TOWARD QUALITY SURFACE MESHING

Jean Cabello

EDS/PLM Solutions, 2000 Eastman Dr, Milford, OH, USA jean.cabello@eds.com

ABSTRACT

This paper presents recent progress and extensions to **T**ri**Q**ua**M**esh (**TQM**) [1], targeted at providing good quality surface meshes: Increased robustness of the 1D mesh generator to handle highly non linear size variations; interior node generation driven by a size variation interpolation domain; improved mesh distortion reduction between the parameter space and the physical space. The concepts of Size Control, Size Map and Triangle Map are introduced to increase the flexibility and the control on the final mesh. These concepts are general and apply to any meshing algorithm, although they will be illustrated with TQM.

Keywords: surface mesh generation, triangular, size map, curvature adaptation.

1. INTRODUCTION

The most common tetrahedral meshing algorithms, advancing front and Delaunay, require the surface mesh to be generated first, prior to filling in the interior with tetrahedral elements. For volume skin surfaces with geodesic distances between two points on the surface high compared to the Euclidian distance (i.e. narrow and high curvature passageway), adaptation of the surface mesh to the curvature might be critical to the success of an automatic volume mesher by preventing geometrical surface mesh intersection. The success and the quality of the volume mesher is then directly impacted by the success and quality of the surface mesher.

Although tremendous progress has been made with regard to meshing algorithms in both two and three dimensions, it still remains a difficult task to surface mesh any collection of surfaces with good quality and size control.
Many approaches are available depending on the surface definition available (continuous or discrete).
For CAD parametric surfaces, a 2D parameter space representation of the surface is available and surface meshing is reduced to a 2D meshing problem. However, the surface can be poorly parameterized leading to high distortion when mapping the mesh back from 2D to 3D space. Some methods have been presented to account for the distortion between the 2D and 3D space using the CAD Riemannian surface evaluators [2].

Most CAD systems can export an STL or facetted representation of any parametric surface. This is a lower level definition of the surface that has the advantage of a simple and common format independent of the CAD system.

For discrete data representations of the surface (STL data or legacy data), some techniques work directly on the 3D discrete data to obtain a good quality mesh [3] while others use a divide and conquer approach to select a region and derive a parameter space to reduce the surface meshing problem to 2D. The two most common techniques used to derive a parameter space are: projection techniques, for example Maximum Area Plane (MAP) in I-DEAS, and flattening techniques based on angles [4] or based on lengths [5].

Adaptive meshing based on error estimation is another instance where controlling the mesh size variation to refine in areas of high error and coarsen in areas of low error is critical to obtain a good solution with reduced node and element count.

In this paper a simple method to account for the distortion between the 2D and 3D space for a surface represented by STL data is presented. Adaptive meshing based on discrete surface curvature is also presented in order to increase the mesh fidelity to the original surface at an economical cost compared to a constant size mesh.

1. TQM MESHING ALGORITHM IN A NUTSHELL.

The TQM algorithm is a divide and conquer meshing algorithm. Boundary loops are discretised using a 1D mesh generator. They are then joined into one single contour loop resulting in a loop of nodes. The contour loop is recursively subdivided into two sub contour loops along a "best split line" until the sub contour loop has been reduced to a trivial loop i.e. a loop with 3 points for a triangles or a loop with 4 points for quadrangles. All the details can be found in reference [1][6]. The remainder of this section recalls the two main points of interest for our discussion.

- Generation of contour points: Let's assume that we have a curve Γ parametrised using the arc length with total arc length L. Let's assume that we have a continuous grading function g(s) that represents the grading (or size) value along the parameter location. The 1D mesh generation problem can be stated as 1) *how many points (nPoin) to generate*? 2) *Compute the parameter location for these points that will satisfy the grading function requirements*. In [1] a solution was

proposed assuming a) Grading function is known discretely at (nSample) sample or basis points b) the grading function is assumed to be piecewise harmonic (fig. 1a).

$$g(s) = \frac{g_j + g_{j+1}}{2} + \frac{g_j - g_{j+1}}{2}.\cos\frac{(s - s_j)}{(s_{j+1} - s_j)}\pi$$

$$s \in [s_j, s_{j+1}] \; j = 0, nSample-1; nSample \geq 2 \qquad (1)$$

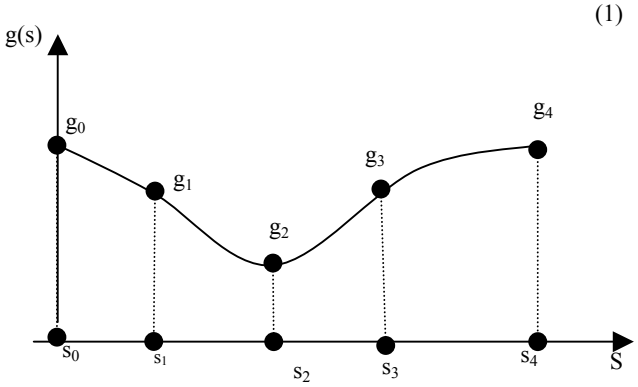

Fig. 1a – grading piecewise harmonic interpolation

Given nSample points, their respective parameter locations and grading values (s_j, g_j), the number of points is derived by requiring to meet "at best" the equidistribution for each interval:

$$\frac{s_i - s_{i-1}}{\frac{1}{2}(g_i + g_{i-1})} = C^{te} \quad \forall\, i = 1, nPoin \qquad (2)$$

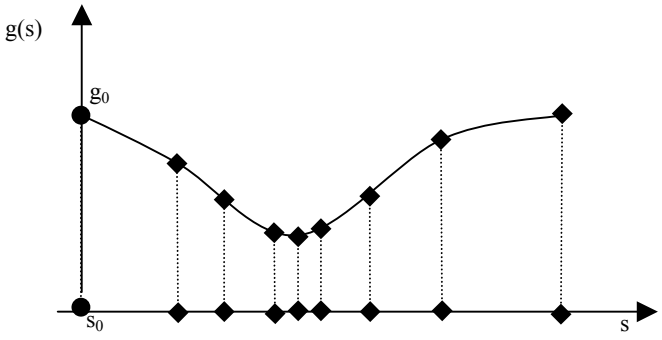

Fig. 1b –Nodal point distribution

Notice that in the equation above $nPoin$ and s_i are unknowns as well as $g_i = g(s_i)$. Summing equation (2) over all the intervals we obtain:

$$nPoin - 1 = \frac{1}{C^{te}} \int_0^L \frac{1}{g(s)} ds = \frac{1}{C^{te}} \sum_{j=1}^{j=nSample} \frac{s_j - s_{j-1}}{\frac{1}{2}(g_j + g_{j-1})} \qquad (3)$$

