Gabriel-constrained Parametric Surface Triangulation

Oscar E. Ruiz[1], John E. Congote[1], Carlos Cadavid[1], Juan G. Lalinde[1], Guillermo Peris-Fajarnés[2], Beatriz Defez-García[2] and Ricardo Serrano[1]
[1]*EAFIT University, Medellín, Colombia.*
[2]*Universidad Politécnica de Valencia, Spain*

1. Introduction

Glossary

S:	Parametric Surface. $S: \mathbb{R}^2 \rightarrow \mathbb{R}^3$. is an (infinite) 2-manifold without border.
F, H:	Faces. Connected subsets of a parametric surface $(F, H \subset S)$.
$S^{-1}(F)$:	Pre-image of F in parametric space $U-V$.
T_F:	Triangulation of face F in Euclidean space.
T_{UV}:	A triangulation in parametric space $U-V$.
$T = S(T_{UV})$:	Triangulation in $\mathbb{R}^3$ as a mapping, via S, of the triangulation T_{UV} in $U-V$ parametric space.
∂X:	Boundary of the set X.
L_i:	A loop ($L_i \subseteq \partial F$), is a 1-manifold without border. It is a connected subset of the boundary of F.
E_j	An edge ($E_j \subseteq L_i$), is a 1-manifold with border.
t:	A triangle of the triangulation T.
p,q:	Points in Euclidean space. $p, q \in \mathbb{R}^3$.
u, v, w:	Real parameters of a curve C(w) or a surface $S(u, v)$.
$cl(A)$:	Closure of the set A. cl(A) $= A \cup \partial A$.
$int(A)$:	Interior of the set A. int(A) $= A - \partial A$.
$B_G(p, q, r)$:	Gabriel Ball in $\mathbb{R}^3$. Spherical point set whose center is contained in the plane pqr, passing through the points $p, q, r \in \mathbb{R}^3$.
$B_G(p, q)$:	Gabriel Ball in $\mathbb{R}^3$. Spherical point set whose center is contained in the edge pq, passing through the points $p, q \in \mathbb{R}^3$.
e:	Edge of a triangle.

Boundary Representations, B-Reps, are the computer formalization of the boundary of a body ($M = \partial BODY$). Shortly, M is a collection of SHELLs, which in turn are collections of FACEs. For convenience, we will assume that the SHELLs are 2- manifolds without border in $\mathbb{R}^3$. Each SHELL is decomposed into FACEs, which must have boundary. It is customary in geometric modeling to make a FACE F a connected proper subset of *one* parametric surface $S(u,v) \subset \mathbb{R}^3$. In this article we consider the b-reps as closed 2-manifolds with continuity C^2 inside each face and C^0 among them.

The border of F is ∂F, which is the collection of LOOPs L_i embedded in S. The LOOP L_i can be thought of as a 1-manifold without border, with C^∞ continuity except in a finite number of points, where it is C^0-continuous. In such locations L_i is split into EDGEs E_j, each one being a C^∞ 1-manifold with border. The problem of surface triangulation takes place in one of such FACEs F. A PL approximation T_F of face F is required which: (a) is formed by triangles, (b) departs from F in less than a distance ε, (c) has triangles as equilateral as possible, (d) has as few triangles as possible, and, (e) each edge e_j of the triangle set has exactly two incident triangles. Property (e) is a consequence of the fact that a B-Rep is a 2-manifold without boundary. The triangulation T is also a 2-manifold (of the C^0 class) without boundary. Condition (e) also holds for edges e_j whose extremes lie on any loop L_i. This means, this edge e_i receives a triangle from the triangulation T_F (face F) and another from the triangulation T_H (face H).

An important aspect to control in triangulating a face F is that having a triangulation T_{UV} correctly covering $S^{-1}(F)$ in parametric space $U-V$ is not a guarantee for the triangulation $T = S(T_{UV})$ in $\mathbb{R}^3$ to be correct. Several problems may arise: (i) Fig. 1 illustrates that a completely internal triangle $[a,b,c]$ in parametric space $U-V$ may not be mapped by S to an internal triangle $[S(a),S(b),S(c)]$ in $\mathbb{R}^3$. (ii) roughly equilateral triangles t in $U-V$ space may map to extremely deformed triangles $S(t)$ in $\mathbb{R}^3$ because of sharp warping caused by S, (iii) neighboring triangles $t_i, t_j, t_k, \ldots$ in $U-V$ space mapped via $S()$ may form a fish scale effect in R^3 because of the same warping in S.

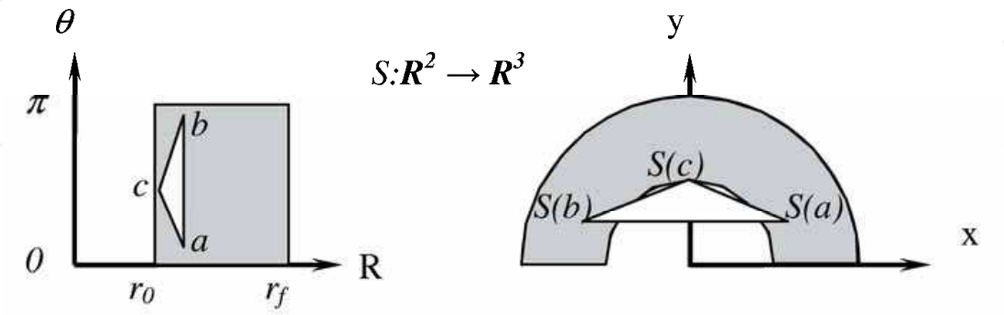


Fig. 1. Triangle *abc* is internal in parameter space. Triangle $S(a)S(b)S(c)$ is external to the surface $S(r,\theta) = (r\cos(\theta), r\sin(\theta), 0)$

2. Related Work

2.1 Fundamental definitions

As discussed in [1] a smooth 2-manifold with boundary (face) F is a sub-manifold of a smooth 2-manifold S without boundary. If the neighborhood of a point $p \in F$ is homeomorphic to a 2 dimensional euclidean space, then we say that the p is in the interior of F ($int(F)$). If the neighborhood of a point p in F is homeomorphic to a half euclidean space then we say that the point is in the boundary of F (∂F). The exterior of the submanifold F is composed by the points $p \in S$ and not in the closure of F ($p \notin cl(F)$). It includes all the points neither in the interior nor the boundary of F but still in S. The boundary is a closed set and the interior and exterior are open sets. In Fig. 4 the interior, boundary and exterior are shown ($A - B$ denotes the difference between sets A and B).

