A Hierarchical Adaptive Mesh Generation Strategy for Parametric Surfaces Based on Tree Structures

Daniel Siqueira, Joaquim Bento Cavalcante-Neto, Creto A. Vidal and Romildo J. da Silva
Federal University of Ceara
Fortaleza - CE, Brazil
Contacts: {siqueira, joaquimb, cvidal}@lia.ufc.br, rjsdusk@gmail.com

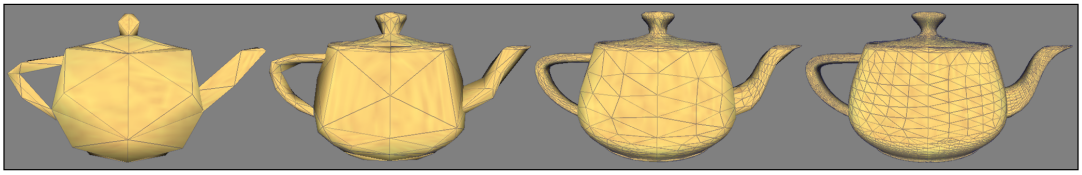

Figure 1. The Utah teapot mesh generated by this adaptive method.

***Abstract*—This work presents a hierarchical adaptive mesh generation strategy for parametric surfaces based on tree structures. The error measured between the analytical and discrete curvatures guides the adaptive process. While the analytical curvature is an mathematical representation that models the domain, the discrete curvature is an approximation of that curvature and depends directly on the used mesh configuration. The presented strategy possess the following properties: it is able to refine and coarsen regions of the mesh; it ensures compatibility between neighbouring regions; it considers the contribution of the local error measures to ensure good global mesh quality and it works with any type of parametric surface geometry, since the process is performed in the parametric space.**

***Keywords*-adaptivity; parametric surfaces; surface meshes**

I. Introduction

Triangular meshes are a spatial representation of great importance. They are widely used in various areas, and allow the manipulation and visualization of complex surfaces with relative ease. The meshing of surfaces is a subject that has been studied since the 70's and is still a topic of great interest in engineering and computer graphics.

Among the main aspects taken into account in the assessment of a technique for generating surface meshes, two important ones are the processing time for generation and quality of the mesh generated. The processing time depends on the desired discretization, which is represented by the number of elements present in different regions of the model. The quality of the mesh, on the other hand, takes into account the quantity, distribution and shape of its elements [1], [2]. This quality is essential for many different applications: in computer graphics, for example, the quality directly influences the visual aspect of the model.

Depending on the application, it is necessary to save the amount of data used in the mesh without losing its quality. To achieve this goal, the use of adaptive meshes is an interesting option. These meshes have a higher concentration of elements in regions of greater curvature and a lower concentration of them in regions of lower curvature. The mesh adaptation improves the size, organization and arrangement of elements, facilitating the storage or transmission of the mesh and processes such as rendering and simulations.

The techniques for adaptive meshes attempt to generate the mesh according to some criterion adopted, modifying the mesh where necessary. Among the most common changes there are: resizing, insertion and removal of elements. These changes are applied until a stopping criterion is reached. The criteria adopted for the modification must not be computationally expensive. In this work, an approach of low cost and good convergence is adopted, which is the use of geometric criteria [3], [4], [5], [6], [7]. Here, the curvature is considered as an important parameter in determining the density distribution of elements. The closer the discrete curvature measured in the mesh is to the analytical curvature, the mesh will be more faithful to the geometric characteristics of the surface.

Following this concept, this paper presents a strategy for generating hierarchical adaptive meshes of parametric surfaces based on tree structure, adding the advantages of two classical approaches: the advancing front and Delaunay triangulation. It is a *posteriori* strategy, i.e., the user provides a geometric definition for the model, and from this geometry, the strategy constructs a first approximation with a very coarse mesh. By each interaction, a new mesh, closer to the mathematical definition of the surface, is generated. The strategy presented seeks to ensure the following properties: i) to be able to refine and coarsen regions of the mesh, ii) to ensure a smooth transition between regions of the mesh with higher and lower refinement, iii) to ensure compatibility

between different regions, iv) to consider the contribution of the local error to measure the overall error in order to ensure a good quality for the entire mesh.

This paper is organized into 5 sections. Section 2 presents some related work, understanding some operators to calculate the curvature and some existing techniques for generating surface meshes using curvature control. Section 3 presents an overview of the strategy, explaining each stage of the adaptive process. This Section also shows the mathematical concepts used for the calculation of curvatures. These curvatures are used to calculate the estimation error to guide the adaptive process. Section 4 presents some results achieved by generating meshes using the strategy. The results show its effectiveness and robustness in dealing with various levels of curvature. In Section 5, the conclusions about the strategy are presented.

