segmentations.

## 3.3 Mesh segmentation similarity metrics

In benchmark-based evaluation methods, the quality of the evaluation depends on the quality of the corpus but also on the quality of the segmentation similarity measure.

[6] http://www.mturk.com/

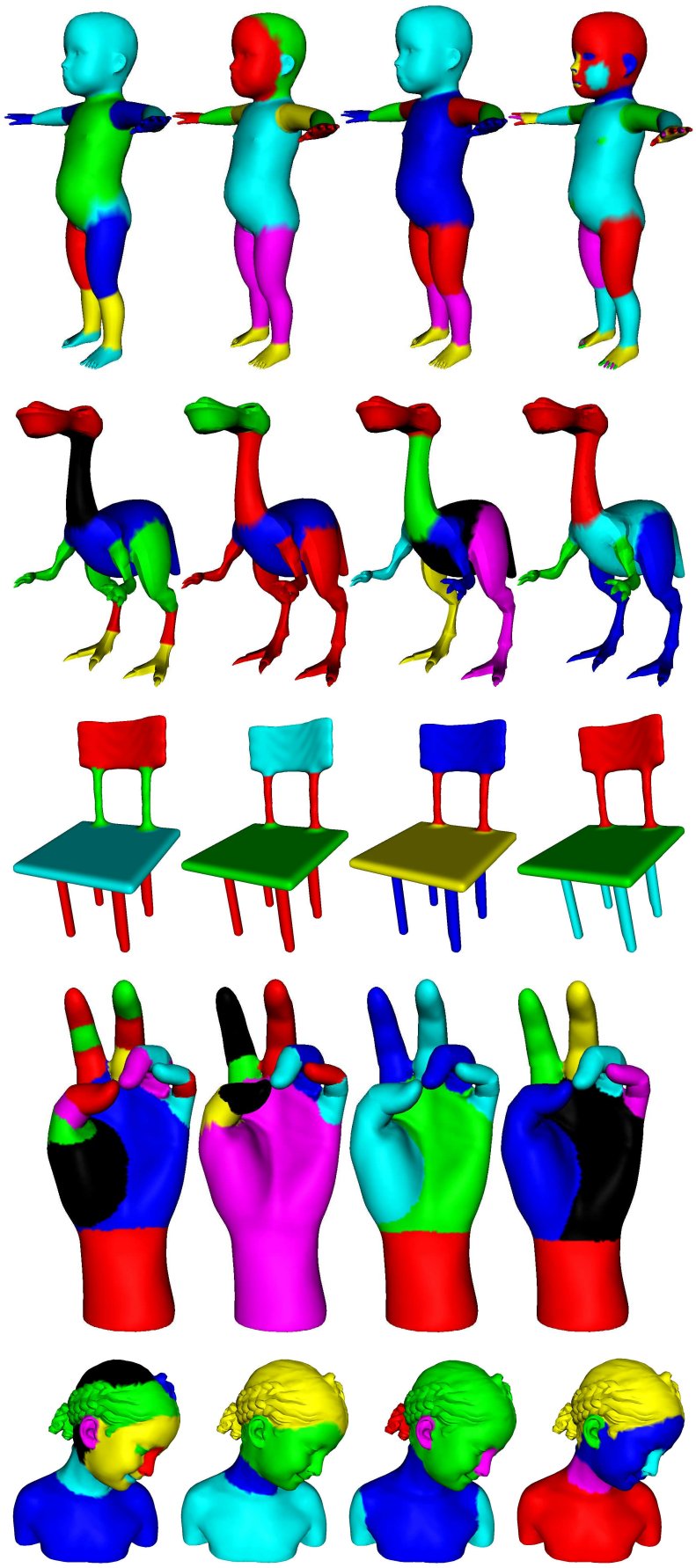


Figure 3.6 – *Examples of ground-truth segmentations from our corpus made by different persons.*

This leads to conclude that the choice of an accurate measure is critical in order to provide a relevant evaluation and to reflect the real quality of an automatic segmentation with comparison to a manual one. In this context, less efforts were investigated to propose a reliable measure of mesh segmentation similarity. Indeed, in the work proposed by Chen *et al.* (CGF09) more interest has been paid on the design of the ground-truth corpus and they presented rather simple metrics suffering from degeneracies and low discriminative power.

### 3.3.1 Properties of a reliable similarity metric

A reliable measure of mesh segmentation similarity has to possess the following set of properties:

**Non degenerative cases.** The score's measure must be proportional to the similarity degree between an automatic segmentation and the ground-truth segmentations of the same model. For example, an over-segmentation where each vertex (or facet) is represented by a segment must give a very low value of similarity, since no ground-truth segmentation can be represented in such a way.

**Tolerance to refinement.** The segmentation performed by some human observers can be coarse while the segmentation performed by others can be finer. However they basically remain consistent; the difference just lies in the level of refinement. Hence, a reliable segmentation measure has to accommodate and to be invariant to these segmentation granularity differences.

**Cardinality independence.** The measure must neither assume equal cardinality nor depend on this attribute. This means that two segmentations to be compared can have different numbers of segments and different sizes of segments.

**Tolerance to cut boundary imprecision.** Humans define the segment boundaries of 3D-objects in a subjective way. Indeed, it is possible that

two volunteers define the same segment on a model with a slight difference between boundaries, however, from a semantic point of view, the segments remain similar. Hence, a reliable measure has to accommodate this imprecision of cut boundaries.

**Multiple ground-truths.** The measure has to be able to compare one automatic segmentation with multiple ground-truths (reference segmentations) for a given model, otherwise, providing multiple ground-truths in a benchmark is useless. An alternative solution is to simply average the similarity scores obtained between an automatic segmentation and each manual segmentation (reference segmentation), however, this may bias the result and not really reflect how much an automatic segmentation agrees with the multiple ground-truths.

**Meaningful comparison.** The scores obtained by the measure have to allow a meaningful comparison between different segmentations of the same model and between segmentations of different models. For the first case (segmentations of the same model), the scores have to vary according to the segmentation quality, then, more the automatic segmentation is similar to the ground-truth segmentations of the same model, and better the score is. For the second case (segmentations of different models), the scores have to indicate which kind of 3D-models is the most convenient to segment by an automatic algorithm.

### 3.3.2 Categories of mesh segmentation similarity metrics

Essentially, the measures used to evaluate 3D-mesh segmentation can be classified into three categories: boundary matching, region differencing and non-parametric tests based measures.

In order to be able to formulate the above measures, we use definition 2.4 of mesh segmentation reported in chapter 2. We will use this latter definition for the remainder of this chapter.

**Boundary matching metrics**

This kind of measures compute the mapping degree between the region boundaries of two segmentations. Chen *et al.* (CGF09) proposed to use such a measure called *Cut discrepancy*. It measures the distances between cuts, where each cut represents an extracted region boundary. Let $S_1$ and $S_2$ be two segmentations of a 3D-mesh $M$ and $C_1$, $C_2$, their respective sets of points on the segment boundaries. Let $d_G(p_1, C_2) = min\{d_G(p_1, p_2), \forall p_2 \in C_2\}$ be the geodesic distance from a point $p_1 \in C_1$ to a set of cuts $C_2$.

The Cut discrepancy between $S_1$ and $S_2$ is then:

$$CD(S_1, S_2) = \frac{DCD(S_1 \Rightarrow S_2) + DCD(S_2 \Rightarrow S_1)}{avgRadius}$$

where, *avgRadius* is the average Euclidean distance from a point on the surface to the centroid of the mesh, and DCD is a directional function defined as $DCD(S_1 \Rightarrow S_2) = mean\{d_G(p_1, C_2), \forall p_1 \in C_1\}$.

A value of 0 indicates a perfect matching between $S_1$ and $S_2$ and is greater otherwise. As observed by Chen *et al.* (CGF09) the measure is undefined when the model has no cuts and decreases to zero as more cuts are added to a segmentation. Hence it suffers from a degenerative case (see section 3.3.1). In addition, it is not tolerant to refinement since for two segmentations that are perfect mutual refinements of each other, it can provide a large value. Moreover, for the unmatched points in $C_1$ and $C_2$ (points that have a geodesic distance which is greater than 0), it is possible to change their locations randomly and the measure will keep the same value. This measure is also not tolerant to imprecision of cut boundaries since it is based on geodesic distances. Finally, it allows to compare an automatic segmentation to only one ground-truth segmentation.

**Region differencing metrics**

These measures compute the consistency degree between the regions produced by two segmentations $S_1$ and $S_2$. Berretti *et al.* (BDBP09) have proposed an overlap index representing the extent to which a region $R_i$ of

an automatic segmentation overlaps to closest region $R_j$ of a ground-truth segmentation. The overlap index $O_{index}$ of $R_i$ is defined as:

$$O_{index} = max_j \frac{A(R_i \cap R_j)}{A(R_i)}$$

with $A(.)$ the operator that returns the area of a region. If we suppose that $S_1$ is the automatic segmentation and $S_2$ is the ground-truth segmentation, then the distance between them is the average of the Overlap index over-all regions of $S_1$. A value of 1 means that the two segmentations are exactly the same and is lower otherwise. This measure falls in a degenerative case when each region $R_i$ of a segmentation $S$ is represented by one facet. Indeed, in this latter case, computing the similarity, using the overlap index, between $S$ and any other segmentation will give a value equals to 1. This means that the segmentation $S$ is similar to any other segmentation. The measure does not allow a comparison to multiple ground-truths.

We and Chen *et al.* (CGF09) generalized the consistency error measure (MFTM01), used to evaluate 2D-image segmentation, for 3D-mesh segmentation evaluation. The measure is based on the computation of a local refinement error $L_{3D}$ of a vertex (or facet) $v_i$ between $S_1$ and $S_2$ and is defined as:

$$L_{3D}(S_1, S_2, v_i) = \frac{|R(S_1, v_i) \backslash R(S_2, v_i)|}{|R(S_1, v_i)|}$$

where the operator $\backslash$ denotes the set differencing, $|x|$ the cardinality of the set $x$, and $R(S, v_i)$ the region in segmentation $S$ that contains the vertex $v_i$, i.e. the subset of vertices corresponding to a sub-mesh $M_j$ of $S$ containing $v_i$.

This local refinement error produces a positive real valued output that represents the ratio of the number of vertices not shared between the first segment and the second one.

Given this $L_{3D}$, there exist two ways to combine it for all vertices into a global measure for the entire 3D-mesh: the Global Consistency Error (GCE) and the Local Consistency Error (LCE).

The Global Consistency Error (GCE) forces all local refinements to be in the same direction and is defined as:

$$GCE(S_1, S_2) = \frac{1}{N} min\{\sum_i L_{3D}(S_1, S_2, v_i), \sum_i L_{3D}(S_2, S_1, v_i)\}$$

The Local Consistency Error (LCE) allows for different directions of refinement in different segments of the 3D-mesh:

$$LCE(S_1, S_2) = \frac{1}{N} \sum_i min\{L_{3D}(S_1, S_2, v_i), L_{3D}(S_2, S_1, v_i)\}$$

where $N$ is the number of vertices. For both the GCE and the LCE, a value of 0 indicates a complete similarity, whereas a value of 1 indicates a maximum deviation between the two segmentations being compared. There are two degenerative segmentations that achieve a GCE and a LCE score of zero: one vertex per segment, and one segment for the entire mesh. We can also notice that the measure does not allow a comparison to multiple ground-truths.

Chen *et al.* (CGF09) proposed to use another measure namely Hamming distance. The Hamming distance between two segmentations $S_1$ and $S_2$ measures the region differencing between their respective sets of segments. The directional Hamming distance is defined as:

$$D_H(S_1 \Rightarrow S_2) = \sum_i \left\| R_2^i \backslash R_1^{i_t} \right\|$$

where the operator $\backslash$ denotes the set differencing, $\|x\|$ the cardinality of the set $x$, and $i_t = argmax_k \left\| R_2^i \cap R_1^k \right\|$ the closest segment in $S_1$ to the region (or segment) $R_2^i$ in $S_2$.

Given this $D_H$, and considering $S_2$ as the ground-truth, the authors of (CGF09) defined the missing rate $M_r$ and the false alarm rate $F_r$ as follow:

$$M_r(S_1, S_2) = \frac{D_H(S_1 \Rightarrow S_2)}{\|S\|},\ F_r(S_1, S_2) = \frac{D_H(S_2 \Rightarrow S_1)}{\|S\|}$$

and the Hamming distance as the average of missing rate and false alarm rate:

$$HD(S_1, S_2) = \frac{1}{2}(M_r(S_1, S_2) + F_r(S_1, S_2))$$

Similarly to the GCE and LCE, a value of 0 indicates a complete similarity between the two segmentations, and it is higher otherwise. As observed by the authors (CGF09) the measure has a good behavior when the correspondences between segments are correct but it fails when they are not. Another limit is the comparison to only one ground-truth.

**Non-parametric tests metrics**

In the statistical literature there exists a lot of non-parametric measures. We can cite for example Cohen's Kappa (Coh60), Jaccard's index (FM83), Fowlkes and Mallow's index (FM83). The latter two are variants of Rand index (Ran71). Chen *et al.* (CGF09) generalized Rand index (UPH07), used to evaluate 2D-image segmentation, for 3D-mesh segmentation evaluation. This index converts the problem of comparing two segmentations $S_1$ and $S_2$ with different numbers of segments into a problem of computing pairwise label relationships. If we denote $l^i_{S_1}$ the corresponding label of vertex $v_i$ (or facet) contained in region $R$ of $S_1$ and similarly $l^i_{S_2}$ the corresponding label of vertex $v_i$ in region $R$ of $S_2$, the Rand index can be computed as the ratio of the number of pairs of vertices or facets having a compatible label relationship in $S_1$ and $S_2$ and can be defined as:

$$RI(S_1, S_2) = \frac{1}{\binom{N}{2}} \sum_{\substack{i,j \\ i<j}} \mathbf{I}(l^i_{S_1} = l^j_{S_1})(l^i_{S_2} = l^j_{S_2}) + \mathbf{I}(l^i_{S_1} \neq l^j_{S_1})(l^i_{S_2} \neq l^j_{S_2})$$

where $\mathbf{I}$ is the identity function, and the denominator is the number of possible unique pairs among $N$ vertices or facets. This gives a measure of similarity ranging from 1, when the two segmentations are identical, to 0 otherwise. This measure does not allow comparison to multiple ground-truth segmentations.

We can notice that all existing measures suffer from either degenerative cases and/or sensitivity to refinement and/or sensitivity to cut boundary imprecision and/or limitation in term of comparison to multiple reference (i.e. ground-truth) segmentations. Therefore none of these measures satisfies the whole set of properties defined in section 3.3.1.

### 3.3.3 3D Probabilistic Rand Index (3D-PRI)

The goal of this measure is to perform a quantitative comparison between a mesh segmentation algorithm and a set of ground-truth segmentations (of the same model). In the field of 2D-image, Unnikrishnan *et al.* (UPH07) proposed a probabilistic interpretation of Rand Index to evaluate the performance of 2D-image segmentation algorithms and shown the relevance of the obtained results.

Hence we have generalized this measure for 3D-mesh segmentation evaluation.

Let $S_a$ be the automatic segmentation to be compared to a set of manual segmentations (ground-truths) $\{S_1, S_2, ..., S_K\}$ of a 3D-mesh $M$. We denote the corresponding label of a vertex $v_i$ (label of the segment to which belongs vertex $v_i$) by $l^i_{S_a}$ in segmentation $S_a$ and by $l^i_{S_k}$ in the ground-truth segmentation $S_k$. It is assumed that the label $l^i_{S_k}$ takes a value ranged between 1 and the number of segments of $S_k$ and similarly $l^i_{S_a}$ takes a value ranged between 1 and the number of segments of $S_a$. The label relationships for each vertex pair is modeled by an unknown underlying distribution. This can be considered as a process where each human segmenter provides information about the segmentation $S_k$ of the 3D-mesh in the form of binary numbers $\mathbf{I}(l^i_{s_k} = l^j_{s_k})$ for each pair of vertices $(v_i, v_j)$. The set of all perceptually correct segmentations defines a Bernoulli distribution over this number, giving a random variable with expected value denoted as $p_{ij}$. Hence, the set $\{p_{ij}\}$ for all unordered pairs $(i, j)$ defines a generative model of correct segmentations for the 3D-mesh $M$. The 3D Probabilistic Rand Index is then defined as:

$$\text{3DPRI}(S_a, \{S_k\}) = \frac{1}{\binom{N}{2}} \sum_{\substack{i,j \\ i<j}} e_{ij}p_{ij} + (1 - e_{ij})(1 - p_{ij}) \tag{3.1}$$

where $e_{ij}$ denotes the event of a pair of vertices $i$ and $j$ belonging to the same segment (or region) in the automatic segmentation:

$$e_{ij} = \mathbf{I}(l^i_{S_a} = l^j_{S_a})$$

and $p_{ij}$ denotes the probability of the vertices $i$ and $j$ belonging to the same segment in the ground-truth set $\{S_k\}$ and is given by the sample mean of

the corresponding Bernoulli distribution as suggested by Unnikrishnan *et al.* (UPH07):

$$p_{ij} = \frac{1}{K} \sum_{k} \mathbf{I}(l^i_{S_k} = l^j_{S_k})$$

The 3D-PRI takes a value ranged between 0 and 1, where 0 indicates no similarity between $S_a$ and $\{S_1, S_2, ..., S_k\}$, and 1 indicates a perfect similarity.

Note that with this formulation for $p_{ij}$, computing the 3D-PRI is equivalent to averaging the RI over the multiple ground-truths. However the 3D-PRI formulation is generic and we can imagine a different and more efficient way to compute the $p_{ij}$. The main advantage of the simple mean estimator is its fast computation.

We have noticed in practice, however, that the 3D-PRI suffers from a lack of discriminative power in its values. Indeed, the values obtained by the index do not allow to clearly decide if a segmentation obtained by an automatic algorithm is relevant or not. This is due to the limited effective range of 3D-PRI in term of maximum and minimum value. To address this drawback, we present in the next section, the 3D normalized probabilistic Rand index (3D-NPRI).

### 3.3.4 3D Normalized Probabilistic Rand Index (3D-NPRI)

Our objective is to normalize the 3D-PRI, in order to increase its dynamic range and thus its discriminative power. Hence we need to define a baseline to which the index can be expressed. For 3D-mesh segmentations, the baseline may be interpreted as the expected value of the index under some particular segmentations of the input 3D-model. A popular strategy (FM83, UPH07) of index normalization with respect to its baseline is:

$$Normalized\ index = \frac{Index - Expected\ index}{Maximum\ index - Expected\ index} \tag{3.2}$$

As observed by Unnikrishnan *et al.* (UPH07) there is a little agreement in the statistics community regarding whether the value of "Maximum index" should be estimated from the data or set constant. We choose to follow what was done by Unnikrishnan *et al.* (UPH07) and set the value

to be 1 (the maximum possible value of the 3D-PRI). Thus, we avoid the practical difficulty of estimating this quantity for complex data sets.

Another parameter to define is the expected probabilistic Rand index E(3D-PRI). One may draw an analogy between the E(3D-PRI) and the 3D-PRI in equation 3.1 as follow:

$$E\left[\text{3DPRI}(S_a, \{S_k\})\right] = \frac{1}{\binom{N}{2}} \sum_{\substack{i,j \\ i<j}} e'_{ij} p_{ij} + (1 - e'_{ij})(1 - p_{ij}) \tag{3.3}$$

where $e'_{ij} = E\left[\mathbf{I}(l^i_{S_a} = l^j_{S_a})\right]$. This latter quantity has to be computed in a meaningful way. Unnikrishnan *et al.* (UPH07) proposed to estimate it from segmentations of all images of the database for all unordered pairs $(i, j)$. Let $\Phi$ be a number of images in data-set and $K_\phi$ the number of ground-truth segmentations of image $\phi$. Then, $e'_{ij}$ is expressed as:

$$e'_{ij} = \frac{1}{\Phi} \sum_{\phi} \frac{1}{K_\phi} \sum_{k=1}^{K_\phi} \mathbf{I}(l^i_{S^k_\phi} = l^j_{S^k_\phi})$$

However, this estimation can only be used in a data-base of 2D-images having equal sizes (where each pixel has its correspondent over all the other segmented images). In the 3D case, it is not possible, since the different models of the corpus have different number of vertices and different connectivities. One possible way to compute the E(3D-PRI) while keeping a correct baseline and without having any constraint on the corpus, is to use random segmentations $S_r$:

$$E\left[\text{3DPRI}(S_a, \{S_k\})\right] = \frac{1}{N} \sum_{r=1}^{N} \text{3DPRI}(S_r, \{S_{K_r}\}) \tag{3.4}$$

where $N$ is the number of 3D-models in our corpus and $\{S_{k_r}\}$ are ground-truths of the model concerned by $S_r$. We then define the 3D-NPRI of an automatic segmentation of a given 3D-model as follow:

$$\text{3DNPRI}(S_a) = \frac{\text{3DPRI}(S_a, \{S_K\}) - E\left[\text{3DPRI}(S_a, \{S_k\})\right]}{1 - E\left[\text{3DPRI}(S_a, \{S_k\})\right]} \tag{3.5}$$

The random segmentations were generated using a simple algorithm: $L$ seed vertices were randomly chosen on the object, then $L$ connected regions were obtained by a simple region growing mechanism. The number of segments takes a value ranged between 2 and the number of vertices of

the concerned model. Figure 3.7 shows some 3D-models of the corpus on which the random segmentation algorithm was applied. We have to precise here that the 3D-NPRI is not affected by the choice of these random segmentations. Indeed we will show later (see figure 3.8) that the 3D-PRI provides very stable values when comparing ground-truth segmentations to random segmentations (even with very different granularities) hence the normalization constant E(3D-PRI) (see equation 3.4) is almost invariant to the choice of the random segmentations $S_r$.

Figure 3.7 – *Random segmentations of some 3D-models of the corpus.*

Hence, the 3D-NPRI takes a value with a lower bound of $-1$ and an upper bound of 1, where $-1$ indicates no similarity between the automatic segmentation and the ground-truth segmentations of the same model, and 1 indicates a perfect match. The lower bound of $-1$ is explained by the fact that the expected Index can not exceed 0.5 since we compare a set of random segmentations to a set of ground-truth segmentations (see section 3.4.1). Therefore, the worst case will be:

$$3\text{DNPRI}(S_a) = \frac{0 - 0.5}{1 - 0.5} = -1$$

where the automatic segmentation has no similarity with its corresponding ground-truths.

Note that the metric does not allow to compare different segmentations of the same model with different sampling (the same model with vertex number varying). However, in our case, it is not really a drawback since we always compare segmentations of the same model while keeping the same sampling and the same order of vertices.

## 3.4 Experimental comparison of existing segmentation similarity metrics

In what follows, we provide an experimental study of the 3D-PRI/3D-NPRI properties and we compare them to the existing metrics for assessing 3D-mesh segmentation quality. For this end, we use our corpus models and their corresponding ground-truths.

Most of the measures introduced in section 3.3 quantify *dissimilarity* (the lower is the number, the best is the segmentation result) between segmentations rather than *similarity*. In order to have a meaningful comparison between these measures and the 3D-PRI/3D-NPRI, we define the quantities $CDI(S_1, S_2) = 1 - CD(S_1, S_2)$, $GCI(S_1, S_2) = 1 - GCE(S_1, S_2)$, $LCI(S_1, S_2) = 1 - LCE(S_1, S_2)$, and $HDI(S_1, S_2) = 1 - HD(S_1, S_2)$. The "I" in the acronyms stands for "Index", complying with the popular usage of the term in statistics when quantifying similarity. Hence, except the CDI, all of the other indexes are in the range $[0, 1]$ with a value of 0 indicating no similarity between segmentations of the same model and a value of 1 indicating a perfect match. The CDI is in the range $]-\infty, 1]$.

### 3.4.1 Sensitivity to degenerative cases

The first property to study is the sensitivity of each index regarding degenerative cases. For this end, we compare our Probabilistic Rand Index (3D-PRI) with the Cut Discrepancy Index (CDI), the Hamming Distance Index (HDI), the Global and Local Consistency Index (GCI/LCI), and the Overlap Index (OI) for three kinds of random segmentations namely *extreme-low segmentation* (segmentation composed of 2 or 3 segments), middle-segmentation (segmentation composed of a number of segments which is similar to that of ground-truths of the corresponding model), and *extreme-high segmentation* (segmentation composed of more than 50 segments). They were generated using a random segmentation algorithm. Figure 3.8 presents the results obtained by the comparison of these random segmentations to the set of the ground-truths for each model of the corpus. Each index of the figure is computed for the three

kinds of segmentation (extreme-high segmentation, middle-segmentation, and extreme-low segmentation) and averaged across the entire data set. Since the segmentations are random, the scores obtained by the metrics are expected to be low for the three kinds of segmentation, and it is the case for the 3D-PRI. We can notice, however that although the random segmentations are totally different from the ground-truths, the scores of the other metrics are very high (i.e. very good) for certain segmentations with degenerative granularity (extreme-high and/or extreme-low). Hence the 3D-PRI is the most stable regarding degenerative cases considering its scores which are always less than 0.32.

