Semi-supervised Segmentation of 3D Surfaces using a Weighted Graph Representation

Filippo Bergamasco, Andrea Albarelli, and Andrea Torsello

Dip. di Scienze Ambientali, Informatica e Statistica, Universitá Ca' Foscari Venezia

Abstract. A wide range of cheap and simple to use 3D scanning devices has recently been introduced in the market. These tools are no longer addressed to research labs and highly skilled professionals. By converse, they are mostly designed to allow inexperienced users to easily and independently acquire surfaces and whole objects. In this scenario, the demand for automatic or semi-automatic algorithms for 3D data processing is increasing. Specifically, in this paper we concentrate on the segmentation task applied to the acquired surfaces. Such a problem is well known to be ill-defined both for 2D images and 3D objects. In fact, even with a perfect understanding of the scene, many different and incompatible semantic or syntactic segmentations can exist together. For this reasons, we refrain from any attempt to offer an automatic solution. Instead we introduce a semi-supervised procedure that exploits an initial set of seeds selected by the user. In our framework segmentation happens by iteratively visiting a weighted graph representation of the surface starting from the supplied seeds. The assignment of each element is driven by a greedy approach that accounts for the curvature between adjacent triangles. The proposed technique does not require to perform edge detection or to fit parametrized surfaces and its implementation is very straightforward. Still, despite its simplicity, tests made on scanned 3D objects show its effectiveness and easiness of use.

1 Introduction

Segmentation is an important preliminary task in many 2D and 3D data processing pipelines. For instance, splitting an image or a 3D object in smaller parts is very useful to perform high-level recognition [1, 2], reverse engineering [3, 4] and even tracking [5]. Of course, the expected outcome of a segmentation procedure is different depending on the intended use of the resulting parts. If the goal is to produce a set of image macro pixels, segments will be searched at a purely syntactic level, grouping together pixels of uniform color or texture, regardless of their belonging to one object or another. By contrast, different scenarios require a more semantical splitting, with the aim of separating foreground from background or finding the boundaries of the objects found in the scene. Of course these approaches tend to be more specialized, since the cues to exploit strongly depend on the problem context and on the availability of humans in the loop.

When dealing with the 3D domain, segmentation is mostly targeted at splitting an object or a surface into subdomains that can be later interpreted as

parametrized primitives. The complexity of such primitives can range from basic items, such as planes, cylinders or spheres, to complete parametrized models, depending on the overall goal of the pipeline. Simple primitives are fitted to the segmented parts mainly for object simplification [6, 7], while completed models can be used for direct 3D object recognition with resilience to clutter [8] or invariance to scale [9]. Finally, another important application that needs surface decomposition is the angle and distance-preserving piecewise parametrization needed to apply textures to objects [10].

Surface segmentation can happen through many different methods. Some of them use standard clustering techniques or borrow segmentation procedures from the 2D domain, other exploit graph partitioning algorithms, shape fitting or even the distribution of symmetry planes over watertight objects.

Shlafman at al. propose to use a variation of K-means to group the triangles of the mesh into clusters [11]. This is a quite direct adaptation: first the user specify the desired number of clusters (k), then the process start by randomly selecting a set of k well spaced seed triangles and it iterates by alternating an assignment step (where each non-seed triangle is assigned to the nearest seed) and an adjustment step (where new seeds are selected by picking the triangle nearest to the center of each cluster).

Another classical technique is adapted by Moumoun et al., that suggests the use the Watershed principle [12] on a hierarchical transformation of connected faces structure based on the principal curvature. An interesting perspective on 3D segmentation is supplied by Podolak et al. [13], whose solution exploits the relation between mesh elements and the symmetry planes of the whole object.

