published in *Proceedings of the 13th Canadian Conference on Computational Geometry (CCCG), August 2001*
Electronic proceedings: contact authors for reprints at *balmelli@us.ibm.com* or *geometry@balmelli.net*

Computational analysis of 4-8 meshes with application to surface simplification using global error

Laurent Balmelli
Visual and Geometric Computing
IBM Research, T.J Watson Center, USA.

Thomas Liebling
Mathematics Dept.
EPFL, Lausanne, Switzerland.

Martin Vetterli
Communication Systems Dept.
EPFL, Lausanne, Switzerland.

Abstract

We present computational results when computing approximations of a class of meshes with subdivision connectivity, known as 4-8 meshes. We consider algorithms using vertex decimation or vertex insertion. We explain that a full decomposition of a 4-8 mesh using global error can be obtained with an $\Theta(n \log n)$ decimation algorithm. Our algorithm produces progressive and adaptive representations of terrain data or subdivision surfaces in $\mathbb{R}^3$ having arbitrary topology.

1 Introduction

1.1 Motivation and previous work

This paper presents several computational results when processing a class of meshes with *subdivision connectivity*, known as 4-8 meshes. Meshes with subdivision connectivity are regular triangulations constructed using iterated subdivision rules. 4-8 meshes are known as quadtree triangulation in terrain visualization[1, 3, 4]. These meshes are also used to compute approximations of subdivision surfaces [5]. We look at algorithms to compute progressive and adaptive representations for these meshes, e.g Figure 1a.

We consider simplification algorithms for 4-8 meshes based either on vertex decimation (e.g. when starting from a dense mesh) or vertex insertion (e.g. when starting from a coarse version of the mesh). In Section 2, we explain that a hierarchy (very similar to the hierarchy in a tree structure) is imposed over the vertices by the mesh construction (Figures 2a-e). Hence, decimating/inserting an arbitrary vertex often implies jointly decimating/inserting additional vertices to preserve the hierarchy (think of pruning a tree node and all its descendants, or inserting a leaf node and all its parents). Joint decimation/insertion also ensures that the resulting mesh is conforming [2], or equivalently, a triangulation. The advantage of preserving the hierarchy is that a global error measure can be computed in order to simplify the mesh. For any vertex v, its set of descendants (i.e. jointly decimated with v) is called its *merging domain* and its set of parents (i.e. jointly inserted with v) is called its *splitting domain*.

In terrain visualization, previous algorithms are based on insertion only [4], or based on restricted cases of decimation [3] (see below). Also, all previous methods use local error metrics. For subdivision surfaces, most implementations are based on nonadaptive representations to avoid the added complexity and performances penalty traditionally associated with adaptive schemes. When simplifying a mesh, an error criterion, e.g. based on local variations in curvature, is used to select vertices to insert or decimate. Each simplification step modifies the model's shape and errors must be recomputed. In previous work, algorithms are given in order to recompute local errors after a vertex insertion [4] or restricted decimation [3]. However, no such algorithm is described in the general case of decimation [1].

This paper gives an extended summary of our results. More details are available in the long version, available in the electronic proceedings. Details are also available in [1]. We organize this paper as follows: We give our contributions in Section 1.2. In Section 2, we introduce 4-8 meshes and explain the constraints attached to their processing. We present simplification operations in Section 3 and give our algorithm using global error in Section 4.

