Computers in Industry 92 (2017) 67–83

Contents lists available at ScienceDirect

Computers in Industry

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

Analysis of digitized 3D mesh curvature histograms for reverse engineering

S. Gauthier[a,b,*], W. Puech[a], R. Bénière[b], G. Subsol[a]

[a] LIRMM Laboratory, UMR 5506, CNRS, University of Montpellier, 860 rue de St Priest, Montpellier, France
[b] C4W, 219 rue Le Titien, Montpellier, France

A R T I C L E I N F O

Article history:
Received 8 February 2017
Received in revised form 4 May 2017
Accepted 13 June 2017
Available online xxx

Keywords:
3D mesh
Digitized
Curvature
Distribution
Reverse engineering

A B S T R A C T

Today, it has become more frequent and reasonably easy to digitize the surface of 3D objects. However, the obtained results are often inaccurate and noisy. In this paper, we present an efficient method to analyze a curvature histogram from a digitized 3D surface using a real object. Moreover, we propose to use the curvature histogram analysis for many steps of a reverse engineering process, which can be used to retrieve a CAD model from a digitized one for example. Our objective is to design a fast and fully automated method, which is seldom seen in reverse engineering. Experimental results applied on digitized 3D meshes show the efficiency and the robustness of our proposed method.

1. Introduction

The availability of 3D scanners has increased the fast development of applications in Computed-Aided Design (CAD), reverse engineering, medicine and inspection. Many 3D processes use the objects shape, like segmentation, recognition or classification for example. In the production line of manufactured objects, steps can be distributed to many partners, and during the process, some data can be lost. An industrial reverse engineering application aims to reconstruct an object as a combination of geometric primitives, from a digitized 3D mesh or 3D point cloud [1,2]. For mechanical objects, we search for planes, spheres, cylinders and cones, but also torus and more specifically developable or ruled surfaces. This can lead to quality control or object modification issues for example. To reconstruct the initial geometry, we must take into account the shape of the objects and their relationship with each other. But an object shape can be very complex, and the measured data can often be noisy. So, we need robust 3D descriptors to accurately define the objects shape.

In previous work, geometry descriptors like the curvatures [3,4] allow us to deal with the 3D mesh shape. But the curvature is computed locally, while it is often necessary to characterize the shape globally. To do this, we can construct curvature distributions [5,6] 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. [7] who incorporates two other methods [8,9]. 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 compute some statistics depending on the chosen application. Indeed, curvature distribution approximately describes the objects shape, so these distributions can be useful to many applications. In this paper, we propose to use a curvature histogram to segment 3D meshes, detect primitive type and measure the quality of the mesh.

This paper is organized as follows. Previous work in this topic is presented in Section 2. In Section 3, we present in detail our distribution construction and analysis. Section 4 is dedicated to three uses of curvature distribution, which are mesh segmentation, primitive type detection with tolerances adaptation and mesh quality evaluation. In Section 5, we apply our proposed analysis on digitized 3D surfaces of real objects and we show that our analysis hugely improves the obtained results. Finally, we conclude and propose directions for future research in Section 6.

2. Previous work

We present in Section 2.1 previous work on curvatures and distributions. Then, we show three fields in 3D mesh processing:

* Corresponding author at: LIRMM Laboratory, UMR 5506, CNRS, University of Montpellier, 860 rue de St Priest, Montpellier, France.
E-mail address: silvere.gauthier@lirmm.fr (S. Gauthier).

http://dx.doi.org/10.1016/j.compind.2017.06.008
0166-3615/© 2017 Elsevier B.V. All rights reserved.

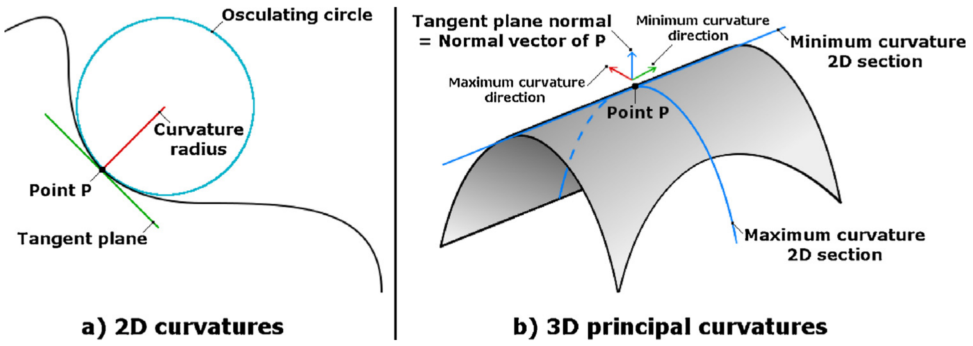

Osculating circle
Curvature radius
Point P
Tangent plane
a) 2D curvatures
Tangent plane normal = Normal vector of P
Minimum curvature direction
Minimum curvature 2D section
Maximum curvature direction
Point P
Maximum curvature 2D section
b) 3D principal curvatures

Fig. 1. Curvature representation on: (a) 2D curves and (b) 3D surfaces.

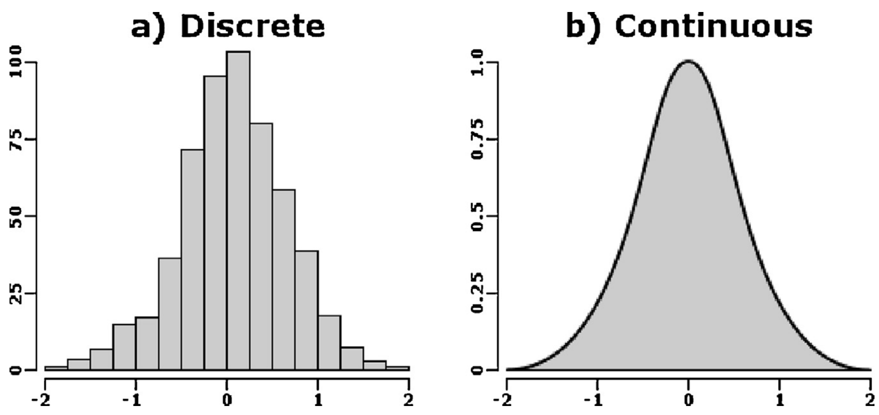


Fig. 2. (a) Discrete and (b) continuous distribution samples.

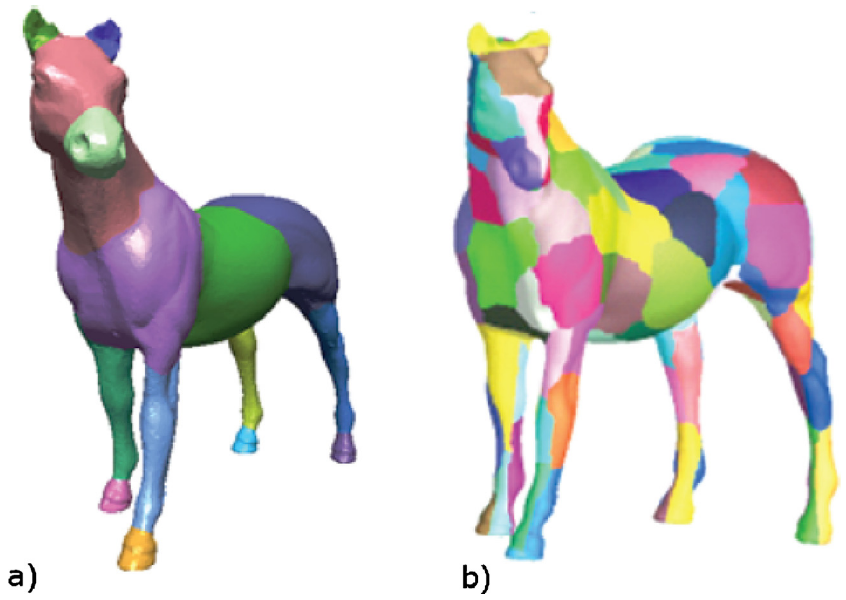


Fig. 3. Examples of mesh segmentation from [19]: (a) A section-type segmentation, (b) a surface-type segmentation.

mesh segmentation in Section 2.1.1, geometric primitive type detection in Section 2.1.2 and mesh quality evaluation in Section 2.1.3.

2.1. Curvature distribution

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. 1a).

On a 3D surface, an infinity of curvature directions exists around the normal vector of P (Fig. 1b). So, we need to distinguish particular curvatures. Principal curvatures are the minimum and maximum curvatures. Mean curvature and Gaussian curvature equal respectively the mean and the product of principal

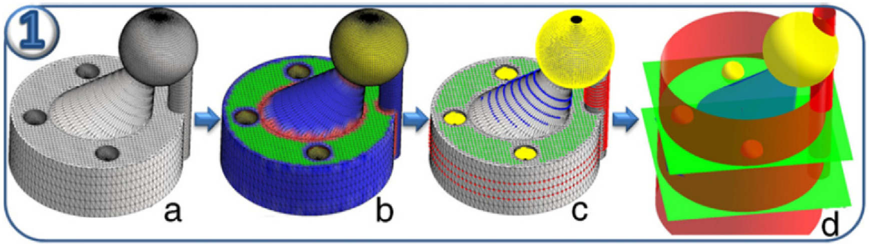


Fig. 4. (a) Original 3D mesh, (b) curvature: planar (green), spherical (yellow), convex (blue) and concave (red), (c) point areas, (d) extracted primitives [2]. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

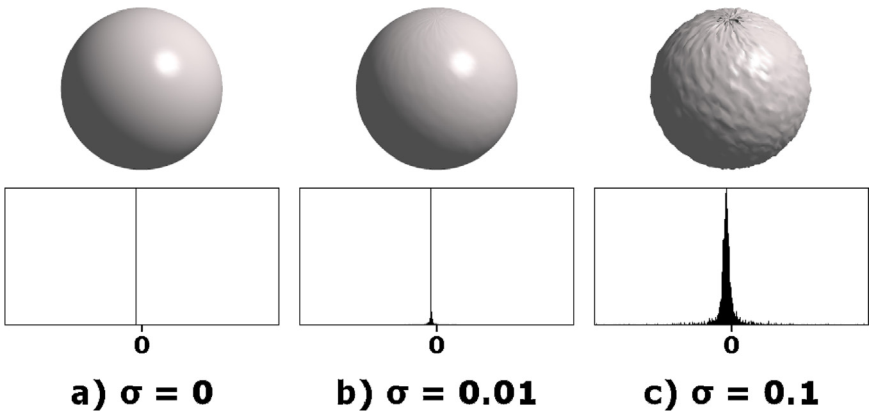


Fig. 5. Spheres of radius $r = 10$ with Gaussian noise and their corresponding mean curvature histogram: (a) original sphere, (b) $\sigma = 0.01$, (c) $\sigma = 0.1$.

