Simplification of Tetrahedral Meshes with Accurate Error Evaluation

P. Cignoni, D. Costanza, C. Montani, C. Rocchini, R. Scopigno
Istituto Scienza e Tecnologia dell'Informazione – Consiglio Nazionale delle Ricerche*

Abstract

The techniques for reducing the size of a volume dataset by preserving both the geometrical/topological shape and the information encoded in an attached scalar field are attracting growing interest. Given the framework of incremental 3D mesh simplification based on edge collapse, the paper proposes an approach for the integrated evaluation of the error introduced by both the modification of the domain and the approximation of the field of the original volume dataset. We present and compare various techniques to evaluate the approximation error or to produce a sound prediction. A flexible simplification tool has been implemented, which provides different degree of accuracy and computational efficiency for the selection of the edge to be collapsed. Techniques for preventing a geometric or topological degeneration of the mesh are also presented.

Keywords: Simplicial Complexes, Mesh Simplification, Volume Visualization, Unstructured Grids

1 Introduction

Many papers have been published over the last few years concerning the simplification of simplicial complexes. Most of them concern the simplification of 2D simplicial meshes embedded in 3D space, hereafter called *surfaces*. Only a minor subset are concerned with 3D simplicial decompositions, hereafter called *meshes*. In particular, we consider in this paper the class of irregular volume datasets, either convex or non convex, with scalar field values associated with the vertices of the tetrahedral cells. Let $\mathcal{D} = (V, \Sigma, \Phi)$ be our dataset where V is a set of n vertices, $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ is a tetrahedralization of m cells with vertices in V, and $\Phi = \{\phi_1, \phi_2, \ldots, \phi_m\}$ is a set of functions such that each function ϕ_i is defined over cell σ_i of Σ. All functions of Φ are linear interpolants of the scalar field known at the vertices of V. Given an irregular dataset $\mathcal{D}$, the term *simplification* refers to the problem of building an approximate representation $\mathcal{D}'$ of $\mathcal{D}$ with a smaller size, built by choosing a set of vertices V' (usually $V' \subset V$) and a new triangulation Σ' of V' that covers [almost] the same domain. This problem has some similarities with scattered data interpolation and thinning techniques [10], but the main problem of these approaches is that the shape of the data domain is not taken into account (erroneous interpolation between unconnected data becomes possible).
Surface/mesh simplification can be driven by two different objectives: producing a more compact mesh which is sufficiently similar in terms of visual appearance, or to produce a model which satisfies a given accuracy. In the first case the main goal is to reduce visualization time. In the second case, special emphasis is given to data quality and representation accuracy; this is often the case for scientific visualization applications, where the user requires measurable and reliable data quality.
Our goal is therefore to design and evaluate different tetrahedral mesh simplification methods in the framework of scientific visualization applications, with a special commitment to the quality of the mesh obtained (considering both geometry and the associated scalar field). The approach adopted lies in the general class of *incremental simplification methods*: simplification proceeds through a sequence of local mesh updates which, at each step, reduces the mesh size and [monotonically] decreases the approximation precision. Specifically, we adopt an approach based on iterative *edge collapse*. The main contributions of this paper are as follows:

- The geometric/topology correctness of the mesh produced. Topology and geometry are preserved, and checks are introduced to prevent possible inconsistencies in the simplified mesh (cell flipping, degeneration, self-intersections);
- The evaluation of the approximation error. We introduce a characterization of the approximation error, using two conceptually different classes of *domain*-error and *field*-error, and propose a new approach for the integrated evaluation of *domain*-error and *field*-error;
- Different criteria to *predict* and *evaluate* the approximation error are proposed and compared with the direct evaluation approach. In particular, we propose an extension of the *quadrics error metric* to the case of field error evaluation on 3D meshes;
- Finally, the computational efficiency of the techniques proposed is evaluated empirically on sample datasets.

The work takes also into account the constraints introduced when the goal is the construction of a multiresolution representation of the dataset.

2 Related Works

Many different simplification methods have been developed for the simplification of surfaces. These methods generally try to select the smallest set of points approximating a dataset within a given error. A detailed review of these algorithm is beyond the scope of this document, and for a survey on this subject see [12]. Very briefly, we can summarize by saying that effective solutions to the simplification problem have often been obtained through incremental techniques, based on either a *refinement* strategy (refine a coarse representation by adding points [11]) or a *decimation (or coarsening)* strategy (simplify the dataset by removing points [21, 17]).

*CNR Research Park, S. Cataldo - 56100 Pisa, ITALY. Email:{cignoni | montani | rocchini}@iei.pi.cnr.it, roberto.scopigno@cnuce.cnr.it

Many of these techniques could be extended to the 3D case, i.e. to volume data simplification. In the following we review the specific results regarding tetrahedral meshes. We do not consider here the many lossless compression solutions that have appeared in the last few years, because the focus here is on simplification and multiresolution.

2.1 Refinement Strategies

Hamann and Chen [16] adopted a refinement strategy for the simplification of tetrahedral convex complexes. Their method is based on the selection of the most important points (based on curvature) and their insertion into the convex hull of the domain of the dataset. When a point is inserted into the triangulation, local modifications (by face/edge swapping) are performed in order to minimize a local approximation error.

Another technique, based on the Delaunay refinement strategy, was proposed by Cignoni *et al.* [5]; here the vertex selection criterion was to choose the point causing the largest error with respect to the original scalar field. This technique was successively extended in [6] to the management of non-convex complexes obtainable by the deformation of convex domains (e.g. curvilinear grids).

The refinement-based strategy was also used by Grosso and Greiner [15]. Starting from a coarse triangulation covering the domain, a hierarchy of approximations of the volume is created by a sequence of local adaptive mesh refinement steps. A very similar approach based on selective refinement, but limited to regular datasets, was presented in [24].

All the techniques based on the refinement strategy share a common problem: the domain of the dataset has to be convex (or at least it has to be defined as a warping of a regular computational grid [6]). The reason lies in the intrinsic difficulty in fulfilling strict geometric constraints while refining a mesh (from coarse to fine) and using just the vertices of the dataset.