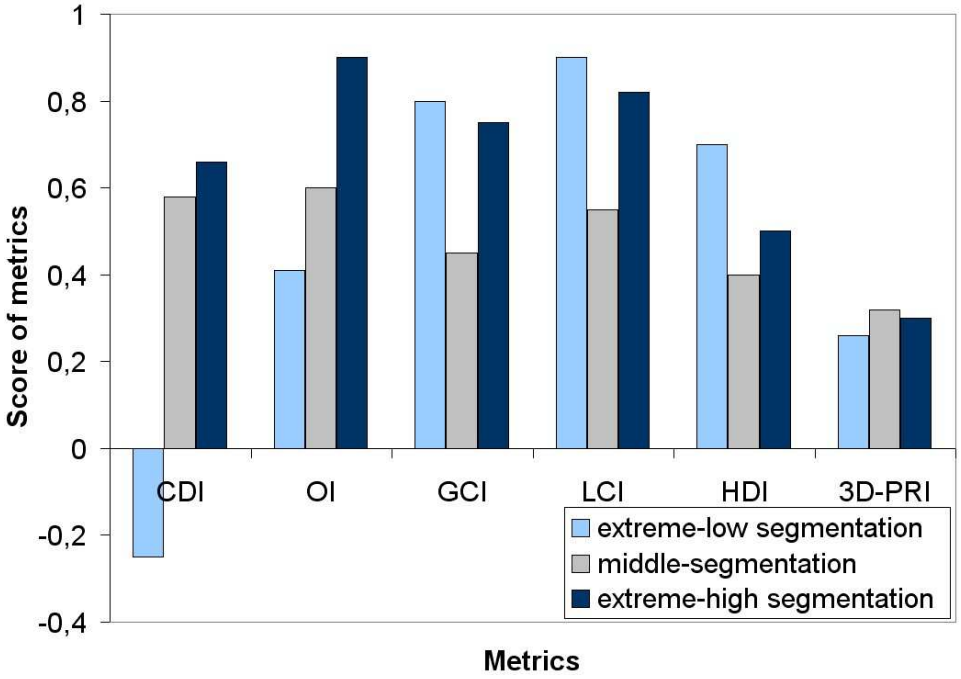



Figure 3.8 – *Comparison of three levels of random segmentation (extreme-low, middle, and extreme-high) to the ground-truths for the whole corpus using different indexes.*

### 3.4.2 Tolerance to refinement

The second property to study is the tolerance of each index to refinement. For this end, we perform two kinds of experiments. The first one uses segmentations with mutual refinements, and the second one uses segmentations with hierarchical refinements. The obtained results for the first experiment are presented in figure 3.9.

It shows two segmentations of the *dinopet* model which are perfect mu-

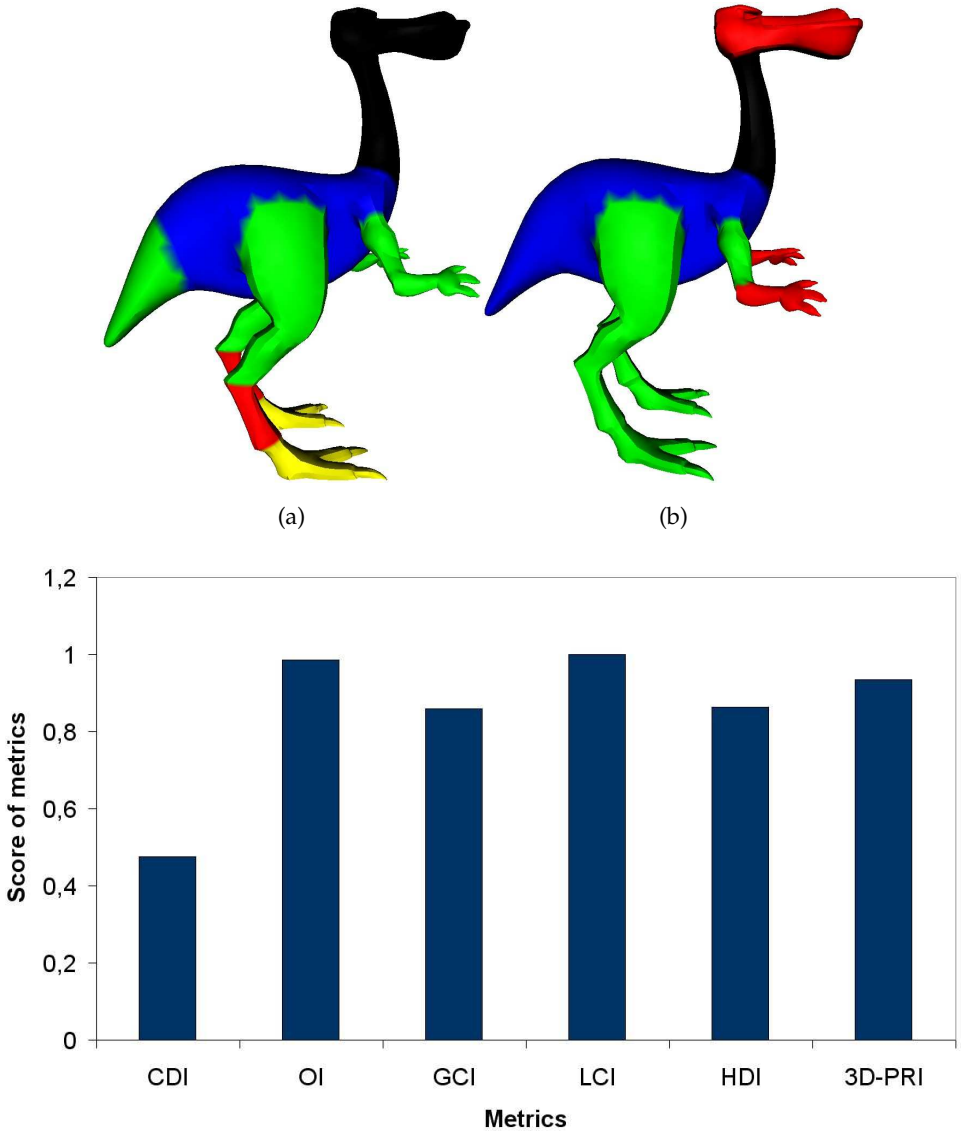



Figure 3.9 – *Tolerance to mutual refinement of different indexes, by comparing two segmentations (a,b) with perfect mutual refinement for the dinopet model.*

tual refinements of each other, and a plot in which is computed the similarity between the two segmentations using different metrics. The plot of figure 3.9 clearly shows that the CDI fails to capture the similarity between the two segmentations (a) and (b). Although the two segmentations are similar (the difference just lies in the level of refinement). However, the other metrics have a good behavior toward this kind of refinement since all of them give scores which are close to 1.

The second experiment was performed using the hierarchical segmentation algorithm of Attene *et al.* (AFS06). We generated several levels of segmentation (from 4 segments to 15 segments) on the *horse* model of our corpus then we compared these 12 versions to the ground-truths. Figure 3.10 illustrates the obtained results using different indexes. The OI and the GCI does not appear on the figure since they have the same behavior as the LCI. The figure clearly shows that the CDI is less stable toward hierarchical refinement than the other indexes. The LCI seems completely invariant while the 3D-PRI and the HDI present a slight variation; they are not fully invariant but present a good tolerance to refinement.

### 3.4.3 Independence from cardinality

The third property to study is the independence of each index toward segmentation cardinality. According to the previous performed experiments about the first two properties (degenerative cases and refinement), the CDI seems to be the only metric which depends on the cardinality, in a critical way. Indeed, the comparison between two segmentations with different number of segments will give a bad score using this metric whatever the quality.

### 3.4.4 Tolerance to imprecision of cut boundaries

The fourth property to study is the tolerance of each index to the imprecision of cut boundaries. For this end, we manually segmented a simple model (bitorus) into two segments. We proposed five segmentations (figure 3.11 (a to e)) where each one of them has a slight difference in the boundary position with comparison to the others, then we computed the

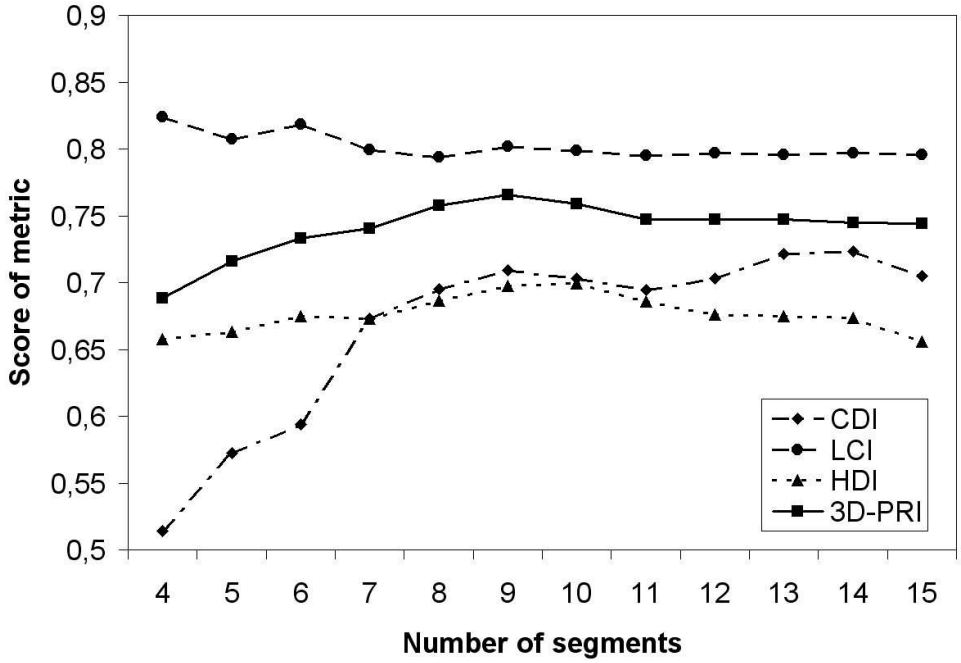



Figure 3.10 – *Tolerance to hierarchical refinement of different indexes, by comparing several levels of segmentation of the horse model to its corresponding ground-truths.*

similarity between segmentation (c) and the other segmentations. The plot in figure 3.11 shows the obtained results using different indexes. Contrary to the other indexes, the CDI gives low values of similarity between segmentations. Although the CDI is not in the same range as the other metrics, the plot still allows to illustrate the qualitative behavior of this latter index toward the imprecision of cut boundaries. We can notice also that except the 3D-PRI which presents a slight variation but a good tolerance, the other indexes are almost invariant.

At this point, we have shown that the 3D-PRI satisfies the five properties: ability to compare one automatic segmentation with multiple ground-truth segmentations, non degenerative cases, tolerance to refinement, independence from segmentation cardinality, and tolerance to imprecision of cut boundaries. We also have shown that the 3D-PRI outperforms the other indexes in terms of the first two properties. We show in the next experiments that the normalization of this index (into 3D-NPRI) improves its discriminative power and give better results in term of meaningful comparison.

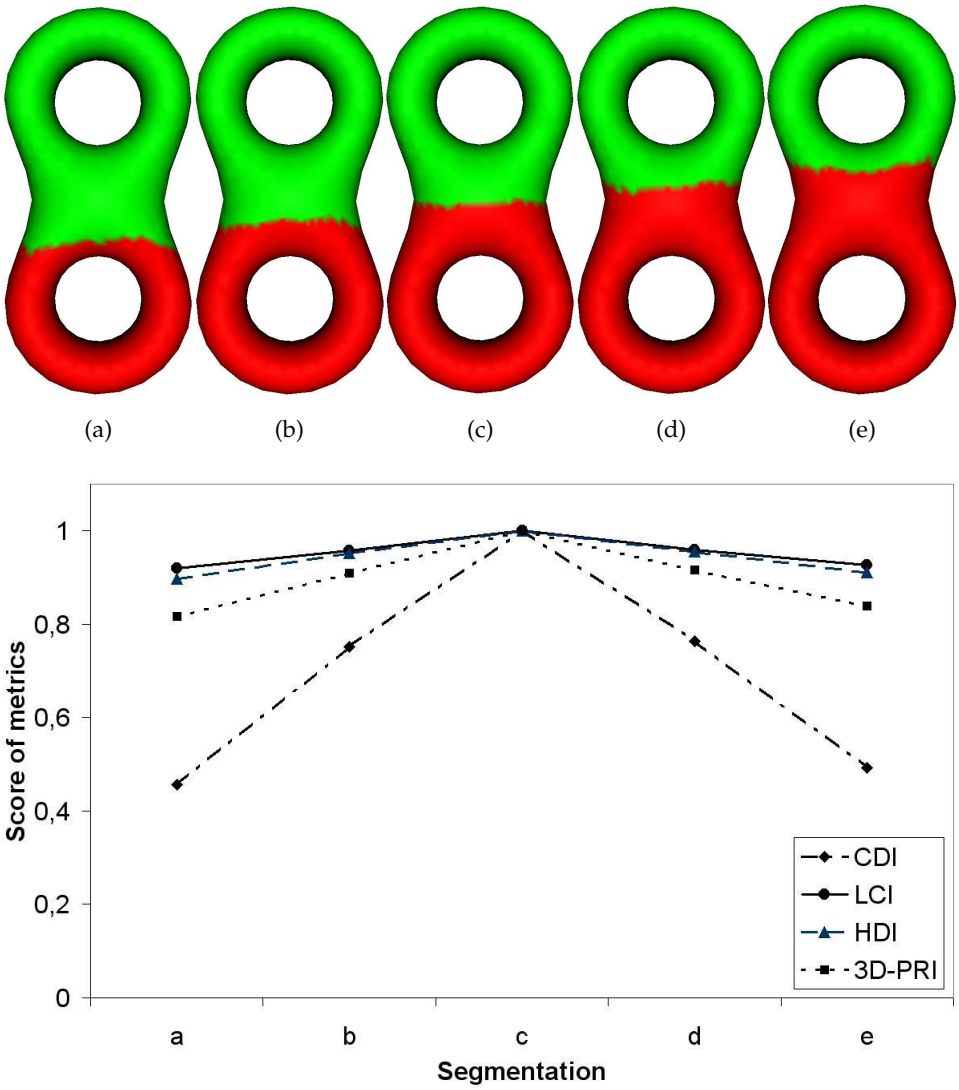



Figure 3.11 – *Tolerance to imprecision of cut boundaries of different indexes, by comparing segmentation (c) to segmentations (a to e) for the bitorus model.*

### 3.4.5 Meaningful comparison

The main advantage of the 3D-NPRI is the ability to provide values that allow a meaningful comparison between segmentations of different 3D-models. Figure 3.12 demonstrates this behavior. The top two rows show different 3D-models of our corpus segmented at different granularity using the hierarchical algorithm from Tierny *et al.* (TVD07). These automatic segmentations are compared to the ground-truth corpus (see figure 3.3 on page 49) using the previous indexes and our 3D-NPRI. Visually, regarding the ground-truth, segmentations a and b (figure 3.12) seem very poor,

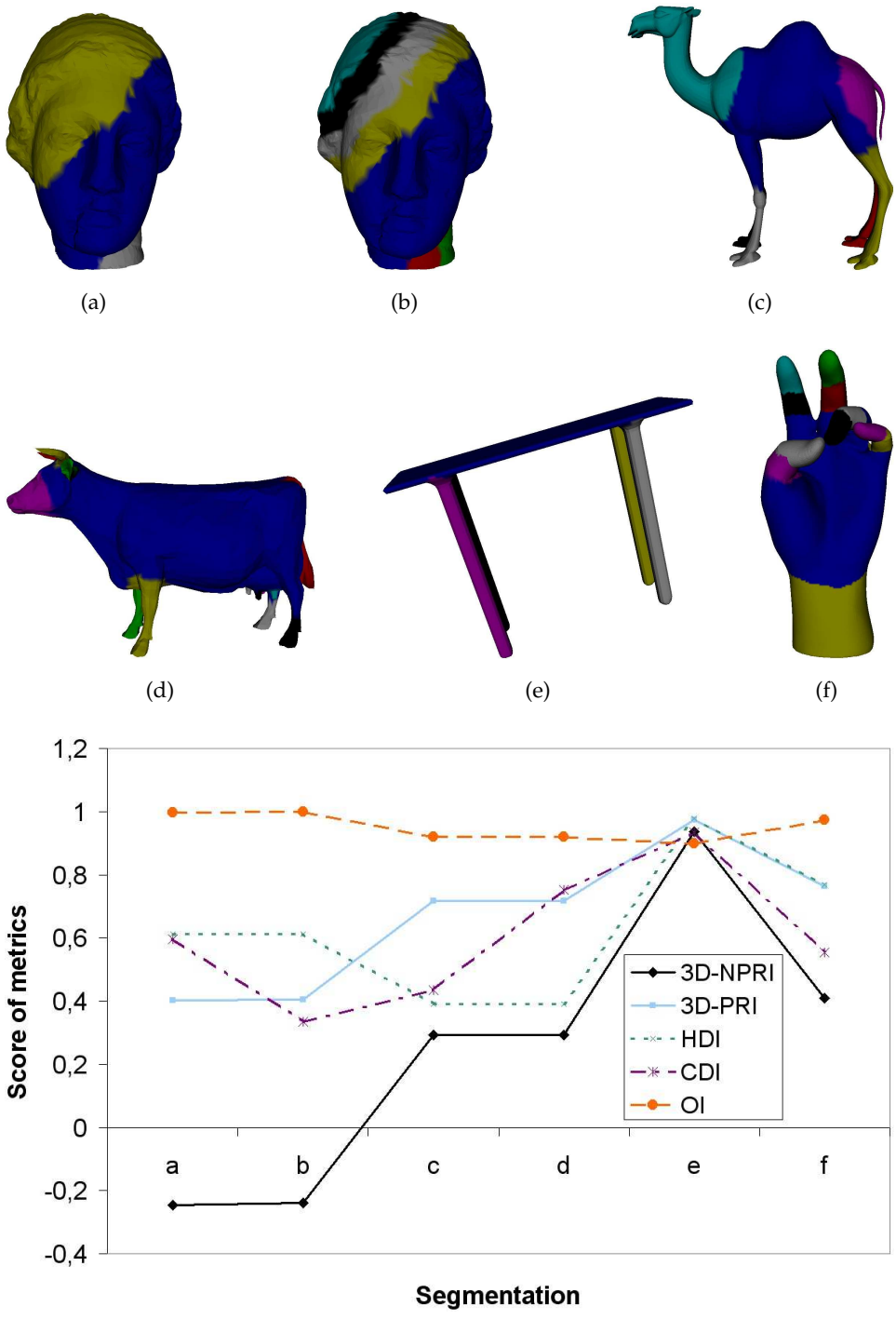



Figure 3.12 – *Example of comparing segmentations of different models: From a to f segmentations using algorithm from (TVD07). The plot shows the scores of different indexes for each segmentation (a to f).*

segmentations c, d, and f are correct, and segmentation e is perfect. One can notice that the OI similarity is high for all of the 3D-models. Hence, it cannot indicate which segmentation is the best. Note that although the HDI gives lower scores than the OI, it also fails to distinguish between correct and poor segmentations since it gives high values for poor ones (figure 3.12.a and 3.12.b) and low values for correct ones (figure 3.12.c and 3.12.d). The GCI/LCI does not appear in the plot in order to keep a clear display. This latter metric has the same behavior than HDI. The CDI has slightly a better behavior than HDI but still to fail distinguishing between correct and poor segmentations. The 3D-PRI reflects the correct relationship among the segmentations. However, its range is small, and the expected value is unknown, hence it is difficult to determine which segmentation is really good. The 3D-NPRI fixes all of these drawbacks. It reflects the desired relationship among the segmentations with no degenerate cases. Besides, any segmentation which gives a score significantly above 0 can be considered as relevant (since it provides results significantly better than random segmentations).

## 3.5 Subjective experiment

In order to attest the discriminative power of our 3D-NPRI and to quantify the efficiency of our ground-truth corpus, we have conducted a subjective experiment in which human observers have rated a set of segmentations issued from different automatic algorithms including ground-truths of our corpus and random segmentations. To this end, we carefully designed a protocol with respect to several factors namely the rendering conditions, the possible interactions, the rating range, and the number of human subjects.

### 3.5.1 The corpus of segmentations

The design of stimulus is a critical step in a subjective protocol. In our case, we need to select a set of 3D-models that will be segmented by different algorithms and then rated by human subjects. To this end, we use our corpus of 3D-models. The size of the corpus is reasonable (28 3D-models),

and its content is representative since it contains different categories of 3D-models.

In our experiment, we asked human subjects to rate segmentations of these objects coming from different automatic algorithms. We have created a set of 250 segmentations based on the corpus of 28 models.

For this task, we have considered four automatic segmentation algorithms: Attene *et al.* (AFS06), Lavoué *et al.* (LDB05b), Shapira *et al.* (SSCO08) and Tierny *et al.* (TVD07). The source code and/or the binary were provided by the authors for each algorithm. Except the algorithm of Lavoué *et al.* (LDB05b), the others are hierarchical. Hence, for each algorithm, we generated two levels of segmentation per model namely coarse and fine, which gave $28 \times 2$ segmentations per algorithm and 28 segmentations for the Lavoué's *et al.* algorithm. Figure 3.13 illustrates an example of coarse and fine segmentation of the *hand* model using the algorithm from Tierny *et al.* (TVD07).

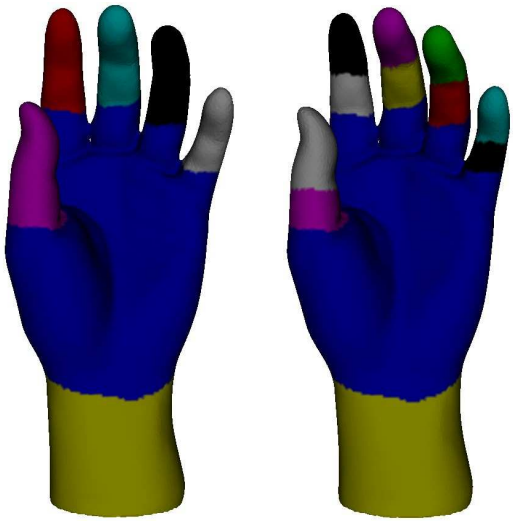


Figure 3.13 – *From left to right, coarse and fine segmentation of the hand model using Tierny's* et al. *(TVD07) algorithm.*

Note that the number of segments of a given level of segmentation (coarse for example) is not the same through the different models and through the different algorithms. For the algorithms from Shapira *et al.* (SSCO08) and Tierny *et al.* (TVD07), the number of segments is automatically computed. We just need to fix the level of detail of the desired

segmentation. For the algorithm from Attene *et al.* (AFS06), the number of segments is manually fixed, we then select two numbers (a small one and a big one). These two numbers vary according to the complexity of the model and to the consistency of the segmentation. For the algorithm from Lavoué *et al.* (LDB05b) the number of segments was also manually chosen so as to optimize the quality. To these $28 \times 7$ segmentations were added 28 manual segmentations coming from our ground-truth corpus and 28 random segmentations generated using a simple algorithm based on a random region growing mechanism. Figure 3.14 illustrates different segmentations of the *camel* model. Thus we obtained a whole corpus of 250 segmentations to rate.

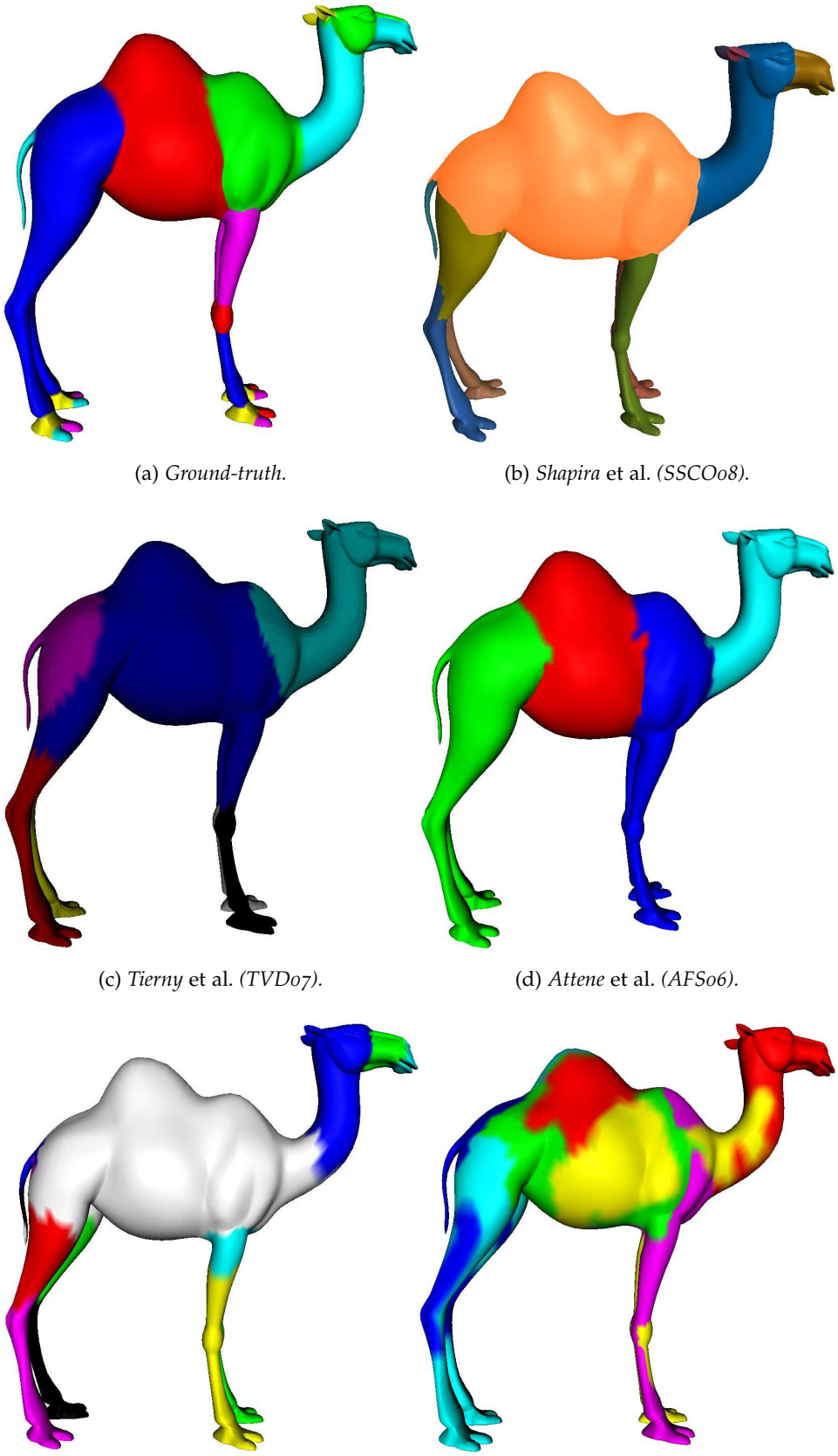
### 3.5.2 Subjective protocol

The protocol that we propose is inspired from existing ones used for video segmentation quality evaluation, 3D-watermarking quality evaluation, and image quality evaluation (GEKS06, CDGEB07, RR01). They are all based on *Single Stimulus Continuous Quality Scale* (SSCQS) which is a standard technique used to evaluate the quality of video and multimedia content. Our protocol consists of the following stages:

- ***Oral instructions.*** We give instructions to our volunteers and make them familiar with the rating task, the 3D-models, and the available interactions.