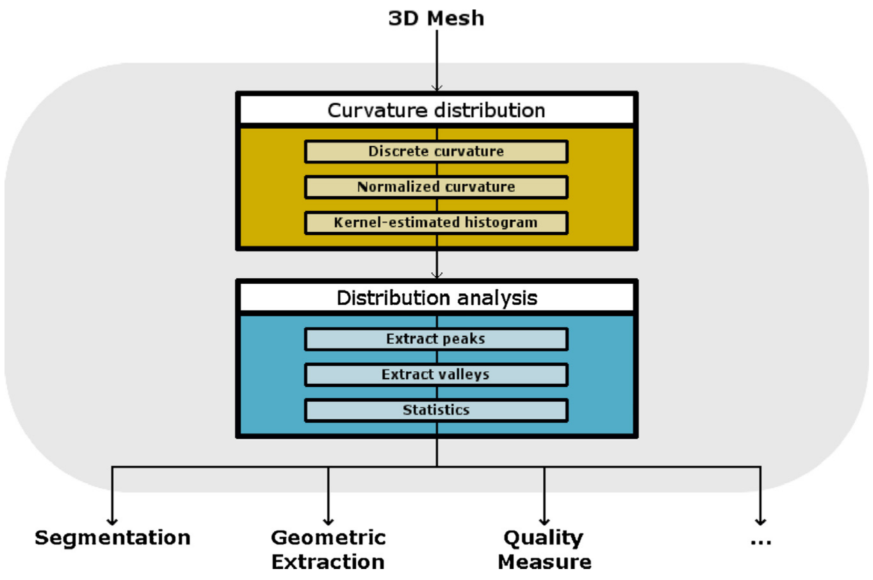


Fig. 6. Method overview.

curvatures. The Euler formula gives the continuous curvature at a point P for each tangent vector t_i:

$$k_n(t_i) = k_{\max}\cos^2(\theta) + k_{\min}\sin^2(\theta), \quad (1)$$

where k_{max} and k_{min} are the principal curvatures, and θ is the angle between the maximum principal direction $\vec{d}^{\max}$ and the direction of k_n. But since a mesh is a discrete object, we need to approximate the curvature from a point cloud. Chen and Schmitt [8] compute "discrete" curvatures at P with the Meusnier theorem:

$$k_n(t) = k_C * \cos(\theta), \quad (2)$$

where k_C is the curvature at P of the curve obtained by intersection between the surface and a plane P_C, and θ is the angle between the normal at P and the normal of the plane P_C. Each neighbor pair of P defines a plane with P, and the curvature circle is the circumscribed circle of the three points. A linear regression is then applied on all discrete curvatures to retrieve approximated principal curvatures.

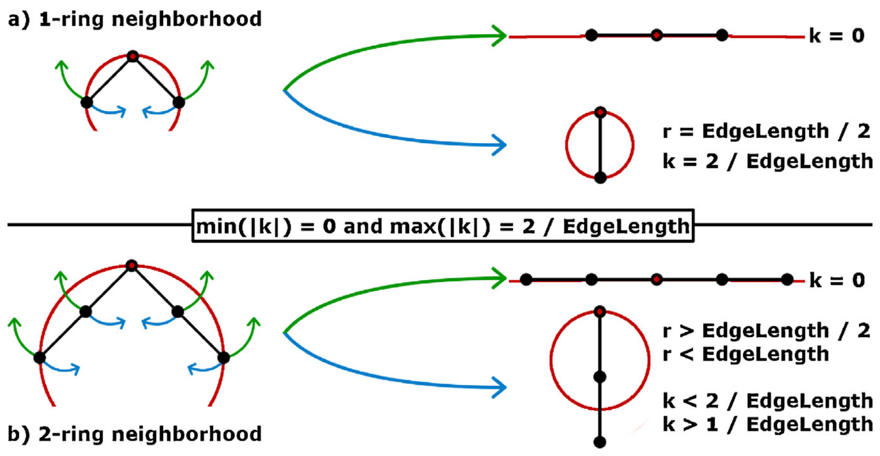


Fig. 7. Minimum and maximum possible for absolute curvature values.

Dong and Wang [9] compute discrete curvatures with:

$$k_n(t) = \frac{< P_i - P, N_i - N >}{\|P_i - P\|^2}, \quad (3)$$

where P_i is a neighbor of P with a normal N_i, and $\vec{t}$ the projection of 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 the formula (3) for each neighbor of P, and apply the linear regression of [8]. For our proposed approach, we prefer to use this approximation because it is more accurate. Indeed, Eq. (3) 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 characterize surface shape [10,11]. So, for example, it is possible to analyze a shape to detect saliency [12] or apply segmentation [13,5].

There are many different types of distributions. But overall, we can distinguish discrete and continuous distributions. A discrete distribution is often represented by a histogram, as illustrated in Fig. 2a. On the other hand, continuous distributions correspond to mathematical models. Common models are Gaussian or Normal distributions (Fig. 2b).

In our case, computed curvature values are real. So, 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 estimates the parameters of a density function by maximizing the joint likelihood of the observed samples to be generated by this model. A common optimization algorithm is the Expectation–Maximisation (EM) [14]. For example, these distributions can be used in image processing to improve compression [6], for perceptual hashing systems [15], or even automatic image thresholding [16].

Previous methods in 3D mesh processing proposed to use distributions [17,5]. For example, Demarsin et al. [17] compute an absolute mean curvature histogram used to extract object edges. They analyze many histogram resolutions to define a sufficient bin number. However, Chen and Feng [5] construct a signed mean curvature histogram, then apply a Laplacian smooth modifier and analyze the histogram to segment the object. They can separate homogeneous intervals by histogram analysis. But the main limitation of these two methods is that the histogram is constructed with a discrete approach, whereas curvature values are real. To extend their methods, we can construct a histogram by optimization or kernel estimation for example.

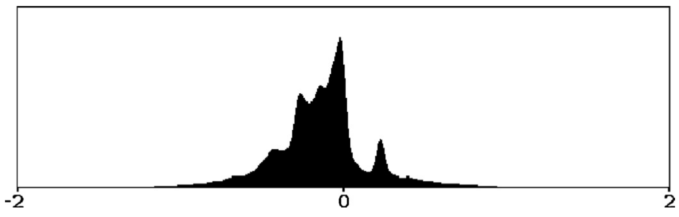


Fig. 8. Normalized mean curvature kernel-estimated histogram example.

2.1.1. Segmentation

A segmentation is a partitioning of a digital image, a 3D mesh or a 3D point cloud in several regions, as illustrated in Fig. 3. For 3D objects, we distinguish between cloud-based [18] and mesh-based approaches [19].

Many different 3D mesh segmentation algorithms have been published [20,21], 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 [22], hierarchical clustering [23], iterative merging [24] or remeshing [25]. In reverse engineering, the best results are reached when using curvatures, because primitives are extracted from curvatures analysis. Some methods use curvatures to segment by discontinuities [1,5], clustering [26], or to better digitize [27], but they are often not robust enough around object edges or are sensitive to noise [13,2]. Indeed, curvatures are often inaccurate around object edges because adjacent points can run over many different primitives. Moreover, we are searching for a fully automatic method, so we cannot use parameters like a cluster number [28].

The edge extraction of Demarsin et al. [17] is interesting, but it is not entirely automatic. Indeed, they compute an absolute mean curvature histogram and define a threshold leading to edge extraction. Then, they ask for user help to validate the result or compute another threshold. To extend this threshold computation, we can normalize the curvature and fix the histogram range.

Chen and Feng [5] propose to construct the mean curvature histogram, and then to apply a Laplacian smooth modifier. After, they compute valleys on the histogram, which define segmentation thresholds. Finally, they retrieve the isolated regions and improve their boundaries. But their method is limited since they do not have a single primitive per submesh. To extend this segmentation, we can apply a recursive extraction of salient edges.

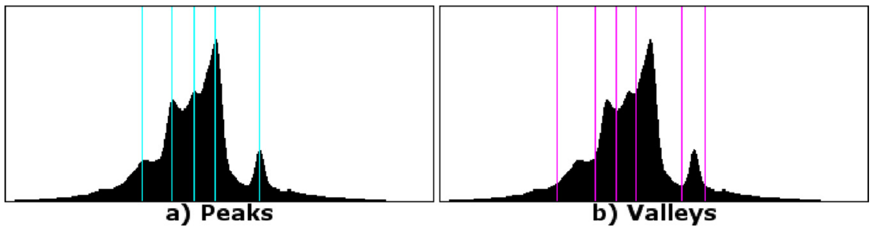


Fig. 9. Histogram: (a) peaks and (b) valleys.

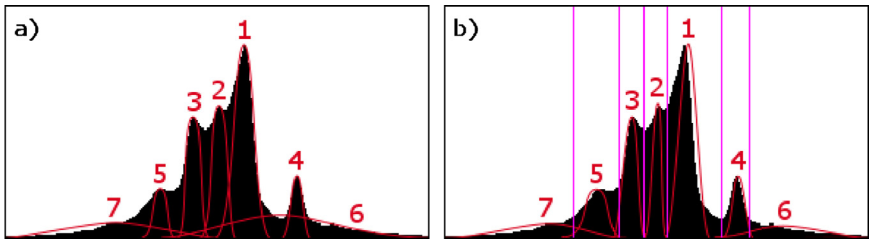


Fig. 10. (a) Gaussian Mixture Model of a histogram, (b) starting models with peak and valley detection.

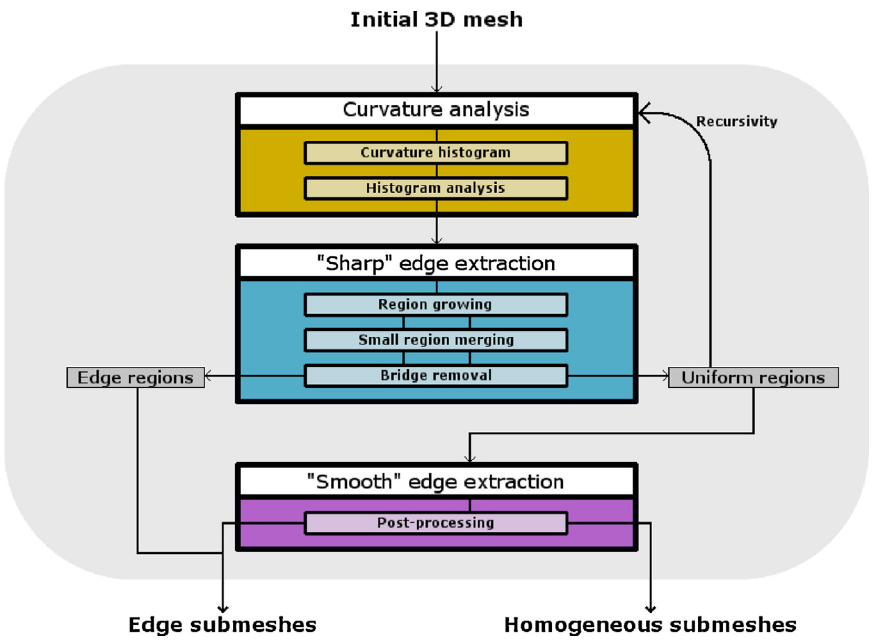


Fig. 11. Segmentation overview based on the proposed curvature analysis.

