algorithms

MDPI

Article

Analytic Form Fitting in Poor Triangular Meshes

Cristian Rendon-Cardona [1,2], Jorge Correa [2], Diego A. Acosta [3] and Oscar Ruiz-Salguero [1,*]

[1] Laboratory of CAD CAM CAE, Universidad EAFIT, Cra 49 no 7-sur-50, Medellín 050022, Colombia; crendo11@eafit.edu.co
[2] Manufactura Cohesiva SAS, Cra 32B #10-30, Medellín 050021, Colombia; jorge.correa@cohesivemanufacturing.com
[3] Grupo de Desarrollo y Diseno de Procesos (DDP), Universidad EAFIT, Cra 49 no 7-sur-50, Medellín 050022, Colombia; dacostam@eafit.edu.co
* Correspondence: oruiz@eafit.edu.co

Abstract: Fitting of analytic forms to point or triangle sets is central to computer-aided design, manufacturing, reverse engineering, dimensional control, etc. The existing approaches for this fitting assume an input of statistically strong point or triangle sets. In contrast, this manuscript reports the design (and industrial application) of fitting algorithms whose inputs are specifically poor triangular meshes. The analytic forms currently addressed are planes, cones, cylinders and spheres. Our algorithm also extracts the support submesh responsible for the analytic primitive. We implement spatial hashing and boundary representation for a preprocessing sequence. When the submesh supporting the analytic form holds strict C^0-continuity at its border, submesh extraction is independent of fitting, and our algorithm is a real-time one. Otherwise, segmentation and fitting are codependent and our algorithm, albeit correct in the analytic form identification, cannot perform in real-time.

Keywords: analytic form fitting; poor triangular meshes; spatial hashing; real time

Citation: Rendon-Cardona, C.; Correa, J.; Acosta, D.A.; Ruiz-Salguero, O. Analytic Form Fitting in Poor Triangular Meshes. *Algorithms* **2021**, *14*, 304. https://doi.org/10.3390/a14110304

Academic Editor: Frank Werner

Received: 10 July 2021
Accepted: 23 September 2021
Published: 22 October 2021

1. Introduction

1.1. Research Target

This manuscript reports an algorithm for analytic form (cone, cylinder, sphere) submesh extraction and fitting, using as input low statistical quality triangular meshes. Our algorithm has been successfully applied in actual industrial low-quality datasets, when good quality ones are impractical due to their large size and processing times. Our method favors sequential small submesh probing and vector closure formulation for the identification of analytic form parameters. Our method avoids, as much as possible, multi-variable regression, which requires a statistically rich input.

1.2. Context

Point samples of a two-manifold (surface) in 3D are achieved by contact, ultrasound or optical scanning. Under Nyquist–Shannon conditions of this geometrical point cloud information, topological (i.e., connectivity) information may be inferred. A widely used result of the topology inference is the triangular mesh surface approximation. This manuscript addresses the next step of shape inference, namely the fitting of a analytic form $g()$ to a subset of the triangular mesh M.

This manuscript is concerned with triangular meshes whose facets strongly depart from equilateral, isotropic, and homogeneous size conditions. The reason for this choice is that in many cases, mesh conditioning is not only labor-expensive but is produces massive triangle sets. Fitting analytic form to heavy datasets is prohibitively slow for many industrial applications.

A strong demand exists in industrial applications for planar, cylindrical, conical and spherical forms, identified in low-quality triangle sets. Therefore, this manuscript

addresses those surface types, tessellated with such low-quality triangles. The implicit form ($g(x, y, z) = k$) is pursued. In this manuscript, $g()$ equally notes the parametric or implicit forms, as well as the defining feature set of the form, as follows:

a For cylinders: axis pivot point p_0, axis direction $\hat{v}$, radius R.
b For cones: axis pivot point p_0, axis direction $\hat{v}$, cone angle γ.
c For spheres: center pivot point p_0, radius R.

This manuscript addresses the following problem:

Given:

1 $M(\mathcal{T}, \mathbf{P})$: triangular mesh representing a shell with manifold properties.
2 $t_s \in \mathcal{T}$: seed triangle (selected by the user).
3 $\mathscr{T}$: type of analytic form (selected by the user from the options cylinder, sphere, and cone).

Goal:

i S: largest connected subset of M, formed by triangles from M, containing t_s and fitting the analytical form of $\mathscr{T}$.
ii $g()$: analytic form of the chosen primitive, fit to subset S.

where the analytic form is expressed by the identified parameters in (a)–(c) above.

It is important to note that the continuity level (C^0, C^1 or higher) at the borders ∂S triangle subset S and $M - S$ has a dramatic impact on the decoupling of solutions for items (i) and (ii) above. If the surface M has only C^0-continuity at border ∂S, this fact determines S, without influence from the fitting stage. Then, the fitting of $g()$ proceeds in real time. On the other hand, if M presents continuity C^1 or higher at border ∂S, there is a codependency between S and $g()$. S can be extended only as far as it produces a good quality fit $g()$. Low-quality $g()$ fitting indicates that the subset S includes mistaken triangles. Those triangles must be eliminated from S, other triangles might be included, and a new guess for $g()$ is computed. This iteration is obviously more expensive than the one in which identification of S does not depend on the quality of $g()$ fitting.

This manuscript is organized as follows: Section 2 reviews the existing literature and argues our claim for novelty in our initiative. Section 3 explains the methodology followed. Section 4 conveys the results of our implementation and compares it in some aspects against similar competitor approaches. Section 5 concludes the manuscript and discusses relevant future endeavors.

2. Literature Review

This section executes a taxonomy on the prevailing methods for fitting of analytic forms to point of triangle sets. The available literature may be divided into (1) stochastic, (2) parameter-space, and (3) mesh segmentation fitting methods. This taxonomy is based on the one described by Ref. [1].

2.1. Stochastic Methods

These methods randomly segment the input (point or triangle) set. They apply local regressions with alternative goal primitives (cylinder, cone, sphere, etc.), keeping the one with the best fit. The methods allow for the evolution of the primitive type and its parameters as well as evolution of the partition of the input mesh into local support submeshes. The methods require a dense, isotropic, homogeneous input set.

Ref. [2] requires point clouds with surface normal information. It fits cones and cylinders by solving local linear regressions. It intends to keep a small support point set, at the expense of actually producing a collection of guesses for the user to choose from. Ref. [3] presents analytical fitting of cones, cylinders and ellipsoids from dense, noisy point clouds. Ref. [3] finds an initial guess for the analytical surface by calculating their characteristic variables. Then, tuning of the analytic form parameters takes place by minimizing the point vs. analytic surface accumulated distances.

2.2. Mapping to Parameter Space

In these methods, type $\mathscr{T}$ of the analytic form is assumed. For a given point sample support set, a regression is performed, thus finding an estimate of the n parameters which that particular form type has. A point in the parameter space R^n is then located with such values. As other neighborhoods of the input point set are used as support sets, a point population of R^n arises, with clusters in the most likely parameter combinations. These high-density clusters are identified with standard statistical tools and the parameters of the analytic form are determined. Mapping to parametric space is particularly expensive when computing (storage), sharply growing with n, the dimension of the R^n parameter space. They also require dense input point sets. Due to these reasons, these methods are mostly applied to identification of polygonal flat faces in a polytope.

Ref. [4] uses the Hough transform as mapping to parametric space to identify planes. Ref. [5] lowers the high cost of detecting cylinders (parameter space R^6) by breaking the six parameter sets (i.e., 6D Hough transform) into axis vector, axis pivot, and cylinder radius and carrying sequential lower-dimension Hough transforms.

2.3. Mesh Segmentation Fitting Methods

In these methods [6–8], the priority is to segment the input (point-normal vector set or point graph). The fitted analytic form is a by-product. These methods create input clusters using diverse strategies: fitting-segmentation iteration, grouping using X-means or mean shift (separate segmentation), or classification by ranges of additional information (scalar or vector fields) such as color, depth, abrupt dihedral angle, etc. These methods have limitations as they need at least one of the additional pieces of information mentioned above and also require dense datasets.

Ref. [6] describes a hierarchical segmentation of dense point clouds. This method generates the k-nearest graph by merging edges of the input connectivity graph. This merging uses as criteria a penalty function related to the quality of fitting of the point clusters to a given primitive (one of sphere, cone, plane, torus). Ref. [7] applies segmentation-fitting iterations seeking to simultaneously segment the full input set and to fit quadratic analytic primitive options (plane, sphere, cylinder, ellipsoid, paraboloid) to the support subsets. It executes an initial segmentation of the input dataset before the iterative process. Ref. [7] does not provide detailed information on such a preprocessing. Ref. [8] executes a color-based segmentation of indoor 3D depth map scenarios. Then, a matching is conducted between the generated patches and a template database of predefined 3D indoor furniture models. Finally, the user refines the segmentation. The authors train a matching algorithm of random-regression forest to obtain the desired result. The process of matching the support set S to various reference models is an expensive task and is out of our scope since we do not need to train any learning algorithm. Ref. [9] executes a rough segmentation of a dense good quality point cloud or triangular mesh. This segmentation seeks to find neighborhoods of the input data which present *slippage* compatibility, i.e., which can slide onto themselves (e.g., spheres, cylinders, planes). The segmentation is carried out by using a greedy clustering, based on the local slippage compatibility. This reference aims for an approximate segmentation. It does not determine the $g()$ analytic form, nor presents execution times or complexity information.

2.4. Conclusions of Literature Review

The literature reveals that most fitting techniques require statistically dense datasets complying with the Nyquist–Shannon theorem. Additionally, these input datasets must include supplementary information (i.e., scalar maps or vector fields). Therefore, our approach offers (1) fitting of analytic surfaces to statistically poor triangulations as the industry demands it. (2) When the continuity and the border ∂S is exclusively C^0, we provide real-time performance. (3) When the border ∂S presents C^1 or higher continuity with $M - S$, segmentation and fitting are codependent. The fitted form $g()$ guarantees the perfecting of the support subset S. In turn, the support subset S provides the grounds

for the $g()$ fitting. Table 1 presents a summary of the different approaches with their advantages and drawbacks.

Table 1. Competitor approaches vs. current approach.

Approach	Refs.	Advantages	Disadvantages
Stochastic Methods. Randomly apply local regressions of alternative goal primitives and keeping the best fit.	[2,3,10–12]	(1) Robust to outliers; (2) application-specific	(1) Requires dense, isotropic, homogeneous input dataset.
Mapping to Parameter Space. Implement a mapping to the parameter space, identify the number of parameters needed and search for clusters of similar parameters.	[4,5,13,14]	(1) Robust to noise and outliers; (2) work with missing data	(1) High computational cost; (2) requires dense datasets
Mesh Segmentation Fitting Methods. Create clusters of the dataset using different grouping techniques. Fitting can occur during or after the segmentation.	[6–9,15–17]	(1) Spatial consistency	(1) Sensitive to noise and outliers; (2) require supplementary information in the dataset.
Our approach. A primitive fitting in poor triangular meshes. Segmentation driven by dihedral angle is implemented when continuity is strict C^0. Segmentation and fitting are codependent for C^1 or higher continuity.		(1) Fitting in poor datasets; (2) low computational cost; (3) real-time performance for C^0-continuity; (4) perfecting of S for C^1 or higher continuity.	(1) Requires the type of analytic form intended to be fitted as an input.

