Computer-Aided Design 42 (2010) 129–138

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Parallel extraction and simplification of large isosurfaces using an extended tandem algorithm

Guilhem Dupuy [a,*], Bruno Jobard [a,c], Sebastien Guillon [a,b,c], Noomane Keskes [a,b,c], Dimitri Komatitsch [b,c]

[a] LIUPPA, Computer Science Laboratory of the University of Pau, France
[b] MIGP, Geophysics Laboratory of the University of Pau, France
[c] INRIA-Magique3D, France

ARTICLE INFO

Article history:
Received 23 June 2008
Accepted 14 April 2009

Keywords:
Geometric computation
Distributed design and geo-scientific applications

ABSTRACT

In order to deal with the common trend in size increase of volumetric datasets, in the past few years research in isosurface extraction has focused on related aspects such as surface simplification and load-balanced parallel algorithms.

We present a parallel, block-wise extension of the tandem algorithm [Attali D, Cohen-Steiner D, Edelsbrunner H. Extraction and simplification of iso-surfaces in tandem. In: SGP '05: Proceedings of the third Eurographics symposium on Geometry processing. Aire-la-Ville, Switzerland: Eurographics Association; 2005. p. 139–148], which simplifies on the fly an isosurface being extracted. Our approach minimizes the overall memory consumption using an adequate block splitting and merging strategy along with the introduction of a component dumping mechanism that drastically reduces the amount of memory needed for particular datasets such as those encountered in geophysics. As soon as detected, surface components are migrated to the disk along with a meta-data index (oriented bounding box, volume, etc.) that permits further improved exploration scenarios (small component removal or particularly oriented component selection for instance).

For ease of implementation, we carefully describe a master and worker algorithm architecture that clearly separates the four required basic tasks. We show several results of our parallel algorithm applied on a geophysical dataset of size $7000 \times 1600 \times 2000$.

1. Introduction

Surface reconstruction for shape modeling is widely used in a large variety of fields (medicine, geophysics, etc.). The marching cubes algorithm, introduced by Lorensen et al. [1], is the most classical algorithm used for isosurface extraction. Due to the increasing size of processed datasets and extracted surfaces, many improvements of the marching cubes have been proposed. They concern, for instance, ambiguity treatment [2], reduction of the number of traversed cells [3,4], load balancing in parallel approaches [5,6] and reduction of the number of generated triangles [7].

In order to cope with the increasing size of datasets, isosurfaces might need to be simplified to reduce the number of generated triangles both for memory storage or more importantly for visualization purposes. In a brute-force approach one would extract the full mesh and simplify it in a second pass [8–10]. The problem with this method lies in the generation of a first very large mesh that may not fit in the main memory before further simplification. In [11], Attali et al. reduce memory requirements by introducing a tandem algorithm that combines isosurface extraction and simplification stages in one pass. Their method drastically reduces the amount of vertices and triangles stored in memory during extraction, allowing larger datasets to be processed.

Similarly to how Attali et al. addressed the memory problem raised by larger datasets, we introduce a parallel, block-wise extended version of the tandem algorithm to accelerate the computation of simplified isosurfaces. The dataset is split into blocks and sent to compute nodes. In each node a local isosurface is extracted and semi-simplified based on a slightly modified tandem algorithm. Then nodes can receive an adjacent semi-simplified isosurface that will be merged with the local one. This merge operation ends with a simplification stage with relaxed edge constraints at their common interface that remove seams between them. The algorithm finishes when all local semi-isosurfaces have been merged and simplified. Our splitting/merging strategy forces

* Corresponding author. Tel.: +33 0 5 59 83 59 28; fax: +33 0 5 59 83 48 58.
E-mail addresses: guilhem.dupuy@univ-pau.fr (G. Dupuy), bruno.jobard@univ-pau.fr (B. Jobard), sebastien.guillon@total.com (S. Guillon), noomane.keskes@total.com (N. Keskes), dimitri.komatitsch@univ-pau.fr (D. Komatitsch).