Katz et al. [13] split the problem into two separate steps: first a probabilistic clustering is used in order to obtain meaningful but fuzzy components, then exact boundaries are constructed by assigning shared faces to the final cluster. A couple of years later, the same authors present an additional hierarchical method [14] that performs three steps: the transformation of the surface into a pose insensitive representation by mean of Multi-Dimensional Scaling (MDS); the localization of prominent feature points to be used as seeds; the extraction of clusters by refinement of core components obtained using spherical mirroring.

Mortara et al. [15] propose a multi-scale method based on blowing bubbles. The surface segmentation happens by clustering vertices with respect to their morphological behavior at different scales. This is done by centering on each vertex spheres of increasing diameter and using the curves resulting by their intersection with the mesh as a characterizing descriptor for the clustering process. Shapira et al. [16] describe a method that exploits the Shape Diameter Function (SDF), a measure related to the object volume in the neighborhood of each point that is computed for the barycenter of each triangle. The segmentation procedure relies on a two phase process. In the first phase, a Gaussian Mixture Model is used to fit k Gaussians to the histogram of all SDF values in order to produce a probability vector of length k for each triangle indicating its likelihood to be assigned to each of the SDF clusters. In the second step the segmentation is refined using the alpha expansion graph-cut algorithm, which is

used to minimize an energy function that combines the vectors obtained in the first phase along with boundary smoothness and concaveness.

Attene et al. [17] introduce the use of geometric primitives to drive a hierarchical segmentation. Specifically, at an initial stage each surface triangle represents a singleton cluster associated to the primitive that best fits it. Such primitives can be planes, spheres and cylinders. At each step, all the adjacent clusters are considered for merging and those that can be better approximated with one of the primitives form a new single cluster. The process stops when the desired number of segments has been obtained. Lai et al. [18] describe a procedure based on random walks that operates in two steps. Initially a set of seeds is chosen and the mesh is over-segmented by assigning each face to the seed that has the highest probability of reaching it by a random walk. The obtained segments are then hierarchically merged until the desired number of cluster is obtained. This is done following an order based on the relative lengths of the intersections and total perimeters of adjacent segments.

Finally, Golovinskiy and Funkhouser [19] use both normalized cuts and randomized cuts. In a similar manner to [17], normalized cut segmentation happens by first assigning each face of the mesh to its own cluster and then by merging them hierarchically in an order determined by the area-normalized cut cost, i.e. the sum of each segment perimeter (weighted by concavity) divided by its area. In this way it is possible to obtain segments that exhibit small boundaries along concave seams while maintaining segments with roughly similar areas. Differently, randomized cuts segmentation is initially applied to a strongly decimated mesh, obtaining very large segments. Those segments are then hierarchically splitted in a top-down manner, starting with all the faces in a single segment. For each split, a set of randomized cuts is computed over the segment, and the cut that is most consistent with others in the randomized set is identified. Among this set of candidates, the one that results in the minimal normalized cut cost is chosen. In both cases, the process stops when the required number of segments has been reached.

In this paper we introduce a novel graph-based segmentation approach. Differently from other previously proposed algorithms we do not adopt any global optimization method and we only rely on surface normals. While an initial seeding is required by the user, our approach is very simple to implement and the experimental validation highlights its speed and its good performance when compared with other graph-based systems.

2 Weighted Graph-Based Seeded Segmentation

Graph-based segmentation has been previously explored by several authors [19, 18, 20]. Most of the approaches found in literature perform some global computation over the graph in order to evaluate random walk reachability or optimal cuts. The algorithm presented in this paper, after building a weighted dual graph, adopts a straightforward greedy approach that directly extends an initial set of seeds by picking one new vertex at a time.