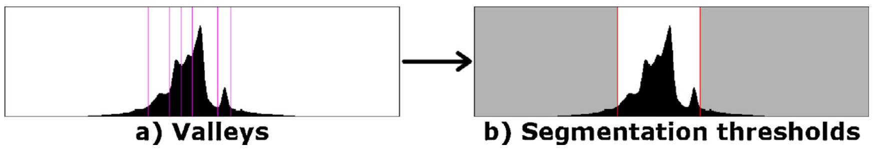


Fig. 12. Edge extraction.

2.1.2. Primitive type detection

Primitive extraction starts from an initial set of measured data and builds derived values, which are primitives (Fig. 4).

We generally search for planes, spheres and cylinders because they are the most common primitives contained in mechanical objects, but we can also search for cones, torus or ruled surfaces for example. Primitive extraction can be based on fitting profiles [1], quadratic surfaces [29] or freeform surfaces [30]. But most of the time, primitive extraction uses curvature analysis, after a mesh segmentation [2], with a distribution [31], or even with outlier handling [32]. Since two primitives with identical parameters are

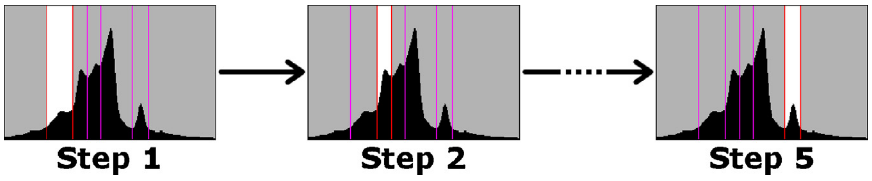


Fig. 13. Homogeneous area extraction.

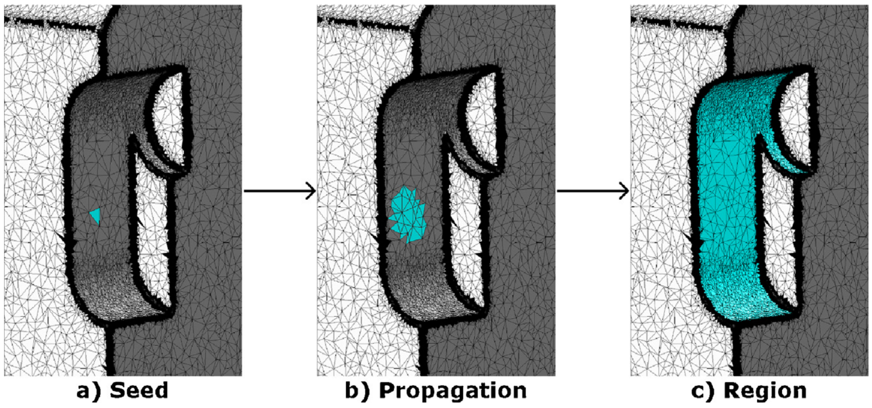


Fig. 14. Region growing algorithm.

not distinguished by their curvature, these methods cannot directly handle primitive positions. But curvature is robust to noise, and can be analyzed locally to extract each primitive separately.

The limit of these methods is that they can not estimate the primitive type before extraction. To detect this primitive type, we propose to analyze the principal curvature distributions.

2.1.3. Noise and mesh quality evaluation

In many 3D processes, it is important to handle noise properly to avoid distorted representation. But noise characterization depends on the chosen scanner. In fact, noise characterization for depth sensors [33] and for laser beams [34] are different. So, digitization noise estimation is difficult. Most of the time, methods presume a Gaussian and isotropic noise, which does not reflect reality. To properly deduce the real noise type, it is necessary to know the object shape before the analysis. But in case of digitized meshes, we cannot know this shape. Moreover, many different noises can be present at the same time. In Fig. 5, we apply different values of Gaussian noise to a sphere and compare curvature histograms. We show that curvature distributions of noisy spheres have a Gaussian shape, which is related to the noise type.

We can also introduce an estimation for the roughness of the surface [35], which is defined by a local analysis of the curvature values. Like the noise on coordinates, the roughness can be difficult to characterize since it depends also on the object material and the digitization. Sometimes, roughness can quantify locally the noise.

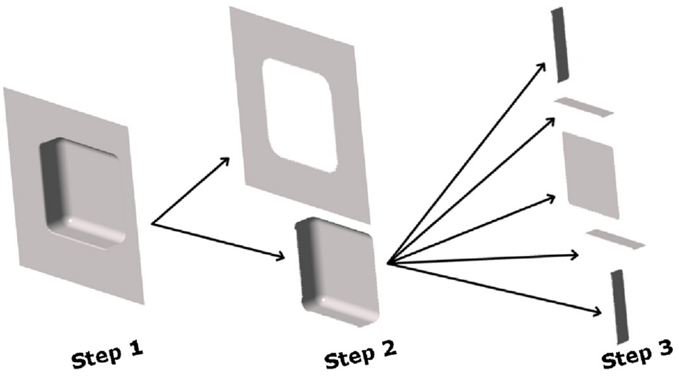


Fig. 15. Recursive segmentation.

3. Proposed curvature distribution analysis

Our proposed method first computes a discrete curvature on each vertex of the initial 3D mesh. This method constructs a continuous estimated and normalized histogram for each curvature (Section 3.1), then analyzes it (Section 3.2). An overview of our proposed method is illustrated Fig. 6.

We provide, for example, a peak and valley extraction used for segmentation (Section 4.1). In the same way, some statistics computed on each histogram are useful to adapt geometric extraction tolerances (Section 4.2) or measure the quality of the mesh (Section 4.3).

3.1. Probability curvature distribution

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 probabilities can be nonzero only if they refer to intervals.

To approximate continuous curvatures on a discrete 3D mesh, we use the method proposed by Bénière et al. [2]. In our case,

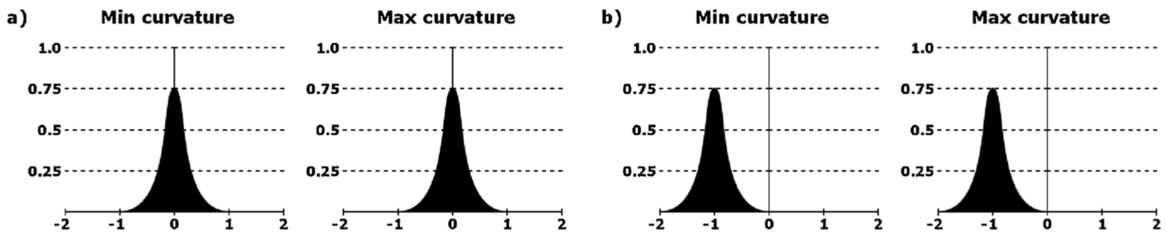


Fig. 16. Principal curvature histograms: (a) for a plane and (b) for a sphere.

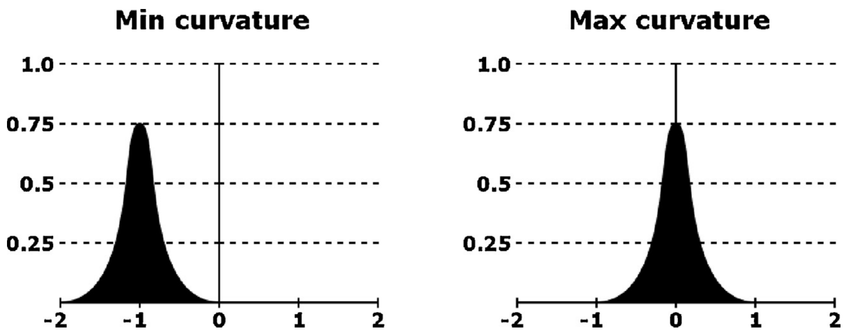


Fig. 17. Principal curvature histograms for a cylinder.

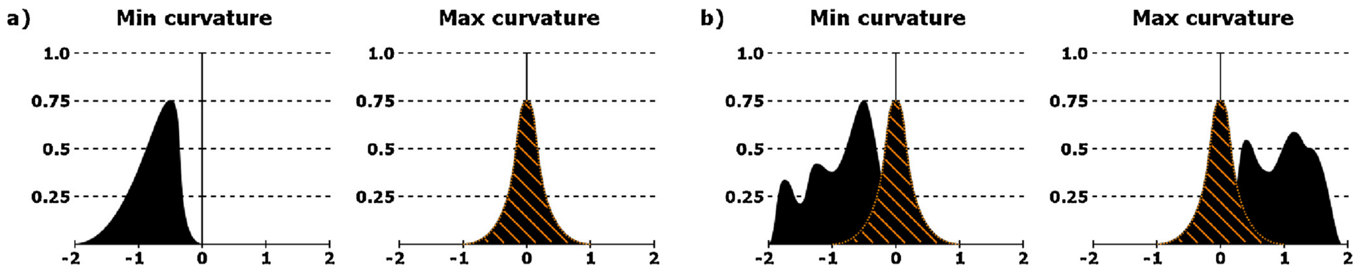


Fig. 18. Principal curvature histograms: (a) for a cone and (b) for a developable surface. We can see in hatched orange a constant curvature which can be analyzed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

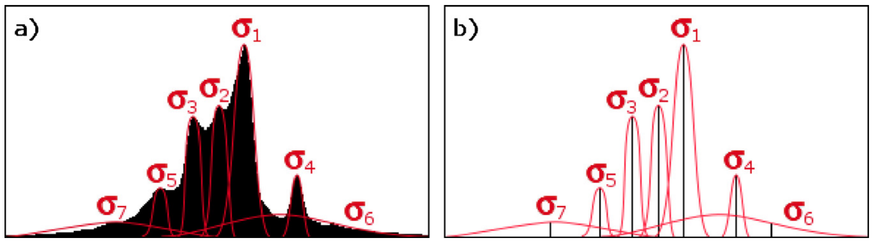
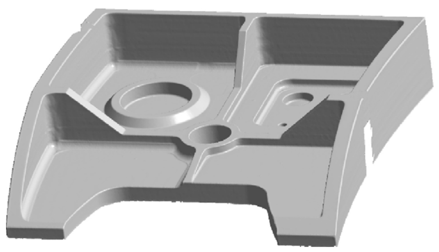


Fig. 19. (a) Example of GMM standard deviations. (b) Corresponding CAD curvature distribution.