0010-4485/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2009.04.016

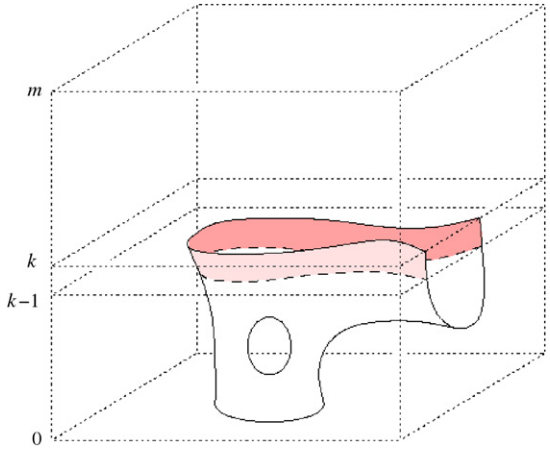

Fig. 1. The kth layer (in red) is the set of vertices, edges and triangles extracted with the marching cubes algorithm between cross-sections k − 1 and k. Figure courtesy of Dominique Attali.

adjacent nodes to be processed together, maintaining memory consumption as low as possible during the overall isosurface extraction.

We also introduce an early component dumping mechanism that frees the memory as soon as independent surface components have been extracted, simplified and stored to disks along with meta-data for later high-level exploration. This strategy has proven to be very useful for particular datasets such as those in geophysics for which the extracted features are numerous but small compared to the global size of the volume. The meta-data stored along these disconnected components can be used for filtering purposes during their visualization, for instance discarding those with too small volumes or keeping particularly aligned ones. This dumping mechanism can also be beneficial for noisy datasets or when a pertinent isovalue is not yet well determined and leads to many small disconnected components.

In the first part of this article we describe the tandem algorithm. In the second part we propose a parallel extension of this algorithm with dumping of completed objects. The third part presents some computational experiments.

2. The "Tandem Algorithm"

The main idea of the algorithm of Attali et al. [11] is to alternate the extraction of a layer (see Fig. 1) and the simplification of the current overall extracted surface in order to reduce the amount of occupied memory. Indeed, a global simplification after a complete extraction would require to store all vertices and triangles, while a simplification stage during extraction reduces the number of vertices and triangles at each step.

The tandem algorithm is then a simple loop that iterates over the cross-sections to implement two operations:

- an "extraction" operation that adds a layer (vertices, edges and triangles obtained with the marching cubes algorithm between two subsequent cross-sections) to the current triangulation.
- a "simplify" operation that simplifies the current triangulation. The last layer added is not simplified so that the next layer can be appended.

The simplification stage consists in applying the classical edge collapse algorithm [12]. Each edge collapse operation has a cost that measures the numerical error it would introduce in the triangulation. Edge candidates for edge collapse are kept in a priority queue $\mathcal{Q}$ ordered by their cost. To evaluate this cost, Attali et al. [11] revisited the quadratic metric of Garland and Heckbert [12] and proposed a quadratic error which is a weighted sum of a shape measure criterion and a mesh isotropy criterion.

The shape measure is similar to that used in the original edge contraction algorithm. It measures the deviation introduced by the collapse operation between the new vertices and the original surface. The shape measure of a point x is defined by

$$h_c(x) = \frac{1}{W_c}\sum_{t\in U_c} w_t d^2(x, P_t) = \frac{1}{W_c} x^{\mathrm{T}}\mathbf{H}_c x,$$

where U_c is the patch defined by all the neighboring triangles of point c. P_t is the plane spanned by the triangle t, w_t is its area and $W_c = \sum_{t\in U_c} w_t$. $\mathbf{H}_c$ is a positive definite matrix. Edge contraction $ab \mapsto c$ leads to $\mathbf{H}_c = \mathbf{H}_a + \mathbf{H}_b$ and $W_c = W_a + W_b$.