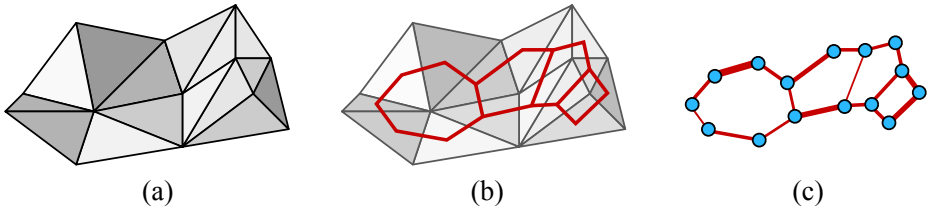

Fig. 1. Steps of the graph creation. From the initial mesh (a) the dual graph is built creating a vertex for each face and connecting adjacent faces (b). Each edge of this graph is then weighted according to the dot product between normals of the connected faces.

2.1 Graph Creation

As for any graph-based mesh segmentation approach our first task is the definition of an apt graphical model for the surface that will be processed. As shown in Fig. 1 each node of this graph corresponds to the a triangle of the mesh. There are no geometrical relations between these node and the absolute position of the triangles in space. For this reason we do not need any attribute on the graph nodes. By converse, we are interested in the relations between adjacent faces, thus we are going to define a scalar attribute for the graph edges. Specifically, we want to assign to each edge a weight that is monotonical with the "effort" required to move between the two barycenters of the faces. This effort should be higher if the triangles exhibit a strong curvature with a short distance between their centers and it should be low if the opposite happens. To this extent, given two nodes of the graph associated to faces i and j, we define the weight between them as:

$$\omega(i,j) = \frac{1 - < n_i, n_j >}{|p_i - p_j|} \tag{1}$$

where $\bar{p} = (p_1, p_2 ... p_k)$ is the vector of the barycenters of the faces and $\bar{n} = (n_1, n_2 ... n_k)$ are the respective normals. $< \cdot, \cdot >$ denotes the scalar product and $|\cdot|$ the Euclidean norm.

In Fig. 1 (c) edge weight is represented by using a proportional width in the drawing of the line between two nodes. It can be seen how edges that connect faces with stronger curvatures exhibit larger weight.

2.2 Seeding and Greedy Growing

Once the weighted graph has been created, the segmentation can happen. In our framework the surface is segmented starting from one or more hints provided by the user. This human hint expresses a binary condition on the mesh by assigning a small fraction of all the nodes to a set called *user selected green nodes* and another small portion to a set called *user selected red nodes*. We call *green nodes* the faces (nodes) belonging to the segment of interest and *red nodes* the ones

Algorithm 2.1: GROW($graph, userSelectGreenNodes, userSelectRedNodes$)

$greenNodes \leftarrow \emptyset$
$redNodes \leftarrow \emptyset$
$unassignedNodes \leftarrow \emptyset$
$seeds \leftarrow \emptyset$

for each $n \in graph.nodes$
do { **if** $n \in userSelectGreenNodes$
  then $seeds = seeds \cup \{< n, green, 0 >\}$
if $n \in userSelectRedNodes$
  then $seeds = seeds \cup \{< n, red, 0 >\}$
$unassignedNodes = unassignedNodes \cup \{n\}$
while $seeds \neq \emptyset$
do { $s \leftarrow \arg\min_{seed} seed.w, seed \in seeds$
if $s \in unassignedNodes$
  then { **if** $s.type = green$
    then $greenNodes = greenNodes \cup \{s.n\}$
  if $s.type = red$
    then $redNodes = redNodes \cup \{s.n\}$
  $unassignedNodes = unassignedNodes \setminus \{s.n\}$
  for each $m \in graph.neighbors(s.n)$
    do { **if** $m \in unassignedNodes$
      then $seeds = seeds \cup \{< m, s.type, \omega(s.n, m) >\}$

Fig. 2. The simple, yet effective, algorithm proposed to iteratively expand the initial user-specified seeds to cover the whole mesh.

that are not belonging to it, regardless of the fact that those nodes have been manually or automatically labeled. The proposed algorithm distributes all graph nodes in the *green nodes* and *red nodes* sets in a greedy way.

