https://doi.org/10.2352/ISSN.2470-1173.2017.20.3DIPM-005

Digitized 3D mesh segmentation based on curvature analysis

S. Gauthier[1,2,*]***, W. Puech***[1]***, R. Bénière***[2]***, G. Subsol***[1]

[1] ***LIRMM Laboratory, UMR 5506, CNRS, University of Montpellier, Montpellier, France***
[2] ***C4W, Montpellier, France***
[*] ***silvere.gauthier@lirmm.fr***

Abstract

Today, it is increasingly frequent and easy to digitize the surface of real 3D objects. However, the obtained meshes are often inaccurate and noisy. In this paper, we present a method to segment a digitized 3D surface from a real object by analyzing its curvature. Experimental results - applied on real digitized 3D meshes - show the efficiency of our proposed analysis, particularly in a reverse engineering process.

Introduction

The availability of 3D scanners has increased the fast development of applications in Computed-Aided Design (CAD), reverse engineering, medical research and inspection. Many 3D processes need to segment an object as a preprocess. In the production line of manufactured objects, steps can be distributed to many partners, and some data can be missing at the end of the line. In reverse engineering, we try to reconstruct the continuous geometry of an object from a 3D mesh, which is discrete. This can lead to quality control or object modification issues for example. In fact, the mesh can be a discretization of a CAD object or a digitized one. To reconstruct the initial geometry, we must take into account the shape of the objects as they are related. But an object shape can be very complex, and the measured data is often noisy. So, we need robust 3D descriptors to accurately define the objects shape.

In previous works, geometry descriptors like curvatures [5, 8] allow us to deal with the 3D mesh shape. But curvatures are locally computed, while it is often necessary to characterize the shape globally. To do this, we can construct curvature distributions [4, 10] and analyze them.

In this paper, we propose a method based on the analysis of a digitized 3D mesh curvature histogram. We use the curvature approximation from Bénière *et al.* [2] who incorporates two other methods [5, 8]. Then, a distribution is constructed continuously by a kernel estimation from all of the curvature values. Finally, an accurate curvature distribution analysis is realized. In the distribution, we propose to search for peaks and valleys, and automatically compute segmentation thresholds. Indeed, curvature distribution approximately describes the object shape, so these thresholds allow us to extract points belonging to object edges. In the proposed approach, we use the curvature histogram to segment digitized 3D meshes.

This paper is organized as follow. We first present previous works in this topic. Then, we expose in detail our proposed method of 3D mesh segmentation. After, we apply it on digitized 3D surfaces of real objects and show that our curvature analysis hugely improves the obtained results, particularly in reverse engineering. Finally, we conclude and propose directions for future research.

Previous works

In this section, we first present previous research in 3D mesh segmentation. Next, we introduce curvature and previous works on 3D curvatures. Then, we develop previous research in distributions.

Segmentation

A segmentation is the partitioning of a digital image, a 3D mesh or a 3D point cloud in several regions, as illustrated in Fig. 1.

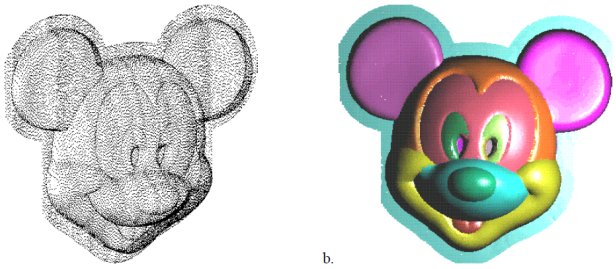

Figure 1. *Example of a segmentation: a) A 3D point cloud, b) Segmentation of the point cloud [12].*

Many different 3D mesh segmentation algorithms have been proposed in the literature [12, 13, 14], but each segmentation gives more or less good results depending on the chosen application. Most of the time, a segmentation brings together points with similar criteria. For example, the segmentation can be based on a waterfall, hierarchical clustering, iterative merging or remeshing. In our case, the best results are reached using curvatures, because in reverse engineering, features are extracted from curvature analysis. Some methods use curvatures to segment by discontinuities [3, 4] or clustering [7], but are often not robust enough around object edges or are designed for CAD meshes [9, 1]. Indeed, curvatures are often wrong around object edges because adjacent points can run over many different features. Moreover, we are searching for a fully automatic method, so we cannot use parameters like a cluster number or ask for user help.