The mesh isotropy criterion is introduced in order to prevent the creation of long and skinny triangles. Considering S_{ab}, the set of triangles containing the vertices a, b or both in the current triangulation, the mesh isotropy criterion for point c is defined as the squared distance of the point to the patch:

$$g_c(x) = \sum_{t\in S_{ab}} w_t(\|x - \hat{t}\|^2 + avg(t)) = x^{\mathrm{T}}\mathbf{G}_c x$$

where $\hat{t}$ is the barycenter of the triangle t and $avg(t) = \frac{1}{12}(\|p\|^2 + \|q\|^2 + \|r\|^2)$ with p, q, r the vectors from $\hat{t}$ to the vertices of t. The term g_c is normalized by $W = 3 \times area(S_{ab}) W_c^{1/2}/E_0$ in order to balance its influence with h_c in the global cost defined by :

$$\varepsilon_\alpha(c) = \sqrt{c^{\mathrm{T}}\left[(1-\alpha)\frac{\mathbf{H}_c}{W_c} + \alpha\frac{\mathbf{G}_c}{W}\right]c}$$

α is called the isotropy parameter, and represents a compromise between the shape measure criterion and the anisotropy measure criterion. In practice α is set to 0.4 for a good compromise between the two criteria.

For more details about those mathematical formulations, please refer to the original article [11].

During edge contraction, the resulting vertex position c is obtained by minimizing the local error function ε_α and the final error value $\varepsilon_\alpha(c)$ is used to order the priority queue $\mathcal{Q}$. In any case, a candidate cannot be accepted if its shape measure, $\varepsilon_0(c)$, exceeds a positive constant error threshold E_0. The *simplify* function then consists in emptying the queue $\mathcal{Q}$ by applying consecutive edge collapses.

Right after the extraction stage, the simplification stage has to cope with the heavy edge constraints on the last extracted layer. To prevent the creation of artifacts that these blocked edges would introduce, an innovative way of scheduling the edge collapse was proposed, called time lag. The main idea was to delay edge collapses near the advancing front. The time lag is based on the rank of a vertex, equal to its coordinate along the extraction direction (for example z if the cross-sections are taken along the z-axis). The rank of the front is the maximum rank that has been extracted. Considering, $height(u) = rank(u)$ and $rad(u) = 1$ for new vertices introduced by extraction, the contraction of an edge $ab \mapsto c$ leads to :

$$height(c) = (height(a) + height(b))/2$$
$$rad(c) = (\|a - b\| + rad(a) + rad(b))/2$$
$$reach(c) = height(c) + rad(c).$$

The contraction of an edge is prevented as long as its reach value is greater than or equal to the rank of the advancing front. As detailed in [11], if a and b belong to the last extracted layer, $height(c) = rank(front)$ and since $rad(c) > 0$, the contraction of ab would be prevented. Similarly, if a vertex of ab lies in the front plane, *reach(c)* would be greater than the rank of the advancing front.

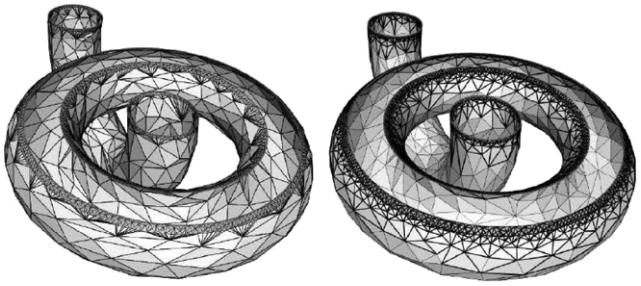

Fig. 2. Two partial triangulations constructed without (left) and with (right) the time lag: the length of the edges gets progressively longer as they proceed further to the front layer. Figure courtesy of Dominique Attali.

The blocked edges are kept in a priority queue $\mathcal{W}$ ordered by reach value. The function *delay* adds all edges of the last layer k in $\mathcal{W}$. An edge is moved from $\mathcal{W}$ to $\mathcal{Q}$ if its reach value is lesser than the rank value of the advancing front. The function *activate*, described in algorithm 1, moves edges from $\mathcal{W}$ to $\mathcal{Q}$. Fig. 2 illustrates the effect of the time lag near the advancing front.

```
Procedure activate( k : integer)
    While (reach(top(W)) < k) do
        add top(W) in Q;
        pop(W);
    done
End
```

Algorithm 1: At the kth layer, the "*activate*" function fills up the edge collapse candidates queue $\mathcal{Q}$ with previously delayed edges of $\mathcal{W}$.

