cerca
Centre de recherche en calcul appliqué

ÉCOLE POLYTECHNIQUE MONTRÉAL

Constructing Anisotropic Geometric Metrics Using Octrees and Skeletons

Ko-Foa Tchon Mohammed Khachan François Guibault Ricardo Camarero

Centre de recherche en calcul appliqué (CERCA)
5160, boul. Décarie, bureau 400, Montréal (Québec) H3X 2H9, Canada
[tchon|khachan|francois|ricardo]@cerca.umontreal.ca

Rapport CERCA R2003–??
soumis le 1er mai 2003
révisé le 28 juillet 2003

Accepté à la
12th International Meshing Rountable,
Santa Fe, Nouveau Mexique, É.-U.,
14–17 septembre 2003.

CONSTRUCTING ANISOTROPIC GEOMETRIC METRICS USING OCTREES AND SKELETONS

Ko-Foa Tchon Mohammed Khachan François Guibault Ricardo Camarero

Centre de recherche en calcul appliqué (CERCA)
5160, boul. Décarie, bureau 400, Montréal (Québec) H3X 2H9, Canada.
[tchon|khachan|francois|ricardo]@cerca.umontreal.ca

ABSTRACT

A three-dimensional anisotropic metric for geometry-based mesh adaptation is constructed from a triangulated domain definition. First, a Cartesian background octree is refined according to not only boundary curvature but also a local separation criterion from digital topology theory. This octree is then used to extract the domain skeleton through a medial axis transform. Finally, an efficient anisotropic metric is computed on the octree using the curvature tensor estimated from the boundary triangulation and the local domain thickness information embedded in the skeleton. Applications to geometric adaptation of overlay meshes used in grid-based methods for unstructured hexahedral mesh generation are also presented.

Keywords: geometric adaptation, anisotropic metric, octree skeleton, boundary curvature, domain thickness.

1. INTRODUCTION

Accuracy of finite element and finite volume methods is strongly dependent on the quality of the domain discretization and, more precisely, its mesh. Control of the size, stretching and orientation of the mesh elements is thus crucial. User experience can guide the generation of the mesh to manually adapt it to the problem at hand. Higher vertex densities can be requested in expected high load regions or boundary layers, for example. Such an *a priori* approach is, however, tedious and error prone. Automatic methods based on *a posteriori* error estimators have received extensive attention over the years and proved the effectiveness of solution-based mesh adaptation. See [1] and the references cited therein, among others. The Object-Oriented Remeshing Toolkit (**OO**$\mathcal{RT}$) developed at CERCA implements such methods [2]. However, if no solution is available, when generating an initial mesh for example, alternative methods based on domain geometry must be used

The numerous unstructured mesh generation methods presented in the literature propose many different geometric adaptation approaches. However, like their solution-based counterparts, these algorithms always need to first map the characteristics of the target mesh elements at every point of the domain. Early advancing front methods relied on user specified sample points manually triangulated to form a coarse simplicial background mesh [3]. Target mesh properties are then computed at any point of the domain by locating the host background element and linearly interpolating the sample vertex values. Automated alternatives have then been developed using unconstrained Delaunay triangulations of the vertices of pre-meshed domain boundaries [4]. The target mesh spacing is then interpolated in the domain from boundary specified parameters. Furthermore, the discretization of the boundary itself can be automated using curvature, angle and proximity criteria, see [5] for example. However, such so-called empty Delaunay meshes are very coarse and can result in unwanted abrupt variations of the target mesh properties. To alleviate this side effect, an alternative interpolation scheme based on natural neighbors has been proposed [6]. Even smoother maps can be generated by diffusing target mesh parameters in uniform Cartesian background grids using point and line sources and a Poisson equation [7]. The resulting mesh gradation is very smooth and the uniform structure of the background grid facilitates host location for target parameter interpolation. A uniform grid cannot, however, capture very complex target maps with extreme length scale variations. Quadtrees, in two dimensions, and octrees, in three dimensions, are better suited for such maps because they enable local refinement while retaining implicit recursive structures facilitating host location. The use of quadtrees and octrees for unstructured simplicial mesh generation has been pioneered two decades ago [8] and a review can be found in [9]. These methods recursively divide the domain bounding box until the boundary features are adequately resolved and store the result in a tree structure. Allowing only a difference of one refinement level between neighboring cells results in smooth gradation. To generate a valid mesh, the tree cells are then usually split into simplicial elements and the boundary is recovered. However, since the size distribution of the terminal cells is well adapted to the domain geometry by construction, the final tree structure can also be used almost directly as a target map for other meshing algorithms such as the ad-

vancing front method [10–13]. Quadtrees and octrees can also be used solely as support mediums for more elaborate sizing functions. Their refinement is then not directly based on the domain geometry but rather on the adequate capture of the sizing function gradients [14].

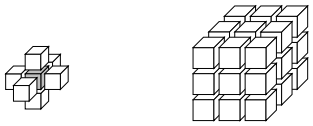
The above list of geometry-based mesh sizing control strategies is far from exhaustive and their combination would give infinite possibilities. Two main ideas emerge however. First of all, target mesh specifications may take many forms but storage in a background mesh, instead of on the fly recomputation for example, is the most flexible approach. It decouples the control map from both the adaptation algorithm as well as the target mesh type, structured, unstructured or hybrid for example. This approach is thus potentially compatible with solution-based adaptation algorithms. The second common idea is that geometric adaptation should be based on the local curvature of the domain. Curvature-based sizing is commonly used for curvilinear and surface meshes and has a solid theoretical foundation [15]. It is, however, insufficient to simply diffuse such a sizing throughout a three-dimensional domain. An additional adaptation criterion based on the local thickness of the domain must be introduced to take into account regions with narrow gaps for example. Designing such a criterion is not trivial. At present, most attempts use heuristics based on proximity between boundary vertices, segments and facets and strongly depend on the boundary mesh itself. The present work proposes to use digital topology theory to extract local thickness information from the domain skeleton on a Cartesian background octree. To resolve possible small gaps in the domain, this octree is first refined according to not only boundary curvature but also a topologic separation criterion. Furthermore, to enable anisotropic adaptation, the octree is only used as a support for a Riemannien metric extracted from the domain boundary curvature tensor and the local thickness information retained by the skeleton. The resulting algorithm has been implemented in a package called $\mathcal{G}eo\mathcal{M}etric$ and applied to overlay mesh adaptation for grid-based unstructured hexahedral mesh generation methods.

2. SOME DIGITAL TOPOLOGY

Pioneered by Azriel Rosenfeld [16], digital topology is mainly used in image processing and provides discrete analogs to Euclidean topology. It is build on the notion of connectedness of adjacent pixels in two dimensions and voxels in three dimensions. Consider, for example, a two-dimensional grid that partitions space in square pixels. Connectivity in this grid is based on two types of adjacency: two pixels are 4-*adjacent* if they share an edge and 8-*adjacent* if they share a vertex. Note that 4-*adjacency* implies 8-*adjacency* because two pixels sharing an edge also share vertices. Similarly, a three-dimensional digital grid partitions space in cubic voxels. Two voxels are 6-*adjacent* if they share a face and 26-*adjacent* if they share a vertex. Furthermore, two pixels (resp. voxels) are α-*connected* if there is a path of α-*adjacent* pixels (resp. voxels) between them. The set of α-*adjacent* neighbors of a pixel or voxel i is called its α-*neighborhood* and noted $\mathcal{N}_\alpha(i)$ or simply $\mathcal{N}_\alpha$ (Figs. 1 and 2). Using these definitions, digital analogs to curves, surfaces and skeletons are presented hereafter for uniform grids before being extended to quadtrees and octrees.

