Isosurface extraction and interpretation on very large datasets in geophysics

Guilhem Dupuy*
LIUPPA

Bruno Jobard †
LIUPPA, INRIA-Magique3D

Sebastion Guillon ‡
Noomame Keskes §
Total

Dimitri Komatitsch ¶
MIGP, INRIA-Magique3D

Abstract

In order to deal with the heavy trend in size increase of volumetric datasets, research in isosurface extraction has focused in the past few years on related aspects such as surface simplification and load balanced parallel algorithms.
We present in this paper a parallel, bloc-wise extension of the tandem algorithm [Attali et al. 2005], which simplifies on the fly an isosurface being extracted. Our approach minimizes the overall memory consumption using an adequate bloc splitting and merging strategy and 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 will allow further improved exploration scenarios (small components removal or particularly oriented components selection for instance).
For ease of implementation, we carefully describe a master and slave algorithm architecture that clearly separates the four required basic tasks. We show several results of our parallel algorithm applied on a $7000\times1600\times2000$ geophysics dataset.

CR Categories: I.3.5 [Computer graphics]: Computational Geometry and Object Modeling F.2.2 [Analysis of algorithms and pronlem complexity]: Nonnumerical Algorithm and problems—Geometrical problems and computations I.4.10 [Image processing and computer vision]: Image Representation—Volumetric J.2 [Computer Applications]: Physical sciences and engineering D.1.3 [Programming techniques]: Concurrent Programming—Distributed programming

Keywords: Geometric computation and processing, Collaborative and distributed design, Geo-scientific applications

1 Introduction

Surface reconstruction for shape modeling is widely used in a large variety of domains (medicine, geophysics, ...). The marching cubes algorithm, introduced by Lorensen and al. [Lorensen and Cline 1987], is the most classical algorithm used for isosurface extraction.
Due to the increasing size of processed datasets and extracted surfaces, a lot of improvements of the marching cubes have been proposed. Those improvements concern for instance ambiguities treatment [Gelder and Wilhelms 1994], reduction of the number of traversed cells [Wilhelms and Gelder 1992, Livnat et al. 1996], load balancing in parallel approaches [Niguet and Nicod 1995, Mackerras 1992] and reduction of the number of generated triangles [Shekhar et al. 1996].

In order to cope with the increasing size of datasets, isosurfaces might have to be simplified for reducing the number of generated triangles both for memory storage or more likely for visualization purposes. In a brute-force approach one would extract the full mesh and simplify it in a second pass. The problem with this method lies in the generation of a first heavy mesh which may not fit into the main memory before further simplification. In [Attali et al. 2005], Attali et al contribute in lowering memory requirements by introducing a tandem algorithm which combines isosurface extraction and simplification stages in one pass. Their solution reduces drastically the amount of vertices and triangles stored in memory during extraction allowing larger datasets to be processed.

The same way Attali et al addressed the memory problem raised by larger datasets, we propose in this paper a parallel, bloc-wise extended version of the tandem algorithm to accelerate the computation of simplified isosurfaces.

With our approach, the dataset is split in blocs and sent on computing nodes. On each node a local isosurface is extracted and *semi-simplified* with a slightly modified tandem algorithm. Then nodes might 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 ensures adjacent nodes to be processed together maintaining the memory budget as low as possible during the overall isosurface extraction.

We furthermore introduce an early *components dumping* mechanism that frees the memory as soon as independent surface components have been extracted, simplified and stored on disk along with meta-data for later high level exploration. This strategy has proved to be very useful on particular datasets such as in geophysics where 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 purpose 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 paper 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 on our method.

2 The “Tandem Algorithm”

The main idea of Attali et al’s algorithm is to alternate the extraction of a *layer* (see figure 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

*e-mail: guilhem.dupuy@univ-pau.fr
†e-mail : bruno.jobard@univ-pau.fr
‡e-mail : sebastien.guillon@total.com
§e-mail : noomame.keskes@total.com
¶dimitri.komatitsch@univ-pau.fr

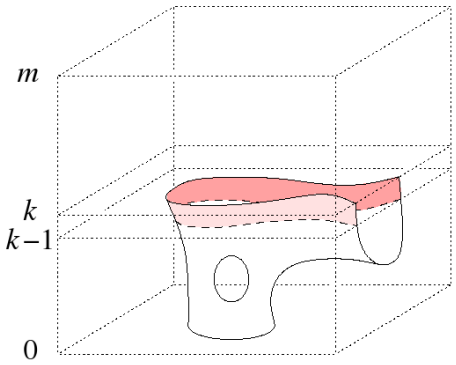


Figure 1: *The k-th layer (in red) is the set of vertices, edges and triangles extracted with the marching cube algorithm between the cross-sections k-1 and k.*

would require to store all vertices and triangles, while a simplification stage during the extraction reduces vertices and triangles number at each step.