3. Methodology

Fitting of an analytic primitive $g()$ to the support (triangle) subset S requires (obviously) the determination of S. Extraction of support subset S from the input set M is heavily based on triangle neighborhood interrogations on M. The acceleration of neighborhood queries on M was implemented by building the triangular boundary representation of the triangle set. This construction was itself accelerated by applying a spatial hashing process on input M (Spatial Hashing based on techniques from [18,19]). Spatial hashing and B-Rep construction are reported in Section 3.1 on preprocessing. Section 3.2 discusses the specific problem of inferring the analytic form $g()$ from a triangle set S. Sections 3.3 and 3.4 address the determination of triangle subset S for applying Section 3.2. Section 3.4 focuses on cases where the border between support subset S and $M - S$ is sharp (C^0-continuity), thus allowing identification of S by using continuity information alone. Section 3.4 addresses more difficult cases where continuity at the border between S and $M - S$ is C^1 or better. In these cases, S is legitimated by a good fitting $g()$, but at the same time, $g()$ depends on S. In these cases, fitting ($g()$) and segmentation (S) are codependent and cannot be executed independently from each other.

Preprocessing

The process of extracting and identifying analytic forms from a manifold triangle set (Figure 1) is preceded by a preprocessing stage encompassing: (1) a spatial hashing; (2) boundary representation (B-Rep) construction.

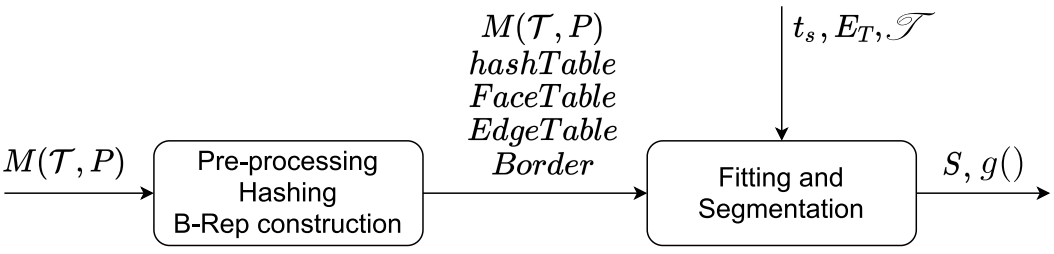


Figure 1. Overall primitive extraction process.

Analytic Form Fitting

If the triangle subset S supporting the analytic form has C^0 -continuity alone at the border ∂S between S and M_S, the determination of S is exclusively based on the dihedral angle at ∂S (red box, left portion of Figure 2). Otherwise, if the continuity at ∂S is C^1 or higher, the identification of support subset S and the fitting of $g()$ on that S are codependent (blue box, right portion of Figure 2). Triangles are added to S, or rejected, according to their effect on the fitting quality of the analytic form $g()$. Growth of the support subset S stops when all triangles at border ∂S have an adverse effect on $g()$.

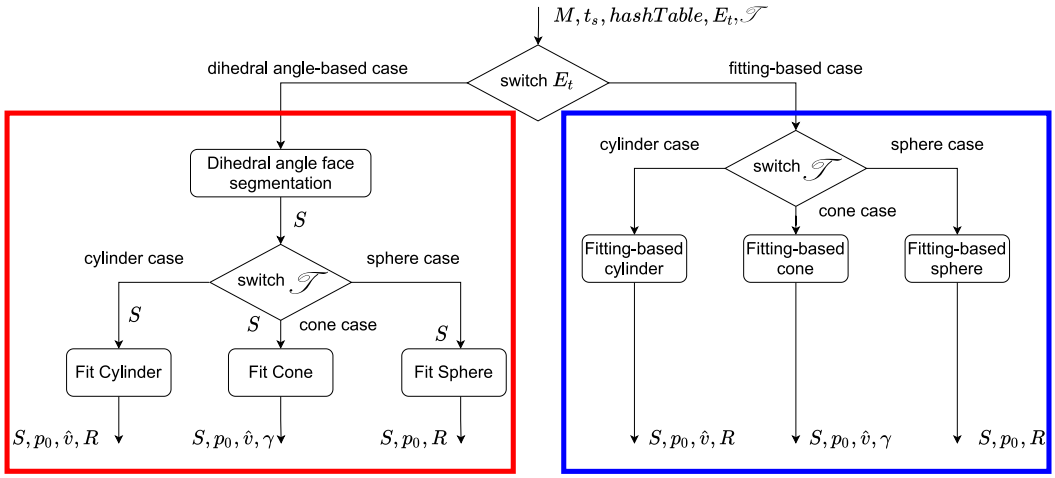


Figure 2. Fitting process. **Left** (red) block: segmentation/fitting decoupled. Real-time identification of $g()$. **Right** (blue) block: codependent segmentation and fitting.

3.1. Preprocessing

3.1.1. Spatial Hashing

Spatial hashing on the input triangle set M is used in this work for accelerating the construction of explicit triangle neighborhood information (boundary representation). The I/O specification of the spatial hashing preprocessing follows.

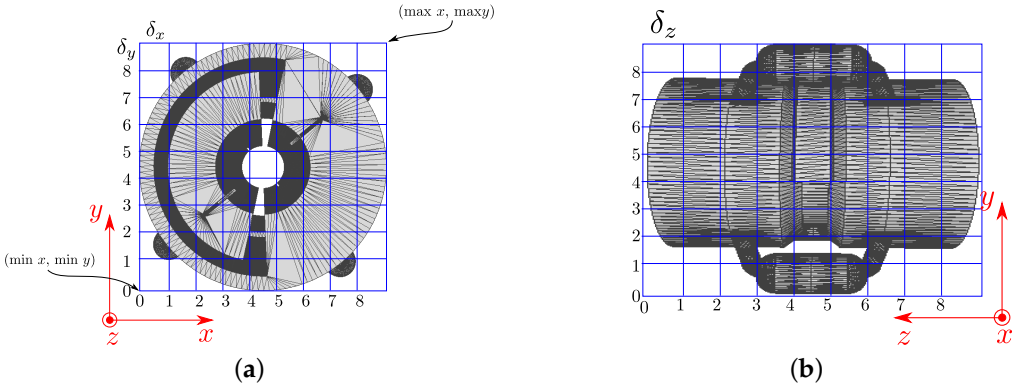
Input:

1. $M(\mathcal{T}, \mathbf{P})$: Triangle (two-manifold) set with geometry **P** and topology $\mathcal{T}$.
2. Ω: Rectangular prism in R^3, orthogonally oriented w.r.t. coordinate axes, equipped with a regular grid of voxels with dimension $delta_x$, $delta_y$, $delta_z$.
3. N_v : A predefined number of voxels per dimension of Ω.

Output:

1. h: Hashing function: $h : P \rightarrow N \times N \times N$, which maps each vertex of the input triangle set M into the (i,j,k) indices of the voxel $v_{ijk} \subset \Omega$, which contains such a vertex.
2. H: $\Omega \rightarrow T^2$, which maps each voxel v_{ijk} of Omega to the set of triangles of M which contain a vertex inside voxel v_{ijk}. 2^T denotes the power set of T (all sets made with triangles from T).

The algorithm divides the prism Ω into a set of rectangular voxels. These voxels are represented by a three-dimensional array where each cell contains the indices of the triangles with vertices in that voxel. Figure 3 shows a graphical representation of the voxel subdivision for an input triangle set.



(**a**) (**b**)

Figure 3. (**a**) Bounding box and (**b**) rectangular voxel subdivision.

The hash function $[i, j, k] = h(p_i)$ maps a point $p_i \in \Omega$ to a position $[i, j, k]$ in the hash table H.

$$\begin{aligned} i &= floor((p_i.x - \Omega.x_{min})/\delta_x), \\ j &= floor((p_i.y - \Omega.y_{min})/\delta_y), \\ k &= floor((p_i.z - \Omega.z_{min})/\delta_z). \end{aligned} \tag{1}$$

where $\Omega.x_{min}$, $\Omega.y_{min}$, $\Omega.z_{min}$ are constants corresponding to the minimal positions of the prism Ω. δ_x, δ_y, δ_z are the x, y, and z voxel side dimensions.

Figure 4 represents a block a diagram of the algorithm and how the vertices are stored in the hash table. As the input triangles sets are representation of two-manifold shells, H is a sparse array.

Notice that the spatial hashing $[H, h]$ ensures that all triangles incident on (EDGEs of) a given triangle are stored in at most three voxels of the hash array H, and they are directly accessible via the function $h()$. Due to this reason, the spatial hashing accelerates the construction/weaving of the boundary representation of the input triangle set M.

3.1.2. Boundary Representation Construction

After the spatial hashing process, a boundary representation, B-Rep, of the input triangle set M is constructed. We use a variation of the half-edge data structure [20]. The construction of the B-Rep produces the FACEs, EDGEs, VERTEXs and BORDER tables (see Tables 2–5).

Input:

- A two-manifold triangular mesh $M(\mathcal{T}, P)$.

Output:

- Boundary representation for M (Tables 2–5).

Table 2. VERTEX table.

Vertex	x	y	z	Edge
1	2.34	3.00	1.12	13
2	2.22	9.00	10.36	27
3	5.20	4.00	9.12	53

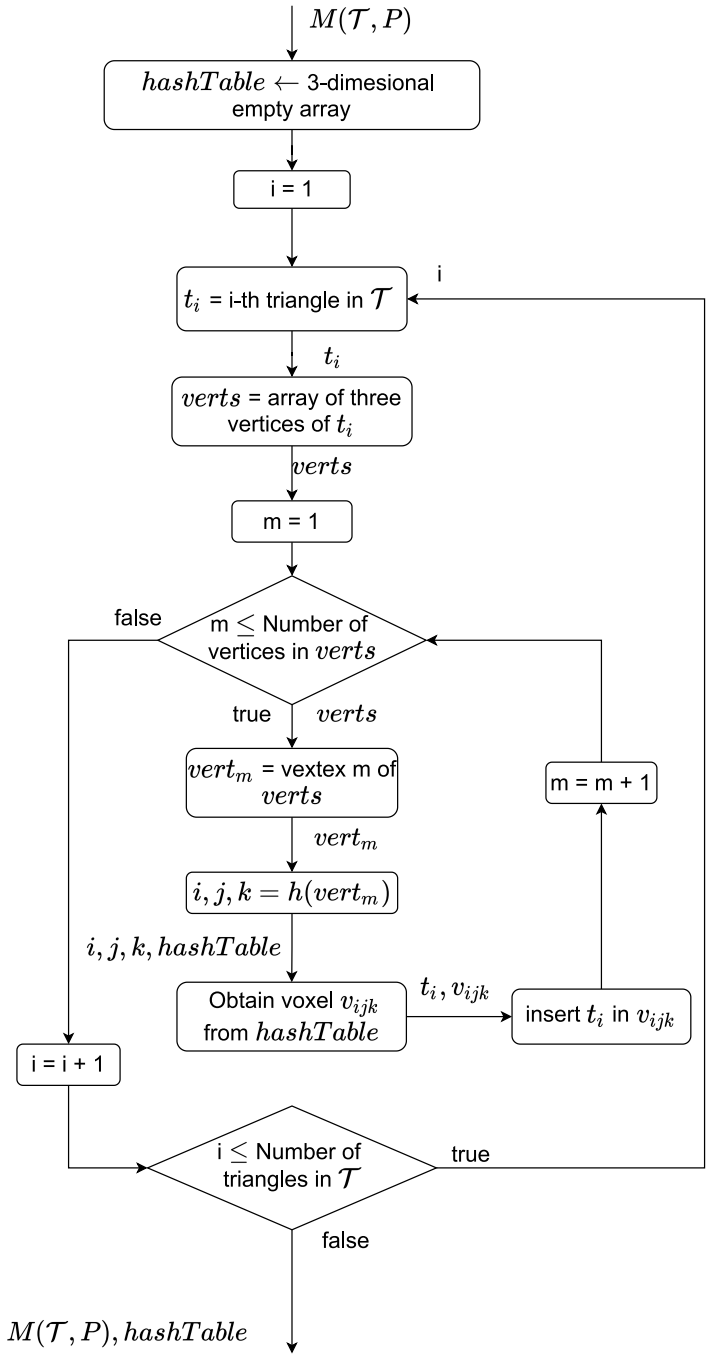


Figure 4. Spatial hashing preprocessing. The hashing is executed before the fitting process starts.