(a) 4-*neighborhood* (b) 8-*neighborhood*

Figure 1: Two-dimensional neighborhoods.



(a) 6-*neighborhood* (b) 26-*neighborhood*

Figure 2: Three-dimensional neighborhoods.

2.1. Digital Curves and Surfaces

One of the fundamental topological property of Euclidean space is the Jordan theorem stating that a simple, i.e., non-self-intersecting, closed curve in two dimensions, or surface in three dimensions, partitions it in exactly two components: an interior and an exterior. Digital curves and surfaces obey the discrete version of the theorem [17]. Consider a two-dimensional binary grid where each pixel is either black or white. Let us call $\mathcal{N}_\alpha^b$ the black or object pixels of a neighborhood $\mathcal{N}_\alpha$ and $\mathcal{N}_\alpha^w$ its white or complement pixels. A 4-*connected* path of black pixels is a digital curve if and only if each of its pixels is simple, i.e., it verifies the following properties:

1. its $\mathcal{N}_8^w$ neighbors are divided in exactly two 8-*connected* components, i.e., the interior and the exterior;
2. its $\mathcal{N}_4^b$ neighbors are 8-*adjacent* to these interior and exterior components.

In Fig. 3, the $\mathcal{N}_8$ neighborhood of the simple curve pixel i contains pixels 1 to 8. Pixels 1, 2, 5 and 8 are black and form $\mathcal{N}_8^b$. Pixels 3, 4, 6 and 7 are white complements and represent $\mathcal{N}_8^w$. This complement is indeed divided in two 8-*connected* components satisfying thereby Property 1. Pixel 4, 6 and 7 constitute one component, the interior for example, while the other component, i.e., the exterior, is composed only of pixel 3. The $\mathcal{N}_4^b$ neighborhood of pixel i contains only two black pixels, 2 and 5. Both pixels are 8-*adjacent* to the interior and the exterior satisfying thereby Property 2. Figure 4 shows an example of a digital curve.

Similarly, a 6-*connected* path of black voxels in a three-dimensional binary grid is a digital surface if and only if each of its voxels is simple, i.e., it verifies the following properties:

1. its $\mathcal{N}_{26}^w$ neighbors are divided in exactly two 26-*connected* components, i.e., the interior and the exterior;
2. its $\mathcal{N}_6^b$ neighbors are 26-*adjacent* to these interior and exterior components.

This concept of digital surface is essential to determine the grid resolution necessary to digitally represent a topological equivalent of a given domain geometry and will be used as a criterion for the background octree refinement.

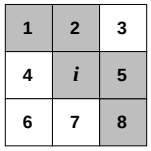


Figure 3: Simple curve pixel.

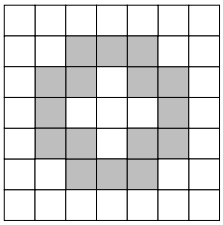


Figure 4: Simple closed 4-*connected* curve with two 8-*connected* components in the complement.

2.2. Digital Skeletons

The skeleton concept was introduced in continuous space by Blum as a shape descriptor [18]. Its discrete counterpart, the digital skeleton, is now used as a compact representation of binary shapes in image processing and pattern recognition. Conceptually, skeletonization transforms a two-dimensional object into its one-dimensional median line and a three-dimensional object into a two-dimensional median surface. Practically, digital skeletons are thin subsets of a binary shape that reflect its connectivity.

Topological thinning algorithms are commonly used to obtain skeletons. Those algorithms erode layer by layer a digital object by turning off all pixels that can be removed without altering the topology of the original object. Thinning algorithms tend, however, to produce excessive erosion and have to be constrained. Alternatively, skeletons can also be generated using a Medial Axis Transform (MAT). Let δ be the distance to the boundary of any point inside an object. This distance transform δ, measured in grid cells, can be computed for each pixel or voxel i as follows:

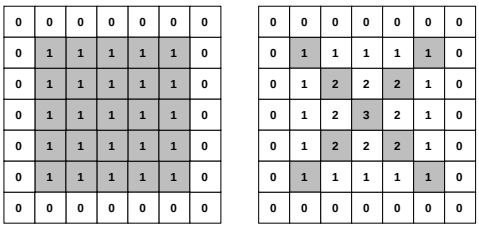
1. Initialize δ_i^0 to 1 for all interior and boundary pixels or voxels and to 0 all exterior ones;
2. Set $\delta_i^n = \delta_i^0 + \min_{j \in \mathcal{N}_\alpha(i)} \delta_j^{n-1}$;
3. Iterate step 2 until $\delta_i^n = \delta_i^{n-1}$.

The MAT skeleton is then formed by all the pixels or voxels such that $\delta_i^n \geq \max_{j \in \mathcal{N}_\alpha(i)} \delta_j^n$. Figure 5 shows a square shape before and after skeletonization using a distance transform.

The medial axis, or skeleton, is the locus of the centers of the maximal balls contained by an object and can, therefore, be used to extract boundary proximity and local domain thickness information.

2.3. Extension to Quadtrees and Octrees

Quadtrees and octrees can be considered as irregular polygonal and polyhedral meshes and the above definitions can be extended to such meshes [19]. Two polygonal cells are



(a) Initial shape (b) Skeleton

Figure 5: Skeletonization of a square shape using a distance transform with a 4-*neighborhood*.

edge-adjacent if they share an edge. This is equivalent to 4-*adjacency* in a two-dimensional regular grid. Similarly, two polyhedral cells are *face-adjacent* if they share a face. This is equivalent to 6-*adjacency* in a three-dimensional regular grid. Furthermore, two polygonal or polyhedral cells are *vertex-adjacent* if they share a vertex. This is equivalent to 8-*adjacency* in a two-dimensional regular grid and to 26-*adjacency* in a three-dimensional regular grid. *Edge-adjacency* and *face-adjacency* imply *vertex-adjacency*. Let $\mathcal{N}$ with the subscripts e, f and v note the *edge*, *face*, and *vertex-neighborhoods* respectively. Using this notation, a polygonal cell of a two-dimensional *edge-connected* curve is simple if it verifies the following properties:

1. its $\mathcal{N}_v^w$ neighbors are divided in exactly two *vertex-connected* components, i.e., the interior and the exterior;
2. its $\mathcal{N}_e^b$ neighbors are *vertex-adjacent* to these interior and exterior components.

In Fig. 6, the $\mathcal{N}_v$ neighborhood of the simple quadree curve cell i contain cells 1 to 7 but not cells 8 and 9. Cells 2, 3, 5 and 6 are black and form $\mathcal{N}_v^b$. Cells 1, 4 and 7 are white and represent $\mathcal{N}_v^w$. This white complement is indeed divided in two *vertex-connected* components satisfying thereby Property 1. Cell 1 forms one component, the interior for example, while cells 4 and 7 constitute the other component, i.e., the exterior. The $\mathcal{N}_e^b$ neighborhood of cell i contains only two black cells, 2 and 6. Both cells are *vertex-adjacent* to the interior and the exterior satisfying thereby Property 2.