Fig. 2 displays the general situation in which a face F is carried by a parametric surface S in $\mathbb{R}^3$. F is a connected subset of S, with the boundary of F, $\partial F = \{L_0,..,L_n\}$ being the set of loops $\mathbf{L_i}$ which limit F on S. If the function $S(u, v)$ is 1-1 (which can be guaranteed by a convenient decomposition of the overall B-Rep) then there exists

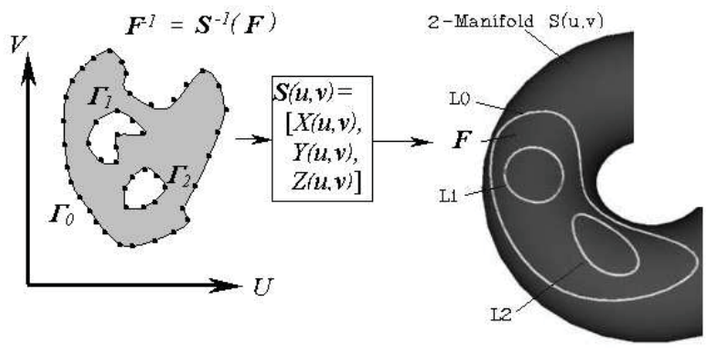


Fig. 2. Pre-image $F^{-1} = S^{-1}(F)$ of the face F by the parametric surface S.

a pre-image of F in parametric space $U \times V$, that we call F^{-1}. Such a region can be calculated as $F^{-1} = S^{-1}(F)$. To do so, a point sample of ∂F formed by points $p_i \in \mathbb{R}^3$ is tracked back to their pre-images $(u_i, v_i) \in (U \times V)$ therefore rendering a connected region $F^{-1} \subset (U \times V)$, most likely with holes, bounded by a set of planar Jordan curves $\partial F^{-1} = \{\Gamma_0,..,\Gamma_n\}$.

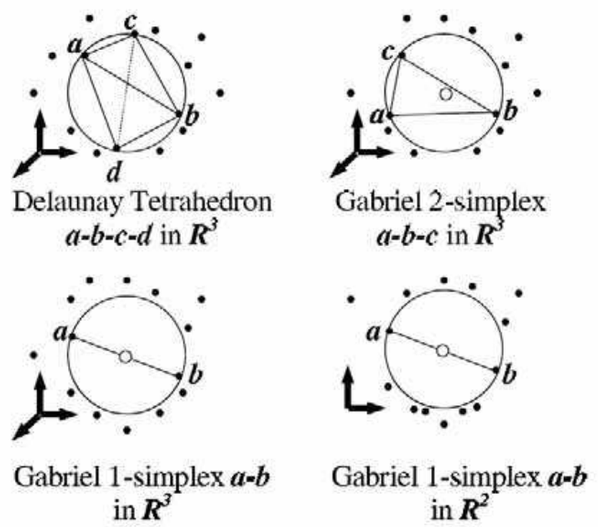


Fig. 3. Delaunay tetrahedron for points $a, b, c, d \in \mathbb{R}^3$, Gabriel 2-simplex for $a, b, c \in \mathbb{R}^3$, Gabriel 1-simplex for $a, b \in \mathbb{R}^3$, Gabriel 1-simplex for $a, b \in \mathbb{R}^2$.

Fig. 3 displays a short collection of Delaunay and Gabriel complexes. A Delaunay tetrahedron in a set of points in 3D is a tetrahedron (3-simplex) formed by four points whose circumscribed sphere contains no other point of the set. Given vertices $v_i v_j v_k$ in the point set, they form a Gabriel triangle (2-simplex) if the smallest sphere through them contains no other point of the set. The triangle $v_i v_j v_k$ is embedded in the equatorial plane of such a sphere. A Gabriel edge $v_i v_j$ (1-simplex) is one with v_i and v_j in the point set, such that the sphere centered in $(v_i + v_j)/2$ with radius $r = \mathrm{d}(v_i, v_j)/2$ contains no point of the sample other than v_i and v_j. Such a sphere is the smallest one containing v_i and v_j. Each Gabriel 1-simplex makes part of at least one Gabriel 2-simplex, and each Gabriel 2-simplex makes part of at least one Delaunay tetrahedra.

The present article applies the Gabriel variant (1- and 2- simplices) to Delaunay connectivity to calculate a triangulation for a point sample V_F (sensitive to curvature and independent of the parameterization) on the face F, carried by a parametric surface S. Section 2 reviews theoretical and algorithmic knowledge related to triangulations and surface curvatures. Section 3 discusses the algorithms devised and implemented to triangulate Boundary Representations. Section 4 presents five complex Boundary Representations with manufacturing and organic surfaces and high genii triangulated by the implemented algorithm. Section 5 concludes this article and sketches directions for future work.

2.2 Curvature Measurement in Parametric Surfaces

A parametric surface is a function $S : \mathbb{R}^2 \rightarrow \mathbb{R}^3$, which we assume to be twice derivable in every point. The derivatives are named in the following manner ([10], [20],):

$$S_u = \frac{\partial S}{\partial_u};\ S_v = \frac{\partial S}{\partial v};\ S_{uu} = \frac{\partial^2 S}{\partial u^2};\ S_{vv} = \frac{\partial^2 S}{\partial v^2};$$

$$S_{uv} = S_{vu} = \frac{\partial^2 S}{\partial u \partial v};\ n = \frac{S_u \times S_v}{|\ S_u \times S_v\ |}$$

with n being the unit vector normal to the surface S at $S(u, v)$. The Gaussian and Mean curvatures are given by:

$$K = \frac{LN - MM}{EG - FF}; H = \frac{LG - 2MF + NE}{2(EG - FF)};$$

where the coefficients E, F, G, L, M, N are:

$$\begin{aligned} E = S_u \bullet S_u;\quad F = S_u \bullet S_v = S_v \bullet S_u; \\ G = S_v \bullet S_v;\quad L = S_{uu} \bullet n; \\ M = S_{uv} \bullet n;\quad N = S_{vv} \bullet n; \end{aligned} \tag{3}$$

Minimal, Maximal, Gaussian, Mean Curvatures from the Weingarten Application The Weingarten Application ([10]), W is an alternative way to calculate the Gaussian and Mean curvatures.

$$W = \begin{bmatrix} \frac{a_{11}}{a_{21}} & \frac{a_{12}}{a_{22}} \end{bmatrix}$$

with $a_{11}, a_{12}, a_{21}, a_{22}$ being:

$$a_{11} = \frac{MF - LG}{EG - F^2};\quad a_{12} = \frac{NF - MG}{EG - F^2};$$

$$a_{21} = \frac{LF - ME}{EG - F^2};\quad a_{22} = \frac{MF - NE}{EG - F^2}$$

The following facts allow to calculate the curvature measures for S from the Weingarten Application: (i) The eigenvalues $k1\ y\ k2$ of W are called **Principal Curvatures**, with k1 being the *maximal* curvature and k2 being the *minimal* curvature (assume that $|k1| \geq |k2|$). (ii) $K = det(W)$ is the **Gaussian Curvature**, with $K = k1 * k2$. (iii) $2H = \text{trace}(W)$ is twice the **Mean Curvature**, with $H = \frac{k_1 + k_2}{2}$ (iv) The maximal and minimal curvatures are: $k_1 = H + \sqrt{H^2 - K}$ and $k_2 = H - \sqrt{H^2 - K}$.

$W * v = k * v$ is the eigenpair equation for the W matrix. The solutions for such an equation are the eigenpairs $(k1, v1)$ and $(k2, v2)$. Therefore, $W * v_1 = k_1 * v_1$ and

$W * v_2 = k_2 * v_2$. The directions of principal curvature *in* $U \times V$ *space* are v_1 and v_2 ($v_1 = (w_{11}, w_{12})$ and $v_2 = (w_{21}, w_{22})$). The directions of maximal and minimal curvatures in R^3 are $u_1 = w_{11} * S_u + w_{12} * S_v$ and $u_2 = w_{21} * S_u + w_{22} * S_v$, respectively.