We define a seed as triple $< n, t, w >$ where n is the graph node referred by this seed, t is a boolean flag that indicates if n has to be added to green or red nodes, w is a positive value in $\mathbb{R}^+$. At the initialization step, for each initial green and red node selected by the user, a seed is created and inserted into a priority queue with an initial weight value $w = 0$. All nodes are also added to the *unassigned nodes* set. At each step, the seed $< n, t, w >$ with lowest value of w is extracted from the priority queue and its referred node n is added to *green nodes* or *red nodes* according to the seed's t flag. The node is also removed from *unassigned nodes* to ensure that each node is evaluated exactly once during the execution of the algorithm. For each node $n' \in$ *unassigned nodes* connected to n in the graph, a new seed $< n', t' = t, w' = \omega(n, n') >$ is created and added into the queue. It has to be noted that it is not a direct consequence of such insertion

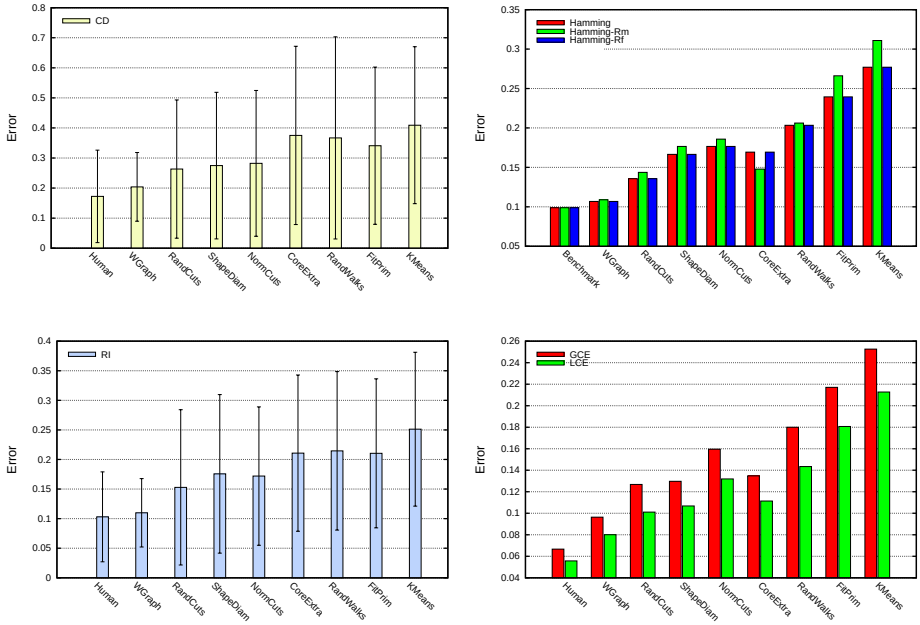

Fig. 3. Benchmark evaluation of our approach (WGraph) with respect to manual segmentation (Human) and other segmentation techniques. The used metrics are respectively the Cut Discrepancy (CD), the Hamming Distance (Hamming), the Rand Index (RI) and the Consistency Error (CE). See the text for details.

that the final type of n' (either green or red) is determined by the type t' of this seed. At any time multiple seeds referring the same node can exist in the queue, with the only condition that a node type can be set only once. During the execution of algorithm either the region of green nodes and the region of red ones expands towards the nodes that would require less weight to be reached. Once all nodes in the same connected component are visited, the result of this assignment is shown to the user who can either refine his initial hint or accept the proposed segmentation. Of course the procedure can be iterated to obtain an hierarchical segmentation. In any condition, the algorithm will run in $O(N)$ time since, with the described greedy approach, each node is visited once.