2.2 Decimation Strategies

Renze and Oliver in [19] proposed the first 3D mesh decimation algorithm based on vertex removal. Given a tetrahedral complex Σ, they evaluate the internal vertices of the mesh for removal, in random order. The re-triangulation of the hole left by the removal of a vertex v is done by building the Delaunay triangulation Σ_v of the vertices adjacent to v, and searching for, if it exists, a subset of the tetrahedra of Σ_v whose (d-1)-faces match the faces of Σ. If such a subset does not exist the vertex is not removed. The latter condition may very often hold if the original complex is not a Delaunay one. This method neither measures the approximation error introduced in the reduced dataset, nor tries to select the vertex subset in order to minimize the error.

Popovic and Hoppe [18] have extended the Progressive Meshes (PM) algorithm [17], a surface simplification strategy based on edge-collapse, to the management of generic simplicial complexes. However, their work is very general, and it does not consider in detail the impact on the approximation accuracy of a possible scalar field associated with the mesh. The PM approach has been recently extended by Staadt and Gross [22]. They introduce various cost functions to drive the edge-collapsing process and present a technique to check (and prevent) the occurrence of intersections and inversions of the tetrahedra involved in a collapse action. The approach is based on a sequence of tests that guarantees the construction of a robust and consistent progressive tetrahedralization. A simplification technique based on iterative edge collapsing has also been sketched by Cignoni *et al.* in [6].

A technique based on error-prioritized tetrahedra collapse was proposed by Trotts *et al.* [23]. Each tetraedron is weighted based on a predicted increase in the approximation error that would result after its collapse; tetraedral cell collapse is implemented via three edge collapses. The algorithm gives an approximate evaluation of the *scalar field error* introduced at each simplification step (based on the iterative accumulation of *local evaluations*, following the approach proposed by Bajaj *et al.* for the simplification of 2D surfaces [3], which gives an overestimation of the actual error). The mesh degeneration caused by the modification of the (possibly not convex) mesh boundary and the corresponding error are managed by forcing every edge collapse that involves a boundary vertex to be performed on the boundary vertex, and avoiding the collapse of corner vertices. This approach preserves the boundary in the case of regular datasets, but cannot be used to decimate the boundary of a dataset with a more complex domain (e.g. non rectilinear or not convex, as occurs frequently on irregular datasets).

3 Incremental Simplification via Edge Collapse

We adopt an iterative simplification approach based on edge collapse: at each iteration, an edge is chosen and collapsed. The atomic edge collapse action is conceived here as a simple vertex unification process. Given a maximal[1] 3-simplicial complex Σ and an edge e connecting two vertices v_s and v_d (the *source* and *destination* vertices), we impose that v_s becomes equal to v_d and we consequently modify the complex[2]. This operation causes the edge $(v_s - v_d)$ to collapse to the point v_d and all the tetrahedra incident on the edge $(v_s - v_d)$ to collapse to triangles. Again, these new triangles are unified with the corresponding identical triangles contained in Σ.

This simplification process is always limited to a local portion of the complex: the set of simplices incident in v_s or v_d. We introduce the following terminology: given a edge collapse $e = (v_s, v_d)$ we define: $D(e)$ the set of *deleted* tetrahedra incident in e; $M(e)$ the set of *modified* tetrahedra, i.e. those tetrahedra incident in v_s but not in v_d. Therefore, an edge collapse step results in some *modified* and some *deleted* tetrahedra. The geometric information is simply updated by the unification of the vertex coordinates. The topology update is somehow slightly more complex; relations TV, VT, EV and VE have to be updated after each atomic collapse action.

The order in which edges are collapsed is critical with respect to the simplified mesh accuracy. The result of the iterative simplification is a sequence $\Sigma_0, \Sigma_1, \ldots, \Sigma_i, \ldots, \Sigma_n$ of complexes [17, 4]. When the goal is the production of a high quality multiresolution output, the approximation error should increase slowly and smoothly. Analogously to many other simplification approaches, we adopt a heap to store the edges which are to be collapsed. At the beginning, all the edges are inserted in the heap and sorted with respect

[1] I.e., a complex which does not contain dangling non-maximal simplices.

[2] The position and the field value of the vertex v_d can also be changed, obtaining the so-called *interpolatory edge collapse*. We do not adopt this approach because the choice of the v_d optimal location is not easy with most of the error evaluation criteria.

to an estimated error, known in the following sections as the *predicted error* (see Section 5). The edges in the heap are *oriented*, that is we have both the oriented edges (v_j, v_i) and (v_i, v_j) in the heap, because they identify different collapses. For each simplification step: the edge e with the lowest error is extracted from the heap; the collapse of e is tested, checking the topological and geometric consistency of the mesh after the collapse. If the geo-topological checks are verified, the following actions are performed:

- the tetrahedra in $D(e)$ are deleted;
- the topology relation TV is updated, i.e. v_s is replaced with v_d in all tetrahedra $\sigma \in M(e)$;
- the VT relation is updated on vertex v_d: $VT(v_d) = VT(v_d) \cup M(e) \setminus D(e)$;
- the VE relation is updated by setting $VE(v_d) = VE(v_d) \cup VE(v_s) \setminus \{(v_s, v_d)\}$;
- the EV relation is updated by substituting v_s with v_d on all the edges in $VE(v_s)$;
- a new estimated error is evaluated for all former edges $VE(v_s)$ in the heap.

Otherwise, we reject the edge collapse and continue with the next edge in the heap.
Consistency checks are evaluated before updating the mesh. There are two classes of consistency conditions: *topological* and *geometrical*. The first one ensures that the edge collapse will not change the topological type of our complex. The second one ensures geometric consistency, i.e. that no self-intersecting or badly-shaped (e.g. slivery) tetrahedra are introduced by the edge collapse.

3.1 Topology Preserving Edge Contraction

Given an edge collapse, a set of necessary and sufficient conditions that preserves the topological type of our complex has recently been proposed in [9]. We adopted this approach in our simplification system to guarantee the topological correctness of the simplification process.
Let $St(\sigma)$ be the set of all the co-faces of σ, i.e. $St(\sigma) = \{\tau \in \Sigma \mid \sigma \text{ is a face of } \tau\}$. Let $Lk(\sigma)$ be the set of all the faces belonging to $St(\sigma)$ but not incident on σ (i.e. the set of all the faces of the co-faces of σ disjoint from σ).
Let Σ_i be a 3-simplicial complex *without boundary*, $e = (v_s, v_d)$ an edge of the complex, and Σ_{i+1} the complex after the collapse of edge e. According to [9], the following statements are equivalent:

1. $Lk(v_s) \cap Lk(v_d) = Lk(e)$
2. Σ_i, Σ_{i+1} are homeomorphic

It is therefore sufficient to check statement (1) to prove that statement (2) holds, that is to ensure the topological correctness of the current simplification step (see Figure 1).