Curvatures

Intuitively, curvature quantifies the deviation between a curve and a straight line, or between a surface and a plane in 3D. The curvature of a 2D curve at a point P equals the inverse of the osculating circle radius r at P. The osculating circle is the circular arc which best approximates the curve around P (Fig. 2.a).

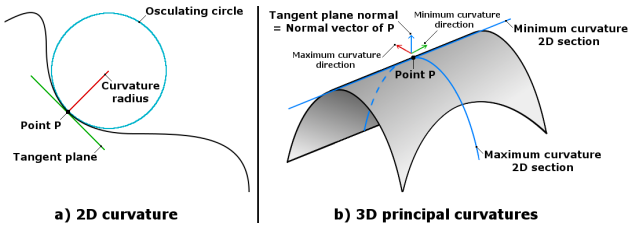

Figure 2. *Curvature representation on: a) 2D curves and b) 3D surfaces.*

On a 3D surface, an infinity of curvatures exists, each computed on a slice containing the normal vector of point P (Fig. 2.b). So, we need to distinguish particular curvatures. Principal curvatures are minimum and maximum curvatures. If principal curvatures are different, their matching planes are orthogonal. The intersections between these matching planes and the tangent plane at P define the principal directions. Mean curvature and Gaussian curvature equal respectively the mean and the product of principal curvatures. Dong and Wang [8] compute discrete curvatures with:

$$k_n(t) = \frac{< P_i - P, N_i - N >}{||P_i - P||^2}, \quad (1)$$

where P_i is a neighbor of P with a normal N_i, and $\vec{t}$ the projection of $\vec{P_iP}$ on the tangent plane of P. A linear regression is also applied on all discrete curvatures, but with a coefficient fixed to the maximum computed value.

Bénière *et al.* [2] compute discrete curvatures with formula 1 for each neighbor of P, and apply the linear regression of [5]. In our work, we prefer to use this approximation because it is more accurate. Indeed, equation 1 uses each neighbor independently and avoids some curve distortion. Moreover, fixing a linear regression coefficient when we do not know if we have computed the real maximum curvature is dangerous. Curvatures are often used to caracterize surface shape. So, for example, it is possible to analyze a shape to detect saliency or apply segmentation.

Distributions

There are many different types of distributions. For example, we can find probability, frequency or cumulative distributions. But overall, we can distinguish discrete and continous distributions. A discrete distribution is often represented by a histogram, as illustrated in Fig. 3.a. A histogram is a set of intervals with the same width, usually named "bins", which group homogeneous values. Each bin represents an occurence frequency of values from its interval. On the other hand, continous distributions are numerous because they correspond to mathematical models. Common models are Gaussian or Normal distributions (Fig. 3.b), which are couples of a mean value μ and a standard deviation σ.

In computer science, most of the value sets are discrete. So, to generalize methods for continuous spaces, it is necessary to approximate a discrete distribution with a continuous model. Most of the time, models are optimized from the measured data. Model optimization is a minimisation of the difference between the model and the data, updating the model until convergence. A common optimization algorithm is the Expectation-Maximisation (EM) [11]. For example, these distributions can be used in image processing to improve compression [10]. Some methods in 3D mesh processing also begin to use distributions, for example to apply segmentation [4] or extract object edges [6].

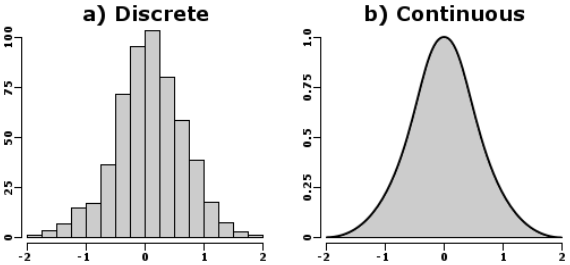

Figure 3. *a) Discrete and b) Continuous distribution samples.*

Proposed digitized 3D mesh segmentation

Our proposed method first computes a discrete curvature on each point of the initial 3D mesh. The method constructs a continuous estimated and normalized histogram for each curvature, then analyzes it. Next, a histogram analysis leads to an automatic curvature threshold computation, allowing us to extract object edges. Finally, our proposed method retrieves all disconnected regions using many processes. An overview of our proposed method is illustrated in Fig. 4.

