Pattern Recognition Letters 33 (2012) 2057–2064

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A graph-based technique for semi-supervised segmentation of 3D surfaces

Filippo Bergamasco, Andrea Albarelli *, Andrea Torsello

Dipartimento di Scienze Ambientali, Informatica e Statistica – Università Ca' Foscari Venezia, Italy

ARTICLE INFO

Article history:
Available online 3 April 2012

Keywords:
3D segmentation
Directional curvature metric
Greedy label propagation

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, but rather, they are mostly designed to allow inexperienced users to acquire surfaces and whole objects easily and independently. In this scenario, the demand for automatic or semi-automatic algorithms for 3D data processing is increasing. In this paper we address the task of segmenting the acquired surfaces into perceptually relevant parts. Such a problem is well known to be ill-defined both for 2D images and 3D objects, as even with a perfect understanding of the scene, many different and incompatible semantic or syntactic segmentations can exist together. For this reason recent years have seen a great research effort into semi-supervised approaches, that can make use of small bits of information provided by the user to attain better accuracy. We propose a semi-supervised procedure that exploits an initial set of seeds selected by the user. In our framework segmentation happens by propagating part labels over a weighted graph representation of the surface directly derived from its triangulated mesh. 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 a standard database of scanned 3D objects show its effectiveness even with moderate user supervision.

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 (Kokkinos and Maragos, 2009; Ferrari et al., 2006), reverse engineering (Courtial and Vezzetti, 2008; Kim et al., 2002), parts retrieval (Antini et al., 2005) and even tracking (Colombari et al., 2007). 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, i.e. by 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. Obviously, 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 (Lafarge et al., 2010; Baillard and Zisserman, 1999), while completed models can be used for direct 3D object recognition with resilience to clutter (Mian et al., 2006) or invariance to scale (Bariya and Nishino, 2010). Finally, another important application that needs surface decomposition is the angle and distance-preserving piecewise parametrization needed to apply textures to objects (Wang et al., 2005).

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 et al. propose to use a variation of K-means to group the triangles of the mesh into clusters (Shlafman et al., 2002). This is a quite direct adaptation: first the user specify the desired number of clusters (k), then the process starts 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 suggest the use the Watershed principle (Moumoun et al., 2010) on a hierarchical transformation of connected faces structure based on

* Corresponding author. Tel.: +39 042 2348465; fax: +39 041 2348419.
E-mail address: albarelli@unive.it (A. Albarelli).

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.03.015

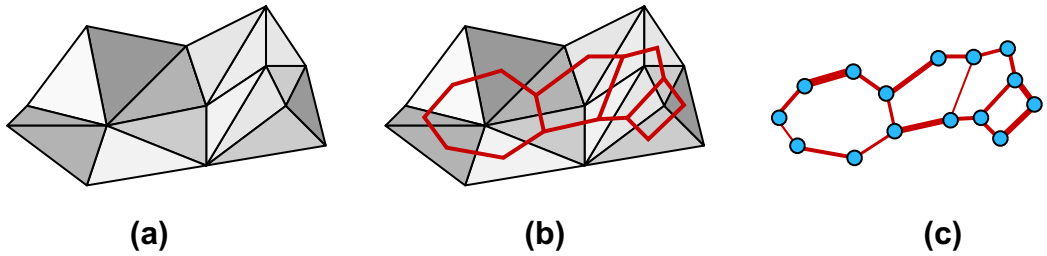

Fig. 1. Steps of the graph creation process. From the initial mesh (a) the dual graph is built creating a vertex for each face and connecting each pair of adjacent faces. (b) Finally, each edge of this graph is then weighted according to the dot product between the normals of the connected faces.

the principal curvature. An interesting perspective on 3D segmentation is supplied by Podolak et al. (2006), whose solution exploits the relation between mesh elements and the symmetry planes of the whole object.

Katz and Tal (2003) 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 (Katz et al., 2005) that performs three steps: the transformation of the surface into a pose insensitive representation by means 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. (2003) 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. (2008) 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. (2006a) 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. (2008) 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 clusters is obtained. This is done following an order based on the relative lengths of the intersections and total perimeters of adjacent segments.

Golovinskiy and Funkhouser (2008) use both normalized cuts and randomized cuts. In a similar manner to Attene et al. (2006a), 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.

More recently Berretti et al. (2009) introduced an approach based on Reeb graphs that is motivated by perceptual principles and supports identification of salient object protrusions.

For a recent review and comparison of the most well known mesh segmentation techniques see Attene et al. (2006) or Shamir (2008).