- ***Training.*** We show some ground-truth and random (bad) segmentations of several models, in order to clarify the concept of good and bad segmentation for the user and to establish a referential range for him. The goal for the user is not to learn the ground-truth of each model, but to learn what is a good segmentation so as to be able to rate the quality of a given segmentation independently from ground-truths.

- ***Experimental trials.*** For each segmentation from the corpus, we ask the volunteer to give a score between 1 and 10 indicating its quality from a semantic point of view. 10 for a perfect segmentation and 1



(a) *Ground-truth.* (b) *Shapira* et al. *(SSCO08).*

(c) *Tierny* et al. *(TVD07).* (d) *Attene* et al. *(AFS06).*

(e) *Lavoué* et al. *(LDB05b).* (f) *Random.*

Figure 3.14 – *Segmentation of the camel model using different algorithms.*

for a bad one. This scale range allows the volunteers to distinguish more easily between the quality of segmentations.

During the experiment trials, each segmentation is displayed one by one to the observer on a 22-inch LCD monitor, without the ground-truth.

In order to avoid the effect of the temporal sequencing factor, the sequence of segmentations was randomly generated for each participant. Interaction was allowed (rotation, scaling, translation). It is important to notice that rating 250 segmentations represents a too much fastidious task for an observer; hence we only asked each human subject to rate 50 segmentations from the corpus (randomly chosen with a bias to obtain enough scores for all the 250 segmentations). The user interface which was developed for this rating task is illustrated in figure 3.15.

The Mean Opinion Score (MOS) is then computed for each segmentation of the corpus:

$$MOS_i = \frac{1}{n} \sum_{j=1}^{n} m_{ij} \tag{3.6}$$

$MOS_i$ is the mean opinion score of the $i^{th}$ segmentation, $n$ is the number of test subjects, and $m_{ij}$ is the score (from 1 to 10) given by the $j^{th}$ subject to the $i^{th}$ segmentation. This subjective experiment has been conducted on 50 people (students and staff) from the University of Lille, which provided a total of 10 opinion scores per segmentation.

### 3.5.3 Results and data analysis

#### Consistency of the ratings

Firstly in order to check the suitability of our evaluation protocol and the relevance of the mean opinion scores, we have assessed the variation between the different observers in their subjective ratings of the objects. The value of the intraclass correlation coefficient (ICC) is 0.65, that is a rather good value that means that the observers had a good agreement on their visual estimations; hence we can assert that our protocol was correct since it led to produce meaningful consistent ratings among the observers.

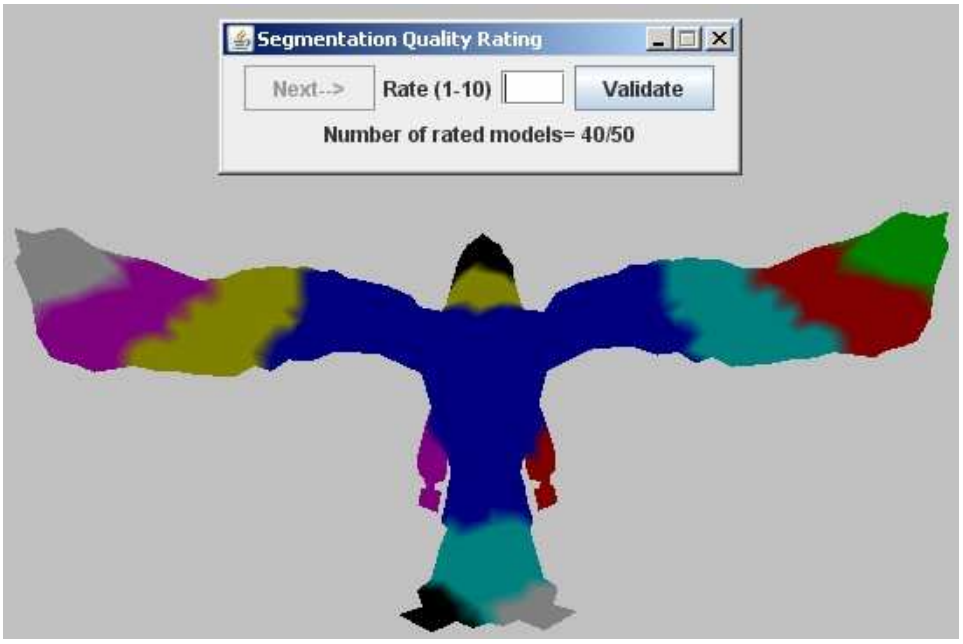



Figure 3.15 – *User interface for rating the segmentations.*

**Discriminative power**

The best way to attest the discriminative power of our 3D-NPRI, is to show that its values are well correlated with the rates given by the observers for the 250 segmentations. To this end, we computed the quality index of these 250 segmentations (using our benchmark) using the different metrics. Hence each segmentation was associated with quality index values from the different metrics and a subjective Mean Opinion Score (MOS) from the human observers.

For the correlation, we considered a statistical indicator namely the Pearson Product Moment Correlation (Dan99). This indicator measures the linear dependence between two variables X and Y. In order to optimize the matching between the values of the different metrics and the MOS of observers, we performed a psychometric curve fitting using the Gaussian psychometric function (recommended by Corsini *et al.* (CGEB07)).

Table 3.1 shows the results of Pearson correlations between the values of the different metrics and the MOS of observers.

The results in the table show that the 3D-NPRI outperforms the other metrics in term of correlation for each category and for the whole corpus. Moreover, the Pearson correlation value of the 3D-NPRI for the whole cor-

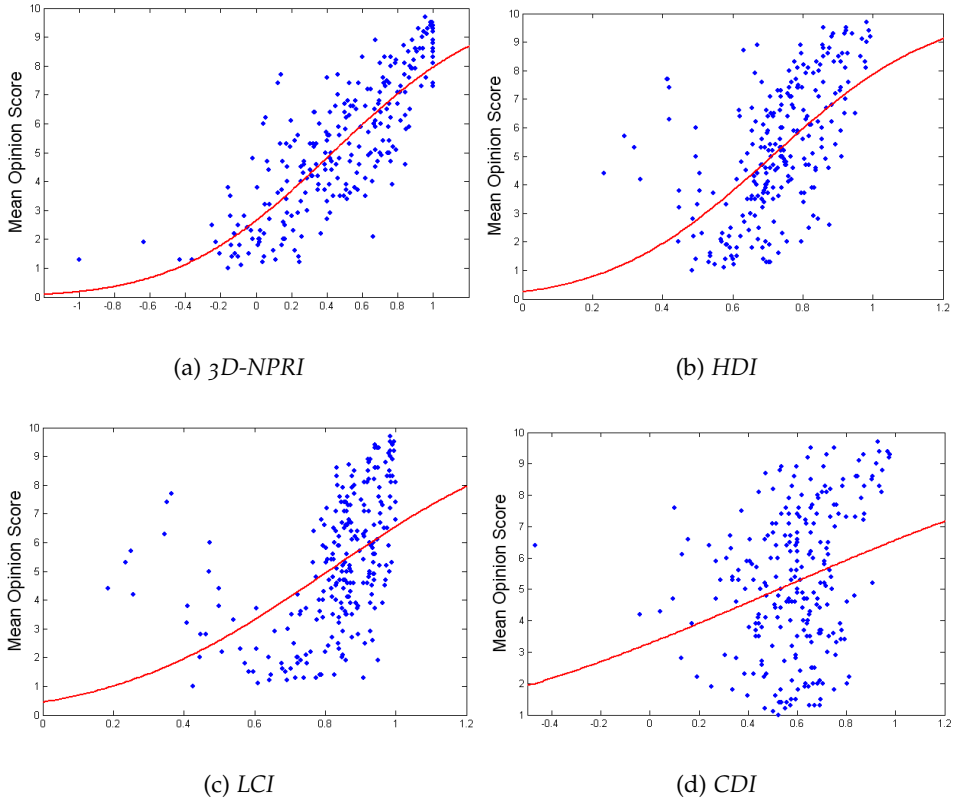
| | CDI | OI | GCI | LCI | HDI | 3D-NPRI |
|---|---|---|---|---|---|---|
| Animal | 2.6 | 2.3 | 9.3 | 8.3 | 16.9 | 58.7 |
| Bust | 10.9 | 0 | 45.9 | 61.1 | 54.8 | 77.4 |
| Furniture | 5.8 | 14.8 | 49.9 | 50.5 | 63 | 73.2 |
| Hand | 21.2 | 1 | 54.1 | 54.4 | 57.5 | 70.2 |
| Human | 1.5 | 5.5 | 32.1 | 32.6 | 39 | 51.6 |
| **Whole** | 7.1 | 2.6 | 23.7 | 20.9 | 32.9 | 66.1 |

Table 3.1 – *Pearson correlation values (%) between the Mean Opinion Scores and the values of the different metrics for each model category of our corpus.*

pus is high (66.1%), when those of the other metrics are quite bad (less than 33%). This means that except the 3D-NPRI, the other metrics fail to distinguish between good and bad segmentations. Figure 3.16 presents the psychometric curve fitting between the objective and subjective scores for 3D-NPRI, HDI, LCI and CDI for the 250 segmentations of the corpus models. It visually illustrates the superiority of the 3D-NPRI for predicting the subjective opinion, and leads to conclude that the 3D-NPRI has the best discriminative power. These results clearly validate the 3D-NPRI. Moreover, we attest that our benchmark (ground-truth corpus and metric) is able to predict the subjective quality of a segmentation since the obtained results are in agreement with the human opinion.

**Performance comparison of several segmentation algorithms**

Table 3.2 presents the rank, based on the MOS, of each algorithm (fine segmentation for hierarchical algorithms) for each category of models of the corpus including random segmentations and ground-truths. The MOS mean values are also displayed. As expected, our ground-truths have the best ranks for each category and for the whole corpus, when random segmentations have the worst ones. This validate once again the relevance of our ground-truth corpus. The table shows that there is no automatic algorithm which outperforms the others in all categories. It also shows that the models of the *bust* category, seem to be the most difficult to segment by automatic algorithms, since the average of their MOS is the lowest with




(a) *3D-NPRI* (b) *HDI*

(c) *LCI* (d) *CDI*

Figure 3.16 – *Subjective MOS vs. metric values for the whole corpus models and for different metrics. Each circle represents a segmentation. The Gaussian fitted curve is displayed in red.*

comparison to other categories. This may be due to the geometrical and topological complexity of these models, but the main reason is probably the fact that these models represent human faces. Human face images are well-known in subjective experiments as a high-level factor attracting human attention, hence some features not relevant from a geometrical point of view can be considered highly relevant for human observers. Globally, the algorithm from Shapira *et al.* (SSCO08) seems to be the best one after ground-truths.

#### Influence of the refinement on the segmentation quality

Some automatic algorithms are hierarchical, i.e. they are able to produce segmentations with different levels of refinement. An interesting experiment is to study whether this level of granularity influences the quality perceived by the observers. For this end, we averaged the MOS of the

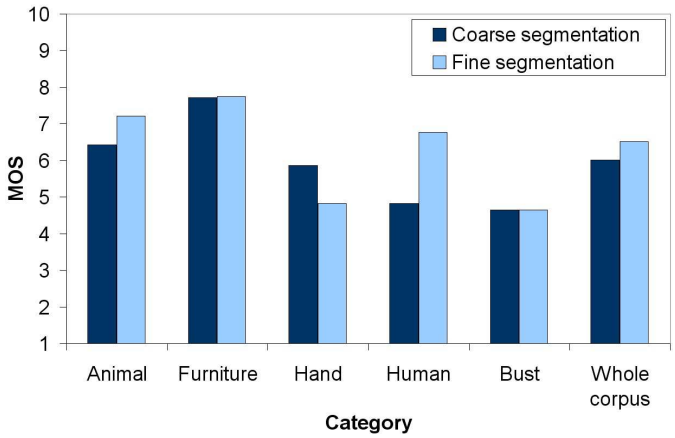
| | **Ground-truth** | **Shapira *et al.* (SSCO08)** | **Tierny *et al.* (TVD07)** | **Attene *et al.* (AFS06)** | **Lavoué *et al.* (LDB05b)** | **Random** |
|---|---|---|---|---|---|---|
| Animal | 1 / 8.26 | 2 / 7.20 | 3 / 5.72 | 5 / 4.83 | 4 / 5.01 | 6 / 2.37 |
| Bust | 1 / 8.03 | 2 / 4.64 | 4 / 2.81 | 3 / 3.64 | 5 /2.64 | 6 / 1.78 |
| Furniture | 1 / 9.25 | 3 / 7.74 | 5 / 3.35 | 2 / 8.53 | 4/6.21 | 6 / 1.99 |
| Hand | 1 / 8.68 | 5 / 4.82 | 2 / 7.64 | 4 / 4.85 | 3 / 5.53 | 6 / 1.60 |
| Human | 1 / 7.77 | 2 / 6.77 | 3 / 5.20 | 5 / 4.54 | 4/ 4.62 | 6 / 2.28 |
| **Whole** | 1 **/** 8.36 | 2 **/** 6.51 | 3 **/** 5.27 | 4 **/** 5.21 | 5 **/** 4.92 | 6 **/** 2.10 |

Table 3.2 – *Algorithms ranking associated with the average of MOS for corpus categories.*

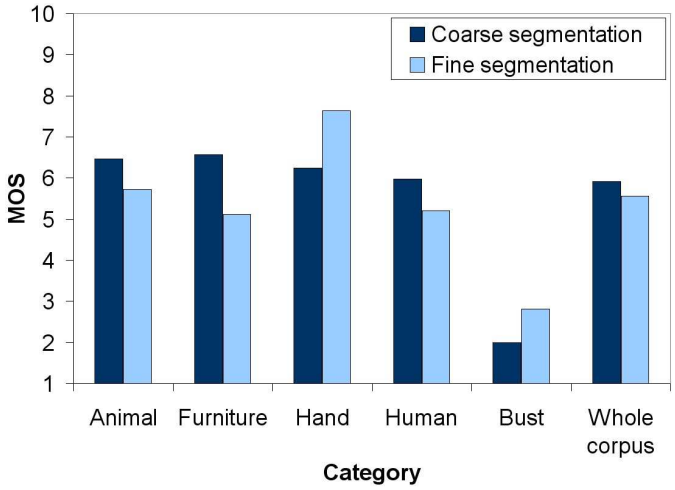
models for each category, for each algorithm and for both levels of segmentation (coarse and fine), then we compared the results of the two levels. Figure 3.17 illustrates the obtained results for the three hierarchical algorithms. One can notice that the averages of the two levels of segmentation for a given category or for the whole corpus are close to each other. More exactly, the average variation between the two levels for the whole corpus and for each algorithm: Shapira *et al.* (SSCO08), Attene *et al.* (AFS06) and Tierny *et al.* (TVD07) is respectively of 7%, 11%, and 10%. This means that the segmentations remain consistent whatever their level of refinement.

## 3.6 Application for the evaluation of recent segmentation algorithms

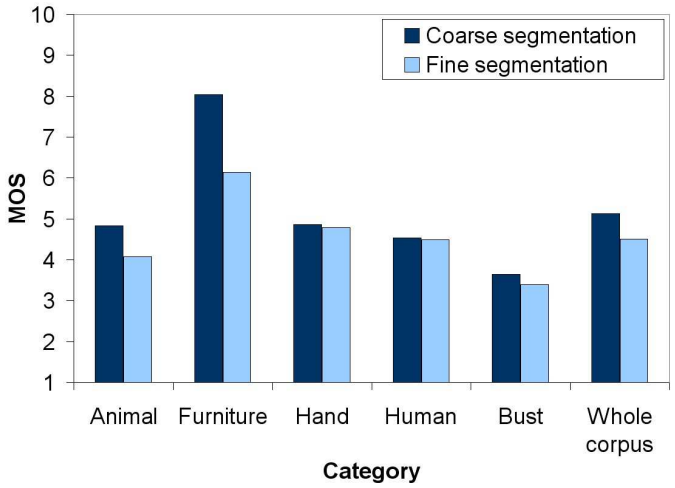
In this section, we apply the 3D-NPRI together with the Chen's *et al.* (CGF09) corpus and our corpus to evaluate a set of recent automatic segmentation algorithms, then we compare the results obtained by the two corpuses.




(a) *Shapira* et al. *(SSCO08).*




(b) *Tierny* et al. *(TVD07).*




(c) *Attene* et al. *(AFS06).*

Figure 3.17 – *Average of MOS of segmentations obtained from different hierarchical algorithms.*

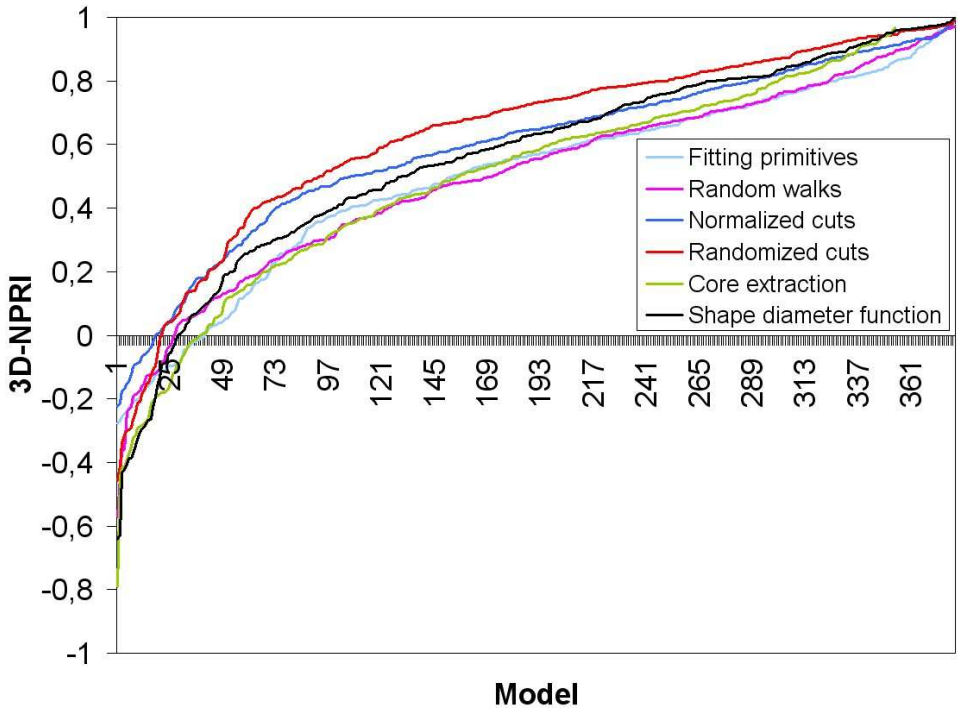
We have considered the six recent automatic segmentation algorithms used in the benchmark from Chen *et al.* (CGF09): Attene *et al.* (AFS06), Lai *et al.* (LHMR08), Golovinskiy *et al.* (GF08), Katz *et al.* (KLT05), and Shapira *et al.* (SSCO08). These algorithms are respectively based on: fitting primitives, random walks, normalized cuts/randomized cuts, core extraction, and shape diameter function. The segmentations from these algorithms for the Chen's corpus are available on-line. On the other hand, we used Attene's *et al.* (AFS06), and Shapira's *et al.* (SSCO08) algorithms (the only algorithms available on-line among the previous six) to generate automatic segmentations on our corpus models. The reader can refer to chapter 2 for more details about the six algorithms.

Note that all the algorithms cited above are part-type hierarchical segmentation methods. Hence for each of them we can generate several levels of segmentation. Chen *et al.* (CGF09) provided only one level of segmentation for each algorithm applied on their corpus. To this end, they used the parameter settings recommended by the authors of the algorithms. To keep a valid comparison between the two corpuses, we also used the parameter settings recommended by the authors of the algorithms to generate segmentations on the models from our corpus. Note that the level of segmentation should not influence the evaluation results since we proved that the 3D-NPRI is tolerant to hierarchical refinement (see figure 3.10).

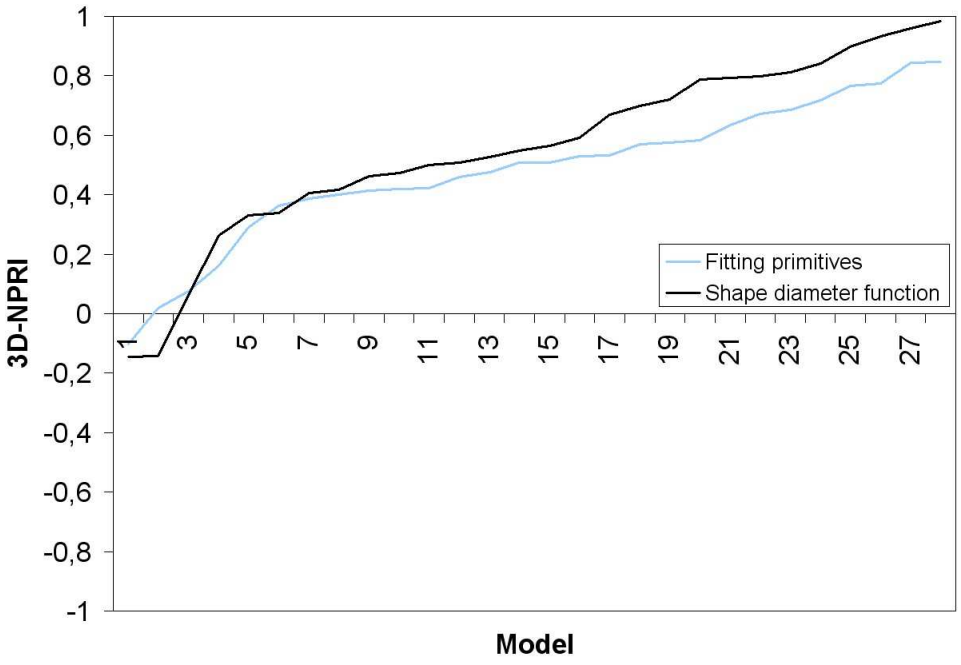
To ensure a relevant comparison between the algorithms, we compute the 3D-NPRI for every 3D-model of the Chen's corpus and of our corpus. Figure 3.18 shows the 3D-NPRI for each model of the two corpuses and for each algorithm. The values are sorted in increasing order for each algorithm, hence the $j^{th}$ model may not be the same across algorithms. This kind of graph was already applied for segmentation evaluation in the field of 2D-image (UPH07).

Table 3.3 presents the rank of each algorithm together with the 3D-NPRI mean value over all the two corpuses.

Table 3.3 and figure 3.18 demonstrate, as expected, that the segmentations obtained by the six algorithms are relevant since most of the values of the 3D-NPRI are greater than zero. The Randomized Cut algorithm




(a) *Results on Chen'*set al. *(CGF09) corpus*




(b) *Results on our corpus*

Figure 3.18 – *Scores of 3D-NPRI sorted in increasing order over all the two corpus models.*

| **Algorithm** | **3D-NPRI mean** | **Rank** |
|---|---|---|
| Fitting primitives | 0.49/0.49 | 5/2 |
| Random walks | 0.50/- | 4/- |
| Normalized cuts | 0.59/- | 2/- |
| Randomized cuts | 0.63/- | 1/- |
| Core extraction | 0.46/- | 6/- |
| Shape diameter function | 0.56/0.55 | 3/1 |

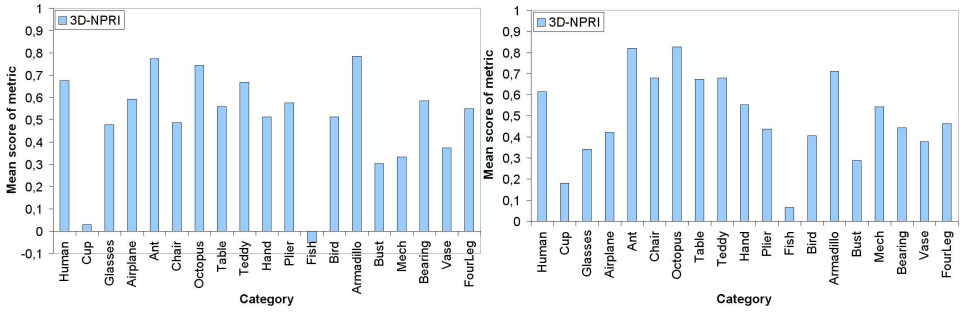
Table 3.3 – *Algorithms ranking applied on respectively the Chen's corpus and our corpus.*

seems to provide the best results. It is very interesting to notice that the Fitting Primitives and Shape Diameter keep similar behavior for the two corpuses although these two corpuses are very different: the profiles of the 3D-NPRI distributions (see figure 3.18) and the mean 3D-NPRI values (see table 3.3) for these algorithms are almost exactly the same for both corpuses. Hence it validates the fact that our corpus, since it presents high quality manual segmentation and heterogeneous models, is efficient for segmentation evaluation despite its small size.