The *tandem algorithm* is then a simple loop which iterates over the cross-sections applying two operations:

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

The simplification stage consists in applying the classical *edge collapse* algorithm [Garland and Heckbert 1997]. Each edge collapse operation has a cost which 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 have revisited the quadratic metric of Garland and Heckbert [Garland and Heckbert 1997] 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 the one used in the original edge contraction algorithm. It measures the deviation introduced by collapse operation between the new vertices and the original surface. The *shape measure* of a point x is then defined by

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

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 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 counteract 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 the point c is defined as the square distance of the point to the patch :

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

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 normalize by $W = 3area(S_{ab})W_c^{1/2}/E_0$ in order to balance its influence with h_c in the global cost defined by :

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

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 refers to the original paper [Attali et al. 2005].

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 consists then in emptying the queue $\mathcal{Q}$ by applying consecutive edge collapses.

Just after the extraction stage, the simplification stage has to cope with the heavy edge constraints on the last extracted layer. To counteract the artifacts theses blocked edges would introduce, an innovating way of scheduling the edge collapse was proposed, called *time lag*. The main idea is to delay edge collapses near the advancing front.

The *time lag* is based on the *rank* of a vertex, equals to its coordinate along the extraction direction (for example z if the cross-sections are taken along the z-axis). The *rank of 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 :

$$\begin{aligned} height(c) &= (height(a) + height(b))/2 \\ rad(c) &= (||a-b|| + rad(a) + rad(b))/2 \\ reach(c) &= height(c) + rad(c) \end{aligned}$$

The contraction of an edge is prevented a long as its *reach* value is greater or equal to the rank of the advancing front. As detailed in [Attali et al. 2005], 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.

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 $\mathcal{Q}$. The figure 2 illustrates the effect of the time lag near the advancing front.

Procedure *activate*(k : **integer**)
  While (*reach*(*top*($\mathcal{W}$)) ¡ k) **do**
    add *top*($\mathcal{W}$) in $\mathcal{Q}$;
    pop($\mathcal{W}$);
  done
End

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

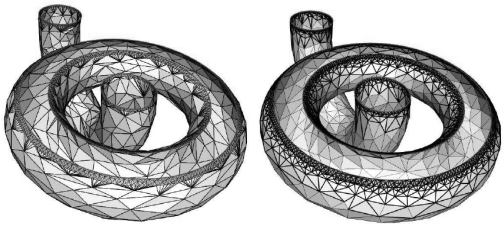


Figure 2: *Two partial triangulations constructed (left) without and (right) with the time lag : edges' length get progressively longer as they get further to the front layer.*