2.3 Previous Work

[12] implements an algorithm which starts with an already valid triangulation on a trimmed surface S(u, v) and originates a new triangular mesh. It proposes a surface triangulation with a Delaunay method given 3 points in $\mathbb{R}^3$ which determine a sphere whose equatorial plane is defined by the 3 given points. The algorithm creates a point set which may be more dense as needed by a particular criterion (e.g. curvature). This algorithm uses expensive operations (e.g. surface-line intersection). The boundary of the triangulated trimmed and meshed face is expressed and calculated in handled in parametric space. Since the algorithm in [12] starts with a given triangulation and modifies it, if such triangulation is not correct, or it does not respect the boundary of the trimmed surface, the triangulations following keep such characteristic. According to [16], the restricted Delaunay triangulation of general topological spaces is defined.

The restricted Delaunay triangulation in the case of trimmed surface in $\mathbb{R}^3$ is the dual of the Voronoi diagram intersected with the surface. Therefore, a triangle is created in each intersection of 3 voronoi cells with the surface. A contribution of the paper is to show that Chew's algorithm is a restricted Delaunay triangulation.

In the problem of the triangulation of manifolds with boundary the theoretical guaranties that serve for surface reconstruction do not apply. For example ε-*samples* ([4],[3]) which use the smallest distance of a sample point to the medial axis of the solid (i.e. the ε). Since a trimmed surface may be close or far from the medial axis, such criteria do not apply for surface triangulations.

In [7], The ball pivoting algorithm, (BPA), is presented. It computes a triangle mesh interpolating a given point cloud: 3 points form a triangle if a ball of radius smaller than ρ (a user specified radius) touches them without containing any other point. This triangle is a Gabriel 2-simplex in $\mathbb{R}^3$. The algorithm makes a region of triangles grow by adding a triangle to one of the boundary edges of the triangle mesh.

The reconstruction algorithm needs a very uniform sample.

In [19] the intrinsic Delaunay triangulation of a Riemannian manifold is shown to be well defined in terms of geodesics. A smooth surface embedded in $\mathbb{R}^3$ can define a Riemannian manifold. The Riemannian manifolds have the property that if all the calculations and definitions are done in a small subset of the manifold, (as they can be done with a good sampling condition), the Delaunay triangulation and the Voronoi diagram are defined exactly as with the euclidean metric and are dual. Although defining triangulations with geodesics is theoretically sound, it has a prohibitively high complexity because it implies the solution of simultaneous algebraic systems.

In [2] the Gabriel complex is defined for $\mathbb{R}^n$. For a set of points in $\mathbb{R}^3$ the Gabriel complex is composed of triangles whose smallest defined circumsphere is free of points in the set. The advantage with respect to [12] is that it does not need information about the surface. The

Umbrella filter algorithm described produces topologically correct triangulations. Our article takes advantage of such a definition, along with a curvature - sensitive point sample.
[5] gives lower bounds for densities of well distributed points in surfaces, based on Delaunay triangulations. [11] presents an algorithm to sample and triangulate a surface, but it uses computer expensive and not common operations. In [8] the concept of loose ε-sample is developed but the operations which implement it are computationally expensive.
[9] presents the Lipschitz-samples, analogous to ε-samples, but applied to piecewise smooth (Lipschitz) surfaces. Such a distance permits to sample a Lipschitz surface and to define a mesh on it. However, [9] does not present actual examples of the performance of the algorithm (as we do here). We do also address the sampling of edges which bound two incoming smooth surfaces by using the most larger of the two involved curvatures.
In [13], the greedy Delaunay - based surface reconstruction algorithm from a point sample is presented. The algorithm uses the fact that the Gabriel graph is a subset of the Delaunay triangulation (DT). From a starting triangle, it grows matching each of the edges in the boundary with a triangle in the DT that has the minimum radius. As disadvantages, we may note that the algorithm: (i) requires the usual distance for Delaunay triangulations, (ii) needs a very uniform sampling in the loops and (ii) does not provide guarantee in the reconstruction.
[1] is focused in the notion of envelope that is the covering of a 3-manifold created with spheres of λ size and centered in the points of the surface. From the envelope a surface with boundaries can be reconstructed, but this approach does not conserve the original points sampled in the boundary, and parameters are needed. In practice the envelope approach does not seem to produce topologically correct results. We dispose of information about the surface and boundaries and use another approach to the problem.
In [14] an advancing front method to triangulate parametric surfaces is presented. The method triangulates a B-Rep by discretizing edges and surfaces. The number of triangles generated can be adapted to any density function in the surface. The correctness of the solution depends on the density function provided for the edges and for the surface. In [6] a parameterization-independent algorithm is proposed to triangulate a surface. In the algorithm, a circle in the normal plane of a point p in the surface S, $Tp(S,\mathrm{p})$, is chosen. A polygon of n sides, (with n ≥ 4), and defined by vertices $\{p_1, p_2, .., p_n\}$, is inscribed in the circle. Rays from the vertices and perpendicular to $T_p(S, p)$, intersect the surface and generate new vertices for the triangulation. The algorithm has the advantage that the connectivity of the triangles is present through the algorithm. In the other side, the paper handles the boundary in the parameter domain and reports a non-uniform sample near to this. The paper reports problems are in regions of high curvature. Also in [21], the algorithm described in this paper is implemented and problems are reported near the boundaries. The generalization of their algorithm to closed surfaces needs a sewing procedure that creates additional borders. In [23], an algorihtm that triangulates parametric surfaces is presented. The algorithm uses an advancing front method. The loops aren't taken into account. This algorithm generates two fronts of triangles that advance one towards the other. The two fronts are in oposite sides of the parameter space. The main drawback in this algorithm is that: only a squared parameter space is considered. No holes or complex features are reported in the paper. In [22] an algorithm to triangulate b-reps is presented. In the algorithm all the triangulation occurs in parametric space and is mapped to

R^3. In [21] two sampling methods and a triangulation algorithm are proposed. In the algorithm the boundaries are isosampled, i.e not sensitive to the curvature or any other parameter. In the triangulation algorithm, a parametric information is needed, so it can fix problems, and the boundaries are not handled well in all the situations.

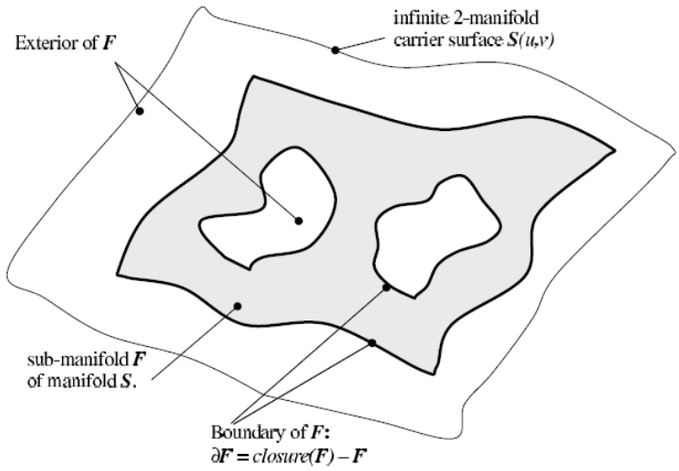


Fig. 4. Interior, boundary and exterior of a submanifold F with respect to a manifold S.