In this paper we introduce a novel graph-based segmentation approach aimed at achieving high speed on triangulated meshes. The proposed method is designed to exploit the connected graph induced by the mesh itself and thus can not be applied directly to unstructured or partially structured 3D data such as point clouds or sets of range images. However many segmentation techniques exist to deal with such scenarios (Golovinskiy and Funkhouser, 2009; Zou and Ye, 2007; Min and Bowyer, 2005). The described algorithm adopts a greedy label propagation approach based only on easily computable curvature information along the edges of the mesh. 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 (Golovinskiy and Funkhouser, 2008; Lai et al., 2008; Zhang et al., 2008). 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 (see Fig. 2), 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.

2.1. Graph creation

As for any graph-based mesh segmentation approach our first task is the definition of an apt graph model for the surface that will

Algorithm 2.1: PROPAGATE($graph, userSelectedPositiveNodes, userSelectedNegativeNodes$)

$PositiveNodes \leftarrow \emptyset$
$NegativeNodes \leftarrow \emptyset$
$unassignedNodes \leftarrow \emptyset$
$seeds \leftarrow \emptyset$

for each $n \in graph.nodes$
do
- **if** $n \in userSelectedPositiveNodes$
  - **then** $seeds = seeds \cup \{< n, positive, 0 >\}$
- **if** $n \in userSelectedNegativeNodes$
  - **then** $seeds = seeds \cup \{< n, negative, 0 >\}$
- $unassignedNodes = unassignedNodes \cup \{n\}$

while $seeds \neq \emptyset$
do
- $s \leftarrow \arg\min_{seed} seed.w, seed \in seeds$
- **if** $s \in unassignedNodes$
  - **then**
    - **if** $s.type = positive$
      - **then** $positiveNodes = positiveNodes \cup \{s.n\}$
    - **if** $s.type = negative$
      - **then** $negativeNodes = negativeNodes \cup \{s.n\}$
    - $unassignedNodes = unassignedNodes \setminus \{s.n\}$
    - **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.

be processed. Here we opt to use the dual graph of the triangulated mesh, where nodes correspond to the triangles of the mesh, and vertices represent adjacency information. See Fig. 1 for an illustration of the dual graph construction. With this representation all the relevant information rests in the relative position and orientation of the triangles, thus we do not need any attribute on the graph nodes, but we keep a scalar attribute measuring the compatibility of two triangles for belonging to a common part. Specifically, we want to assign to each edge of the dual graph a weight that is monotonically increasing with the "effort" required to move between the two barycenters of the faces. This effort should be higher if the triangles exhibit a *directional curvature*, i.e. strong variation in surface normal with a short distance between their centers. Conversely, the effort should be low if the opposite happens. To this end, given two nodes of the dual graph associated to faces i and j, we define the weight between them as:

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

where $\bar{p} = (p_1, p_2, \ldots, p_k)$ is the vector of the barycenters of the faces and $\bar{n} = (n_1, n_2, \ldots, n_k)$ are the respective normals. $\langle \cdot, \cdot \rangle$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 label assignment

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 positive nodes* and another small portion to a set called *user selected negative nodes*. We call *positive nodes* the faces (nodes) belonging to the segment of interest and *negative nodes* the ones that are not belonging to it, regardless of the fact that those nodes have been manually or automatically labeled. Our goal is to assign all the unlabeled nodes in the graph to the *positive nodes* and *negative nodes* sets. We do this greedily by assigning to each triangle a label equal to the closest seed according to a metric derived from the directional curvature metric ω. In fact, the directional curvature metric ω defines a Riemannian metric tensor over the triangulated surface. This metric tensor can be extended to measure the distance between any two triangles by integrating it over an optimal path. Let $\mathcal{M}$ be a manifold with metric tensor $T_{\mathcal{M}}$, and let $\delta : [0,1] \rightarrow \mathcal{M}$ be a curve in $\mathcal{M}$, the L_p norm of the path according to the metric in $\mathcal{M}$ is

$$PL_p^{T_{\mathcal{M}}}(\delta) = \left(\int_0^1 (\|delta'(t)\|_{T_{\mathcal{M}}})^p dt \right)^{\frac{1}{p}}$$

Similarly, we define the PL_p distance between two points $p, q \in \mathcal{M}$ as the infimum of $PL_p^{T_{\mathcal{M}}}(\delta)$ where $\delta(0) = p$ and $\delta(1) = q$. The PL_p distance is a proper generalization of the Riemannian distance $d_{\mathcal{M}}$, since it reverts to it when $p = 1$, i.e. $PL_1(p,q) = d_{\mathcal{M}}(p,q)$.