II. Related works

From a purely theoretical point of view, triangular meshes have no curvature, since all faces are flat and the curvature is not properly defined along the edges and vertices, because the surface has no continuity C^2 defined for edges and vertices. But, by taking into account that a triangular mesh is an approximation of the solid surface, it is possible to estimate the curvatures by the information in the mesh.

The Gaussian curvature is one of the most known methods to estimate the curvature of a surface. This measure has been extensively used in various fields such as image processing, computer aided geometric design, robotics, computer graphics and many others. Magid et al. [7] present some of the most widely used algorithms for calculating the Gaussian and mean curvatures. In their work, such algorithms are presented, analysed and tested to determine what is the best alternative for the calculation of curvatures. Through comparative tests, the authors conclude that the best algorithms to calculate the Gaussian curvature are those that employ Gauss-Bonnet scheme. Xu [6] analyses the convergence of the scheme of Gauss-Bonnet, which was presented in [7]. The convergence of discretization scheme was considered by Meek and Walton [4].

Kim et al. [3] presents an error metric in order to generate several different mesh refinements to use in applications that involve levels of detail. This metric uses the method of discrete curvature as a criterion for simplification. The operators used follows the scheme of Gauss-Bonnet and can be used to calculate the curvatures for both the vertices located within the mesh and for those located on a border. Because of this feature, and for following the scheme of Gauss-Bonnet, such operators have been chosen for the calculation of discrete curvatures in the presented work.

Lau and Lo [8], [9] present a technique for generating finite element meshes using the curvature as a measure to determine the size of the elements and guide the discretization. In the first work [8], it is proposed a scheme for automatic generation of triangular meshes with arbitrary distribution of elements on curved surfaces and without the use of a parametric space. The second work [9] brings a similar approach, but proposes more control over the curvature.

Seibold [10] presents a method for generating a mesh surfaces using the parametric space and Delaunay triangulation. An initial set of points may be created in the parametric space, which can be random or entered manually. While this set of points is not mandatory, the starting point may be the parametric space square divided into two triangles, creating an initial very coarse mesh. The mesh error is measured and if this error is less than a certain precision ε and the number of triangles is less than a maximum, the triangle of greatest error is split at the point of greatest error according with the Delaunay criteria. The error of the mesh is the mainly point discussed in the article. The authors present some techniques of measurement error: the angle between the normals of two adjacent triangles, an analytical error and a geometric one. None of these measures takes into account the curvature of the surface and are based on distances between a point on the surface and its corresponding parametric space. Such approaches may have problems as the distance between points on the surface and its parametric representation may not reflect the real geometry of the model or indicate an error greater than the actual.

Dyn et al. [11] proposes an algorithm to obtain an optimal mesh triangulated using the operation of swapping edges in a sequential manner in order to reduce a cost function that measures the quality of the mesh. The cost function used by the authors is developed to take into account the curvature of the surface.

Miranda and Martha [12] describe an algorithm to generate triangulated meshes of finite elements in arbitrary surfaces with high curvature. The mesh is generated in the parametric space and mapped to the 3D space. In this work, the curvature taken into account is not calculated, since its values are given as an entry to the process.

The majority of these works and other works presented in the literature are able to refine regions of the mesh but are not able to coarsen regions of the mesh where there are more elements than necessary. They also do not ensure mesh compatibility between regions since they do not refine the curves independently from the domain. This can be very important in many applications and essential in applications where compatible meshes are required. Finally, although some of these works are in the parametric space, they do not allow the mesh to be generated for different mathematical description of the patches such as Hermite, Bézier and so on.

III. Adaptive strategy

The adaptive strategy presented herein assumes that the surface to be meshed is composed of parametric patches, such as Coons patches, each of them bounded by parametric

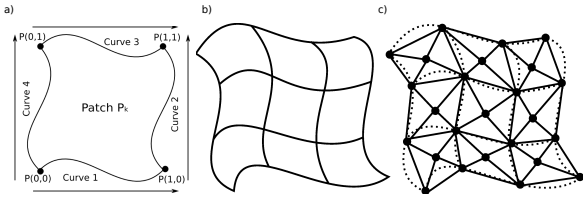

Figure 2. Patch definition.

curves. Although this kind of patch is generally bounded by four parametric curves, the method can handle any kind of parametric patch, since what is relevant is the parametric coordinates.