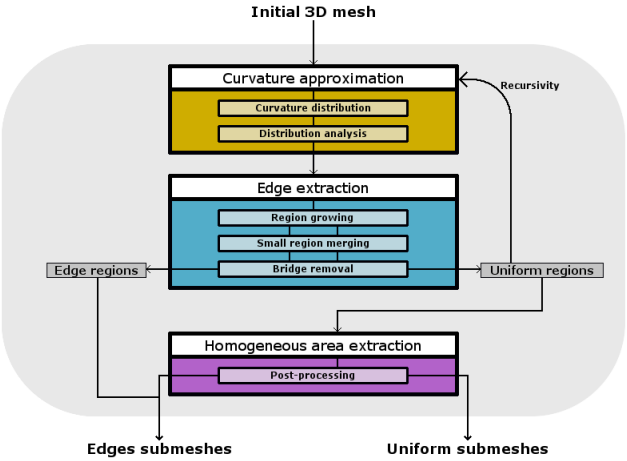

Figure 4. *Method overview.*

In fact, we provide region growing to retrieve disconnected regions, an artefact merging and bridge removal to handling inaccuracies leading to wrong regions, and finally a recursivity handling objects which are a composition of mechanical parts.

Probability curvature distribution

In probability and statistics, a probability distribution assigns a probability to each measurable subset of the possible outcomes of a random experiment, survey, or procedure of statistical inference. We can represent a probability distribution by a histogram. But to define the probability distributions for the simplest cases, we need to distinguish between discrete and continuous random variables. In the discrete case, we can easily assign a probability to each possible value. By contrast, when a random variable like curvature takes values from a continuum then, typically, probabilities can be nonzero only if they refer to intervals.

To approximate continuous curvatures on a discrete 3D mesh, we use the method from Bénière *et al.* [2]. In our case, meshes can be defined with different scales and can give different curvature ranges. So, to construct curvature histograms which can be compared between many objects, we must normalize curvature values with each mesh. Moreover, curvature values are real, so it is more suitable to estimate a continuous histogram, with kernel-estimation for example.

Normalized curvature

Histograms must have the same range to make a comparison. To homogenize histogram range, curvature values have to be normalized. For example, our method normalizes curvature values by multiplying them by the mean edge length of the mesh.

If the object is correctly meshed, we can deduce minimum and maximum possible curvature values, as illustrated Fig. 5. So we propose to limit the histogram range according to extreme possible values, since we normalize curvature by edge length.

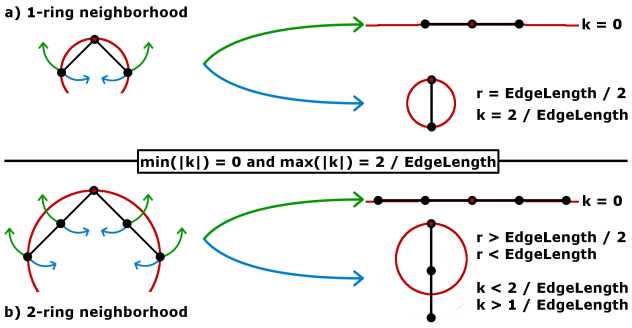

Figure 5. *Minimum and maximum possible for absolute curvature values.*

Kernel estimation

Curvature values are real, so it is more suitable to compute a histogram with a continuous estimation. Our method computes histograms with a kernel-type estimation. We chose a gaussian kernel because digitized meshes with only one feature often give gaussian-type curvature distributions. We compute the histogram with:

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i), \tag{2}$$

where x is the central value of a bin, $\hat{f}_h(x)$ is the quantity inside the bin, n is the number of points, x_i is the i^{th} point and h is the kernel standard deviation. The gaussian kernel is defined by:

$$K_h(x - x_i) = \frac{1}{h\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-x_i}{h}\right)^2}. \tag{3}$$

Each bin of the histogram is computed by centering the kernel on it. Then, kernel density estimation (KDE) is applied on each curvature value and added to the bin. So each bin is computed with a neighborhood defined by the kernel standard deviation. Fig. 6 shows an example of normalized mean curvature distribution with a kernel deviation $h = 0.01$.