Similarly, a polyhedral cell of a *face-connected* surface is simple if it verifies the following properties:

1. its $\mathcal{N}_v^w$ neighbors are divided in exactly two *vertex-connected* components, i.e., the interior and the exterior;
2. its $\mathcal{N}_f^b$ neighbors are *vertex-adjacent* to these interior and exterior components.

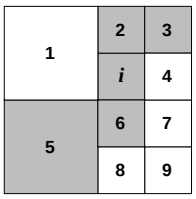


Figure 6: Simple quadtree curve cell.

Finally, the medial axis transform can also be extended to quadtrees [20] and octrees by modifying the distance transform to take into account variable cell sizes:

1. Initialize δ_i^0 to half the cell size for all interior and boundary cells and to 0 all exterior ones;
2. Set $\delta_i^n = \delta_i^0 + \min_{j \in \mathcal{N}_v(i)} \left(\delta_j^0 + \delta_j^{n-1} \right)$;
3. Iterate step 2 until $\delta_i^n = \delta_i^{n-1}$.

Let the largest box associated with each cell be the square, for a quatree, or the cube, for an octree, of size 2δ centered at the cell. A maximal cell is then a cell i whose largest box is not completely contained by the largest box of any other cell, i.e., $\delta_i^n > \max_{j \in \mathcal{N}_v(i)} \left(\delta_j^n - \delta_j^0 - \delta_i^0 \right)$, and the skeleton is the set of all maximal cells.

3. OCTREE GENERATION AND SKELETONIZATION

The skeletonization process described above is meaningful only if the octree is fine enough to resolve the significant features of the domain geometry. To ensure such a resolution, both local curvature and thickness refinement criteria are used. After presenting the required geometry definition, this section describes these criteria as well as the refinement process itself and the skeletonization of the resulting octree.

3.1. Domain Geometry Definition

The required input for the Cartesian background octree generation is a domain geometry definition. This definition must enable us to perform boundary intersection and inside-outside tests for the octree cells as well as closest point and local curvature interrogation. For the present project, triangulated boundary representations, from STereo Lithography (STL) files for example, were used. Since such triangulations simply serve as a support for geometric information, they do not have to be of high quality. They can be too fine but should not be too coarse or essential details will be lost. Ultimately, the user decides the level of details to be taken into account. Triangulations can also be dirty, i.e., not watertight. Dirt size should, however, be inferior to the size of the neighboring cells to make it invisible to the octree. One way to insure that is to make dirt size inferior to the size of the smallest possible cell. To accelerate intersection tests, the triangles are stored in an Alternating Digital Tree (ADT) [21] and, to improve accuracy and robustness, adaptive precision arithmetic is used [22]. Furthermore, Simulation of Simplicity (SoS) copes with degenerate intersection configurations such as barely touching entities [23]. Finally, curvature information can be given by the user along with the triangle vertices or it can be estimated directly from the triangulation [24].

Figure 7 shows the triangulation of an intricate mechanical part, a water jacket, downloaded from AVL [25]. The duplicate vertices of the STL file were first merged and the resulting triangulation was partitioned along sharp feature lines. The curvature tensor was then estimated separately for each patch. This geometry will be used throughout the present paper to illustrate the different steps of the algorithm.

Figure 7: Water jacket — Triangulated geometry [25].

3.2. Octree Refinement

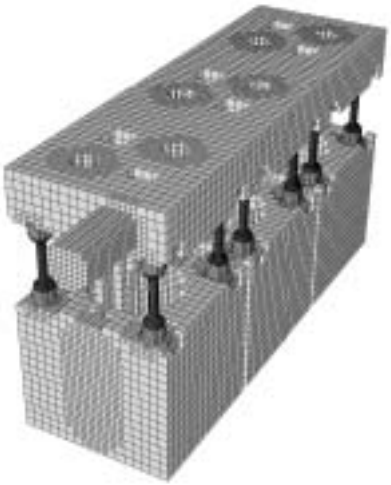
The Cartesian background octree is generated using recursive non-conformal refinement of the input geometry bounding box. To resolve the significant features of the domain, a curvature-based criterion is first used. As in [26], each triangle of the geometry definition is associated with a target cell size based on the maximal curvature estimated anywhere on the facet. The higher the curvature, the smaller the target size. Octree cells are then simply refined until their sizes is smaller than the target sizes of the triangles they intersect. Note that only interference with the triangle bounding boxes are checked instead of actual intersection. The resulting octree will be slightly finer than strictly necessary but it greatly accelerates the refinement process.

A curvature criterion is, however, insufficient to ensure an adequate skeletonization. To find the medial axis, at least some cells must indeed be located strictly inside the domain. For example, small and almost planar gaps will not be adequately resolved using curvature-based refinement only. To avoid this problem, the octree could be refined until the boundary of the domain is discretized by a digital surface. However, in practice, such a refinement proved excessive for the present application. Digital surfaces as defined in Section 2 indeed introduce two criteria. The first one is a separation criterion requiring that the octree can be partitioned around each intersecting cell into an inside and an outside. The second criterion requires that the digital surface is thin enough for each intersecting cell to see this interior and exterior. This last criterion is mainly useful to triangulate digital surfaces using marching-cube algorithms. Applying this criterion tends to drive the refinement of octree cells intersecting non-axis aligned boundary surfaces up to the minimum size allowed. However, for our purposes, a thickness of two octree cells can be allowed as long as the surface still separates the inside from the outside of the domain. The second criterion has thus been dropped and the separation criterion slightly modified. Let the superscripts b and w note the boundary intersecting and non-intersecting cells respectively. Experimentally, it proved sufficient to require that the extended neighborhood $\bigcup_{j \in \mathcal{N}_v^b(i)} \mathcal{N}_v^w(j)$ of

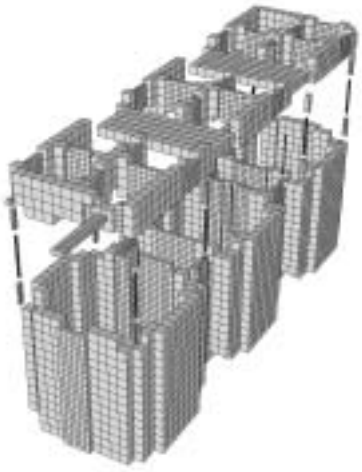
(a) Boundary intersecting cells

(b) Skeleton cells

Figure 8: Water jacket — Octree generation and skeletonization using a curvature criterion only.



(a) Boundary intersecting cells



(b) Skeleton cells

Figure 9: Water jacket — Octree generation and skeletonization using both curvature and separation criteria.

each boundary intersecting cell i is partitioned in exactly two *vertex-connected* component, i.e., an interior and an exterior. This modified criterion results in a thicker discretized surface in exchange for a substantial reduction of the octree size. The actual savings depend on the domain geometry but, for the water jacket case, a reduction of the octree size by a factor of two was achieved.