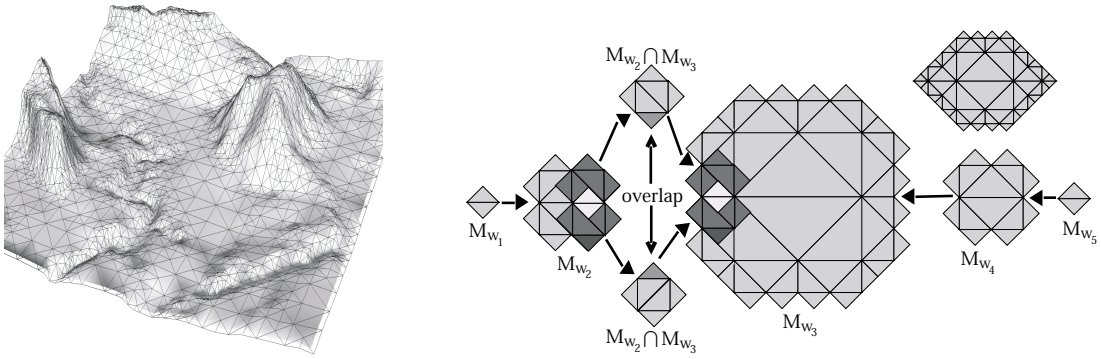
1.2 Contributions and plan

Our contributions are summarized below:

Computational analysis of mesh operations. We explain that $\Theta(\log n)$ operations are necessary, on average, to insert or decimate an arbitrary vertex (Section 3.1). Call *ancestors* the vertices whose error is modified after a decimation or an insertion. We explain that $\Theta(\log n)$ ancestors exist in each case (Section 3.2).

Merging domain intersections. We explain that an interesting problem is to determine *which* are the removed

[1] We explain the difference between restricted and general decimation in Section 2.



(a) Terrain visualization using a 4-8 mesh. (b) Decomposition of the intersection between a pair of merging domains.

Figure 1: Surface simplification using global error.

vertices in the merging domain of ancestors after decimating a vertex. This problem requires the computation of *merging domain intersections* (Section 3.3). It allows for building a global error metric for algorithms using decimation. We explain that $\Theta(\log^2 n)$ operations are sufficient to find merging domain intersections.

Decimation algorithm using global error. We propose an $\Theta(n \log n)$ decimation algorithm to produce progressive and adaptive representations of 4-8 meshes using global error (Section 4). In comparison, we show that a direct approach using the same error criterion requires at least $\Theta(n^2)$ operations.

2 4-8 meshes and constraints

We present a simple construction of a 4-8 mesh connecting an amplitude matrix (e.g. terrain data). Note that subdivision surfaces are used to create 4-8 meshes with arbitrary topology [5]. Our results are also valid for these meshes since we only assume a 4-8 connectivity for the vertices. Consider a matrix of amplitudes z, i.e. the coordinates x, y are implicit. A 4-8 mesh connecting the dataset is created using the recursive procedure depicted in Figures 2a-e. The iterative procedure used to connect the vertices imposes *hierarchical constraints* over the set of vertices, as explained below. The advantage of preserving the hierarchy is that a global error measure can be computed in order to simplify the mesh.

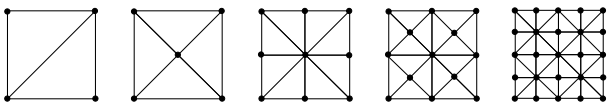


Figure 2: 4-8 connection of a 5×5 matrix of amplitudes: (a)-(e) Sucessive connection steps.

For example, consider the central vertex connected in Figure 2b. A decimation *perserving the hierarchy* operates as follows: When this vertex is decimated (e.g. in the mesh of Figure 2e), the edge orginally split by the vertex (diagonal in Figure 2a) is recovered. Consequently, all the vertices in the mesh are also decimated. Call this vertex v, then M_v denotes the set of vertices that must be removed jointly in order to recover the original edge and then preserve the hierarchy. We call this set *merging domain*. A merging domain is attached to each vertex in the mesh. For the vertices v connected at the step depicted in Figure 2e, $M_v = \{v\}$ since it suffices to remove v to recover the corresponding edge in Figure 2d. Such decimation is referred as a *restricted* case of decimation (as used in [3]). In contrast, the general case of decimation refers to when an arbitrary vertex (i.e. with $|M_v| > 1$) can be removed. In the long version of the paper, we explain the constraints imposed by the hierarchy in detail.