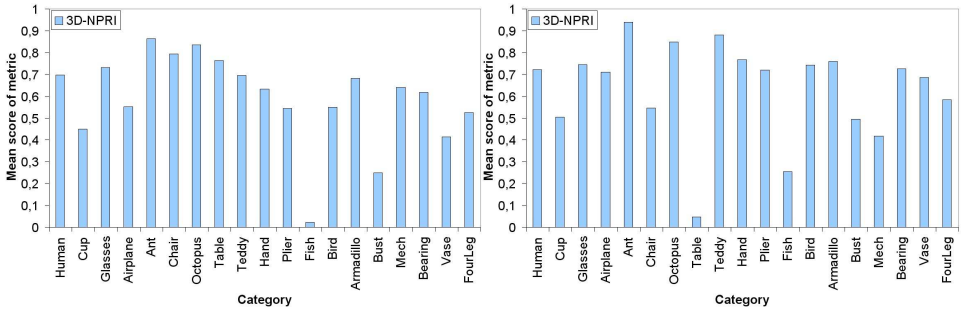
Another interesting experiment is to study which category of models the algorithms fail to segment accurately. To this end, we average the 3D-NPRI for each category of the two corpuses. Figure 3.19 and 3.20 illustrate the results obtained for the six algorithms. One can notice that whatever the corpus, there is no algorithm that reaches the highest scores for all categories. Moreover, each algorithm has at least one category inadequately segmented where the mean 3D-NPRI value is very low (close to 0 or less). The core extraction algorithm for instance fails to adequately segment the *Bearing* and *Mech* categories (see figure 3.19(e)). Indeed, it tries to detect the core of a model which from a semantic point of view is hard to define in such categories. As observed by Chen *et al.* (CGF09), some algorithms which are expected to produce good segmentations for certain categories seem to be not quite efficient. We can notice this behavior on our corpus too. For instance, the algorithm based on Fitting Primitives was expected to produce the best segmentations for the *furniture* category of models

(The components of these models feet very well with *cylinder* and *plan* primitives), but it is not the case.

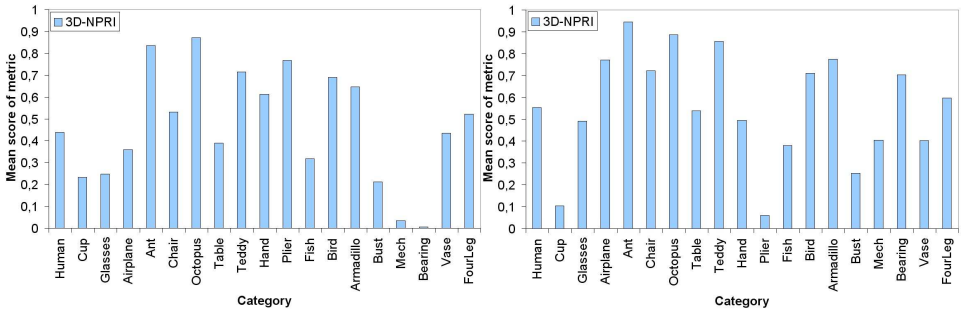
As raised by Chen *et al.* (CGF09), this means that either the human observers do not segment models in the expected way, or the part structures of these models are revealed by other properties.




(a) *Fitting primitives* (b) *Random walks*



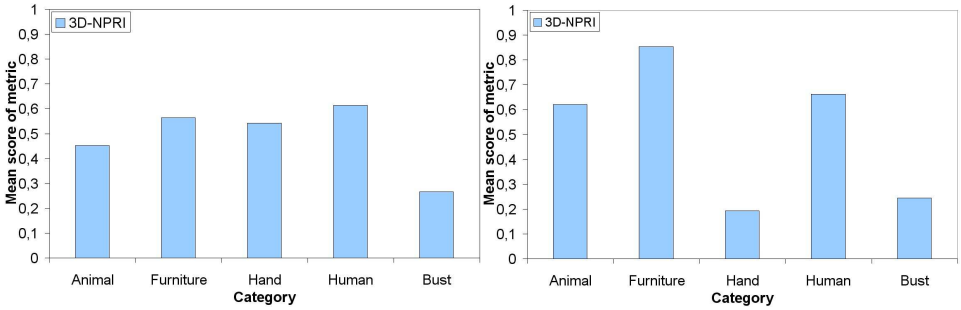

(c) *Normalized cuts* (d) *Randomized cuts*




(e) *Core extraction* (f) *Shape diameter function*

Figure 3.19 – *Scores of 3D-NPRI averaged for each category models of the Chen's corpus.*

According to the experiments conducted in this section, we can conclude that our results and those of Chen *et al.* (CGF09) are coherent.




(a) *Fitting primitives* (b) *Shape diameter function*

Figure 3.20 – *Scores of 3D-NPRI averaged for each category models of our corpus.*

## 3.7 Conclusion

In this chapter, we proposed a benchmark for the quantitative evaluation of 3D-mesh segmentation algorithms. The benchmark is available on-line and includes a ground-truth corpus, allowing an objective evaluation, and a new reliable similarity metric named the 3D Normalized Probabilistic Rand Index (3D-NPRI). We have shown how we built the ground-truth corpus. This corpus is composed of a set of 3D-models associated with several manual segmentations produced by human observers. The new metric (3D-NPRI) is a probabilistic interpretation of the Rand Index (Ran71) which allows to quantify the consistency between multiple segmentations of a 3D-mesh model. Then we presented a thorough comparison between existing similarity metrics and the new one, and shown that this new metric outperforms existing ones in terms of properties. Moreover, to validate the quality of our segmentation ground-truths and the discriminative power of the 3D-NPRI, we proposed a subjective segmentation rating experiment. The protocol has been carefully designed so as to be able to obtain relevant results. The obtained results have attested the efficiency of our benchmark (quality of ground-truth segmentations and discriminative power of the 3D-NPRI metric). Finally we applied the 3D-NPRI together with the Chen's *et al.* (CGF09) corpus and our corpus (the two benchmarks) to evaluate six recent 3D-mesh segmentation algorithms. The evaluation allowed to compare the obtained results depending on the corpus and showed their coherence.

In the following chapter, we present in detail the contributions of this thesis regarding learning 3D-mesh segmentation.

# 4 Learning 3D-mesh segmentation

Contents



This chapter presents a 3D-mesh segmentation algorithm based on a learning approach. A large database of manually segmented 3D-meshes is used to learn a boundary edge function. The function is learned using a classifier which automatically selects from a pool of geometric

features the most relevant ones to detect candidate boundary edges. We propose a processing pipeline that produces smooth closed boundaries using this edge function. This pipeline successively selects a set of candidate boundary contours, closes them and optimizes them using a snake movement. Our algorithm is evaluated quantitatively using our benchmark proposed in the previous chapter and the Princeton benchmark. It shows to outperform most recent segmentation algorithms from the state-of-the-art. The chapter presents also an application of this new segmentation algorithm for kinematic skeleton extraction of dynamic meshes.

## 4.1 Motivation

As we have shown in chapter 2, a significant attention has been paid, by the computer graphics community, to 3D-mesh segmentation. Basically, the segmentation methods proposed so far in the literature focus on analyzing either low level geometric information, or topological information of the input mesh. For instance, the use of geometric criterion includes curvature (LDB05b), geodesic distances (KT03), dihedral angles (ZTS02), shape diameter function (SSCO08), planarity and normal directions (AFS06), etc. The use of topological criterion includes mainly Reeb-graphs (TVD07) and spectral analysis (LZ07). Such criteria suffer either from sensitivity to local surface features and to pose changes or from the deterioration of the topology when connecting or disconnecting parts of the mesh. Moreover, as raised by Kalogerakis *et al.* (KHS10), the main drawback of this kind of algorithms is the fact that they are limited to a single generic rule (e.g. skeleton topology) or a single feature (e.g. curvature tensor). Indeed, such algorithms cannot be suited to segment an input 3D-mesh which requires a combination of these criteria.

On the other side, *learning* for mesh segmentation has become possible thanks to the recent creation of ground-truth databases like the one we presented in the previous chapter. In other words, the ground-truths (man-made segmentations) allow to understand how do people decompose (or segment) 3D-objects, and thus make possible to learn a model that would generates similar segmentations to those created by humans. A recent work proposed by Kalogerakis *et al.* (KHS10), has demonstrated the efficiency of learning for segmentation through the improvement of the results over the state-of-the-art of mesh segmentation.

For all these reasons, we propose a new fully automatic 3D-mesh segmentation algorithm based on boundary edge *learning*. Human perception theory (HS97) says that to recognize a shape, the human visual system decomposes it into its significant parts. The decomposition is carried through the definition of each part boundaries by means of general com-

putational rules such as the minima rule[1]. This observation is our primary motivation for basing our learning algorithm on edges rather than on regions. Our algorithm is carried out using two main steps: an off-line step in which an objective boundary edge function is learned from a set of segmented training meshes, and an on-line step in which the learned function is used to segment any input 3D-mesh. The boundary function is learned using the AdaBoost classifier (FS97), which automatically selects from a set of geometric features the most relevant ones to detect candidate boundary edges. In the on-line step, the learned edge function is used successively to select a set of candidate boundary contours, to close them and to optimize them using a snake movement to produce the final segmentation. The best results are obtained when the learning is performed on objects from the same category as the object to segment (see figure 4.1). However, the results remain excellent even when we generalize the learning on different categories. Hence, we do not need to know the category of the input model to segment it.

## 4.2 Related work

According to our knowledge, only one work has been proposed that involves learning for 3D-mesh segmentation (KHS10). It allows to simultaneously segment and label the input mesh, and is expressed as an optimization problem. As described in chapter 2, the problem consists in optimizing a Conditional Random Field (CRF) of which an objective function is learned from a collection of labeled training meshes. We differ from this latter work in that instead of determining the suited label of each mesh facet and then implicitly defining a segmentation resulting from this labeling, we explicitly determine the boundary edges that allow then to obtain smooth closed contours that define the segmentation. Moreover, even complex boundaries can be captured (see section 4.3), while in the previous work, the method rather aims to find compact regions.

Before this recent work for 3D-mesh segmentation, several advanced

[1]The minima rule states that human vision defines part boundaries along negative minima of the principle curvatures on surfaces (HR84).

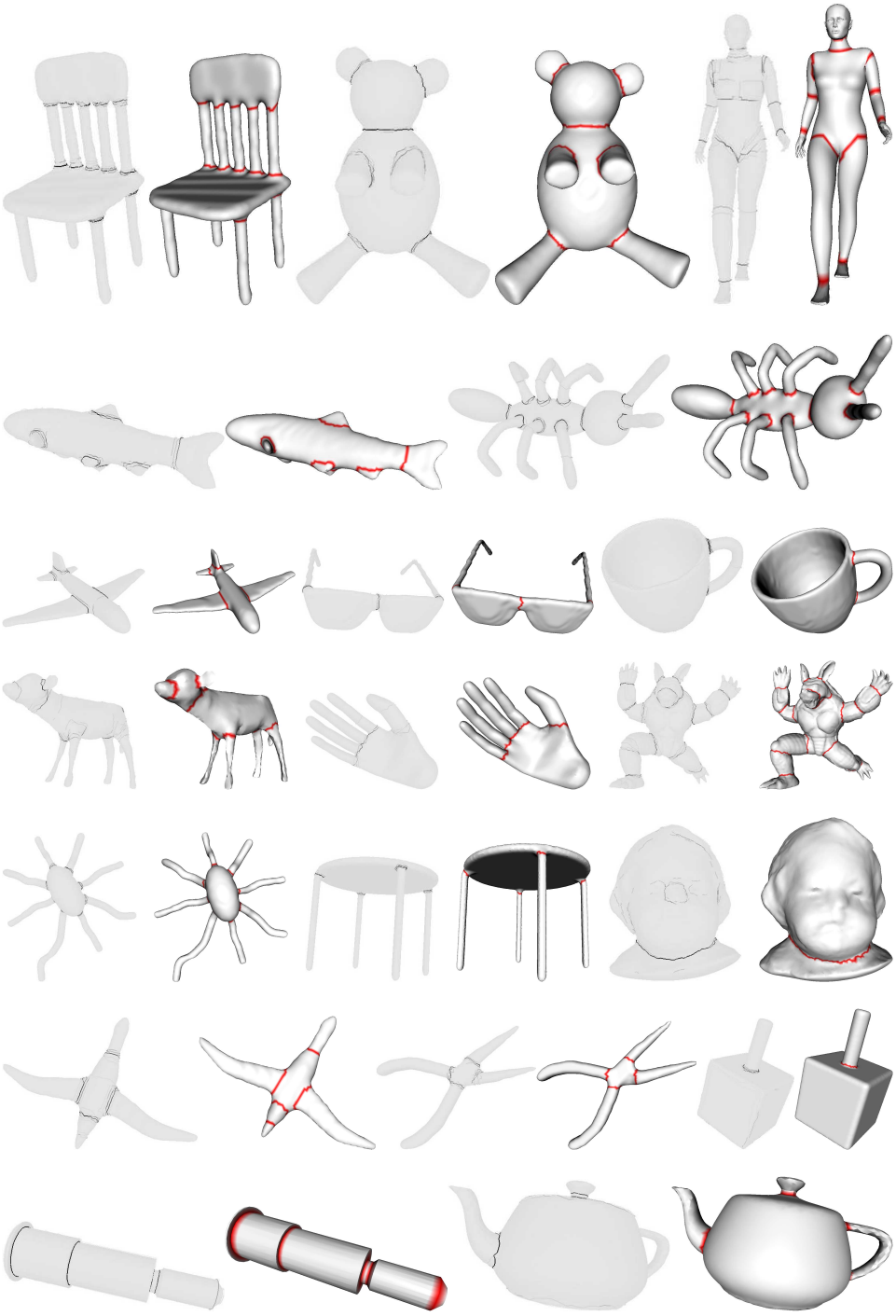


Figure 4.1 – *For each pair of model: on the left, manual boundaries from the Princeton segmentation benchmark (CGF09) (the darkness degree of contours indicates that people have selected the same edges in their cuts); on the right, automatic boundaries from our algorithm.*

works have already been introduced for 2D-image segmentation based on learning approaches. Like for the 3D case, these algorithms use a model learned from a database of segmented 2D-images. These 2D-image segmentation algorithms based on learning can be grouped into two categories.

The first category covers algorithms that learn an optimal affinity function between each pixel of the input image and a set of prior defined labels (SWRC09, HZCp04). A ground-truth (segmented and labeled images) is employed to train the classifier that allows to affect the proper label to each pixel.

The second category covers algorithms that use an objective function to classify edges (MFM04, KH04). Each edge is classified as a boundary or a non-boundary using a classifier trained on the ground truth (segmented images), resulting in an edge image estimating human designated boundaries.

Our algorithm is inspired by the second category since it classifies edges as boundary or not, while the previous 3D-work (KHS10) is inspired by the first category.

## 4.3 Our segmentation algorithm

In this section, we describe our approach. We provide details on the two main steps of our algorithm: the off-line step in which the objective boundary function is learned using a set of segmented models, and the on-line step in which the learned function is used to segment the input mesh.

### 4.3.1 Off-line (learning) step

We formulate the problem of learning the boundary edges as a classification problem. The classification model is learned on a corpus of 3D-meshes accompanied by their manual segmentations using the AdaBoost classifier. The classifier takes as input a training data set and generates a function. The training data set is composed of a set of feature vectors $F_E$ computed for each edge of the ground-truth corpus meshes. A feature

vector $F_E$ of a given edge contains a set of geometric criteria and is associated with its proper class label $L$ so that $L = +1$ if the edge is a boundary (according to the manual segmentations of the mesh containing this edge) and $L = -1$ if the edge is not a boundary. Figure 4.2 illustrates the off-line step. Once the learning is done, the classifier produces a function (the boundary edge function). This function takes as input a feature vector from any given edge and outputs a signed scalar value whose sign will provide the estimated classification of the edge (positive for boundary and negative for non-boundary).

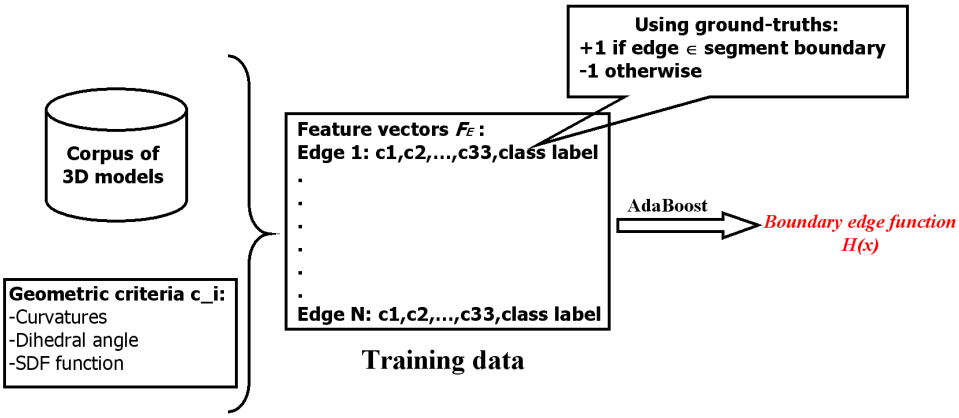



Figure 4.2 – *Overview of the off-line step.*

Now we summarize, the set of geometric criteria that we use to characterize edges (and which compose the feature vector), and the AdaBoost classifier.

### Feature vector

We compute a 33 dimensional feature vector for each edge. It includes a set of geometric criteria which are as follows:

- Dihedral Angle. The angle between two adjacent facets.
- Curvature. We compute different curvatures using the VTK library[2]. Let $K_1$, $K_2$ be the principal curvatures, we include: $K_1$, $K_2$, $K_1 \times K_2$ (Gaussian curvature), $(K_1 + K_2)/2$ (Mean curvature),

[2] http://www.vtk.org/

$2/\pi * \arctan[(K_1+K_2)/(K_1-K_2)]$ (Shape index), and $\sqrt{(K_1^2+K_2^2)/2}$ (Curvedness).

- Global curvature. We compute the mean curvature integrated over a large geodesic radius (10% of the bounding box lentgh) as in (LW08).
- Shape diameter. The shape diameter function (SSCO08) is computed using the default parameters: a cone with an opening angle of 120°, 30 rays, and a normalizing parameter $\alpha = 4$.

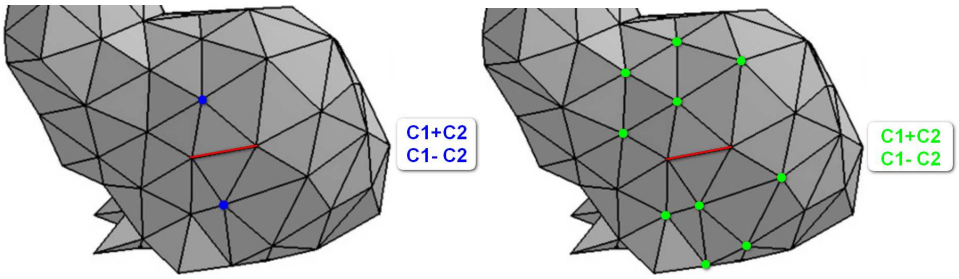
Note that we do not propose new geometric criteria, but we only employ existing ones used in previous segmentation algorithms. As stated in the introduction, the idea is to combine these criteria, then automatically select relevant ones with the appropriate weights using the classifier.

Except the dihedral angle which is computed for each edge, the other criteria (8 criteria) are computed for each vertex of the mesh. As illustrated in figure 4.3, to derive the criteria for each edge (the red one in the figure), we consider its two opposite vertices (blue points in figure 4.3(a)). Then, considering that $C_1$ and $C_2$ are the values of a certain criterion computed respectively on these two vertices, we derive two feature values for the edge: $C_1 + C_2$ and $C_1 - C_2$. The idea is that, according to the nature of the criterion, in certain cases the sum can be relevant while in others the difference can carry a better information.

In order to bring a certain robustness to noise or sampling and to integrate a kind of multi-resolution behavior we also consider, in a second step, the 1-level neighborhood from each side of the edge (see green points in figure 4.3(b)). In that case $C_1$ and $C_2$ are respectively the means of the criterion from vertices at each side of the edge. This yields 32 features (16 in each case) to which we add the dihedral angle feature.

**AdaBoost classifier**

AdaBoost is a machine-learning method that builds a strong classifier by combining weaker ones (FS97). The algorithm takes as input a set of training examples $(x_1, y_1), .., (x_N, y_N)$; each $x_i$ $(i = 1, ..., N)$ represents a feature vector (a vector $F_E$ in our case), and each $y_i$ represents the class label of the




(a) (b)

Figure 4.3 – *Example of edge criterion computation with one vertex on each side (a), and with a set of vertices (b).*

example $x_i$ ($y_i$ belongs to the domain $Y = \{-1, +1\}$ in our case). A large number of hypothesis (or classifiers), $h_i : X \longrightarrow Y$, are generated, each one computed by a decision stump associated with a single dimension of the feature vector. Then, along a finite number of iterations $(t = 1, 2, ..., T)$, the algorithm iteratively selects the hypothesis which minimizes the current classification error. At the end a strong classifier $H$ is produced as the combination of the hypotheses weighted with $\alpha_t$: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

As stated at the beginning of subsection 4.3.1, the generated function $H(x)$ is now able to produce a signed scalar for each edge of an input 3D-mesh, whose sign gives its class label (positive sign for boundary edges and negative sign otherwise).

Note that we tested some other existing classifiers from the literature including non-parametric and parametric models such as Density estimation, HME (Hierarchical Mixtures of Experts), and SVM (Support Vector Machine). The performance was always nearly the same. We favor the AdaBoost since it yields a slight improvement over the other classifiers, and has the best running time.

### 4.3.2 On-line (segmentation) step

Figure 4.4 shows examples of edge classification results of some 3D-meshes. On the top, the result of the binary decision: boundary ($H(x) > 0$), non-boundary ($H(x) < 0$) is displayed; while in the bottom, the edge function scalar values $H(x)$ are displayed using a color map (for visualiza-

tion quality reason, we colored incident vertices of edges instead of coloring the edges themselves). One can notice that the boundary edges from the binary decision (in red) are, neither smooth, nor closed. This result is expected since our classification model is learned on different objects (even if they belong to the same category), and uses multiple ground-truths per model which are not necessarily the same (boundaries are defined in a subjective way by humans, see figure 4.1). Hence it is not possible to directly consider this classifier output as a segmentation result.

To overcome this problem we propose a processing pipeline that transforms these non-connected fuzzy regions into thin, closed and smooth contours, by using the edge function. This processing pipeline comprises four stages.

In the first stage of the process, given an input 3D-mesh, the edge function is computed (figure 4.5(a)), and all edges having positive function values are selected. Theses edges constitute a set of interest regions (figure 4.5(b)). Then, for each interest region (connected set of edges), a thinning algorithm (HG01) is applied. This latter algorithm gives as output a set of open linear contours (figure 4.5(c)). Next, each open contour is completed using an improved version of the algorithm proposed by Lee *et al.* (LLS*05) based on the edge function (figure 4.5(d)). At this step we have created a set of closed contours which represent a first version of the segmentation boundaries. However, these boundaries are often not smooth nor precise since in the thinning stage we do not consider any geometric information. To overcome this drawback, we apply an improved version of the snake movement algorithm proposed by Jung and Kim (JK04) based also on the learned edge function. The snake movement allows to improve the quality of the boundaries in term of smoothness and precision without changing the mesh connectivity (figure 4.5(e)). This set of improved boundaries defines the final segmentation (figure 4.5(f)). These steps are detailed in the following subsections.