3 Experimental Validation

3.1 Quantitative Evaluation

In order to assess the results obtained by the proposed technique in a quantitative manner some kind of benchmark is needed. We chose to adopt the dataset and metrics proposed in [21]. Namely, the dataset consists in 380 surface meshes,

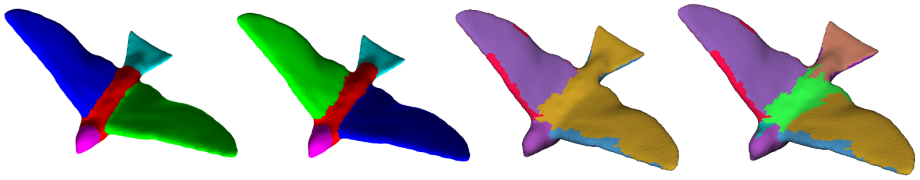

Fig. 4. Qualitative comparisons between the segmentation obtained with our method (second column) and with a semi-supervised Fitting Primitives respectively with 5 segments (third column) and 10 segments (fourth column). In the first column the ground truth segmentation is shown.

organized in 19 different object categories. A ground truth of 4300 manually generated segmentations is supplied. Since the authors provided both the dataset and the code for running the benchmark, we replicated some of their experiments and added our approach to the set of algorithms to be tested against the ground truth. The evaluation adopts four different metrics. The first one is the *Cut Discrepancy*, that sums the distances from points along the cuts in the obtained segmentation with respect to the closest cuts in the ground truth and vice-versa (CD in Fig. 3). The idea is to measure how well the segment boundaries overlap with the ground truth. The second metric is the Hamming Distance, that measures the overall region-based difference between two segmentations. In particular we evaluate two directional Hamming Distances, the missing rate (Hamming R_m in Fig. 3) and the false alarm rate (Hamming R_f in Fig. 3). In addition also the average between the two is calculated (Hamming R_f in Fig. 3). The third metric is the Rand Index (RI in Fig. 3), that accounts for the likelihood that two triangles belong both to the same o to different clusters in two segmentations. Finally also two Consistency Error metrics are evaluated to measure a triangle-based compatibility between segments that is neutral to differences in hierarchical granularity. Specifically the Global Consistency Error (GCE in Fig. 3), that forces all local refinements to be in the same direction, and the Local Consistency Error (LCE in Fig. 3), that allows for different refinement directions in different parts of the same object (refer to [21] for more details about these metrics). The compared method were Randomized and Normalized Cuts [19], Shape Diameter Functions [16], Core Extraction [14], Random Walks [18], Fitting Primitives [17] and KMeans [11]. While most of these method are not supervised, some required parameters such as the number of segments to extract or initial seeds. For each approach we used the optimal parameter set suggested in [21]. In addition also a set of totally human-supervised segmentation is available in the benchmark. From the results shown in Fig. 3, it is apparent that the proposed method outperforms all the compared approaches. Of course this is somewhat expected since we use an initial set of hints supplied by the user. However the algorithm was always fed with less than 10 seeds, resulting in just a few seconds of operator time. It is interesting to observe that, even with this limited user interaction, the performance obtained is on par with the fully supervised human segmentation.