It is also assumed an initial mesh just to start the procedure. The strategy consists of two phases per iteration: first, given the geometric models of the patches and curves, as well as the initial mesh for the current iteration, the mesh quality measure based on error of curvature is computed; second, the mesh is modified by adding more elements in certain regions, and removing elements from other ones. The iterative process stops when a predefined global quality criterion is achieved. In phase 2, the curves are rediscretized first and, only then, the patches are rediscretized. Each part of the adaptive strategy is discussed in details in the sequel.

A. Initial model

The mathematical description of the patches together with the description of their boundary curves define the geometry, which is used in all steps of the adaptive strategy. The geometry description and an initial mesh compose the initial model that is used to start the adaptive process.

The geometric description is hierarchical: the descriptions of the curves come first, followed by the patches descriptions. The model depicted in Figure 2 (a) describes a patch P_k and its curves C_i . This model is used throughout this work to illustrate the adaptive strategy.

The initial mesh of all patches compose the initial mesh of the model, and it is a first approximation of the desired mesh. If P_k is a patch and C_i represents its boundary curves, the generation of the initial mesh, M_k, starts with the discretization of C_i into segments. Thus, the initial mesh is generated considering these segments, ensuring compatibility of meshes at common borders of neighbouring patches. The number of elements, forming mesh M_k, depends on the discretization of P_k's boundary curves. In Figure 2 (b), the discretization of the boundary curves in three segments each determines the discretization of the patch into nine regions. The subdivisions of curves and patches are performed in parametric space and mapped into the three-dimensional space. It is important to mention, however, that this mesh is generated if no initial mesh is given, which is used instead. This makes the process completely generic for any given initial mesh.

B. Mesh quality

The quality of every mesh generated during the adaptive process must be evaluated, according with an error criterion. Herein, a geometric criterion is adopted. It computes, at every mesh vertex, the error between the surface's curvatures evaluated both analytically and approximately. If the overall error distribution approaches zero in the average sense, the mesh quality is considered good, and the adaptation process stops. The local error indicates whether the mesh should be refined or coarsened at a particular region.

The analytic curvatures adopted to compute the error measures are the mean curvature

$$H = \frac{1}{2}(k_{min} + k_{max}) \tag{1}$$

and the Gaussian curvature

$$K = (k_{min}.k_{max}) \tag{2}$$

where k_{min} and k_{max} are the principal curvatures. The computation of H and K for a bi-parametric surface Q(u,v) can be found in [13] and elsewhere.

The discrete curvatures, unlike the computations of the analytical curvatures, which use the mathematical formulation of the patch, are evaluated based on the local information about the mesh. That is, the discrete curvatures are computed at a given mesh vertex and are based on the adjacent triangles' configurations. The discrete Gaussian and mean curvatures used in this work are computed with the discrete curvature operators presented in [3]. The discrete Gaussian curvature K for a inner vertex (see Figure 3a) is expressed as

$$K = \frac{2\pi - \sum \phi_i}{\frac{1}{3}A} \tag{3}$$

where A is a sum of each triangles area, and ϕ_i denotes the angle at a vertex.

For a boundary vertex (see Figure 3b), the Gaussian curvature is defined by the following equation

$$K = \frac{\pi - \sum \phi_i}{\frac{1}{3}A} \tag{4}$$

where A again represents the sum of each triangles area and ϕ_i the angle at a vertex.

The discrete mean curvature *H* is given by

$$H = \frac{\Sigma m(e_i)}{\frac{1}{3}A} \tag{5}$$

where e_i represents an edge of a vertex and $m(e_i)$ is a function of an edge e_i that return the angle between two adjacent surface normals (see Figure 4).

$$m(e_i) = \begin{cases} \gamma, \, if \; the \; adj. \; faces \; are \; convex \\ 0, \, if \; the \; adj. \; faces \; are \; plane \\ -\gamma, \, if \; the \; adj. \; faces \; are \; concave \end{cases} \tag{6}$$

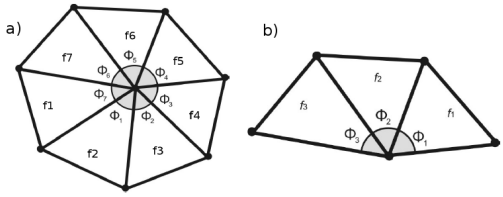

Figure 3. (a)The Gaussian curvature of an inner vertex. (b)The Gaussian curvature of a boundary vertex.

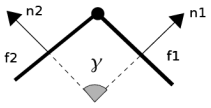

Figure 4. The angle γ for the mean curvature.