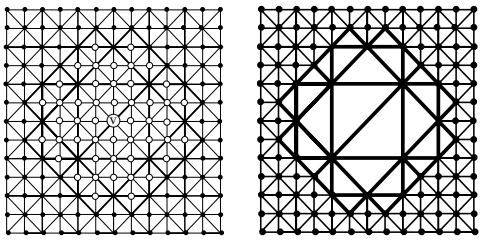


Figure 3: Mesh operations: (a) The white vertices represent the set M_v. (b) Support of the merging domain (set of remaining triangles when M_v is decimated).

Figure 3a depicts an example of merging domain: The white vertices belong to the domain attached to the central vertex v. Assume that l steps were used to construct the mesh, then v was connected at step $l - 4$ in this example. Therefore the smaller the connection step, the larger the merging domain. Figure 3b depicts the triangulation when M_v is decimated. We call *support* the set of triangles tiling the merging domain once M_v is decimated. Moreover, we

denote by $\breve{M}_v$ an empty merging domain.

We also attach a *splitting domain* to each vertex v, denoted by S_v, defined as the set of vertices to insert jointly in order to fulfill the hierarchical constraints [1]. Finally, an important consequence of attaching a merging domain or splitting domain to each vertex is that error criteria for vertices are evaluated *over the domains*.

3 Simplification operations

3.1 Vertex decimation and insertion

Consider a mesh of n triangles and assume that $|M_v|_\triangle$ denotes the number of triangles connected with at least one vertex in M_v (Figure 3a). Furthermore, $|\breve{M}_v|_\triangle$ counts the number of triangles tiling the support of the domain (Figure 3b). We give closed forms for $|M_v|_\triangle$ and $|\breve{M}_v|_\triangle$ in the long version of the paper and use them to derive asymptotical bounds for these values, i.e.

$$|M_v|_\triangle \in \Theta(n), \quad |\breve{M}_v|_\triangle \in \Theta(\sqrt{n}). \tag{1}$$

Recall that the merging domain size varies according to the vertex position in the hierarchy given by the connection steps (Section 2). We compute the expected values, over all of the vertices in the mesh, to obtain the average sizes and find that

$$E[|M_v|_\triangle] \in \Theta(\log n), \quad E[|\breve{M}_v|_\triangle] \in \Theta(c). \tag{2}$$

Denote by the $|S_v|_\triangle$ the number of triangles connected with at least one vertex in S_v. We show that the asymptotical value of $|S_v|_\triangle$ and the average size over the whole mesh are given by

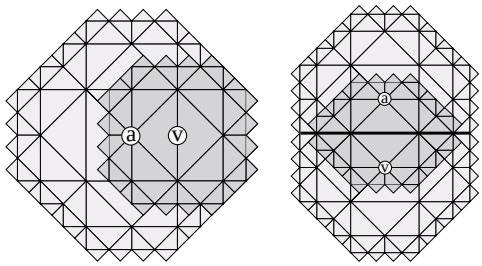
$$|S_v|_\triangle \in \Theta(\log n),\, E[|S_v|_\triangle] \in \Theta(\log n), \tag{3}$$

respectively.

3.2 Set of ancestor vertices

We denote by A_{M_v} and A_{S_v} the set of ancestors of v when decimating and inserting this vertex, respectively. The set A_{M_v} contains the vertices whose error is modified after *decimating* a vertex. Therefore, these vertices are still in the mesh. To find A_{M_v}, we are looking for vertices a such that $M_v \subset M_a$ (Figure 4a). We are also looking for vertices a whose domain M_a partially overlaps M_v (e.g. Figure 4b). In the latter figures, we depict the merging domains using their support. The set A_{S_v} contains the vertices whose error is modified after *inserting* a vertex. In contrast to A_{M_v}, the vertices in A_{S_v} are not yet in the mesh since an algorithm using A_{S_v} has an insertion approach to generate the adaptive surface. To find A_{S_v}, we are looking for the vertices a such that $\forall w, w \in S_v, w \in S_a$. In the long version of the paper, we show that we need to find a set of vertices in S_v with minimal connection step l, denoted by $\min_l(S_v)$, such that

$$A_{S_v} = \bigcup_{w \in \min_l(S_v)} M_w. \tag{4}$$



(a) (b)

Figure 4: Visual representation of the ancestors of M_v: (a) $M_v \subset M_a$. (b) $M_v \cap M_a$. The dark region depicts the domains' overlap and the thick line is the intersection boundary.