This continuous estimation makes histograms less sensible to noise and bin number. Indeed, high frequency fluctuations are naturally smoothed by the kernel.

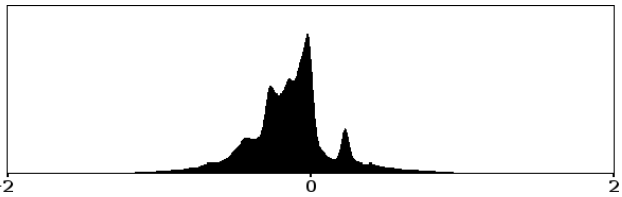

Figure 6. *Normalized mean curvature kernel-estimated histogram sample.*

Distribution analysis

Many caracteristics of a histogram can be useful to many applications. We can for example search for modal number and positions (named "peaks" here), pattern, sparsity and statistics. In our method, we essentialy provide robust peaks and valleys detection, and use those to compute segmentation thresholds.

To detect homogeneous curvature intervals, we need to detect peaks and valleys in the histogram, as illustrated in Fig. 7. Here, a peak defines a dominant curvature value and a couple of two consecutive valleys defines an homogeneous interval of curvature.

We begin by detecting robust peaks (Fig. 7.b). A peak is a bin with a higher probability than the two adjacent bins. A robust peak is a bin with the higher probability in a window. Thus, we can detect valleys between peaks (Fig. 7.c). For each adjacent peaks couple, we compute the line passing through the two peaks. Then, a valley is a bin with a higher distance to the line than the two adjacent bins. A robust valley is a bin with the higher distance in a window. Our method uses sliding windows of 10 bins for both peaks and valleys detection.

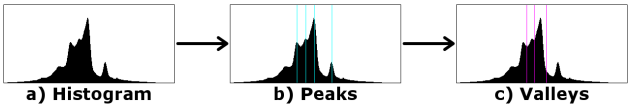

Figure 7. *Histogram peaks and valleys.*

For valleys in the extremities, we proceed differently. Because we have only one adjacent peak, we must define another extremity for the line. So we can compute mean Mu and standard deviation $Sigma$ of the first and last interval values (Fig. 8.a), and set the second extremity of the line as the matching bin with $Mu - 3 \times Sigma$ on the left and $Mu + 3 \times Sigma$ on the right (Fig. 8.b).

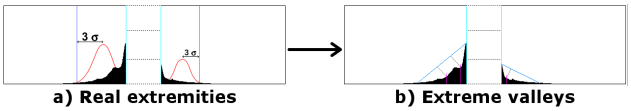

Figure 8. *Valley computing in the histogram extremities.*

Segmentation

In reverse engineering, a segmentation is often needed. In our case, we work on digitized meshes, with point coordinate inaccuracies and noise. To correctly segment those, we aim to extract object edges matching with intersections between geometric features. We can retrieve disconnected and homogeneous regions matching with these features. Finally, we also apply some post-processing and recursivity to improve results. Our proposed seg-

mentation, based on curvature histogram analysis, is completely automatic.

Edge extraction

Edges of an object are mesh areas caracterized by a high curvature. We propose to extract those edges with curvature histogram analysis. Indeed, high curvatures are on the extremities of curvature histograms. To detect and extract edges, we apply a thresholding on curvature values. Our method defines two thresholds, matching with the extreme left and right valleys of the histogram (Fig. 9). Points with curvature between the two thresholds are labelled "uniform", and others are labelled "edge".

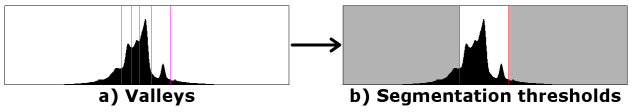

Figure 9. *Edge extraction.*

We can also realize multiple thresholding by taking each valley couple as thresholds (Fig. 10). This method directly isolates homogeneous intervals, but we need to be aware of potential curvature inaccuracy.

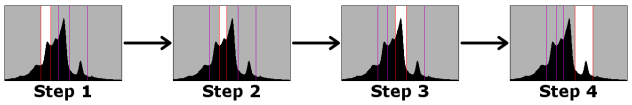

Figure 10. *Homogeneous areas extraction.*

To improve edge detection, we may have to remove some valleys, which are due to low sampling on some curvature intervals.