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

```
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: *Tandem Algorithm*

3 Extended Tandem algorithm

Tandem algorithm was written to work on large datasets but experimental results (test on a noisy dataset of size 1626x7028x2000) showed us the limit of the scalability of this algorithm. On this cube, each extraction on the advancing front generate 1 250 000 triangles (see figure 3). In this case, queues' updates and simplification steps are too memory consuming. We propose then 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 sub parts of the dataset.

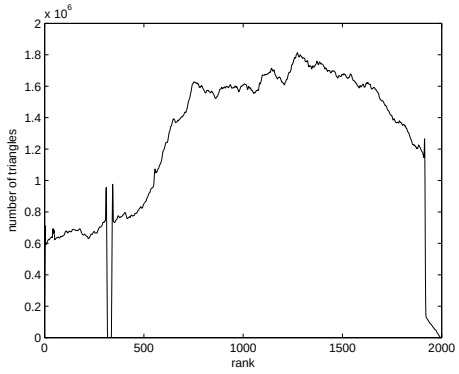


Figure 3: *Number of triangles generated for each layer of the (1626x7028x2000) dataset. The portion with no triangles extracted is due to a hole in the dataset.*

3.1 Components Dumping

Classical out of core approaches migrate mesh to disk during extraction. Generated files are then soups of triangles ordered by direction extraction. Isosurfaces extraction, especially in geophysics, generate many disconnected components. Figure 4 illustrate an usual repartition of geologic events : finished components are shown in light gray and components which are still growing or which 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.

Figure 4: *Distribution of extracted components. Finished components are colored in light grey and growing ones are in dark grey. Black arrow indicates the direction used for extraction.*

The dumping requires to detect the end of a components extraction, but as we use the *time lag* technique, a component could be entirely extracted but its simplification could have been prevented. So we 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 the vertices defining its shape. We consider Γ the set of all the active components, $\Gamma = \{\gamma_i\}$. We 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 $\mathcal{W}$), this proposition could be written as :

$$\gamma \; is \; finalized \Leftrightarrow \gamma \cap V_w = \emptyset$$

We define the function *save*(γ) which migrate a component to disk and the function *clear*(γ) which delete γ in current triangulation. *Dumping* function could be written as in the 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: *Dumping function*

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: *Tandem algorithm with dumping*

An intrinsic property of our component-based dumping is that our generated file is ordered by component. We can then access easily to a sub set of components by reading sub parts of generated file. To optimize access to components in this file (called raw data file) we generate an index file (called index component file). To enhance the exploration of extracted surface, meta-data (oriented bounding box, volume, number of vertices and facets, ...) are computed on each component and stocked in the index file (see figure 5). One exploration scenario could be based on volume filtering like in [Pivot et al. 2007] where Pivot et al propose a workflow for complex volume seismic interpretation. They propose to extract isosurfaces on seismic attributes and to delete automatically inconsistent small "bubbles" (disconnected components with very small volumes regarding to other components). Another useful way to separate components is an analysis based on depositional direction (azimuth direction).

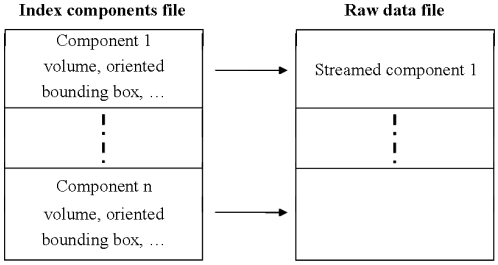


Figure 5: *File organization. On the left, the index file referring to raw data file (on the right).*

3.2 Parallel Algorithm

A feature of the Marching cubes is its granularity, facets of a grid cell (composed by 8 values of the data set) are computed independently of other grid's cells. So we propose to split datasets in sub blocks, to extract simplified surfaces in these sub blocks using the extended tandem algorithm and to merge then in a final surface.

Splitting We propose a hierarchical scheme for surface extraction. As in [Muller and Stark 1993], dataset is split recursively in two parts perpendicularly to its longest direction as long as sub blocks size is greater than a given threshold (see figure 6). Isosurfaces extraction are done on leafs of the generated tree and recursively merge to recompose the global surface.

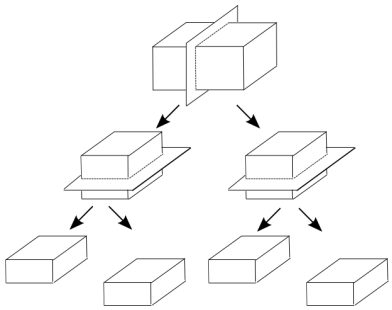


Figure 6: *Binary partitioning*

Extraction in leaf nodes are done using the tandem algorithm combined with the dumping approach. In order to avoid refinement dependencies between adjacent blocks and prevent the artifacts due to bias in shape during simplification, we extend the *time lag* notion. As for original *time lag* the radius of 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 with center on the middle of ab and radius rad(c) is not totally include in the block. We show on the figure 7 how the introduction of the *time lag* influence the refinement progression near borders of the blocks.