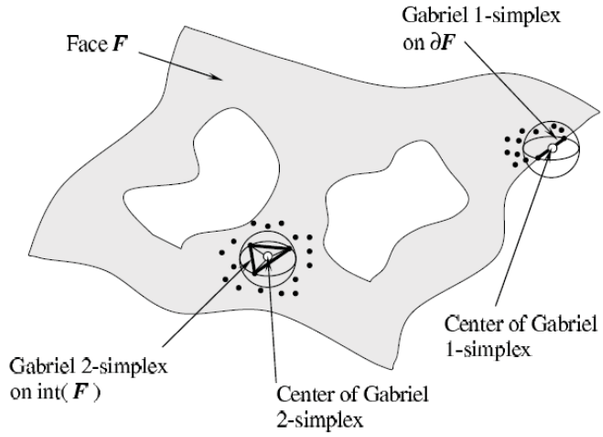


Fig. 5. Gabriel 1- and 2-simplices on face F

3. Methodology

The implemented algorithm to triangulate a face F mounted onto a parametric surface S (Fig. 4) has the following layout, whose details will be discussed later: (1) Calculate the pre-image F^{-1} of the face F through the function S (Fig. 2). (2) Initialize the vertex set V_T with a curvature-sensitive sample of the loops $L_0,..,L_n$ of the face boundary ∂F. (3) Introduce points in the sampled loops $L_0,..,L_n$; such that, all the segments in ∂F are Gabriel 1-simplex. (4) Sprinkle the face F with vertices v_i achieving a vertex density proportional to

the local curvature of F, K_{max}, inserting those vertices in set V_T. Segments in ∂F remain Gabriel 1-simplex during this stage. (5) Calculate a Gabriel connectivity T for the vertex set V_T.

3.1 Edge Sampling

Algorithm 1 is used to produce a curvature - sensitive sample of an Edge $\mathbf{E}$. Unlike previous approaches ([22]) such a sample is not an iso-distance one. Instead, the sampling interval at point p on the underlying curve C is sensitive to the largest of the maximal curvatures of S_1 and S_2 in such a point p (line 6). Notice that the curvature *of the curve* C at p needs not to be considered in addition to the surface curvatures because it will be always less than or equal to the surface maximal curvatures ($K_{max}(S_1, p), K_{max}(S_2, p)$).

Algorithm 1 Sample of the Edge $\mathbf{E}$ between Faces F_1 and F_2

$S_1(u,v), S_2(u,v)$: Underlying surfaces for Faces F_1 and F_2.
C(λ): Underlying Curve for $\mathbf{E}$.
Λ_0, Λ_f: Parameters of the extremes of $\mathbf{E}$ in curve C.
$V_E = \{p_1, p_2, .., p_n\}$: Output. Sequence of point sample of $\mathbf{E}$.
$K_{max}(S, p)$): Maximal curvature of Surface S at point p.
N_{sides}: Number of sides of a regular polygon.
1: $V_E = \{\}$
2: $\lambda = \Lambda_0$
3: **while** $\lambda \leq \Lambda_f$ **do**
4: $p = C(\lambda)$
5: $V_E = V_E \cup \{p\}$
6: $k = max(K_{max}(S_1, p), K_{max}(S_2, p))$
7: $r = 1/k$
8: $\delta = polygon\ determined\_arc(r, N_{sides})$
9: $\Delta\lambda = dist\_to\_param(\delta)$
10: $\lambda = \lambda + \Delta\lambda$
11: **end while**

Fig. 6 displays the geometrical idea behind lines 7 and 8 of the algorithm: the radius of curvature r is the inverse of the curvature k. A circle tangent to a curve with such a curvature may be approximated by a regular polygon of N_{sides} sides. The arc δ determined by such a polygon is considered as a good euclidean sampling distance

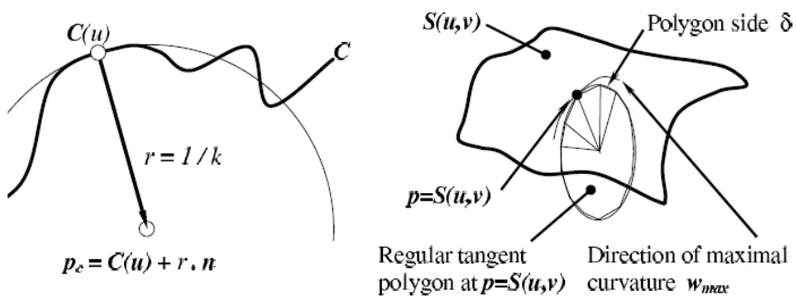


Fig. 6. Locally planar curve and local curvature. Approximation by regular polygon of N sides.

for the curve C at p (line 8). Such an euclidean distance must be transformed to a local parameter distance δλ at C(λ) (line 9).

3.2 Loop Resampling. Ensuring that each edge of each loop is a Gabriel 1-simplex

Algorithm 2 creates new vertices in the loops sampled by algorithm 1, in such a way that each segment in the new sample is a Gabriel 1-simplex. Between lines 4 and 16, each loop V_{Li} is traversed as a circular linked list. Each segment $v_{curr}v_{next}$ is tested to be a Gabriel 1-simplex in line 7. If it is not a Gabriel 1-simplex, a new point, returned by function *point_middle_of_arc* (lines 8 and 9), is inserted to the circular linked list after v_{curr} and previous to v_{next} (lines 10 and 11). Let C_Z (λ) be a curve parameterized by arc length. Let p_x and p_y be two points in C_Z (λ). Let Λ_X and Λ_Y be the parameters of p_x and p_y respectively with $\Lambda_X < \Lambda_Y$. Function *point_middle_of_arc* $(C_Z(\lambda), p_x, p_y)$ performs the following procedure:

1. Finds the arc length δ between p_x and p_Y in curve C_Z (λ).
2. Returns a point $p_{new} = C\left(\Lambda_x + \frac{\delta}{2}\right)$.

If any segment $v_{curr}v_{next}$ is not Gabriel 1-simplex, the variable $finished$ is set to $false$ (line 12). In line 21 the variable $finished$ is tested *true*, to ensure that this procedure is repeated until all segments are Gabriel 1-simplex.

Fig. 7 shows the behavior of algorithm 2. In Fig. 7(a), point $v_X \in V_{Li}$ is inside $B_G(v_{curr}, v_{next})$ and segment $v_{curr}v_{next}$ is not Gabriel 1-simplex. After v_{new} is inserted to V_{Li}, the new segments are (v_{curr}, v_{new}) and (v_{new}, v_{next}). As shown in Fig. 7(b), $B_G(v_{curr}, v_{new})$ and $B_G(v_{new}, v_{next})$ are empty of other points in $V_{\partial F}$; and segments (v_{curr}, v_{new}) and (v_{new}, v_{next}) are Gabriel 1-simplex.