In equation (3) the number of sample points, their parameter locations and their grading values is known. $nPoin$ is computed by rounding the result of equation (3) to the nearest integer. The parameter locations of the 1D mesh points (fig.1b) are obtained by solving the system:

$$\frac{s_i - s_{i-1}}{(g_i + g_{i-1})} = \frac{s_{i+1} - s_i}{(g_{i+1} + g_i)} \quad \forall\, i = 1, nPoin - 1$$
$$s_0 = 0 \; ; \; s_{nPoin} = L. \qquad (4)$$

derived from the equidistribution equation (2). More details on the solution of equation (4) are given in section 2.1.

Once the boundary loops have been discretised, boundary nodes are assigned grading values. The loops are joined into one single contour loop. A "best split line" criterion is used to join the loops and the 1D mesh generation technique is applied to determine how many points and their locations along the split line. The sample points are the two end points of the split line where the grading values are known. This process is applied recursively.

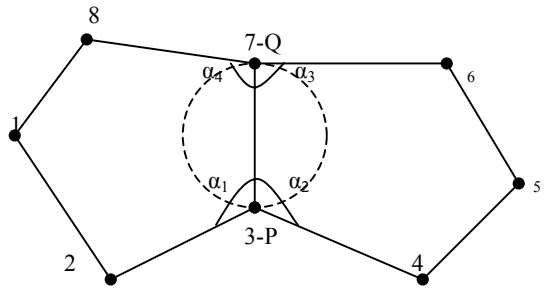

Fig. 2 – Parameters for the best split line (P-Q) and polygon diameter

- Best Split line: Given a point P, we find the set of admissible points Q that are visible from P. The best split line from P is the line [P,Q] that minimizes the objective function :

$$F_P(Q) = W_{angle}\delta(\alpha) + W_{Length}\delta(l) + W_{nPoin}\delta(n)$$

The weight factors W are constants. The first term, is minimal when the angles $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ (fig. 2) tend to multiples of 60 or 90 (respectively for triangles and quadrilaterals). The second term is minimal when the split line length tends to the minimum diameter of the boundary loop (i.e. the diameter of the smallest inscribed circle passing through two boundary points). The last term is minimal when the round off in equation (3) to obtain the integer $nPoin$ is minimal. Other choices for the function are possible, see [6], [7].

2. TQM MESHING ALGORITHM SHORTCOMNGS

In this section we try to highlight some of the limitations of the TQM approach as it is currently implemented in I-DEAS.

2.1 1D Boundary discretisation

Given $Npoin$, their location is found by solving the non-linear system of equations (4). One has to find the solution $S = (s_{1,} s_{2,\cdots,} s_{nPoin-1})$ that satisfies the system of non linear equations:

$$f_i(S) = 0\,; \quad \forall\, i = 1, nPoin - 1 \qquad (5)$$

$$f_i(S) = L_i \times s_{i-1} + D_i \times s_i + U_i \times s_{i+1} \qquad (6)$$

$$L_i = g(s_{i+1}) + g(s_i)\,; U_i = g(s_i) + g(s_{i-1})$$
$$D_i = -(g(s_{i-1}) + 2 \times g(s_i) + g(s_{i+1})$$

Using Newtown-Raphson method for the non linear system of equations reduces to solve the tridiagonal system of equations:

$$\underline{T}.\delta S = RHS \qquad (7)$$

$$\begin{cases} T_{i,i-1} = \dfrac{\partial f_i}{\partial s_{i-1}} = L_i + g'(s_{i-1}) \times \Delta s_i \\ T_{i,i} = \dfrac{\partial f_i}{\partial s_i} = D_i + g'(s_i) \times (\Delta s_i - \Delta s_{i-1}) \\ T_{i,i+1} = \dfrac{\partial f_i}{\partial s_{i+1}} = U_i - g'(s_{i+1}) \times \Delta s_{i-1} \end{cases}$$

$$with\ \Delta s_i = s_{i+1} - s_i\,;\ RHS_i = -f_i(S); \delta S = S^{(n)} - S^{(n-1)}$$

The index n represents the iteration number. The system is solved using LU decomposition with forward-backward substitution. The initial solution is taken as uniform distribution of the interior points. The convergence of the system depends on the matrix T condition number, which is not known. However, notice that if the system is linearised, setting $g'(s) = 0$, the matrix becomes well conditioned and is diagonally dominant. Based on this observation, a strategy is described in section 3.2 that improves the robustness of the 1D boundary mesh generation.

2.2 Boundary driven control only

The TQM algorithm first creates nodes along "best" split lines and then creates the elements. The node distribution along the split line is mainly driven by the grading value at the end points. The effect is that the boundary mesh is the main driver for the interior mesh, and undesirable boundary effects can propagate in the interior. The user has a good control on the element size variation on the boundary, but the control in the interior and the mesh transition is more difficult. For example, in I-DEAS, the user can input a local element length in the interior, but he cannot control its radius of influence. Also, when the surface exhibits curvature in its interior but its boundaries are flat there is no easy way to automatically refine the mesh with respect to the curvature. In most cases the user will have to manually add interior local element lengths in these areas to get the desired effect. To overcome these drawbacks and provide a mean to automatically refine the mesh in the interior of a surface, with no user interaction, TQM was extended to work with a background mesh, presented in section 3.2.

2.3 High distortion

TQM is a 2D mesh generator that generates a triangular or quadrangular mesh in a parameter domain. This parameter domain can be developed directly from a CAD parameter space or indirectly through projection or flattening techniques. In many cases, the mapping between the parameter domain and the physical domain is not isometric and elements size and quality need to be adjusted in the 2D domain to result in the desired mesh size and quality in the 3D domain. There are many approaches available to account for the local mesh distortion during the mapping. The most common approach [2] relies on the CAD query of continuous operators such as Curvature. These can turn to be expensive queries.