Merging In our splitting strategy (binary partitioning), each node p is split in two parts $p1$ and $p2$, called children of p written c_p. Reciprocally p is the father of $p1$ and $p2$ written respectively f_{p1} and f_{p2}. $p1$ is the brother of $p2$ written $b_{p2} = p1$ and reciprocally $b_{p1} = p2$.

Intuitively, the optimal strategy for block merging should be to recursively merge brothers together. In practice components are not equally distributed in sub blocks : extraction time of two brothers could be drastically different, and merging them implies to wait for the slower one. To counteract this problem we allow to merge a block with its brother or with one of its brother's children if they have a common border.

We thus need to define a criterion determining if two blocks have a common border. We consider that a block p could be split by 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 typed respectively *left* and *right*. Similarly, cut along *y* types children as *up* and *down*, and cut along *z* as *front* and *bottom*.

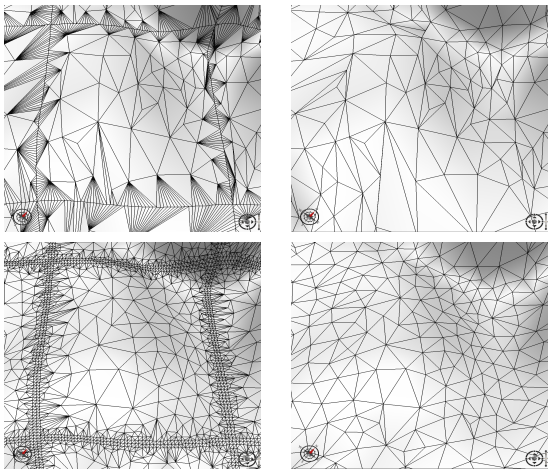


Figure 7: *On the left, figures illustrate the effect of the time lag on refinement blocks' boundaries before simplification. On the right, surfaces have simplified. On the top row, just edges which have a least a vertex on a border have been preserved. On the bottom row, edges have been prevented using the time lag method. This figure shows that applying the time lag method on the borders of blocks leads to better quality triangulations.*

Defining $\overline{.}$ the opposite operator, we have :

$$\overline{left} = right \text{ and } \overline{right} = left$$
$$\overline{up} = down \text{ and } \overline{down} = up$$
$$\overline{front} = bottom \text{ and } \overline{bottom} = front$$

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

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

Considering a block split in $p1$ and $p2$, then a node p belonging to T_{p1} has a common border with $p2$ if

$$\forall\, p' \in A_{p,\,p1},\ type(p') \neq \overline{type(p2)}$$

Then if p fulfills this requirement it could be merge with $p2$.

Merging two sub blocks requires first to identify common points extracted on each block. Due to numerical inaccuracy in vertices extraction, we cannot identify common points with a simple coordinates comparison. Identification is then done by using a property of the Marching cubes algorithm : as a cell edge could have only one or zero points generated by the algorithm, 2 points belonging to common cell edges are identical and then merged.
Once this merging step done, one has to simplify pasted area with the same error criteria as for the surface extraction E_0. Like during the extraction step, the contraction of an edge ab is prevented as long as the sphere with center on the middle of ab and radius rad(c) is not totally included in merged blocks. We show on the figure 7 the merge of surfaces and the effect of the *time lag* on the quality of result surfaces. Finally the last step of the merge consists in dumping finished components on the disk.

3.3 Master/Slaves Implementation

In our application the target parallel machine is a *load balancing cluster* managed by Platform LSF-HPC. The common use of this parallel machine is to decompose the process in independant tasks, the task scheduler (hear LSF-HPC) dispatch tasks on the distant grid. When a task is finished, processor used for this task is liberated. The task scheduler could realloc it for the process if resources are sufficient for other running processes.

We propose an architecture composed of *workers* and a *worker manager*. *Workers* are distant processes which have to extract, merge and dump surface. The *worker manager* is a process, distant or not, which assign tasks to *workers*. This architecture, and our formalism, is adapted to our target parallel machine (number of allocated processor is time-varying) but could used with others parallel machines.

Workers could be written as an infinite loop which request *tasks* from the *worker manager*. *Tasks* are composed of a *task type* and a *worker id* (each worker as an unique id use to send messages). We distinguish four *task types* :

- *EXTRACT* : *worker* has to extract surface from a sub block of the dataset. The extraction is made with the extended tandem algorithm. Sub block borders, excepted the ones which are common to dataset borders, would prevent the *edge collapse* at their proximity using *time lag*. The extracted surface is appended to its current surface.
- *SEND* : *worker* received a *worker id*, referring to a *target worker*. *Worker* has to send its current mesh to the *target worker* which would merge it with its current one. After this sent, current surface of the worker is emptied.
- *MERGE* : *worker* has to merge its current surface with a mesh sent by another *worker*. After the merge operation, the new current surface is simplified and finished components are dumped.
- *FINISHED* : *worker* has to stop. The time life of a *worker* is not necessary equal to global process duration.

The algorithm 5 describes the pseudo code of the *worker*.