If Σ_i is a complex *with boundary* (which is the usual case), we can go back to the previous case by ideally adding a dummy vertex w and its cone of simplices to the boundary (i.e. we add a dummy simplex for each boundary 2-face of Σ_i).The insertion of w and the corresponding simplices allows us to also manage the boundary faces of Σ_i with the previous checking rule.

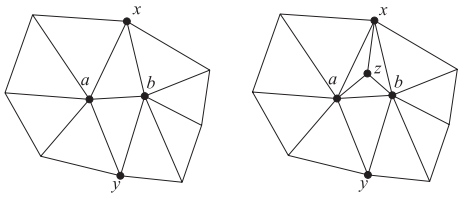

Figure 1: Topology checks: in the example on the left, the condition $Lk(a) \cap Lk(b) = \{x, y\} = Lk(ab)$ indicates a valid collapse. Conversely, an invalid collapse is detected in the configuration on the right because $Lk(a) \cap Lk(b) = \{x, y, z, zx\} \neq Lk(ab)$.

3.2 Preserving Geometric Consistency

Three possible dangerous situations should be prevented in the simplification process:

- tetrahedra *inversion*;
- generation of *slivery/bad shaped* tetrahedra,
- *self-intersection* of the mesh boundary.

The first two situations are easy to check. In the first case it is sufficient to check that each *modified* tetrahedron in $M(e)$ preserves the original orientation (the first vertex sees the other three ones counterclockwise), or in other words the cell volume does not become negative.
In the second case, we reject every collapse that produces one or more tetrahedra in $M(e)$ having an aspect ratio smaller than a given threshold ρ. Note that, in order to allow the simplification of meshes which contain slivery tetrahedra, it is useful to allow the collapse of an edge also if the aspect ratio of modified tetrahedra improves after the collapse.
The detection of self-intersections is the most complex subtask, because this is the only case where the effects of an edge collapse can be *non-local*. After an edge collapse, some boundary faces that are topologically non-adjacent but geometrically close can become self-intersecting. The intrinsic non-locality of this kind of degeneration makes it difficult to efficiently and correctly prevent it without using auxiliary structures. To speedup self-intersection checks (a quadratic problem in its naive implementation) a uniform grid [1] could be adopted, to store all the vertices of the current boundary of the mesh. For each edge collapse (v_s,v_d) that involves a boundary edge, we should check whether after the collapse, all the edges on the boundary incident in v_d do not intersect the mesh boundary. If an intersection is found, the collapse is aborted and the original state of the mesh before the collapse is restored.

4 Error Characterization and Evaluation

When an atomic simplification action is performed, a new mesh Σ_{i+1} is generated from Σ_i with, in general, a higher approximation error. The approximation error can be described by using two measures: the *domain* error and the *field* error.

4.1 Domain Error

The collapse of an edge lying on (or adjacent to) the boundary of the mesh can cause a deformation of the boundary of the mesh. In other words, Σ_i and Σ_{i+1} can span different

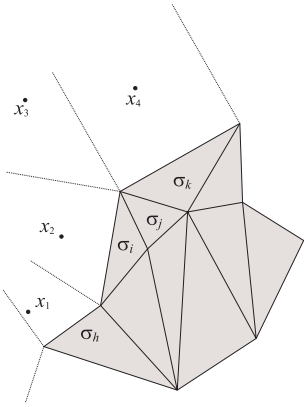

Figure 2: Every point which is not contained in the complex is assigned to one or to a group of cells, e.g.: $V_{\mathcal{D}}(x_1) = \{\sigma_h\}$, $V_{\mathcal{D}}(x_2) = \{\sigma_i\}$, $V_{\mathcal{D}}(x_3) = \{\sigma_i, \sigma_j, \sigma_k\}$ $V_{\mathcal{D}}(x_4) = \{\sigma_k\}$.

domains. This problem has been ignored in many previous solutions. A correct measure of the *domain error* can be obtained by measuring the symmetric Hausdorff distance between the boundary surface of the input mesh Σ and the boundary of each intermediate simplified mesh Σ_i. A function for measuring the approximation between two surfaces can be efficiently implemented [7]. But the overheads become excessive if this function is used to evaluate the accuracy of each simplification step that involves a boundary vertex.
A more efficient solution can be implemented by deploying the locality of the simplification action (see Subsection 4.4).

4.2 Field Error

The approximation of the original scalar field defined on Σ with the under-sampled field defined by the simplified complex Σ' causes another type of error.
Let $\mathcal{D}' = (V', \Sigma', \Phi')$ be our approximate representation. Assuming that the two domains Ω and Ω' of Σ and Σ' coincide, we can measure the error ε_f introduced in the representation of the original field as follows:

$$\epsilon_f(\mathcal{D}, \mathcal{D}') = \max_{x \in \Omega}(|\Phi(x) - \Phi'(x)|)$$

But measuring only the maximum difference between the two fields does not give a precise estimation. In fact it can happen that a very small modification of the shape of a single tetrahedron with a large field variation cause a very large error, even if the incorrect volume is almost negligible. For this reason is also useful to measure the average square error ϵ_f^q over the whole domain of the mesh:

$$\epsilon_f^q(\mathcal{D}, \mathcal{D}') = \frac{1}{|\Omega|} \int_{\Omega} |\Phi(x) - \Phi'(x)|^2 \, dv$$