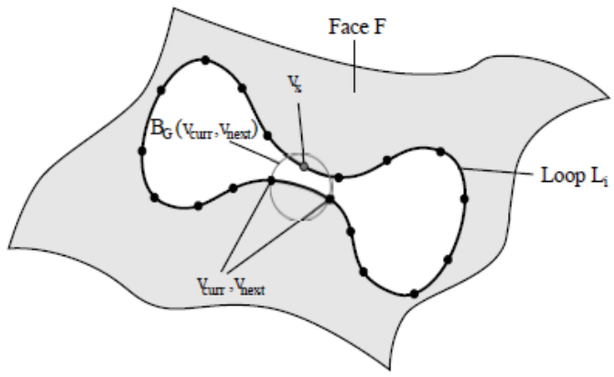
Sometimes, B-rep models are not well stitched ([24]), and that creates extremely narrow faces. Every time the loop between lines 1 and 21 is executed, at least 2 segments become shorter. In line 18, function *is_any_segment_too_short* $(V_{\partial F})$ evaluates this case and returns failure when an edge is too short (i.e the loop is being repeated too many times). This adds

robustness to algorithm 2. Otherwise, if two lines of a b- rep are geometrically equal, but have not been merged in the model, algorithm 2 would never stop.

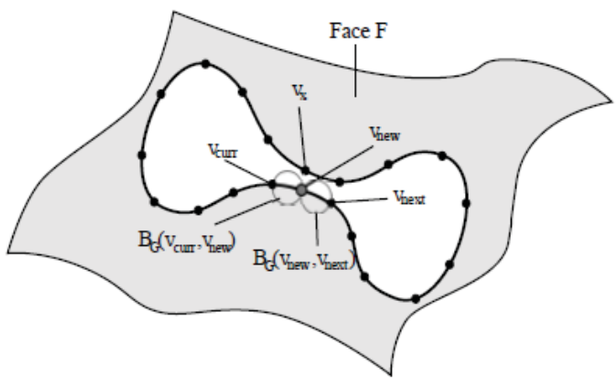
Algorithm 2 Insert vertices in the sampled loops until all the segments are Gabriel 1-simplex.

$V_{\partial F} = \{V_{L1}, V_{L2}, V_{Ln}\}$: is the set of vertices that sample the boundary of the face *F*.
$V_{Li} = \{V_{E1}, V_{E2}, .., V_{Em}\}$: is a circular linked list that contains all the points sampled in the loop with algorithm 1 and $V_{\mathbf{Ej}}$ is the ordered sample of edge $\mathbf{E_j}$.
$V_{\partial F} = \{V_{L1}, V_{L2}, V_{Ln}\}$. Output. The set of vertices that sample the boundary of face *F*.

1: **repeat**
2: $finished = true$
3: **for all** $V_{L1} \in V_{\partial F}$ **do**
4: $v_{curr} = head(V_{L1})$
5: $v_{next} = next(V_{L1}, v_{curr})$
6: **repeat**
7: **if** $\exists v_X \in (V_{L1} - \{v_{curr}, v_{next}\})$, **such that:** $v_X \in B_G(v_{curr}, v_{next})$ **then**
8: $C_j(\lambda)$ is the curve, of an edge $\mathbf{E_j}$, that contains $\{v_{curr}, v_{next}\}$.
9: $v_{new} =$ $point\_middle\_of\_arc(C_j(\lambda)\, v_{curr}, v_{next})$.
10: next of $(V_{L1}, v_{curr}) = v_{new}$
11: next of $(V_{L1}, v_{new}) = v_{next}$
12: $finished = false$
13: **end if**
14: $v_{curr} = v_{next}$
15: $v_{next} = \text{next}(V_{L1}, v_{next})$
16: **until** $v_{curr} \equiv \text{head}(V_{L1})$
17: **end for**
18: **if** $is\_any\_segment\_too\_short(V_{\partial F})$ **then**
19: **return** $FAILURE$
20: **end if**
21: **until** $finished \equiv true$



(a) First a sampled vertex vx is inside of B_G (v_{curr}, v_{next}). Segment (v_{curr}, v_{next}) is not Gabriel 1-simplex.



(b) When algorithm 2 inserts v_{new}, segment v_{curr} v_{next} is replaced by segments v_{curr} vnew and v_{new} v_{next}. No point sampled is inside balls B_G ($v_{curr,}$ v_{new}) and B_G (v_{new}, v_{next}). Segments v_{curr} v_{new} and v_{new} v_{next} are Gabriel 1-simplex.

Fig. 7. The two basic steps of algorithm 2.

3.3 Face Sampling. Vertex Sprinkle on Face F

Algorithm 3 constructs the vertex set V_F of the triangulation sought for face F. The initialization of V_F (line 1) is done with the vertices sampled on the boundary loops of F , $\partial F = \{L_0, .., L_n\}$, as per algorithm 1. Such vertices correctly sample ∂F. However, the interior $int(F)$ needs to be sampled. To do so, trial vertices are generated inside the pre-image F^{-1} in $U \times V$ space (line 4) and their image via S is calculated (line 7). Such a trial vertex p is rejected if (a) it is too close to other vertices already accepted in V_F (line 11) or (b) if it is contained in the smallest ball defined by a pair of vertices consecutive on a loop L_j. The closeness criteria is dictated by the maximal curvature $K_{max}(S(u,v))$ at $p=S(u,v)$ (line 5). In case (a) each already accepted vertex in V_f is tested for inclusion inside a ball $B(p,R)$

centered at p with radius $R = polygon\_side\ (r, N_{sides})$ (line 9). In case (b) each segment $v_i v_j$ in the sample of the border is tested as a Gabriel segment (1-simplex) with respect to the candidate p. If every segment of the border is Gabriel with respect to p, we assume that p is not too close to the border (line 10).

Algorithm 3 Sprinkle triangulation vertices on Face F

F: Input. Face to triangulate.
F^{-1}: pre-image of Face F in space $U \times V$
$S(u,v)$: Underlying surface for Face F.
$\partial F = \{L_0,..,L_n\}$: Loops Bounding the Face F.
N_f: Number of tolerated failures.
V_F: Output. Vertex set sampled on Face F.

1: V_F = sampling of boundary ∂F
2: $fails = 0$
3: **while** $fails \leq N_f$ **do**
4: generate parameter pair $(u,v) \in F^{-1}$
5: $k = K_{max}(S(u,v))$
6: $r = 1/k$
7: $p = S(u,v)$
8: $R = polygon\ side(r, N_{sides})$
9: **if** $\nexists q \in V_F$ such that $q \in B(p,R)$ **then**
10: **if** $\exists v_i v_j$, a segment of the boundary, **such that:** $p \in B_G(v_i, v_j)$ **then**
11: $fail = fail + 1$
12: **else**
13: $V_F = V_F \cup \{p\}$
14: $fail = 0$
15: **end if**
16: **else**
17: $fail = fail + 1$
18: **end if**
19: **end while**

Fig. 8. Goal Point Population on face F

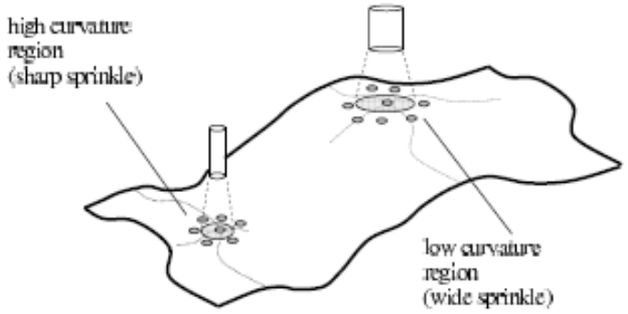


Fig. 9. Curvature-sensitive Sprinkle Airbrush F

A segment is said to be *sampled in the boundary*, if its two end vertices are consecutive in a loop $L_j \in \partial F$. If tests (a) and (b) are passed, p is accepted in V_F (line 13). Fig. 6 depicts that the value for R is computed as the cord of the N_{sides} -regular polygon inscribed in the circle with radius $1/k$. Function *polygon_side*(r, N_{sides}) equals to $2r \sin(\pi/N_{\text{sides}})$. Fig. 5 displays the two tests mentioned in items (a) and (b) above.