These can be approximated over the dual graph of the discrete triangulation of the surface as:

$$PL_p^{\omega}(\delta) = \left(\sum_{i=1}^{l-1} \omega(\delta_i, \delta_{i+1})^p\right)^{\frac{1}{p}}$$

where $\delta \in \mathbb{R}^l$ and δ_i is adjacent to $\delta i + 1$ for all $i = 1,\ldots,l-1$, and

$$PL_p^{\omega}(i,j) = \min_{\delta} PL_p^{\omega}(\delta)$$

where the minimization is over the $\delta \in \mathbb{R}^l$ for which $\delta_1 = i$ and $\delta_l = j$.

Taken to the limit the PL_∞ norm assumes the value of the maximum directional curvature over the optimal path, and in general, as the parameter p increases, the PL_p distance becomes less sensitive to uniform areas and more sensitive to isolated variations in the metric as expected when crossing part boundaries. This property was already used in (Torsello et al., 2007) for generating a robust boundary-crossing measure for pairwise image segmentation. In this paper we propose to greedily assign to each unlabeled node the label of the seed with smaller PL_∞ distance, thus the segments form generalized Voronoi patches around the user provided seed. While in general the PL_p norm is rather computationally intensive to obtain, we can efficiently assign labels according to the PL_∞ norm using a greedy propagation approach reminiscent of fast marching methods.

2.3. Greedy label propagation

We define a seed as triple $\langle n,t,w\rangle$ where n is the graph node referred by this seed, t is a boolean flag that indicates if n has to be added to positive or negative nodes, w is a positive value in $\mathbb{R}^+$. At the initialization step, for each initial positive and negative 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 $\langle n,t,w\rangle$ with lowest value of w is extracted from the priority queue and its referred node n is added to *positive nodes* or *negative 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 $\langle n', t' = t, w' = \omega(n,n')\rangle$ is created and added into the queue. It has to be noted that it is not a direct consequence of such insertion that the final type of n' (either positive or negative) 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 positive nodes and the region of negative 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 a hierarchical segmentation by setting additional seeds in the obtained segments. In any condition, the algorithm will iterate linearly in the unassigned labels yielding a time complexity of $O(nf(n))$ where $f(n)$ is the complexity of the *minimum extraction* operation from a priority queue, which can be log n with a normal heap as in our implementation, or can be improved to $\sqrt{\log n}$ using fusion trees (Fredman and Willard, 1994).

3. Experimental validation

The evaluation of the result produced by a segmentation technique has proven to be a difficult task for a number of different reasons. The first hurdle is related to the fact that the segmentation problem itself tends to be elusive with respect to a clear definition. In fact, it is not always clear if the goal of the process is to isolate different object parts by virtue of their semantic or rather split them by the characteristics of their surface boundaries or even by clustering them in surface patches that are homogeneous with respect to some geometric primitive. Additionally, even benchmark based on human made segmentation are often not perfectly consistent, since different operators tend to produce different segmentations.

For this reason a number of different metrics and benchmarks have been proposed over time (Gerig et al., 2008; Corsini et al., 2007; Benhabiles et al., 2009; Benhabiles et al., 2010). In this paper we chose to adopt the dataset and metrics proposed by Chen et al. (2009). This choice was driven by three factors: the richness of the dataset in terms of number of meshes and categories, the presence of a ground-truth segmentation operated by a number of different individuals and, finally, the availability of a large set of results obtained by running most modern 3D mesh segmentation techniques with this benchmark.

3.1. Quantitative evaluation

In order to assess the results obtained by the proposed technique we first performed a set of quantitative experiments over the dataset presented by Chen et al. (2009).

Namely, the dataset consists in 380 surface meshes, organized in 19 different object categories. A ground truth of 4300 manually generated segmentations is supplied. The resolution of the meshes is relatively low as the vertex count ranges from a minimum of about 3000 to a maximum of about 30000 vertices. 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 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 or 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 (Chen et al., 2009) for more details about these metrics). The compared method were Randomized and Normalized Cuts (Golovinskiy and Funkhouser, 2008), Shape Diameter Functions (Shapira et al., 2008), Core Extraction (Katz et al., 2005), Random Walks (Lai et al., 2008), Fitting Primitives (Attene et al., 2006a) and K-Means (Shlafman et al., 2002).