Note that in all our experiments (more than 350 models), we never encountered any topological problem (e.g., broken regions representing the same boundary) like in the work from Lee *et al.* (LLS*05). The main reason

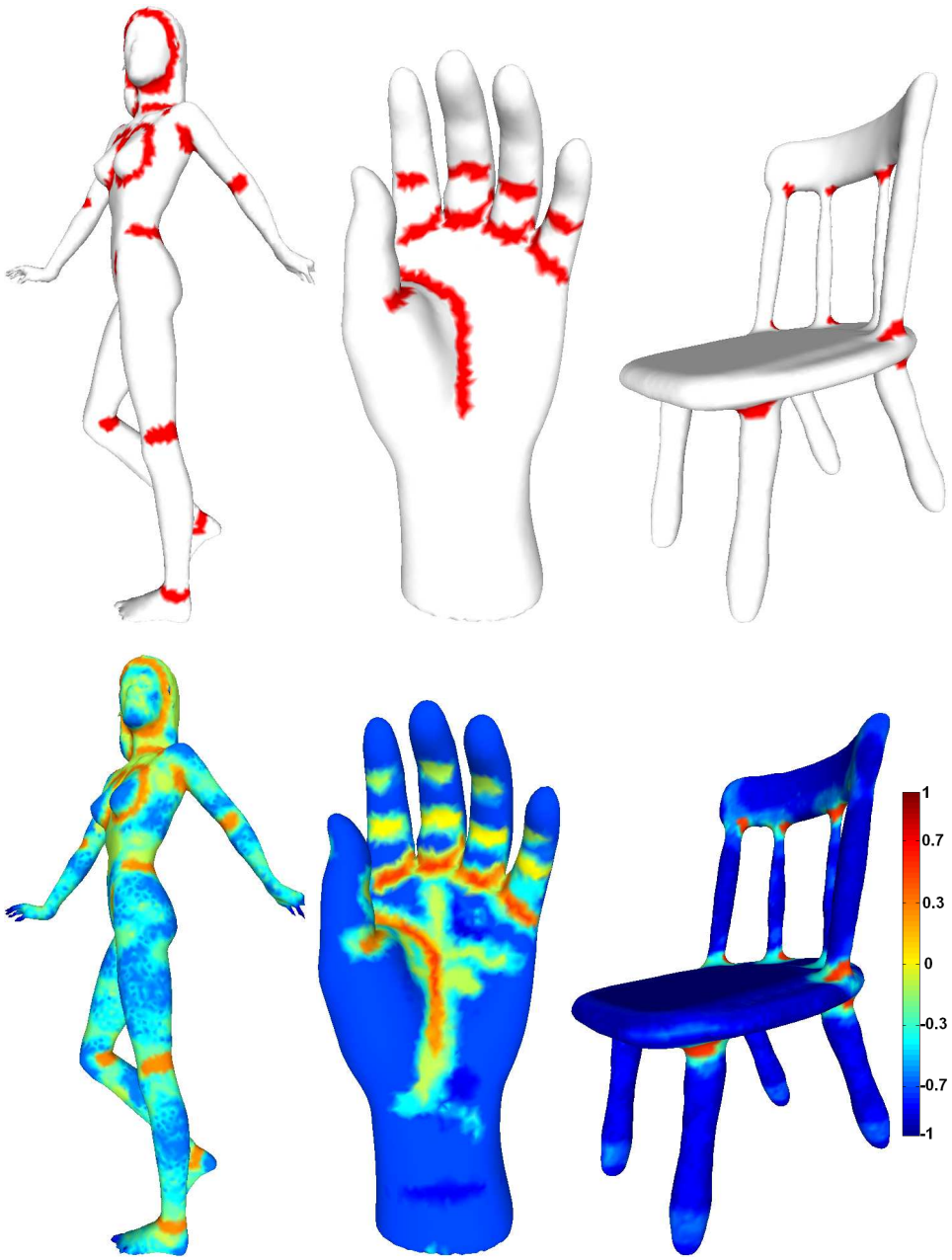
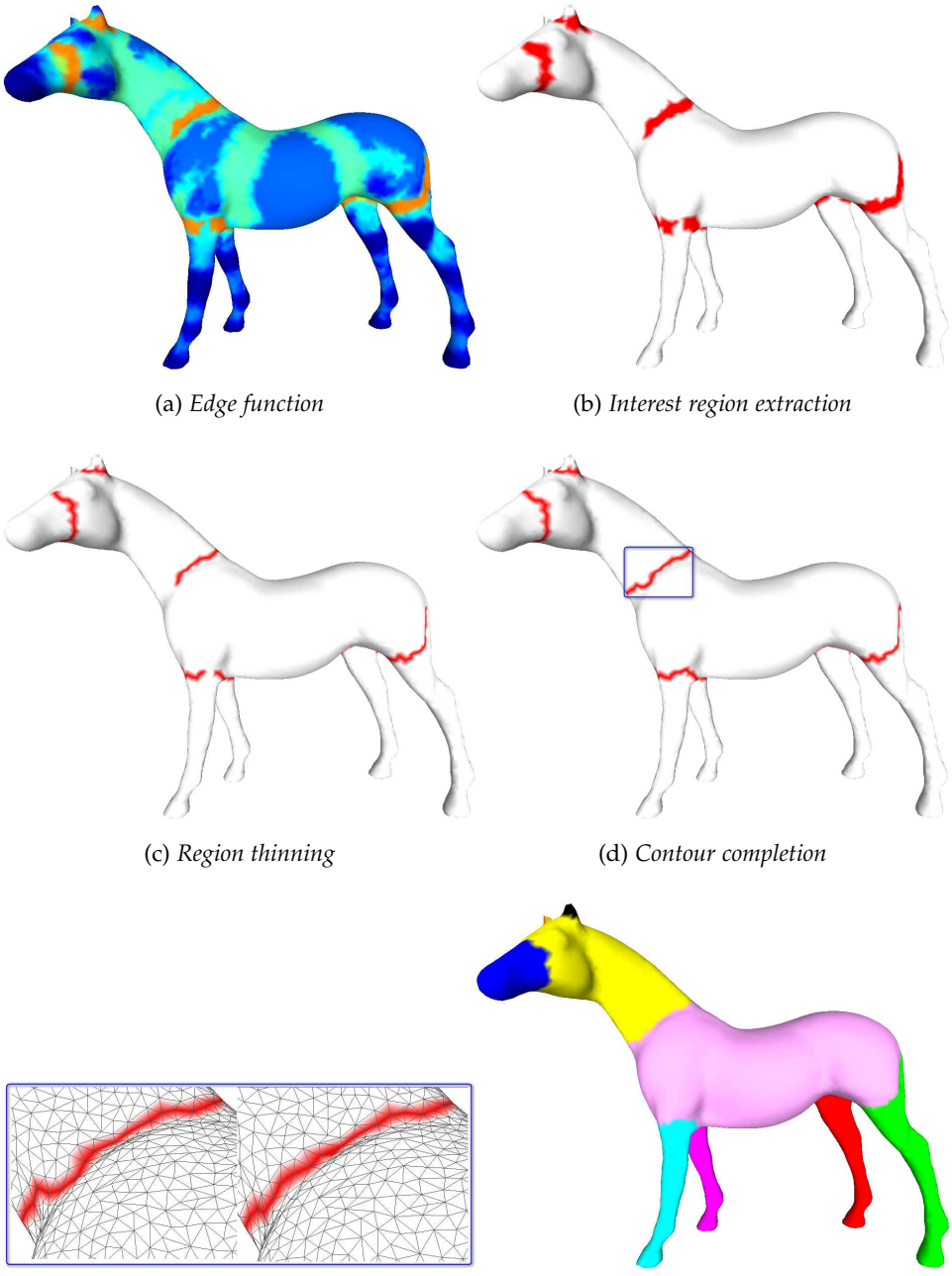



Figure 4.4 – *Edge classification results for some 3D-meshes; (top: boundary edges after binary decision in red color; bottom: edge function scalar field).*



(a) *Edge function* (b) *Interest region extraction*

(c) *Region thinning* (d) *Contour completion*

(e) *Snake movement* (f) *Final segmentation*

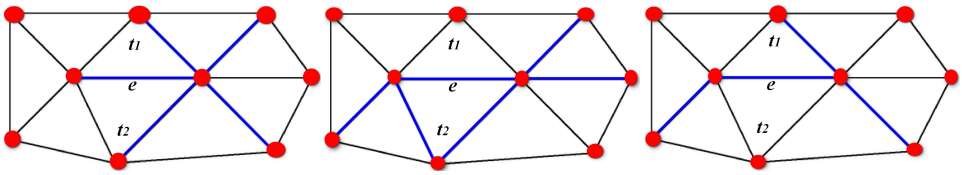
Figure 4.5 – *Overview of the processing pipeline.*

is probably that our regions of interest come from a learning processed on different models associated with different manual segmentations; hence it introduces a kind of fuzziness which smooth/filter the results. Thus, we create one closed boundary for each connected region, even if two regions are close to each other. We do not process any contour filtering or contour merging.

**Region thinning**

In this stage, we transform a set of interest regions into a set of thin contours; this set of contours will be further used as the initial set of boundaries. Each interest region is represented by a set of connected edges. The algorithm from Hubeli and Gross (HG01) allows to thin a given interest region to a piecewise linear contour by deleting the edges from the border of the patch toward the inside. Initially, the algorithm inserts all border edges into a list. A border edge is an edge of which at least one of the four edges of its two opposite triangles does not belong to the interest region. Then, each border edge is deleted from the interest region if it does not disconnect this latter one. More precisely, a border edge $e$ is deleted if it satisfies one of the two following conditions:

1. All the incident edges of one of the two end points of $e$ does not belong to the interest region (see figure 4.6(a)).
2. The edges of one of the two opposite triangles of $e$ belong to the interest region (see figure 4.6(b)).




(a) *e can be deleted* (b) *e can be deleted* (c) *e cannot be deleted*

Figure 4.6 – *Deleting a border edge e from the interest region (set of connected blue edges).*

Otherwise the edge $e$ is not deleted as illustrated in figure 4.6(c). The deleting operation produces new border edges which are added to the list.

The algorithm is performed until there is no removable edge. This leads to produce a connected skeleton of the interest region. Moreover, the algorithm allows to obtain directly a closed contour if the interest region forms a loop. Note that this algorithm does not contain any parameter setting. The different steps of the region thinning are summarized in algorithm 5.

**Algorithm 5**: Region Thinning

***Input:*** *a list L containing the interest region edges*
***Initialize*** *an empty border list BL*
**for all** *edges in L* **do**
  *e = L.next();*
  **if** *e is a border edge* **then**
    *BL.insert(e);*
  **end if**
**end for**
**while** *BL.notempty()* **do**
  *e = BL.next();*
  *BL.remove(e);*
  **if** *e can be deleted from L* **then**
    *L.delete(e);*
    *BL.insertNewBorderEdges();*
  **end if**
**end while**
***Output*** *updated list L, organized under the form of an open contour*

However, it is possible to obtain a branching skeleton. Figure 4.7 illustrates an example in which a model has an interest region that leads to the creation of a branching skeleton after undergoing a thinning algorithm. This skeleton is composed of external and internal branches. An external branch is limited by one endpoint and one junction point while an internal branch is limited by two junction points. For a given branching skeleton coming from the thinning of an interest region, we consider that only two external branches are correct regarding the real boundary and we consider others like noise; to select these two relevant branches we compute a weight for each external branch by summing the learned

function values of their edges, and we keep the two branches that have the highest weights together with the internal branches that connect them (in the case where they do not share the same junction point). We precise here that according to our experiments, such branching skeletons appear mostly when the corresponding interest region is very large, which almost never happens.

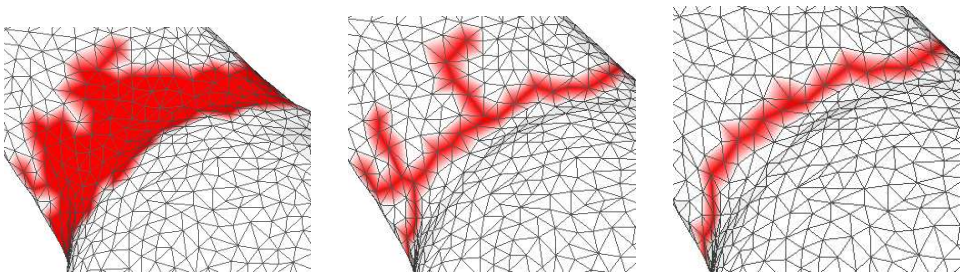


Figure 4.7 – *From left to right: the interest region, the branching skeleton after thinning, the open boundary after removing the noisy branches for the horse model.*

**Contour Completion**

In this stage, each open contour is completed to form a closed boundary around a specific part of the input mesh. To this end, we propose a modified version of the completion algorithm from Lee *et al.* (LLS*05). The principle is to find the weighted shortest path between the two endpoints of the contour.

Let $\zeta$ be an open contour composed of a set of mesh vertices $v$. To close the open contour, we search the shortest path between the two end points of $\zeta$ by selecting among candidate edges using the following edge cost function:

$$cost(e) = \eta_d(e)^{w_d}.\eta_n(e)^{w_n}.\eta_e(e)^{w_e}$$

where $\eta_d(e)$ and $\eta_n(e)$ are defined as the average of the values of the two incident vertices of the edge $e$.

$\eta_d$ is the distance function. It measures the distance from $\zeta$ to a given vertex $v$ as:

$$\eta_d(v) = \sum_{v_i \in \zeta} \frac{1}{d(v, v_i)}$$

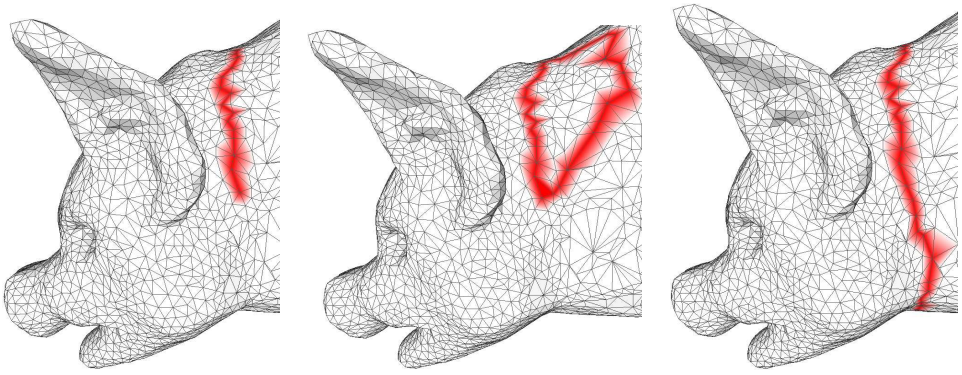
where $d(v, v_i)$ is the Euclidean distance. The function is high in the neighborhood of the contour $\zeta$ and decreases otherwise.

$\eta_n$ is the normal function. It helps to go over the other side of the mesh, and is defined for a given vertex $v$ as:

$$\eta_n(v) = \begin{cases} 1, & \text{if } n_\zeta . n_v \geq \cos(\alpha) \\ \frac{n_\zeta . n_v + 1}{\cos(\alpha) + 1}, & \text{else} \end{cases}$$

$n_\zeta$ is the average normal vector of $\zeta$, $n_v$ is the normal vector of vertex $v$, and $\alpha$ is the angle between the normals of the two endpoints of $\zeta$.

$\eta_e$ is the feature function; in the original work from Lee *et al.* (LLS*05) the feature function includes minimum curvature and centricity. The centricity of a vertex is defined as the average geodesic distance from the given vertex to all other vertices of the mesh. As stated by the authors, the original algorithm sometimes failed to correctly close the open contours. In our modified version, we replace the feature function by our learned edge function; it guides the path towards the regions learned as boundaries according to the results of the classifier. The results are significantly improved (see an example in figure 4.8).



(a) (b) (c)

Figure 4.8 – *Example of completing a contour on a 3D-mesh (a) using: the original version of the algorithm from Lee* et al. *(LLS*05) based on their feature function (b), and the improved version based on our learned edge function (c).*

Note that the three function values are normalized in the range $[0, 1]$. We set the weights $w_d$,$w_n$ to 1 and $w_e$ to 0.4.

**Snake movement**

The snake movement is used to optimize the set of closed contours resulting from the previous stage. Each contour is used as the initial position of the snake. We propose a modified version of the snake movement algorithm from Jung *et al.* (JK04). The algorithm is based on an iterative process in which the snake evolves (each vertex of the snake is moved to one of its neighbor vertices on the mesh) by minimizing an energy functional $E$ composed of internal $E_{int}$ and external $E_{ext}$ parts until it is adjusted (see algorithm 6).

**Algorithm 6**: One iteration of the snake movement

1: ***Input:*** *a list L containing the vertices of the closed contour*
2: **for** *each vertex* $v_i$ *in L* **do**
3: $E_{min} = E_{int}(v_i) + E_{ext}(v_i)$
4: **for** *each neighbor* $v_j$ *of* $v_i$ **do**
5: $E = E_{int}(v_j) + E_{ext}(v_j)$
6: **if** $E \leq E_{min}$ **then**
7: $E_{min} = E$
8: *Replace,in L,* $v_i$ *by* $v_j$
9: **end if**
10: **end for**
11: **end for**
12: ***Output*** *updated list L*

In the original work from Jung *et al.* (JK04), the internal energy controls the length and the smoothness of the snake (i.e. the closed contour), and is defined for a given vertex $v_i$ as:

$$E_{int}(v_i) = \alpha \left\| v_i - v_{i-1} \right\| + \beta \left\| v_{i+1} - 2v_i + v_{i-1} \right\|$$

where $\alpha$ and $\beta$ are tuning parameters that affect respectively the smoothness of the snake in term of distance and curvature. We set the $\alpha$ to 0.2 and the $\beta$ to 0.8. The external energy controls the fitting of the snake to the desired feature, and is defined in (JK04) by the maximum principal curvature. In our modified version, we replace the maximum

curvature by the learned edge function again. The external energy of a given vertex is then computed by averaging the edge function values of its incident edges, and normalizing them in the range $[0, 1]$ after reversing the sign since we aim to minimize the energy. The modification of the external energy is justified by the fact that the original algorithm aims to find features related to ridges and valleys based only on a single geometric criterion (maximum curvature). Hence, when replacing the maximum curvature by the edge function, the quality of boundaries is clearly improved (see section 4.4.4), since this latter function is based on multiple geometric criteria which are learned to detect boundaries.

## 4.4 Experiments and results

Our segmentation method was evaluated and compared quantitatively to the most recent algorithms from the state-of-the-art. To this end, we used two recent benchmarks dedicated to 3D-mesh segmentation evaluation, namely the Princeton benchmark[3] (CGF09), and our benchmark which is presented in the previous chapter.

Note that we used the same control parameter values in all our experiments, except when we explicitly modify the threshold of the $H$ function for the hierarchical segmentation experiment (see section 4.4.5). The different parameters are set as follows:

- $H$ Function threshold: $H(x) > 0$.
- Thinning and branch-filtering: no parameter.
- Contour completion: $w_d$,$w_n$ to 1 and $w_e$ to 0.4 of $cost(e)$.
- Snake: $\alpha$ to 0.2 and $\beta$ to 0.8 of $E_{int}$.

### 4.4.1 Segmentation results on the Princeton benchmark

The Princeton segmentation benchmark provides 19 categories of 3D-meshes, each one containing 20 3D-models accompanied with multiple ground-truth segmentations (manual segmentations). Our segmentation

[3] http://segeval.cs.princeton.edu/

method was trained and tested on this benchmark, using different learning strategies, namely categorical learning and global learning.

In the first type of learning (categorical), we train and test our algorithm class by class. Similarly to Kalogerakis *et al.* (KHS10), we evaluate our method using leave-one-out cross-validation. For each mesh $i$ in each class, we learn the edge function on the 19 other meshes from that class using all ground-truths, and then we use that function to segment the mesh $i$. In order to analyze the effect of the training set size on the quality of the results, we repeat the same experiment, using less meshes in the training set: we learned the edge function on 6 meshes randomly selected for each class.

In the second type of learning (global), we learn the edge function in a generic way using a subset of the whole database (6 models randomly selected from each category), then we test on the remaining models ($14 \times 19$ models). In this generic (or global) learning scenario, we do not need to know the category of the model to segment.

To evaluate the quality of the segmentation produced by our algorithm, we follow the protocol defined in the Princeton segmentation benchmark. Figure 4.9 shows the Rand Index error averaged over all models of the corpus for our algorithm, using the different learning strategies, and for the most recent algorithms from the state-of-the-art.

The first point to make is that, when using a categorical learning with a training set size of 19 models, our algorithm provides very high quality results; indeed, our algorithm yields the smallest Rand Index error (8.8%) among all the other algorithms. One can also notice on the figure that when reducing the training set size to 6 models, and keeping a categorical learning, our algorithm still provides very good results with a slight drop of performance (9.7% Rand Index Error). The second point to make is that, our algorithm performs better than the algorithm from Kalogerakis *et al.* (KHS10) which is also based on learning (9.5% and 12.2% Rand Index Errors with respectivly a training set size of 19 and 6 models).

However, categorical learning involves the fact that before segmenting a model, you have to know its category hence it is not really fair to

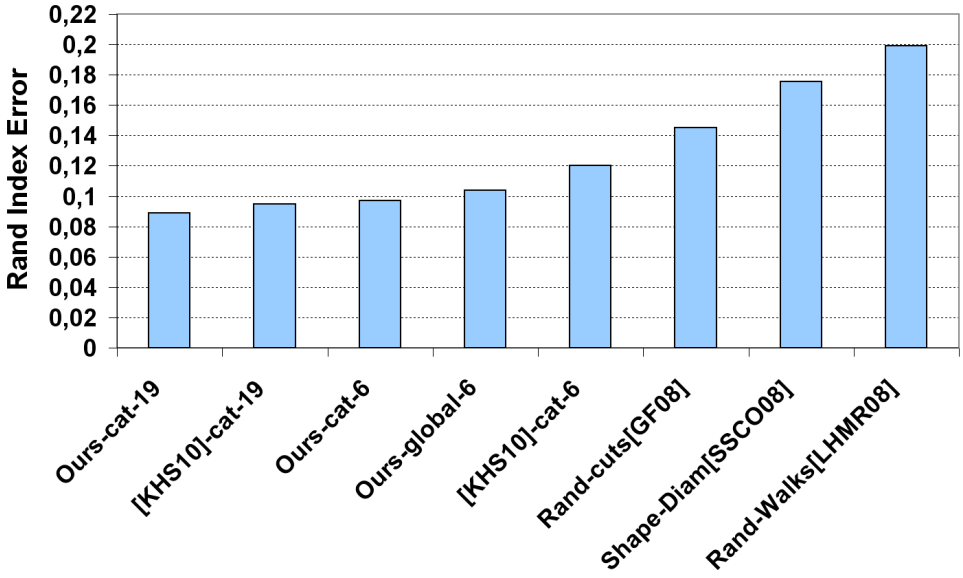



Figure 4.9 – *Rand Index Error averaged over all the Princeton corpus models and sorted in increasing order for different algorithms. Reference-type-size represent the Index Error of algorithms based on learning with: learning type (categorical or global), size of the used training set (19 and 6 models).*

compare these results with generic algorithms (GF08, SSCO08, LHMR08, AFS06) which do not need such knowledge of the data. When using Global learning, our algorithm does not need this prior knowledge and thus can be compared with generic algorithms; the fact is that it significantly outperforms them: its Rand Index Error is 10.4% while the value of the second best (GF08) is 14%.

It is interesting to study which criteria are selected by the classifier in the learning step. Figure 4.10 illustrates the percentage of each criterion selected by the AdaBoost classifier for both types of learning (categorical for several categories such as *bust*, *human*, etc. and global). We can notice that, whatever the learning strategy (categorical or global), all the criteria are selected by the classifier, hence they all contribute to the results. However, the distribution of the criteria percentages differs from a category to another. For instance, in the *fourleg* category, the most used criteria are the shape diameter and minimum curvature, while in the *table* category, the most used are the dihedral angle and maximum curvature. One can notice also a more isotropic distribution between the different criteria in the global learning with comparison to the categorical learning. This is

due to the variety of 3D-meshes included in the training. All categories together may have different shapes and topologies, and then they do not share necessarily the same important features. In this case one or two criteria are clearly not sufficient to obtain a correct segmentation.

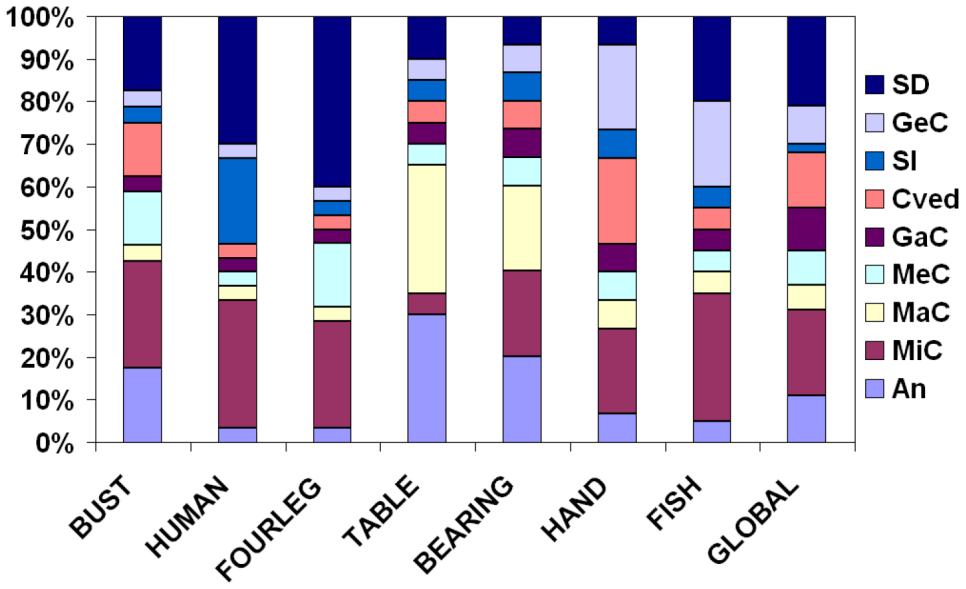



Figure 4.10 – *Percentage of criteria selected by AdaBoost: for a categorical learning of size 19, and for a global learning of size 6. Legend: An (Angle), MiC (Minimum Curvature), MaC (Maximum Curvature), MeC (Mean Curvature), GaC (Gaussian Curvature), Cved (Curvedness), SI (Shape Index), GeC (Geodesic Curvature), SD (Shape Diameter).*

Figure 4.11 shows a visual comparison between our segmentation results (categorical learning - 19 training models) and those from recent algorithms from the state-of-the-art on some 3D-meshes from the Princeton benchmark; the average of manual segmentations (ground-truths) is also included. The quality of our algorithm is confirmed; indeed, our segmentations appear better than those of the other methods in term of similarity to the ground-truths, and particularly regarding boundary precision.