Using these two refinement criteria, octree generation proceeds as follows:

1. Create a root cell encompassing the whole domain and flag it as intersecting the boundary;
2. Iterate to refine according to curvature:
   (a) Mark the set of cells interfering with boundary triangles whose target curvature-based size is inferior to their own current size;
   (b) Add to this set the $\mathcal{N}_v^b$ neighborhood of the cells marked in step 2a;
   (c) Refine the cells of the resulting set;
   (d) Balance the tree to allow a difference of only one level of refinement between *face-adjacent* cells;
   (e) Identify boundary intersecting children cells to update intersection flags.
3. Identify interior and exterior cells;
4. Iterate to refine according to separation criterion:
   (a) Mark the set of boundary intersecting cells invalidating the separation criterion;
   (b) Add to this set the $\mathcal{N}_v^b$ neighborhood of the cells marked in step 4a;

(c) Refine the cells of the resulting set;

(d) Balance the tree to allow a difference of only one level of refinement between *face-adjacent* cells;

(e) Update intersection, interior and exterior flags.

Note that propagation to immediate neighbors and balancing are used to diffuse and smooth out the refinement. Furthermore, octree cells cannot be refined beyond a minimum size $h_{\min}$. This may preclude adequate resolution of some geometric features, blunt some corners and fill some gaps. Adverse effects are, however, minimized if $h_{\min}$ corresponds to the minimum mesh element size allowed during the actual adaptation process.

Figures 8(a) and 9(a) show the boundary intersecting cells of the octree generated for the water jacket geometry without and with the separation criteria. Those are relatively moderate size octrees counting 204777 and 217769 cells respectively. Practical geometries could however be much more complex and need bigger octrees. To accommodate such applications, explicit mesh-like data structures storing the vertices, edges and faces of the cells have been avoided in favor of an implicit tree structure storing only the parent and children for each cell. The size and position of the cells are then computed from the octree root. Such a data structure can, however, be very taxing during neighbor searches performed to verify the separation criterion. That is why binary coordinates were added to each cell to accelerate tree traversal and cell localization [27]. The resulting data structure is fast and compact.

3.3. Octree Skeletonization

Using the medial axis transform, skeletonization of the resulting octree is rather straightforward but may result in unwanted branches. Corners in the domain can indeed produce terminal skeleton branches going all the way to its boundary (Fig. 5). Those terminal branches are not desirable because the radius of the maximal balls tends to zero as we approach the boundary and do not always indicate an adequate local mesh size. For example, if the corner angle is very small then effectively the local mesh size should be small, i.e., the minimum allowable. However, if the angle is around 90 degrees or more then the size of the maximal balls is not a good indicator of the necessary local mesh size. Furthermore, skeletonization is sensitive to noise from the domain discretization by the octree, i.e., its stair-step boundary. It can produce very small branches terminating at noisy border cells. Those noise induced branches behave like corners with very wide angles close to 180 degrees.

A skeleton simplification to prune those unwanted branches is thus needed. The same strategy was used to prune corner and noise induced terminal branches. Following [28], the separation angle filters unwanted skeleton cells. By definition, this angle is formed by the vectors connecting an actual medial axis point to its closest boundary points (Fig. 10). It is approximated on skeleton cell edges by the angle formed by the vectors to the closest boundary point of each end vertex. For each skeleton cell, it is then taken as the minimum of its edge separation angles. The separation angle is big for branches resulting from sharp corners and small for blunt ones. Cells with a separation angle smaller than a given

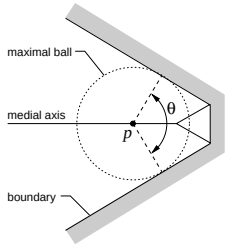


Figure 10: Separation angle θ for a medial axis point p of the two-dimensional object.

threshold are pruned from the skeleton. Best results were obtained with a threshold of 120 degrees.

Note also that the present medial axis tranform is based on a chessboard distance measure on the octree and may thus suffer from digitization bias [30, 31]. The induced error is, however, acceptable for our purposes and is partly compensated by the pruning algorithm that uses Euclidean distances.

Figure 9(b) presents the final skeleton for the water jacket geometry. When compared to the skeleton extracted from an octree refined using a curvature criterion only (Fig. 8(b)), the necessity of the separation criterion is clear. Without it, some important branches of the skeleton are missing. Note also that these skeletons are rather fat, and sometimes disconnected. Because the octree is refined only at the domain boundary, the center of the domain is coarsely discretized and the resulting skeleton is only a rough approximation. It is, however, sufficient because only an approximation of the local thickness of the domain is needed. The skeleton is fat in thick regions of the domain and svelte in thin ones. The relative error made on the size of the maximal boxes reflecting the local thickness of the domain is thus more or less constant and quite acceptable for our purposes.

4. METRIC EXTRACTION

By construction, the size distribution of the final octree cells adequately resolves the domain because it takes into account both boundary curvature and local thickness. This information is, however, isotropic and not optimal, i.e., it only gives the most constraining limit. Consider, for example, a long and narrow gap misaligned with the root cell. The octree leaf cells in the neighborhood of this gap reflect the thickness of the gap and not its length. Meshing such a gap with elements restricted to the size of these cells would be wasteful. This octree is, however, the ideal support medium for an anisotropic geometric metric map because it is already adapted to its expected variations. After a brief summary on Riemannian control metrics, this section presents the extraction of more efficient anisotropic geometry-based sizing information for such a map from the domain boundary triangulation and the octree skeleton.

4.1. Riemannian Metric

To control adaptation, an anisotropic control map must be used to prescribe not only the size but also the stretching and orientation of the mesh elements to be built. These specifications can be given as the metric of the transformation that

maps a perfect mesh element into a unit cube for hexahedral meshes or a unit equilateral tetrahedron for simplicial meshes. In three dimensions, this Riemannian metric is defined at every point of the domain by a symmetric positive-definite 3×3 matrix $\mathcal{M}$. This matrix can be factored as the product of a rotation matrix $\mathcal{R}$ and a diagonal scaling matrix Λ:

$$\mathcal{M} = \mathcal{R}\Lambda\mathcal{R}^{-1} = \mathcal{R} \begin{pmatrix} h_1^{-2} & 0 & 0 \\ 0 & h_2^{-2} & 0 \\ 0 & 0 & h_3^{-2} \end{pmatrix} \mathcal{R}^{-1} \quad (1)$$

where h_1, h_2 and h_3 are the target element sizes along the three axes of the local coordinate system defined by $\mathcal{R}$. Such a size specification map can be given analytically or constructed from *a posteriori* error analysis, from the geometrical properties of the domain, as in the present paper, or from any other user defined inputs. An isotropic size specification map reduces to an identity matrix multiplied by h^{-2} where h is the desired element size. Whatever it's origin, the control metric contains information on the prescribed size, stretching and orientation of the mesh to be built as an anisotropic metric field. See [32] as well as [33] and [1] for a more complete discussion on metrics.

Using metrics promotes decoupling of the actual adaptation algorithm from the target mesh specifications. Algorithm traditionally used for solution-based adaptation through global remeshing or local mesh modifications can then be used for geometric adaptation.