While most of these methods 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 by Chen et al. (2009). In addition, 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 100

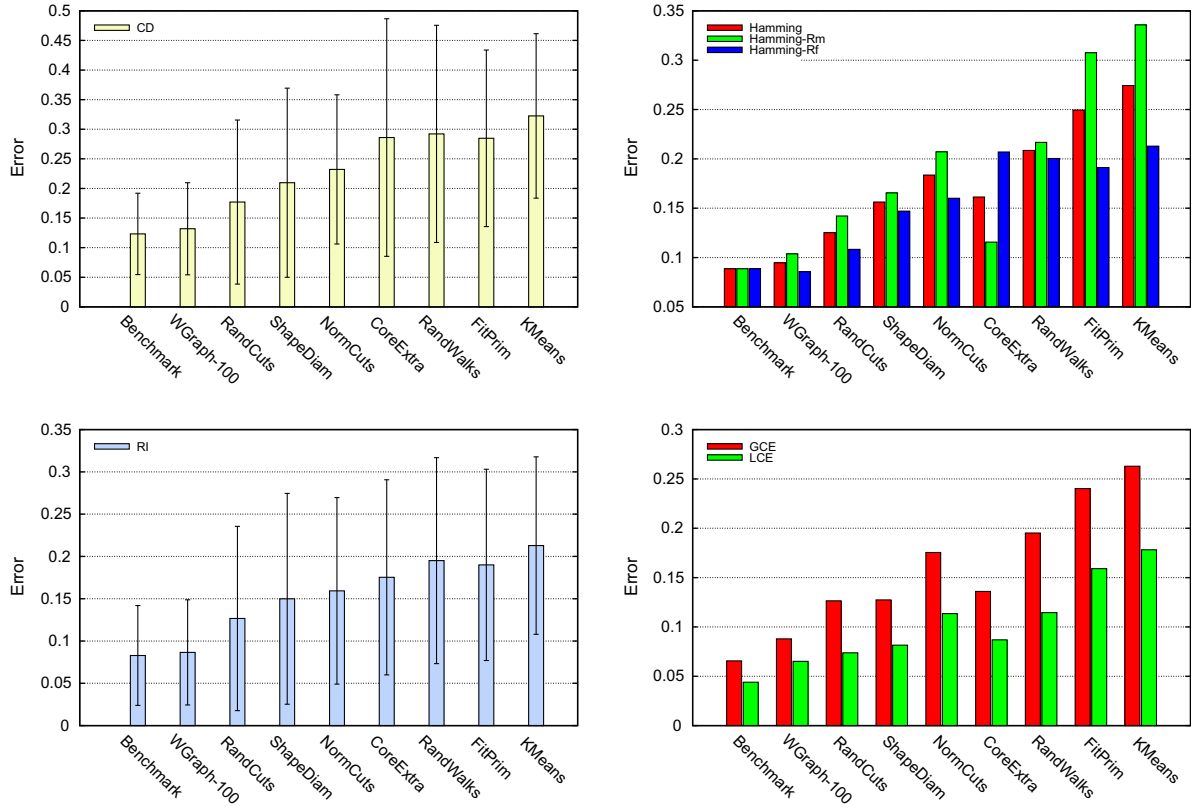

Fig. 3. Benchmark evaluation of our approach (WGraph) with respect to manual segmentation (Human) and other segmentation techniques. The used metrics are, respectively, the Cut Discrepancy (CD), the Hamming Distance (Hamming), the Rand Index (RI) and the Consistency Error (CE). See the text for details.

seeds. Since each click on our interface lays down about 10–15 hints all the initial seeding counts less than 10 selections, 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.

Given the importance of requiring as little user input as possible, we performed an additional set of quantitative experiments in order to study the relation that ties the number of seeds to the quality of the segmentation obtained. Specifically, we run the full benchmark several times, changing for each run the number of initial seeds assigned to the part to be segmented. The metrics used are the same adopted for the first batch of quantitative experiments. This way it is possible and meaningful to perform a direct comparison between the quality obtained for a given number of seeds and the results attained by other methods previously reported.

In Fig. 4 the results of this set of experiments can be seen. It is apparent from the results in Fig. 3 that even with as little as 10 seeds the segmentation, although improvable, is on par with most of the best methods. When more hints are introduced, as expected the quality gets better. We stopped our tests at 200 seeds since we felt that beyond this level the improvement were mostly related to human supervision rather than to the ability of the proposed algorithm to correctly localize the separation of different surfaces.

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. 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.

In Fig. 6 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 Attene et al. (2006a), however were not able to obtain a proper segmentation with only 5 clusters (third column), while adding more clusters led the method to oversegmentation (fourth column).

In Fig. 7 we show the influence of the initial seeding. Specifically, we performed the segmentation procedure on a cup-shaped object with the goal of splitting the handle from the cup itself. In the first row we deliberately placed two seeds very far apart, while in the second row they are placed as near as possible. In both cases the obtained segmentation is similar (if not identical) as can also be seen in the zoomed images. This is due to the fact that once the label propagation reaches the zone with strong curvature between the cup and the handle it stops until all the smooth areas are completely assigned to the seeds.