3.4 Face Triangulation. Gabriel Connectivity on Vertex Set V_T.

Algorithm 4 builds the connectivity inside the vertex set $V_{F.}$ The algorithm seeks to complete edges (v_0, v_1) already known to belong to the triangulation T (line 6) with an additional vertex v_{new} to build a Gabriel Triangle (v_0, v_1, v_{new}) (line 9).

Any internal Gabriel triangle is the first formed triangle (lines 1,4). It is also a seed to initialize the Queue of edges potentially able to span Gabriel triangles.

If the edge extracted from the Queue is part of the boundary, it is not expanded any more (line 7). All the edges which are part of the boundary will be found because they are Gabriel 1-simplex and make part of a Gabriel 2-simplex. If a Gabriel triangle (v_0, v_1, v_{new}) can be built, it is added to the triangulation T (line 10). If a Gabriel triangle can be built using only an existing edge (v_0, v_1) and a new vertex v_{new}, the general situation is that the new edges (v_0, v_{new}) and (v_{new}, v_1) should be queued to be eventually expanded (line 20). However, this is not always the case, since such a triangle may use 1 or 2 *additional* edges

Algorithm 4 Triangle Connectivity in the set V_F

V_F : Input. Vertex set sampled on Face F

Queue: List of triangle edges to expand.

$\partial F = \{L_0,..,L_n\}$: Loops Bounding the Face F.

T: Output. Triangulation.

1: $seed = triangle\_in\_interior(F)$
2: $\{(v_0,v_1),(v_1,v_2),(v_2,v_0)\} = edges\_of\_triang(seed)$
3: $Queue = \{(v_0,v_1),(v_1,v_2),(v_2,v_0)\}$
4: $T = \{seed\}$
5: **while** $(Queue = \Phi)$ **do**
6: *edge_to_expand = extract(Queue)*
7: **if** *edge_to_expand is not part of the sample of the boundary* **then**
8: $(v0,v1) = vertices(edge\ to\ expand)$
9: $v_{new} = vert\_for\_Gabriel\_2\_Simplex(V_F,v_0,v_1)$
10: $T = T \cup \{(v0,v1,v_{new})\}$
11: **if** $((v0,v_{new}) \in Queue) \wedge ((v_{new},v_1) \in Queue)$ **then**
12: $Queue = Queue - \{(v_0,v_{new}),(v_{new},v_1)\}$
13: **else if** $((v_0,v_{new}) \in Queue)$ **then**
14: $Queue = Queue - \{(v_0,v_{new})\}$
15: $Queue = Queue \cup \{(v_1,v_{new})\}$
16: **else if** $((v_{new},v_1) \in Queue)$ **then**
17: $Queue = Queue - \{(v_{new},v_1)\}$
18: $Queue = Queue \cup \{(v_{new},v_0)\}$
19: **else**
20: $Queue = Queue \cup \{(v_1,v_{new}),(v_{new},v_0)\}$
21: **end if**
22: **end if**
23: **end while**

already in the queue. In the first case, the triangle is filling a corner (lines 13-18). In the second case, the triangle is filling a triangular hole (lines 11,12). In such special cases additional edges (1 or 2 besides the expanded one) should be taken away from the queue.

4. Complexities of the algorithms

Time and space complexities of all the algorithms were found. They are all output sensitive; that is, their complexities depends on the size of the output given by them. The first 3 algorithms depend on the number of nodes generated by them. The last algorithm depends on the number of nodes in the input and in the number of triangles generated.

4.1 Edge Sampling

The time and space complexities of algorithm 1, have been found in the following manner.
1. Time complexity. The operations with the curve and the operations to find the curvatures are dependent upon the parameterization and not in the number of points generated. Because of this, the time complexities of all the operations within the loop, (lines 3 to 11), can be assumed as O (1). The loop is repeated N_{Ej} times. N_{Ej} is the number of points generated to sample the edge E_j. The time complexity of algorithm 1 is O (N_{Ej}). 2. Space complexity. As the algorithm only stores the points generated, the space complexity is O (N_{Ej}).

4.2 Loop Resampling

The time and space complexities of algorithm 2, have been found in the following manner:

1. Time complexity. Let $N_{\partial F}$ be the number of vertices in $V_{\partial F}$, at the end of algorithm 2. For algorithm 2 the following facts hold:

   (a) $N_{\partial F}$ changes. In the worst case it grows as an arithmetic progression with difference 1. That is why in this paper the calculations are simplified by considering, at any step, $N_{\partial F}$ as the number of vertices in $V_{\partial F}$.

   (b) The number of segments in $V_{\partial F}$ is the same as the number of points.

   (c) Each time a segment $v_{curr}\, v_{next}$ is tested to be Gabriel 1-simplex, (line 7), algorithm 2 tests all the points in $V_{\partial F}$. This takes time O ($N_{\partial F}$).

   (d) The number of segments tested will be O ($N_{\partial F}$), no matter the number of points added to the sample in the previous step.

   (e) The worst case scenario occurs when only one point is added at the time.
   This is because of fact (d). In that case, the loop from lines 1 to 21 is repeated $N_{\partial F}$ times.

   (f) The worst case escenario occurs when only 3 vertices have been generated by algorithm 1. This is the worst case because it means that all but 3 of the points in $V_{\partial F}$ are generated by algorithm 2. The number of times that the loop between lines 1 to 21 is repeated is O ($N_{\partial F}$).

   Combining facts (c), (d) and (e) the worst case time complexity of the algorithm 2 is O ($N_{\partial F}^3$).

2. Space complexity. Only $V_{\partial F}$ is stored by the algorithm. The space complexity of algorithm 2 is O ($N_{\partial F}$).

4.3 Face Sampling

The time and space complexities of algorithm 3, have been found in the following manner:

1. Time complexity. The algorithm terminates if variable $fails > N_f$; so for each new point, the algorithm tries at most N_f times. The number of times that the loop between lines 3 and 19 is O ($N_f \times N$), being N_F the number of points generated in the interior of the face. In the loop, for a new generated point p two tests are performed:

(a) In line 9, every q ∈ V_F is tested for inclusion in $B(p,R)$. R is as described in line 8. This operation can be performed in O ($N_F + N_{\partial F}$).

(b) In line 10, p is tested for inclusion in every B_G (v_i , v_j), where v_i , v_j are two consecutive points in the sample of the boundary of F. This operation can be performed in O ($N_{\partial F}$).