The tandem algorithm can then be written as in algorithm 2. It takes E_0, the maximum shape error allowed, as a parameter.

```
Procedure tandem( E0 : float)
    For k from 1 to number of cross-sections - 1 do
        extract(k);
        delay(k);
        activate(k);
        simplify(E0);
    end For
    activate(∞);
    simplify(E0);
End
```

Algorithm 2: The tandem algorithm

3. Extended tandem algorithm

The tandem algorithm was designed to work on large datasets but experimental results (a test that we performed on a noisy dataset of size $1626 \times 7028 \times 2000$) showed us the limited scalability of this algorithm. On this cube, each extraction on the advancing fronts generates 1 250 000 triangles (Fig. 3). In this case, updates of the queues and simplification steps are too memory consuming. We therefore propose an extension of the tandem algorithm to increase its scalability. This extension consists in dumping parts of the extracted surface and dispatching the extraction process on subsets of the dataset.

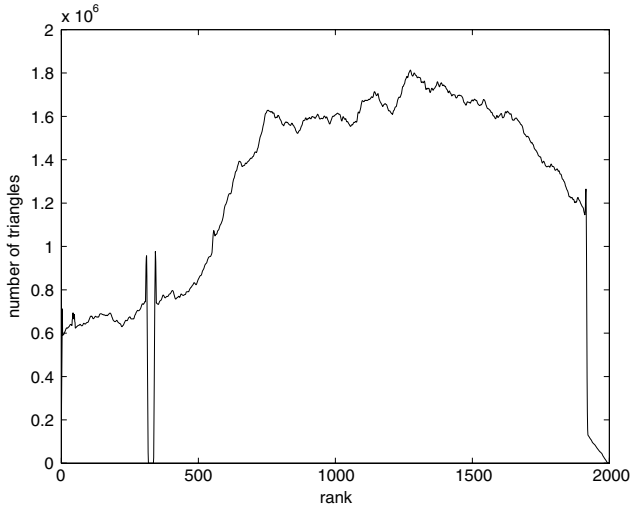

Fig. 3. Number of triangles generated for each layer of the $(1626 \times 7028 \times 2000)$ dataset. The region with no triangles extracted is due to a hole in the dataset.

Fig. 4. Distribution of extracted components. The finished components are shown in light gray and growing ones in dark gray. The black arrow indicates the direction used for extraction.

3.1. Component dumping

Classical out-of-core approaches migrate mesh to disk during extraction. The generated files are then soups of triangles ordered by direction extraction. But isosurfaces extraction, especially in geophysics, generate many disconnected components. Fig. 4 illustrates a usual repartition of geological sets: the finished components are shown in light gray and components that are still growing or that are not completely simplified are shown in dark gray. This distribution of disconnected components allows us to dump completed surfaces as soon as their simplification is finished.

The dumping process requires to detect the termination of a component extraction, but since we use the time lag technique, a component may be entirely extracted but its simplification may have been prevented. We therefore need to detect the end of its extraction as well as the end of its simplification.

To formalize the finalization of a component, we define an active component γ as the set of vertices defining its shape. We consider Γ the set of all the active components, $\Gamma = \{\gamma_i\}$ and define V_w the set of all the vertices defining edges in the queue $\mathcal{W}$ (all the edges prevented by the time lag technique). A component is then finalized if it has no more edges to collapse (γ has no more edges in $\mathcal{W}$), which can be written as :

γ is finalized $\Leftrightarrow \gamma \cap V_w = \emptyset$.