Error estimation based on the curvatures: The adaptive process consists of curves and patches rediscretization. This is performed based on the error estimation of the discrete curvature relative to the analytical curvature evaluated at the vertices. If the computed error indicates a high discrepancy between these curvatures, it suggests that the mesh be locally refined or coarsened, based on a size parameter *h*.

Table I illustrates the possible scenarios and the determination of the new size parameter. k_a and k_d represent the analytic and discrete curvatures. The Gaussian curvature is the measure used to perform the error estimation, but when the Gaussian curvature is zero, the mean curvature is used instead.

In the first scenario, k_a is approximately equal to k_d. If these values are not close to zero ($k_a \nrightarrow 0$), the mesh captures well the local curvature of the surface and no refinement is necessary ($h_{new} = h_{old}$). However, if the mean curvatures k_a is close to zero ($ka \rightarrow 0$), the surface is locally planar and a coarsening of the mesh may be possible ($h_{new} = h_{old} * f,\ f > 1$).

In the other scenarios, the discrepancy between k_a and k_d is large. Then, when the mean curvature k_a is close to zero ($k_a \rightarrow 0$), the mesh is too coarse to capture the local flatness of the surface and should be refined. When k_a is not close to zero ($k_a \nrightarrow 0$), the mesh is again too coarse to capture the correct curvature and should be refined.

The factor *f* used for rediscretization is determinant to the rate of convergence of the process. Which value of *f* to use is empirical at the moment and deserves more study in the future. In this work, a factor of 4 was used in the examples. This value experimentally showed good results. However, what is more important is that the strategy converges for many ranges of *f*.

Table I
ERROR AND ESTIMATION OF ELEMENT SIZES.

$k_a \approx k_d$	$k_a \rightarrow 0$	$h_{new} = h_{old} * f$	coarsening
$(\frac{k_a}{k_d} \rightarrow 1)$	$k_a \nrightarrow 0$	$h_{new} = h_{old}$	stop
$k_a \gg k_d$	$k_a \rightarrow 0$	$h_{new} = h_{old}/f$	refinement
	$k_a \nrightarrow 0$	$h_{new} = h_{old}/f$	refinement
$k_a \ll k_d$	$k_a \rightarrow 0$	$h_{new} = h_{old}/f$	refinement
	$k_a \nrightarrow 0$	$h_{new} = h_{old}/f$	refinement

Global error estimate: Although the rediscretization is based on local errors from the curvatures, it is necessary a global measure to guide the whole iterative process. When a global quality measure is reached, the process stops. This measure implies the use of a global error estimate, η_{global}, which, in this work, is evaluated as follows

$$\eta_{global} = \frac{\sum_{j=1}^{N_v} \eta_j}{N_v} \tag{7}$$

where N_v is the number of vertices of the whole mesh and η_j is the absolute value of the relative discrepancy between the analytical and the discrete curvatures at vertex *j*, computed as

$$\eta_j = \frac{|k_a - k_d|}{k_a}. \tag{8}$$

The mesh has good quality when $\eta_{global} < \epsilon$. What value of ϵ is considered reasonable is a matter of judgment. Therefore, in this work, no suggestion is made about what value of ϵ to use. It is simply shown that the errors decrease when the mesh is improved, indicating convergence of the iterative process.

C. Curve adaptation

The first step of the presented strategy requires that the boundary curves of every patch be rediscretized prior to, and independent of, the rediscretization of the domain. It turns out that this is one of the main advantages of the strategy, since it leads to a more regular discretization of the patches' boundaries, consistent with the geometric characteristics of the surface near these curves. This is also important to insurance of mesh compatibility between two adjacent patches. The new discretization of the complete set of boundary curves defines the basis for the rediscretization of the domain, generating a new mesh for the model.

The boundary curves' rediscretization procedure is a one-dimensional version of the procedure based on a quaternary tree (quadtree) used to refine the domain [14]. It employs a recursive spatial enumeration technique, associated with a binary tree data structure, and adopts a criterion of curvature error to refine or coarsen a curve discretization, taking into account the curvature characteristics of its adjacent surfaces. Thus, at a given vertex of a curve's approximation polygon, discrete and analytic curvatures are computed. The discrete curvature considers all the triangles adjacent to the vertex that lie on the neighbouring surfaces. The analytic curvature, on the other hand, considers the maximum of the curvatures computed for the surfaces sharing the curve. A curvature error measure guides the determination of new size of mesh triangles adjacent to that vertex.

The rediscretization of each curve is divided into three phases, as described in the following for a generic curve C_i.