```
Procedure worker( E0 : float)
    finished : boolean;
    finished ← false;
    While ( ¬finished) do
        task ← get_task();
        Switch
            task.type=EXTRACT:
            tandem(E0);

            task.type=SEND:
            send_surface(task.target);

            task.type=MERGE:
            merge_surface(get_surface(), E0);

            task.type=FINISHED:
            finished ← true;

        end Switch
    done
End
```

Algorithm 5: *Worker*

Worker manager is a loop which iterate as long as the global surface

has not been totally extracted and recomposed. *Worker manager* has a global view of the process and can dynamically dispatch tasks to *workers*, which ask for tasks when they begin or when they have finished their last given task. In order to define the *worker manager* mechanism (algorithm 6), we define the three following operations : *has_block*, *has_neighbor* and *has_surfaces*.

Function *has_block* determines if there's some cubes which have not been extracted. If some cubes have not been extracted the *worker manager* send it to a *worker*. If current *worker* have a triangulation which could be merge, un-extracted block candidates must have a common border with blocks of the *worker*. If current *worker* have no triangulation, one of the un-extracted block is chosen.

Function *has_neighbor* determines if a querying *worker* has a neighbor in the current working *workers*. If there's some neighboring *workers*, querying *worker* has to send its surface to one of them. If no neighboring *worker* is found, the *worker manager* look for a sub block to extract.

Function *has_surface* determines if a querying *worker* has some waiting surfaces sent by other workers. If there's some surfaces, the query *worker* has to merge them with it current surface. This operation has priority due to memory consuming minimization policy.