Regarding the complexity of the algorithm, both the graph creation and the segmentation are in the order of $O(nlog(n))$ since a computation that involves an access to and ordered queue is performed for each vertex. In practice, however, the size of such queue is rather limited most of the time and it does not grows as big as the total number of vertices. In order to show this we performed a last set of experiments that measure the execution time of both the graph creation and the segmentation step. This has been done using random meshes of different sizes.

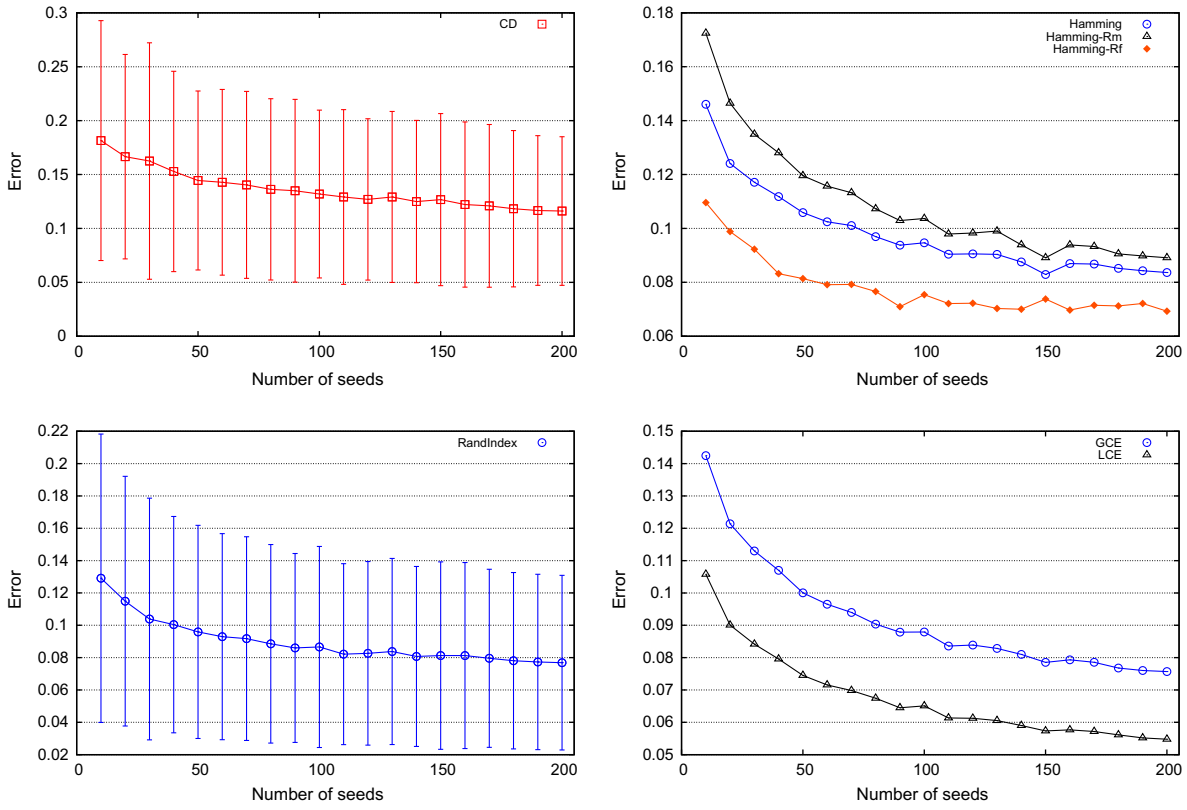

Fig. 4. The quality of our approach is evaluated with respect to the number of seeds (hints) supplied by the human operator during the supervised part of the process. See the text for details.