If Ω and Ω' differ, and this is the case when the simplification of the boundary of the mesh is allowed, we have to reciprocally extend the domains Ω' and Ω to compare Φ' and Φ in a common space. The main problem is how to evaluate the field of the points belonging to Ω but not to Ω', and viceversa. A possible solution may be to adopt the following definition of the field error:

$$\epsilon_f(\mathcal{D}, \mathcal{D}') = \max(\epsilon'_f(\mathcal{D}, \mathcal{D}'), \epsilon_f^{\mathcal{D}}(\mathcal{D}, \mathcal{D}'), \epsilon_f^{\mathcal{D}'}(\mathcal{D}, \mathcal{D}'))$$

where:

$$\epsilon'_f(\mathcal{D}, \mathcal{D}') = \max_{x \in \Omega \cap \Omega'}(|\Phi(x) - \Phi'(x)|)$$

$$\epsilon_f^{\mathcal{D}}(\mathcal{D}, \mathcal{D}') = \max_{x \in \Omega \setminus \Omega',\ \sigma \in V_{\mathcal{D}'}(x)}(|\Phi(x) - \phi'_\sigma(x)|)$$

$$\epsilon_f^{\mathcal{D}'}(\mathcal{D}, \mathcal{D}') = \max_{x \in \Omega' \setminus \Omega,\ \sigma \in V_{\mathcal{D}}(x)}(|\phi_\sigma(x) - \Phi'(x)|)$$

where $V_{\mathcal{D}}(x)$ is the set of cells of Σ that have the minimum distance to the point x (see Figure 2). Note that a set of cells can have the same distance to the same point x (all those cells incident in a given boundary vertex are associated with the same external space partition). In Figure 2 we show a 2D example of some $V_{\mathcal{D}}()$ sets. There is a strict relation between this partitioning scheme and the Voronoi Diagram [2] of the simplicial complex Σ.

4.3 Walking in the Field/Domain Error Space

One important characteristic of a simplification method is the quality of the multiresolution output produced. The error increase should be as smooth as possible in order to allow the maximal flexibility in the extraction of models at different resolutions.
Many incremental simplification methods store the estimated approximation errors in a heap. But following our approach, we have an error pair (field and domain) for each edge collapse. The corresponding 2D error space (field error on the X axis, domain error on the Y axis) is shown in Figure 3. Let us suppose that the user fixes a pair of thresholds $(\epsilon_f^{max}, \epsilon_d^{max})$ for the field and the domain errors. During the simplification process the error moves on a polyline that [hopefully] interconnects the origin with the user-specified maxima. Suppose that the current mesh has an error (ϵ_d, ϵ_f), shown in Figure 3 with a circled dot. The other dots in Figure 3 denote the predicted error pairs for every possible edge-collapse. How do we choose the next edge to collapse?
Giving priority to edges with either minimal domain errors or minimal field errors may be a mistake (for example, error pair a in the figure represents the edge with minimal field error, but it has a very high domain error; analogously, error pair b has a minimal domain error but a great field error).
A common approach is to use a weighted sum of different error estimates, $\varepsilon = w_1\varepsilon_1 + .. + w_k\varepsilon_k$ [17, 14, 22]. As an example, a multivariate error cost evaluation was proposed in [22] in the framework of a tetrahedra mesh simplification solution based on edge-collapse. This measure is a weighted sum of three components:

$$\varepsilon = w_1\varepsilon_{grad}(e_i) + w_2\varepsilon_{vol}(e_i) + w_3\varepsilon_{equi}(e_i)$$

which evaluate the edge *gradient*, the change in *volume* of the updates mesh section and the *average length* of the edges affected by the collapse. But a solution based on a weighted sum has a drawback: coming back to the example in Figure 3, error pair c might have the best weighed sum but also an excessively high field error. Therefore, we do not consider criteria based on weighted sums adequate for the integrated *field* and *domain* error evaluation.
A better solution is defined as follows. Given a normalized error space with weights w_d and w_f defined such that:

$$w_d\epsilon_d^{max} = w_f\epsilon_f^{max},$$

we can choose the edge e that has the *smallest* error ε defined as follows:

$$\varepsilon = \min_{e \in Heap}(\ \max(w_d\epsilon_d(e), w_f\epsilon_f(e))). \qquad (1)$$

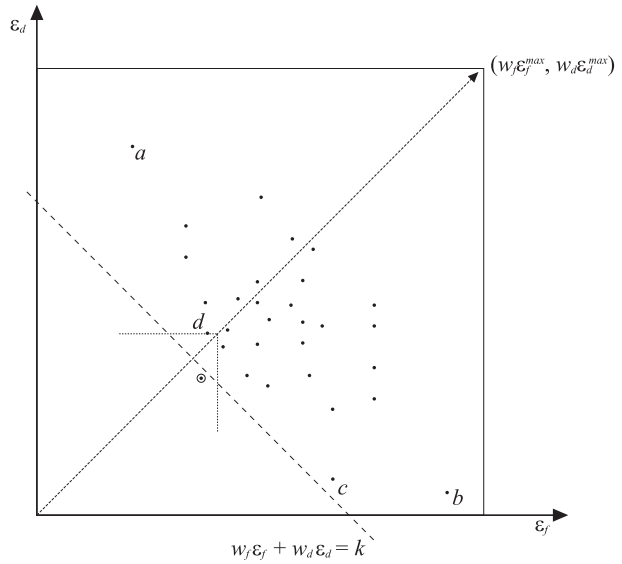

Figure 3: The domain/field error space. During the simplification process the error walks from the origin towards a user-specified maximal error point.

This strategy can be intuitively interpreted as choosing at each step the first error pair (e.g. point d in Figure 3) that is enclosed by a square which grows along the line joining the origin of the error space with point $(\epsilon_d^{max}, \epsilon_f^{max})$.
The same approach can obviously be extended to treat the case of error evaluation functions which consider k variables. When we want to produce (and use) a multiresolution model, it is also useful if *both* errors can be identified using a single value. In this case, it should exist a precise relation between this value and the real field and warping errors.

4.4 Efficient Error Evaluation

A tool for the correct evaluation of the accuracy of a simplified mesh, taking into account the *field* and *domain* errors, has been developed [8] following an approach similar to the one used for the evaluation of surface simplification solutions [7]. It applies Montecarlo sampling on the domains of the original and simplified meshes, evaluating on each sample the relative field difference. The Hausdorff distance between the two domains is evaluated by the same technique of [7]. However, due to performance reasons, this approach can only be applied as a post-processing step to evaluate post-mortem the quality of the simplified mesh. Conversely, the following subsections introduce two evaluation rules that are simple enough to be used during the simplification process.