We define the function *save*(γ) that migrates a component to the disk and the function *clear*(γ) that deletes γ in the current

triangulation. The *Dumping* function can then be written as in algorithm 3.

```
Procedure dumping()
    For Each γ in Γ do
        If (γ ∩ V_w = ∅) then
            save(γ);
            clear(γ);
        end If
    end For
End
```

Algorithm 3: The "dumping" function

The *dumping* function is then introduced in the tandem algorithm (algorithm 4).

```
Procedure tandem( E_0 : float)
    For k from 1 to number of cross-sections - 1 do
        extract(k);
        delay(k);
        activate(k);
        simplify(E_0);
        dumping();
    end For
    activate(∞);
    simplify(E_0);
    dumping();
End
```

Algorithm 4: The new tandem algorithm with dumping

An intrinsic property of our component-based dumping is that our generated file is ordered by component. We can then easily access a subset of components by reading a subset of the generated file. To optimize access to components in this file (called the raw data file) we generate an index file (called the index component file). To enhance the exploration of the extracted surface, meta-data (such as oriented bounding box, volume, number of vertices and facets, etc.) are computed for each component and stored in the index file (Fig. 5). One exploration scenario could be based on volume filtering such as in [13] in which Pivot et al. propose a workflow for complex volume seismic interpretation. They suggest to extract isosurfaces on seismic attributes and to delete automatically inconsistent small "bubbles" (disconnected components with very small volumes with respect to other components). Another useful way of separating components is an analysis based on the geological depositional direction (azimuthal direction).

3.2. Parallel algorithm

A good feature of the marching cube is its spatial independence. Each grid cell can be processed independently of the others and in any traversal order. We therefore propose to split datasets into subsets, to extract simplified surfaces in these subsets using the extended tandem algorithm and to merge them in a final surface.

Our parallel algorithm considers the dataset as a layout of subsets on which a slightly modified tandem algorithm is first applied. In fact, during the continuous simplification of the extracted triangulations, edge collapse operations are not applied

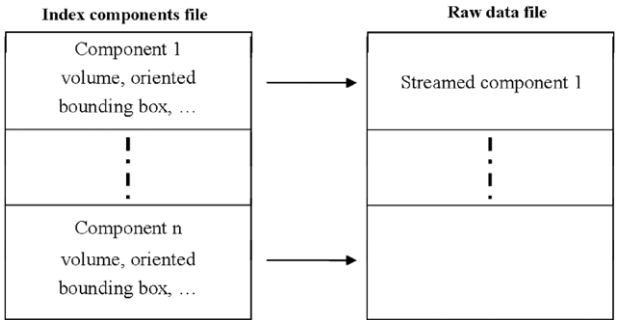

Fig. 5. File organization. The index file (left) refers to the raw data file (right).

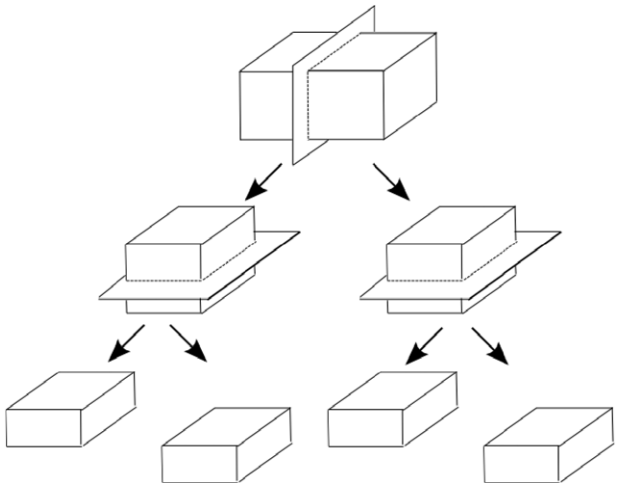

Fig. 6. Binary partitioning. The dataset is recursively split perpendicularly to its longest direction until it goes down a given size threshold.

near boundaries in order to enable a later merge with adjacent blocks of isosurfaces. Blocks of semi-simplified isosurfaces are then aggregated as they become available and aggregates get simplified again to remove seams in the merged triangulation.

Dealing with semi-simplified isosurface blocks must imply a high priority in merging them as soon as possible to keep the overall number of triangle as low as possible. The splitting strategy and the subset traversal order must be designed with this requirement in mind.