4.2. Extracting Thickness and Curvature Data

The adaptation metric has to take into account both the local thickness and curvature of the domain. Thickness information can be extracted from the octree skeleton cells using the distance transform δ. Curvature information, on the other hand, can be extracted from octree cells intersecting the triangulated domain boundary. To construct the geometric metric, these two types of information must be combined and diffused in the whole domain.

For each octree cell, δ is an approximation of the distance to the closest domain boundary. By definition, it is also an approximation of the radius of the maximal ball centered on a skeleton cell and the local thickness τ of the domain is equal to twice this radius. However, at non-skeleton cells, τ is not related to the local value of δ but rather to the thickness associated to the closest skeleton cell. Computing τ at those non-skeleton cells could thus be reduced to searching for the closest skeleton cell. To get a smoother distribution, τ can also be simply diffused from the skeleton cells to the rest of the octree using a Laplacian operator:

$$\tau_i^{n+1} = \tau_i^n + \frac{\sum_{j \in \mathcal{N}_f(i)} \left(\tau_j^n - \tau_i^n\right) / l_{ij}}{\sum_{j \in \mathcal{N}_f(i)} 1/l_{ij}} \quad (2)$$

where n is an iteration counter and l_{ij} is the Euclidean distance from the center of cell i to the center of cell j. This latter approach was used. Dirichlet conditions were imposed at skeleton cells where τ was set to twice the value of the distance transform δ. To simplify the metric interpolation and minimize memory requirements to store the background octree, the metric, and therefore τ, was considered constant over each octree cell. The octree resolution proved experimentally sufficient since the generation process locally refined cells in expected high gradient regions.

After its diffusion, the local thickness τ can be combined with the curvature tensor at each boundary intersecting cell to give the following metric:

$$\mathcal{M} = \left(\vec{n} \ \vec{t}_1 \ \vec{t}_2 \right)^T \begin{pmatrix} h_\tau^{-2} & 0 & 0 \\ 0 & h_{\kappa_1}^{-2} & 0 \\ 0 & 0 & h_{\kappa_2}^{-2} \end{pmatrix} \left(\vec{n} \ \vec{t}_1 \ \vec{t}_2 \right) \quad (3)$$

where $\vec{n}$ is the unit normal to the boundary while $\vec{t}_1$ and $\vec{t}_2$ are the unit tangents in the direction of the principal curvatures κ_1 and κ_2 evaluated at the boundary point closest to the center of the intersecting cell. The directional target sizes h_τ, h_{κ_1} and h_{κ_2} are computed as functions of τ, κ_1 and κ_2. To diffuse this metric tensor throughout the domain, a term by term Laplacian operator is used with intersecting cell values acting as Dirichlet conditions:

$$\mathcal{M}_i^{n+1} = \mathcal{M}_i^n + \frac{\sum_{j \in \mathcal{N}_f(i)} \left(\mathcal{M}_j^n - \mathcal{M}_i^n\right) / l_{ij}}{\sum_{j \in \mathcal{N}_f(i)} 1/l_{ij}} \quad (4)$$

where the notation of Eq. (2) is used. Finally note that, at cells located outside the domain, the metric tensor is set to the identity matrix times the squared inverse of the prescribed size at infinity, usually chosen as the size of the domain bounding box.

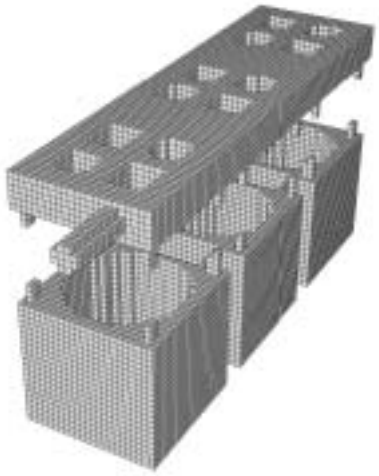
5. APPLICATION

The application we propose to explore is geometric adaptation for superposition or grid-based hexahedral mesh generation methods [35–42]. Such methods overlay an initial mesh, usually Cartesian, on the domain geometry and keep only its interior elements. The boundary of the resulting mesh has then to be fitted to the domain through cutting, projection or isomorphism. These methods are robust but often generate poor quality elements at the boundary of the domain because of the misalignment of the initial mesh [43,44]. Furthermore, the body fitting step of grid-based methods can be performed reliably only if local mesh density is sufficient to capture the features of the domain geometry [26]. The present metric construction algorithm combined with an appropriate adaptation tool is ideally suited to generate the initial mesh required by such methods.

Relocation algorithms are commonly used for solution-based adaptation and essentially smooth the mesh in the target anisotropic metric space. The particular algorithm used here is based on a spring analogy that considers the mesh as a network of vertices linked by springs. The optimal position of each vertex i is computed iteratively by the following length equidistribution formula:

$$\vec{x}_i^{\,n+1} = \vec{x}_i^{\,n} + \omega \frac{\sum_j k_{ij}^n \left(\vec{x}_j^{\,n} - \vec{x}_i^{\,n}\right)}{\sum_j k_{ij}^n} \quad (5)$$

where n is an iteration counter, ω is a relaxation factor, j denotes all vertices sharing an edge with vertex i and the spring rigidity constant k_{ij} is the metric length of edge ij



(a) Interior elements before geometric adaptation

(b) Interior elements after geometric adaptation

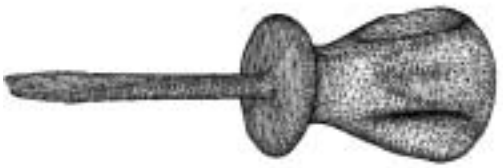
Figure 11: Water jacket — Overlay mesh to be used with a grid-based method.

divided by its Euclidean length. The metric length of an edge ij is given by:

$$l_{ij}^{\mathcal{M}} = \int_0^1 \sqrt{(\vec{x}_j - \vec{x}_i)^T \mathcal{M}(\vec{x}_t)\,(\vec{x}_j - \vec{x}_i)}\, dt \qquad (6)$$

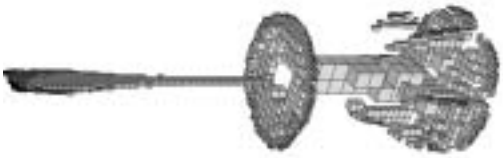
where $\vec{x}_t = \vec{x}_i + t\,(\vec{x}_j - \vec{x}_i)$. This metric length is integrated numerically using a simple trapezoidal rule with $\mathcal{M}$ being interpolated on the background octree. See [45] for more details on relocation methods.