Region growing

After curvature thresholding, we can retrieve connected points by region growing. Our method retrieves triangles instead of points, but it is based on the same principle. A triangle type is defined by its dominant point type. Region growing is a common algorithm that groups similar adjacent elements: take a "seed" triangle and assign a unique ID (Fig. 11.a) and propagate seed ID on its neighbors (Fig. 11.b) until it reaches the borders (Fig. 11.c).

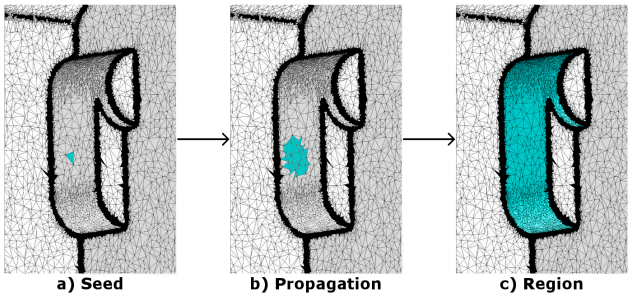

Figure 11. *Region growing algorithm.*

Post-processing

Because of digitization inaccuracy, some regions obtained by region growing can be wrong. To improve segmentation quality, we need to take those into account. Indeed, we often have many regions with an insufficient area, so we chose to merge them with adjacent regions. A small region in our method is defined by an area smaller than:

$$Threshold_{area} = 20 \times \frac{1}{n} \sum_{i=1}^{n} Area(T_i), \tag{4}$$

where n is the number of triangles and T_i the i^{th} triangle.

If a small region has only one adjacent region, it can be merged (Fig. 12). Otherwise, we prefer to remove it to avoid region connectivity modifications.

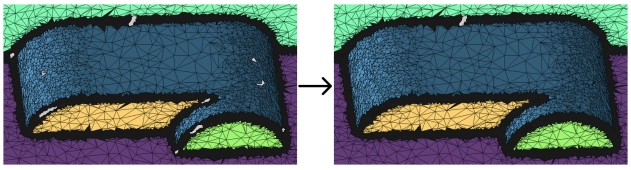

Figure 12. *Small regions (grey) are merged if they only have one adjacent region.*

Sometimes, because of digitization inaccuracies, region connectivity can be corrupted by small bridges (Fig. 13.a). We chose to remove these bridges to disconnect the regions into many subregions. To do this, we use an erosion-dilatation algorithm. First, the connected region is eroded many times and disconnected subregions are retrieved by region growing (Fig. 13.b and Fig. 13.c). Then, subregions are dilated as many times as the erosion (Fig. 13.d). Finally, we can merge bridges with only one adjacent region, and remove the others (Fig. 13.e).

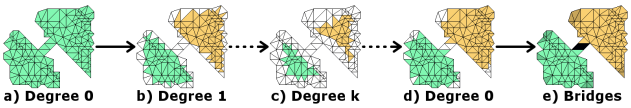

Figure 13. *Small bridges corrupting connectivy are removed.*

Recursivity

A mechanical object can be composed of many parts with different scales. In this case, it is not possible to compute a unique threshold to obtain optimal results. So in our method, we choose to apply recursive segmentation. In fact, we segment the input mesh, then segment again each submesh with the same method, until we obtain only one submesh after segmentation. Since our method is based on curvature histogram analysis, each submesh has a different histogram and we can detect object edges with many scales.

Experimental results

This section presents three meshes from different scanners. We apply segmentation on these meshes and show the results.

To have an accuracy with an order of magnitude of -3 and center values around zero in the same bin, our method uses histograms with 1001 bins. Choosing a good kernel standard deviation is a difficult problem, so we prefer to choose it empiricaly. In fact, our method uses a kernel with a standard deviation $h = 0.01$. Moreover, we limit our histograms in the interval $[-2;2]$ since we normalize curvature by edge length. The whole parameter set of our method has been validated experimentally.

Presentation of the used meshes

For experimental results, we used three digitized meshes from two different structured light scanners. The first two come from a first scanner and are illustrated in Fig. 14 and Fig. 15 respectively. The third comes from a second scanner and is illustrated in Fig. 16.