```
Procedure worker_manager()
    While ( ¬finished) do
        id ← get_task_request_id();
        If (has_surface(id)) then
            send_task(id, MERGE);
        else
            If (has_neighbor(id) ) then
                send_task(id, SEND, targetId);
            else
                If (has_block() ) then
                    send_task(id, EXTRACT);
                else
                    send_task(id, FINISHED);
                end If
            end If
        end If
    done
End
```

Algorithm 6: *Worker manager*

4 Computational Experiments

In order to analyse performances of our algorithm, we focus on the memory consuming during the extraction and the quality of the generated mesh.

Memory consuming A critical point in Marching cubes is the amount of memory needed for the generated triangles. The figure 8 illustrates the comparison between brute-force approach, tandem algorithm and tandem algorithm with dumping. Triangles are counted right after the process of a layer k. We see that tandem reduce drastically the number of triangles generated comparing to the classical Approach. Our dumping strategy enhances the tandem algorithm by clearly reducing number of triangles.

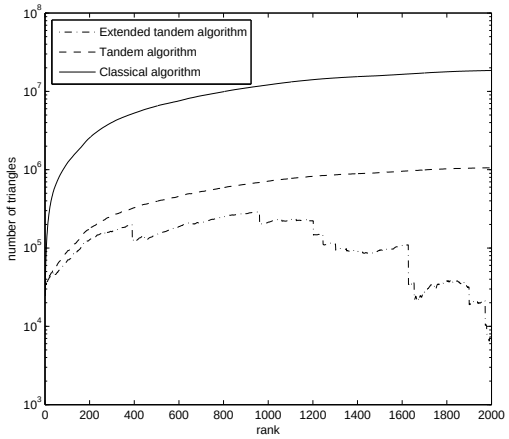


Figure 8: *Evolving size of the triangulated surface for the three algorithms : classical algorithm (extraction of the entire surface and simplification of it), tandem algorithm and tandem algorithm with dumping of finished components. Computations are done on a cube of size 401x1051x2001.*

Mesh quality Many applications need "well-shaped" triangles instead of long and skinny ones. A well-known evaluation of the *aspect ratio* of a triangle $t = abc$ is $\rho(t) = \sqrt{\lambda_2/\lambda_1}$ where $\lambda_1 \geq \lambda_2$ are the greatest eigenvalues of the *inertia matrix* of t. This matrix is defined by :

$$M_t = \frac{1}{3}[(a-\hat{t})(a-\hat{t})^T + (b-\hat{t})(b-\hat{t})^T + (c-\hat{t})(c-\hat{t})^T]$$

with $\hat{t}$ the centroid of the triangle t. $\rho(t) = 1$ if the triangle t is equilateral ($\lambda_1 = \lambda_2$) and $\rho(t) = 0$ if it is degenerate ($\lambda_2 = 0$). As a global measure of the triangulation K (composed by n triangles), Attali et al propose a formulation of the *anisotropy* :

$$anisotropy(K) = 1 - \frac{1}{n}\sum_t \rho(t)$$

with $0 \leq anisotropy(K) \leq 1$ and $anisotropy(K) = 0$ if all triangles are equilateral.