### 4.4.2 Genericity of the learning across databases

In a second experiment, we still have trained our edge function on the Princeton benchmark but we have launched the segmentation on our benchmark which contains a different set of 3D-models. Besides, the 3D-models are associated with vertex-based manual segmentations, while

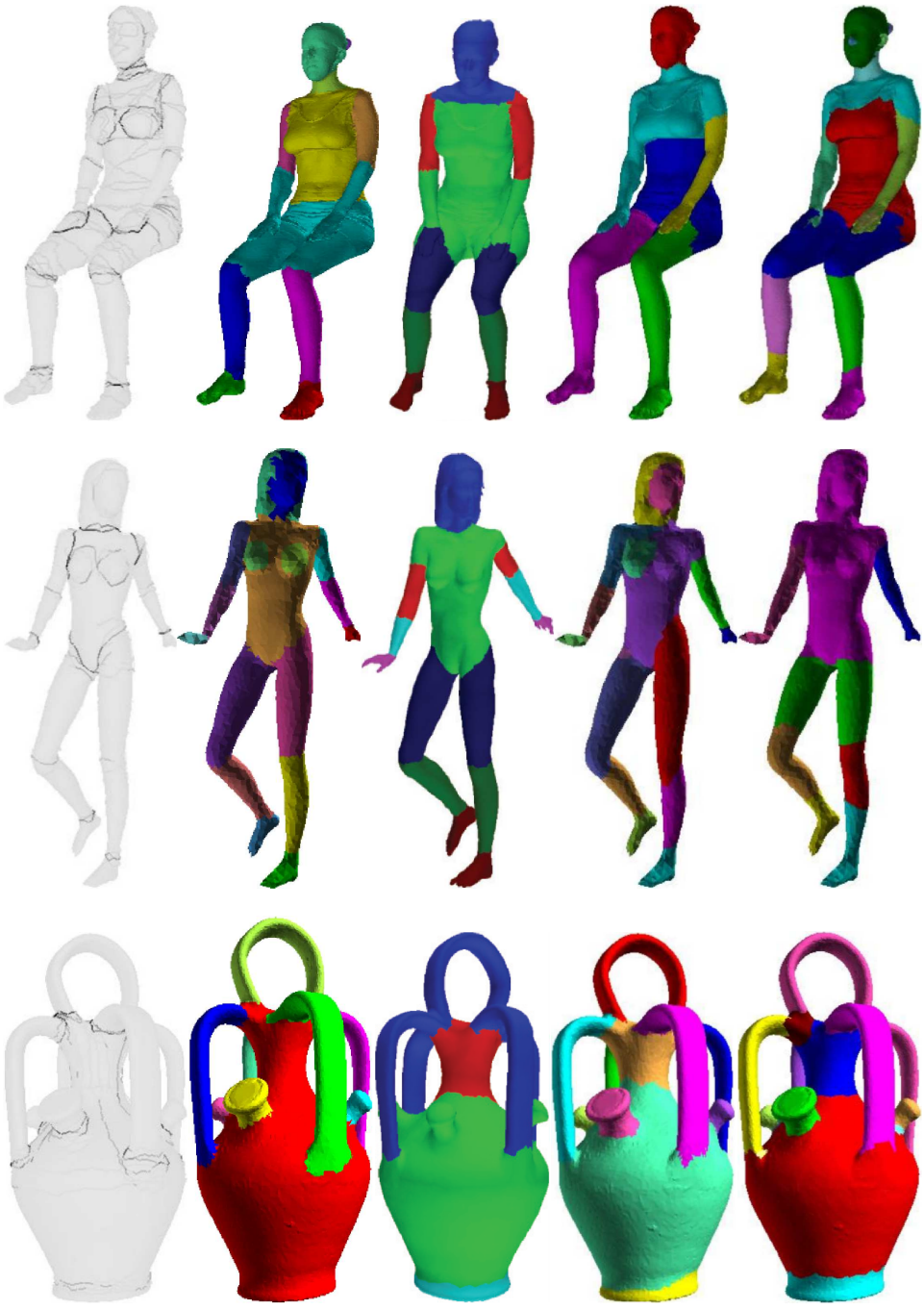


Figure 4.11 – *From left to right segmentations obtained by: average of ground-truths of Princeton benchmark, our algorithm trained on the Princeton benchmark, (KHS10), (GF08), (SSCO08).*

those of the Princeton benchmark are associated with facet-based segmentations. We remind that our benchmark contains 28 3D-models grouped in five classes. Each 3D-model is associated with 4 ground-truths. Similarly to the previous experiment, to evaluate the segmentation produced by our algorithm, we have followed the protocol defined in the benchmark. The edge function used to segment the models of this benchmark was learned on the Princeton segmentation benchmark using the global learning, with a training set size of 6 models.

Figure 4.12 shows the NPRI (Normalized Probabilistic Rand index) scores, computed for each model of the corpus (on the top) and averaged over all the corpus (on the bottom), for different algorithms including ours. Contrary to the Rand Index Error, the NPRI gives an indication about the similarity degree between the automatic segmentation and the manual segmentations. It is in the range [-1,1]. As described in the previous chapter, a value of 1 indicates a complete similarity, whereas a value of $-1$ indicates a maximum deviation between the segmentations being compared. The figure clearly shows the significant improvement of the results obtained by our method with comparison to the others. More precisely, our method reaches 65% of similarity rate, when the best result reached by the other methods on the same corpus is 55%. We have to precise here that these good results confirm the robustness and the genericity of our learning since we have trained our edge function on a different database containing different models.

Figure 4.13 shows segmentations obtained by our algorithm for some 3D-meshes selected from INRIA[4], TOSCA[5], and Stanford[6] databases. The edge function used to segment the models was learned on the Princeton segmentation benchmark (global learning, 6 models). Again, our algorithm correctly segments these meshes, and finds a set of meaningful parts.

[4] http://www-roc.inria.fr/gamma/

[5] http://tosca.cs.technion.ac.il/

[6] http://graphics.stanford.edu/data/

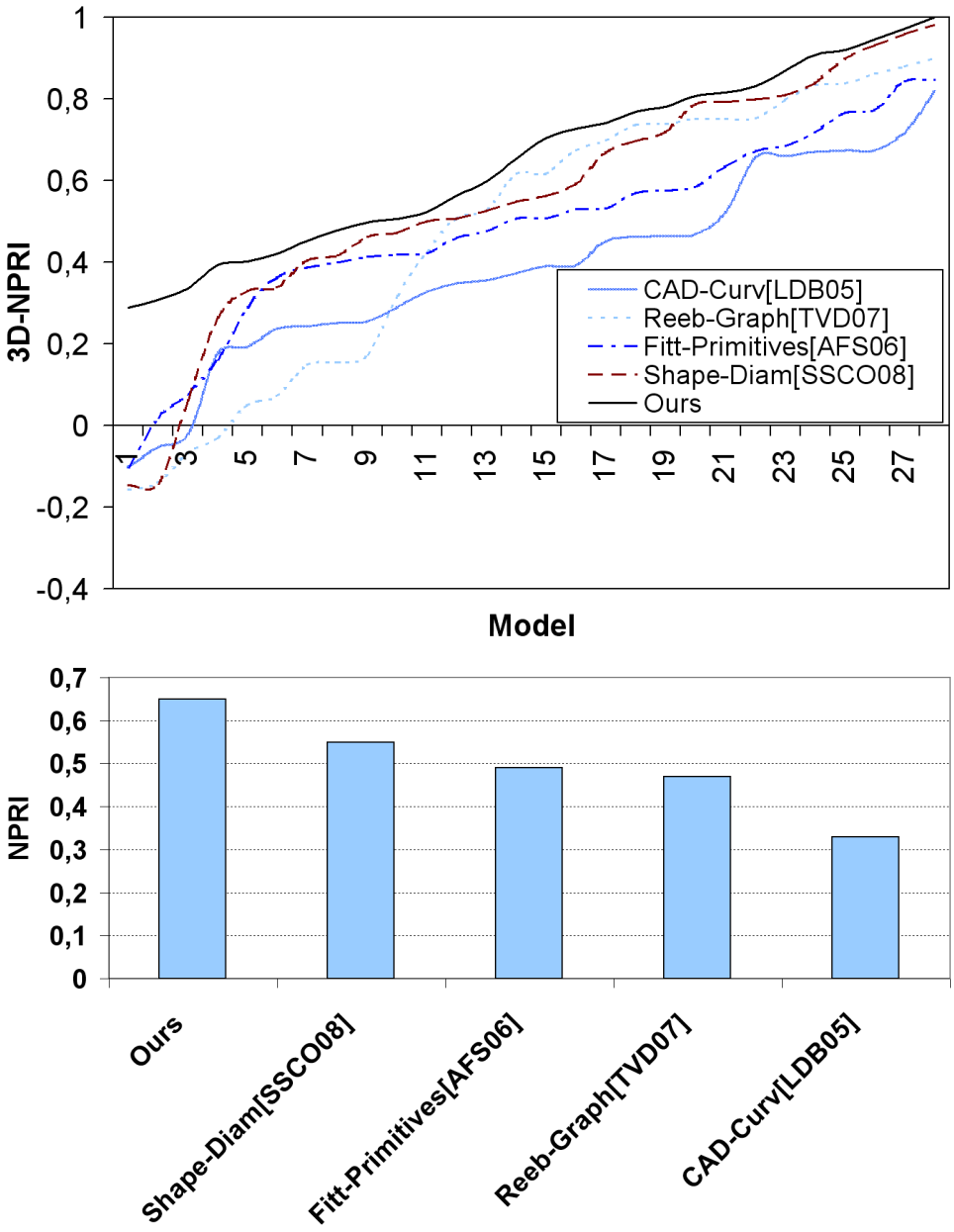



Figure 4.12 – *Comparison of different segmentation algorithms using our benchmark; (top: scores of NPRI sorted in increasing order over all the corpus models, bottom: the average of NPRI over all the corpus). Although in this experiment, our method is based on a global learning, performed on a different database, it outperforms the others.*

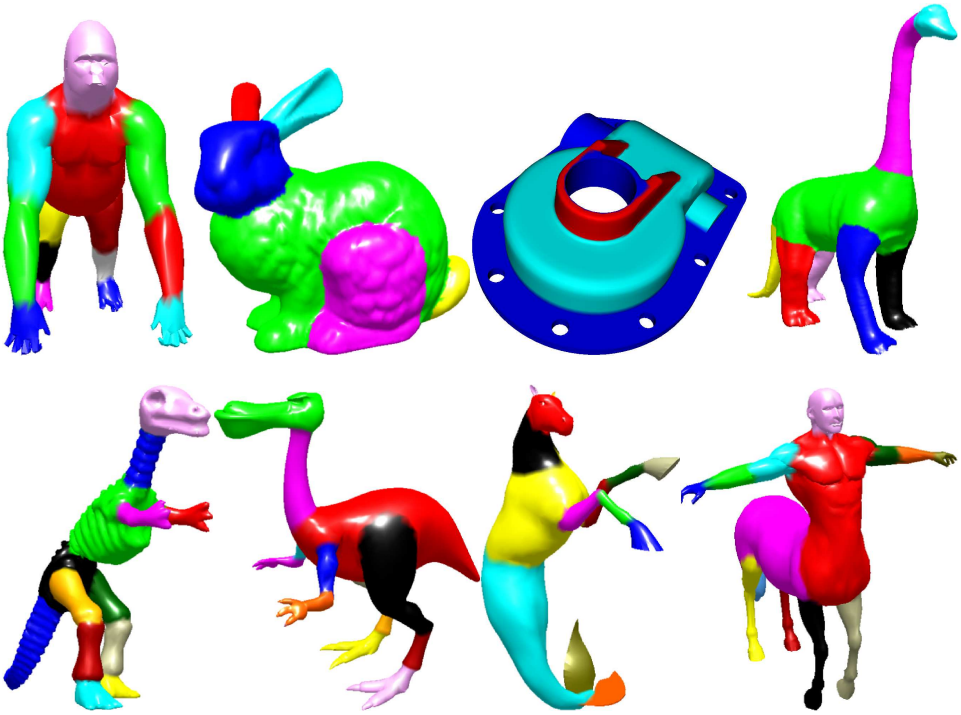


Figure 4.13 – *Segmentations results, obtained by our algorithm trained on the Princeton benchmark, for a variety of meshes from different databases.*

### 4.4.3 Algorithm efficiency regarding the category of models

In this third experiment, we want to study the behavior of our algorithm regarding the different categories of models in both benchmarks. For this reason we use the NPRI which is a more discriminative metric than a simple Rand Index (see previous chapter for more details). The NPRI is computed for each model, then averaged by category. Figure 4.14 illustrates the results obtained by our algorithm for the Princeton corpus models, and for our corpus models. Note that for the first corpus, we use a categorical learning of size 19, while for the second corpus, we use a global learning of size 6, both trained on the Princeton benchmark.

Globally, we can notice that for both corpuses, our algorithm gives quite good results for each category since the scores are much higher than zero. An interesting point is that the scores of common categories among the two corpuses are consistent, with a slight drop of performance for our corpus which is due to the difference of learning strategy (categorical *vs.* global on a different database). The figure illustrates also that the *bust*

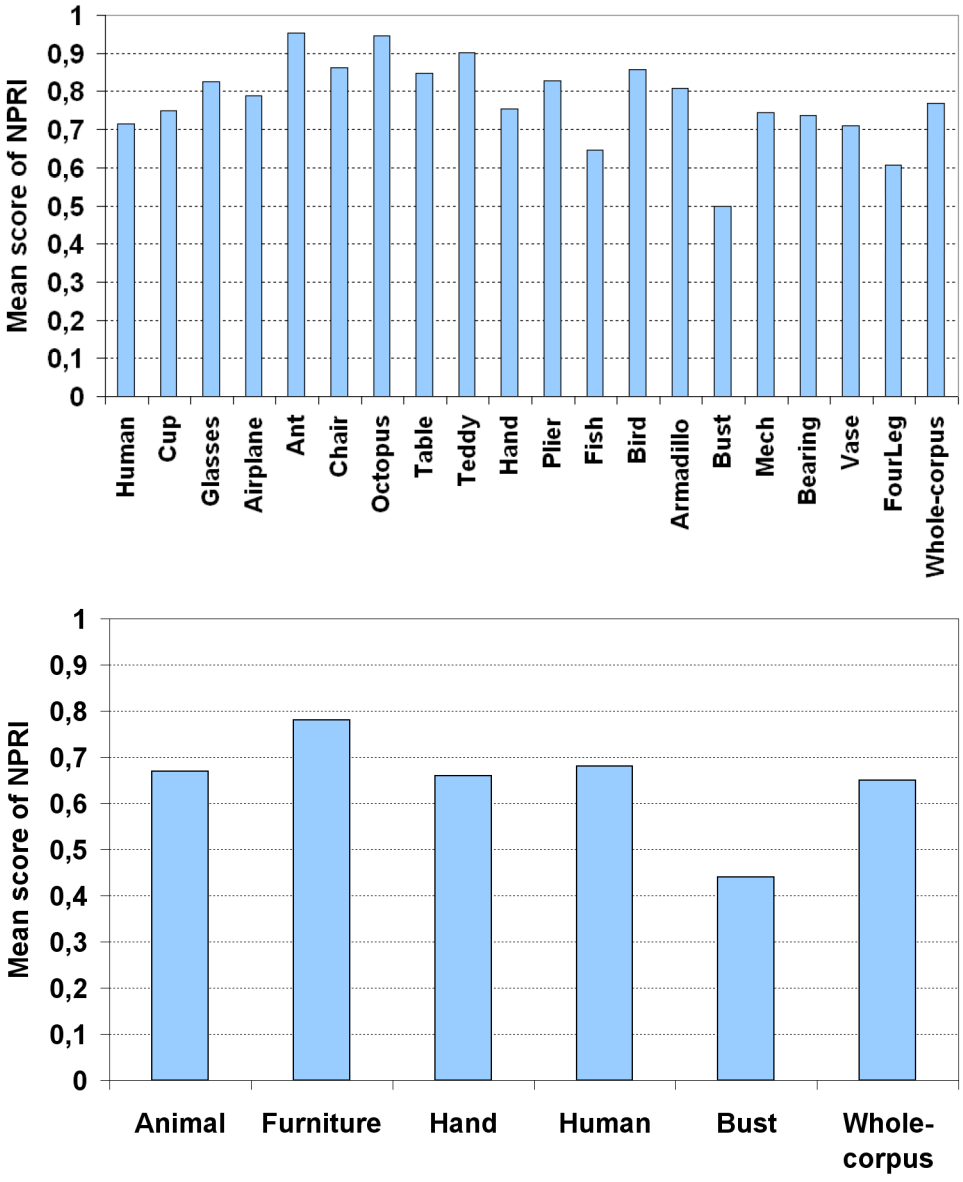



Figure 4.14 – *Scores of NPRI averaged for each category and for all models from the Princeton benchmark (CGF09) (on top), and from our benchmark (on bottom).*

category seems to be the most difficult one to segment (it is associated with the smallest NPRI values for both benchmarks). The fact is that human face images are well-known in subjective experiments as a high-level factor attracting human attention, hence some features not relevant from a geometrical point of view can be considered highly relevant for human observers and thus can influence the manual ground-truth segmentations.

### 4.4.4 Study of the performance of our improved snake movement

In this experiment, we show how the use of the learned boundary edge function improves the original snake movement algorithm from Jung *et al.* (JK04). To this end, we compute the similarity between boundaries extracted by our algorithm using both versions of the snake movement, and the manual boundaries on our corpus models. The most appropriate metric to compute this kind of similarity is the CDI (Cut Discrepancy Index), since it allows to compute the mapping degree between the extracted boundaries of two segmentations of the same model (see previous chapter). The metric is in the range $]-\infty; 1]$, and a value of 1 indicates a perfect matching between boundaries of the two segmentations. Figure 4.15 illustrates the scores of CDI averaged for each category of models and over all models of our corpus for these cases. The results show clearly that the new snake movement always improves the quality of the boundaries, whatever the category of models.

### 4.4.5 User interaction and coarse to fine segmentation

One of the strong point of our algorithm is that it is fully automatic; in particular the number of boundaries (and thus the number of segments) is automatically determined within our processing pipeline; it corresponds to the number of connected interest regions from the edge classification step (see figure 4.5(b)). However if needed, it is quite easy to introduce human interactions in our process. This can be done by:

**Tuning the classification threshold applied on the edge function.** This threshold is set to 0 in our method; however it is still possible to decrease

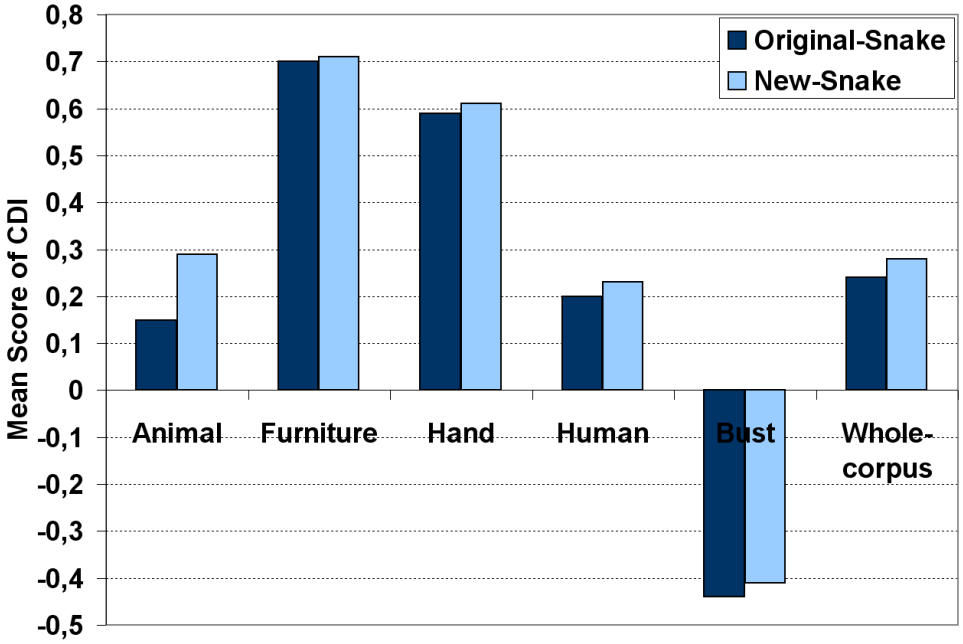



Figure 4.15 – *Scores of CDI averaged for each category and over all models of our corpus with the original (JK04) and our improved version of the snake movement.*

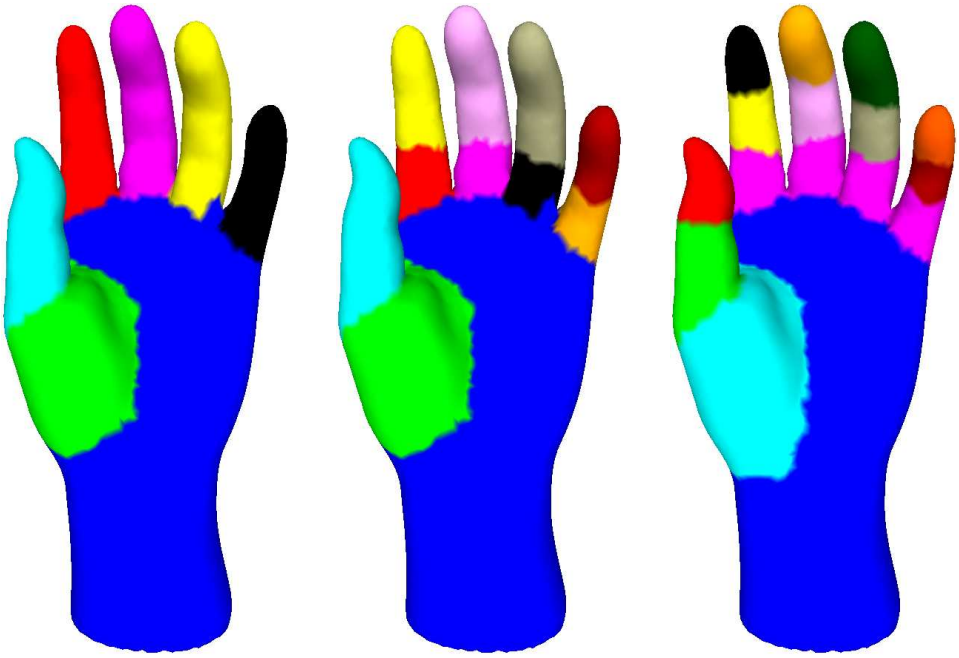
(resp. increase) this threshold to obtain more (resp. less) segments. Such a coarse to fine segmentation is illustrated in figure 4.16. We have to notice however that for a too low threshold (see the right hand in figure 4.16) some regions of interest may become very large and thus may be abnormally merged leading to merging some segments (like the bottom of the fingers). However this kind of problems never happened in all our experiments, with a threshold set to zero.

**Using a paintbrush.** The paintbrush allows to directly select on the mesh a set of edges representing a new interest region, similarly to Lee *et al.* (LLS*05). The segmentation process is then completed by performing the remaining steps.

### 4.4.6 Algorithm robustness regarding geometric transformations

We assessed the robustness of our method against two kinds of transformations, namely pose-variation and noise.

**Pose-variation.** Figure 4.17 shows the segmentations obtained by our algorithm for the *armadillo* and *human* models with different poses. These



(a) $H(x) > 0.3$ (b) $H(x) > 0$ (c) $H(x) > -0.2$

Figure 4.16 – *Example of coarse to fine segmentation obtained by tunning the classification threshold applied on* $H(x)$.

models are available in the Princeton segmentation benchmark. The edge function used to segment the models is based on a global learning, with a training set size of $6 \times 19$ models (6 models selected from each category). Globally, the segmentation of the models is quite stable, which underlines the pose robustness of our algorithm.

**Noise.** We applied on the vertices of the test models two random displacements in the direction of $x-$, $y-$, and $z-$axis (3% and 5% of the bounding box length). Figure 4.18 shows the boundaries extracted by our algorithm for the noisy versions of the *ant* model. On the top, the boundaries are generated using an edge function based on a categorical learning, with a training set size of 19 models (all the training models belong to the *ant* category); while in the bottom, the edge function is based on a global learning, with a training set size of $6 \times 19$ (6 models selected from each category). When applying a noise of 3%, the results remain very good for the categorical learning and correct for the global learning. However when applying a strong noise (5%) the quality of the boundaries is seriously de-

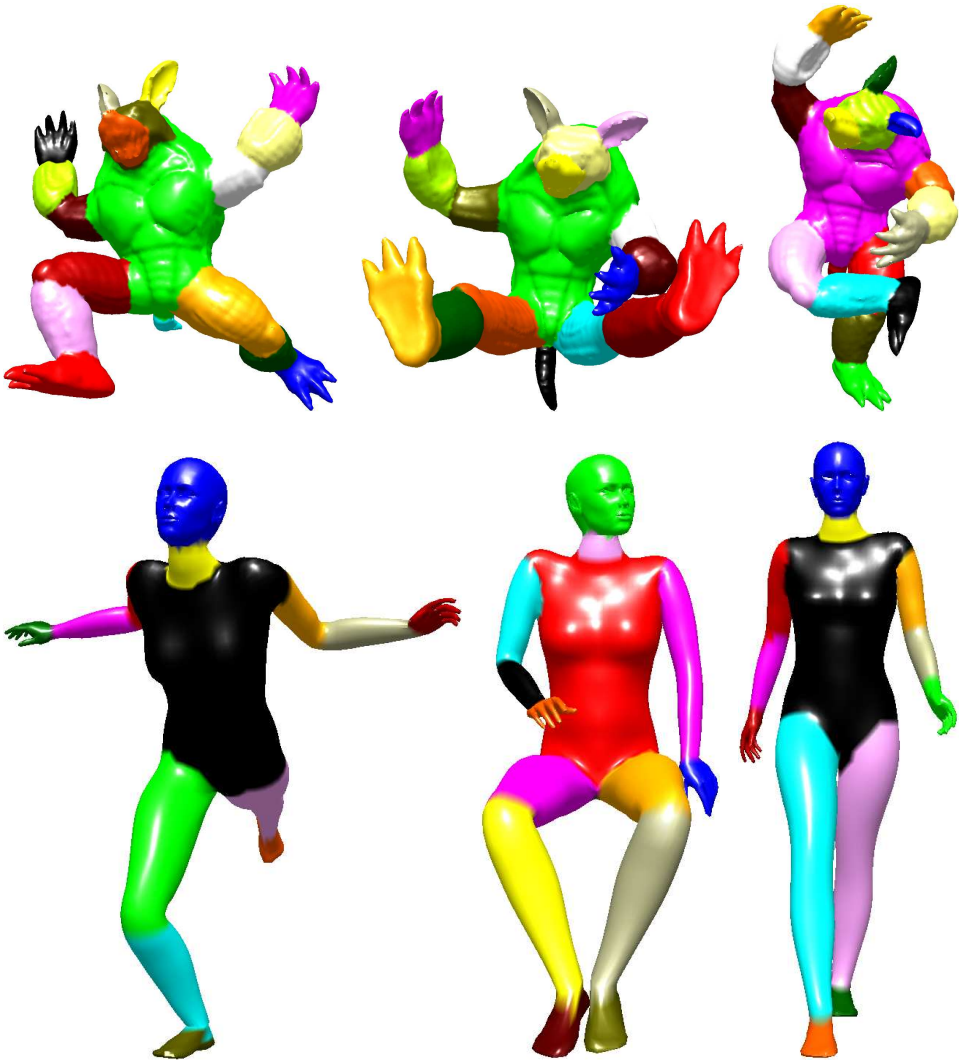
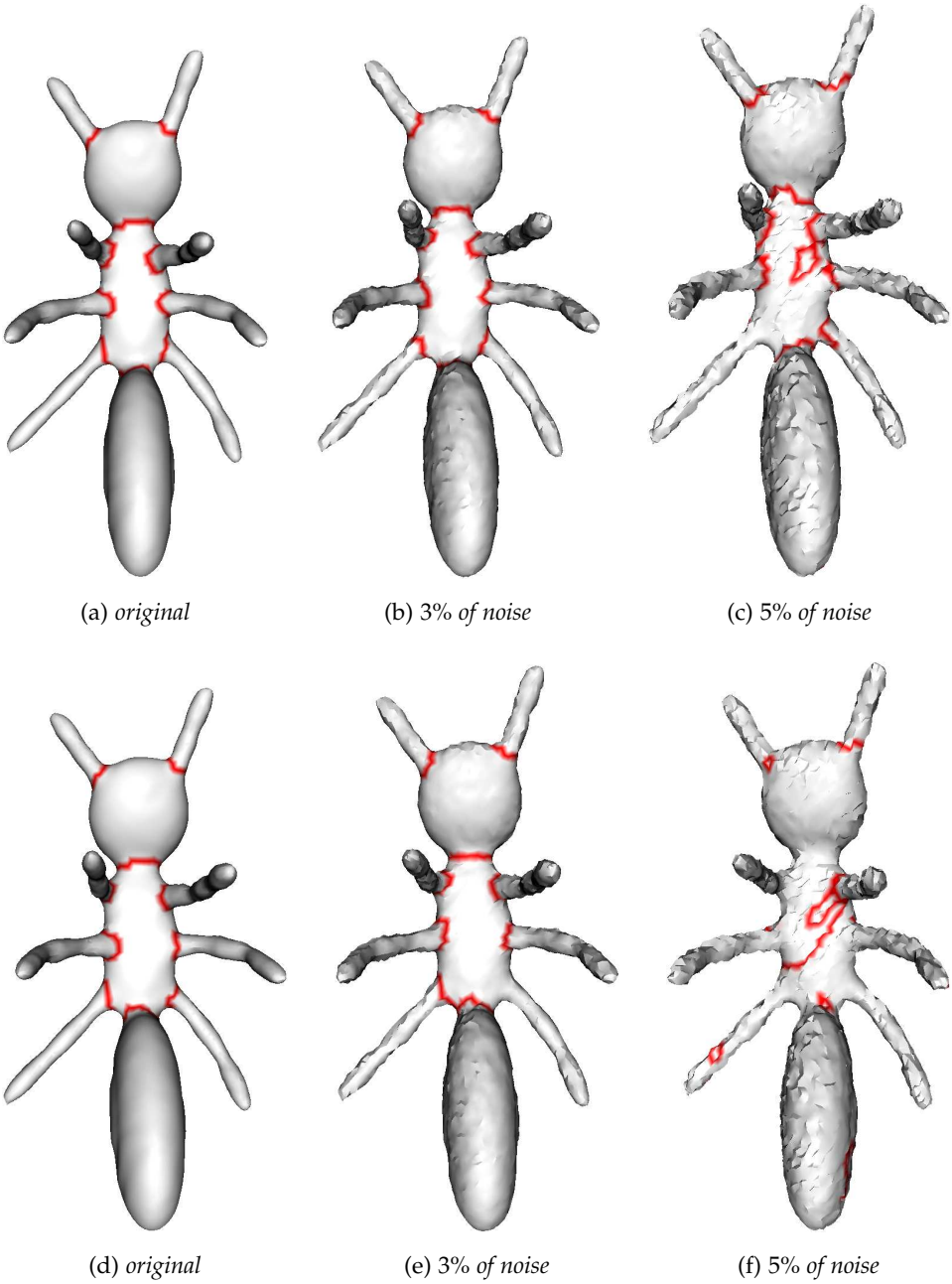


Figure 4.17 – *Algorithm robustness against pose-variation.*

graded particularly when using the global learning. We have conducted tests on 19 models (one model randomly selected from each category of the Princeton segmentation benchmark) for 3% and 5% of noise, the quality of segmentation decreases respectively of 1% and 12% when using the categorical learning, and of 10% and 50% when using the global learning.