A classical dataset subdivision as a regular grid of blocks seems natural for this purpose, but it becomes tricky to choose along time the next good block to process and with which one it is better to merge. To address this aspect we have chosen to complement this grid with an implicit hierarchical organization scheme that allows a fast selection of adjacent blocks without traversing the whole grid of blocks. Process and merge orders are directed thanks to this hierarchical layout.

Splitting phase. We propose a hierarchical scheme for surface extraction. As in [14], the dataset is split recursively into two parts perpendicularly to its longest direction as long as the sub-block size is greater than a given threshold (Fig. 6). These successive subdivisions of the dataset implies a binary tree logical organization (Fig. 7). Isosurface extraction is performed on leaves of the generated tree and recursively merged to reconstruct the global surface.

Extraction in leaf nodes is performed using the tandem algorithm combined with our new dumping approach. The size of the leaves could then be relatively large since the tandem algorithm has been designed for this purpose. However our parallel algorithm is quite independent of this size. In our experiments,

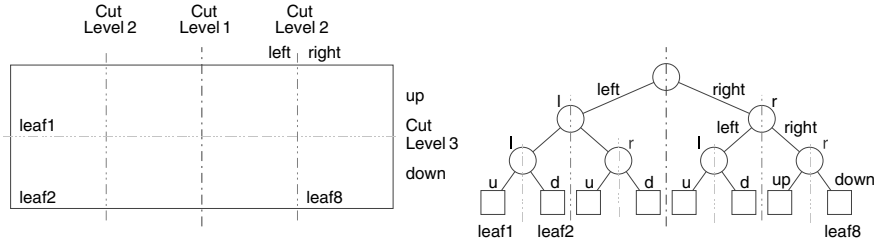

Fig. 7. Binary partitioning of a dataset (left) and the resulting logical organization as a binary tree (right). Each node of the tree is tagged by a type indicating its position regarding the cutting plane.

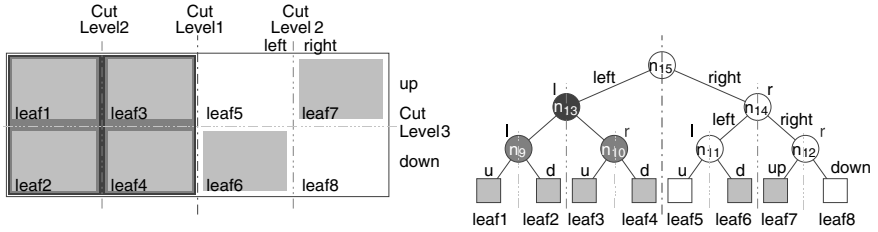

Fig. 8. Restricted adjacency requirement. Merging operations occur between brother nodes or between a node and its nephews that share a common boundary. Light gray leaves have already been processed. Darker gray nodes are considered as merged and completely processed.

overall extraction timings do not significantly vary with 128^3, 256^3 or 512^3 leaf block sizes.

In order to avoid refinement dependencies between adjacent blocks and prevent artifacts due to a bias in shape during simplification, we extend the time lag notion. As in the original time lag the radius associated with a point c (result of the collapse $ab \mapsto c$) is given by

$$rad(c) = (\|a - b\| + rad(a) + rad(b))/2$$

with $rad(x) = 1$ if x is an original vertex of the isosurface. The contraction of edge ab is prevented as long as the sphere centered on the middle of ab and of radius $rad(c)$ is not totally included in the block. We show in Fig. 9 how the introduction of the time lag influences the refinement progression near the boundaries of the blocks.

Merging phase. In our splitting strategy (binary partitioning), each node p is split into two children $p1$ and $p2$, denoted c_p (children of p). Reciprocally p is the father of $p1$ and $p2$ denoted by f_{p1} and f_{p2} respectively; $p1$ is the brother of $p2$ by $b_{p2} = p1$ and reciprocally $b_{p1} = p2$.