Name : Aerospace

Points : 401 235
Triangles : 799 296

Scanner 1 : structured light scanning
Accuracy : 12.7 µm

Dimensions : 15 x 15 x 5 cm

Fig. 20. Initial mesh from Scanner 1: *Aerospace*.

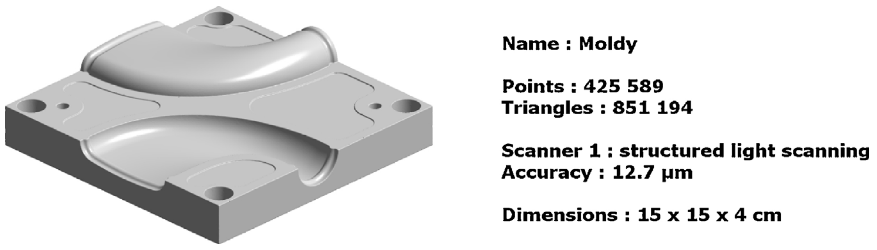


Fig. 21. Initial mesh from Scanner 1: *Moldy*.

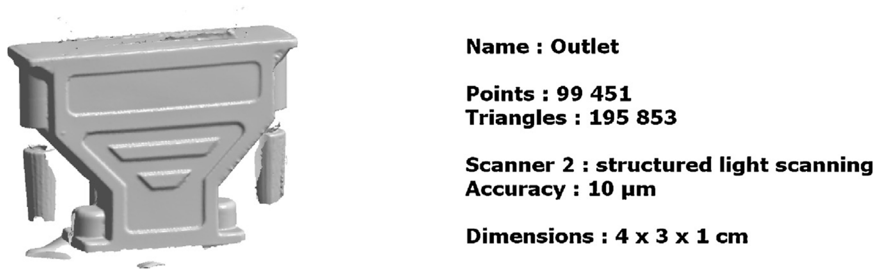


Fig. 22. Initial mesh from Scanner 2: *Outlet*.

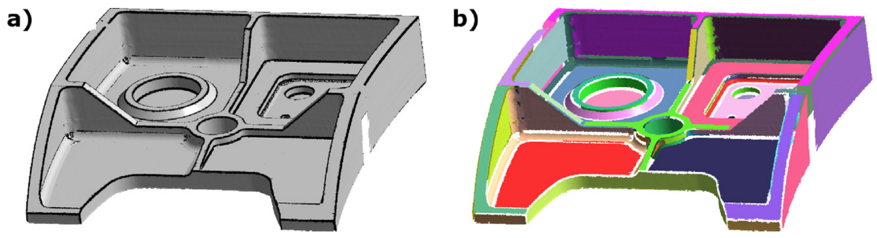


Fig. 23. (a) Edge extraction and (b) segmentation of *Aerospace*.

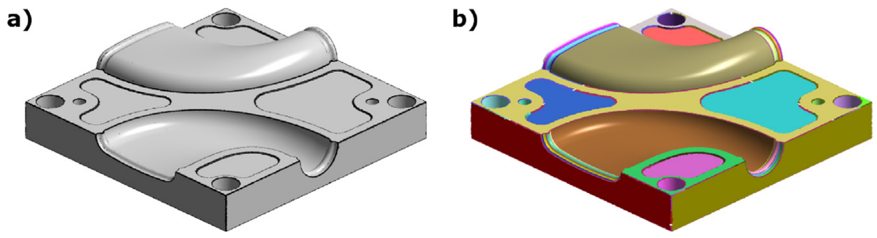


Fig. 24. (a) Edge extraction and (b) segmentation of *Moldy*.

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 (Section 3.1.1). Moreover, curvature values are real, so it is more suitable to construct a histogram with kernel-estimation for example (Section 3.1.2).

3.1.1. Normalized curvature

Histograms must have the same range to make a comparison. Indeed, we must take into account the mesh scale. To homogenize them, curvature values have to be normalized. Our method normalizes curvature values by multiplying them by the mean edge length of the mesh.

If the object is correctly meshed and edge lengths are equal, we can deduce the minimum and maximum possible curvature values, as illustrated in Fig. 7. We propose then to limit the histogram range according to these values.

Edge lengths of a digitized mesh are not all equal, but they are similar enough to not distort the histogram. If a mesh has varying edge lengths, some curvature values can be truncated. Most of the time, there is only a small number of long edges, and the histogram is similar with or without these edges.

3.1.2. 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 primitive often give Gaussian-type curvature distributions. This may be related to

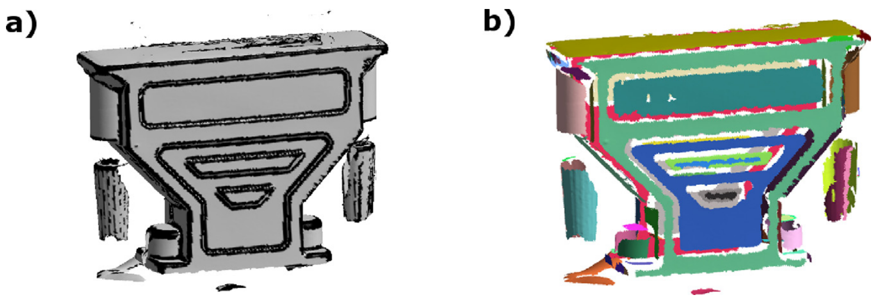


Fig. 25. (a) Edge extraction and (b) segmentation of *Outlet*.

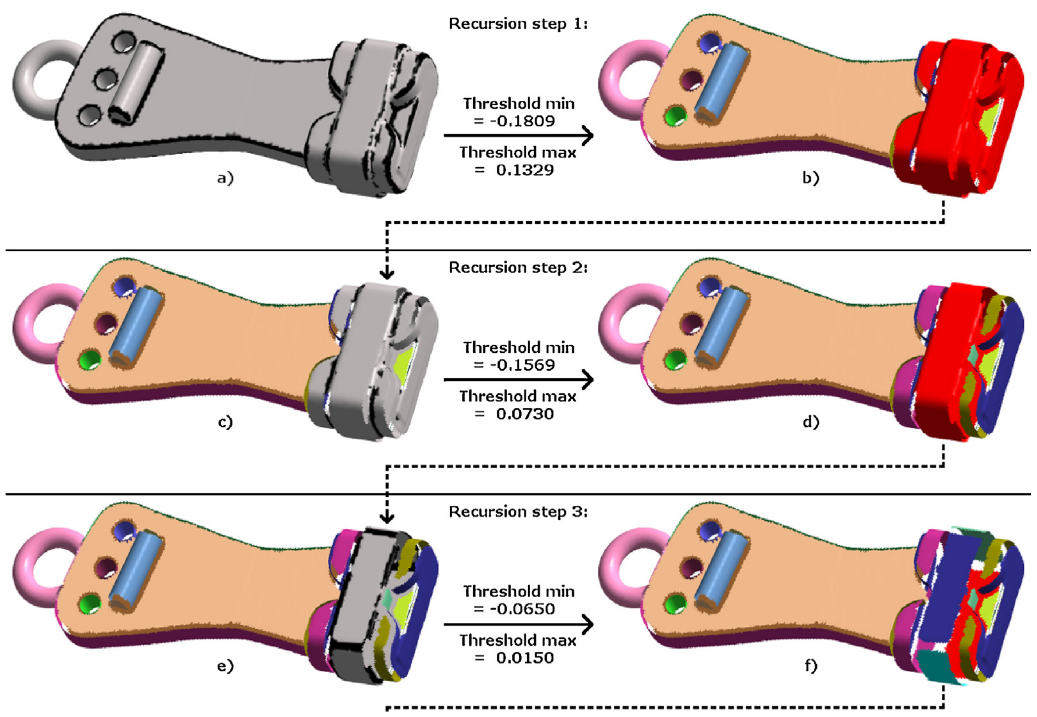


Fig. 26. Example of automatic recursion: (a) Edge extraction and (b) segmentation of *Manique*, (c) edge extraction and (d) segmentation of the first submesh from (b), (e) edge extraction and (f) segmentation of the first submesh from (d). The computed curvature thresholds are shown in the middle, computed from the corresponding gray part.

scanner characteristics. We compute the histogram with:

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i), \quad (4)$$

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 ith point and h is the kernel standard deviation. The Gaussian kernel is defined by:

$$K_h(x - x_i) = \frac{1}{h\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2}. \quad (5)$$

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. 8 shows an example of a normalized mean curvature distribution with a kernel deviation $h = 0.01$.

This continuous estimation makes histograms less sensitive to noise and bin number. Indeed, high frequency fluctuations are naturally smoothed by the kernel. Besides, if the noise deviation is close to or greater than the kernel deviation, curvature values can be too mixed and histogram analysis is limited.

3.2. Analysis

Many characteristics of a histogram can be useful to numerous applications. We can, for example, search for a modal number and positions (named "peaks" here), pattern, sparsity and statistics. In our method, we essentially provide robust peak and valley detection (Section 3.2.1), and use some statistics like mean and standard deviation (Section 3.2.2).

3.2.1. Peaks and valleys

To detect homogeneous curvature intervals, we need to detect peaks and valleys in the histogram, as illustrated in Fig. 9a and b. A peak defines a dominant curvature value and a pair of two consecutive valleys defines an homogeneous interval of curvature.

To detect peaks (or modes), many methods exist like the mean-shift algorithm [36]. But these kind of methods can be heavy and do not detect valleys. Therefore, we prefer to use a simple method based on derivatives.

We begin by computing a discrete approximation of the second derivative of the histogram:

$$D^2(i) = H(i + 1) + H(i - 1) - 2H(i), \quad (6)$$

with $D^2(i)$ the ith second derivative value and $H(i)$ the ith histogram value.

Thus, we aim to detect robust peaks and valleys from the second derivative. A peak (resp. a valley) is a bin with a higher (resp. a lower) probability than the two adjacent bins. A robust peak (resp. a robust valley) is a bin with the highest (resp. the lowest) probability in a window. Finally, a robust peak (resp. a robust valley) in the second derivative corresponds to a valley (resp. a peak) in the histogram. Our method uses small sliding windows for both peaks and valleys detection, just to avoid small fluctuations.

3.2.2. Gaussian Mixture Model

We can also estimate our distribution by a Gaussian Mixture Model (GMM), which provides mean and standard deviation of each homogeneous interval of curvature. To do this, we can use algorithms like K-Means or Bayesian Information Criterion (BIC) to give a number of Gaussian models. Thus, we can use an Expectation–Maximisation algorithm (EM) to optimize these models (Fig. 10a).

With our peak and valley detection, it is possible to obtain a fast convergence of EM. Indeed, we can suppose that the optimal number of models is close to our number of valleys. Moreover, we can compute starting mean and standard deviation of each model by computing it between each couple of consecutive valleys (Fig. 10b). We can see that these models are already close to optimized ones, so fewer iterations are needed to compute them.