Binary tree initialization: The first phase consists of initializing the binary tree that will guide the curve rediscretization. This tree is initialized with the minimum and maximum parametric coordinates (0, 1) of the non-discretized curve. This is done to allow a generic refinement procedure for any type of curve. The depth of the tree is initialized as zero, since the tree has only one level at this point.

Binary tree rediscretization: Each curve keeps the set of old adjacent vertices, which is set up when the initial mesh is generated and is updated after a new mesh is determined at each step. A curve also holds the set of new vertices that are generated during the current phase. Hence, these two sets have to co-exist until a new mesh has been committed. These sets are sorted based on the parametrization of the curve.

The binary tree rediscretization of a curve (Algorithm 1) starts by traversing its sorted set of vertices from the old discretization. The process begins from the minimum and maximum parametric coordinates (0, 1). Every two consecutive vertices represent a segment of the curve. Each leaf of the tree stores the minimum and maximum parametric coordinates of the segment represented by that leaf.

Algorithm 1: Construction of the binary tree.

Initialize the binary tree with the root node empty ;
Assign min, max parametric coordinates (0,1) to the root;
Compute the length of the curve, L_{curve};
for *each segment defined by the old set of curve's vertices,* S_k, $k \leftarrow 1$ **to** n_{seg} **do**
    Compute the segment's length in 3D space, L_{seg_k};
    $h_{old} \leftarrow L_{seg_k}$;
    Compute the segment's center in 3D space, C_{seg_k};
    Compute k_a *at* C_{seg_k};
    Compute k_d *as average of the* k_d*'s at the enfpoints,* $k_d = \frac{1}{2}.(^{k-1}k_d + ^{k1}K_d)$;
    Compute h_{new} *according to the Table I*;
    // $h_{par} \in [0, 1]$
    Compute $h_{par} = \frac{h_{new}}{L_{curve}}$;
    Determine the parametrer u_k *corresponding to* C_{seg_k};
    Determine in wich cell of the tree u_k *is located*;
    while *size of the cell* $> h_{par}$ **do**
        Subdivide the cell in two children;
        Increment the depth of the tree;
        Determine in wich cell of the tree u_k *is located*;

Both h_{par} and the vertex on the center of a curve's segment have to be considered in the parametric space because the curve is treated in this space.

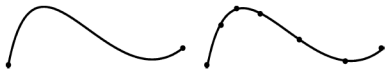

Figure 5. Refinement of a curve C_i.

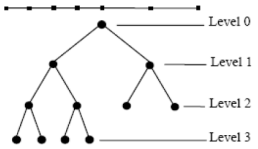

Figure 6. Tree for the curve C_i of Figure 5.

After the curve segments' traversal, the resulting binary tree reflects the refinement according to the error in the curvatures computed at the vertices of the polygon approximation of curve C_i. Each leaf of the tree will generate a new segment of the curve's approximation polygonal line. These segments will be the sides of elements adjacent to the curve when the domain is rediscretized. Figure 5 illustrates an example of refinement of a curve C_i and Figure 6 illustrates the corresponding binary tree.

Update of the discretization of the curve based on the binary tree: When the binary tree of curve C_i is finally redefined, the new discretization is included in the curve (Algorithm 2) and at the end of this process, the new curve discretization is obtained, in a sorted way. The new set of vertices is kept and the old set of vertices from the previous step is released.

Algorithm 2: Curve discretization.

Compute the 3D coordinates associated with u = 0;
Include these coordinates in the curve's structure;
for *each leaf of the binary tree* **do**
    Obtain the max parametric coordinate;
    Compute the corresponding 3D coordinates;
    Include these coordinates in the curve's structure;

D. Mesh adaptation

After the discretization of the boundary curves, the next phase of the adaptive process generates a new mesh on the domain, using a procedure based on quadtree. However, in this work the quadtree is not responsible for the mesh generation in the whole domain. A strip close to the patch's boundary is left out to be discretized by a Delaunay-based technique. The procedure ensures that the mesh on the interior of a patch agrees with the discretization of its boundary curves. This is important because meshes on adjacent patches can be combined.

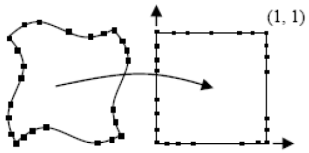

Figure 7. Patch's curves discretization.

The mesh generation by the quadtree procedure is based on the work by [14], while the elements generated on the strip close to the boundary is generated by an advancing front technique, combined with a Delaunay criterion. The advancing front technique was modified in this work to account for the quadtree information, in order to speed up the process by avoiding that a element is generated by a vertex far from its base edge. Figure 7 illustrates the boundary discretization for a given patch P_k. Since the procedure is based on a quadtree on a two-dimensional parametric space, it is necessary to map each vertex on the three-dimensional space to the parametric space of each patch. Moreover, at the end of the mesh generation, it is necessary to map the vertices of the elements generated in the parametric space back to the three-dimensional space.