Currently in I-DEAS, to minimize the computational cost and account for length distortion between the 2D and 3D space, each split line is sampled with $nSample$ (10) interior points. These points are mapped back in 3D space and we compute the variation $d\upsilon / ds$ where $d\upsilon$ is the arc length variation of the curve poly-line in 3D space and ds is the corresponding variation of the split line in 2D space. This local scaling factor is then used to map the 3D mesh size to a corresponding 2D mesh size in the parameter domain. This is a very simple and robust approach, however one main drawback is that the scaling is unidirectional (along the split line). In section 3.3, a different approach to account for the local distortion is discussed.

3. TQM EXTENSIONS AND ENHANCEMENTS

The main usage of TQM in I-DEAS is for structural analysis with a constant mesh size. In this range the software performs fairly well. For boundary curvature adaptation the process is automated but might sometimes become unstable, resulting in poor node distribution transition. The interior surface curvature adaptation is not automated and has to be done through user input of interior local element sizes.

From now on, a stitched tessellation representing "accurately" the 3D surface or surfaces to be meshed (for example an STL representation from a CAD system) is assumed to exist. The corresponding 2D tessellation in the parameter domain, thus a discrete one to one mapping between 2D and 3D space, is also assumed to exist. In short, a triangle map, discuss in section 6, is available (see figures 7a, 7b).

They are many possible answers to the critical and common adaptive sampling questions: How many sample points? Where? What size? One answer could be to delegate the responsibility to the user. To get the desired sampling adaptation to the curvature both on the boundary and on the surface, it is proposed to leverage the surface STL representation. For the boundary curves, the facet points provide an adaptive sampling of the boundary curvature (see fig. 4).

3.1 1D boundary discretisation

For uniformly distributed and smoothly varying grading values, the non-linear tridiagonal system (7) exhibits a unique solution because the non-linear terms cancel out and the matrix is still well conditioned. However, when using an adaptive sampling point strategy with high grading values gradients, the system becomes ill–conditioned and might never converge to a solution. The solution algorithm has been enhanced by monitoring the convergence of the non linear system and when the solution oscillates and does not converge, after a fixed number of iterations, we restart the solution using the current solution but this time solving the linear system rather than the non linear one. This approach has proven to be very robust and is able to handle highly non-linear distributions and grading variations, even extreme cases with noisy input data. This strategy has the desired effect to smooth out the non-linearity due to high frequency input data.

3.2 Flexible background mesh approach

In order to provide better control over the interior element size variation, as is necessary in surface curvature adaptation, the TQM algorithm was enhanced to work in conjunction with a background mesh that provides an interpolation domain for the size variation. There are two type of background meshes used: 1) a background mesh resulting from the flattened faceted representation that is used as an interpolation domain for the 2D to 3D mapping function (see fig. 7d) and 2) a background mesh resulting from an initial sample mesh (see fig. 7e).
The split line node generation was modified. As was discussed in section 2.3, the split line is still sampled with 10 uniformly distributed sample points, however, the grading at the sample points is determined by interpolation. As the nodes are equally spaced along the straight line, the interpolation is quite fast since the result of the previous node triangle location is used to start a triangle walk to locate the next one. Also, special attention has been given to the robustness of the triangle walk algorithm in order to handle highly stretched, even flat, triangles that often occur in STL data.

3.3 Distortion correction

Our approach to account for the local distortion is to first create a sample mesh in the 2D domain, map it back to 3D space using the facet triangle map and compute the length distortion at the sample points. This gives a length scale factor $\lambda_i = \sum_{j \in S(i)} l_{ij}^{2D} / \sum_{j \in S(i)} l_{ij}^{3D}$ at each sample points, that multiplied to the 3D grading value represents the 2D grading value. S(i) represent the ring of first neighbor nodes to node i. The length scale factor provides a local and isotropic estimate of the size distortion, is computationally inexpensive. One drawback is that there is no attempt to create stretched elements in the 2D space, only size varies. Given the 3D size variation, the scaling factor is applied. For example, a 3D constant size in 3D space will result in a varying element size in 2D space. The 2D mesher is then instantiated again with the 2D sample mesh as the background mesh with computed 2D sample grading values that drive the resulting final mesh. The 2D final mesh is mapped back to 3D space using the facet background grid.

3.4 Mesh transition

For constant size meshing, using the interpolation domain to determine the grading values along the split line can lead to sudden jumps in mesh size. One could smooth out the field of mesh size to get a smoother distribution. Instead, a parabolic distribution of the mesh size was simulated by keeping the two end points of the split line and adding a sample point half way with grading value equal to the global size. The grading at the sample points along the split line is then obtained as the minimum value from interpolation and the harmonic interpolation using the grading values at the end points and the midpoint along the line. This strategy proved to be valuable in cases where the split line connected two small features (i.e. end points have small grading values) so that the small feature size did not propagate along the split line.

4 SIZE CONTROL

There are various types of size control that a user may want, each one with different computational cost. Three types are defined, ranging from the lower cost to the higher cost: None, constant, curvature.

- No size control: The sample mesh is a coarse mesh formed by the boundary nodes and with no additional interior nodes. This approach is fast and can be used if the quality/distortion of the final mesh is not critical or if the space strategy used produces very little distortion (fig. 8b, 8c, 8d).
- Constant size control: The sample mesh is the initial mesh obtained without any account for the distortion. Distortion at sample points is computed leading to a 2D size interpolation domain that drives the 2D mesher. This is the preferred approach if the final mesh quality/size control is critical for a given constant size (fig. 8e, 8f, 8g)
- Curvature size control: The sample mesh is the same as in the constant size control case but this time the 3D

mesh size is computed as a function of the curvature (fig. 9b, 9c).

Although, we discuss a self-contained approach with the sample mesh internally generated, all the concepts are general and the sample mesh could be provided as input with the field of 3D sizes derived from an analysis, as is the case of adaptive meshing to a solution. With that data as input, the mesh generator will provide the desired mesh.

Smart size control [8] is another important variation on the size control that has not yet been implemented.

4.1 Curve Size Control

In order to adapt the boundary to the curvature we first need to compute the curve curvature. Two options are possible: line curvature or surface curvature.