The worst complexity is that of test (a).

Combining test (a) with the number of times the loop between lines 3 and 9 is repeated, we have that the complexity of the algorithm is: O ($N_f \times N_F$ ($N_F + N_{\partial F}$)).

2. Space complexity. The algorithm only stores the points that are accepted. The space complexity of the sampling algorithm is O (N_F).

4.4 Face Triangulation

The time and space complexities of the algorithm 4, have been found in the following manner.

1. Time complexity. For algorithm 4, the following facts hold:

(a) Each time that the loop (lines 5 to 23) is repeated, this algorithm checks a different edge that belongs to the triangulation. The number of edges that belong to the triangulation is a linear function of the number of triangles (i.e each new triangle adds a maximum of 3 edges). The number of triangles generated will be denoted as: N_T.

(b) The operation *vert_for_Gabriel_2_Simplex* (line 9), is the one that has the highest complexity within the loop (lines 5 to 23). The rest of the operations have O (1) complexity.

(c) For the *vert_for_Gabriel_2_Simplex* (line 9) operation, first a candidate vertex (r) is chosen. This vertex can complete a Gabriel simplex given the edge $v_0 v_1$. All the points in V_F , except for $v_0 v_1$ and r are tested for inclusion in B_G ($v_0 v_1, r$). Using a naive approach, the time complexity of this operation would be $O(N^2)$, where N is the number of vertices in V_F.

Combining facts (a), (b) and (c), the complexity of algorithm 4 is O ($N_T \times N^2$).

2. Space complexity. The algorithm stores a set of edges in Queue. As a topological constrain, Queue can only contain the same edge twice. The number of edges stored is, in the worst case, a linear function of the number of triangles stored. The space complexity is O (N_T).

5..Results

Several Boundary Representations B-Reps were used to test the implemented algorithm, proposed in this article. Such B-reps have genera 3 or superior, and present faces F whose underlying surfaces S are parametric ones of the NURBS or Spline types. An $N_f = 1000$ maximal number of failed trials was used to stop the sprinkle of vertices on F (generation of the set V_F). The number of sides for the approximating polygon was $N_{sides} = 30$. Figs. 10,

11 and 13 show complex B-Reps. Other examples of B-reps triangulated include a model of a pre-columbian fish in Fig. 14, a support of an axle in Fig. 15, and a stub axle in Fig. 16.
The attention of the reader is called to the fact that the connectivity construction is a process completely independent of the vertex generation process. Since the vertex generation algorithm (Sprinkle) is the most critical one, the execution time was recorded for such an aspect.
For the models *Pump* and *Hands*, Figs. 12(a) and 12(b) show execution times, corresponding to the vertex generation process. Fig. 12(c) shows the comparison of vertex generation times for such runs.

Fig. 10. Pump carter [17]. Colormap according to quality of triangles.

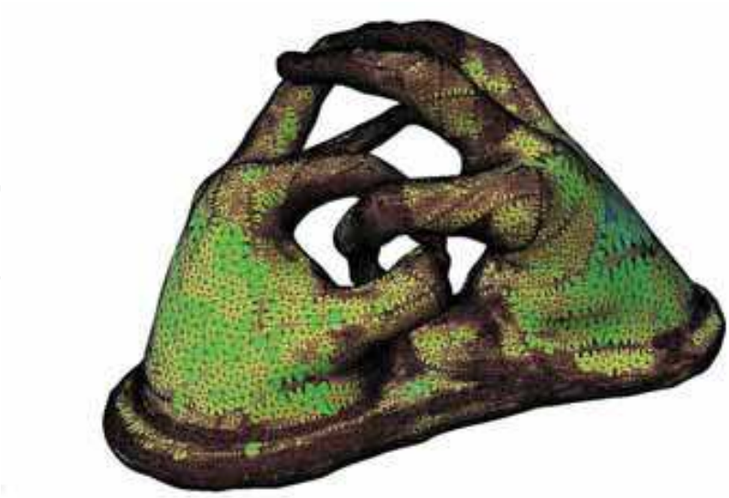
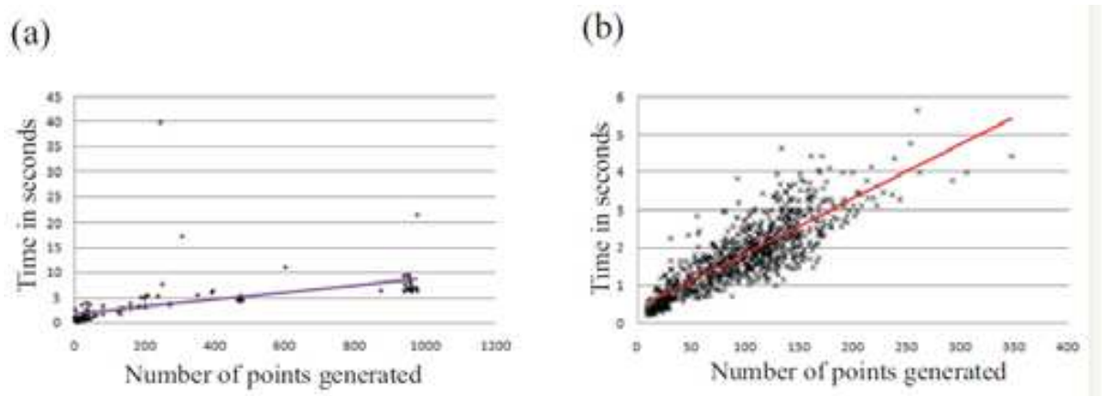
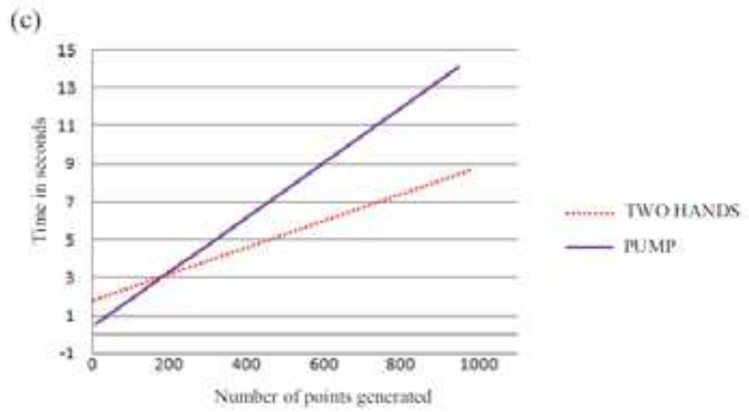


Fig. 11. 2 hands with 3 genus, scanned and reconstructed using RainDrop Geomagic. Colormap according to the size of the triangles



(a) Sample time per face for the pump. (b) Sample time per face for the two hands.



(c) Comparison between the hands (NURBS) and the Pump Carter. Hands time in solid line, Pump time in dotted line

Fig. 12. Times spent sampling the faces and their comparison.

Fig. 13. Other view of the 2 hands with 3 genus. Colormap according to the quality of the triangles.