We give efficient algorithms for finding the ancestor sets in each case. Moreover we show that, on average,

$$|A_{M_v}| \in \Theta(\log n), \quad |A_{S_v}| \in \Theta(\log n), \tag{5}$$

ancestors exist for the sets M_v and S_v, respectively. In the long version, we use the above results to show that, using local error, $\Theta(n \log n)$ operations are necessary to fully decompose or refine a 4-8 mesh with n triangles.

3.3 Merging domain intersections

In this section we summarize our approach to finding merging domain intersections. The full description of our solution is available in the long version of the paper.

We describe the intersection between two merging domains (e.g. the dark region in Figure 4b) as *the union of a set of (smaller) merging domains*. Using merging domains as building blocks provides a compact and efficient description for intersections. Finding the vertices in M_v only requires searching around v using a single pattern, whereas finding $M_v \cap M_a$ is difficult due to the multiplicity of cases: Just consider all the possible locations for the ancestors a. Hence, Figure 4b is just a particular example of arrangement for $M_v \cap M_a$. In total, $\Theta(\log n)$ such arrangements exist.

The intersection between the domains in Figure 4b is decomposed in Figure 1b. Then, the intersection is

$$M_{w_1} \cup M_{w_2} \cup M_{w_3} \cup M_{w_4} \cup M_{w_5}, \tag{6}$$

which makes sense, since the vertices contained in the intersection belong to the domains $M_{w_i}, i = 1, 2, 3$. We would like to represent the intersection as an *exclusive* set of vertices, i.e. as in (6). However, in Figure 1b, M_{w_3} overlaps

with M_{w_2} and M_{w_4}. To express this constraint, we write the intersection as

$$\bigcup_{i=1}^{5} M_{w_i} = \bigoplus_{i=1}^{5} M_{w_i} \setminus D, \tag{7}$$

where the operator $\bigoplus$ "gathers" the vertices in the sets M_{w_i}, and D denotes the set of vertices *to remove* in order to obtain an exclusive set – in our example, the vertices in $M_{w_2} \cap M_{w_3}$ and $M_{w_3} \cap M_{w_4}$ –. Finally, the set D in (7) is also expressed as the union of smaller domains. For example, in Figure 1b, $M_{w_2} \cap M_{w_3}$ is further decomposed into a pair of smaller domains which again overlap. Therefore, computing D also involves removing redundant vertices. This hints that finding an intersection requires recursively adding ($\oplus$) and subtracting ($\setminus$) domains (inclusion exclusion principle).

In the long version of the paper, we propose a model for this problem and show how to obtain a nonrecursive formulation. We obtain a closed form for the cost of computing an intersection and derive computational bounds. We find that the average cost to solve a single intersection is $\Theta(\log n)$. Finally, we give an $\Theta(\log^2 n)$ algorithm to compute all intersections between M_v and the domains $M_a, \forall a \in A_{M_v}$.

4 Application

We introduce now our decimation algorithm based on global error and show that it outperforms a direct approach using the same error criterion. We use *mesh functionals* $u : M_v \to \mathbb{R}$ to compute properties for v *over its domain* M_v. We use two mesh functionals R and D: R is called the *rate* and counts the number of triangles, whereas D measures the distance in l_2 norm between the original surface and an approximation. Hence, for each v we compute the vector value $\mathbf{u}(M_v) = (R(M_v), D(M_v))$.