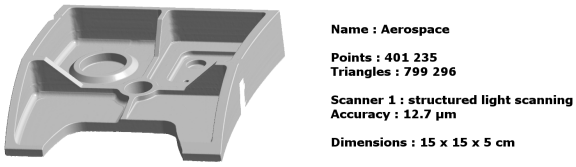

Figure 14. *Initial mesh from Scanner 1:* Aerospace.

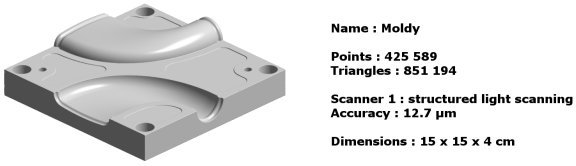

Figure 15. *Initial mesh from Scanner 1:* Moldy.

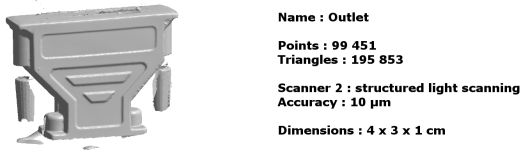

Figure 16. *Initial mesh from Scanner 2:* Outlet.

Segmentation result

This section presents results of our segmentation using curvature histograms analysis. Each figure shows edge extraction and final set of submeshes. For a reverse engineering application, our results are very good, since most of the features are correctly disconnected. Indeed, edge extraction is accurate since curvature thresholds are computed from distribution and so are adaptative.

The sharp edges of *Aerospace* are properly detected (Fig. 17.a) and the features are correctly disconnected, except for a few tangent ones. We obtain 70 submeshes whose 94.3% match with only one feature (Fig. 17.b).

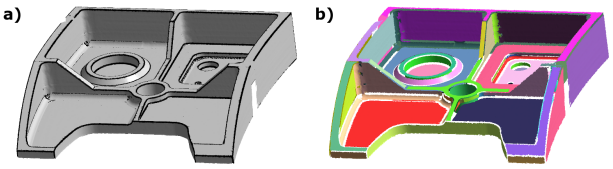

Figure 17. *a) Edges extraction and b) Segmentation of* Aerospace.

The sharp edges of *Moldy* are properly detected (Fig. 18.a) and the features are correctly disconnected. We note that freeforms are not over-segmented. We obtain 48 submeshes whose 100% match with only one feature (Fig. 18.b).

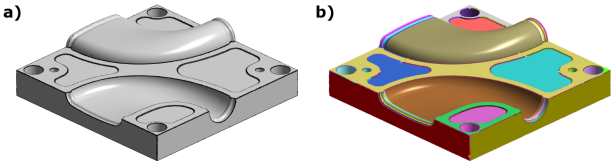

Figure 18. *a) Edges extraction and b) Segmentation of* Moldy.

The sharp edges of *Outlet* are properly detected (Fig. 19.a), except for the serrated cylinders, and the features are almost correctly disconnected. We obtain 72 submeshes whose 100% match with only one feature (Fig. 19.b).

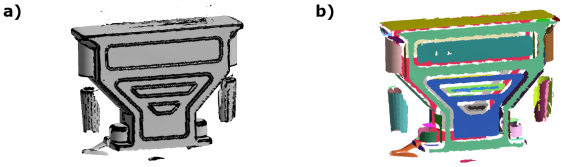

Figure 19. *a) Edges extraction and b) Segmentation of* Outlet.

We have segmented about 30 meshes, with a processor Intel® Core™ i7-4710 CPU @ 2.50GHz. Some results are presented in Table 1.

We can see that our segmentation is fast: less than one minute, except for very large amounts of triangles. Morover, these times also include curvatures and mesh topology computation, which represent a large part of the time.

To validate our approach, we count the number of submeshes that contain only one feature. We can see in Table 1 that about 96% of submeshes match with only one feature (the remaining 4% can contain similar tangent features).

Since features are correctly disconnected, obtained results are suitable for reverse engineering applications. Indeed, it is more accurate and easy to extract only one feature, than many on the same mesh, since we do not encounter curvature neighborhood problems or feature intersections.

Conclusion

In this paper, we propose a new digitized 3D mesh segmentation based on curvature anaylsis. Our proposed method is fully automatic, which is an advantage for industrial applications like reverse engineering. Our proposed segmentation first constructs a continuous normalized curvature distribution and analyzes it. Then, thresholds are computed from the distribution to extract object edges. In this way, it is possible to retrieve disconnected regions corresponding to object features. We also provide some post-processing like artefact merging, bridge removal and recursivity to improve results.