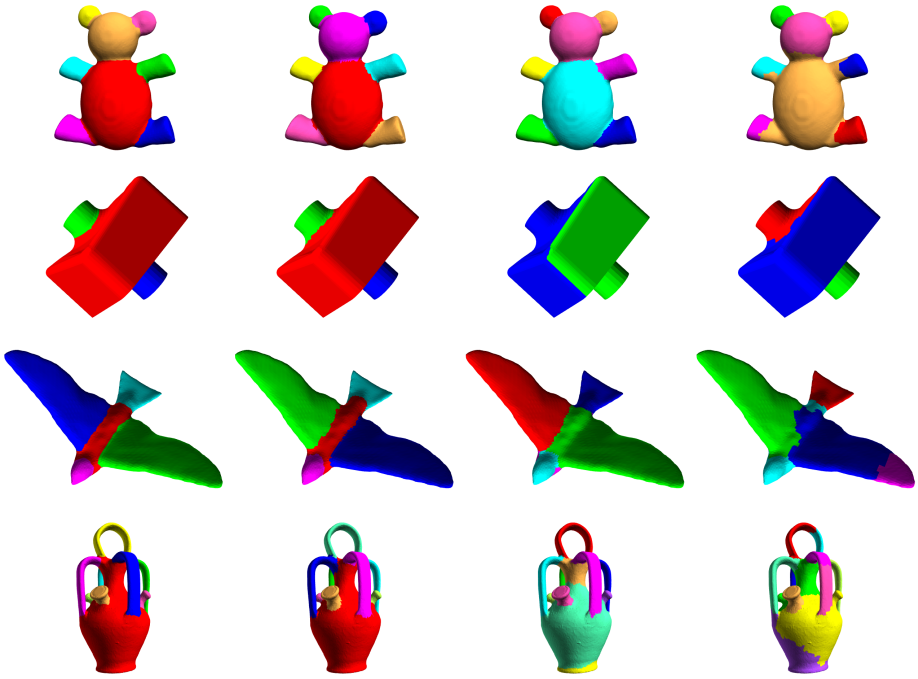

Fig. 5. Qualitative comparisons between the supervised segmentation obtained with our method (second column) and with other state-of-the art unsupervised methods. Respectively Randomized Cuts (third column) and Normalized Cuts (fourth column). In the first column the ground truth segmentation is shown.

3.2 Qualitative Evaluation and Running Time

In addition to the quantitative results provided by the benchmark we also present some sample segmentation images in order to give an insight about how the different methods actually compare from a qualitative point of view. In Fig. 4 we compare the fully supervised ground truth segmentation (first column) with the result obtained by our method setting just 2 hint seeds for each segment. We tried to replicate this result using the interactive Fitting Primitive tool available from [17], however were not able to obtain a proper segmentation with only 5 clusters (third column), while adding more clusters led the method to over-segmentation (fourth column). In Fig. 5 comparisons are made with the best performing automatic approaches. While objects with sharp edges are usually easily segmented by all methods (teddy bear in the first row), some synthetic shapes can lead to failures for Randomized Cuts and a slight imprecision for Normalized Cuts (CAD model in the second row). Organic shapes (such as the bird in the third row and the vase on the fourth) can make it difficult for automatic algorithm to spot semantically relevant edges.

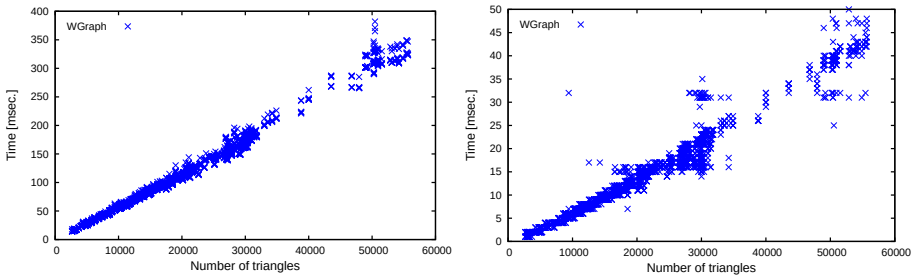

Fig. 6. Evaluation of the time required respectively for creating the weighted graph and to perform a single segmentation on an object. Both scatter-plots relate the execution time to the number of triangles in the mesh.

Regarding the execution time (Fig. 6), the graph building step is the more time consuming, but it can always be performed in less than a second even for large models. The segmentation step itself is very fast and it typically requires less than 50 milliseconds.

4 Conclusions

In this paper we introduced a simple yet effective segmentation procedure for 3D objects and surfaces. While our method requires an initial set of user hints, results that are on par with fully supervised segmentations can be obtained even with a very limited amount of seeds placed without special care by the user. Moreover the time required to perform a segmentation is in the order of few millisecond with meshes that count tens of thousands of vertices. This allows for the inclusion of the method in tools that exploit real-time interactive use. Finally, the proposed growing algorithm is very simple and can be easily implemented.