Table 3. FACE table constructed from B-Rep.

Face	Edge 1	Edge 2	Edge 3
1	1	2	3
2	3	5	4
3	3	1	4

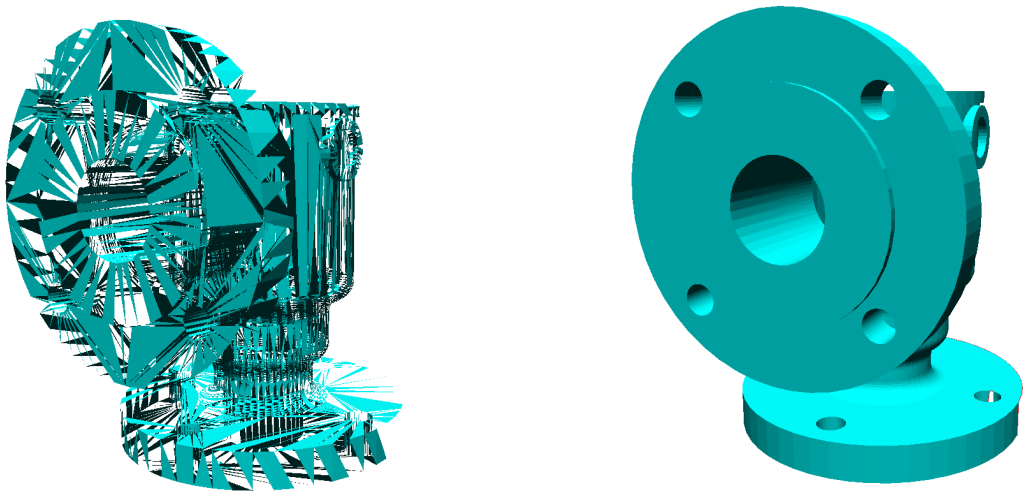
Table 4. EDGE table. If counter-EDGE cell is empty, indicates border.

Edge	Vertex 1	Vertex 2	Counter-EDGE	Face
1	1	2	4	1
2	2	3	5	1
3	3	4	-	1

Table 5. BORDER table.

Border Loop	Vertex 1	Vertex 2	Vertex 3	Vertex 4	Vertex 5	Vertex 6	...
1	4	7	8	91	<End-of-loop>		
2	10	71	31	11	1	90	<End-of-loop>
3	63	83	47	121	9	<End-of-loop>	

In the present case, the SHELL table is omitted because the input mesh is connected (i.e., there is only one SHELL). The B-Rep guarantees that all FACEs within a SHELL have consistent orientation. The B-Rep delivers constant time expenses $O(k)$ in the following interrogations: (a) EDGES incident on a VERTEX, (b) FACEs incident on an EDGE, (c) FACEs neighboring a FACE, (d) ordered LOOPs of a SHELL border, (e) SHELL owner of a FACE, (f) EDGEs of a LOOP, (g) LOOPs of a FACE. Figures 5 and 6 present plain-sight visible effects of building the B-Rep of two input triangle sets M. Those visible effects are the consistent orientation of the FACEs in a SHELL and the identifications of the SHELL BORDER.



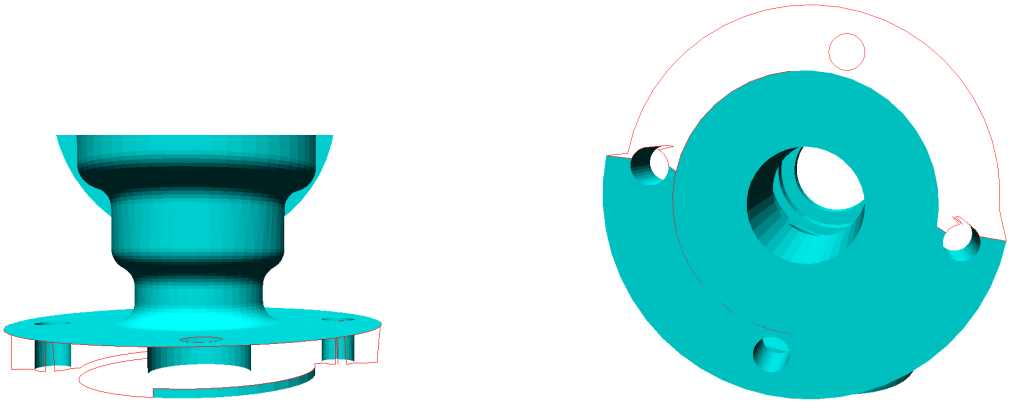
(**a**) Raw triangle set M. Disoriented triangles become transparent.

(**b**) B-Rep woven from the raw triangle set M. No disoriented triangles are present.

Figure 5. Input triangle set M with inconsistent FACE orientation. Output B-Rep of M. (**a**) and (**b**) displayed with culling based on the triangle orientations.

3.2. Primitive Fitting from Triangle Set

This section addresses the determination of the parameters of a given analytic form (cylinder, cone, sphere) of type $\mathscr{T}$, which the user wants to fit to a triangle set S. The strategy used in our algorithm is to locally *probe* the triangle set, progressively determining parameters of the form (radius, center, axis, apex, etc.). This strategy was chosen instead of *statistical* fitting, which requires statistically meaningful (dense, large, homogeneous, isotropic) triangle set inputs. Our implementation does use quadratic form fitting for the purpose of determining minimal and maximal curvature directions on specific and small neighborhoods of the triangle set S.



(**a**) (**b**)

Figure 6. Result of border identification in the triangle set. (**a**) and (**b**) present the identified border of the triangle set after weaving the B-Rep.

The fitting of the $g()$ analytic form to set $S \subset M$ is indifferent to the manner in which triangle subset S is determined. The description of the fitting follows.

Input:

1. S: triangle support set ($S \subset M$) to fit the analytic form $g()$.
2. $\mathscr{T}$: primitive type, selected by the user from the set {Cylinder, Sphere, Cone}

Output:

1. $g()$: The analytic form of the chosen primitive, fit to subset S.

For each analytic form, we implement an approach for the identification of its parameters.

3.2.1. Cylinder Fitting

The geometrical parameters to define the cylinder are:

1. p_0: axis pivot point.
2. $\hat{v}$: axis direction.
3. R: radius.

Our algorithm takes advantage of the fact that the input triangle set provides information on the vectors locally normal to the cylinder. This information enables the calculation of p_0 and $\hat{v}$.

Axis point p_0

The axis pivot point is calculated as follows:

1. Create a set of lines $L_n = \{\ell_1, \ell_2, ..., \ell_n\}$, where ℓ_i is a line generated by the baricenter b_i and the normal n_i of the triangle t_i.
2. Obtain C_p: as the set containing the points of the minimum distance between each pair of lines from L_n. Every line ℓ_i is approximately perpendicular to, and nearly intersects, the cylinder axis $[p_0, \hat{v}]$. Therefore, C_p is a point sample of the axis $[p_0, \hat{v}]$.
3. Compute p_0 as the geometric center of gravity of C_p.

Figure 7 presents an axial view of a cylindrical triangle set. Lines $[b_i, n_i]$ ideally intersect, and are perpendicular to, the cylinder axis $[p_0, \hat{v}]$. Figure 8 displays an actual example set C_p and cylinder axis identification.

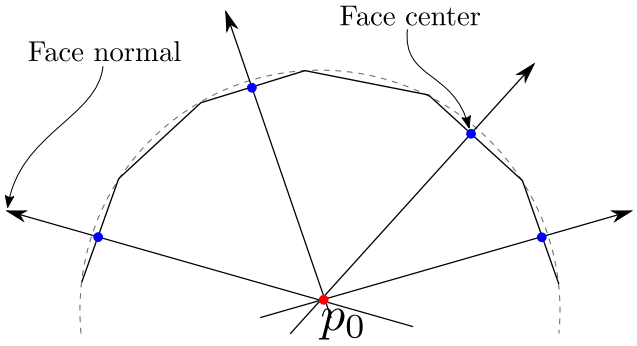


Figure 7. p_0 calculation with the intersection of lines $[b_i, n_i]$ whose directions are triangle normal vectors n_i and pass through triangle baricenters b_i.

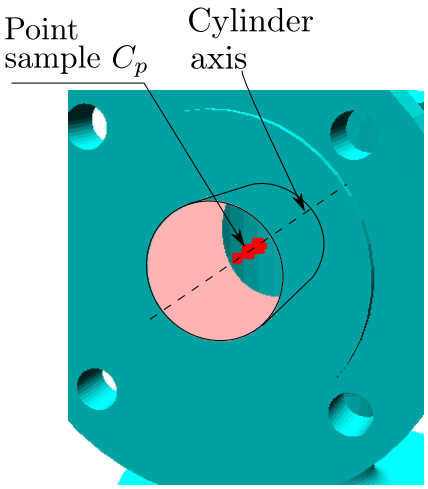


Figure 8. Set of points C_p approximating the cylinder axis.

Cylinder Axis direction $\hat{v}$

To calculate the cylinder axis, the method relies on the ability to:

1. Translate the unitary vectors n_i normal to the triangles from their supposed pivot point (triangle baricenter b_i) to the origin.
2. Collect the tips of the origin-based vectors resulting from step (1). These tip points form a tilted flat disk d centered at the origin (Figure 9).
3. Obtain (via Principal Component Analysis (PCA)) the cylinder axis direction $\hat{v}$ as the one of smallest dispersions of the points of disk d (Figure 9).

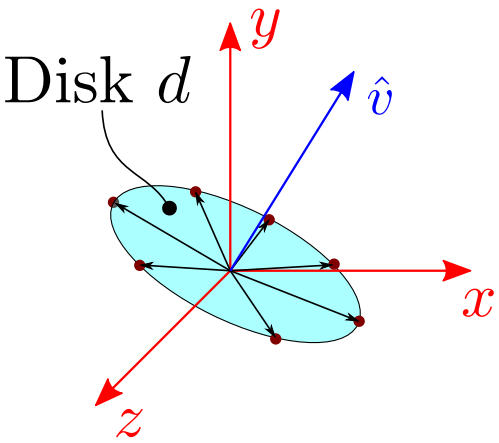


Figure 9. Cylinder fitting. Coplanar circular set of tips of cylinder normal unitary vectors, applied at origin. $\hat{v}$: direction of minimal dispersion of disk d, cylinder axis direction.

Cylinder Radius *R*

Knowing the cylinder axis $[p_0, \hat{v}]$, the radius R of the cylinder is determined as follows:

1. Rigidly move the triangle set S so its axis $[p_0, \hat{v}]$ maps to the Y_w World axis and the geometric center of S maps to the origin O_w.
2. Project the point set resulting from (1) onto the $X_w Z_w$ plane, resulting in a flat circular point set.
3. Compute the radius of such a circular point set.

3.2.2. Cone Fitting

The geometrical parameters of the cone are:

1. p_0: cone apex.
2. $\hat{v}$: axis direction.
3. γ: opening angle.

Cone axis direction $\hat{v}$

As in the cylinder case (Section 3.2.1), the vectors normal to the cone triangles allow us to estimate the direction of the cone axis.