The generation of each patch's new mesh occurs in three phases: the generation in the interior by a quadtree procedure; the generation in the transition zone by an advancing front approach and the smoothing of the generated mesh, as described in the following for a generic patch P_k.

Mesh generation in the interior of the domain by a quadtree procedure: Each patch P_k keeps the set of old adjacent elements, which is set up when the initial mesh is generated and is updated after a new mesh is determined at each step. A patch also holds the set of new elements that are generated during the current phase. Hence, these two sets have to co-exist until a new mesh has been committed.

The quadtree procedure not only ensures that the interior density of the cells is sensitive to the boundary discretization but also guarantees a good transition between regions with different degrees of refinement. The procedure consists of the following steps: 1) Construction of the initial quadtree, 2) Adjustments due to errors of curvature for the elements, 3) transformation of the tree into a restricted quadtree, 4) Elimination of cells close to the boundary, and 5) Generation of elements in the interior cells by patterns.

The initial quadtree is constructed taking into account the discretization of the patch's boundary curves obtained in the previous phase of the process (Figure 7 shows the patch P_k in both 3D and parametric spaces). The construction process is performed in the parametric space according with the algorithm described in Algorithm 3.

Figure 8 shows the initial quadtree associated with the boundary curve partition of Figure 7.

The initial quadtree is then modified according with both the characteristics of the domain's mesh and the curvature

Algorithm 3: Construction of the initial quadtree from boundary discretization.

```
1  Initialize the tree structure with the root node empty;
2  for each boundary curve, C_i, i ← 1 to 4 do
3      Compute the length of the curve C_i in 3D space, L_i;
4      for each curve's segment, S_j, j ← 1 to n_seg do
5          Compute the size of S_j in 3D, L_seg_j;
6          Compute the segment's center in 3D space, C_seg_j;
7          Find the parametric coordinates of C_seg_j, (u_j, v_j);
8          Determine in which cell (u_j, v_j) is located;
9          while size of the cell > L_seg_j / L_i do
10             Subdivide the cell into four children;
11             Determine in which cell (u_j, v_j) is located;
```

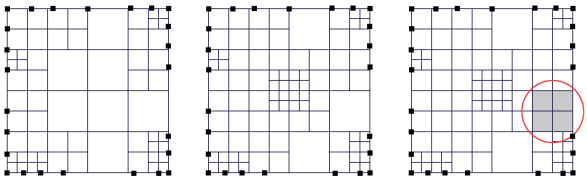

Figure 8. Quadtrees: initial, adjusted by errors and restricted.

errors at the mesh vertices. This is done by the procedure shown in Algorithm 4. In the evaluation of h_{old}, the area is computed in the 3D space. This helps to avoid distortions between the parametric and Cartesian spaces, which can occur since a square bounding box [(0,0) to (1,1)] is used. At the end of this step, the old set of elements is released. Figure 8 shows the adjusted quadtree after taking into account the error of curvatures in the domain's mesh.

Transforming the quadtree obtained in the previous step into a restricted quadtree is required because interior elements are created in interior cells of the quadtree by patterns. Moreover, only one level of difference between adjacent cells ensures a good mesh transition. Figure 8 illustrates the restricted quadtree. The marked cell, for instance, had to be divided.

The next step is the triangulation of the cells. However, due to the boundary curves' discretization, not only the patterns cannot be applied but also distorted elements may be generated. In order to avoid this problem, a transition zone is created by the union of all cells which have contact with the boundary (see Figure 9). The cells that are not part of the transition zone will be meshed with the same patterns presented in [14] (see Figure 9). The cells that are in the transition zone are not used for meshing this region. Instead,

Algorithm 4: Adjustments due to curvature.

$A_{total} \leftarrow 0$;
for *each element of the old set,* $E_k, k \leftarrow 1$ **to** n_{el} **do**
    $A_{total} = A_{total} + A_{E_k}$;
for *each element of the old set,* $E_k, k \leftarrow 1$ **to** n_{el} **do**
    Compute h_{new} *according with Algorithm 5*;
    Compute the center of E_k *in 3D space,* C_{E_k};
    Find parametric coordinates of C_{E_k}, (u_k, v_k);
    Determine in which cell (u_k, v_k) *is located*;
    while *size of the cell* $> h_{new}$ **do**
        Subdivide the cell into four children;
        Determine in which cell (u_k, v_k) *is located*;

Algorithm 5: Computing h_{new} for an element.