Figure 11 presents an initial structured hexahedral mesh for the water jacket geometry with and without geometric adaptation. This cubic overlay grid counts $100 \times 100 \times 100$ hexahedra. Without adaptation, only 23071 elements are located inside the geometry and are retained for the body-fitting step of the grid-based method. Fine details cannot be captured adequately by such a Cartesian mesh unless the resolution is drastically increased. Global refinement would be very



(a) Triangulated geometry [34]

(b) Background octree



(c) Octree skeleton

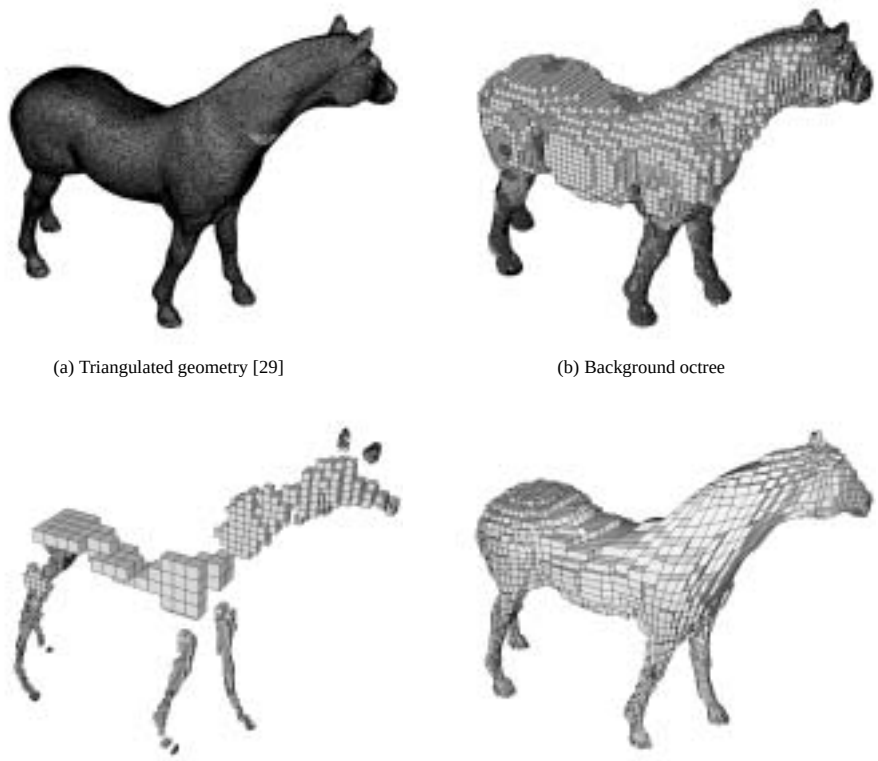
(d) Adapted overlay mesh

Figure 12: Screwdriver — Geometric adaptation of an overlay mesh for a grid-based method.

wasteful while local refinement is usually isotropic and resort to non-conformal transition elements. However, with adaptation by point relocation and a geometric metric, those details are easily resolved. The adapted overlay mesh counts 116430 elements inside the geometry and those elements are better aligned with the boundary. The present geometric adaptation strategy should thus make the overlay meshes less sensitive to misalignment problems typical of grid-based methods. The anisotropy introduced by the geometric metric is also much more efficient than the usual Cartesian refinement, global or even local, for long and narrow regions. Finally, although the general shape of the domain can already be recognized, the actual boundary of the model has yet to be recovered. The present adaptation process should, however, greatly facilitate the body-fitting process.

Figures 12 and 13 present further adaptation examples for geometries found on the Internet [29, 34]. In addition to the adapted overlay mesh to be used by a grid-based method, the triangulated domain, the intersecting cells of the background octree and the corresponding skeleton are also presented. Note how much more efficient is the adapted anisotropic mesh compared to the isotropically refined octree.

As illustrated by these examples, surprisingly good results can be obtained using only point relocation to adapt the over-



(a) Triangulated geometry [29]

(b) Background octree

(c) Octree skeleton

(d) Adapted overlay mesh

Figure 13: Horse — Geometric adaptation of an overlay mesh for a grid-based method.

lay mesh. However, for geometries with very large length scale variations, local mesh refinement may be needed. To avoid hanging-nodes, a conformal all-hexahedral refinement method based a pillowing or shrink and connect strategy can be used [46]. Consider, for example, the toy dinosaur [34] presented in Fig. 14. Again, the triangulated domain, the intersecting cells of the background octree and the corresponding skeleton are presented in addition to the adapted overlay mesh. This adapted mesh was generated from a $5 \times 5 \times 5$ cubic grid encompassing the domain that was coarsely refined using the shrink and connect strategy. Each resulting hexahedron was then diced in 8 and the point relocation algorithm smoothed the resulting mesh in the target geometric metric. The final mesh counts 48954 elements located inside the geometry. Figure 15 plots a cut of the complete overlay mesh before the removal of non-interior elements and shows how the geometric metric drives the combined point relocation and local refinement algorithm to effectively carve the model geometry out of the initial cube. Although the elements located outside the geometry are very distorted, the geometric metric, and the smoothing process used to generate it, gives very good elements inside the geometry itself without any inverted cell.

Finally note that, for all the examples presented in this section, the following size functions were used: $h_\tau = \tau/3$ and $h_{\kappa_{1,2}} = \pi/8\,|\kappa_{1,2}|$. Furthermore, a relaxation factor ω of 0.5 was necessary in Eq. 5 to stabilize the point relocation algorithm.