Intuitively, a good strategy for block merging should be to recursively merge brothers, from the leaves up to reaching the root of the tree. Considering brothers ensures that they share a large common boundary and that the simplification of the merged triangulation will remove a significant number of triangles. In practice isosurface components are not uniformly distributed in the sub-blocks and therefore extraction time of two brothers could be drastically different, and merging them implies waiting for the slower one. To solve this problem we allow the algorithm to only merge a block with its brother or with one of its nephews if they have a common boundary.

We thus need to define a restricted adjacency criterion determining if a sub-block of a brother node has a common boundary. We consider that a block p can be split along three different planes (according to its longest direction). We define the type of a node as the position of the node after the split of its father. If a node p is split along x, $p1$ and $p2$ are denoted respectively left and right. Similarly, cut along y defines children as up and down, and cut along z as front and back.

Denoting by $\overline{\cdot}$ the opposite operator, we have:

$$\overline{\text{left}} = \text{right} \quad \text{and} \quad \overline{\text{right}} = \text{left}$$
$$\overline{\text{up}} = \text{down} \quad \text{and} \quad \overline{\text{down}} = \text{up}$$
$$\overline{\text{front}} = \text{back} \quad \text{and} \quad \overline{\text{back}} = \text{front}.$$

We define *tree* of root p (T_p) as the set of nodes containing p and its recursive children. In this tree, *ancestry* of a node n ($A_{p,n}$) is defined as the set of nodes containing n and its recursive fathers up to p:

$$A_{p,n} = \begin{cases} n & \text{if } n = p \\ n \cup A_{p,f_n} & \text{else.} \end{cases}$$

Considering a block split into $p1$ and $p2$, a nephew node n belonging to $T_{c_{p2}}$ has a common boundary with $p1$ if

$$\forall\, n' \in A_{c_{p2},n}, \quad \text{type}(n') \neq \overline{\text{type}(p1)}.$$

If n then fulfills this restricted adjacency requirement it can be merged with $p1$. This criterion can efficiently be implemented via an iterative tree traversal through the successive fathers while verifying the condition on their types. When two brothers have been merged, their father node is considered completely processed.

To understand better these merging requirements, let us consider Fig. 8. Leaves 1 and 2 are processed and are brother nodes, they can then be merged; n_9 is then considered as completely processed. Same case for n_{10} with the leaves 3 and 4. It results that n_9 and n_{10} can also be merged and that n_{13} is then completely processed. n_{13} cannot be directly merged with n_{14} because n_{14} is not completely processed. However leaf6 can be merged with n_{13} because it fulfills the requirements on the types of its ancestry nodes ($A_{c_{n_{14}},\text{leaf6}} = \{\text{leaf6}; n_{11}\}$): $\text{type}(\text{leaf6}) = \text{down} \neq \overline{\text{type}(n_{13})} = \text{right}$ and $\text{type}(n_{11}) = \text{down} \neq \overline{\text{type}(n_{13})} = \text{right}$. Finally, leaf7 cannot be merged with n_{13} because one of its ancestry nodes ($A_{c_{n_{14}},\text{leaf7}} = \{\text{leaf7}; n_{12}\}$) does not fulfill the requirement: $\text{type}(n_{12}) = \text{right}$ is not different from $\overline{\text{type}(n_{13})} = \text{right}$. This last case can be seen in Fig. 8, leaf7 does not have any common boundary with the dark node n_{13}.

Geometrically, merging two sub-blocks requires first to identify common points extracted in each block lying on their shared boundary. Due to potential numerical inaccuracy in vertex extraction, we cannot identify common points based on a simple coordinate comparison. Identification is then done by using a property of the marching cubes algorithm: as a cell edge can have only one or zero points generated by the algorithm, two points belonging to common cell edges are identical and then merged.