The quality is measured by computing the NPRI for each test model. This moderate robustness is due to the fact that the learning step is performed on clean data; hence the learned function fails to extract the right boundaries in the noisy models since the edges composing them do not share the same geometric properties as in the clean ones. A solution could



(a) *original* (b) 3% *of noise* (c) 5% *of noise*

(d) *original* (e) 3% *of noise* (f) 5% *of noise*

Figure 4.18 – *Algorithm robustness against noise; (top: boundary extraction based on categorical learning, bottom: boundary extraction based on global learning).*

be to artificially add noise on the segmented training data, before the feature extraction and learning.

### 4.4.7 Running time

All the previous experiments were carried out on a 2.99 GHz Intel(R) Core(TM) 2 Duo CPU with 3.25 Gb memory. Globally, the running time for the learning step is less than 10 minutes for most of the categories. The process is longer, around 40 minutes, for some categories of which the size of models is important such as the *armadillo* category. The on-line step (segmentation) runs at an interactive time (around 1 minute), except for the *armadillo* category for which the running time is more important (around 9 minutes). This is due to the size of mesh and the number of extracted boundaries which are both important (24k vertices and 16 boundaries on average). More precisely, the average running time in seconds for the thinning, contour completion, and snake movement steps is respectively 0.58, 3.81, and 45.95. Table 4.1 presents in detail the average running time for both steps (off-line and on-line) of our algorithm when applied on the Princeton corpus models.

## 4.5 Application to dynamic surfaces

With the recent progress of acquisition systems, and the increasing of the processors calculation power, the use of dynamic surfaces in the multimedia domain has become important. This kind of data can be created by a designer using animation softwares, or obtained from a scanner, or a scientific simulation, etc. Generally, the dynamic surface is represented under the form of a sequence of 3D-meshes (or a sequence of frames) with constant connectivity, and time-varying geometry (the position of vertices changes over time). Similarly to static meshes, dynamic meshes require different preprocessing steps before to be used in a given application. In this section we focus our interest on a specific preprocessing step which is the kinematic skeleton extraction. Most of existing works that address this latter task make use of motion-based geometric segmentation methods (DTTS08, SY07, AKcPT04). These methods seek to decompose the

| Category | Average nb of vertices | Average nb of segments | Learning(m.s.) | Thinning/branch-filter(s.) | Completion(s.) | Snake(s.) |
|---|---|---|---|---|---|---|
| Airplane | 5400 | 6 | 4.06 | 0.06 | 0.98 | 5.65 |
| Ant | 6370 | 11 | 6.08 | 0.03 | 2.25 | 7.95 |
| Armadillo | 24473 | 16 | 39.06 | 0.73 | 36.9 | 472.2 |
| Bearing | 1663 | 5 | 2.01 | 0.01 | 0.35 | 0.23 |
| Bird | 3478 | 6 | 2.59 | 1.62 | 0.71 | 3.01 |
| Bust | 9252 | 3 | 8.53 | 0.04 | 3.17 | 94.57 |
| Chair | 8499 | 8 | 8.35 | 0.01 | 2.6 | 7.82 |
| Cup | 15198 | 2 | 36.47 | 0.01 | 1.64 | 9.37 |
| Fish | 7121 | 6 | 6.58 | 1.04 | 1.42 | 10 |
| Fourleg | 6938 | 10 | 6.1 | 0.31 | 3.79 | 31.5 |
| Glasses | 7016 | 4 | 6.31 | 0.14 | 0.89 | 2.73 |
| Hand | 7242 | 6 | 7.54 | 0.01 | 1.57 | 11.92 |
| Human | 4706 | 15 | 3.51 | 0.01 | 2.03 | 17.59 |
| Mech | 14995 | 2 | 28.08 | 0.06 | 1.04 | 20 |
| Octopus | 5944 | 9 | 4.3 | 0.12 | 1.71 | 6.43 |
| Plier | 4487 | 5 | 3.31 | 0.57 | 0.82 | 4.34 |
| Table | 13926 | 5 | 9.42 | 0.67 | 2.7 | 65.5 |
| Teddy | 13826 | 8 | 9.26 | 5.51 | 4.82 | 57.7 |
| Vase | 14476 | 4 | 26.07 | 0.14 | 3.14 | 44.65 |

Table 4.1 – *Average computation time for the categorical learning step with a training set size of 19 models, and in sec for the on-line step.*

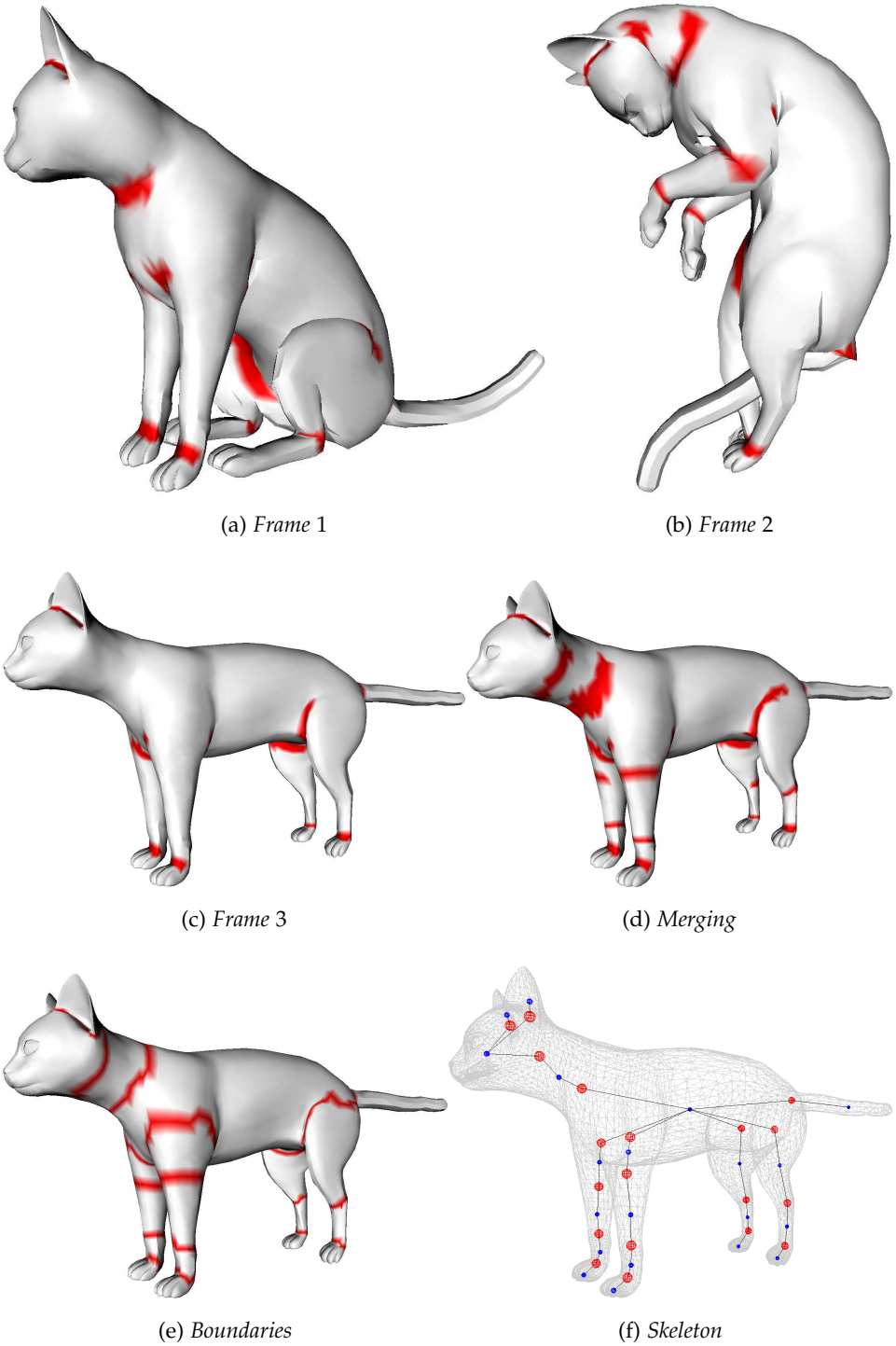
dynamic mesh into rigid parts by exploiting the temporal information. They assume that the vertices of such a part are characterized by a uniform motion with a single rigid transformation along the sequence. Once the parts are determined, an articulated skeleton is computed.

In what follow we propose a simple and precise kinematic skeleton extraction method for dynamic meshes. The method begins by extracting the different interest regions along the sequence (figure 4.19(a,b,c)) using our boundary edge function described in section 4.3.1. Next, these regions are merged along the different frames (figure 4.19(d)) to compute the final boundaries (figure 4.19(e)) using the pipeline described in section 4.3.2. Finally a skeleton is computed by connecting the centroids of these boundaries (red points in figure 4.19(f)) with the centroids of their adjacent segments (blue points in figure 4.19(f)).

The choice to use our segmentation method is justified by the fact that dynamic meshes are characterized by motion boundaries. Indeed, each part that undergoes a rigid transformation along the sequence of meshes involves the appearance of new boundaries (figure 4.19(a,b,c)). Consequently, our segmentation method allows to compute these boundaries for any input frame, while the merging allows to cover all of them. Moreover, contrary to the classical methods which seek to segment only parts characterized by motions, our method allows to capture even immobile shape parts since it is able to detect their boundaries (see the ears of the cut model in figure 4.19(d)). To summarize, we obtain a unique segmentation for the whole dynamic sequence that is based both on motion and geometric features.

To compute the skeleton, we use a simple algorithm which takes as input the dynamic mesh together with the set of closed boundaries and gives as output a structure composed of a set of edges and points that represents the kinematic skeleton. Figure 4.20 shows some dynamic surfaces and their extracted kinematic skeletons.

The algorithm begins by computing the centroids of both boundaries and segments, then it connects each boundary centroid with its two adjacent segment centroids. Two segments are adjacent if they share the



(a) *Frame* 1 (b) *Frame* 2

(c) *Frame* 3 (d) *Merging*

(e) *Boundaries* (f) *Skeleton*

Figure 4.19 – *Overview of our kinematic skeleton extraction method for dynamic meshes.*

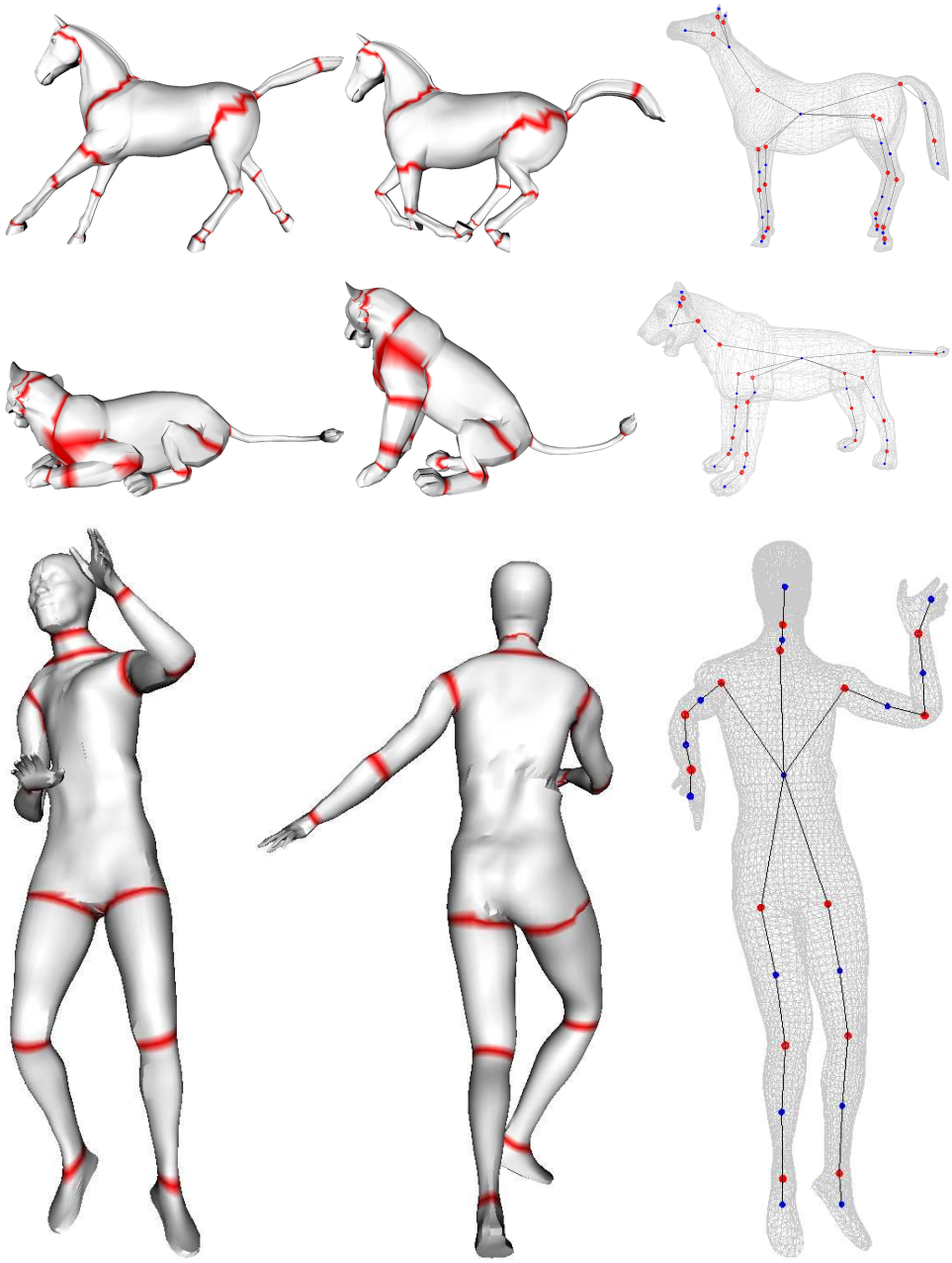
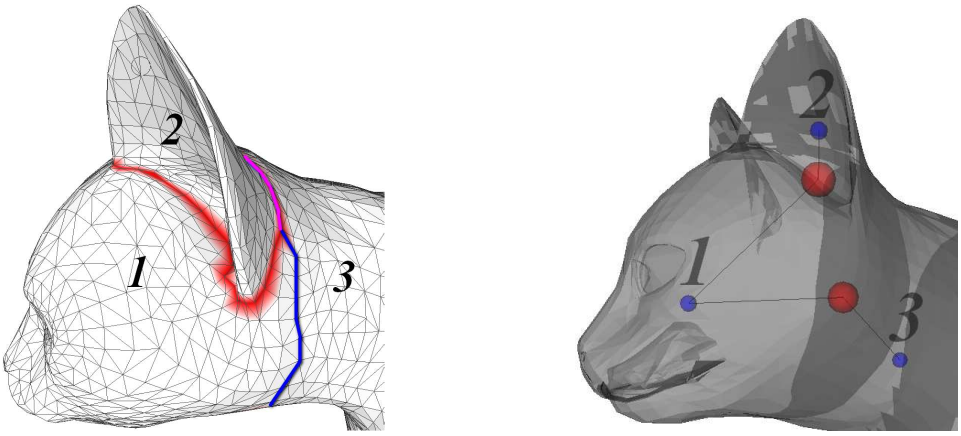


Figure 4.20 – *For each row, dynamic surfaces and their corresponding kinematic skeletons.*

same boundary edges or a part of them. However, it is possible that a part of a given boundary be shared between more than two segments. Figure 4.21(a) illustrates an example in which three segments are adjacent since they share a common part of their boundaries (see the magenta part in the figure). In this latter case, each boundary is connected with the two segments that share its maximum common part as illustrated in figure 4.21(b).




(a) (b)

Figure 4.21 – *Example of three adjacent segments (a), and its resulting skeleton (b).*

The whole process for computing the kinematic skeleton runs at an interactive time (only few minutes). Table 4.2 summarizes the running time for some models. Note that experiments were carried out on a 2.99 GHz Intel(R) Core(TM) 2 Duo CPU with 3.25 Gb memory.

| **Dynamic-mesh** | **Vertices** | **Frames** | **Skeleton (s.)** |
|---|---|---|---|
| Cat | 7207 | 10 | 37 |
| Dance | 7061 | 201 | 411 |
| Horse | 8431 | 49 | 204 |
| Lion | 7207 | 10 | 28 |

Table 4.2 – *Computation time for kinematic skeleton extraction of some dynamic meshes.*

## 4.6 Conclusion

In this chapter, we have presented a framework for segmentation based on a learning approach. A boundary edge function is learned from a set of ground-truths using AdaBoost classifier, and then is used to segment any input 3D-mesh through different processing steps. Our framework is the first to address the problem of learning boundary edges for 3D-mesh segmentation. We have shown the possibility to make a generic (or global) learning that can lead to segment a model without having a prior information about its category. We have conducted a battery of experiments using different benchmarks to validate the segmentation quality produced by our algorithm. The different experiments have demonstrated that our algorithm outperforms the most recent algorithms from the state-of-the-art, and produces segmentations which are very similar to those created by humans. For instance, we achieved 8.8% Rand Index error on the Princeton segmentation benchmark (CGF09), while the last best result achieved on this database is 14.9% Rand Index error obtained by Golovinskiy and Funkhouser (GF08) (algorithm without learning), and 9.5% Rand Index error obtained by Kalogerakis *et al.* (KHS10) (algorithm with learning). This latter algorithm needs a consistent labeling of the training data, which may requires some manual interactions. However, it provides semantic labeling which allows for instance to retrieve a part of interest across a data-base of labeled 3D-models without additional processing. We have also presented an application of our segmentation method for the kinematic skeleton extraction of dynamic meshes. To this end, we propose a simple method that makes use of our segmentation algorithm to extract motion boundaries that characterize the dynamic mesh. We have seen through some first tests that the obtained results are promising.

# Conclusion

# 5

## 5.1 Summary

The need of mesh segmentation as a preprocessing step for numerous applications such as shape recognition, modeling by example, compression, etc., has involved the development of an important amount of segmentation algorithms by the computer graphics community. We have seen through the state-of-the-art chapter that the different existing segmentation techniques seek to segment the 3D-mesh either from a geometric point of view (surface-type segmentation) or from a semantic point of view (part-type segmentation). In this thesis, we focused our interest rather in the second type of segmentation (part-type one) together with the evaluation of the algorithms that aim to produce segmentations of such type.

We first studied the mesh segmentation evaluation problem (chapter 3) due to the fact that before starting this thesis, no automatic tool had been proposed addressing this task in a general purpose context. To this end, we proposed a benchmark that includes a human-made ground-truth segmentation corpus and a relevant similarity metric that quantifies the consistency between these ground-truth segmentations and automatic ones produced by a given algorithm on the same models. Additionally, we have conducted extensive experiments including subjective ones to respectively demonstrate and validate the relevance of our benchmark.

Then we studied the part-type mesh segmentation problem with learning techniques (chapter 4). Indeed, earlier segmentation algorithms, developed by the computer graphics community, are not only sensitive to topological/geometric changes due to lack of relevant descriptors characterizing the semantic of the shape but are also, as raised by Kalogerakis

*et al.* (KHS10), limited to a single generic rule (e.g. skeleton topology) or a single feature (e.g. curvature tensor). To overcome these drawbacks and improve the design of segmentation algorithms, we exploited the human factor for segmentation via learning techniques. Motivated by human perception theory (HS97) statements concerning shape recognition, we proposed to learn, using multiple geometric criteria, a boundary edge function from a set of human segmented training meshes and then used this learned function, through a processing pipeline, to segment any input mesh without having any prior information about the category of this mesh. The battery of experiments conducted using different benchmarks had shown the performance superiority of our algorithm over the state-of-the-art. Moreover, we generalized our algorithm for dynamic meshes in order to extract their kinematic skeletons, and the first results are promising.

## 5.2 Future work

Our first contribution (benchmark for mesh segmentation evaluation) has been throughly studied in this thesis and we do not plan to extend our work in this direction. However, learning mesh segmentation is a new research area and can benefit from further improvements. Additionally, this new type of algorithms produces automatically meaningful segmentations (similar to those created by humans) that reflect the semantic of the shape. This means that all applications requiring such type of segmentation can use this algorithm as a preprocessing step. In what follow we point out some research areas concerning possible improvements that can be brought to our algorithm and a direct application of this latter one.

**Algorithm improvements.** Although the results of our algorithm are accurate, some limitations remain and require a thorough analysis and future work. First, the number of geometric criteria used for classification could be more important. In particular, adding some rich features from 3D shape retrieval research field would certainly improve the results and the sensitivity regarding to noise. The bust models are difficult to seg-

ment, and our algorithm rather produces a coarse segmentation for this kind of models. This is probably due to the high semantic aspect carried by a face. This semantic aspect influences the manual segmentations, and is difficult to capture using simple local geometric criteria. A solution could be to include more global features such as topological ones; another solution would be to learn a whole structure of the segmentation per category (a prior graphical model from the manual segmentations), instead of a simple binary classification model.

**Partial indexing.** Partial indexing systems allow to find, in a database, models that have similar parts even if they are globally different. We have seen in the state-of-the-art chapter that many implemented systems make use of segmentation. Basically, they first segment the models, extract the corresponding weighted adjacency graphs, and then apply a graph-matching algorithm to find models containing similar parts. The graph nodes correspond to the mesh segments and the edges reflect the adjacency between these segments. The weights of nodes correspond to a geometric descriptor computed for each segment. The main advantage of these systems is the fact that they combine both of topological information (thanks to the adjacency graph), and geometric information (thanks to the geometric descriptor) to describe the shape and the relationship between its parts. Consequently, the retrieval process should be performed more efficiently. It is clear that in this kind of systems the segmentation plays a major role and the quality of the retrieved results is directly affected by the quality of segmentation. Hence, these systems can benefit from our segmentation algorithm. Nevertheless, many questions remain open and need further investigations, for instance: should we use only one geometric shape descriptor or multiple ones? if so, how to combine them? otherwise, which one is the most suited to describe the shape parts? which graph-matching algorithm to use?