4. Curvature distribution applications

Our method can be useful in many applications. This section presents three applications which can be used, for example, in a reverse engineering process: digitized 3D mesh segmentation (Section 4.1), primitive type detection with tolerances adaptation (Section 4.2) and mesh quality measurement (Section 4.3).

4.1. Segmentation

In our case, we work on digitized meshes, with point coordinate inaccuracies and noise. To correctly segment these, we aim to extract salient object edges matching with intersections between geometric primitives (Section 4.1.1). We can retrieve isolated and homogeneous regions (Section 4.1.2). Finally, we also apply recursivity (Section 4.1.3) to improve results and obtain only one primitive in each submesh. Our proposed segmentation, based on curvature histogram analysis, is fast and completely automated. Fig. 11 shows an overview of our proposed segmentation.

4.1.1. Edge extraction

The edges of an object are the salient mesh areas characterized by a high curvature. We propose to extract these edges with curvature histogram analysis, as described in Section 3. Indeed, high curvatures are on the extremities of curvature histograms. To detect and extract the 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. 12). Points with curvature between the two thresholds are labeled “uniform”, and others are labeled “edge”.

We can also realize multiple thresholding by taking each valley pair as thresholds (Fig. 13). This method directly isolates homogeneous intervals, but we need to be aware of potential curvature inaccuracy.

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

4.1.2. 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

Table 1
Segmentation performances: time (*T.*) and region number (*R.*).

Mesh	Triangle	*T.*	*R.*
Vase	20 000	<1 s	6
Fandisk	23 964	<1 s	21
Lego	24 748	<1 s	35
Lego_small	26 371	<1 s	10
Cup	55 552	1 s	32
Yoke	62 276	1 s	8
Manique	65 090	1 s	40
Nespresso	71 012	1 s	5
MediumBolt	89 000	2 s	11
StripedShoe	100 000	1 s	16
Connector	195 424	4 s	36
Outlet	195 853	5 s	72
Etui	210 963	5 s	3
Shoe	258 994	3 s	4
Czslowakei	400 026	8 s	162
Part2	414 823	12 s	20
Chair	500 000	7 s	85
Gear	500 000	6 s	283
Aerospace	799 296	16 s	70
Master	820 793	29 s	53
Moldy	851 194	13 s	48
Watertight	921 216	16 s	40
OilPump	1 064 031	22 s	175
Carter	1 067 079	34 s	108
Pump	1 105 570	21 s	518
Block	1 125 832	33 s	113
Te	1 297 428	40 s	39
Splint	2 095 079	1 min 09 s	21
Metrologic	2 159 724	1 min 27 s	14
ProductPart	3 427 245	2 m 16 s	191

Table 2
Submeshes with only one primitive (O.P.).

Mesh	O.P.	Total	in %
Vase	6	6	100
Fandisk	20	21	95.2
Lego	35	35	100
Lego_small	10	10	100
Cup	32	32	100
Yoke	7	8	87.5
Manique	33	40	82.5
Nespresso	5	5	100
MediumBolt	10	11	90.9
StripedShoe	16	16	100
Connector	33	36	91.7
Outlet	72	72	100
Etui	3	3	100
Shoe	4	4	100
Czslowakei	162	162	100
Part2	20	20	100
Chair	79	85	92.9
Gear	279	283	98.6
Aerospace	66	70	94.3
Master	49	53	92.5
Moldy	48	48	100
Watertight	35	40	87.5
OilPump	169	175	96.6
Carter	106	108	98.1
Pump	502	518	96.9
Block	111	113	98.2
Te	36	39	92.3
Splint	19	21	90.5
Metrologic	13	14	92.9
ProductPart	182	191	95.3

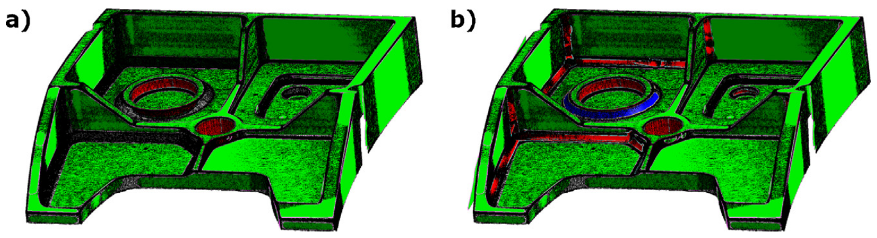


Fig. 27. Geometric primitives extraction with (a) mesh analysis and (b) curvature analysis of *Aerospace*.

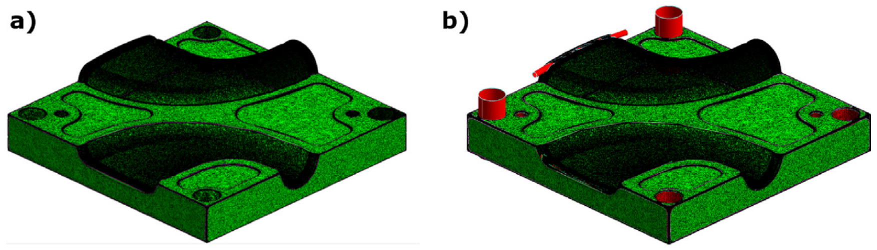


Fig. 28. Geometric primitives extraction with (a) mesh analysis and (b) curvature analysis of *Moldy*.

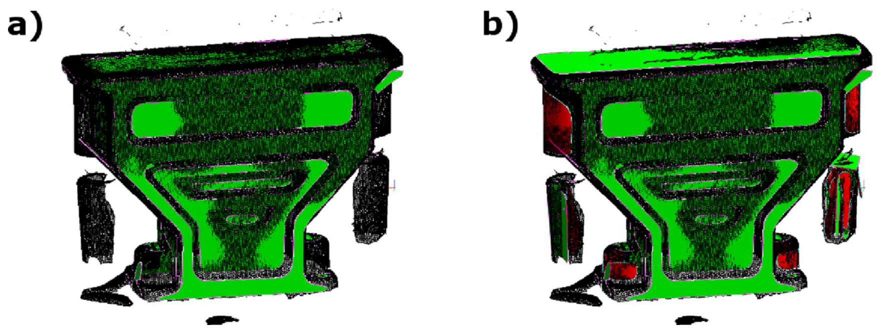


Fig. 29. Geometric primitives extraction with (a) mesh analysis and (b) curvature analysis of *Outlet*.

algorithm that groups similar adjacent elements: take a "seed" triangle and assign a unique ID (Fig. 14a) and propagate the seed ID to its neighbors (Fig. 14b) until it reaches the borders (Fig. 14c).

The choice of the seed does not matter since triangles have only two possible values: threshold or non-threshold. We always retrieve the same set of regions.

4.1.3. 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 region by submesh. Since our method is based on curvature histogram analysis, each submesh has a different histogram and we can detect object edges with many scales (Fig. 15).

4.2. Primitive type detection tolerances

Most of the time, geometric primitive extraction uses many tolerances. These tolerances, on curvature for example, are often specific to the primitive type (plane, sphere, cylinder, cone, torus, . . .). If we can determine the dominant surface type in a mesh, we can also adapt tolerances thanks to this information. In fact, we can deduce it directly from curvature histogram analysis. So, it is possible to improve geometric surface fitting. This section presents tolerance adaptation for planes and spheres in Section 4.2.1, cylinders in Section 4.2.2 and more complex primitives in Section 4.2.3.

4.2.1. Planes and spheres

On a plane, the two principal curvatures equal zero. So we can define a planar mesh as a mesh with minimum and maximum curvature histogram values around zero (Fig. 16a).

The tolerance of zero curvature can be computed from the principal curvature histograms:

$$CurvatureZero_{Plane} = \frac{Sigma_{Min} + Sigma_{Max}}{2}, \quad (7)$$

with $Sigma_{Min}$ and $Sigma_{Max}$ the standard deviations of minimum and maximum curvature histograms.

On a sphere, the two principal curvature histogram values are equals and far from zero. So we can define a spherical mesh as a mesh with minimum and maximum curvature histogram values which are similar and far from zero, as illustrated in Fig. 16b.

The curvature similarity tolerance can be computed with:

$$\begin{aligned} &LowerBound = \min(Mu_{Min} - Sigma_{Min}; Mu_{Max} - Sigma_{Max}),\\ &UpperBound = \max(Mu_{Min} + Sigma_{Min}; Mu_{Max} + Sigma_{Max}),\\ &SimilarCurvatures_{Sphere} = [LowerBound, UpperBound], \end{aligned} \quad (8)$$

with (Mu_{Min}, $Sigma_{Min}$) and (Mu_{Max}, $Sigma_{Max}$) the mean and standard deviation of minimum and maximum curvature histograms respectively.

Table 3
Primitive extraction with mesh analysis (F.M.A.) proposed by Bénière et al. [2] and curvature analysis (F.C.A.) tolerance adaptation.

Mesh	F.M.A. [2]	F.C.A.
Vase	1	4
Fandisk	14	20
Lego	15	30
Lego_small	5	9
Cup	1	4
Yoke	1	3
Manique	28	32
Nespresso	1	4
MediumBolt	4	6
StripedShoe	2	2
Connector	22	28
Outlet	31	52
Etui	2	3
Shoe	2	2
Czslowakei	102	107
Part2	11	16
Chair	19	44
Gear	277	279
Aerospace	71	105
Master	0	7
Moldy	37	55
Watertight	8	28
OilPump	17	56
Carter	3	9
Pump	241	263
Block	85	89
Te	20	28
Splint	8	14
Metrologic	0	16
ProductPart	0	62

4.2.2. Cylinders

On a cylinder, one of the principal curvatures equals zero and the other is far from zero. So we can define a cylindrical mesh as a mesh with a principal curvature histogram around zero and the other far from zero (Fig. 17).

We can compute a zero curvature tolerance from the histogram with curvature values around zero:

$$CurvatureZero_{Cylinder} = Sigma_{Zero}, \tag{9}$$

with $Sigma_{Zero}$ the standard deviation of the histogram which is around zero.

4.2.3. Other primitives

We can also find other primitive signatures in curvature histograms.

On a cone, one of the principal curvatures equals zero and the other is variable (Fig. 18a). So we can only analyze the zero curvature histogram.

On a developable surface, one of the curvature equals zero, and the other is variable. The difficulty is that we may have an inversion of minimum and maximum curvatures values (Fig. 18b). A possible solution can be to use the principal directions to test the consistency of the two sets of curvatures.

We can extend this to other surfaces. For example, on a torus or a uniform generalized cylinder, one of the curvatures is constant, even if we can have the same curvature inversion as a developable surface. In all cases, if we can find a constant curvature, we can use it to adapt tolerances and also measure mesh quality, as described in Section 4.3.