Once this merging step is finished, one has to simplify the pasted area with the same error criterion as for the surface

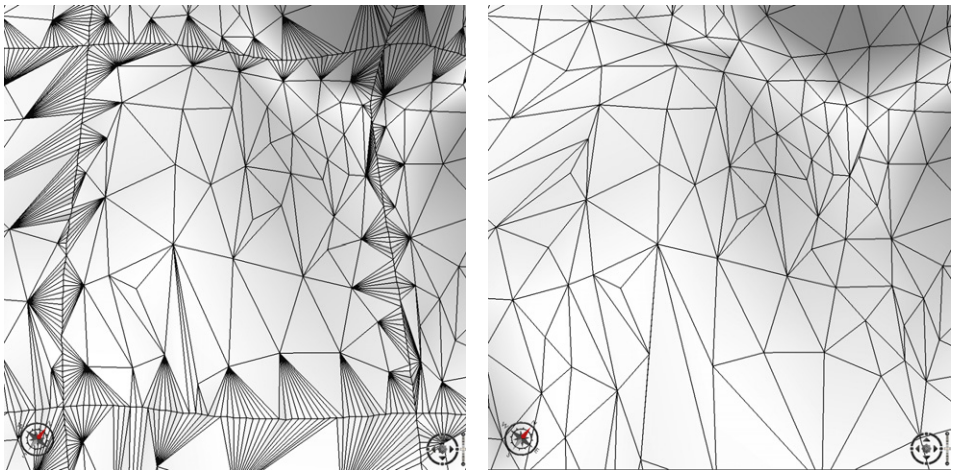

Fig. 9. On the left we illustrate the effect of the time lag on the refinement at the boundaries of the blocks before simplification. On the right, surfaces have been simplified. On the top row, only edges that have at least a vertex on a boundary have been kept. On the bottom row, edges have been excluded from simplification using the time lag method. This shows that applying the time lag method near the boundaries of blocks leads to triangulations of better quality.

extraction E_0. As during the extraction step, the contraction of an edge *ab* is prevented as long as the sphere centered on the middle of *ab* and of radius $rad(c)$ is not totally included in merged blocks. We show in Fig. 9 the merge of surfaces and the effect of the time lag on the quality of the resulting surfaces. Finally the last step of the merge consists in dumping finished components to the disk.

3.3. Manager/workers implementation

In our application the target parallel machine is a *load-balanced cluster* managed by the Platform LSF-HPC software. A typical use of this parallel machine is to decompose a process into independent tasks and then let the task scheduler (here LSF-HPC) dispatch tasks to the distant grid. When a task is finished, the processor used for this task is freed. The task scheduler can reallocate it for the process if resources are sufficient for other running processes.

We propose a parallel algorithm adapted to this kind of architecture that is composed of workers and a worker manager. Workers are distant processes that have to extract, merge and dump surfaces or send them to other workers. The worker manager is a process, distant or not, that assigns tasks to workers. This approach and our formalism are adapted to our target parallel machine (number of allocated processors varies in time) but could also be used on other parallel machines. It is also worth mentioning that our algorithm can also be run with no modification with only one worker. In this case, this worker will process and aggregate successively all the blocks of the isosurface in the grid.

Workers do not have any consciousness of the tasks they have already accomplished, the portions of surfaces they currently own or if any other worker is processing adjacent blocks, the worker manager does. Workers simply ask and obey the worker manager that uses the hierarchical structure to quickly find what is the next best block to extract for a given worker or to what worker another one should send its aggregate of isosurfaces before its termination.

A worker is written as an infinite loop that requests tasks from the worker manager (algorithm 5). This loop ends when the worker manager decides so. Tasks are composed of a task type and a worker id (each worker has an unique id used to send messages). We distinguish four task types:

- *EXTRACT*: the worker has to extract a surface from a sub-block of the dataset. The extraction is performed with the extended tandem algorithm. Sub-block boundaries, except those that are common to dataset boundaries, would prevent the edge collapse in their neighborhood using time lag. The extracted surface is appended to its current surface.
- *SEND*: the worker received a worker id, referring to a target worker. The worker has to send its current mesh to the target worker that will merge it with its current one. After this send, the current surface of the worker is emptied.
- *MERGE*: the worker has to merge its current surface with a mesh sent by another worker. After the merge operation, the new