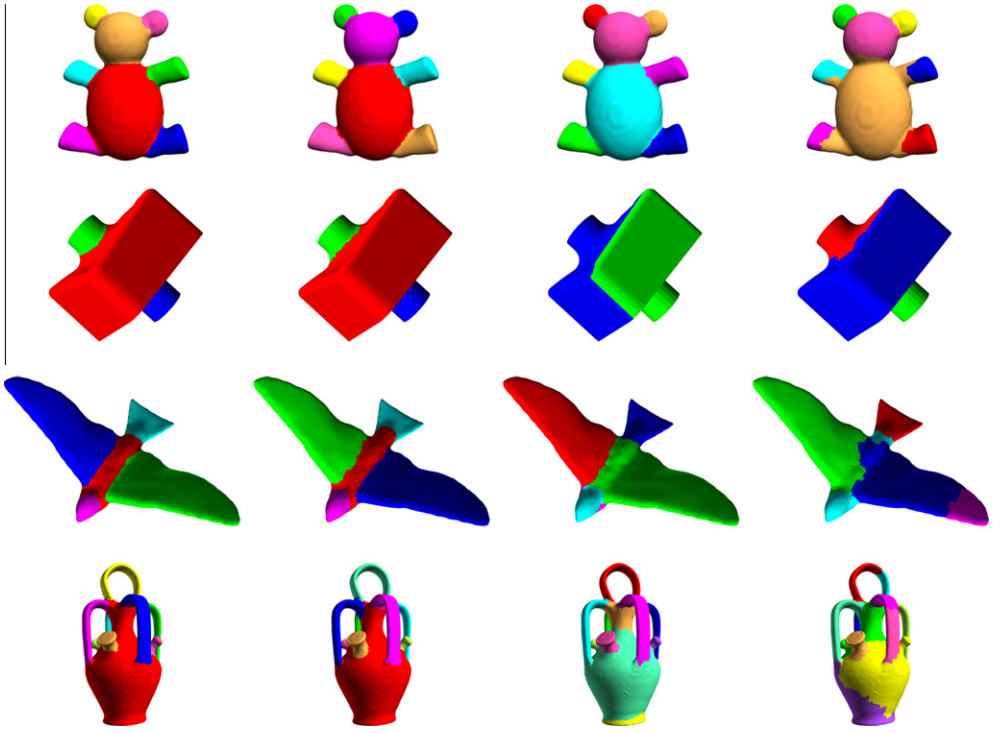

Fig. 5. Qualitative comparisons between the supervised segmentation obtained with our method (second column) and with other state-of-the art unsupervised methods. Respectively Randomized Cuts (third column) and Normalized Cuts (fourth column). In the first column the ground truth segmentation is shown.

The results of these experiments are shown in Fig. 8. It is apparent that 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

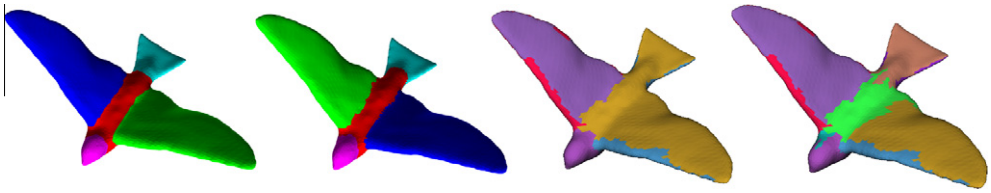

Fig. 6. Qualitative comparisons between the segmentation obtained with our method (second column) and with a semi-supervised Fitting Primitives respectively with 5 segments (third column) and 10 segments (fourth column). In the first column the ground truth segmentation is shown.

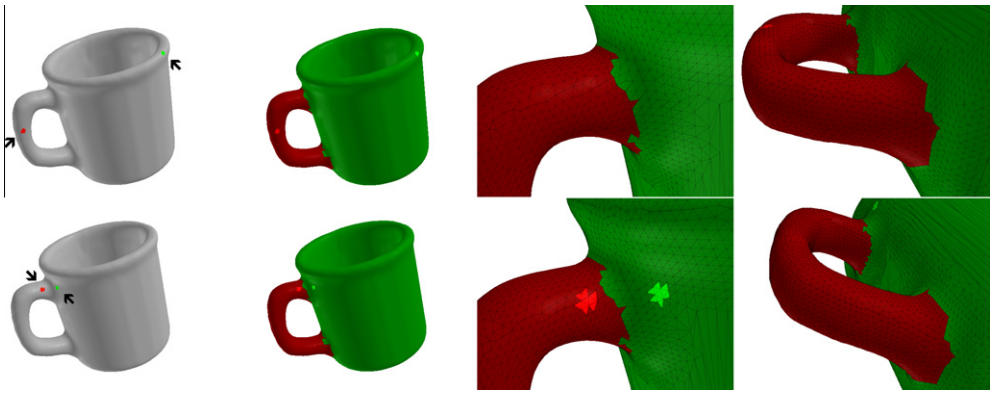

Fig. 7. Stability of the segmentation with respect to different placements of the initial seeds. In the first column the initial seeds are shown. In the remaining columns we present the obtained segmentations.