Efficient Field Error The computation of the *field* error can be simplified by evaluating the difference between the two scalar fields only on the vertices of the original complex:

$$\epsilon_f^*(\mathcal{D}, \mathcal{D}') = \max_{x \in V}(|\mathcal{F}(x) - \mathcal{F}'(x)|)$$

To easily compute ϵ^* during the simplification process, we need to maintain for each tetrahedron $\sigma \in \Sigma'$ a corresponding subset of deleted points $\{v_i\}$ such that: v_i lies inside σ, or v_i is associated with σ in the sense of Subsection 4.2 (see the definition of the $V_{\mathcal{D}}(x)$ set). A similar approximation has already been used in [6], and has been extended here by taking into account the removed vertices which are external to Ω'. After each collapse the vertices associated to the modified tetrahedra are redistribute according the local modifications in order to maintain the ϵ^* error.

To improve the accuracy of the estimation of the ϵ_f error we add a small number of random samples inside each tetrahedron of the original mesh, and evaluate the field difference also on these samples. To limit the time and memory overhead introduced by this technique we have found that it is convenient to add points proportionally to the field variation of the original meshes.

Efficient Domain Error A sufficiently good estimate of the domain error can be obtained by using the following approximation of the Hausdorff distance:

$$\epsilon_d^*(\Omega, \Omega') = \max_{x \in V - V',\ x \notin \Omega'} d(x, \Omega')$$

that is evaluated for all the removed vertices x which are external to the domain Ω'. This approximation can be computed efficiently during the simplification process storing for each boundary face of Σ', the list of the corresponding removed vertices not contained in Ω' as described in [4].

5 Error Prediction

For each step in the simplification process, we need to choose the edge whose collapse causes the minimal error increase, according to the error definition 1 introduced in Subsection 4.3. A heap is used to hold error-sorted edges. Therefore, we need to know in advance what error is introduced by a single edge collapse. This can be done in two different manners:

- **exact evaluation**: the collapse can be simulated on each oriented edge of the complex, producing an evaluation of the approximation error (according to the measures defined in Subsection 4.4);
- **approximate evaluation**: faster heuristics can be adopted, to estimate the error that will be introduced by the collapse.

Note that the use of an *approximate evaluation* in the error prediction phase (i.e. to update the heap) will not affect the actual evaluation of the error associated with each intermediate mesh Σ_i, which in any case is operated after each edge collapse by adopting the measures presented in Section 4.4.

The use of an approximate evaluation can reduce the running time substantially, because when we collapse an edge we need to update the heap error for all the edges incident on v_d, and the average number of adjacent edges is around 20-30. Moreover, in many cases it is more important to support the rapid choice of a *probable* good edge than to select the best edge according to an exact error estimate. An example is when a simplified mesh of a given size is needed, and we do not have a strict commitment to the approximation precision bound.
Three different error prediction approaches are described in the following, which can be used to choose the *probable best* edge.

Local Error Accumulation. This heuristic measures both the domain and the field errors *locally*, i.e. with respect to

the vertex that has been unified and removed in the current edge collapse action. These error estimates are then accumulated during the simplification process to give an approximate global estimate.

Gradient Difference. In order to estimate the error increase, we pre-compute the *field gradient* ∇_v at each vertex v of the input mesh. This can be done by computing the weighted average of gradients in all tetrahedra incident at v. The weight to be associated with the contribution of each tetrahedron σ is given by the solid angle of σ at v. Then, for each vertex v in the mesh, we search the vertex w, among those adjacent to v, such that the difference $\Delta\nabla_{v,w}$ between the gradient vectors ∇_v and ∇_w is minimal. Value $\Delta\nabla_{v,w}$ gives a rough estimate of how far from linear the field is in the neighborhood of v (in particular, on the edge (v,w) direction). The smaller $\Delta\nabla_{v,w}$ is, the smaller the expected error increase is if v is removed by collapsing it onto w. The value $(\Delta\nabla_{v,w} \cdot L(e))$, where $L(e)$ is the length of the edge to be collapsed, is therefore used as an estimate of the field error.
This solution is more precise and more complex in terms of space (because gradients have to be explicitly stored) than the one proposed in [22], which takes into account only the difference of the field values on the collapsed edge extremes.

Quadric Error. Another approximate measure can be defined by extending the *quadric error metric* introduced by Garland *et al.* [13]. This metric was proposed to measure the geometric error introduced on a surface during the simplification process. We use it to measure not only the domain error, but also the field error. The main idea of the quadric error metric is to associate a set of planes with each vertex of the mesh. The sum of the squared distances from a vertex to all the planes in its set defines the error of that vertex. Initially each vertex v is associated with the set of planes passing through the faces incident in v. When, for each collapse of a given v_s onto v_d, the resulting set of planes is the union of the sets of v_s and v_d. The most innovative contribution in [13] (and the main improvement over [20]) is that these sets of planes are not represented explicitly. Let $\mathbf{n}^\top\mathbf{v} + d = 0$ be the equation representing a plane, where $\mathbf{n}$ is the unit normal to the plane and d its distance from the origin. The squared distance of a vertex v to this plane is given by:

$$D = (\mathbf{n}^\top\mathbf{v} + d)^2 = \mathbf{v}^\top(\mathbf{n}\mathbf{n}^\top)v + 2d\mathbf{n}^\top\mathbf{v} + d^2$$

According to [13] we can represent this quadric Q, which denotes the squared distance of a plane to a vertex, as:

$$Q = (A, b, c) = (\mathbf{n}\mathbf{n}^\top, d\mathbf{n}, d^2)$$

$$Q(\mathbf{v}) = \mathbf{v}^\top A\mathbf{v} + 2b^\top\mathbf{v} + c$$

The sum of a set of quadrics can easily be computed by the pairwise component sum of their terms, therefore for each vertex we maintain only the quadric representing the sum of the squared distances of all the planes implicitly associated with that vertex, which is just ten coefficients.