$h_{old_k} = \sqrt{\frac{A_{E_k}}{A_{total}}}$;
$h_{new} \leftarrow 0$;
for *each vertex of* $E_k, n_j, j \leftarrow 1$ **to** 3 **do**
    Compute k_a *and* k_d *at vertex* n_j;
    Compute h_{new_j} *as as in Table I*;
    $h_{new} = h_{new} + h_{new_j}$;
$h_{new} = \frac{h_{new}}{3}$;

an advancing front based on a Delaunay triangulation is used.

The cell elimination process results in a quadtree composed of two types of cells: transition and interior cells. If the cell touches the boundary, it is a transition cell; otherwise, it is an interior cell. The leaves of the tree that are interior cells represent the sub-regions that will be meshed with the patterns [14]. The use of patterns facilitates the generation of the interior mesh, accelerating the process. The subdivision made according to the curvature concentrates more elements, generated by patterns, in the most curved areas, mitigating the effect of distortion caused by the parametrization.

Meshing of the transition zone: The transition zone is meshed by an advancing front procedure based on Delaunay triangulation. The advancing front was adapted to search only points in the adjacency of the chosen active edge, avoiding the unnecessary work of searching points occluded by the inner cells. This is the last phase of mesh generation for patch P_k, and is performed as described in the Algorithm

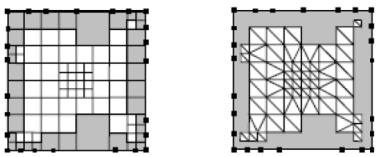

Figure 9. The transition zone.

Algorithm 6: Advacing front.

Initialize the active edge list with boundary curves' segments;
while *the active edge list is not empty* **do**
    `// for a new triangle's base`
    choose an edge;
    for *each chosen edge* **do**
        Build a list of inner cells that are adjacent to the cells that contain the initial and final vertex of the chosen base edge;
        if *there is no such inner cells* **then**
            Build a list of cells that are in the 8-adjacency of the cells that contain the initial and final vertex of the chosen base edge;
    Find the best vertex from the adjacent cell list;
    Build the triangle;
    Update the active edge list;

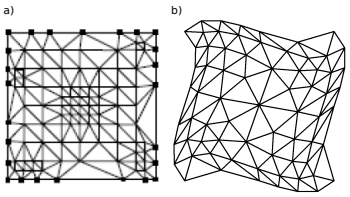

Figure 10. a) Final mesh generated in parametric space. b) Final mesh generated in 3D space.

6. Figure 10 (a) illustrates the mesh after the advancing front. The quality of the mesh is already good, but there are some distortions that can be fixed.

Finally, after the whole new mesh is generated, a smoothing of the mesh is performed. The smoothing adopted here is to compute new coordinates (in the parametric space) for all the vertices of the mesh (except for those of the boundary) as the average of the coordinates of the vertices of the elements which are adjacent to the vertex being modified. This average is repeated several times to provide a more satisfactory result. It is noteworthy that as the smoothing is performed in the parametric space, important features of the surface are not lost and the mesh will represent the parametric surface.

IV. Examples

In this Section, a series of examples is shown. The patches are modeled as bi-parametric surfaces resulting from blending of the boundary curves. As the strategy works directly with parametric surfaces, this approach gives independence to the technique to work with any kind of parametric surface. The convergence criterion used was the η_{global} (see Equation 7). The ϵ used is 3%, but this value is user dependent.

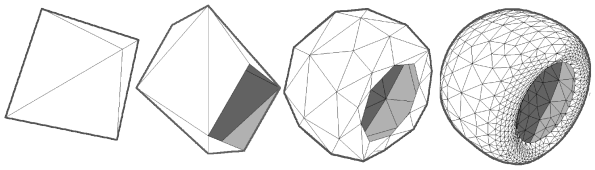

Figure 11. The tire mesh generation.

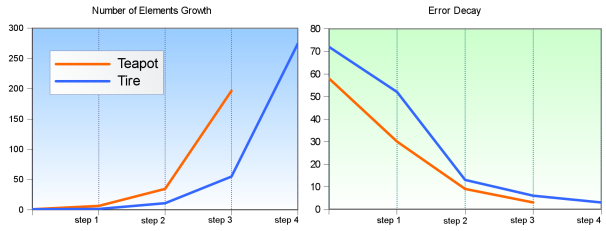

Figure 12. The mesh quality error decay.