4.1.1 Line curvature

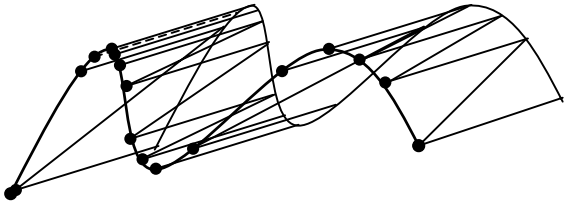

Fig 4. Poly-line formed by STL facet points
(ruled surface : line curvature =surface curvature)

The line curvature does not take into account the adjacent surface curvature. The curve has a poly-line representation formed by facet points (fig. 4). At each interior point, to the curve, the line curvature is computed as the inverse of the radius of the circle passing through 3 consecutive points. When these 3 points are collinear, the radius is set to infinity. At the curve end points, the curvature is computed by extrapolation. Furthermore, the minimum line curvature at end points is taken from all curves that share the vertex.

4.1.2 Surface curvature

In section 5 we will present discrete surface operators to evaluate curvature. The minimum radius of curvature is used at the boundary points to estimate the local surface curvature.

4.1.3 Line curvature versus surface curvature.

Either type of boundary curvature, line or surface, an average or a minimum can be chosen depending on the type of adaptation the user wants. The line curvature tends to highly refine small holes in flat areas. These can be very small geometry features compared to the mesh size that only need to be represented with a minimum of 3 to 4 points (fig. 6b). In all examples presented in this paper, only the surface curvature has been used.

4.1.4 Sampling refinement

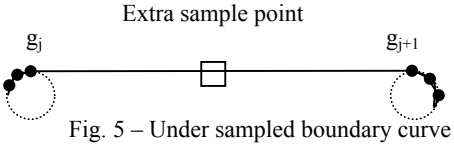

Fig. 5 – Under sampled boundary curve

For curvature size control, the facet point representation of boundaries near long flat regions close to fillets (fig. 5) need special attention. In a sense, these boundaries are "under sampled" and sample points need to be added to properly capture the flatness of the curve. The algorithm works as follows :

- Given a curve C, the arc length parameter location t_j of its sample points P_j and a global mesh size S_g
- Loop over segments $[t_j, t_{j+1}]$
  - Compute it length L_j
  - Segment grading average $\bar{g}_j = \frac{1}{2}\left(g_j + g_{j+1}\right)$
  - If $\lambda_g \bar{g}_j < S_g < \lambda_l L_j$
    - Add extra sample point at the mid point.
    - Assign grading value equal to S_g at this extra sample point.

The parameter λ_g is a constant representing the grading ratio between the local grading and the global size, while the parameter λ_l represents the inverse of the minimum number of intervals desired. The first part of the inequality states that the grading at the segment end points is very small compared to the global size. The second part of the inequality states that the segment length is large compared to the global size. By adding a point halfway, a parabolic node distribution will result. In the examples, values of $\lambda_g = 4.0$ and $\lambda_l = 1/3$ have been chosen.

5 SURFACE SIZE CONTROL

For a constant size control the 3D mesh size is a field of constant values. For a curvature size control the mesh size becomes function of the local surface curvature evaluated at the sample points.

5.1 Continuous surface operators

Given a surface S, the two principal curvatures K_1 and K_2 of the surface along the two orthogonal principal direction vectors $(\vec{e}_1, \vec{e}_2)$ are the extrema values of all the normal curvatures. The normal curvature $K_N(\alpha)$ to the

surface S at a point P with unit normal $\vec{N}$ along a unit tangent vector $\vec{e}_\alpha$ is defined as the line curvature of the curve formed by intersecting the plane $(P, \vec{N}, \vec{e}_\alpha)$ with the surface S. The mean curvature K_H is defined as the average of the normal curvature:

$$K_H = \frac{1}{2\pi}\int_0^{2\pi} K_N(\alpha)d\alpha \tag{8}$$

The Gaussian curvature K_G is defined as the product of the two principal curvatures:

$$K_G = K_1.K_2 \tag{9}$$

and the mean curvature is expressed as the average of the two principal curvatures :

$$K_H = \frac{1}{2}(K_1 + K_2) \tag{10}$$

5.2 Discrete surface operators

Given a triangle map, the first and second order attributes of the surface (normal vector, mean curvature K_H, Gaussian curvature K_G) can be approximated [9], [10].

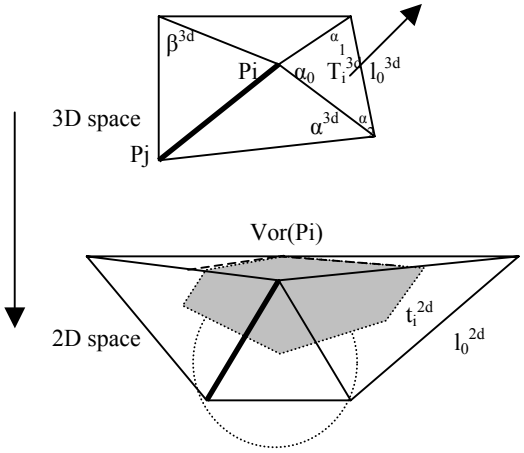

Fig. 5 Triangle map 3D to 2D space (Voronoi area with modification for obtuse triangles)

First we compute the discrete normal as an angle weighted average of the normals to the facets surrounding the point.

$$\vec{N}_i = \frac{\sum_{k\in S(i)} \alpha_k^{3D}\vec{N}_{T_k}}{\theta(P_i)} \tag{11}$$

$$\theta(P_i) = \sum_{k\in S(i)} \alpha_k^{3D} \tag{12}$$

with $\theta(P_i)$ representing the total vertex angle.

The Radius of curvature at a point P_i along the direction of the edge P_iP_j is:

$$\rho_{ij} = \frac{1}{2}\frac{\| P_iP_j \|^2}{< \vec{N}_i, P_iP_j >} \tag{13}$$

and the minimum radius of curvature at point Pi :

$$\rho_i = \min_j \rho_{ij}$$

At each point P one can compute the *vertex angle excess* $2\pi - \theta(P)$ that also represents the (total) Gauss curvature at an interior point:

$$\iint K_G dA = 2\pi - \theta(P) \tag{14}$$

The discrete gaussian curvature at point P can be approximated by:

$$K_G(P) = (2\pi - \theta(P))/Vor(P) \tag{15}$$

with $Vor(P)$ computed as the voronoi area at a point if all triangles are acute and for obtuse triangles the *containment circle* is used instead of the circumscribed circle criteria (i.e. instead of joining adjacent edges midpoints to the center of the circumscribed circle they are joined to the midpoint of the (opposite) longest edge (fig. 5).