In the case of 3D mesh simplification the domain error can be easily estimated by providing a quadric for each boundary vertex of the 3D mesh. Quadrics can also be used to measure the field error. In this case we associate with each vertex v a set of linear functions ϕ_i (that is, the linear functions associated with the cells incident in v), and we measure the sum of squared differences between the linear functions and the field on v. Each linear function can be represented by $\phi(\mathbf{v}) = \mathbf{n}^\top\mathbf{v} + d$ where, analogously to the geometric case, $\mathbf{n}$ is a 3D vector (not unitary in this case and representing the gradient of the field) and d is a constant (the value of the scalar field in the origin).
The management of this kind of quadric is therefore exactly the same as the previous case, but with a slightly different meaning. In this case the quadric represents the sum of squared differences between the linear functions and the field on v. In this way with two quadrics, one for the field and one for the domain error, we can have a measure of both errors, which are then composed as described in Subsection 4.3.

6 Results

We have implemented and tested some of the possible combinations of the error evaluation strategies proposed above. We present in the following some results concerning the combinations of different techniques for the error prediction phase and the post-collapse error evaluation phase:

LN : we use the **L**ocal error accumulation for the error prediction phase, and the approximation error obtained after the collapse is **N**ot evaluated (that is, simplification is driven by the mesh reduction factor).

GN : we use the **G**radient Difference for the error prediction phase, and the approximation error obtained after the collapse is **N**ot evaluated.

QN : we use the **Q**uadric measure of error for the error prediction phase, and the approximation error obtained after the collapse is **N**ot evaluated.

BF : **B**rute **F**orce, we apply a full simulation of all possible collapses, using the efficient error evaluation described in Section 4.4.

BFS : **B**rute **F**orce with added **S**amples, a set of random sample points are added in each tetrahedron of the original mesh; the domain and field errors are evaluated on these sample points and on the original mesh vertices.

These solutions represent various mixes of accuracy and speed. The last one (BFS) is the slowest but the most accurate (especially if a very accurate management of the *domain error* is requested). But its running times are so high (6x - 10x with respect to the running time of the BF method), that the improvement in terms of precision does not justify its adoption in many applications. The first three techniques (LN, GN, QN) do not precisely evaluate the error during the simplification, and therefore we cannot guarantee the mesh approximation to be lower than the given threshold. This allows much faster and lighter algorithms, but also prevents the generation of a high quality multiresolution output.

We have chosen four datasets to benchmark the presented algorithms: **Fighter** (13,832 vertices, 70,125 tetrahedra) which is the result of an air flow simulation over a jet fighter, courtesy of Nasa; **Sf5** (30169 vertices, 151173 tetrahedra) that represents wave speed in the simulation of a quake in the San Fernando valley, courtesy of Carnegie Mellon University (http://www.cs.cmu.edu/∼quake); **Turbine Blade** (106,795 vertices, 576,576 tetrahedra), dataset courtesy of Avs Inc. (tetrahedralized by O. G. Staadt).

Fighter Dataset (input mesh: 13,832 vertices 70,125 tetrahedra)														
vert.	input	BF			BFS			LN		GN		QN		
	%	ϵ_f	ϵ_f^q	time	ϵ_f	ϵ_f^q	time	ϵ_f	ϵ_f^q	ϵ_f	ϵ_f^q	ϵ_f	ϵ_f^q	time
6,916	50	40.58	1.34	61.0	17.61	1.54	654	47.46	1.42	52.11	1.65	66.70	1.63	27.0
2,766	20	65.34	2.58	88.9	29.27	2.28	1155	54.17	2.55	66.13	1.85	60.99	2.23	39.8
1,383	10	65.34	2.70	99.9	39.13	2.48	1395	50.87	3.15	67.54	1.99	69.20	2.41	45.2

Table 1: Results of the simplification of the Fighter mesh. Errors are expressed as a percentage of the field range, times are in seconds.

The numerical results are presented in Tables 1, 2, and 3. The code was run on a 450MHz PII personal computer with 512MB RAM and running WinNt. Various mesh sizes are shown in the tables, out of the many different resolutions produced. The tables show the processing time in seconds of each different algorithm[3], and the actual approximation error of each simplified mesh. The errors reported in the tables are the maximum error ϵ_f and the mean square error ϵ_f^q, which have been evaluated using the Metro3D tool [8]. Metro3D performs a uniform sampling on the high resolution dataset (i.e. the number of samples taken for each cell is proportional to the cell volume); for each sample point it measures the difference between the fields values interpolated on the high resolution and the simplified mesh.

Some different simplified representations of the Turbine Blade dataset, produced using the different error evaluation heuristics, are shown in Figure 4 in Color Plates. The figure also shows how complex simplification is: for example, the Turbine dataset contains some very small regions where the field values change abruptly (near the blue blades the field spans over the 70% of the whole field range). This means that a slightly incorrect collapse action, localized in one of these these regions, may introduce a very large maximal error.

Having introduced a combined field and domain error evaluation allows us to simplify meshes with very complex domain, preserving its boundary with high accuracy. See an example in Figure 5 in Color Plates.

7 Conclusions

The main results that we have presented consist of the definition of a new methodology to measure the approximation error introduced in the simplification of irregular volume datasets, used to prioritize potential atomic simplification actions. Given the framework of the incremental 3D mesh simplification based on edge collapse, the paper proposes an approach for the integrated evaluation of the error introduced by both the modification of the domain and the approximation of the field of the original volume dataset. These two different errors, the *domain error* and *field error*, are used as components of a unified error evaluation function. Using a multi-variate error evaluation function is not a new idea, but we have shown that the adoption of a simple weighted sum can lead to a non optimal priority selection of the elements to be collapsed. A new error function is devised by considering the two-dimensional *(domain, field)* error space and introducing an original heuristic.
In this framework, we present and compare various techniques to precisely evaluate the approximation error or to produce a sound prediction. These solutions represent various mixes of accuracy and speed in the choice of the edge to be collapsed. They have been tested on some common datasets, measuring their effectiveness in terms of simplification accuracy and time efficiency. Moreover, techniques for preventing geometric or topological degeneration of the mesh have also been presented.