The figure 9 shows *anisotropy* variation in function of the *isotropy parameter* α for the three approaches : sequential tandem algorithm, parallel algorithm with and without time lag. We clearly see effect of the *time lag* on the quality of the generated meshes. Parallel extraction generates a mesh with an equivalent quality as the sequential approach. The increasing of anisotropy (α greater than 0.9) is due to the fact that the shape criterion is no more took into account in the choice of candidate for edge collapse. Then the collapse operation is only based on triangle shape quality, and proposed points far from the original surface : most of those candidate are rejected by the error threshold E_0. In practice, cost function must always take care of the *shape measure* and then prevent these degenerated cases : a value of α equals to 0.4 is a good compromise.

Application to geophysics Seismic cubes are very common data used on hydrocarbon exploration and analysis of those data is very helpful to specify depositional environment. A common tool for seismic analysis is based on texture analysis and seismic attributes computation. A seismic attributes highlight specific features of the image. For example figure 10 shows an example of

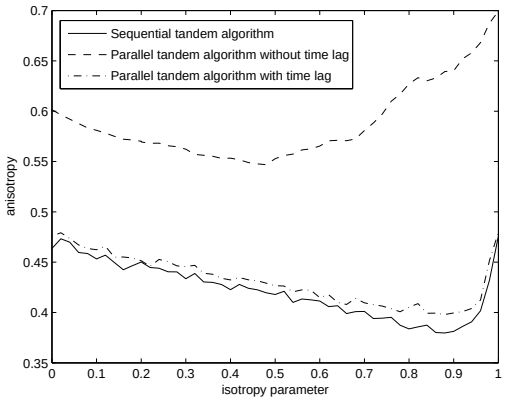


Figure 9: *Anisotropy of the mesh variation in function of the isotropy parameter* α.

attribute which detect all the chaotic and high amplitude area of the seismic data. As those facies could be related to the presence of hydrocarbon, making an inventory of all these area is very helpful for geophysicists. In order to make this inventory, we apply the following workflow : attribute computation, surface extraction and disconnected components sorting. As the attribute highlight the interesting part, then by adjusting a threshold we extract all the 3D shapes with our proposed approach (see result on figure 10).

This workflow is applied to 1626x7028x2000 dataset (40 Go). The first interest of our approach is its parallelism : according to the number of available processor, we reduce the extraction time (the extraction have been performed in 5 minutes with 56 processors instead of more than 3 hours for the tandem). The figure 11 shows the all extracted surfaces, and can see clearly a lot of small extracted objects : these objects could be due to noise on the attribute or to very small objects. In order to make a ranking of all these surfaces, we can easily sort them according to geometric criteria. Figure 12 illustrates this sorting by elimination of small objects. Only 328 components are now selected instead of 18 111. This workflow is very helpful to locate all these area and to analyse their repartition and relation.

5 Conclusion

Due to the increasing size of the datasets, the literature on iso-surfaces extraction has focused on approaches which propose to extract simplified surfaces or to dump surfaces as a soup of triangles. But none of these approaches propose parallel extraction with simplification and dumping of disconnected components. Our component-based dumping approach induces a file organization allowing interactive exploration of the volume. This organization could be applied on all surface reconstruction algorithms which generate many disconnected components. Our parallel processing based on dataset splitting and time lag extension could benefit to simplification methods which split their surfaces into patches.