To our opinion, the number of geometric descriptors to use depends on the refinement level of the segmentation produced for a given 3D-shape. Indeed, if the segmentation is coarse, this means that the shape parts are still remaining complicated from a semantic point of view, so their geom-

etry is not uniform. For instance, if the segmentation algorithm produces for a human model a coarse segmentation in which the arms and the hands belong to the same segments, it is clear that one simple geometric descriptor will fail to distinguish such segments. Consequently, multiple descriptors should be combined using a simple mean for example. However, if the segmentation is fine, this means that the parts induced by the partitioning of the shape can be approximated to a set of primitives such as sphere, plane, cylinder, etc. In this latter case, one geometric descriptor such as the shape diameter function (SSCO08) should be sufficient to distinguish such kind of segments. Regarding the matching, existing techniques in the literature basically transform the matching problem which is NP-complete into an optimization problem to find an approximate solution. We therefore suggest to use one of the existing techniques in the literature such as the Successive Projection Graph Matching algorithm (SPGM), proposed by Wyk and Wyk (vWvW04). This latter algorithm had already been used by Mademlis *et al.* (MDA*08) for the same task (3D indexing based on graph matching) and shown its performance.

# Bibliography


[ADIV03] Alliez P., Devillers O., Isenburg M., Valette S.: Compression de maillages, un état de l'art. In *CORESA'03: Compression et Représentation des Signaux Audiovisuels* (2003). (Cited pages xii and 10.)

[AFS06] Attene M., Falcidieno B., Spagnuolo M.: Hierarchical mesh segmentation based on fitting primitives. *Vis. Comput. 22*, 3 (2006), 181–193. (Cited pages xii, 12, 17, 18, 67, 72, 73, 74, 79, 80, 81, 89, and 106.)

[AG05] Alliez P., Gostman C.: *Advances in Multiresolution for Geometric Modelling*. 2005, ch. Recent advances in compression of 3D meshes, pp. 3–26. (Cited page 32.)

[AGC*05] Atkar P. N., Greenfield A., Conner D. C., Choset H., Rizzi A. A.: Hierarchical segmentation of surfaces embedded in r3 for auto-body painting. In *International Conference on Robotics and Automation* (2005), pp. 572–577. (Cited page 14.)

[AKcPT04] Anguelov D., Kollerhoi-cheung D., Praveen P., Thrun S. S.: Recovering articulated object models from 3d range data. In *20th conference on Uncertainty in artificial intelligence* (2004), pp. 18–26. (Cited page 118.)

[AKM*06] Attene M., Katz S., Mortara M., Patané G., Spagnuolo M., Tal A.: Mesh segmentation, a comparative study. *IEEE International Conference on Shape Modeling and Applications* (2006), 7–7. (Cited pages 12, 31, and 40.)

[APP*07] Agathos A., Pratikakis I., Perantonis S., Sapidis N., Azariadis P.: 3d mesh segmentation methodologies for

cad applications. *Computer-Aided Design and Applications 4(6)* (2007), 827–842. (Cited pages 12 and 31.)

[ARMS09] Attene M., Robbiano F., M. Spagnuolo B. F.: Characterization of 3d shape parts for semantic annotation. *Computer-Aided Design 10(41)* (2009), 756–763. (Cited pages xii, 34, and 35.)

[ATC*08] Au O. K.-C., Tai C.-L., Chu H.-K., Cohen-Or D., Lee T.-Y.: Skeleton extraction by mesh contraction. *ACM Transaction on Graphics 27(3)* (2008), 1–10. (Cited page 24.)

[BDBP09] Berretti S., Del Bimbo A., Pala P.: 3d mesh decomposition using reeb graphs. *Image Vision Comput. 27*, 10 (2009), 1540–1554. (Cited pages 40 and 56.)

[BH03] Brand M., Huang K.: A unifying theorem for spectral embedding and clustering. In *Proc. 9th International Workshop on Artificial Intelligence and Statistics* (2003). (Cited page 21.)

[CDGEB07] Corsini M., Drelie Gelasca E., Ebrahimi T., Barni M.: Watermarked 3d mesh quality assessment. *IEEE Transaction on Multimedia 9*, 2 (February 2007), 247–256. (Cited page 73.)

[CEL*04] Chabrier S., Emile B., Laurent H., Rosenberger C., Marche P.: Unsupervised evaluation of image segmentation: Application to multi-spectral images. In *International Conference on Pattern Recognition* (2004), pp. 576–579. (Cited page 39.)

[CERL06] Chabrier S., Emile B., Rosenberger C., Laurent H.: Unsupervised performance evaluation of image segmentation. *EURASIP J. Appl. Signal Process. 2006* (2006), 217–229. (Cited page 39.)

[CG06] Chen L., Georganas N. D.: An efficient and robust algorithm for 3d mesh segmentation. *Multimedia Tools Appl. 29* (2006), 109–125. (Cited pages 14, 15, and 16.)

[CGEB07] Corsini M., Gelasca E. D., Ebrahimi T., Barni M.: Watermarked 3d mesh quality assessment. *IEEE Transaction on Multimedia 9* (2007), 247–256. (Cited page 76.)

[CGF09] Chen X., Golovinskiy A., Funkhouser T.: A benchmark for 3d mesh segmentation. *ACM Transactions on Graphics (SIGGRAPH) 28(3)* (2009). (Cited pages xiv, xv, 4, 26, 38, 41, 44, 52, 54, 56, 57, 58, 59, 79, 81, 82, 83, 84, 85, 91, 104, 112, and 124.)

[Chu97] Chung F. R. K.: Spectral graph theory. In *CBMS Regional Conference Series in Mathematics. American Mathematical Society* (1997). (Cited page 21.)

[Coh60] Cohen J.: A coefficient of agreement for nominal scales. *Educ. and psychological Measurment* (1960), 37–46. (Cited page 59.)

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. *ACM Transaction on Graphics 23(3)* (2004), 905–914. (Cited page 19.)

[Dan99] Daniel W. W.: *A Foundation For Analysis In The Health Sciences Books*. 7th edition.John Wiley and sons., 1999. (Cited page 76.)

[DTTS08] DeAguiar E., Theobalt C., Thrun S., Seidel H.-P.: Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum (Proc. Eurographics EG'08) 27*, 2 (2008), 389–397. (Cited page 118.)

[FBCM04] Fowlkes C., Belongie S., Chung F., Malik J.: Spectral grouping using the nyström method. *IEEE Transaction on Pattern Analysis and Machine Intelligence 26* (2004), 214–225. (Cited pages 21 and 23.)

[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 (Proc. SIGGRAPH)* (2004). (Cited pages xiii, 35, and 36.)

[FLL11] Fan L., Liu L., Liu K.: Paint mesh cutting. *Computer Graphic Forum journal (Proceedings of Eurographics) 30*, 2 (2011), to appear. (Cited pages 25 and 26.)

[FM83] Fowlkes E. B., Mallows C. L.: A method for comparing two hierarchical clusterings. *Journal of the American statistical association 78(383)* (1983), 553–569. (Cited pages 59 and 61.)

[FS97] Freund Y., Schapire R. E.: A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*, 1 (1997), 119–139. (Cited pages 90 and 94.)

[GBP07] Giorgi D., Biasotti S., Paraboschi L.: Shrec: Shape retieval contest: Watertight models track. In *http://watertight.ge.imati.cnr.it/* (2007). (Cited page 52.)

[GEKS06] Gelasca E. D., Ebrahimi T., Karaman M., Sikora T.: A framework for evaluating video object segmentation algorithms. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop* (2006), IEEE Computer Society, p. 198. (Cited page 73.)

[GF08] Golovinskiy A., Funkhouser T.: Randomized cuts for 3d mesh analysis. *ACM Trans. Graph. 27*, 5 (2008), 1–12. (Cited pages xv, 27, 81, 106, 108, and 124.)

[GG04] Gelfand N., Guibas L.: Shape segmentation using local slippage analysis. In *Proc. Eurographics Symposium on Geometry Processing* (2004), pp. 214–223. (Cited page 17.)

[GJC01] Gerig G., Jomier M., Chakos A.: Valmet: A new validation tool for assessing and improving 3d object segmentation. In *MICCAI 2001: Fourth International Conference on Medical Image Computing and Computer-Assisted Intervention* (2001), Springer, pp. 516–523. (Cited pages 40 and 45.)

[Got03] Gotsman C.: On graph partitioning, spectral analysis, and digital mesh processing. In *International Conference on Shape*

*Modeling and Applications (SMI)* (2003), pp. 165–174. (Cited pages xii and 22.)

[GWH01] Garland M., Willmott A., Heckbert P.: Hierarchical face clustering on polygonal surfaces. In *Proc. ACM Symposium on Interactive 3D Graphics* (2001), pp. 49–58. (Cited page 17.)

[HG01] Hubeli A., Gross M.: Multiresolution feature extraction for unstructured meshes. In *Conference on Visualization (VIS'01)* (2001), pp. 287–294. (Cited pages 96 and 99.)

[HR84] Hauffman D., Richards W.: Parts of recognition. *Cognition 18* (1984), 65–95. (Cited page 90.)

[HS97] Hauffman D., Singh M.: Salience of visual parts. *Cognition 63* (1997), 29–78. (Cited pages 89 and 126.)

[HZCp04] He X., Zemel R. S., Carreira-perpiñán M. .: Multiscale conditional random fields for image labeling. In *IEEE Computer Vision and Pattern Recognition (CVPR)* (2004), pp. 695–702. (Cited page 92.)

[JK04] Jung M., Kim H.: Snaking across 3d meshes. In *12th Pacific Conference on Computer Graphics and Applications (PG'04)* (2004), IEEE Computer Society, pp. 87–93. (Cited pages xvi, 96, 103, 113, and 114.)

[JKS05] Julius D., Kraevoy V., Sheffer A.: D-charts: Quasi-developable mesh segmentation. *Computer Graphics Forum (Proc. Eurographics) 24* (2005), 581–590. (Cited page 19.)

[JLCW06] Ji Z., Liu L., Chen Z., Wang G.: Easy mesh cutting. *Computer Graphics Forum 3(25)* (2006), 283–291. (Cited page 25.)

[KG00] Karni Z., Gostman C.: Spectral compression of mesh geometry. In *SIGGRAPH'00: ACM International conference on computer graphics and interactive techniques* (2000), pp. 279–286. (Cited page 32.)

[KH04] Kaufhold J., Hoogs A.: Learning to segment images using region-based perceptual features. *IEEE Computer Vision and Pattern Recognition (CVPR)* (2004). (Cited page 92.)

[KHS10] Kalogerakis E., Hertzmann A., Singh K.: Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics 29*, 3 (2010). (Cited pages xv, 26, 31, 89, 90, 92, 105, 108, 124, and 126.)

[KJS07] Kraevoy V., Julius D., Sheffer A.: Modeling with interchangeable parts. In *Proc. Pacific Graphics* (2007), pp. 214–223. (Cited page 19.)

[KLT05] Katz S., Leifman G., Tal A.: Mesh segmentation using feature point and core extraction. *The Visual Computer 21*, 8-10 (2005), 649–658. (Cited pages 28 and 81.)

[KT03] Katz S., Tal A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (SIGGRAPH) 22(3)* (2003), 954–961. (Cited pages xii, 12, 19, 20, 22, and 89.)

[LDB05a] Lavoué G., Dupont F., Baskurt A.: High rate compression of cad meshes based on subdivision inversion. *Annals of Telecommunications 38(8)* (2005), 1286–1310. (Cited page 32.)

[LDB05b] Lavoué G., Dupont F., Baskurt A.: A new cad mesh segmentation method, based on curvature tensor analysis. *Computer Aided Design 37(10)* (2005), 975–987. (Cited pages 13, 31, 72, 73, 74, 79, and 89.)

[LHMR08] Lai Y.-K., Hu S.-M., Martin R. R., Rosin P.: Fast mesh segmentation using random walks. In *Proc. ACM Symposium on Solid and Physical Modeling* (2008), pp. 183–191. (Cited pages 27, 81, and 106.)

[LKA06] Lien J.-M., Keyser J., Amato N.: Simultaneous shape decomposition and skeletonization. In *Proceedings of ACMSolid*

*and Physical Modeling Symposium (SPM'06)* (2006), pp. 219–228. (Cited pages xiii, 36, and 37.)

[Llo82] Lloyd S.: Least suqare quantization in pcm. *IEEE Transaction on Information Theory 28(2)* (1982), 129–137. (Cited page 19.)

[LLS*04] Lee Y., Lee S., Shamir A., Cohen-Or D., Seidel H.-P.: Intelligent mesh scissoring using 3d snakes. In *Proceedings of the Computer Graphics and Applications, 12th Pacific Conference* (2004), pp. 279–287. (Cited page 25.)

[LLS*05] Lee Y., Lee S., Shamir A., Cohen-Or D., Seidel H.-P.: Mesh scissoring with minima rule and part salience. *Computer Aided Geometric Design 22*, 5 (2005), 444–465. (Cited pages xv, 96, 101, 102, and 114.)

[LPRM02] Lévy B., Petitjean S., Ray N., Maillot J.: Least squares conformal maps for automatic texture atlas generation. *ACM Trans. Graph. 3(21)* (2002), 362–371. (Cited pages xii and 33.)

[LTTH01] Li X., Toon T., Tan T., Huang Z.: Decomposing polygon meshes for interactive applications. In *Proc. Symposium on Interactive 3D Graphics* (2001), pp. 35–42. (Cited page 24.)

[LW08] Lavoué G., Wolf C.: Markov random fields for improving 3d mesh analysis and segmentation. In *Eurographics Workshop on 3D Object Retrieval* (2008). (Cited page 94.)

[LZ04] Liu R., Zhang H.: Segmentation of 3d meshes through spectral clustering. In *Proc. 12th Pacific Conference on Computer Graphics and Applications* (2004), pp. 298–305. (Cited pages 21, 22, and 23.)

[LZ07] Liu R., Zhang H.: Mesh segmentation via spectral embedding and contour analysis. *Computer Graphics Forum (Special Issue of Eurographics*, 3 (2007), 385–394. (Cited pages xii, 21, 23, and 89.)

[LZHM06] Lai Y. K., Zhou Q. Y., Hu S. M., Martin R. R.: Feature sensitive mesh segmentation. In *Proc. ACM Symposium on Solid and Physical Modeling* (2006), pp. 17–26. (Cited pages 19 and 20.)

[MDA*08] Mademlis A., Daras P., Axenopoulos A., Tzovaras D., Strintzis M.: Combining topological and geometrical features for global and partial 3-d shape retrieval. *IEEE TRANSACTIONS ON MULTIMEDIA 5(10)* (2008), 819–831. (Cited pages 34 and 128.)

[MFM04] Martin D. R., Fowlkes C. C., Malik J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 26*, 5 (2004), 530–549. (Cited page 92.)

[MFTM01] Martin D., Fowlkes C., Tal D., Malik J.: A database of human segmented natural images and its application to evaluating algorithms and measuring ecological statistics. In *International Conference On Computer Vision* (2001), pp. 416–423. (Cited pages 38, 39, and 57.)

[MGP06] Mitra N. J., Guibas L., Pauly M.: Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics 25*, 3 (2006), 560–568. (Cited page 28.)

[MHYS04] Manay S., Hong B.-W., Yezzi A. J., Soatto S.: Integral invariant signatures. In *ECCV* (2004), pp. 87–99. (Cited page 21.)

[MPS06] Mortara M., Patane G., Spagnuolo M.: From geometric to semantic human body models. *Computers & Graphics (30)* (2006), 185–196. (Cited page 36.)

[MW99] Mangan A. P., Whitaker R. T.: Partitioning 3d surface meshes using watershed segmentation. *IEEE Transaction on Visualization and Computer Graphics 4(5)* (1999), 308–321. (Cited pages 13 and 14.)

[NJW01] Ng A. Y., Jordan M. I., Weiss Y.: On spectral clustering:

Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14* (2001), pp. 849–856. (Cited page 21.)

[PC04] Peyre G., Cohen L.: Surface segmentation using geodesic centroidal tesselation. In *Proc. 3D Data Processing, Visualization, and Transmission, 2nd International Symposium* (2004), pp. 995–1002. (Cited page 19.)

[PF98] Perona P., Freeman W.: A factorization approach to grouping. In *Proc. European Conference on Computer Vision* (1998), pp. 655–670. (Cited page 21.)

[PKA03] Page D. L., Koschan A. F., Abidi M. A.: Perception-based 3d triangle mesh segmentation using fast marching watershed. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* (2003), pp. 27–32. (Cited page 14.)

[PSG*06] Podolak J., Shilane P., Golovinskiy A., Rusinkiewicz S., Funkhouser T.: A planar-reflective symmetry transform for 3D shapes. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (2006). (Cited page 28.)

[QXP*06] Qin A.-h., Xiong H., Peng H.-y., Liu Z., Shi J.-y.: Cluster parallel rendering based on encoded mesh. *Journal of Zhejiang University - Science A 7(7)* (2006), 1124–1133. (Cited page 32.)

[RaAMC07] Randão A., Alface P., Macq B., Cayre F.: Blind and robust watermarking of 3d models : How to withstand the cropping attack ? In *ICIP'07: IEEE Internation conference on image processing* (2007), pp. 465–468. (Cited page 33.)

[Ran71] Rand W. M.: Objective criteria for the evaluation of clustering methods. *Journal of the American statistical association 66(336)* (1971), 846–850. (Cited pages 59 and 85.)

[RB02] Razdan A., Bae M.: A hybrid approach to feature segmentation of 3-dimensional meshes. *Computer-Aided Design 9(35)* (2002), 783–789. (Cited page 13.)

[RR01] Rogowitz B. E., Rushmeier H. E.: Are image quality metrics adequate to evaluate the quality of geometric objects. In *in Human Vision and Electronic Imaging* (2001), pp. 34–348. (Cited page 73.)

[RT07] Reniers D., Telea A.: Skeleton-based hierarchical shape segmentation. In *Proc. IEEE Intenational Conference on Shape Modeling and Applications* (2007), pp. 179–188. (Cited page 24.)

[Sha08] Shamir A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556. (Cited pages xii, 12, and 31.)

[She01] Sheffer A.: Model simplification for meshing using face clustering. *Computer-Aided Design 33* (2001), 925–934. (Cited page 17.)

[SKS06] Simari P., Kalogerakis E., Singh K.: Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Proc. Eurographics Symposium on Geometry Processing* (2006). (Cited page 28.)

[SMKF04] Shilane P., Min P., Kazhdan P., Funkhoser M.: The princeton shape benchmark. In *Proceedings of the Shape Modeling International 2004* (2004), pp. 167–178. (Cited page 48.)

[Spi07] Spielman D. A.: Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science FOCS07* (2007), pp. 29–38. (Cited page 21.)

[SS05] Simari P. D., Singh K.: Extraction and remeshing of ellipsoidal representations from mesh data. In *Proc. Graphics Interface* (2005), pp. 161–168. (Cited page 19.)

[SSCO08] Shapira L., Shamir A., Cohen-Or D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput. 24*, 4 (2008), 249–259. (Cited pages xv, 19, 21, 72, 74, 78, 79, 80, 81, 89, 94, 106, 108, and 128.)

[SSGH01] Sander P., Snyder j., Gortler S., Hoppe H.: Texture mapping progressive meshes. In *SIGGRAPH'01: ACM International conference on computer graphics and interactive techniques* (2001), pp. 409–416. (Cited pages 32, 40, and 45.)

[SSS*10] Shapira L., Shalom S., Shamir A., Cohen-Or D., Zhang H.: Contextual part analogies in 3d objects. *International Journal of Computer Vision 2-3(89)* (2010), 309–326. (Cited pages xii and 34.)

[STK02] Shlafman S., Tal A., Katz S.: Metamorphosis of polyhedra surfaces using decomposition. *Computer Graphics Forum 21(3)* (2002), 219–228. (Cited page 19.)

[SWRC09] Shotton J., Winn J., Rother C., Criminisi A.: Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Intl. Journal of Computer Vision 81*, 1 (2009). (Cited page 92.)

[SWS*03] Sander P., Wood Z., Snyder J., Gortler S., Hoppe H.: Multi-chart geometry images. In *SGP'03: Symposium on Geometry Processing* (2003), pp. 146–155. (Cited page 32.)

[SY07] Schaefer S., Yuksel C.: Example-based skeleton extraction. In *Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), pp. 153–162. (Cited page 118.)

[Tie08] Tierny J.: *Reeb graph based 3D shape modeling and applications*. PhD thesis, LIFL, University of Lille, France, 2008. (Cited page 24.)

[TVD07] Tierny J., Vandeborre J.-P., Daoudi M.: Topology driven 3D mesh hierarchical segmentation. In *IEEE International Conference On Shape Modeling And Application (SMI)* (2007). (Cited pages xii, xiii, xiv, 24, 25, 69, 70, 72, 74, 79, 80, and 89.)

[UPH07] Unnikrishnan R., Pantofaru C., Hebert M.: Toward objective evaluation of image segmentation algorithms. *IEEE*

*Transaction on pattern analysis and machine intelligence 29(6)* (2007), 929–944. (Cited pages 39, 59, 60, 61, 62, and 81.)

[vWvW04] van Wyk B. J., van Wyk M. A.: A pocs-based graph matching algorithm. *IEEE Transaction on Pattern Analysis and Machine Intelligence 26*, 11 (2004), 1526–1530. (Cited page 128.)

[WK05] Wu J., Kobbelt L.: Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum (Proc. Eurographics) 24(3)* (2005), 277–284. (Cited page 19.)

[WLDB11] Wang K., Lavoué G., Denis F., Baskurt A.: Robust and blind mesh watermarking based on volume moments. *Computers & Graphics 1(35)* (2011), 1–19. (Cited page 33.)

[WPP*07] Wu H., Pan C., Pan J., Yang Q., Ma S.: A sketch-based interactive framework for real-time mesh segmentation. In *Computer Graphics International* (2007). (Cited pages xii, 25, and 26.)

[XWS03] Xiao Y., Werghi N., Siebert P.: A topological approach for segmenting human body shape. In *Proc. 12th International Conference on Image Analysis and Processing* (2003), p. 82. (Cited page 24.)

[YGZS05] Yamauchi H., Gumhold S., Zayer R., Seidel H.-P.: Mesh segmentation driven by gaussian curvature. *Visual Computer 8-10(21)* (2005), 649–658. (Cited page 14.)

[YLW06] YAN D., Liu Y., Wang W.: Quadric surface extraction by variational shape approximation. In *Proc. Geometric Modeling and Processing* (2006), pp. 73–86. (Cited page 19.)

[ZFG08] Zhang H., Fritts J., Goldman S.: Image segmentation evaluation: A survey of unsupervised methods. *Computer Vision and Image Understanding 110* (2008), 260–280. (Cited page 38.)

[ZH04] Zhou Y., Huang Z.: Decomposing polygon meshes by means

of critical points. In *Proc. Conference on Multimedia Modeling* (2004), pp. 187–195. (Cited page 13.)

[ZHD*01] Zha H., He X., Ding C., Simon H., Gu M.: Spectral relaxation for k-means clustering. In *Proc. Advances in Neural Information Processing Systems* (2001), pp. 1057–1064. (Cited page 21.)

[ZPK*02] Zhang Y., Paik J., Koschan A., Abidi M., Gorsich D.: A simple and efficient algorithm for part decomposition of 3d triangulated models based on curvature analysis. In *Proceedings of International Conference on Image Processing* (2002), pp. 273–276. (Cited page 13.)

[ZTS02] Zuckerberger E., Tal A., Shlafman S.: Polyhedral surface decomposition with applications. *Computers and Graphics 26(5)* (2002), 733–743. (Cited pages xii, 12, 13, 14, 15, 16, and 89.)

**Titre** Segmentation de maillages 3D : évaluation automatique et une nouvelle méthode par apprentissage

**Résumé** Dans cette thèse, nous abordons deux problèmes principaux, à savoir l'évaluation quantitative des algorithmes de segmentation de maillages ainsi que la segmentation de maillages par apprentissage en exploitant le facteur humain.
Nous proposons les contributions suivantes :

- Un benchmark dédié à l'évaluation des algorithmes de segmentation de maillages 3D. Le benchmark inclut un corpus de segmentations vérités-terrains réalisées par des volontaires ainsi qu'une nouvelle métrique de similarité pertinente qui quantifie la cohérence entre ces segmentations vérités-terrains et celles produites automatiquement par un algorithme donné sur les mêmes modèles. De plus, nous menons un ensemble d'expérimentations, y compris une expérimentation subjective, pour respectivement démontrer et valider la pertinence de notre benchmark.

- Un algorithme de segmentation par apprentissage. Pour cela, l'apprentissage d'une fonction d'arête frontière est effectué, en utilisant plusieurs critères géométriques, à partir d'un ensemble de segmentations vérités-terrains. Cette fonction est ensuite utilisée, à travers une chaîne de traitement, pour segmenter un nouveau maillage 3D. Nous montrons, à travers une série d'expérimentations s'appuyant sur différents benchmarks, les excellentes performances de notre algorithme par rapport à ceux de l'état de l'art. Nous présentons également une application de notre algorithme de segmentation pour l'extraction de squelettes cinématiques pour les maillages 3D dynamiques.

**Mots-clés** Segmentation de maillages 3D, vérité-terrain, tests subjectifs, benchmark, apprentissage.

**Title** 3D-mesh segmentation: automatic evaluation and a new learning-based method

**Abstract** In this thesis, we address two main problems namely the quantitative evaluation of mesh segmentation algorithms and learning mesh segmentation by exploiting the human factor.
We propose the following contributions:

- A benchmark dedicated to the evaluation of mesh segmentation algorithms. The benchmark includes a human-made ground-truth segmentation corpus and a relevant similarity metric that quantifies the consistency between these ground-truth segmentations and automatic ones produced by a given algorithm on the same models. Additionally, we conduct extensive experiments including subjective ones to respectively demonstrate and validate the relevance of our benchmark.

- A new learning mesh segmentation algorithm. A boundary edge function is learned, using multiple geometric criteria, from a set of human segmented training meshes and then used, through a processing pipeline, to segment any input mesh. We show, through a set of experiments using different benchmarks, the performance superiority of our algorithm over the state-of-the-art. We present also an application of our segmentation algorithm for kinematic skeleton extraction of dynamic 3D-meshes.

**Keywords** 3D-mesh segmentation, ground-truth, subjective tests, evaluation, benchmark, learning.