After testing these simplification techniques on a set of different datasets, one could feel that the problem of accurate simplification of a tetrahedral mesh is harder than the simplification of standard 3D surfaces. In fact, for most meshes, obtaining high simplification rates introducing a low or negligible error is not easy, even if a slow but accurate error criterion is adopted. Conversely, there are many good techniques that can produce a drastic simplification of 2D surface meshes while maintaining a very good accuracy. This is probably due to the fact that a common habit is to compare the simplification of a standard 2D mesh (pure geometry) against the simplification of a 3D mesh supporting also a scalar field. A more correct comparison would be to consider the performances of simplification codes on 2D meshes which also have an attribute field attached (e.g. vertex colors). Analogously to the results obtained in this work, it has been demonstrated that in the latter case a drastic simplification cannot easily be obtained, unless the color field has a very simple distribution on the surface. Therefore, the quality of attribute-preserving simplification strongly depends on the distribution of the scalar attribute over the mesh and, at the same time, on the mesh structure. In many cases a drastic reduction cannot be obtained unless we decrease the accuracy constraint. Unfortunately, data accuracy is a more critical requirement in scientific visualization than in standard interactive computer graphics: when we visualize scientific results we must be sure that what we are seeing **is** correct and not only **seems** correct. For this reason we think that data simplification can be safely used in scientific visualization only if it is coupled with sophisticated dynamic multiresolution techniques that easily/efficiently allow to recover the original data when (and, hopefully, where and how) needed. In this way the user can safely exploit the advantages of simplification technology (less data to be rendered) because he is also able to use locally the original data on request (e.g. in small selected focus regions).

[3]Times of LN and GN techniques were not reported because they were obtained using a quick modification of the BF code; therefore, the corresponding times are not adequate for a fair comparison.

References

[1] V. Akman, W.R. Franklin, M. Kankanhalli, and C. Narayanaswami. Geometric computing and uniform grid technique. *Computer-Aided Design*, 21(7):410–420, Sept. 1989.

[2] F. Aurenhammer. Power diagrams: Properties, algorithms and applications. *Siam J. Comput.*, 16(1):78–96, February 1987.

sf5 Dataset (input mesh: 30,169 vertices, 151,173 tetrahedra)														
	% orig.	BF			BFS			LN		GN		QN		
vert.	vert.	ϵ_f	ϵ_f^q	time	ϵ_f	ϵ_f^q	time	ϵ_f	ϵ_f^q	ϵ_f	ϵ_f^q	ϵ_f	ϵ_f^q	time
15,084	50	9.93	0.21	127.55	2.51	0.20	419.47	9.93	0.20	8.59	0.45	23.95	0.23	74
6,033	20	11.55	0.37	202.39	5.53	0.37	895.19	11.55	0.35	18.25	0.82	34.27	0.43	101
3,016	10	11.32	0.53	234.99	5.65	0.58	1208.15	12.46	0.49	25.52	1.06	35.29	0.67	110
1,508	5	13.10	0.74	264.87	6.85	0.69	1538.14	15.95	0.68	39.63	1.23	51.78	1.29	114
603	2	22.11	1.19	296.82	9.99	1.57	1945.57	16.43	1.19	51.80	3.86	49.67	1.60	118

Table 2: Results of the simplification of the sf5 mesh. Errors are expressed as a percentage of the field range, times are in seconds.

Turbine Dataset (input mesh: 106,795 vertices, 576,576 tetrahedra)														
vert.	input	BF			BFS			LN		GN		QN		
	%	ϵ_f	ϵ_f^q	time	ϵ_f	ϵ_f^q	time	ϵ_f	ϵ_f^q	ϵ_f	ϵ_f^q	ϵ_f	ϵ_f^q	time
53,397	50	71.3	0.10	587.3	78.3	0.04	1117.7	78.7	0.23	78.7	0.09	74.3	1.50	330.2
21,359	20	78.3	0.63	954.5	78.7	0.18	2859.9	78.7	0.49	78.6	0.39	81.7	2.85	459.2
10,679	10	78.7	0.58	1098.9	78.1	0.38	4270.2	78.7	0.79	85.7	2.40	80.6	4.31	511.8
5,339	5	78.7	0.86	1193.2	78.7	0.71	5120.9	74.7	1.04	97.3	7.21	90.9	6.54	539.7
2,135	2	76.1	1.42	1276.4	24.1	1.25	5222.0	74.4	2.78	97.3	8.59	97.3	10.26	545.3
1,067	1	81.3	2.92	1318.8	68.6	4.97	6742.2	80.0	9.14	97.3	10.74	93.2	11.71	549.3

Table 3: Results of the simplification of the Turbine mesh. Errors are expressed as a percentage of the field range, times are in seconds.

[3] C. L. Bajaj and D.R. Schikore. Error bounded reduction of triangle meshes with multivariate data. *SPIE*, 2656:34–45, 1996.

[4] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, June 1997.

[5] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and rendering of volume data based on simplicial complexes. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 19–26. ACM Press, October 17-18 1994.

[6] P. Cignoni, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data. *IEEE Trans. on Visualization and Comp. Graph.*, 3(4):352–369, 1997.

[7] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, June 1998.

[8] P. Cignoni, C. Rocchini, and R. Scopigno. Metro 3D: Measuring error on simplified tetrahedral complexes. Technical Report B4-35-00, I.E.I. – C.N.R., Pisa, Italy, May 2000.

[9] T.K. Dey, H. Edelsbrunner, S. Guha, and D.V. Nekhayev. Topology preserving edge contraction. Technical Report RGI-Tech-99, RainDrop Geomagic Inc. Champaign IL., 1999.

[10] M. S. Floater and A. Iske. Thinning algorithms for scattered data interpolation. *BIT Numerical Mathematics*, 38(4):705–720, December 1998.

[11] R.J. Fowler and J.J. Little. Automatic extraction of irregular network digital terrain models. *ACM Computer Graphics (Siggraph '79 Proc.)*, 13(3):199–207, Aug. 1979.

[12] M. Garland. Multiresolution modeling: Survey & future opportunities. In *EUROGRAPHICS'99, State of the Art Report (STAR)*. Eurographics Association, Aire-la-Ville (CH), 1999.

[13] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 209–216. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[14] M. Garland and P.S. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proceedings of the 9th Annual IEEE Conference on Visualization (VIS-98)*, pages 264–270, New York, October 18–23 1998. ACM Press.

[15] R. Grosso, C. Luerig, and T. Ertl. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. In *IEEE Visualization '97*, pages 387–394, Phoenix, AZ, Oct. 19-24 1997.