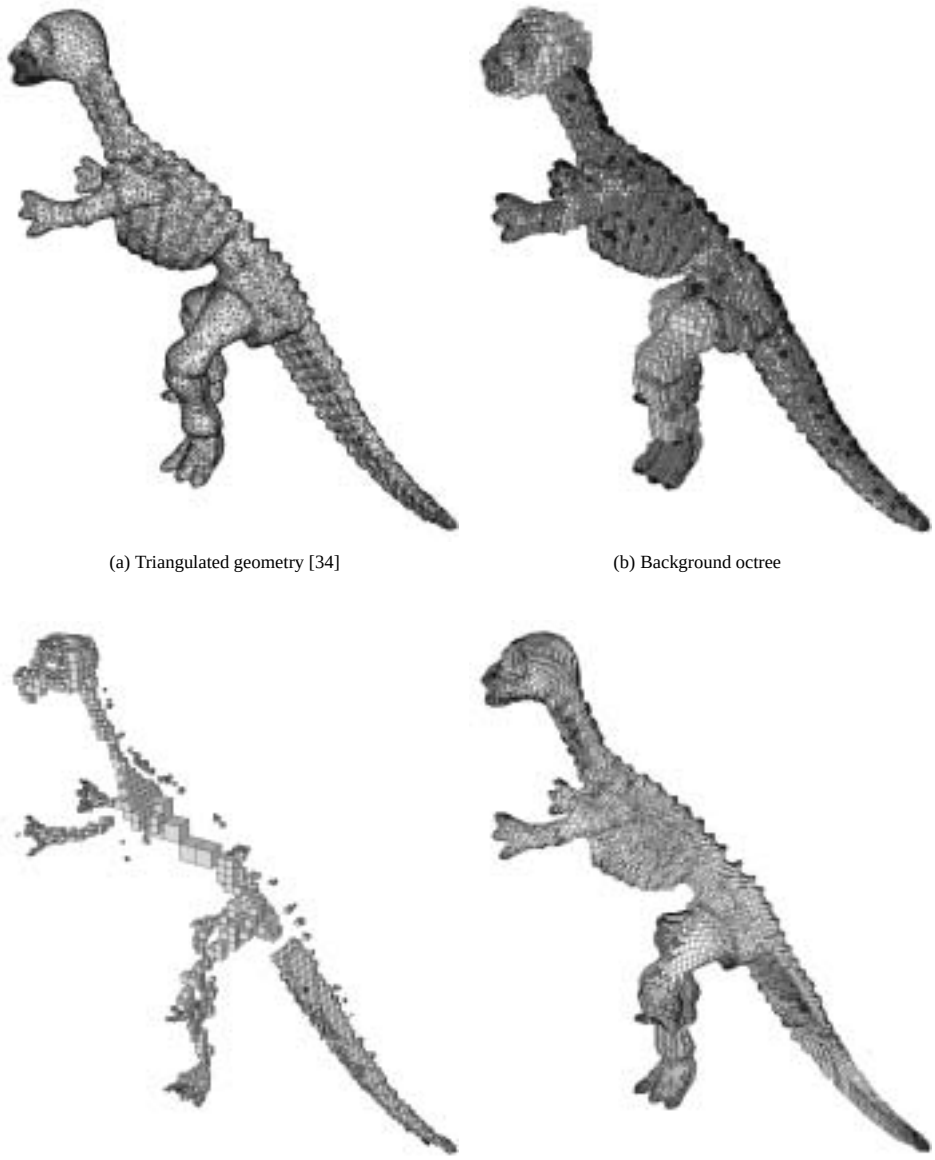
6. CONCLUSION

A new method to construct geometry-based adaptation metrics from triangulated domains was introduced in the present paper. These anisotropic metrics are computed using local domain curvature, estimated from its triangulated boundaries, as well as thickness. Digital topology theory is used to extract this thickness from the domain skeleton on a Cartesian background octree used as a support medium for the anisotropic metric. Applications illustrated the effectiveness of this approach for hexahedral mesh adaptation.

The present geometric metric can, however, be used on any other mesh type, tetrahedral or hybrid for example, as long as the adaptation algorithm uses metrics to specify its target. It could also be combined with other specifications based on user experience. For example, in computational fluid dynamics applications, the metric could be modified to take into account boundary layers around solid walls. Although, it could never replace a solution-based metric computed using *a posteriori* error estimators, such a metric is, however, invaluable to generate and adapt initial meshes when no solution is yet available.

7. ACKNOWLEDGMENTS

The authors would like to thank NSERC for its financial support. Many thanks also to Julien Dompierre for his insight on metrics and mesh adaptation.



(a) Triangulated geometry [34]

(b) Background octree

(c) Octree skeleton

(d) Adapted overlay mesh

Figure 14: Toy dinosaur — Geometric adaptation of an overlay mesh for a grid-based method.

REFERENCES

[1] P. J. Frey and P.-L. George, *Mesh Generation. Application to Finite Elements*. Paris: Hermès, 2000.

[2] J. Dompierre, P. Labbé, and F. Guibault, "OORT (Object-Oriented Remeshing Toolkit)." http://www.cerca.umontreal.ca/oort.

[3] R. Löhner and P. Parikh, "Generation of three-dimensional unstructured grids by the advancing front method," *AIAA–88–0515*, 1988.

[4] É. Seveno, *Génération automatique de maillages tridimensionnels isotropes par une méthode frontale*. PhD thesis, Université Pierre et Marie Curie, Paris VI, Mar. 1998.

[5] A. Cunha, S. A. Canann, and S. Saigal, "Automatic boundary sizing for 2D and 3D meshes," in *AMD-Vol. 220 Trends in Unstructured Mesh Generation*, pp. 65–72, ASME, July 1997.

[6] S. J. Owen and S. Saigal, "Neighborhood-based element sizing control for finite element surface meshing," in *Sixth International Meshing Roundtable*, (Park City, Utah), pp. 143–154, Sandia National Laboratories, Oct. 1997.

[7] S. Pirzadeh, "Structured background grids for generation of unstructured grids by advancing-front method,"

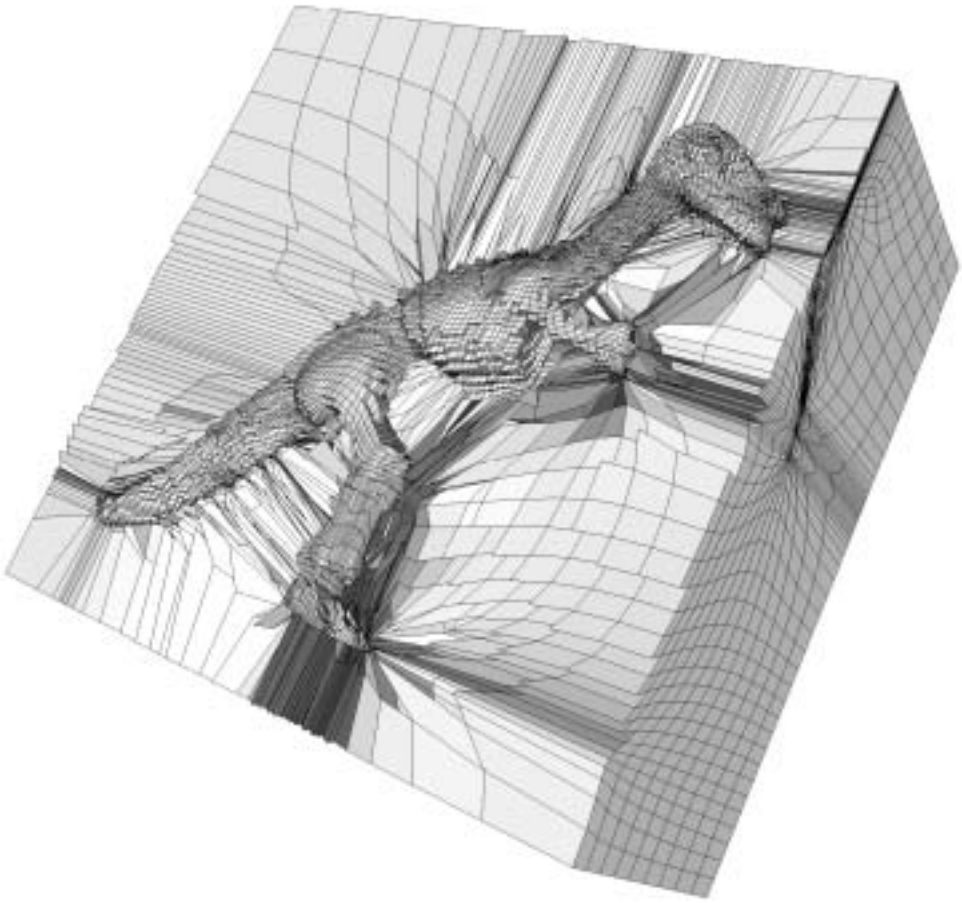


Figure 15: Toy dinosaur — Cut of the whole adapted overlay mesh.

AIAA J., vol. 31, pp. 257–265, Feb. 1993.

[8] M. A. Yerry and M. S. Shephard, "A modified quadtree approach to finite element mesh generation," *IEEE Comput. Graph. Appl.*, vol. 3, pp. 39–46, Feb. 1983.

[9] M. S. Shephard, "Approaches to the automatic generation and control of finite element meshes," *Applied Mechanics Review*, vol. 41, no. 4, pp. 169–185, 1988.