The unitary vectors n_i normal to the cone triangles are translated to the origin (Figure 10). Their tips form a planar circle which is the base of a dual cone whose apex is at the origin. Observe that this dual cone *is not* the one contained in triangle input S. This *dual* cone in Figure 10 is obtained by using the normal vectors in the triangle set S as cone generatrices, but it has the same axis direction vector as the sough cone $g()$.

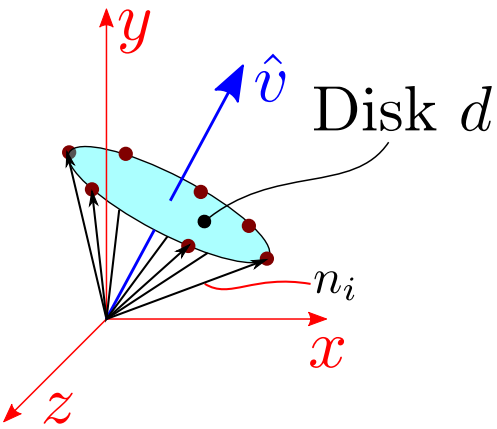


Figure 10. Cone fitting. Dual cone formed with Origin as apex. Unitary vectors normal to S are generatrices for the dual cone. Origin-located tails of n_i vectors normal to triangles t_i at Origin. Normal vector tips contained in a plane form a flat disk d. Minimal dispersion direction (principal component analysis) of disk d is the direction of cone axis $\hat{v}$.

Cone apex p_0

The straight line $[b_i, \hat{e}_i]$ (Figure 11) tangent to triangle t_i passes through the triangle baricenter b_i and has direction $\hat{e}_i$ (i.e., direction of the cone generatrix at b_i). The point at which the lines $[b_i, \hat{e}_i]$ are nearest to (or intersect) each other are the elements of C_p. The cone apex p_0 is the center of gravity of C_p.

Each set of vectors $[n_i, \hat{e}_i, \hat{v}]$ lies in a plane that intersects the cone and contains the apex p_0. We calculate a temporal vector $temp$ which is normal to such a plane as the cross product between n_i and $\hat{v}$. The cross product between the $temp$ vector and n_i guarantees that $\hat{e}_i$ is perpendicular to n_i and lies in the aforementioned plane. Equation (2) summarizes the process.

$$\begin{aligned} temp &= n_i \times \hat{v} \\ e_i &= temp \times n_i \end{aligned} \tag{2}$$

Opening Angle γ

The opening angle of the cone is computed as the average of the angles γ_i (Equation (3)) between the cone axis direction $\hat{v}$ and the calculated vectors $\hat{e}_i$.

$$\gamma = \frac{1}{m}\sum_i \gamma_i. \tag{3}$$

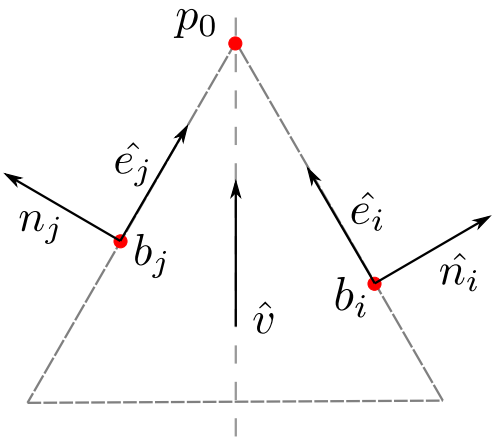


Figure 11. Cone 2D example. Vectors e_i point to p_v and are perpendicular to n_i.

3.2.3. Sphere Fitting

The geometrical parameters to define the sphere are:

1. p_0: sphere center.
2. R: radius.

Sphere center p_0

We consider a sphere which is faceted by a manifold set S of triangles t. The straight line $[b_1, n_i]$ normal to triangle t_i passes through the triangle baricenter b_i. This line has direction n_i. The points at which all lines $[b_i, n_i]$ are nearest to (or intersect) each other are the elements of set C_p (Figure 7) and probabilistically lie at the sphere center p_0 . The center of the sphere p_0 is estimated to be the center of gravity of C_p.

Sphere Radius R

The radius of the sphere is calculated by computing the mean distance from p_0 to all the vertices of the triangle subset.

$$R = \frac{1}{m}\sum_i^m ||p_i - p_0||. \tag{4}$$

3.2.4. Conclusions of Fitting Section

This section presents methods based on *progressive probing* rather than *statistical fitting* to estimate the $g()$ parameters of the analytic cones, cylinders, and spheres which has been faceted with manifold sets S of irregular, anisotropic, heterogeneous triangles. The extraction of the $g()$ support triangle subset $S \subset M$ follows.

3.3. Extraction of Subset S Based on Dihedral Angle

This method for extracting from M the triangle subset S ($\subset M$) which contains the seed triangle t_s, is applicable when the EDGE border between S and $M - S$ holds only C^0-continuity (i.e., ∂S is a sharp border). In this case, (a) S is indifferent to the $\mathscr{T}$ type of analytic form sought and (b) the extraction of S precedes and is independent from the $g()$ fitting process.

In this manuscript, we say that two EDGE-sharing triangles $t_i, t_j \in M$ hold only C^0-continuity if their dihedral angle is above a threshold α_{max}. The dihedral angle $\angle(t_i, t_j)$

between triangles t_i and t_j is the one between their normal vectors (under a uniform LOOP traversal direction). We measure the dihedral angle in the interval $[0, 180)$ degrees. Notice that $\angle(n_1, n_2) = 180$ implies that M would be non-manifold.

The first method for extraction of the triangle subset S is based on the dihedral angle between the triangles of M. This method is employed when the target segment of the triangulation holds strict C^0-continuity with respect to the other segments of M. This extraction process is indifferent to the type $\mathscr{T}$ of the analytic form and occurs separately from the fitting process as shown in Figure 2 (red).

Input:

1. $M(\mathcal{T}, \mathbf{P})$: triangle (two-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.
2. t_s: selected seed triangle from $\mathcal{T}$.
3. α_{max}: maximum angle value for dihedral angle-based extraction.
4. $\mathbf{E}_\alpha()$: dihedral angle equivalence relation defined on M.

Output:

1. S: the equivalence class (or block) containing t_s, of M under the equivalence relation $\mathbf{E}_\alpha()$.

Definition. Equivalence Relation $\mathbf{E}_\alpha()$

The triangles are equivalent under $\mathbf{E}_\alpha()$ if there is a triangle sequence (Figure 12) $L = [t_0, t_1, ..., t_f]$ such as:

1. $L \subset M$
2. Any two consecutive triangles of L, t_i and t_{i+1} share an EDGE.
3. $\angle(t_i, t_{i+1}) < \alpha_{max}$

The equivalence character of relation $\mathbf{E}_\alpha()$ is not proved here. However, it is easy to verify that $\mathbf{E}_\alpha()$ is symmetric, reflexive and transitive. The reason we show this small digression here is that the $g()$ support subset $S \subset M$ containing the seed triangle t_s is computable (when S and $M - S$ hold only C^0-continuity at ∂S) by using well known algorithms for transitive closure.

The boundary representation of input triangle set M supports FACE, VERTEX, EDGE neighborhood interrogations in constant time. Therefore, the dihedral angle-based extraction process of the $g()$ support subset S presents low computational cost and fast execution times.

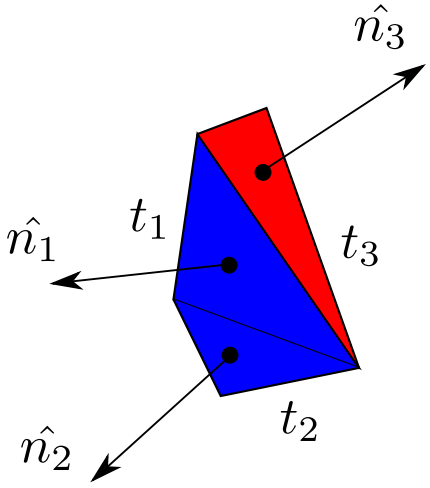


Figure 12. Dihedral angle equivalence $\mathbf{E}_\alpha()$. $\angle(n_1, n_2) \rightarrow 0$ and $\mathbf{E}_\alpha(t_1, t_2)$ (blue equivalence class). $\angle(n_1, n_3) > \alpha_{max}$ and t_3 (red) is outside the equivalence class (blue) of t_1 and t_2.

3.4. Extraction of Subset S Based on Fitting Quality

The dihedral angle-based extraction presents limitations when the input triangle set presents C^1 or higher continuity at border ∂S. The extraction of the $g()$ support subset

S depends on the quality of the $g()$ fitting and on the type $\mathscr{T}$ of the analytical form as follows.

Input:

1. $M(\mathcal{T}, \mathbf{P})$: Triangle (two-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.
2. t_s: selected seed triangle from $\mathcal{T}$.
3. $\mathscr{T}$: type of analytic form (selected by the user from the options cylinder, sphere, cone).

Output:

1. S: Largest connected subset of M, formed by triangles from M, containing t_s and fitting the analytical form of type $\mathscr{T}$.
2. $g()$: The parameters to define the analytic form that fits S.

Figure 13 presents a diagram with the general structure of the fitting-based extraction algorithm. The quality of $g()$ and the extent of the S support subset impact on each other at every iteration. The algorithm in Figure 13 assumes the existence of an initial set S, its border triangles $B(S)$ (in $M - S$) and an initial estimation for the analytic form $g()$. This basic S is formed by the seed triangle (marked by the user) and its neighbor triangles in M. The initial form $g()$ is estimated as per Section 3.2. $B(S)$ contains the triangles in $M - S$ that encircle S.

In Figure 13, the process/box "select candidate triangle from $B(S)$" selects a triangle in $M - S$ adjacent to the border triangles of current S to be screened for its belonging to the currently fitted analytic form $g()$. Figure 14 addresses this task. Its discussion appears after explaining the main algorithm of Figure 13.

In Figure 13, $B(S)$ denotes a set of triangles which encircle S, in $M - S$, and represent an opportunity to expand S (i.e., they do not differ from $g()$). The algorithm acts as long as $B(S)$ is not empty. A triangle t from the $B(S)$ border set is chosen (Figure 14, discussed later). If t strongly fits $g()$, it is accepted in S. If t does not fit $g()$ in a clear manner, the validity of S is examined. If S is a "mature" set ($w() \to 0$, Figure 15, discussed below), $g()$ it is supposed to be reliable and t is rejected. If the set S is "immature" ($w() \to 1$, t is re-assessed: if its distance to $g()$ is large, it is rejected. If it is near $g()$, it is accepted in S. If t is accepted in S, then it is removed from $B(S)$ and its neighbors in $M - S$ are included in $B(S)$. Additionally, $g()$ is re-computed against the new, augmented S.

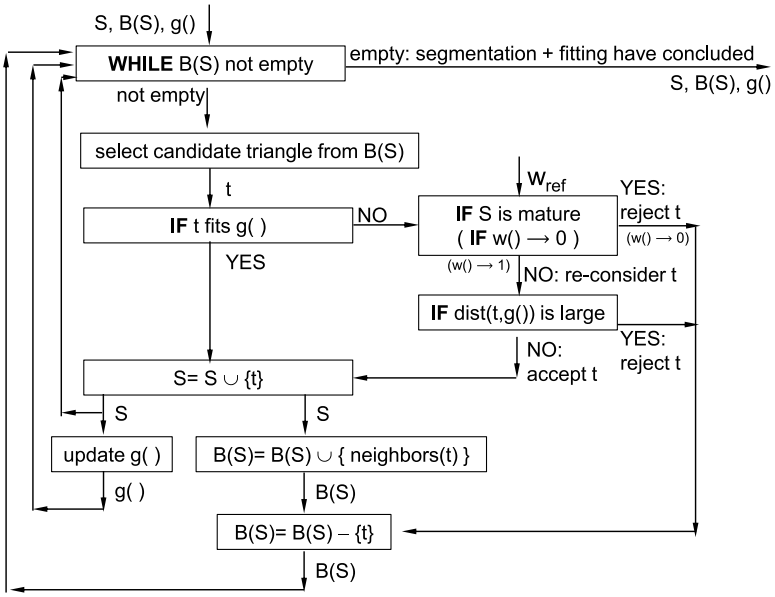


Figure 13. Algorithm for simultaneous fitting of $g()$ and extraction of $g()$ support subset S.