The most important part of our proposed method is the edge extraction based on curvature distribution analysis. Indeed, final submeshes are more homogeneous, and provide important information for reverse engineering, like neighborhood of the feature.

In future research, we propose to analyze more precisely our curvature distribution construction parameters, like bin number or kernel standard derivation, to improve threshold computing accuracy. We can also improve post-processing in order to take more parameters into account. A further objective consists of extending our segmentation to natural objects.

Table 1: Segmentation performances and associated features.

Mesh	Triangles	Time	Regions	One feature regions	in %
Vase	20 000	<1s	6	6	100
Fandisk	23 964	<1s	21	20	95.2
Lego	24 748	<1s	35	35	100
Lego_small	26 371	<1s	10	10	100
Cup	55 552	1s	32	32	100
Yoke	62 276	1s	8	7	87.5
Manique	65 090	1s	40	33	82.5
Nespresso	71 012	1s	5	5	100
MediumBolt	89 000	2s	11	10	90.9
StripedShoe	100 000	1s	16	16	100
Connector	195 424	4s	36	33	91.7
Outlet	195 853	5s	72	72	100
Etui	210 963	5s	3	3	100
Shoe	258 994	3s	4	4	100
Czslowakei	400 026	8s	162	162	100
Part2	414 823	12s	20	20	100
Chair	500 000	7s	85	79	92.9
Gear	500 000	6s	283	279	98.6
Aerospace	799 296	16s	70	66	94.3
MasterCylinder	820 793	29s	53	49	92.5
Moldy	851 194	13s	48	48	100
Watertight	921 216	16s	40	35	87.5
OilPump	1 064 031	22s	175	169	96.6
Carter	1 067 079	34s	108	106	98.1
Pump	1 105 570	21s	518	502	96.9
MachinedBlock	1 125 832	33s	113	111	98.2
Te	1 297 428	40s	39	36	92.3
Splint	2 095 079	1m09s	21	19	90.5
Metrologic	2 159 724	1m27s	14	13	92.9
ProductPart	3 427 245	2m16s	191	182	95.3

References

[1] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, and W. Puech. Recovering Primitives in 3D CAD meshes. *SPIE Electronic Imaging 2011, 3D Imaging, Interaction and Measurement*, 7864:7864 0R–1–9, 2011.

[2] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, and W. Puech. A comprehensive process of reverse engineering from 3d meshes to cad models. *Computer-Aided Design*, 45(11):1382 – 1393, 2013.

[3] P. Benkő, R. Martin, and T. Várady. Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, 33(11):839 – 851, 2001.

[4] J. Chen and H. Feng. Automatic prismatic feature segmentation of scanning-derived meshes utilising mean curvature histograms. *Virtual and Physical Prototyping*, 9(1):45–61, 2014.

[5] X. Chen and F. Schmitt. *Intrinsic surface properties from surface triangulation*, pages 739–743. Springer Berlin Heidelberg, 1992.

[6] K. Demarsin, D. Vanderstraeten, and D. Roose. Meshless extraction of closed feature lines using histogram thresholding. *Computer-Aided Design and Applications*, 5(5):589–600, 2008.

[7] L. Di Angelo and P. Di Stefano. Geometric segmentation of 3d scanned surfaces. *Computer-Aided Design*, 62:44–56, 2015.

[8] C. Dong and G. Wang. Curvatures estimation on triangular mesh. *Journal of Zhejiang University Science*, 6(1):128–136, 2005.

[9] G. Lavoué, F. Dupont, and A. Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design*, 37:975–987, 2004.

[10] A. Masmoudi, S. Chaoui, and A. Masmoudi. A finite mixture model of geometric distributions for lossless image compression. *Signal, Image and Video Processing*, 10:671–678, 2016.

[11] G. Mclachlan and T. Krishnan. *The EM Algorithm and Extensions*. Wiley-Interscience, 1996.

[12] S. Petitjean. A survey of methods for recovering quadrics in triangle meshes. *ACM Computing Surveys*, 2(34):1–61, 2002.

[13] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 2008.

[14] P. Theologou, I. Pratikakis, and T. Theoharis. A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation. *Comput. Vis. Image Underst.*, 135(C):49–82, 2015.