[10] Y. Kallinderis, A. Khawaja, and H. McMorris, "Hybrid prismatic/tetrahedral grid generation for complex geometries," in *AIAA 33rd Aerospace Sciences Meeting and Exhibit*, no. AIAA-95-0211, January 9–12, 1995.

[11] J.-C. Carette and H. Deconinck, "Adaptive hybrid remeshing and SUPG/multiD upwind solver for compressible high-Reynolds number flows," in *13th AIAA Computational Fluid Dynamics Conference*, no. AIAA–97–1857, (Snowmass, CO), June 1997.

[12] P. J. Frey and L. Maréchal, "Fast adaptive quadtree mesh generation," in *Seventh International Meshing Roundtable*, (Dearborn, Michigan), pp. 211–224, Sandia National Laboratories, Oct. 1998.

[13] A. C. O. Miranda and L. F. Martha, "Mesh generation on high-curvature surfaces based on a background quadtree structure," in *Eleventh International Meshing Roundtable*, (Ithaca, NY), pp. 333–342, Sandia National Laboratories, Sept. 2002.

[14] J. Zhu, T. Blacker, and R. Smith, "Background overlay grid size functions," in *Eleventh International Meshing Roundtable*, (Ithaca, NY), pp. 65–74, Sandia National Laboratories, Sept. 2002.

[15] P. J. Frey, "About surface remeshing," in *Ninth International Meshing Roundtable*, (New Orleans, Louisiana), pp. 123–136, Sandia National Laboratories, Oct. 2000.

[16] A. Rosenfeld, "Digital topology," *American Mathematical Monthly*, vol. 86, no. 8, pp. 621–630, 1979.

[17] T.-Y. Kong and A. Rosenfeld, "Survey digital topology: Introduction and survey," *CVGIP*, vol. 48, pp. 357–393, 1989.

[18] H. Blum, "A transformation for extracting new descriptors of shape," in *Models for the Perception of Speech and Visual Form* (W. Wathen-Dunn, ed.), pp. 362–380, MIT Press, 1967.

[19] M. Khachan, *Topological study for localization and reconstruction of geometrical objects*. PhD thesis, University Joseph Fourier, Grenoble, France, January 1998.

[20] H. Samet, "A quadtree medial axis transform," *Communications of the ACM*, vol. 26, pp. 680–693, Sept. 1983.

[21] J. Bonet and J. Peraire, "An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems," *Int. J. Numer. Meth. Engng*, 1991.

[22] J. R. Shewchuk, "Adaptive precision floating-point arithmetic and fast robust geometric predicates," *Discrete & Computational Geometry*, vol. 18, pp. 305–363, Oct. 1997.

[23] H. Edelsbrunner and E. P. Mucke, "Simulation of simplicity: A technique to cope with degenerate cases in geometric algorithms," in *Symposium on Computational Geometry*, pp. 118–133, 1988.

[24] G. Taubin, "Estimating the tensor of curvature of a surface from a polyhedral approximation," in *Fifth International Conference on Computer Vision (ICCV 95)*, (Massachusetts Institute of Technology, Cambridge, Massachusetts, USA), pp. 902–907, IEEE Computer Society, June 1995.

[25] AVL, "AVL eFAME — Web based meshing and grid generation." http://www.avl.com.

[26] K.-F. Tchon, C. Hirsch, and R. Schneiders, "Octree-based hexahedral mesh generator for viscous flow simulations," in *13th AIAA Computational Fluid Dynamics Conference*, no. AIAA–97–1980, (Snowmass, CO), June 1997.

[27] S. F. Frisken and R. N. Perry, "Simple and efficient traversal methods for quadtrees and octrees," Technical Report TR2002-41, Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, Nov. 2002. to appear in Journal of Graphics Tools.

[28] M. Foskey, M. Lin, and D. Manocha, "Efficient computation of a simplified medial axis," in *ACM Confereonce on Solid Modeling*, 2003. To appear.

[29] Georgia Tech College of Computing, "Large geometric models archive." http://www.cc.gatech.edu/projects/large_models.

[30] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. 40, pp. 1528–1538, Sept. 1995.

[31] J. A. Sethian, "Fast marching methods," *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.

[32] M.-G. Vallet, *Génération de maillages éléments finis anisotropes et adaptatifs*. PhD thesis, Université Pierre et Marie Curie, Paris VI, France, 1992.

[33] P.-L. George and H. Borouchaki, *Delaunay Triangulation and Meshing. Applications to Finite Elements*. Paris: Hermès, 1998.

[34] Cyberware, "Sample models." http://www.cyberware.com/samples/index.html.

[35] R. Taghavi, "Automatic, parallel and fault tolerant mesh generation from CAD," *Engineering with Computers*, vol. 12, pp. 178–185, Dec. 1996.

[36] R. Schneiders, "A grid-based algorithm for the generation of hexahedral element meshes," *Engineering with Computers*, vol. 12, pp. 168–177, 1996.

[37] R. J. Smith and M. A. Leschziner, "A novel cartesian grid method for complex aerodynamic CFD applications," in *Proceedings of the 5th international conference on numerical grid generation in computational field simulations*, 1996.

[38] M. J. Aftosmis, M. J. Berger, and J. E. Melton, "Robust and efficient Cartesian mesh generation for component-based geometry," in *35th AIAA Aerospace Sciences Meeting*, no. AIAA–97–0196, (Reno, NV), Jan. 1997.

[39] R. Schneiders, "Octree-based hexahedral mesh generation," *Int. J. of Comp. Geom. & Applications*, vol. 10, no. 4, pp. 383–398, 2000.

[40] W. Oaks and S. Paoletti, "Polyhedral mesh generation," in *Ninth International Meshing Roundtable*, (New Orleans, LA), pp. 57–67, Sandia National Laboratories, Oct. 2000.

[41] L. Maréchal, "A new approach to octree-based hexahedral meshing," in *Tenth International Meshing Roundtable*, (Newport Beach, CA), pp. 209–221, Sandia National Laboratories, Oct. 2001.

[42] K. S. Walton, S. E. Benzley, and J. Shepherd, "Sculpting: An improved inside-out scheme for all-hexahedral meshing," in *Eleventh International Meshing Roundtable*, (Ithaca, NY), pp. 153–160, Sandia National Laboratories, Sept. 2002.

[43] J. Zhu and T. Blacker, "Overcoming Cartesian grid generation obstacles," in *7th International Conference on Numerical Grid Generation*, Sept. 2000.

[44] T. Blacker, "Meeting the challenge for automated conformal hexahedral meshing," in *Ninth International Meshing Roundtable*, (New Orleans, Louisiana), pp. 11–19, Sandia National Laboratories, Oct. 2000.

[45] Y. Sirois, J. Dompierre, M.-G. Vallet, P. Labbé, and F. Guibault, "Progress on vertex relocation schemes for structured grids in a metric space," in *8th International Conference on Numerical Grid Generation*, (Honolulu, USA), pp. 389–398, June 2002.

[46] K.-F. Tchon, J. Dompierre, and R. Camarero, "Conformal refinement of all-quadrilateral and all-hexahedral meshes according to an anisotropic metric," in *Eleventh International Meshing Roundtable*, (Ithaca, NY), pp. 231–242, Sandia National Laboratories, Sept. 2002.