If a triangle t is rejected, it is removed from $B(S)$. In this case, obviously, its neighbors in $M - S$ will not be considered to expand S.

We use a $w() : R^+ \rightarrow [0,1]$ heuristic function (Figure 15) as a measure of the immaturity of S. Let $u = N_S/N_T$ be the ratio between the instant size of S and a threshold number of triangles N_T, which is considered the size of a "mature" S. The function $w(u)$ satisfies:

i $w(0) = 1$ (S is immature, has few triangles).
ii $w(u) = 0$ for all $u \geq 1.0$ (S is mature, has many triangles).
iii $w(u)$ is monotonically decreasing in $\mathbb{R}^+$

Figure 14 presents an expansion of the process/box "select candidate triangle from $B(S)$ " in Figure 13, as follows: (1) A triangle t of the triangle border of S ($B(S)$) is selected. (2) The neighbors of triangle t outside S are determined, forming set N_t. (3) From the set N_t, triangles are eliminated whose dihedral angle with t is large (i.e., form sharp edges with t). (4) From the remaining eligible neighbors of triangle t, N_t, the direction u_{max} of largest curvature is determined. (5) The triangle t_s, neighbor of t, is selected, which materializes this large curvature (i.e., direction u_{max} from t). This algorithm has an implicit bias of considering that expansion of the support triangle subset S in the direction of largest smooth curvature will speedily lead to a meaningful fitting of $g()$. Thus, the evolving of $g()$ would be most efficient. Eventually, all feasible triangles around t are included in the expansion and tested accordingly. It is important to emphasize that the segmentation/fitting algorithm (Figure 13), both support subset S and analytic primitive $g()$ results.

This codependent fitting-segmentation process probes and expands the $g()$ support subset $S \subset M$, authorized by the quality of the $g()$ fitting (Section 3.2). Due to this codependency, this process is slower than the dihedral angle-based identification of S.

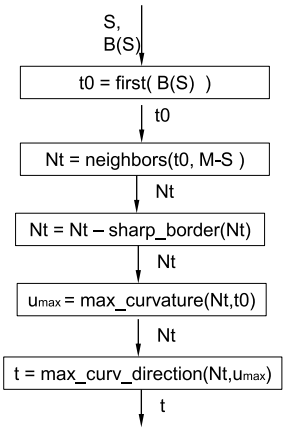


Figure 14. Expansion of box/process "select candidate triangle from $B(S)$" from Figure 13.

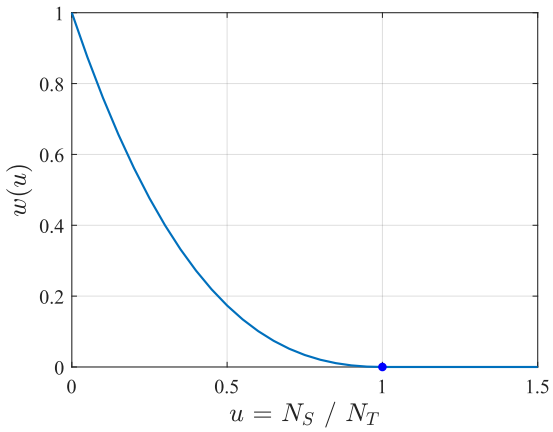


Figure 15. S immaturity function $w()$. $w \rightarrow 1$: S is immature set, with few triangles ($N_S \rightarrow 0$). $w \rightarrow 0$: S is mature set, with many triangles ($N_S > N_T$). N_T = threshold in number of triangles to consider $g()$ as stable analytic form.

3.4.1. Fitting-Based Extraction of Cylinders and Cones

Cones and cylinders are ruled surfaces. As a consequence (Figure 16), at each point of them, there is a direction u_{min} of $K_{min} = 0$ minimal curvature (i.e., generatrix direction). As always, the direction of K_{max} maximal curvature u_{max} is normal to the u_{min} and both span the plane tangent to the surface at such a point.

Our algorithm aims to fit a reliable $g()$ analytic form to the support triangle subset S as early as possible. Triangles in the direction of maximal curvature u_{max} (obviously) provide information on the cone or cylinder radii in addition to generatrices. Due to this reason, S primarily grows in the direction u_{max} (box "select candidate from $B(S)$" in Figures 13 and 14). However, it is important to notice that S must eventually encompass the whole set of triangles supporting $g()$.

The process to calculate the local curvature at a given seed triangle t_s (with normal vector n and baricenter b) is described below.

1 Build a homogeneous coordinate frame S_i:

$$S_i = \begin{bmatrix} e_1 & e_2 & \hat{n} & b \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

with e_1, e_2 spanning the plane tangent to surface M at b and making $[e_1, e_2, n]$ a Special Orthogonal SO(3) (3×3) submatrix.

2 Apply a coordinate transformation T to rigidly move t_s and support subset S to the World Coordinate System S_w with n and b mapped to direction z and point $[0, 0, 0]'$, respectively (Figure 16).

$$\begin{aligned} T\, S_i &= S_w, \\ T &= S_i\, S_w^{-1}. \end{aligned} \tag{5}$$

3 Fit a quadratic surface (Equation (6)) to $T * S$ by multi-variable linear regression.

$$f(x, y) \ = a + bx + cy + dxy + ex^2 + fy^2. \tag{6}$$

4 Find the minimum and maximum curvature directions v_{max}, v_{min} with the Hessian eigenvalues and eigenvectors.

$$\begin{aligned} H(f) \ &= \begin{bmatrix} 2e & d \\ d & 2f \end{bmatrix}, \\ H[v_{max}\, v_{min}] &= [v_{max}\, v_{min}] \begin{bmatrix} K_{max} & 0 \\ 0 & K_{min} \end{bmatrix}. \end{aligned} \tag{7}$$

5 Convert direction $v_{max}\ v_{min}$ back to global coordinates, with:

$$[u_{max}\, u_{min}] = T^{-1}[v_{max}\, v_{min}], \tag{8}$$

where, for the sake of simplified notation, we use the same symbols for 2D, 3D and 3D homogeneous vectors. Notice that $T^{-}1$ does not involve inverting a matrix. Instead, as T is an SO(3) rotation plus translation, the *transpose* of the rotation is its inverse.

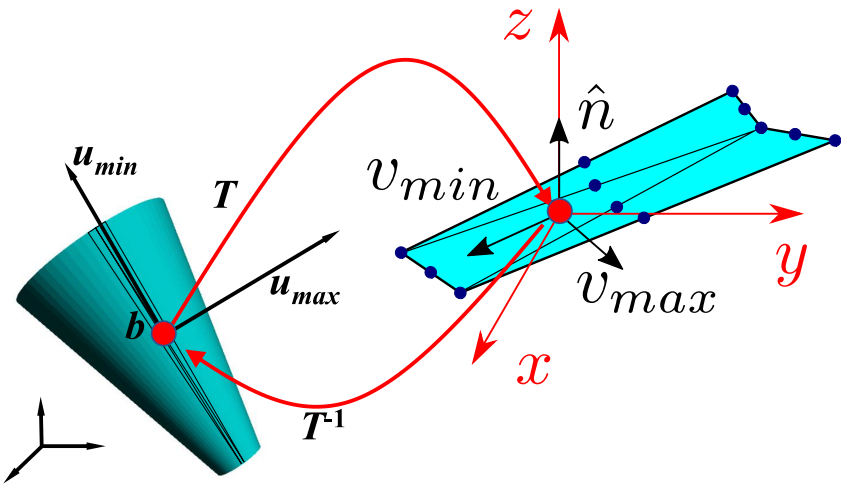


Figure 16. Rigid mapping M of local tangent plane of a ruled surface S onto the XY plane for the purpose of computing maximal and minimal curvature directions on S.

3.4.2. Fitting-Based Extraction of Spheres

This section corresponds to the cases in which the primitive $g()$ and triangle support subset S to extract from M correspond to a sphere making a smooth blend with $M - S$. The fitting and extraction algorithm appears in Figure 13 and is commented on in Section 3.4; at this point, we only add some particularities of the box "select candidate triangle from $B(S)$" from Figure 13.

This section uses the concept of a *topological circle* $C(t,k)$ and *topological ball* $B(t,k)$ defined on a triangular mesh M. $C(t,k)$ consists of the triangles of M that have distance k with respect to a "center" triangle t. A distance k between any triangles t_a and t_b of M corresponds to the minimal number of neighboring triangles in M to be traversed to travel from triangle t_a to triangle t_b, or vice versa. For this definition, two triangles are considered to be a distance of 1 apart if they share an EDGE or a VERTEX. The topological ball $B(t,k)$ includes all triangles of M whose topological distance to triangle t is less than or equal to k. Figure 17 presents a graphical example of the topological circle $C(t_s,2)$ and topological ball $B(t_s,2)$.

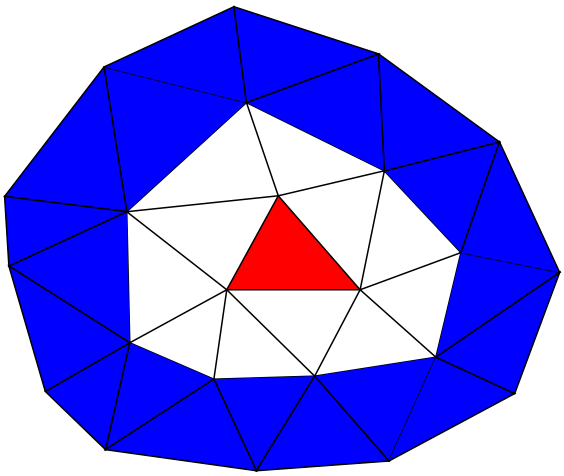


Figure 17. Red: user input seed triangle ($C(t_s,0)$). Blue: topological circle $C(t_s,2)$. White: topological circle $C(t,1)$. Red + White + Blue: topological ball $B(t_s,2)$. The initial guess for the analytic form $g()$ is computed with $C(ts,0) \cup C(ts,2)$.

4. Results

Sections 4.1–4.3 present support subset extraction and fitting for forms cylinder, cone and sphere, respectively. In each section, (1) dihedral angle-based and (2) fitting-based

support subset S extractions are presented, along with the $g()$ analytic forms fitted to S. Section 4.5 presents a complexity analysis comparison of our approach vs. competitor approaches. Section 4.6 discusses the real-time application of our implementation, achieved via spatial hashing and boundary representation preprocessing.

Figure 18 presents an execution of the fitting-extraction algorithm of Figures 13 and 14. The execution fits and extracts a cone S from a triangle set M in which the cone sector S keeps C^1-continuity with the remaining $M - S$ subset, and therefore the dihedral angle criteria are not able to identify S. Figure 18a displays the current S triangle set containing the seed triangle t_s (red) and already accepted neighboring triangles (blue). Figure 18b shows the grey boundary of S, $B(S)$. The bases of the cone contain triangles of $B(S)$ are not considered to be part of the cone due to their abrupt dihedral angle at the border of S, ∂S. The green triangle is detected in the direction of maximal curvature u_{max}. This green triangle fits into the current $g()$ estimation. Figure 18c shows that the green triangle is added to the support subset S, extracted from the boundary ($B(S)$) and its neighbors (grey) are added to $B(S)$, ready for the next iteration. The final result of the S extraction and $g()$ estimation is shown in Section 4.2.