Acknowledgments

This work was supported by Total company.

References

ATTALI, D., COHEN-STEINER, D., AND EDELSBRUNNER, H. 2005. Extraction and simplification of iso-surfaces in tandem. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 139.

GARLAND, M., AND HECKBERT, P. S. 1997. Surface simplification using quadric error metrics. *Computer Graphics 31*, Annual Conference Series, 209–216.

GELDER, A. V., AND WILHELMS, J. 1994. Topological considerations in isosurface generation. *ACM Transactions on Graphics 13*, 4, 337–375.

LIVNAT, Y., SHEN, H.-W., AND JOHNSON, C. R. 1996. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics 2*, 1, 73–84.

LORENSEN, W. E., AND CLINE, H. E. 1987. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 163–169.

MACKERRAS, P. 1992. A fast parallel marching-cubes implementation on the Fujitsu AP1000. Tech. Rep. TR-CS-92-10, Canberra 0200 ACT, Australia.

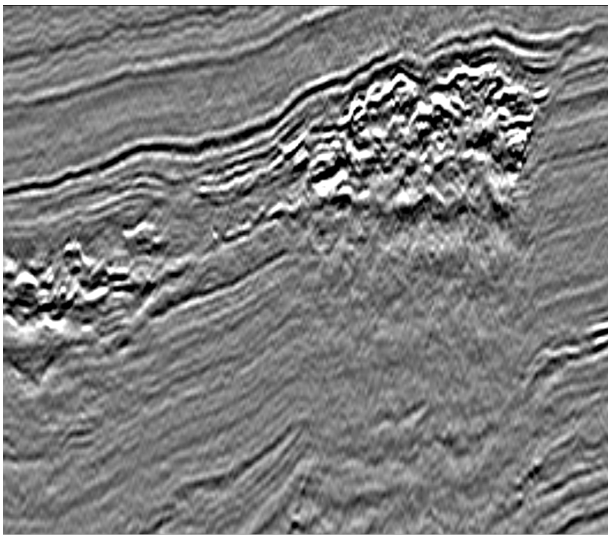
MULLER, H., AND STARK, M., 1993. Adaptive generation of surfaces in volume data.

NIGUET, S., AND NICOD, J.-M. 1995. A load-balanced parallel implementation of the marching-cubes algorithm. Tech. Rep. 95-24, Laboratoire de l'informatique du parallélisme de l'École Normale Supérieure de Lyon.

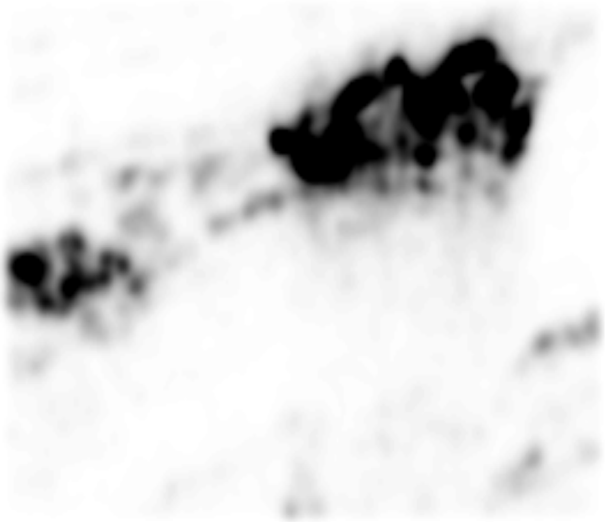
PIVOT, F., DUPUY, G., GUILLON, S., AND FERRY, J. N. 2007. Complex volumic seismic interpretation by new geobodies extraction strategy. In *London 2007, 69th EAGE Conference Exhibition incorporating SPE EUROPEC 2007*.

SHEKHAR, R., FAYYAD, E., YAGEL, R., AND CORNHILL, J. F. 1996. Octree-based decimation of marching cubes surfaces. In *IEEE Visualization*, 335–342.

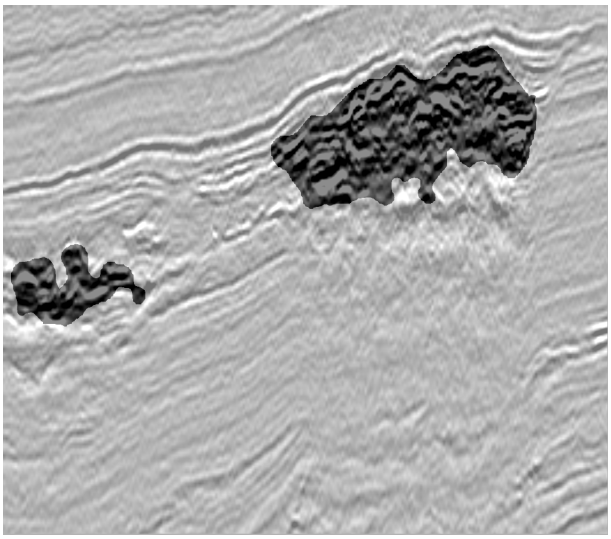
WILHELMS, J., AND GELDER, A. V. 1992. Octrees for faster isosurface generation. *ACM Trans. Graph. 11*, 3, 201–227.



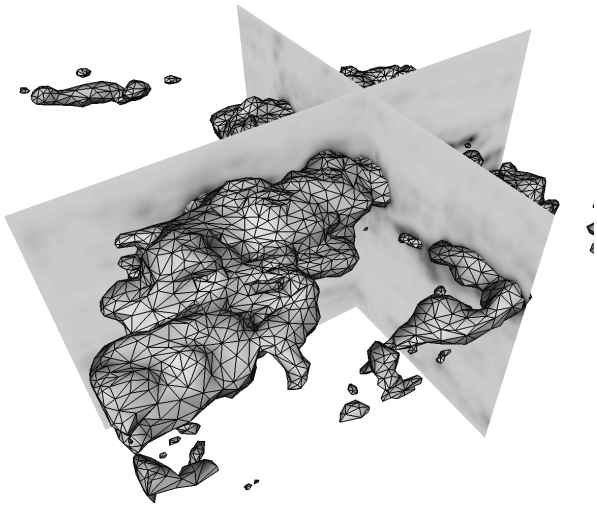
(a)



(b)



(c)



(d)

Figure 10: *Seismic attribute. (a) Cross sectional view of a seismic dataset. (b) Cross sectional view of an attribute computed on the seismic dataset. (c) Superposition of the attribute (color palette is saturated at a given threshold) over the seismic. (d) Global view of the extracted surfaces on the seismic attribute.*

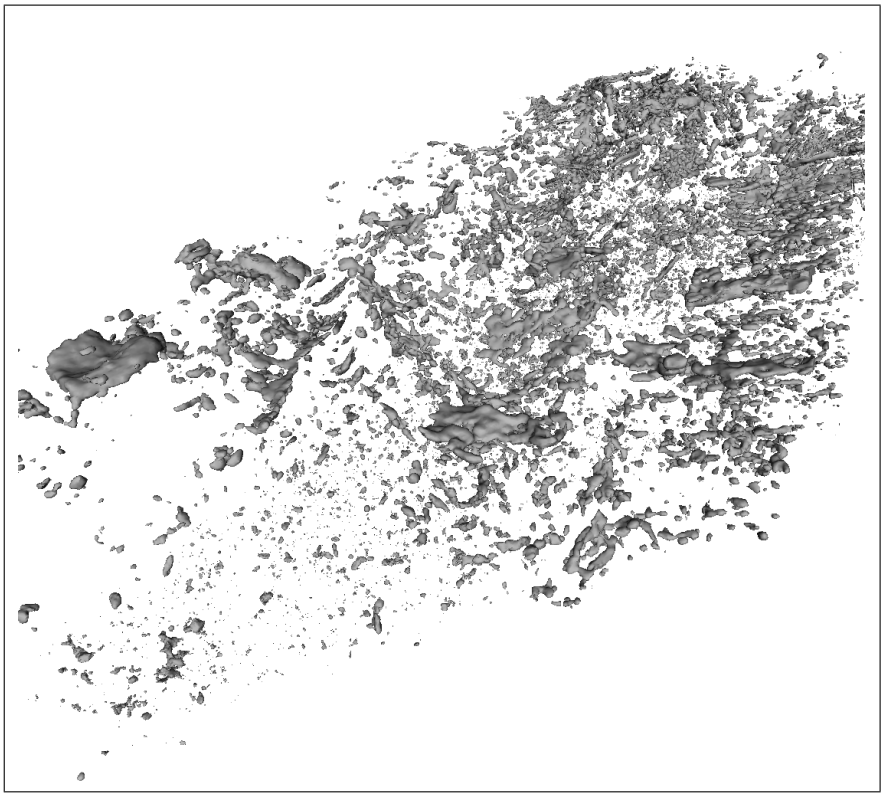


Figure 11: *Example of surface generated on a dataset of size 1626x7028x2000. This surface contains 18 111 disconnected components.*

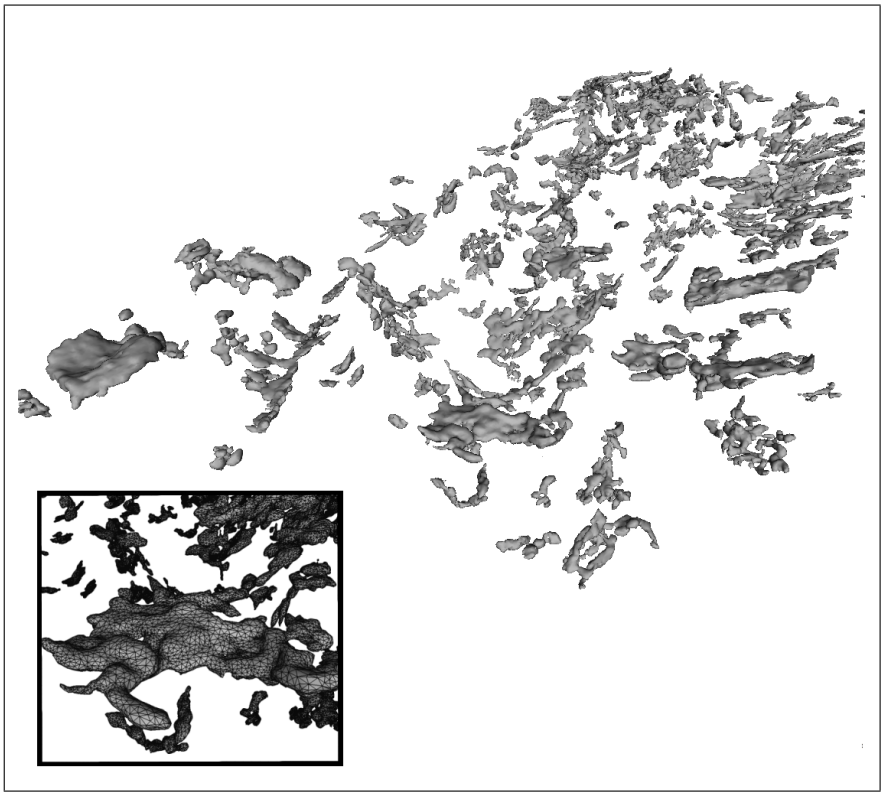


Figure 12: *Components extracted on dataset of size 1626x7028x2000 (see figure 11) have been filtered according to their volumes. This surface contains 328 disconnected components.*