The discrete normal curvature is derived from the formula

$$\iint 2K_H(P).\vec{N}(P)dA = \frac{1}{2}\sum_{j\in S(i)}(\cot\alpha_{ij}^{3d} + \cot\beta_{ij}^{3d})(\vec{P}_i - \vec{P}_j) \tag{16}$$

with α_{ij} and β_{ij} the two angles opposite to the edge (P_i, P_j) in the two triangles sharing the edge (see fig. 5).

The discrete mean curvature normal is given by :

$$\vec{K}_H(P_i) = \frac{1}{2\times Vor(P_i)}\sum_{j\in S(i)}(\cot\alpha_{ij}^{3d} + \cot\beta_{ij}^{3d})(\vec{P}_i - \vec{P}_j) \tag{17}$$

and the approximation of the normal curvature is obtained as:

$$K_H(P_i) = \frac{1}{2}\| \vec{K}_H(P_i) \| \tag{18}$$

All the above formulas are the discrete counterparts of the continuous first and second order attributes of the surface.

In [11] a measure of the deformation of a triangle between 2D and 3D space is proposed :

$$Def(T_i^{3D} \rightarrow t_i^{2D}) = \frac{\cot\alpha_{io}^{3D} \| l_{i0}^{2D} \|^2 + \cot\alpha_{i1}^{3D} \| l_{i1}^{2D} \|^2 + \cot\alpha_{i2}^{3D} \| l_{i2}^{2D} \|^2}{2 \times Area(t_i^{2D})}$$

By summing over all the triangles, this formula provides a measure of the total distortion induced by the mapping used in the triangle map between the 2D and 3D space. The global measure could be used as criteria to select the space development strategy with least distortion and/or to improve an initial parameter domain by minimization of the global distortion measure. Currently the curvature adaptation strategy only considers the minimum radius of curvature, but experimentation with other criteria is underway.

The mapping between the discrete curvature and the 3D size is as follows [10]:

- Given ε a percent deviation to the original geometry.
- Given a global size S_{global}
- Compute the constant $\gamma = \sqrt{\varepsilon(1-\varepsilon)}$
- Look up in the triangle map for the minimum radius of curvature, ρ_i.
- Compute local 3D size: $S_i^{3D} = \rho_i \times \gamma$
- Bound S_i^{3D},

$$\frac{1}{dLenRatio} \times S_{global} < S_i^{3D} < S_{global}$$

A value of $dLenRatio = 10$ was chosen to control the minimum size allowed during curvature adaptation.

6 TRIANGLE MAP

The triangle map keeps a map between the 3D mesh and the 2D mesh and also provides a wealth of information about the surface. There are two triangle maps that we use. The STL triangle map (fig. 7a, 7b) and the sample mesh triangle map. One starts with a given STL of a surface to mesh (fig. 7a). The node coordinates in 3D space and the mesh connectivity are stored in the triangle map. Using a space development strategy (in all the examples presented a flattening strategy is used), the 2D parameter domain (fig. 7e) is created and the map x(u,v), y(u,v), z(u,v) is stored for the facet points. The 2D parameter domain is an interpolation domain for the mapping between the 2D space and the 3D space. Next, all the discrete operators are computed as well as the distortion between the 2D space and the 3D space. The STL triangle map is always used to map the nodes back to 3D space. Another use of the STL triangle map is during the boundary node generation with curvature size control.

As mentioned in section 4.1, we use the surface curvature that we obtain directly in the look up table of the triangle map. The boundary nodes were generated in 3D space and they need to be mapped to their corresponding 2D space value. To do so one could perform an exhaustive 3D point in triangle location. Instead, as the curve is represented by a poly-line of facet points, we store the parameter location of the facet points along the curve. For a mesh point generated along the boundary curve at the parameter location t, we find the facet point interval [ti, ti+1] that contains t and use a linear interpolation to find the corresponding 2D parameter location (u,v). The 3D boundary node loop is then mapped to the 2D plane. The grading value at the boundary nodes is computed as an average of the two adjacent edges length at the points. This gives us the "real" 2D size that already accounts for distortion.

The sample mesh is generated in 2D space using the TQM meshing algorithm with the desired mesh size. At this stage, the size map has not been created yet. The grading values along the split line are computed using the piecewise harmonic interpolation (equation 1). The sample mesh is mainly uniform (unless we are using no size control) and provides a sampling field of interest for the given mesh size. The 2D mesh is transformed back to 3D space using the STL triangle map, and a sample mesh triangle map is created. This latter triangle map provides a look up table to compute for each point in the sample mesh, its distortion, its surface curvature etc …The data (distortion, curvature etc…) is computed at once for the whole mesh and stored in the triangle map.

The boundary discretisation of the sample mesh and the final mesh are identical and need not be regenerated. The size map is created and will be used as a size interpolation field for the final mesh.

7 SIZE MAP

The size map is a combination of the size control and the triangle map. The size control provides the 3D size variation on the surface for the sample mesh while the triangle map provides the size distortion. The size map combines both data into one single value. For example, for a constant 3D size mesh, the size control has a field of constant values and only the distortion factor varies at each point of the sample mesh. The 2D scaled mesh size is the product.

8 EXAMPLES

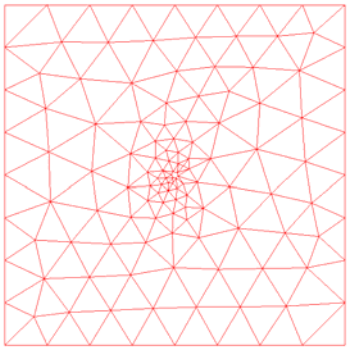

Fig. 6a - Small hole

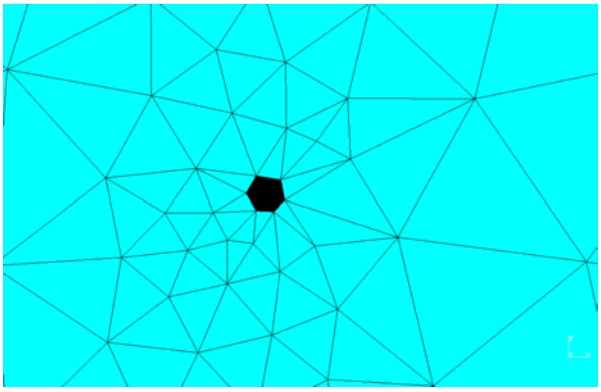

Fig. 6b - Small hole zoom

Figure 6 : smooth size transition for small features.