4.1. Cylinder Fitting

Cylinder. Dihedral Angle-based Support Subset *S* Extraction

Figure 19 presents results for $g()$ cylinder fitting when the $g()$ support subset S has only C^0-continuity with the rest ($M - S$) of the input set M. S is extracted at sharp dihedral angle EDGEs. As $g()$ support subset S extraction is independent of $g()$, the extraction and fitting are real-time processes (taking less than 1 s).

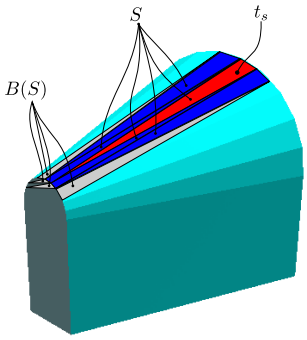
Cylinder. Fitting-based Support Subset S Extraction

Figure 20 displays cases in which the support subset S for $g()$ holds C^1-continuity or higher with $M - S$. In these cases, $g()$ fitting and $g()$ support subset S extraction take place simultaneously.

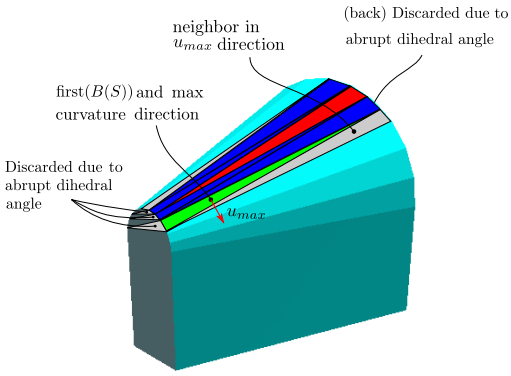
Table 6 presents the numerical results for the test on cylinder extraction, comparing the preset parameters against the parameters obtained by the fitting. Subindex fit indicates the variable obtained via fitting, as opposed to the experimental preset value.

An interesting result is that for the more difficult cases (fitting-based extraction), the estimated $g()$ is more accurate than the one achieved in the simpler cases (C^0-continuity between S and $M - S$). This result shows that the algorithm discussed in Figures 13 and 14, albeit more complex, renders a better accuracy.

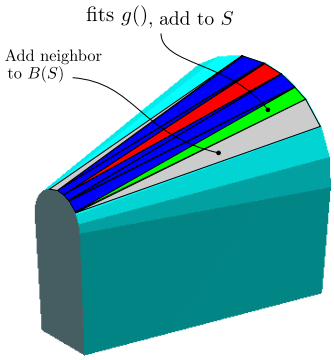
Dihedral angle-based extraction cases (1–4) perform in real time, while fitting-based extraction cases (5–6) do not. The former are faster not only because extraction and fitting are independent from each other, but also because $g()$ is fitted to small triangle subsets of S. These triangle subsets are scattered to make reasonable even though fast estimations.



(a) Initial support set S and $B(S)$.

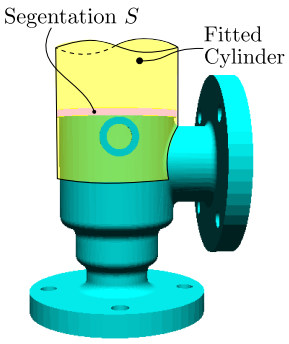


(b) S expansion in u_{max} direction.

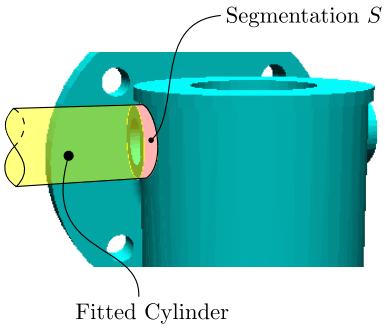


(c) Inclusion of accepted triangle in S and its neighbor in $B(S)$.

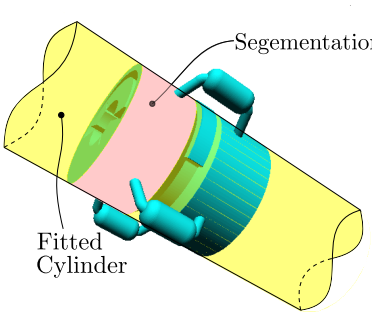
Figure 18. Step-wise execution of the S extraction $g()$ fitting algorithm, applied to a cone which does not accept S extraction based on dihedral angles.



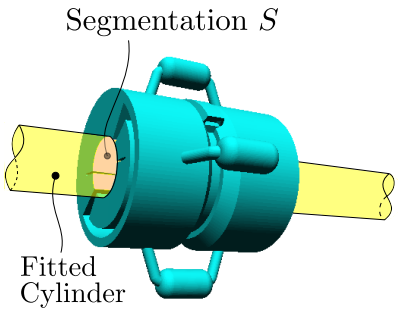
(a) Test 1



(b) Test 2

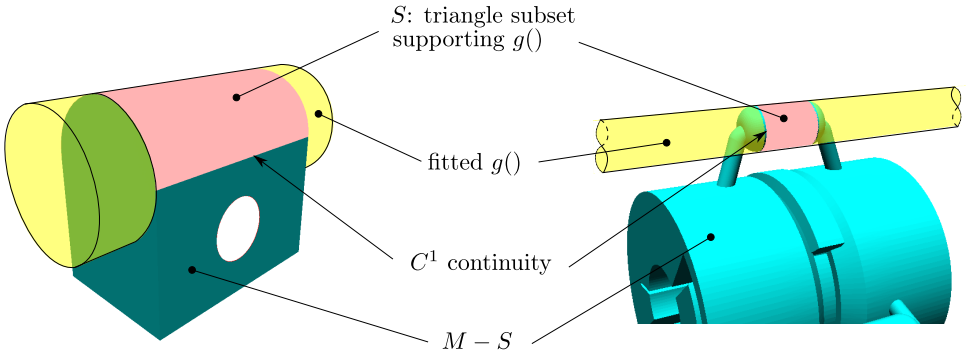


(c) Test 3



(d) Test 4

Figure 19. Cylinder fit. S extraction based on dihedral angles. Pink: extracted S support subset. Yellow: fitted $g()$.



(a) Test 5 (b) Test 6

Figure 20. Cylinder fit. Fitting-based S extraction. Pink: extracted S support subset. Yellow: fitted $g()$.

Cylinder. Fitting performance

Table 6. Cylinder fit. Set vs. found parameters. Tests 1–4: S extraction based on dihedral angles. Tests 5–6: Fitting-based S extraction.

	R	R_{fit}	$\hat{v}$	$\hat{v}_{fit}$	$\angle(\hat{v}, \hat{v}_{fit})$ **Degrees**
Test #1	47	46.9999	$\begin{bmatrix} -0.6104 \\ 0.6169 \\ 0.4967 \end{bmatrix}$	$\begin{bmatrix} -0.6104 \\ 0.6169 \\ 0.4968 \end{bmatrix}$	0°
Test #2	14	13.9972	$\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.9999 \\ 0.0008 \\ 0.0003 \end{bmatrix}$	0.0497°
Test #3	13	13.00004	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.00003 \\ -0.0002 \\ 0.9999 \end{bmatrix}$	0.0136°
Test #4	5	5.0037	$\begin{bmatrix} 0.4968 \\ -0.1904 \\ 0.8467 \end{bmatrix}$	$\begin{bmatrix} 0.4967 \\ -0.1903 \\ 0.8468 \end{bmatrix}$	0.0051°
Test #5	20	20.0000	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	0°
Test #6	2.5	2.4999	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	0°

4.2. Cone Fitting

Figures 21a,b and 22a,b illustrate the results for cone fitting with dihedral angle-based and fitting-based extraction, respectively. Table 7 presents numerical results comparing the actual vs. fitted $g()$ parameters.

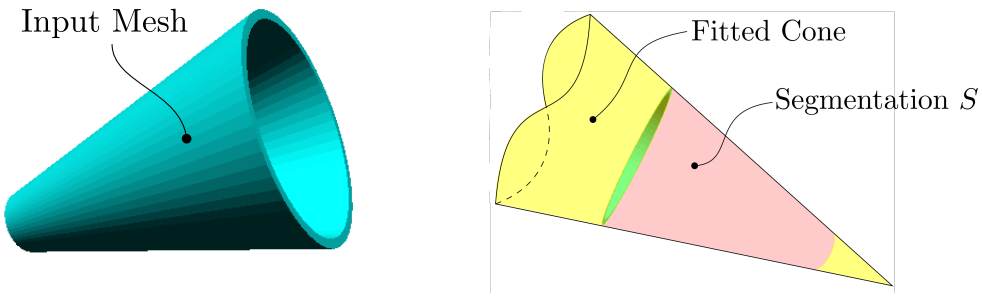


Figure 21. Test 1. Cone fitting test using dihedral angle-based extraction. Pink: segmentation. Yellow: cone fitting.

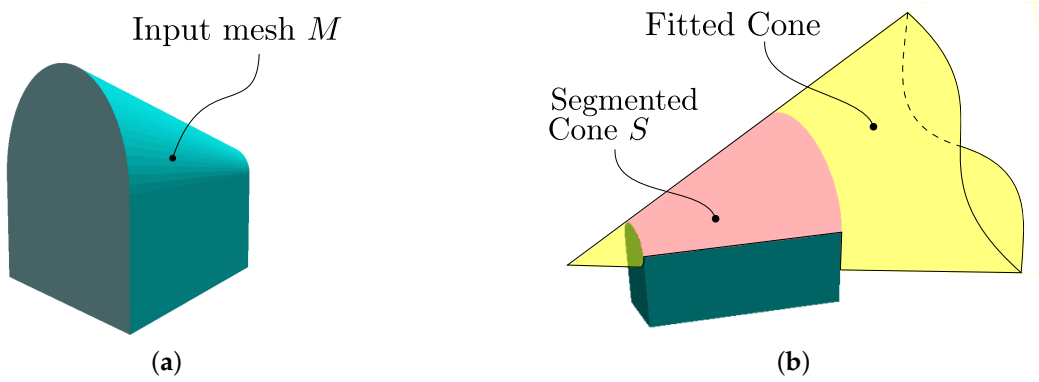


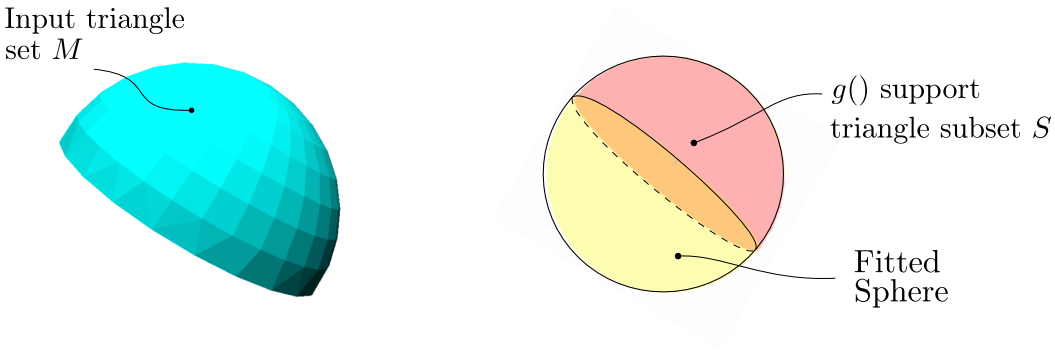
Figure 22. Cone Fit. Fitting-based S extraction. Pink: extracted S support subset. Yellow: fitted $g()$.