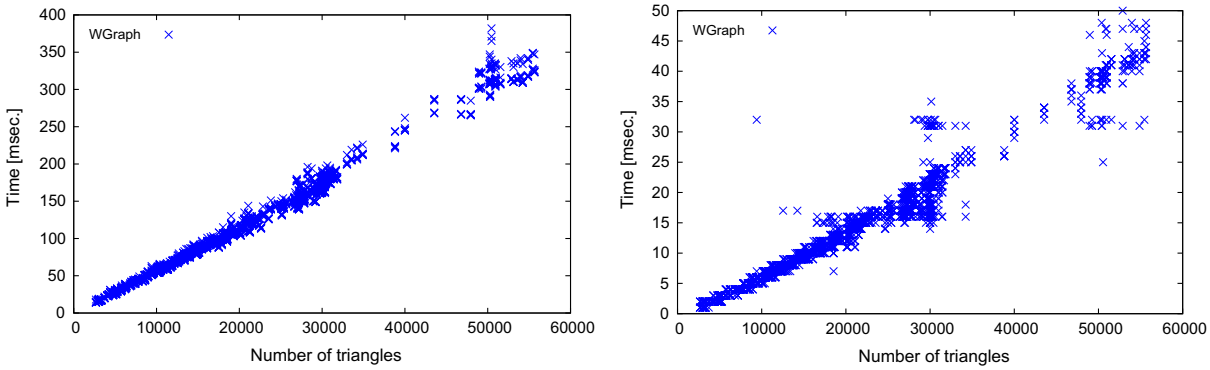

Fig. 8. Evaluation of the time required respectively for creating the weighted graph (first column) and for performing a single segmentation on an object (second column). Both the scatter-plots relate the execution time to the number of triangles in the mesh.

requires less than 50 ms. These measurements are obtained with a standard desktop PC equipped with a Core 2 processor running at 1.3 Ghz. Inside the tested range of mesh sizes the required time seems to be almost linear with respect to the number of triangles.

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 propagation algorithm is very simple and can be easily implemented.

References

Antini, G., Berretti, S., del Bimbo, A., Pala, P., 2005. 3D mesh partitioning for retrieval by parts applications. IEEE Int. Conf. on Multimedia and Expo, 1210–1213.

Attene, M., Falcidieno, B., Spagnuolo, M., 2006a. Hierarchical mesh segmentation based on fitting primitives. Vis. Comput. 22, 181–193.

Attene, M., Katz, S., Mortara, M., Patane, G., Spagnuolo, M., Tal, A., 2006b. Mesh segmentation-a comparative study. In: Proc. IEEE Internat. Conf. on Shape Modeling and Applications 2006. IEEE Computer Society, Washington, DC, USA, p. 7. url <http://dl.acm.org/citation.cfm?id=1136647.1136960>.

Baillard, C., Zisserman, A., 1999. Automatic reconstruction of piecewise planar models from multiple views. In: IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, vol. 2, pp. 2559–2566.

Bariya, P., Nishino, K., 2010. Scale-hierarchical 3D object recognition in cluttered scenes. In: IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2010, pp. 1657–1664.

Benhabiles, H., Vandeborre, J.P., Lavoué, G., Daoudi, M., Jun. 2009. A framework for the objective evaluation of segmentation algorithms using a ground-truth of human segmented 3D-models. In: SMI'09: Internat. Conf. on Shape Modeling and Applications, pp. 36–43.

Benhabiles, H., Vandeborre, J.P., Lavoué, G., Daoudi, M., 2010. A comparative study of existing metrics for 3D-mesh segmentation evaluation. Vis. Comput. 26 (12), 1451–1466.

Berretti, S., Del Bimbo, A., Pala, P., 2009. 3d mesh decomposition using reeb graphs. Image Vision Comput. 27, 1540–1554, <http://dl.acm.org/citation.cfm?id=1555006.1555132>.

Chen, X., Golovinskiy, A., Funkhouser, T., 2009. A benchmark for 3D mesh segmentation. ACM Trans. Graph. (Proc. SIGGRAPH) 28 (3), 73:1–73:12.

Colombari, A., Fusiello, A., Murino, V., 2007. Segmentation and tracking of multiple video objects. Pattern Recognition 40 (4), 1307–1317.

Corsini, M., Drelie Gelasca, E., Ebrahimi, T., Barni, M., 2007. Watermarked 3D mesh quality assessment. IEEE Trans. Multimedia 9 (2), 247–256.

Courtial, A., Vezzetti, E., 2008. New 3d segmentation approach for reverse engineering selective sampling acquisition. In: The Internat. J. Advanced Manufacturing Technology 35, pp. 900–907, <http://dx.doi.org/10.1007/s00170-006-0772-3>.

Ferrari, V., Tuytelaars, T., Van Gool, L., 2006. Simultaneous object recognition and segmentation from single or multiple model views. Internat. J. Comput. Vision 67, 159–188.

Fredman, M.L., Willard, D.E., 1994. Surpassing the information theoretic bound with fusion trees. J. Comput. Syst. Sci. 48 (3), 533–551.