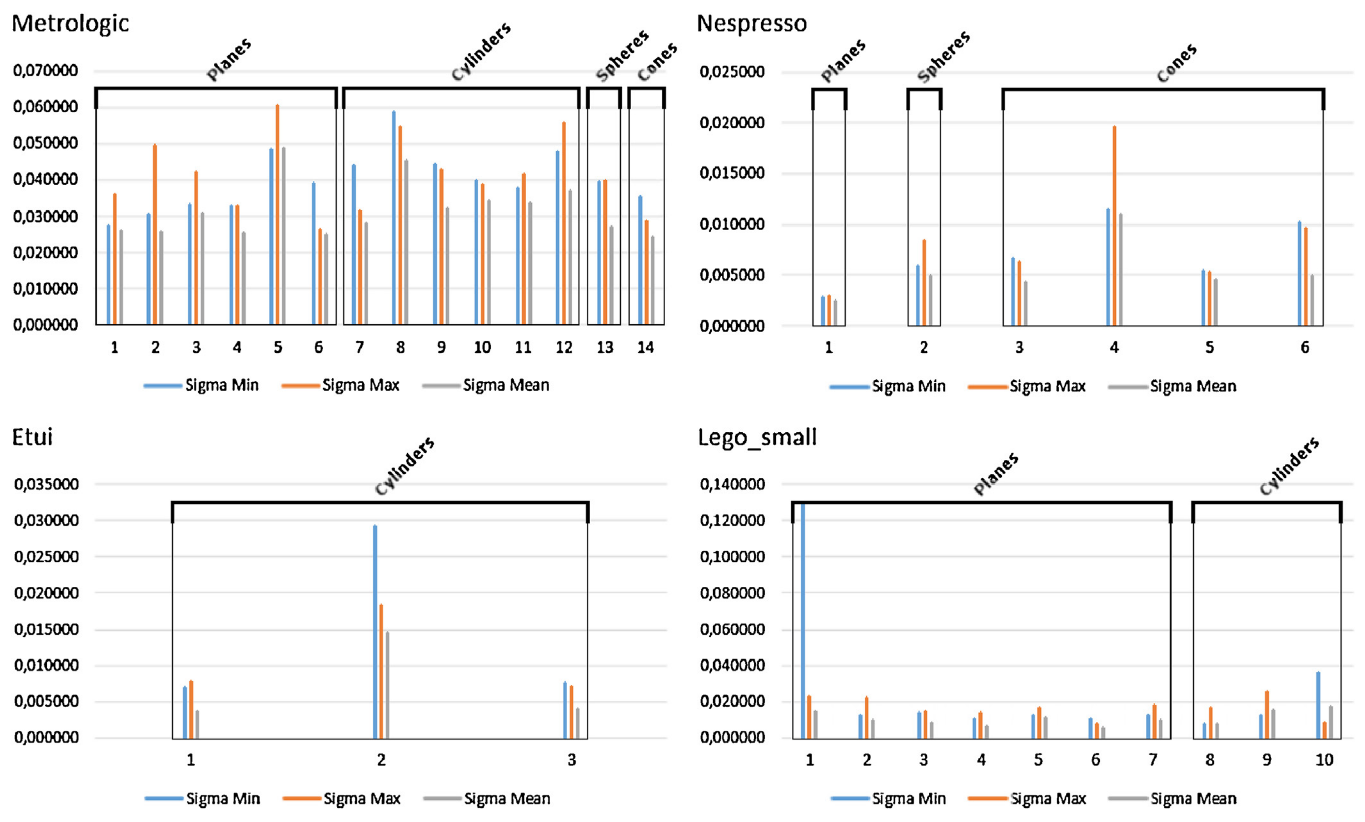


Fig. 30. Noise in submesh minimum (blue, left), maximum (orange, middle) and mean (gray, right) curvature distributions. Each position on horizontal axis represents a different submesh, and vertical axis shows the corresponding standard deviation values. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Measuring the quality: (a) without segmentation and (b) with segmentation.

(a) Without segmentation

Mesh	GMMs	Deviation	Mean	Weighted mean
Aerospace	3	0.015 0.015 0.013	0.014	0.015
Moldy	2	0.023 0.015	0.019	0.022
Outlet	3	0.027 0.028 0.034	0.030	0.028

(b) With segmentation

Mesh	Submeshes	Deviation	Mean	Weighted mean

Aerospace	70	min = 0.003
		max = 0.122
			0.013	0.009

Moldy	48	min = 0.001
		max = 0.086
			0.008	0.006

Outlet	72	min = 0.001
		max = 0.065
			0.013	0.010

4.3. Quality measurement

In many 3D processes, noise leads to distortion and has an impact on result accuracy. So to handle this noisy data, we can quantify it on curvature values.

From our curvature distribution approximated by a GMM, as defined in Section 3.2.2, we can retrieve standard deviation of each homogeneous interval, as illustrated in Fig. 19a, and so estimate a global value representing the noise quantity. For example, we can basically use the mean of standard deviations, before or after removing outliers.

In Fig. 19a, models number 6 and 7 could be outliers because a primitive leads to a sharp mode, whereas smooth ones are often due to noise. We can also weight each standard deviation by the number of corresponding points to compute the mean.

Furthermore, we can also analyze each standard deviation according to the surface type (primitive), size, position, and so characterize noise more accurately. Indeed, the noise depends on the scanner sensor type (depth, laser, . . .), but also on object material, texture, or even scene luminosity and object radiance. In fact, we can approximate curvature distortion between the digitized mesh (Fig. 19a) and the corresponding CAD mesh (Fig. 19b).

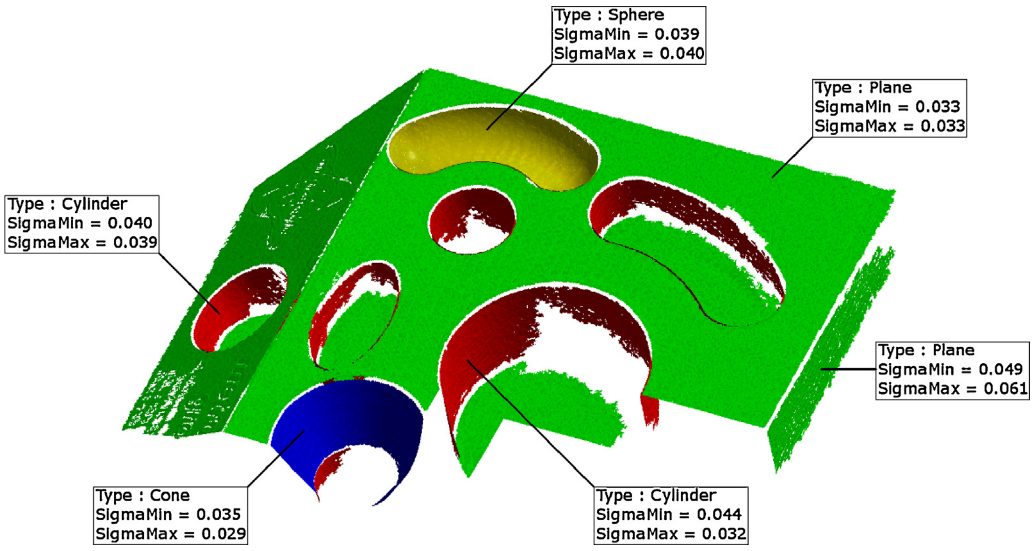


Fig. 31. Primitive type detection and noise evaluation on *Metrologic*.

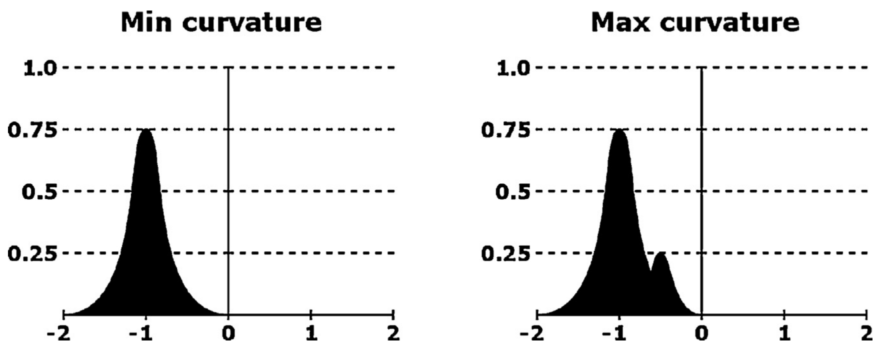


Fig. 32. Principal curvature histograms of the submesh containing the torus and the two spheres (Fig. 31).

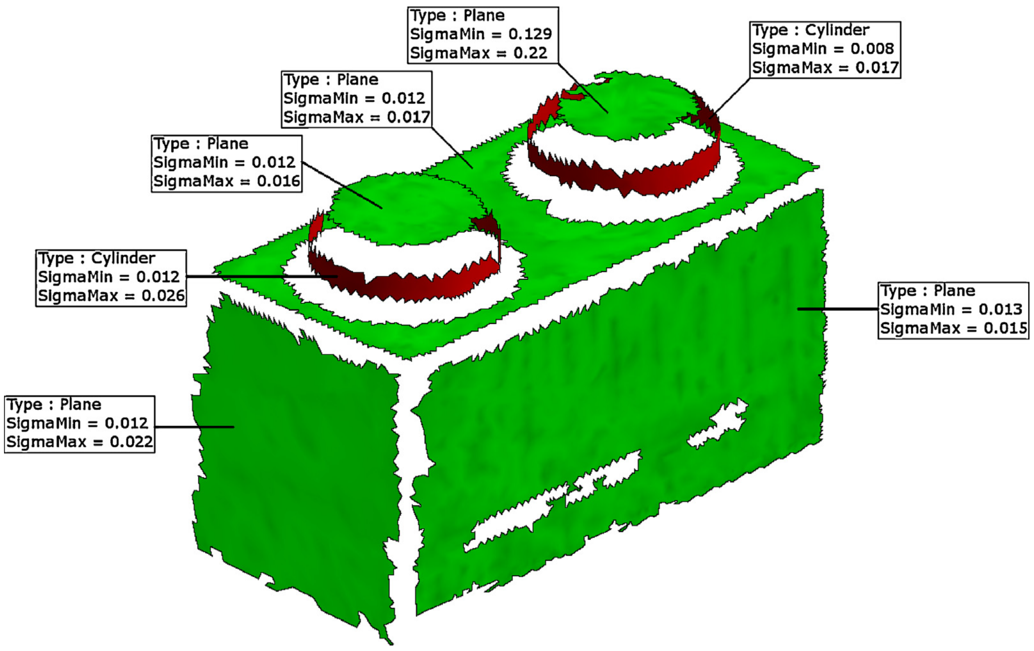


Fig. 33. Primitive type detection and noise evaluation on *Lego_small*.