Table 7. Cone fit. Set vs. found parameters. Test 1: S extraction based on dihedral angles. Test 2: fitting-based S extraction.

	A_p	$A_{p\,fit}$	γ	γ_{fit}	$\hat{v}$	$\hat{v}_{fit}$	$\angle(\hat{v}, \hat{v}_{fit})$ **Degrees**
Test #1	$\begin{bmatrix} 0 \\ 0 \\ 144 \end{bmatrix}$	$\begin{bmatrix} 0.0010 \\ 0.0063 \\ 144.1274 \end{bmatrix}$	13.87	13.8852	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0.0000 \\ 0.0000 \\ 0.9999 \end{bmatrix}$	0.0033°
Test #2	$\begin{bmatrix} 0 \\ 0 \\ 72.27 \end{bmatrix}$	$\begin{bmatrix} -0.1258 \\ -3.0241 \\ 72.2727 \end{bmatrix}$	19	18.9716	$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -4.57\times10^{-11} \\ -5.96\times10^{-9} \\ 1 \end{bmatrix}$	0.0033°

4.3. Sphere Fitting

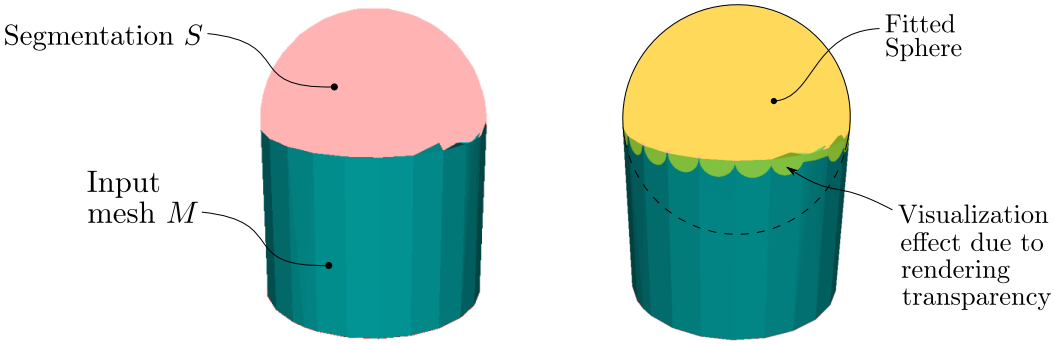
For the sphere fitting, the dihedral angle-based extraction does not produce a segmentation of the input mesh M. Therefore, $S = M$ and the fitting occurs over M. Figure 23 presents the results for that particular case. However, the parameters obtained by the fitting are accurate (see Table 8).



(**a**) Sphere sample $r = 35$ (**b**) Sphere fit $r_fit = 2.99$

Figure 23. Sphere fitting using dihedral angle-based extraction. Pink: segmented sphere. Yellow: fitted sphere. Notice that $S = M$.

Figure 24 illustrates the results for a fitting-based extraction in a sphere coupled with a cylinder and presenting C^1-continuity at the cylinder–sphere border.



(**a**) Input mesh. Sphere (**b**) Segmentation and sphere detection.

Figure 24. Sphere fit. Fitting-based S extraction. Pink: extracted S support subset. Yellow: fitted $g()$. The process leaves out of S few sphere triangles at the C^1 border with $M - S$. However, it succeeds in rejecting inclusion in S of triangles actually belonging to the cylinder.

Table 8. SPHERE Fit. Set vs. found parameters. Test 1: S extraction based on dihedral angles. Test 2: fitting-based S extraction.

	R	R_{fit}	p_0	p_{0fit}
Test #1	3	2.9752	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}-0.0118\\-0.0074\\-0.0399\end{bmatrix}$
Test #2	25	25.0216	$\begin{bmatrix}0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0.0357\\0.1242\\-0.0333\end{bmatrix}$

4.4. Comparison with Competitor Approaches

The datasets, or code, used by competitor approaches are not publicly available. In searching benchmark datasets for analytic form fitting, we found that they consist of point clouds. In contrast, the input for our algorithm should be (poor quality) triangle meshes. In consequence, we resorted to (a) executing CAD models or (b) downloading triangular meshes presenting object similarity and the same challenges as the ones in those references. The results of the comparison are presented in Table 9.

Table 9. Test with reference datasets of competitor approaches.

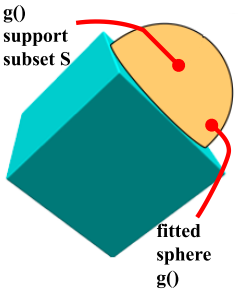
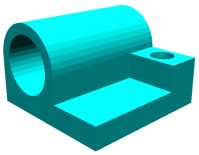
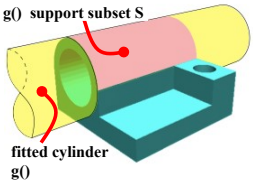
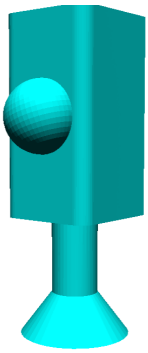
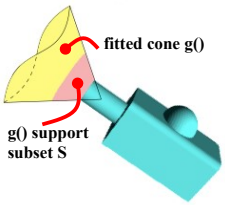
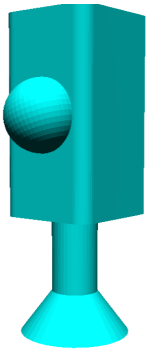
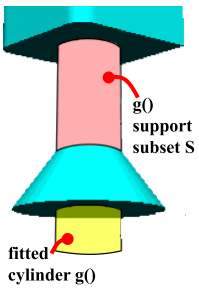
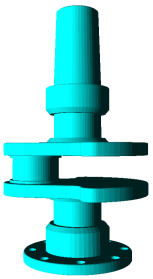
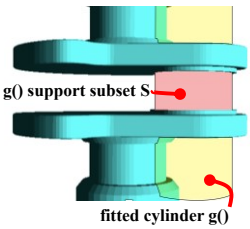
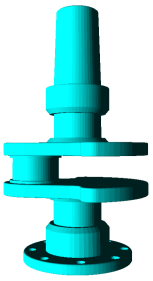
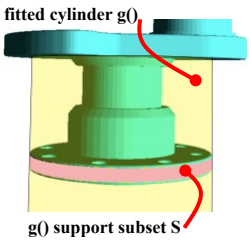
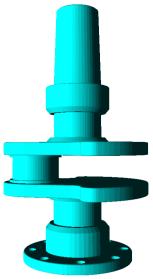
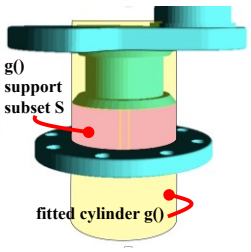
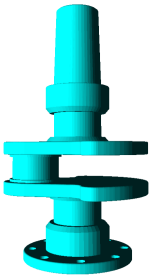
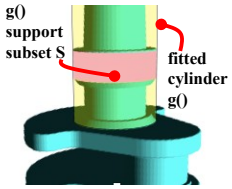
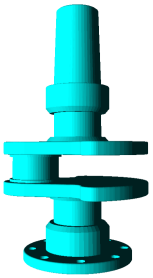
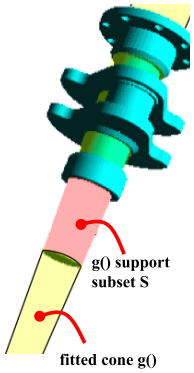
Test	Reference + Dataset	Input Triangular Set M	Primitive + Segmentation	Our Result
1	Ref. [9]. Dataset: Sphere-in-Cube 1. Ref. [9] fails to include all triangles in S 2. Ref. [9] does not inform execution times. 3. Requires dense datasets. 4. Our execution is correct, both in S and $g()$.		Sphere + Dihedral Segmentation	
2	Ref. [17]. Dataset: C^1-cylinder 1. Ref. [17] requires dense datasets. 2. Ref. [17] does not give execution times. 3. Our execution is correct, both in S and $g()$.		Cylinder + Fitting Segmentation	
3	Ref. [16]. Dataset: Lamp Post 1. Ref. [16] uses triangle aspect ratios ≈ 1. 2. Ref. [16] does not inform exc. Times. 3. Our execution is correct, both in S and $g()$.		Cone + Dihedral Segmentation	
4			Cylinder + Dihedral Segmentation	
5	Ref. [7]. Dataset: Crankshaft 1. Ref. [7] uses dense, good quality triangle set ≈ 1. 2. Ref. [7] gives global execution times. 3. Our execution is correct, both in S and $g()$.		Cylinder + Dihedral Segmentation	

Table 9. *Cont.*

Test	Reference + Dataset	Input Triangular Set M	Primitive + Segmentation	Our Result
6			Cylinder + Fitting Segmentation	
7			Cylinder + Dihedral Segmentation	
8			Cylinder + Fitting Segmentation	
9	1. Ref. [7] does not cover cone extraction. 2. Our execution is correct, both in S and $g()$.		Cone + Dihedral Segmentation	

Our algorithm does not fail in any one of the datasets presented by competitor approaches. In row 9, our algorithm is able to fit a correct $g()$ conical analytic form, while Ref. [7] does not attempt this fit. These competitor approaches ([7,9,16,17]) either do not publish execution times at all or publish lumped times spent for fitting the full set of primitives sought in the input triangle set M.

4.5. Complexity

Table 10 exhibits the complexities for distinguishable existing approaches in the literature. The complexity is calculated based on the standard costs of the diverse subprocesses required for each method.

Table 10. Complexities of existing approaches and of our algorithm (n is the number of points or triangles of the input set). * Obtaining initial guess of $g()$. ** Applying optimization to $g()$. † Segmentation and fitting are codependent.

Approach	Preprocessing	B-Rep	Extraction of S	Fitting	Extraction + Fitting †	Overall
[7] Yan et al.	$O(n^2)$	-	$O(n^2)$	$O(n^2)$	$O(n^4)$	$O(n^4)$
[6] Attene et al.	$O(n^2)$	-	-	-	$O(n^3)$	$O(n^3)$
[3] Ruiz et al.	$O(n^3)$ *	-	-	$O(n^2)$ **	-	$O(n^3)$
[12] Li et al.	$O(n^2)$	-	$O(n)$	$O(n^3)$	$O(n^4)$	$O(n^4)$
[4] Hulik et al.	$O(n)$	-	-	$O(n^3)$	-	$O(n^3)$
Our approach	$O(n)$	$O(n^2)$	$O(n)$	$O(n^2)$	$O(n^3)$	$O(n^3)$

Notice that the methods (rows) analyzed in Table 10 have diverse goals. Therefore, their overall complexity is not comparable on an equal basis. The standard cost for extracting the support set S (i.e., computation of neighboring triangles) is $O(n^2)$. However, we achieve complexity $O(n)$ when profiting off the boundary representation for input triangle set M. This B-Rep construction holds a complexity $O(n^2)$, but the hashing preprocessing reduces the search set (n) to, at most, three voxels. Section 4.6 presents the time reduction due to the hashing execution.

4.6. Reduction in Computing Time by Using Hashing

The preprocessing present in our algorithm contains the sequence: 1. spatial hashing and 2. boundary representation. The B-Rep makes the neighborhood relations (among VERTEXs, EDGESs and FACEs) in the input triangle set M explicit. At the same time, it is a diagnosis of non-manifold conditions and guarantees uniform orientation of the input triangle set M.

Although the B-Rep is the visible database for interrogation of M, it is the spatial hashing that enables the efficient construction of B-Rep and subsequent extraction of support subset S and fitting of $g()$ to S. Due to this reason, this section focuses on the acceleration of B-Rep construction due to the spatial hashing.