Gerig, G., Jomier, M., Chakos, M., 2008. Valmet: A New Validation Tool for Assessing and Improving 3D Object Segmentation. In: MICCAI '01 Proc. 4th Internat. Conf. on Medical Image, pp. 516–523.

Golovinskiy, A., Funkhouser, T., 2008. Randomized cuts for 3D mesh analysis. ACM Trans. Graph. (Proc. SIGGRAPH ASIA) 27 (5).

Golovinskiy, A., Funkhouser, T., sep 2009. Min-Cut Based Segmentation of Point Clouds. In: IEEE Workshop on Search in 3D and Video (S3DV) at ICCV.

Katz, S., Leifman, G., Tal, A., 2005. Mesh segmentation using feature point and core extraction. Vis. Comput. 21 (8–10), 649–658.

Katz, S., Tal, A., 2003. Hierarchical mesh decomposition using fuzzy clustering and cuts. ACM Trans. Graph. 22, 954–961.

Kim, H.-C., Hur, S.-M., Lee, S.-H., 2002. Segmentation of the measured point data in reverse engineering. In: The Internat. J. Advanced Manufacturing Technology 20, pp. 571–580, <http://dx.doi.org/10.1007/s001700200193>

Kokkinos, I., Maragos, P., 2009. Synergy between object recognition and image segmentation using the expectation-maximization algorithm. IEEE Trans. Pattern Anal. Mach. Intell. 31 (8), 1486–1501.

Lafarge, F., Keriven, R., Brédif, M., Hiep, V.H., 2010. Hybrid multi-view reconstruction by jump-diffusion. In: The Twenty-Third IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2010. IEEE, San Franscico, US, pp. 350–357.

Lai, Y.-K., Hu, S.-M., Martin, R.R., Rosin, P.L., 2008. Fast mesh segmentation using random walks. In: Proc. 2008 ACM Symposium on Solid and physical modeling. SPM '08. ACM, New York, NY, USA, pp. 183–191.

Mian, A.S., Bennamoun, M., Owens, R., 2006. Three-dimensional model-based object recognition and segmentation in cluttered scenes. IEEE Trans. Pattern Anal. Mach. Intell. 28, 1584–1601.

Min, J., Bowyer, K., 2005. Improved range image segmentation by analyzing surface fit patterns. Comput. Vision and Image Understanding 97 (8), 242–258.

Mortara, M., Patanè, G., Spagnuolo, M., Falcidieno, B., Rossignac, J., 2003. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. Algorithmica 38, 227–248.

Moumoun, L., Chahhou, M., Gadi, T., Benslimane, R., 2010. 3D hierarchical segmentation using the markers for the watershed transformation. Int. J. Eng. Sci. Tech. 2, 3165–3171.

Podolak, J., Shilane, P., Golovinskiy, A., Rusinkiewicz, S., Funkhouser, T., 2006. A planar-reflective symmetry transform for 3D shapes. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers. ACM, New York, NY, USA, pp. 549–559.

Shamir, A., 2008. A survey on mesh segmentation techniques. Comput. Graph. Forum 27 (6), 1539–1556, <http://dx.doi.org/10.1111/j.1467-8659.2007.01103.x>.

Shapira, L., Shamir, A., Cohen-Or, D., 2008. Consistent mesh partitioning and skeletonisation using the shape diameter function. Vis. Comput. 24, 249–259.

Shlafman, S., Tal, A., Katz, S., 2002. Metamorphosis of polyhedral surfaces using decomposition. In: Eurographics. 21, 219–228.

Torsello, A., Di Gesù, M., Pelillo, M., 2007. Integrating boundary information in pairwise segmentation. In: Internat. Conf. on Image Analysis and Processing–ICIAP 2007. IEEE Computer Society, pp. 23–28.

Wang, Y., Gu, X., Hayashi, K.M., Chan, T.F., Thompson, P.M., Yau, S.-T., 2005. Surface parameterization using riemann surface structure. In: Proc. Tenth IEEE Internat. Conf. on Computer Vision - Volume 2. ICCV '05. IEEE Computer Society, Washington, DC, USA, pp. 1061–1066.

Zhang, X., Li, G., Xiong, Y., He, F., 2008. 3D mesh segmentation using mean-shifted curvature. In: Chen, F., Jüttler, B. (Eds.), Advances in Geometric Modeling and Processing, Lecture Notes in Computer Science, vol. 4975. Springer, Berlin/Heidelberg, pp. 465–474.

Zou, W., Ye, X., 2007. Multi-resolution Hierarchical Point Cloud Segmenting. In: Proc. Second Internat. Multi-Symposiums on Computer and Computational Sciences. IEEE Computer Society, Washington, DC, USA, pp. 137–143.