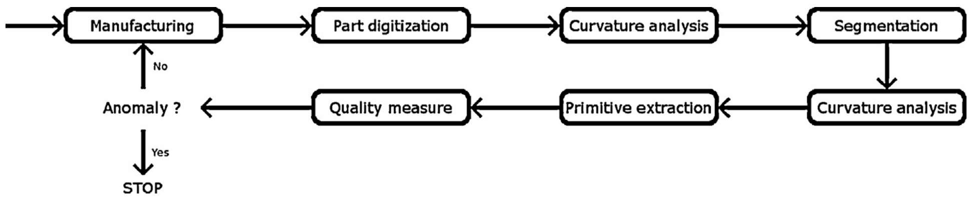


Fig. 34. Example: quality control in a production line.

We can also construct many distributions, like minimum, maximum, mean and Gaussian curvature histograms, and gather all the informations of those to obtain a better characterization. But it is very difficult to have a good, accurate and mostly exhaustive method allowing proper noise characterization. In previous work they try to quantify impact of noise on point coordinates, for depth sensor [33] or laser beam [34] scanners, and propose adapted algorithms.

In our case, we can even characterize the noise on each submesh from a segmentation (Section 4.1), and avoid object edge curvature errors in our analysis. So, it is possible to measure the noise depending on the mesh area, compare values and compute some statistics. This possibly leads to other applications like scanner noise characterization [37], for example.

5. Experimental results

Section 5.1 presents three meshes from different scanners. Then, we show our results on these meshes, for segmentation in Section 5.2, primitive type detection in Section 5.3 and quality measurement in Section 5.4.

5.1. Presentation of the three 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 Figs. 20 and 21 respectively. The third comes from a second scanner and is illustrated in Fig. 22.

To have an accuracy with an order of magnitude of −3 and center values around zero in the same bin, our method constructs histograms with 1001 bins. These histograms are constructed using a kernel with a standard deviation $h = 0.01$. Moreover, we limit our histogram range in the interval [−2; 2] since we normalize curvature by edge length.

5.2. Segmentation

This section presents results of our segmentation using curvature histogram analysis. Each figure shows edge extraction and the final submesh set. In reverse engineering, our results are very good, since most of the primitives are correctly isolated. 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. 23a) and the primitives are correctly isolated, except for a few tangent ones. We obtain 70 submeshes and 94.3% of them contain a single primitive (Fig. 23b).

The sharp edges of *Moldy* are properly detected (Fig. 24a) and the primitives are also correctly isolated. We note that freeforms are not over-segmented. We obtain 48 submeshes and all of them contain a single primitive (Fig. 24b).

The sharp edges of *Outlet* are properly detected (Fig. 25a), except for the serrated cylinders, and the primitives are almost all correctly isolated. We obtain 72 submeshes and all of them contain a single primitive (Fig. 25b).

As illustrated in Fig. 26, curvatures and thresholds are computed from each submesh at each recursion step. Then, we can continue to segment a submesh until we obtain only one region. We can observe that sharp edges are removed from the sharpest to the smoothest. Indeed, each recursion step refines the curvature thresholds to extract smoother edges.

This recursion has a major impact on the robustness of our method, which is fully automated and adapts to each step. For example, it is possible to correctly segment an object with heterogeneous noise.

We have segmented 30 meshes, with a processor Intel® Core™ i7-4710 CPU @ 2.50GHz. These meshes are extremely varied: they are generated from different softwares, with or without preprocessing, small or large, more or less noisy. Results are presented in Table 1, where bold names correspond to the three presented meshes (see Section 5.1).

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

To validate our approach, we count the number of submeshes that contain only one primitive. We can see in Table 2 that about 96% of submeshes match with only one primitive (remaining 4% can contain similar tangent primitives).

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

5.3. Primitive type detection

This section presents results of our geometric extraction using curvature histogram analysis to detect the primitive type and then adapt the tolerances. Each figure shows extracted primitives with tolerances computed by mesh analysis [7] and curvature analysis. The best mesh analysis results are obtained with parameter computations using edge lengths, numbers of points or elements and object size. We show that our results are better with curvature analysis because the tolerances are more accurate and suitable for the purpose. Indeed, we can determine the main primitive on a mesh (Section 4.2), and particularly on each submesh after segmentation. Since our segmentation gives submeshes with only one primitive (Table 2), we can more accurately compute the tolerances for each primitive independently.

Aerospace contains 71 primitives with mesh analysis (Fig. 27a). Some cylinders and cones are not extracted because they are often more noisy and so less stable than planes. We extracted them with curvature analysis and obtained 105 primitives (Fig. 27b).

Moldy contains 37 primitives with mesh analysis (Fig. 28a). The small cylinders are not extracted because the tolerances must be more accurate than those of larger ones. Curvature analysis resolves this problem and leads to extraction of 55 primitives (Fig. 28b).

Outlet contains 31 primitives with mesh analysis (Fig. 29a). All cylinders are missing, because they are too noisy or are serrated on this mesh. With curvature analysis, our tolerance adaptation balances the noise and we obtain 52 primitives (Fig. 29b). The serrated cylinders are still not very well rendered, but it is a relatively specific case where the curvature is not constant.

We have extracted primitives from 30 meshes after segmentation, and compared results between the mesh analysis proposed by Bénière et al. [2] and our proposed curvature analysis in Table 3. These results are related to regions containing only one primitive, presented in Table 2.

We can see that curvature analysis leads to a better primitive extraction, which is suitable for reverse engineering applications. Indeed, tolerances are more accurate and adaptative for each submesh, and take the primitive type into account. Moreover, curvature analysis tolerances can balance a larger amount of noise than mesh analysis.

5.4. Quality measurement

This section presents results of our proposed method using curvature analysis to quantify the noise of a digitized mesh.

We have analyzed the noise of four meshes after segmentation, i.e. the noise in each submesh (Fig. 30). We can see that the noise is often different between submeshes. Moreover, mean curvature is almost always less sensitive to noise. These results can lead to a better primitive extraction tolerance adaptation (Section 4.2), or use in other fields like scanner recognition and authentication [37].

To quantify mesh quality, we can compute statistics on this noise, and then give a global measurement. For example, we can compute a mean or a point number weighted mean of standard deviations. In the same way, we can compute the noise directly from the entire mesh, i.e. without segmentation, from the curvature distribution GMM (Section 4.3). Since we use a Gaussian kernel with a standard deviation h to construct our distribution (Section 3.1.2), we measure a mesh quality by:

$$\begin{array}{ll} \sigma_{Mean} < 2h & \rightarrow \textit{Good quality}, \\ 2h \leq \sigma_{Mean} < 3h & \rightarrow \textit{Middle quality}, \\ \sigma_{Mean} \geq 3h & \rightarrow \textit{Bad quality}. \end{array} \qquad (10)$$

Mean curvature analysis results on the three used meshes (Section 5.1) are compared in Table 4. First of all, we used the entire mesh and a GMM (Section 3.2.2). Then, we used the segmented mesh with its corresponding submeshes (Section 5.2).

Aerospace curvature distribution contains 3 Gaussian models which are similar with a low standard deviation. After segmentation, we obtained a similar basic average, but with a better weighted average, suggesting that there are large areas that are of a high quality. Globally, this mesh has good qualities.

Moldy curvature distribution contains 2 Gaussian models which are different with a low to middle standard deviations. After segmentation, the averages are significantly better, suggesting that distribution analysis on the entire submesh is not adapted. This mesh has an average quality whereas most of its submeshes have a very good quality.

Outlet curvature distribution contains 3 Gaussian models which are different with two middle and a high standard deviations. After segmentation, the averages are better, suggesting that distribution analysis on the entire mesh is not the best approach. This mesh has a high quality when evaluated on submeshes.

Globally, we can see through our results that curvature distribution analysis on a segmented mesh is better than on the entire mesh. Indeed, the submeshes have homogeneous curvature values and so are consistent for noise evaluation. Although, the entire mesh contains many primitives that are mixed in a unique distribution. So, a model can approximate many primitives at the same time. This leads to a degraded approximation and therefore a degraded mesh quality evaluation.

The noise evaluation, associated with primitive type detection after segmentation (Fig. 31), can give a quality coefficient depending on the type.

For example, we can compute the quality coefficient with:

$$\begin{array}{ll} \textit{if Plane}: & Q_f = SigmaMean, \\ \textit{if Sphere}: & Q_f = SigmaMean, \\ \textit{if Cylinder}: & Q_f = \max(SigmaMin, SigmaMax). \end{array} \quad (11)$$

Note that in this object, a submesh contains a torus with two sheres. Our method detects the submesh as spherical, because the minor radius of the torus equals one of the spheres. So, the associated curvature values are similar. However, the major radius leads to a small peak on the maximum curvature histogram, but not significant enough to disturb the method (Fig. 32).

In the same way, we can detect submeshes with higher noise than the others and have a better adaptation of our algorithms. To illustrate this, the upper right plane in Fig. 33 has a slightly higher noise than others for the minimum curvature distribution.

We can also use this to improve some processes. For example, we can apply a recursion on segmentation with noise handling, depending on the primitive type. Indeed, the interpretation of the noise from a plane is slightly different than that from a cylinder or a sphere. So, we can construct histograms with a different bin number and kernel standard deviation, according to this noise.

5.5. Case study: quality control

The proposed approaches developed in this paper can be used in many applications. In case of a process using reverse engineering, we can, for example, control the quality of some parts during a manufacturing process, as illustrated Fig. 34.

In fact, we can digitize separately each part of an object, then analyze their curvature distributions, as presented in Section 3. This allows us to segment the digitized meshes (Section 4.1) and then extract the geometric primitives (Section 4.2). Finally, we can measure distances between the primitives and the initial points (Section 4.3), but also angles and distances between the different primitives. We can thus quantify the quality of the manufactured object, and eventually stop the manufacturing process if an anomaly is detected, as illustrated Fig. 34.

Since our proposed methods are fast and automatic, this process can be used in real time for control on a production line.

6. Conclusion

In this paper, we proposed a new digitized 3D mesh shape analysis based on curvature analysis. Our proposed method is fast and fully automated, which is an advantage for industrial applications like reverse engineering for example. Our proposed analysis first constructs a continuous normalized curvature distribution, then searches for peak and valley positions and values, and finally provides some statistics computed from the curvature distribution.

Our histogram analysis leads to an automatic computation of some parameters. So, it can be useful for a large number of processes using many parameters, which often need to be fixed by an expert.

We chose a reverse engineering process because it is a growing research area, which becomes a hot topic for industrials. Indeed, it is used in many applications since it allows to retrieve directly a parametric model (and thus a CAD model if discretized) from a digitized object. We can also measure object deviation for quality control, reconstruct lost models, copy and understand how a mechanical object works for example. Moreover, our proposed applications can be a reliable base for primitive adjustments (like beautification), assembly or symmetry analysis.