[16] B. Hamann and J.L. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design*, 11:477–489, 1994.

[17] H. Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 99–108. ACM SIGGRAPH, Addison Wesley, August 1996.

[18] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *ACM Computer Graphics Proc., Annual Conference Series, (Siggraph '97)*, pages 217–224, 1997.

[19] K.J. Renze and J.H. Oliver. Generalized unstructured decimation. *IEEE C.G.&A.*, 16(6):24–32, 1996.

[20] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum (Eurographics'96 Proc.)*, 15(3):67–76, 1996.

[21] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[22] O. G. Staadt and M.H. Gross. Progressive tetrahedralizations. In *IEEE Visualization '98 Conf.*, pages 397–402, 1998.

[23] I.J. Trotts, B. Hamann, K.I. Joy, and D.F. Wiley. Simplification of tetrahedral meshes. In *IEEE Visualization '98 Conf.*, pages 287–295, 1998.

[24] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing volume data. In *IEEE Visualization '97 Proceedings*, pages 135–142. IEEE Press, 1997. Roni Yagel and Hans Hagen.

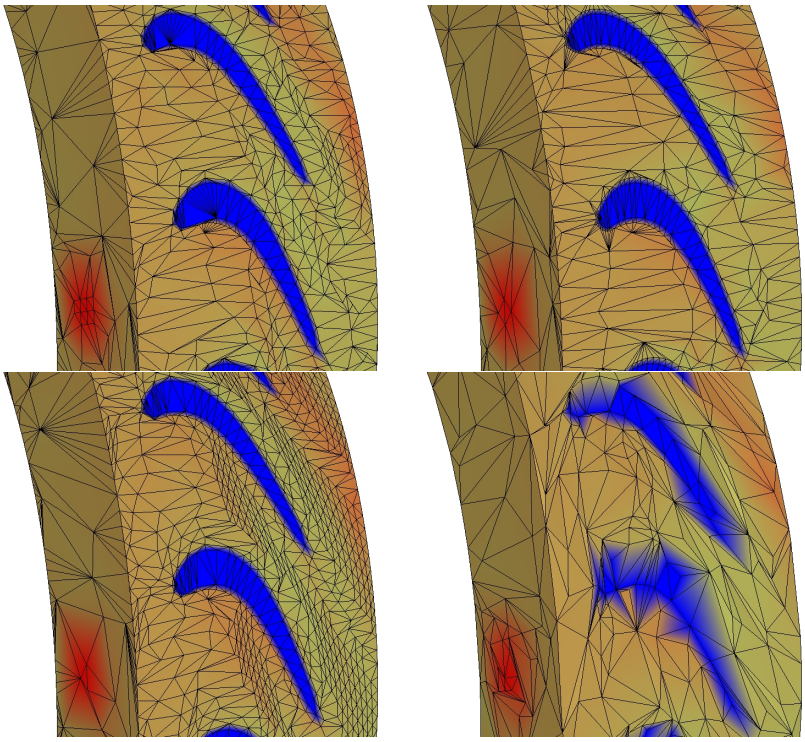

Figure 4: Different simplified meshes produced from the Turbine Blade dataset. The different meshes shown, of size 10,679 vertices, were produced with the **BF**, **BFS**, **LN** and **QD** techniques (from top-left, clockwise).

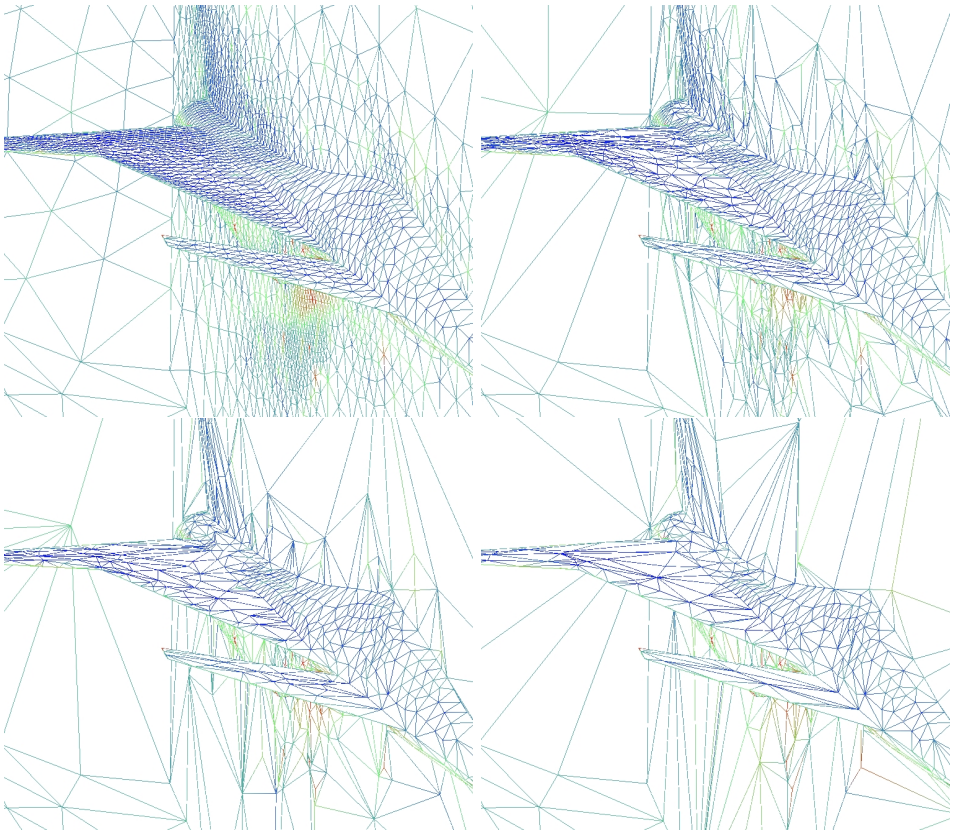

Figure 5: Different simplified meshes produced from the fighter dataset using the **BFS** technique; the mesh shown are composed, respectively, of 13,832, 6,916, 2,766 and 1,383 vertices; the corresponding errors are shown in Table 1. Note how well the boundary is preserved even on the coarsest simplified model.