The spatial hashing reduces the search set space for interrogations of VERTEX, EDGE and FACE neighborhoods. Figure 25 presents the reduction in execution time of the B-Rep construction for different input sizes before and after applying the spatial hashing. The figure shows a reduction factor of about 1000 in the execution time. It is also relevant to mention that the B-Rep + hashing cost represents a very faithful logarithmic reduction in the bare B-Rep expenses.

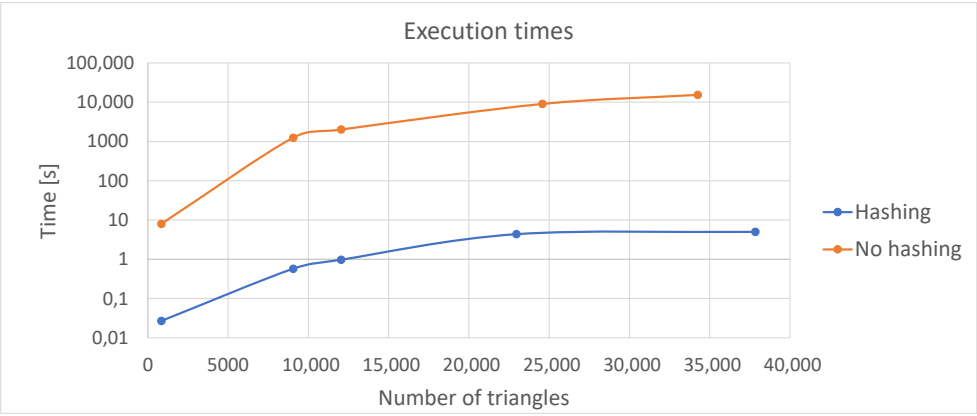


Figure 25. Execution times for B-Rep construction with and without previous spatial hashing.

5. Conclusions and Future Work

This manuscript presents the implementation of an algorithm to identify user-requested analytic forms (cone, cylinder, sphere) and their triangle support subsets S, in a connected manifold triangle set M.

The statistical quality of M is low as it is a heterogeneous and anisotropic, with a poor aspect ratio triangulation. This characteristic aligns with specific industrial applications in which low-polygon sets are purposely preferred at the expense of mesh quality.

Our algorithm favors a one-by-one progressive estimation of the parameters of the $g()$ form, based on local probing. Our approach does not favor the application of generic statistical multi-parameter fittings, which renounce the exploitation of $g()$-specific properties and require a dense, good quality input M.

Three different types of primitives (cylinder, cone, and sphere) were fitted. Two methods were implemented for the extraction of $g()$ support subset S, depending on the C-continuity level at the border ∂S with $M - S$. The results are consistent and accurate in obtaining $g()$ for statistically poor triangulations. The precision in the parameters of each analytical form was sufficient for our industrial application, presenting (for example) a maximal error of ≈ 0.1 % in the opening angle of the cone.

As expected, fitting-based extraction presents higher execution times than a dihedral angle-based one. The iterative process of calculating the primitive parameters introduces an additional cost to the identification of the $g()$ support subset S. In contrast, real-time execution was achieved for all cases of dihedral angle-based S extraction. Both methods are accelerated by the connectivity information provided by the B-Rep.

The spatial hashing is central in accelerating the B-Rep construction, and thus the S support subset extraction and $g()$ primitive fitting. Our approach takes advantage of the B-Rep benefit of constant-time access to VERTEX, EDGE and FACE neighborhoods. Spatial hashing reduces time complexity but adds storage complexity. Therefore, it is not a theoretical advantage. In practice, however, spatial hashing dramatically speeds up B-Rep construction because it delivers bounded neighborhood search times as it (B-Rep construction) acts in a constant search space. Future work is required on the acceleration of fitting-based extraction. A natural upgrade of our method is envisioned for cases in which the primitive sought type $\mathscr{T}$ is not specified but instead is one out of a given finite set.

Author Contributions: Conceptualization, J.C., O.R.-S., D.A.A. and C.R.-C.; investigation, J.C., O.R.-S., D.A.A. and C.R.-C.; methodology, J.C., O.R.-S., D.A.A. and C.R.-C.; software, J.C. and C.R.-C.; resources J.C., O.R.-S. and C.R.-C.; supervision, J.C. and O.R.-S.; visualization, J.C., C.R.-C and O.R.-S.; writing—original draft, J.C., C.R.-C., O.R.-S. and D.A.A; writing—review and editing, J.C., C.R.-C. and O.R.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received funding from Manufactura Cohesiva SAS, Universidad EAFIT and private funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Term	Description
$M(\mathcal{T}, \mathbf{P})$	Triangle (2-manifold) set with geometry $\mathbf{P}$ and topology $\mathcal{T}$.
$\mathcal{T} = \{t_1, t_2, ...\}$	Set of triangles of the input mesh M.
$\mathbf{P} = \{t_1, t_2, ...\}$	Set of vertices of the input mesh M.
t_s	Seed triangle.
$\mathscr{T}$	User-specified type of the analytic form {Cylinder, Sphere, Cone}.
$g()$	Analytic form denoting one of the cone, cylinder or spheric primitives. $g()$ equally expresses parametric or implicit forms, as well as the set of parameters determining a primitive.
E_t	User-specified method for the analytic form fitting.

S	Support set of triangles that fit a particular analytic form.
$C(t,k)$	Topological circle on a triangular mesh M, with center triangle t and distance k.
p_0	Distinguished point of cylinder, sphere or cone definition.
R	Cylinder or sphere radius.
$\hat{v}$	Axis vector for cylinder or cone.
γ	Opening angle for cone.
$\delta_x, \delta_y, \delta_z$	Voxel side length in the x, y, z directions, respectively.
Ω	Rectangular prism orthogonally oriented w.r.t. the coordinate axes, containing a $N_v \times N_v \times N_v$ grid of rectangular voxels v_{ijk}.
N_v	Number of voxels per side of Ω
h	Hashing function: $h : P \rightarrow N_v \times N_v \times N_v$, which maps each vertex of the input triangle set M into the (i,j,k) indices of the voxel $v_{ijk} \subset \Omega$, which contains such a vertex.
H	$\Omega \rightarrow \mathcal{T}$, which maps each voxel v_ijk of Omega into the set of triangles of M which contain a vertex inside voxel v_ijk.
C_p	Set of auxiliary points to calculate the axis of the analytic forms (center for sphere case).
$\hat{n}_i, n$	Vector normal to triangle i.
b_i, b	Baricenter of triangle i.
K_{min}, K_{max}	Maximal and minimal surface curvatures.
u_{min}, u_{max}	Directions of maximum and minimal curvatures.
T	Homogeneous rigid transformation mapping the point b of mesh M onto the $[0,0,0]'$ of $\mathbb{R}^3$, and the plane tangent to M at p onto the plane XY in $\mathbb{R}^3$.
$L_n = \{\ell_1, \ell_2, ...\}$	Set of lines generated by the baricenter b_i and normal $\hat{n}_i$.
N_S	Number of triangles in triangle subset S.
N_T	Threshold in number of triangles needed to obtain a stable approximation of an analytic form.
$w(\lvert S \rvert)$	Maturity measure of the set S. $1 \rightarrow 1$ if $\lvert S \rvert$ is small (S immature) $w \rightarrow 0$ if $\lvert S \rvert$ is large (S mature).
u	Relation between the number of triangles in subset N_S and the threshold N_T.

References

1. Kaiser, A.; Ybanez Zepeda, J.A.; Boubekeur, T. A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data. *Comput. Graph. Forum* **2019**, *38*, 167–196. [CrossRef]
2. Busé, L.; Galligo, A.; Zhang, J. Extraction of cylinders and cones from minimal point sets. *Graph. Model.* **2016**, *86*, 1–12. [CrossRef]
3. Ruiz, O.; Arroyave, S.; Acosta, D. Fitting of Analytic Surfaces to Noisy Point Clouds. *Am. J. Comput. Math.* **2013**, *3*, 18–26. [CrossRef]
4. Hulik, R.; Spanel, M.; Smrz, P.; Materna, Z. Continuous plane detection in point-cloud data based on 3D Hough Transform. *J. Vis. Commun. Image Represent.* **2014**, *25*, 86–97. [CrossRef]
5. Rabbani, T.; Heuvel, F.V.D. Efficient Hough Transform for Automatic Detection of Cylinders in Point Clouds. *ISPRS Workshop Laser Scanning* **2005**, *3*, 60–65.
6. Attene, M.; Patanè, G. Hierarchical structure recovery of point-sampled surfaces. *Comput. Graph. Forum* **2010**, *29*, 1905–1920. [CrossRef]
7. Yan, D.M.; Wang, W.; Liu, Y.; Yang, Z. Variational mesh segmentation via quadric surface fitting. *CAD Comput. Aided Des.* **2012**, *44*, 1072–1082. [CrossRef]
8. Shao, T.; Xu, W.; Zhou, K.; Wang, J.; Li, D.; Guo, B. An interactive approach to semantic modeling of indoor scenes with an RGBD Camera. *ACM Trans. Graph.* **2012**, *31*, 1–12. [CrossRef]
9. Gelfand, N.; Guibas, L.J. Shape segmentation using local slippage analysis. In Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, Nice, France, 8–10 July 2004; Volume 71, pp. 214–223. [CrossRef]
10. Bagautdinov, T.; Fleuret, F.; Fua, P. Probability occupancy maps for occluded depth images. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 2829–2837. [CrossRef]
11. Liu, J. An adaptive process of reverse engineering from point clouds to CAD models. *Int. J. Comput. Integr. Manuf.* **2020**, *33*, 840–858. [CrossRef]

12. Li, Y.; Wu, X.; Chrysathou, Y.; Sharf, A.; Cohen-Or, D.; Mitra, N.J. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.* **2011**, *30*. [CrossRef]
13. Woodford, O.J.; Pham, M.T.; Maki, A.; Perbet, F.; Stenger, B. Demisting the hough transform for 3D shape recognition and registration. *Int. J. Comput. Vis.* **2014**, *106*, 332–341. [CrossRef]
14. Chen, J.; Chen, B. Architectural modeling from sparsely scanned range data. *Int. J. Comput. Vis.* **2008**, *78*, 223–236. [CrossRef]
15. Lafarge, F.; Mallet, C. Creating large-scale city models from 3D-point clouds: A robust approach with hybrid representation. *Int. J. Comput. Vis.* **2012**, *99*, 69–85. [CrossRef]
16. Bénière, R.; Subsol, G.; Gesquière, G.; Le Breton, F.; Puech, W. Recovering primitives in 3D CAD meshes. *Three-Dimens. Imaging Interact. Meas.* **2011**, *7864*, 78640R. [CrossRef]
17. Le, T.; Duan, Y. A primitive-based 3D segmentation algorithm for mechanical CAD models. *Comput. Aided Geom. Des.* **2017**, *52–53*, 231–246. [CrossRef]
18. Lefebvre, S.; Hoppe, H. Perfect spatial hashing. *ACM Trans. Graph. (TOG)* **2006**, *1*, 579–588. [CrossRef]
19. Mejia-parra, D.; Lalinde-pulido, J.; Sánchez, J.R.; Ruiz-salguero, O.; Posada, J.; Cae, C.A.M.; Eafit, U. Perfect Spatial Hashing for Point-cloud-to-mesh Registration. In Proceedings of the Spanish Conference on Computer Graphics (CEIG 2019), San Sebastián, Spain, 26–28 June 2019; pp. 1–9. [CrossRef]
20. Mantyla, M. *An Introduction to Solid Modeling*; Computer Science Press: Rockville, MD, USA, 1988