8.1 Smooth transition from small holes to large constant size

Figure 6a represents the final mesh for a square with a small hole. The diameter of the hole is 1 while the square size is 100. A mesh size of 12.5 has been used. The hole has been represented by 5 elements and the mesh transitions smoothly from the small to the large size (fig. 6b).

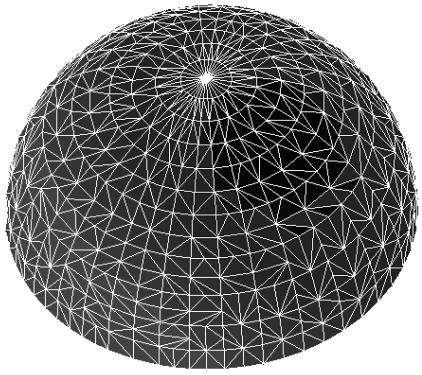

Fig. 7 a –3D STL of an hemisphere

8.2 Developed space distortion comparison.

Figures 7 (b,c,d,e) illustrates the advantage of the flattening technique [5] over the Maximum Area Plane (M.A.P.) projection used in I-DEAS. Figure 7a represents the STL of half a sphere. Figure 7b represents the 2D parameter space resulting from the projection technique. Notice the high distortion along the boundary where triangles have been "squashed". Figure 7c represents on top the 2D final mesh and on the bottom the corresponding 3D mesh obtained with the option of constant size control. Highly distorted elements are generated along the boundary. Clearly, a projection technique is not satisfactory in local areas where the normal to the surface is orthogonal to the direction of projection and a small change $d\varepsilon$ in the 2D space tends to infinity in the 3D space.

Figure 7d represents the flattened STL mesh. This time the lengths have been preserved along the boundary and the highest distortion seems to occur around the pole. The final mesh with constant size control is presented in figure 7e. The flattening space strategy produced a more isometric mapping leading to the good mesh quality.

Projections techniques are computationally inexpensive but they are restricted to domains that can be projected and therefore work well with low curvature domains. On the other hand, flattening techniques, depending on the type of domain at hand, result in more isometric mapping (works well for developable domain independent of the curvature) but are in general more computationally expensive. They also have their own limitations (cannot flatten a closed surface without cutting it), but they are less stringent than projections techniques.

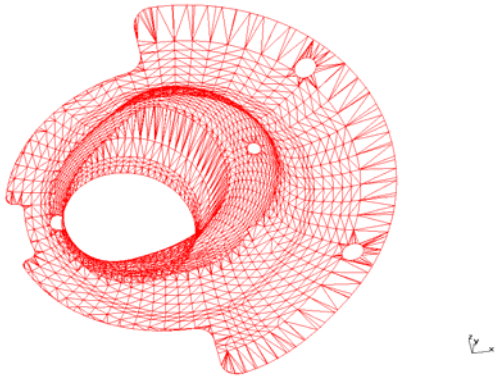

Fig. 8a – 3D STL of a damper

8.3 Size Control comparisons.

This example illustrates the results obtained with various types of size control. The mesh size is 5 in all cases.

Figure 8.a represents the initial STL. In figure 8b and 8e the sample meshes for no size control and constant size control options are presented. Figures 8c and 8f provide a comparison of the resulting 2D final meshes for respectively no size control and constant size control. The mesh in fig. 8f has a smoother variation of the the size than the one in fig. 8c, due to a richer sampling of the curvature variation and therefore a richer interpolation domain for the distortion scaling factor..

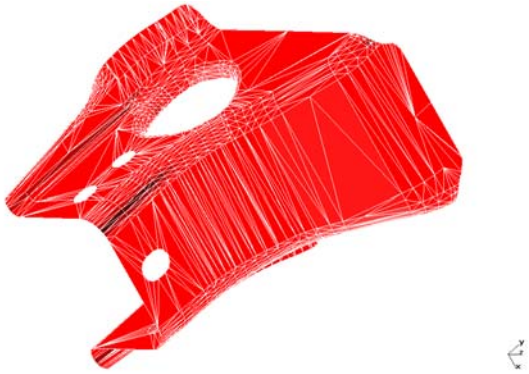

Fig. 9a – 3D STL of a bracket

8.4 Curvature adaptation.

Figure 9a represents a bracket with fairly complex curvature patterns. Figure 9b demonstrates how the 2D mesher is able to accurately adapt to the curvature pattern and figure 9c shows that the refinement, when mapped back in 3D space did occur in the correct locations. Notice also, the 1D boundary curvature adaptation and how the holes in flat regions were not refined as the surface curvature, not the line curvature, was used in these examples.

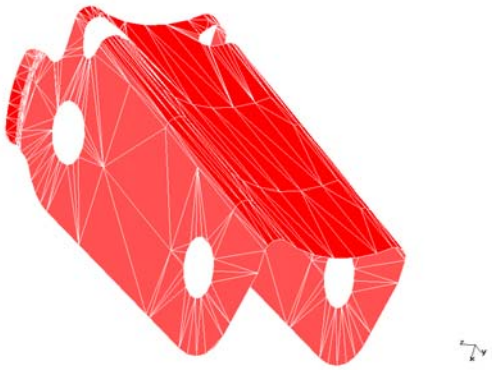

Fig. 10a- 3D STL of another bracket

Figure 10a, 10b and 10c is another example of curvature adaptation. Notice in figure 10b that the 2D mesher accurately captured the high curvature areas and started to pick up the lower curvature of the rear flaps both on the boundary and the interior.

9 CONCLUSION

Recent progress and extensions to increase TQM flexibility to handle large variations in mesh size all across a surface have been presented and demonstrated.

An approach that uses the surface STL data as sample points for the boundary discretisation and automatically generates a sample mesh for the interior has also been presented. A natural way of getting the sample mesh, based on the final mesh global size, proved to be a good sampling strategy. The price to pay for the additional quality is the cost of meshing the surface twice. It is the user's choice whether to incur this extra cost.