The initial meshes were chosen as very coarse meshes. This choice was made on purpose, since the goal is to show the robustness and the effectiveness of the strategy. The Figure 12 shows that even with few steps, where the mesh is still coarse, the error drops drastically. This happens because the number of elements is growing in the correct regions, i.e. where there are great variations of curvatures. This process continues, but tends to stabilize as the mesh represents the geometry more accurately.

A. The tire model

The tire is modelled with two Hermite patches. The first coarse mesh, which is more likely a octahedron than a tire, has 8 triangles and has an error of 71.7%. The final mesh has 2196 elements and an error of 2.9%. Figure 11 shows each step of the process and the time taken from initial to the final mesh was 22.147s.

B. The Utah teapot

This is a well know classic mesh. The Utah teapot was modelled using 32 Bézier patches and the meshes are shown in Figure 1. The first coarse mesh has 128 elements and a error of 58.2% and the final mesh, generated after the 4th step has an error of 2.85% with 25162 elements. This example shows that the method is robust enough to deal with many different levels of curvature, while maintaining a smooth transition between adjacent patches.

V. Conclusions

This work presented a hierarchical adaptive mesh generation strategy for parametric surfaces based on tree structures. It is an curvature-based strategy which is able to refine and coarse regions of the mesh and ensures good mesh transition. It also ensures mesh compatibility between regions since it rediscretizes the curves independently of the domain. The strategy considers the contributions of local error measures to ensure good global mesh quality and it works for any type of parametric surface. The examples shown in this work demonstrated that the adaptive strategy converges to a good quality mesh even if the initial mesh is very coarse.

References

[1] Siu-Wing Cheng, Tamal K. Dey, Edgar A. Ramos and Tathagata Ray, Sampling and Meshing a Surface with Guaranteed Topology and Geometry, Proceedings of the twentieth annual symposium on Computational geometry, pp. 280 - 289 (2004).

[2] Boissonnat, Jean-Daniel and Oudot, Steve, Provably good sampling and meshing of surfaces, Graph. Models, v. 67 n.5, pp. 405-451 (2005).

[3] S.J. Kim, W.k. Jeong and C.H. Kim, LOD Generation with Discrete Curvature Error Metric, In Proceedings of Korea Israel Bi-National Conference, pp. 97-104 (1999).

[4] D. Meek, D. Walton, On surface normal and Gaussian curvature approximations given data sampled from a smooth surface, Computer Aided Geometric Design, v. 17, pp. 521543 (2000).

[5] M. Meyer, M. Desbrun, P. Schrder and A.H. Barr, Discrete Differential-Geometry Operators for Triangulated 2-Manifolds, Visual and Mathematics, v. III, pp. 35-57 (2002).

[6] G. Xu, Convergence Analysis of a Discretization Scheme for Gaussian Curvature over Triangular Surfaces, Computer Aided Geometric Design,vol. 23 no. 2, pp. 193-207 (2006).

[7] E. Magid, O. Soldea, and E. Rivlin, A Comparison of Gaussian and Mean Curvature Estimation Methods on Triangular Meshes of Range Image Data, Computer Vision and Image Understanding, vol. 107, no. 3, pp. 139-159 (2007).

[8] S.H. Lo and T.S. Lau, Finit Element Mesh Generation over Analytical Curved Surfaces, Computers and Structures, vol. 59 no. 2, pp. 301-309 (1996).

[9] S.H. Lo and T.S. Lau, Mesh Generation over Curved Surfaces with Explicit Control on Discretization Error, Engineering Computations, vol. 15 No. 3, pp. 357-373 (1998).

[10] W. Seibold, G. Wyvill, Towards an Understanding of Surfaces through Polygonization, Computer Graphics International Conference, pp. 416 (1998).

[11] N. Dyn, K. Hormann, S.J. Kim and D. Levin, Optimizing 3D triangulations using discrete curvature analysis, Mathematical Methods for Curves and Surfaces, pp. 135-146 (2000).

[12] A.C.O. Miranda and L.F. Martha, Mesh Generation on High-Curvature Surfaces Based on a Background Quadtree Structure, Proceedings of 11th International Meshing Roundtable, Sandia National Laboratories, pp. 333-342 (2002).

[13] D.F. Rogers and J.A. Adams. Mathematical elements for computer graphics, pages 420-421, McGraw-Hill Science/Engineering/Math, 2nd edition (1990).

[14] S. Wittchen, P. Baehmann and M. Shephard, Robust geometrically based, automatic two-dimensional mesh generation. International Journal for Numerical Methods in Engineering, vol. 24, pp. 1043-1078, (1987).