We applied our method on three fields: 3D mesh segmentation, geometric primitive type detection and measurement of quality. We showed that our proposed method is accurate and adapted to many fields, through results on digitized 3D meshes from different scanners.

For 3D mesh segmentation, curvature thresholds are computed from the distribution to extract the object salient edges. Then, it is possible to retrieve isolated regions corresponding to the object primitives. The final submeshes are homogeneous, and each submesh matches with only one primitive. It also provides important information, like the primitive neighborhood through edge connectivity. Our segmentation is fast and automatic.

For primitive extraction, the type of the primitive is deduced from the distribution and then curvature tolerances are computed to fully adapt them to each 3D mesh. Then, more primitives are extracted and they are more accurate.

For quality measurement, statistics are computed in the first instance on the entire mesh, and then on each submesh after segmentation. These statistics, like standard deviation, quantify the noise of a mesh or its submeshes, and so the mesh quality. This quality measurement must be interpretated according to the distribution construction parameters.

We show that our three applications can be associated to improve results. These three fields show the extensibility and the robustness of our method, which can be used with any distribution to quickly and automatically adapt a method according to the input data.

In future work, we will analyze more precisely our curvature distribution construction parameters, like bin number or kernel standard derivation, to improve computing accuracy. In the same way, we will try to compute multiple primitive extraction tolerances in the case of 3D meshes with more than one type of primitive. We can also search for multi-resolution curvature distributions.

References

[1] P. Benkő, R. Martin, T. Várady, Algorithms for reverse engineering boundary representation models, Comput. Aided Des. 33 (11) (2001) 839–851, doi:http://dx.doi.org/10.1016/S0010-4485(01)00100-2.

[2] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, W. Puech, A comprehensive process of reverse engineering from 3D meshes to CAD models, Comput. Aided Des. 45 (11) (2013) 1382–1393, doi:http://dx.doi.org/10.1016/j.cad.2013.06.004.

[3] T. Gatzke, C. Grimm, Estimating curvature on triangular meshes, Int. J. Shape Model. 12 (01) (2006) 1–28, doi:http://dx.doi.org/10.1142/S0218654306000810.

[4] E. Magid, O. Soldea, E. Rivlin, A comparison of Gaussian and mean curvature estimation methods on triangular meshes of range image data, Comput. Vis. Image Underst. 107 (3) (2007) 139–159, doi:http://dx.doi.org/10.1016/j.cviu.2006.09.007.

[5] J. Chen, H. Feng, Automatic prismatic feature segmentation of scanning-derived meshes utilising mean curvature histograms, Virtual Phys. Prototyp. 9 (1) (2014) 45–61, doi:http://dx.doi.org/10.1080/17452759.2013.866874.

[6] A. Masmoudi, S. Chaoui, A. Masmoudi, A finite mixture model of geometric distributions for lossless image compression, Signal Image Video Process. 10 (2016) 671–678, doi:http://dx.doi.org/10.1007/s11760-015-0793-1.

[7] R. Bénière, G. Subsol, G. Gesquière, F. Le Breton, W. Puech, Recovering Primitives in 3D CAD meshes, SPIE Electronic Imaging 2011, 3D Imaging, Interact. Meas. 7864 (2011) 7864, doi:http://dx.doi.org/10.1117/12.872665 0R-1-9.

[8] X. Chen, F. Schmitt, Intrinsic Surface Properties from Surface Triangulation, Springer, Berlin, Heidelberg, 1992, pp. 739–743, doi:http://dx.doi.org/10.1007/3-540-55426-2_83.

[9] C. Dong, G. Wang, Curvatures estimation on triangular mesh, J. Zhejiang Univ. Sci. 6 (1) (2005) 128–136, doi:http://dx.doi.org/10.1631/jzus.2005.AS0128.

[10] J. Koenderink, A. van Doorn, Surface shape and curvature scales, Image Vis. Comput. 10 (8) (1992) 557–564, doi:http://dx.doi.org/10.1016/0262-8856(92)90076-F.

[11] C. Tang, G. Medioni, Robust estimation of curvature information from noisy 3D data for shape description in computer vision, The Proceedings of the Seventh IEEE International Conference, vol. 1 (1999) 426–433, doi:http://dx.doi.org/10.1109/ICCV.1999.791252.

[12] K. Watanabe, A. Belyaev, Detection of salient curvature features on polygonal surfaces, Comput. Gr. Forum. 20 (3) (2001) 385–392, doi:http://dx.doi.org/10.1111/1467-8659.00531.

[13] G. Lavoué, F. Dupont, A. Baskurt, A new CAD mesh segmentation method based on curvature tensor analysis, Comput. Aided Des. 37 (2004) 975–987, doi: http://dx.doi.org/10.1016/j.cad.2004.09.001.
[14] G. Mclachlan, T. Krishnan, The EM Algorithm and Extensions, Wiley-Interscience, 1996, doi:http://dx.doi.org/10.1002/9780470191613.
[15] A. Hadmi, W. Puech, B. Ait Es Said, A. Ai Ouahman, A robust and secure perceptual hashing system based on a quantization step analysis, Signal Process. Image Commun. 28 (8) (2013) 929–948, doi:http://dx.doi.org/10.1016/j.image.2012.11.009.
[16] Z. Huang, K. Chau, A new image thresholding method based on Gaussian mixture model, Appl. Math. Comput. (2008), doi:http://dx.doi.org/10.1016/j.amc.2008.05.130.
[17] K. Demarsin, D. Vanderstraeten, D. Roose, Meshless extraction of closed feature lines using histogram thresholding, Comput. Aided Des. Appl. 5 (5) (2008) 589–600, doi:http://dx.doi.org/10.3722/cadaps.2008.589-600.
[18] R. Sitnik, P. Blaszczyk, Segmentation of unsorted cloud of points data from full field optical measurement for metrological validation, Comput. Ind. 63 (1) (2012) 30–44, doi:http://dx.doi.org/10.1016/j.compind.2011.10.002.
[19] A. Shamir, A survey on mesh segmentation techniques, Comput. Gr. Forum. 27 (6) (2008) 1539–1556, doi:http://dx.doi.org/10.1111/j.1467-8659.2007.01103.x.
[20] S. Petitjean, A survey of methods for recovering quadrics in triangle meshes, ACM Comput. Surv. 2 (34) (2002) 1–61, doi:http://dx.doi.org/10.1145/508352.508354.
[21] P. Theologou, I. Pratikakis, T. Theoharis, A comprehensive overview of methodologies and performance evaluation frameworks in 3D mesh segmentation, Comput. Vis. Image Underst. 135 (C) (2015) 49–82, doi:http://dx.doi.org/10.1016/j.cviu.2014.12.008.
[22] S. Delest, R. Boné, H. Cardot, Fast segmentation of triangular meshes using waterfall, VIIP'06: International Conference on Visualization, Imaging and Image Processing (2006) 308–312.
[23] M. Garland, A. Willmott, P. Heckbert, Hierarchical face clustering on polygonal surfaces, SI3D'01: Proceedings of the 2001 Symposium on Interactive 3D Graphics (2001) 49–58, doi:http://dx.doi.org/10.1145/364338.364345.
[24] D. Kim, I. Yun, S. Lee, Boundary-trimmed 3D triangular mesh segmentation based on iterative merging strategy, Pattern Recognit. 5 (39) (2006) 827–838, doi:http://dx.doi.org/10.1016/j.patcog.2005.11.022.
[25] Y. Lai, Q. Zhou, S. Hu, R. Martin, Feature sensitive mesh segmentation, SPM'06: Proceedings of the 2006 ACM Symposium on Solid and Physical Modeling (2006) 17–25, doi:http://dx.doi.org/10.1145/1128888.1128891.
[26] L. Di Angelo, P. Di Stefano, Geometric segmentation of 3D scanned surfaces, Comput. Aided Des. 62 (2015) 44–56, doi:http://dx.doi.org/10.1016/j.cad.2014.09.006.
[27] A. Courtial, E. Vezzetti, New 3D segmentation approach for reverse engineering selective sampling acquisition, Int. J. Adv. Manuf. Technol. 35 (9) (2008) 900–907, doi:http://dx.doi.org/10.1007/s00170-006-0772-3.
[28] M. Attene, B. Falcidieno, M. Spagnuolo, Hierarchical mesh segmentation based on fitting primitives, Vis. Comput. Int. J. Comput. Gr. 3 (22) (2006) 181–193, doi:http://dx.doi.org/10.1007/s00371-006-0375-x.
[29] J. Wang, D. Gu, Z. Yu, C. Tan, L. Zhou, A framework for 3D model reconstruction in reverse engineering, Comput. Ind. Eng. 63 (4) (2012) 1189–1200, doi:http://dx.doi.org/10.1016/j.cie.2012.07.009.
[30] V. Weiss, L. Andor, G. Renner, T. Várady, Advanced surface fitting techniques, Comput. Aided Geom. Des. 19 (1) (2002) 19–42, doi:http://dx.doi.org/10.1016/S0167-8396(01)00086-3.
[31] J. Chen, H. Feng, Idealization of scanning-derived triangle mesh models of prismatic engineering parts, Int. J. Interact. Des. Manuf. (2015) 1–17, doi:http://dx.doi.org/10.1007/s12008-015-0262-7.
[32] T. Tran, V. Cao, D. Laurendeau, Extraction of reliable primitives from unorganized point clouds, 3D Res. 6 (4) (2015) 1–12, doi:http://dx.doi.org/10.1007/s13319-015-0076-1.
[33] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, R. Siegwart, Noise characterization of depth sensors for surface inspections, 2nd International Conference on Applied Robotics for the Power Industry (CARPI) (2012) 16–21, doi:http://dx.doi.org/10.1109/CARPI.2012.6473358.
[34] H. Masuda, I. Tanaka, M. Enomoto, Reliable surface extraction from point-clouds using scanner-dependent parameters, Comput. Aided Des. Appl. 10 (2) (2013) 265–277, doi:http://dx.doi.org/10.3722/cadaps.2013.265-277.
[35] Lavoué, Guillaume, A local roughness measure for 3D meshes and its application to visual masking, ACM Trans. Appl. Percept. 5 (4) (2009), doi: http://dx.doi.org/10.1145/1462048.1462052 21:1–21:23.
[36] M. Carreira-Perpiñán, A Review of Mean-shift Algorithms for Clustering, (2015) CoRR, abs/1503.00687.
[37] A. Kharboutly, W. Puech, G. Subsol, D. Hoa, Improving sensor noise analysis for CT-Scanner identification, 23rd European Signal Processing Conference (EUSIPCO) (2015) 2411–2415, doi:http://dx.doi.org/10.1109/EUVIP.2014.7018385.