The curvature adaptation presented is robust and transitions smoothly between high and low regions of curvature.

Finally, we have tried to isolate independent concepts such as size control, triangle map and size map that put together provide tremendous flexibility.

This work is still at a preliminary stage with emphasis on flexibility and surface mesh quality. Future work should include smoothing techniques that are "adaptation preserving", a study of the viability of using the STL as sample mesh, smart sizing, and study and development of "best practices/strategies" to get a good quality surface mesh at lowest cost.

10 ACKNOWLEDGMENTS

The author expresses his sincere thanks to his colleagues at EDS/PLM solutions for their active support. In particular, Michael Hancock, Nilanjan Mukherjee, Hui Xiao, Radhika Vurputoor and Kirk Beatty for their active involvement.

11 REFERENCES.

[1] A.J.G. Schoof, L.H.Th. M. Van Beukering and M.L.C. Sluiter, "A general purpose two-dimensional mesh generator", *Advances in Engineering Software*, Vol 1(3) pp.131-136 (1979)

[2] J.R. Tristano, S.J. Owen and S.A. Canann, "Advancing front surface mesh generation in parametric space using a Riemannian surface definition",Proc. 7th IMR, pp 429-445 (1998).

[3] R. Lohner, "Regridding surface triangulations", *Journal of Computational Physics*, Vol 126, pp.1-10 (1996)

[4] A. Sheffer, E. de Sturler, "Surface parametrization for meshing by triangulation flattening", 9th *IMR*, pp.161-172 (2000).

[5] E.C.Sherbrooke, M.R. Lauer and D.C. Gossard,"Membrane surfacing : A triangulated G^1 representation for feature-based design", NTI technical report 00-1, (2000)

[6] J.A.Talbert and A.R. Parkinson, "Development of an automatic two-dimensional finite element mesh generator using quadrilateral elements and bezier curve boundary definition", *IJNME*, Vol 29, pp. 1551-1567 (1990).

[7] J. Sarrate and A. Huerta, "Efficient unstructures quadrilateral mesh generation", *IJNME*, Vol 49, pp. 1327-1350 (2000).

[8] A. Cunha,.S.A Canann and S. Saigal, "Automatic boundary sizing for 2D and 3D meshes", *Trends in*

unstructured mesh generation, ASME, AMD-Vol 220, pp.65-72 (1997)

[9] M. Meyer, M. Desbrun, P. Schroeder, A.H. Barr, "Discrete differential geometry operators for triangulated 2-manifold", VisMath (2002)

[10] P.J. Frey and H. Borouchaki, "Criteres geometriques pour l'evaluation des triangulations de surfaces", Rapport de recherche *INRIA*, N^0 2951 (1996)

[11] K. Hormann, U. Labsik and G. Greiner, "Remeshing triangulated surfaces with optimal parameterizations", *CAD*, Vol 33, pp.779-788 (2001).

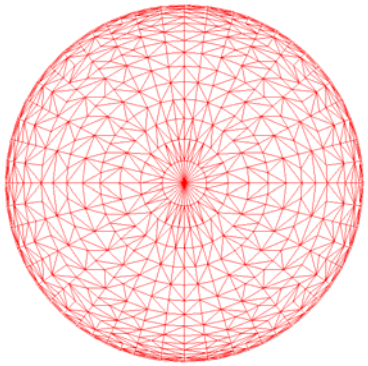

Fig. 7b – 2D projected STL using Maximum Area Plane (M.A.P.)

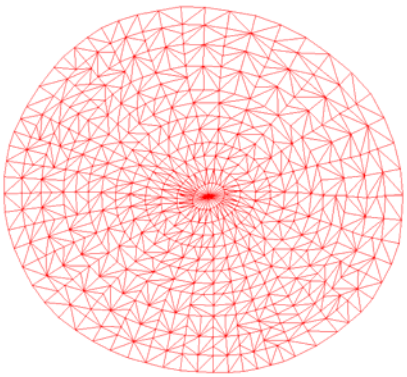

Fig. 7d – 2D flattened STL.

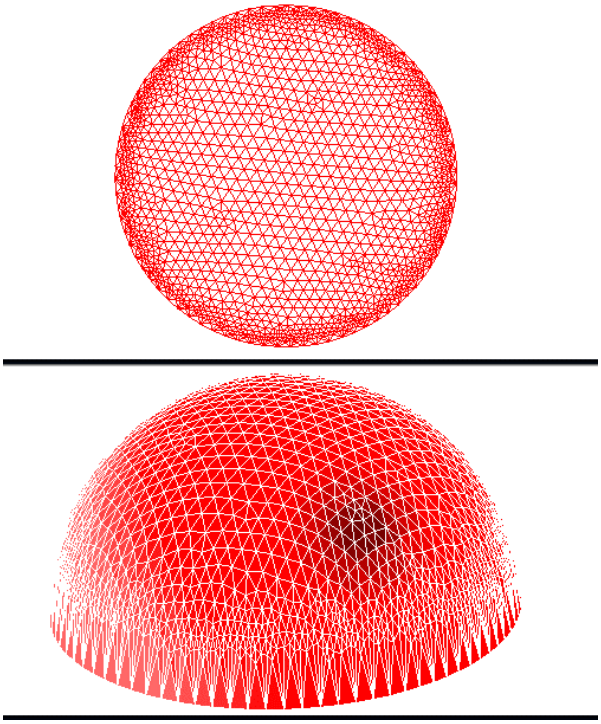

Fig. 7c – M.A.P., constant size control. Top: 2D final mesh. Bottom: 3D final mesh