References

1. Kokkinos, I., Maragos, P.: Synergy between Object Recognition and Image Segmentation Using the Expectation-Maximization Algorithm. Pattern Analysis and Machine Intelligence, IEEE Transactions on **31** (2009) 1486–1501
2. Ferrari, V., Tuytelaars, T., Gool, L.: Simultaneous object recognition and segmentation from single or multiple model views. Int. J. Comput. Vision **67** (2006) 159–188
3. Courtial, A., Vezzetti, E.: New 3d segmentation approach for reverse engineering selective sampling acquisition. The International Journal of Advanced Manufacturing Technology **35** (2008) 900–907 10.1007/s00170-006-0772-3.
4. Kim, H.C., Hur, S.M., Lee, S.H.: Segmentation of the measured point data in reverse engineering. The International Journal of Advanced Manufacturing Technology **20** (2002) 571–580 10.1007/s001700200193.

5. Colombari, A., Fusiello, A., Murino, V.: Segmentation and tracking of multiple video objects. Pattern Recognition **40** (2007) 1307–1317
6. Lafarge, F., Keriven, R., Brédif, M., Hiep, V.H.: Hybrid multi-view reconstruction by jump-diffusion. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Franscico, U.S., IEEE (2010) 350–357
7. Baillard, C., Zisserman, A.: Automatic reconstruction of piecewise planar models from multiple views. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on **2** (1999) 2559–2566
8. Mian, A.S., Bennamoun, M., Owens, R.: Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. Pattern Anal. Mach. Intell. **28** (2006) 1584–1601
9. Bariya, P., Nishino, K.: Scale-hierarchical 3d object recognition in cluttered scenes. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010. (2010) 1657–1664
10. Wang, Y., Gu, X., Hayashi, K.M., Chan, T.F., Thompson, P.M., Yau, S.T.: Surface parameterization using riemann surface structure. In: Proceedings of the Tenth IEEE International Conference on Computer Vision - Volume 2. ICCV '05, Washington, DC, USA, IEEE Computer Society (2005) 1061–1066
11. Shlafman, S., Tal, A., Katz, S.: Metamorphosis of polyhedral surfaces using decomposition. In: Eurographics. Volume 21. (2002) 219–228
12. Moumoun, L., Chahhou, M., Gadi, T., Benslimane, R.: 3d hierarchical segmentation using the markers for the watershed transformation. International Journal of Engineering Science and Technology **2** (2010) 3165–3171
13. Katz, S., Tal, A.: Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. **22** (2003) 954–961
14. Katz, S., Leifman, G., Tal, A.: Mesh segmentation using feature point and core extraction. The Visual Computer **21** (2005) 649–658
15. Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J.: Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica **38** (2003) 227–248
16. Shapira, L., Shamir, A., Cohen-Or, D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. **24** (2008) 249–259
17. Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. **22** (2006) 181–193
18. Lai, Y.K., Hu, S.M., Martin, R.R., Rosin, P.L.: Fast mesh segmentation using random walks. In: Proceedings of the 2008 ACM symposium on Solid and physical modeling. SPM '08, New York, NY, USA, ACM (2008) 183–191
19. Golovinskiy, A., Funkhouser, T.: Randomized cuts for 3D mesh analysis. ACM Transactions on Graphics (Proc. SIGGRAPH ASIA) **27** (2008)
20. Zhang, X., Li, G., Xiong, Y., He, F.: 3d mesh segmentation using mean-shifted curvature. In Chen, F., Jttler, B., eds.: Advances in Geometric Modeling and Processing. Volume 4975 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 465–474
21. Chen, X., Golovinskiy, A., Funkhouser, T.: A benchmark for 3D mesh segmentation. ACM Transactions on Graphics (Proc. SIGGRAPH) **28** (2009)