Call M_0 the input mesh and M a simplified version (e.g Figure 1a), then the problem to solve is

$$D(R) = \min_{|M| \leq |M_0|} \{D(M) | R(M) \leq R\}, \tag{8}$$

where R denotes a constraint in rate. We further define the *variation* of a functional as $\Delta\mathbf{u}(M_v) = \mathbf{u}(M_v) - \mathbf{u}(\breve{M}_v)$. Hence, again $\Delta\mathbf{u}(M_v) = (\Delta R(M_v), \Delta D(M_v))$. The variation $\Delta\mathbf{u}(M_v)$ is the change in rate and distortion when M_v is decimated. Therefore, a vector $\Delta\mathbf{u}(M_v)$ links two simplified meshes in the space of values spanned by R and D (this plane is usually called *rate-distortion* plane). Therefore the ratio $\lambda(v) = -\Delta D(M_v)/\Delta R(M_v)$ is the trade-off between rate and distortion when M_v is decimated and represents a slope in the rate-distortion plane.

The algorithm proceeds as follows: Initially, the variations $\Delta\mathbf{u}(M_v)$ for each vertex are computed. Using $E[|M_v|_\triangle]$ in (2), this step has cost $\Theta(n \log n)$. Then, at each iteration the vertex v with minimal $\lambda(v)$ is chosen. Then, M_v is decimated and the ancestor errors are updated as follows: For each vertex $w \in M_v$, a set of ancestors a such that $M_w \subset M_a$ is built. For each a, the vector-functional values are replaced by

$$\Delta\mathbf{u}(M_a) - \Delta\mathbf{u}(M_w), \tag{9}$$

and $\lambda(a)$ is recomputed. In the long version of this paper, we explain that this update algorithm actually computes merging domain intersections. Also, we show that the error computed at each vertex is global.

The complexity of the algorithm is evaluated as follows: The cost for updating all errors is $\Theta(\log^2 n)$. The algorithm requires $n/\Theta(\log n)$ steps, on average, to fully decompose the mesh. Hence, the total cost for the simplification algorithm is then $\Theta(n \log n)$. A direct algorithm needs to recompute the global error over each ancestors' domain. A lowerbound for this update is obtained as follows: We have roughly $\Theta(4^{l+1})$ vertices at connection step l and $l \in \Theta(\log n)$ ancestors exists. Call a any such ancestor, then $|M_a|_\triangle(i, n) \approx n/4^{i-1}$, $1 \leq i \leq l$. Therefore, a lowerbound for the complexity of the direct algorithm is

$$\sum_{i=0}^{\log_4 n} 4^i \sum_{j=0}^{i} \frac{n}{4^j} = \frac{16}{9}n^2 - \frac{1}{3}n \log_4 n - \frac{7}{9}n \in \Theta(n^2). \tag{10}$$

Note the above approximation accounts only for the ancestors a such as $M_v \subset M_a$. Accounting for the update of the ancestors whose domain partially overlaps does not change the order of magnitude. However, this evaluation is complex due to the $\Theta(\log n)$ cases of overlap one has to deal with. We conclude with the following proposition:

Proposition 1
An algorithm based on global error and using general decimation requires $\Theta(n \log n)$ operations to fully decompose a 4-8 mesh with n triangles when merging domain intersections are used to update the vertex errors.

References

[1] L. Balmelli. Rate-distortion optimal mesh simplification for communications. *Ph.D. Thesis 2260 (2000), Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland.*, 2000.

[2] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, algorithms and applications*. Springer-Verlag, 2000.

[3] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich, and M. B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. *Proceedings of IEEE Visualization*, 1997.

[4] P. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, and G.A. Turner. Real-time continuous level of detail rendering of height fields. *Proceedings of SIGGRAPH*, pages 109–118, 1996.

[5] L. Velho and D. Zorin. 4-8 subdivision. *CAGD*, 2000.