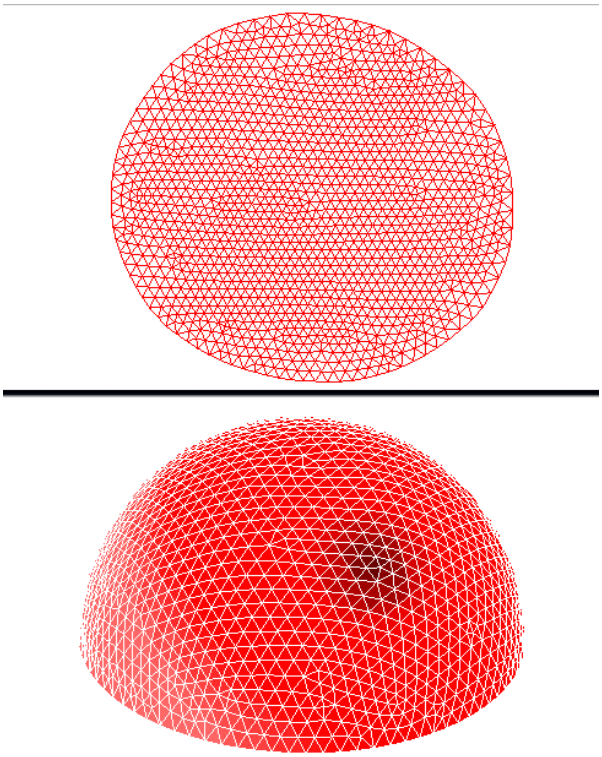

Figure 7e - Flattening, constant size control. Top: 2D final mesh. Bottom: 3D final mesh

Figure 7 : Developed space distortion comparison between (left) Maximum Area Plane and (right) flattening.

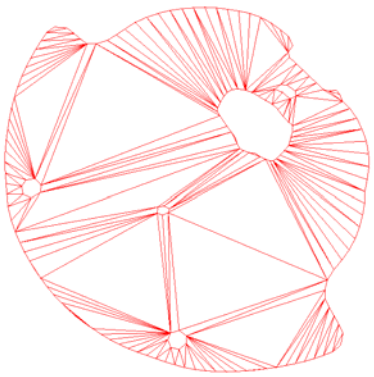

Fig. 8b -2D sample mesh, size control none.

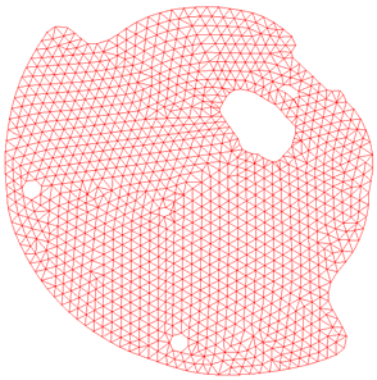

Figure 8e - 2D sample mesh , size control constant

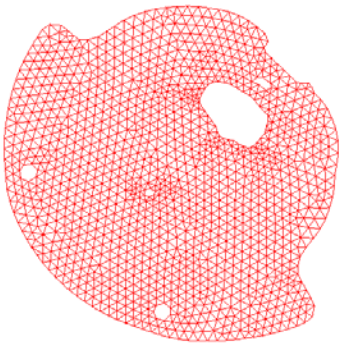

Fig. 8c – 2D final mesh, size control none

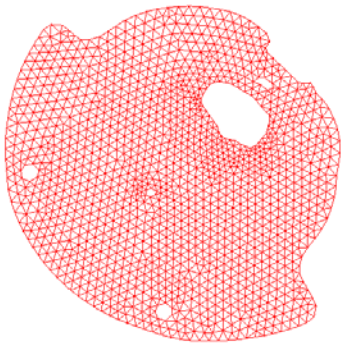

Fig. 8f - 2D final mesh, size control constant

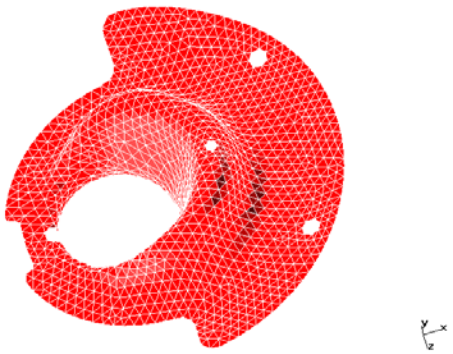

Fig. 8d –3D final mesh , size control none.

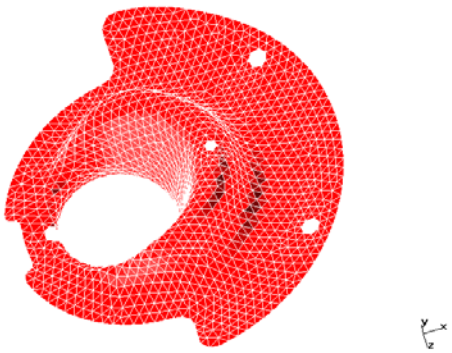

Fig. 8g - 3D final mesh, size control constant

Figure 8 – Size control comparison between (left) size control none and (right) size control constant.

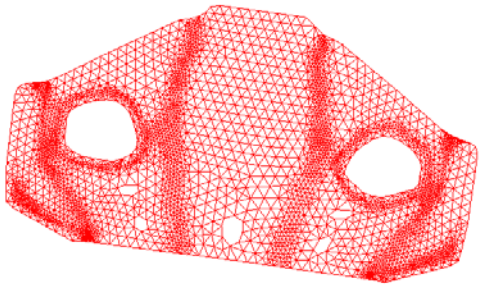

Fig. 9b - 2D final mesh, curvature size control

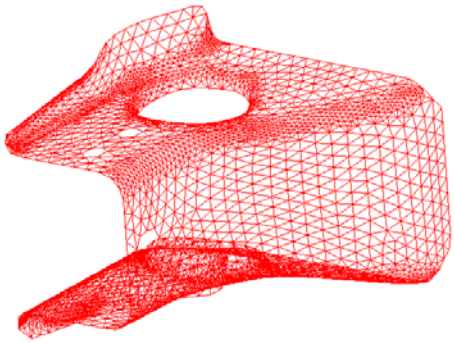

Fig. 9c - 3D final mesh, curvature size control

Figure 9 – Curvature adaptation of a bracket with complex curvature patterns

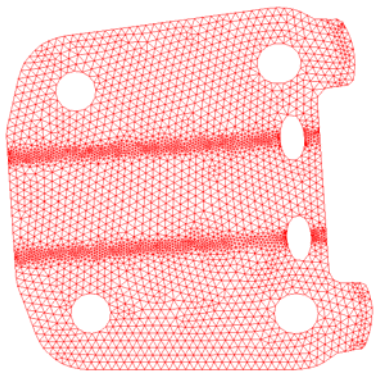

Figure 10b – 2D final mesh

Figure 10c – 3D final mesh

Figure 10 – Curvature adaptation of a bracket around high and low areas of curvature.