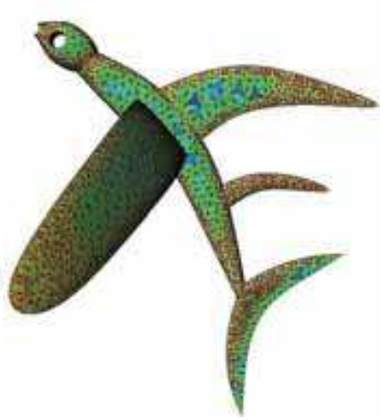


Fig. 14. Artificial replica of a pre-columbian gold fish [15]. Colormap according to size of the triangles

Fig. 15. Support of an axle. Colormap according to size of the triangles

Fig. 16. Stub axle [18]. Colormap according to the quality of the triangles

6. Conclusions and future work

The proposed algorithm for generating triangulation vertex sets and for calculating the connectivity among them proved to function correctly, even for very extreme geometries and topologies. Several aspects of the algorithm must be addressed: the continuity of triangle sizes at the Face Edges, the possibility of undertaking re-meshing of already existing triangulations and its related endeavor, namely the level of detail, necessary for Finite Element Analysis applications. Additional research is needed in algorithms that (i) take advantage of the concepts presented in the heuristic algorithm proposed here, but (ii) can be proved correct.

7. References

K. Abe, J. Bisceglio, T. J. Peters, A. C. Russell, and T. Sakkalis. (2005). Computational topology for reconstruction of surfaces with boundary: Integrating experiments and theory. In *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, pages 290–299, Washington, DC, USA, 2005. IEEE Computer Society.

Udo Adamy, Joachim Giesen, and Matthias John. (2000). New techniques for topologically correct surface reconstruction. In *VIS '00: Proceedings of the conference on Visualization '00*, pages 373–380, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

N. Amenta, M. Bern, and D. Eppstein. (1998). The crust and the beta-skeleton: Combinatorial curve reconstruction. *Graphical models and image processing: GMIP*, 60(2):125–, 1998.

Nina Amenta and Marshall Bern. (1998). Surface reconstruction by voronoi filtering. In *SCG '98: Proceedings of the fourteenth annual symposium on Computational geometry*, pages 39–48, New York, NY, USA, 1998. ACM.

Dominique Attali, Jean-Daniel Boissonnat, and André Lieutier. (2003). Complexity of the delaunay triangulation of points on surfaces the smooth case. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 201–210, New York, NY, USA, 2003. ACM.

Marco Attene, Bianca Falcidieno, Michela Spagnuolo, and Geoff Wyvill. (2002). A mapping-independent primitive for the triangulation of parametric surfaces. *Graphical Models*, 65(5):260 – 273, 2003. Special Issue on SMI 2002.

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, and Gabriel Taubin. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 5(4):349–359, 1999.

J-D Boissonnat and S. Oudot. (2004). An effective condition for sampling surfaces with guarantees. In *SM '04: Proceedings of the ninth ACM symposium on Solid modeling and applications*, pages 101–112, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

Jean-Daniel Boissonnat and Steve Oudot. (2006). Provably good sampling and meshing of lipschitz surfaces. In *SCG '06: Proceedings of the twenty-second annual symposium on Computational geometry*, pages 337–346, New York, NY, USA, 2006. ACM.

Manfredo Do Carmo. (1976). Differential geometry of curves and surfaces, pages 1–168. Prentice Hall, 1976. ISBN: 0-13-212589-7.

Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos, and Tathagata Ray. (2004). Sampling and meshing a surface with guaranteed topology and geometry. In SCG '04: Proceedings of the twentieth annual symposium on Computational geometry, pages 280–289, New York, NY, USA, 2004. ACM.

L. Paul Chew. (1993). Guaranteed-quality mesh generation for curved surfaces. In SCG '93: Proceedings of the ninth annual symposium on Computational geometry, pages 274–280, New York, NY, USA, 1993. ACM.

David Cohen-Steiner and Frank Da. (2002). A greedy delaunay based surface reconstruction algorithm. Research report, INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 2002.

JC Cuillière. (1998). An adaptive method for the automatic triangulation of 3d parametric surfaces. Computer-Aided Design, 30(2):139 – 149, 1998.

Museo del Oro Bogotá DC. Pre-columbian fish. http://www.banrep.org/museo/eng/home4.htm.

Herbert Edelsbrunner and Nimish R. Shah. (1994). Triangulating topological spaces. In SCG '94: Proceedings of the tenth annual symposium on Computational geome- try, pages 285–292, New York, NY, USA, 1994. ACM.

Rosalinda Ferrandes. Pump carter. http://shapes.aimatshape.net/viewgroup.php?id=919.

Rosalinda Ferrandes. Stub axle. http://shapes.aimatshape.net/viewgroup.php?id=917.

Greg Leibon and David Letscher. (2000). Delaunay triangulations and voronoi diagrams for riemannian manifolds. In SCG '00: Proceedings of the sixteenth annual symposium on Computational geometry, pages 341–349, New York, NY, USA, 2000. ACM.

Ángel Montesdeoca. (1996). Apuntes de geometría diferencial de curvas y superficies. Santa Cruz de Tenerife, 1996. ISBN: 84-8309-026-0.

Carlos Cadavid Juan G. Lalinde. Oscar Ruiz, John Congote. (2008). A curvature-sensitive parameterization-independent triangulation algorithm. In 5th Annual International Symposium on Voronoi Diagrams in Science and Engineering. 4th International Kyiv Conference on Analytic Number Theory and Spatial Tessellations. (Kokichi Sugihara and Deok-Soo Kim, eds.). Drahomanov National Pedagogical University, 2008.

Oscar E. Ruiz and Sebastian Peña. (2007). Aspect ratio and size-controlled patterned triangulations of parametric surfaces. In Ninth IASTED Intl. Conf. Computer Graphics and Imaging, February 13-15 2007.

Baohai Wu and Shangjin Wang. (2005). Automatic triangulation over three-dimensional parametric surfaces based on advancing front method. Finite Elements in Analysis and Design, 41(9-10):892 – 910, 2005.

Hong-Tzong Yau, Chuan-Chu Kuo, and Chih-Hsiung Yeh. (2003). Extension of surface reconstruction algorithm to the global stitching and repairing of stl models. Computer-Aided Design, 35(5):477 – 486, 2003.

Recent Advances in Technologies

Edited by Maurizio A Strangio

ISBN 978-953-307-017-9

Hard cover, 636 pages

Publisher InTech

Published online 01, November, 2009

Published in print edition November, 2009

The techniques of computer modelling and simulation are increasingly important in many fields of science since they allow quantitative examination and evaluation of the most complex hypothesis. Furthermore, by taking advantage of the enormous amount of computational resources available on modern computers scientists are able to suggest scenarios and results that are more significant than ever. This book brings together recent work describing novel and advanced modelling and analysis techniques applied to many different research areas.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Oscar E. Ruiz, John E. Congote, Carlos Cadavid, Juan G. Lalinde, Guillermo Peris-Fajarnes, Beatriz Defez-Garcia and Ricardo Serrano (2009). Gabriel-constrained Parametric Surface Triangulation, Recent Advances in Technologies, Maurizio A Strangio (Ed.), ISBN: 978-953-307-017-9, InTech, Available from: http://www.intechopen.com/books/recent-advances-in-technologies/gabriel-constrained-parametric-surface-triangulation

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen