# Simplification of 3D Point Clouds sampled from Elevation Surfaces

Van-Sinh NGUYEN
Aix-Marseille University
CNRS, Laboratory LSIS
UMR 7296, Marseille, France
Tel: 33.4.91.82.85.28
van-sinh.nguyen@univ-amu.fr

Alexandra BAC
Aix-Marseille University
CNRS, Laboratory LSIS
UMR 7296, Marseille, France
Tel: 33.4.91.82.85.32
alexandra.bac@univ-amu.fr

Marc DANIEL
Aix-Marseille University
CNRS, Laboratory LSIS
UMR 7296, Marseille, France
Tel: 33.4.91.82.85.25
marc.daniel@univ-amu.fr

## ABSTRACT

This paper introduces a new technique to simplify a 3D point cloud sampled from an elevation surface and organized in voxels. The method consists of three steps: in a first step, the boundary of the surface is extracted and simplified; in a second optional step, we roughly simplify the surface inside its boundary; in a third step, we present an elaborate method for simplification while keeping its boundary. Our method preserves the distribution of points, the initial geometry and characteristics of the surface, even with high simplification rates.

## Keywords

Boundary Extraction, Boundary Simplification, Surface Simplification, Principal Component Analysis (PCA).

## 1 INTRODUCTION

Simplification of a 3D point cloud belonging to a surface is an important steps in geometric modeling and surface processing. The purpose of surface simplification of a 3D point cloud is to reduce the number of points, save the memory, improve the effect of computation and optimize the processing of the geometric model. During simplification, the original shape of the surface must be kept, without shrinking or deformations.

Nowadays, the modern 3D acquisition and modeling technology allow producing a large amount of point samples from real-world objects. Different existing researches (and especially for meshes) are available for processing of the continuous surfaces, but the case of 3D point clouds simplification remains a challenging issue.

Our problem originates in the questions of processing large 3D point clouds issued from a seismic data (themselves extracted from a 3D sparse volume [Philippe09]). The seismic acquisition does not permit to measure all the points in the 3D volume, explaining the fact that the 3D volume is sparse. The 3D points are actually stored in a voxel structure in this volume (each voxel is considered as a 3D point, and has three real coordinates xyz), hence implicitly the 3D volume contains neighboring information even in a sparse context.

Most existing approaches have a common drawback: in the case of open surfaces (that is surfaces with boundaries), simplification induces a shrinking of the surface. Hence, in order to preserve the initial shape, our approach starts by an extraction and simplification step of the boundary. In a previous work, we have proposed a method for extracting and simplifying the boundary of a surface [Sinh12]. The present paper continues this work and introduces a method to simplify the inside of this surface. To handle potentially huge clouds, our method consists of two steps: an optional initial rough simplification (basically designed to adjust the sampling rate) followed by a more elaborated simplification step. As the point cloud is sampled from elevation surfaces, points are first projected onto a 2D grid in xy plane to process with the first step, while the second step is directly processed in the 3D grid.

The remainder of this paper is organized as follows: in section 2, we present work related to surface simplification of a 3D point cloud. We present our method in detail, which includes problem analysis, building the criteria and implementing the algorithms in sections 3. The results and evaluation of our method are presented in sections 4 and 5. The last section is our conclusion.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

## 2 RELATED WORKS

Different existing methods which have been studied and developed are not only applied to sim-

 ISBN 978-80-86943-75-6

plify the surface of 3D point clouds, but also applied to simplify the surface of triangular mesh [Garland99, Pauly02, Van06, HaoSong09, Zhe07]. Among them, PCA (Principle Components Analysis) is a popular tool, a well known method that can be used to simplify the surface of 3D point clouds [Mederos04, Yu06, Alexandra07, David08].

Garland et al (1999) [Garland99] developed an algorithm to simplify the surface of a polygonal model based on the iterative contraction of vertex pairs. Starting from the initial model M1, an edge $e_{v1,v2}$ will be contracted to a new position $\bar{v}$ if the distance $\|v_1 - v_2\| <$ threshold. The process is repeated until the simplification goals are satisfying. The last model M2 approximates M1. In order to preserve the shape of the surface and optimize the placement of vertices after contraction, the authors used the quadric error matrices to track the approximate error of the model. This method is time and memory demanding, but it avoids distortion of the original shape. However, evaluation of the quadratic error metric is closely related to the mesh structure (and to the face neighborhoods). Hence, it cannot easily be adapted in our setting.

Pauly et al (2002) [Pauly02] introduced, analyzed, compared and implemented a number of methods to simplify the surface of 3D point clouds. One of these methods is called “Clustering". The surface of 3D point clouds is clustered by splitting it into a subset of points; then, replace all points in each cluster by one representative point. This region-growing is terminated when the size of the cluster reaches the maximum bound. This method leads to simplifying the surface effectively. However, each cluster is a sphere with a radius $\alpha$ on the surface. Therefore, the points outside these clusters are not simplified completely after the iterative processing.

Boris et al (2004) [Mederos04] proposed a method to reconstruct and smooth a surface from noisy point clouds. At first, he smoothed the original point clouds to reduce the noisy points by using a robust projection procedure, while keeping the shape of the surface. The next step, data of 3D point clouds are clustered by partitioning into a subset of clusters. Then, he applied PCA to analyze, reduce the size of the original points, and determine a representative point for each cluster. In the next step, a triangular surface is obtained from the representative points of each cluster to obtain a rough surface which approximates the original surface. The last step, this rough surface is refined to get an optimal one. This is a complete method for surface reconstruction of a point cloud. However, the computing is complex during projecting, clustering, reducing, meshing and refining the point clouds, leading to a computation heavy and costly.

Normally, to simplify the surface of 3D point clouds, the existing approaches aim to cluster a subset of points, and then grow on the surface to simplify. The problem is how to determine the neighboring points in a local region of the surface. Y.J Zhang et al (2010) [Zhang10] proposed a way to define the nearest neighbouring points by using a cylinder. The points are dropped into a bounding cylinder based on the specified threshold (the radius of the cylinder); then, they are projected on the line as its center axis to simplify the points inside. The same as method [Pauly02], for each iterative step, the outside points are not simplified completely.

Frey et al (2007) [Frey07] presented a method (the “affinity propagation") to cluster by passing messages between data points. This method measures the similarity of each point-pair of the input data points. Each point in a point set is assigned as a node of a network, the real-valued messages are exchanged between data points (nodes) along the edge of the network until a high-quality set of exemplars corresponds to the cluster which gradually emerge. However, the cost of computing is expensive because the transmission process between the points is computed recursively.

Jae-Young et al (2005) [Jae05] and Tamal et al (2011) [Tamal11] introduced a method by using an octree partitioning to divide the point clouds into a small subset, then process on each subset as a node of an octree on 3D space and quadtree on the 2D grid. At first, a root node of a point cloud is divided into four in 2D or eight in 3D. Then, the child nodes are recursively divided until satisfying the condition of the threshold. After that, each node can be considered as a point during the simplification.

Morales et al (2010) [Morales10] suggested a method to smooth and decimate the points from an unstructured point cloud based on the radial based function (RBF). The points are computed based on the kd-tree nearest neighbors. Starting from a seed point $p_i$, the neighboring points ($p_n$) are calculated by an Euclidean distance $\|p_i - p_n\|$ to determine the radius r. All points within r are mapped from a 3D point set to the 2D space; the point set components are mapped into each axis plane on each square matrix MxMx3 in domain $N_{ix}, N_{iy}, N_{iz}$. The next step is using a convolution Gaussian Kernels function $C = M \otimes G\left(\mu, \sigma^{d(k)}\right)$ for each axis $N_{ij}$ to smooth and estimate the new center point in each component $p'_{x,y,z}$. Finally, the 3D point sets are smoothed and simplified according to the local surface features.

As we have described and analyzed, the above methods are suitable for dispersive data or unorganized point clouds but lead to an expensive computation. In our work, we take advantage on the structure of voxels and their neighborhood information. We can adapt these methods to simplify the surface efficiently; preserve the shape and point distribution of the surface.

 ISBN 978-80-86943-75-6

# 3 OUR METHOD

Our method consists of three steps (see figure 1). The first step (boundary extraction and simplification), we have presented in the previous work [Sinh12]. In this paper, we present the second and the third step for simplifying the surface inside its boundary.

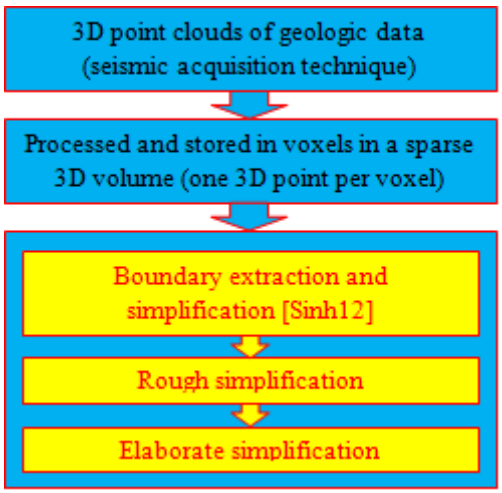


Figure 1: A method for simplifying the surface.

It is interesting to summarize the main idea of the method we applied first to extract and simplify the border of a surface [Sinh12]. We define a method to extract the boundary based on k_square neighborhood of each point up to a fixed integer distance $k$. Our algorithm starts from an initial boundary point of the surface; then, an exterior boundary is built point by point by iteratively computing the next point via growth functions. After that, we build an algorithm to simplify this boundary by first study the alignment of points and second study the variation of elevation. In our method, the complexity of algorithms is proved more efficient than existing methods. Moreover the initial shapes of the surface are also preserved for the simplification step since the boundaries are kept.

## 3.1 Rough simplification

### *3.1.1 Overview*

Rough simplification is a preliminary step designed to handle large point clouds: points are imported in a fine regular grid and each non empty voxel is replaced by a single representative vertex. Hence, the goal of this step is merely to adjust sampling density. In this algorithm, 3D point clouds (organized in a sparse 3D regular grid) are first projected onto the 2D grid in the $x, y$ plane. This 2D point cloud (set of non empty voxels) is subdivided according to a regular grid of size $s$ (this size is defined by the user according to the desired final sampling rate) (see figure 2a). Then, each non-empty cell is replaced by a single representative point: the barycenter of contained points. This step, even if rough, can be justified in terms of resolution: it is merely a resolution adaptation (in case the resolution of the data is too high compared to the expected results). The important point in this step is that we will not simplify boundary points (as they have already been handled in the previous work [Sinh12]); and this step should be applied using a small size of cells in order to avoid distorting the surface.

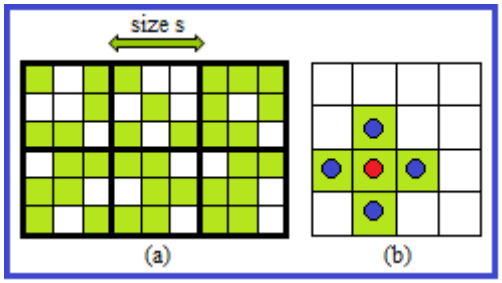


Figure 2: a) The size of a cell. b) The barycenter of the points (red color) in the cell.

### *3.1.2 Notation*

In the sequel, we use the following notations:

- $G$: the 2D initial regular grid,
- $C$: the regular grid of size $s$ built over G,
- $S$: the subset of cells in C which are non empty,
- $S_q$: a cell on the 2D grid belonging to S,
- $p_q$: barycenter of the points included in $S_q$.

### *3.1.3 Algorithm*

As the size of the cells is small and as we want to preserve boundary points, if a cell contains boundary points, no further representative vertex will be inserted, only included boundary points are kept. Otherwise, if a cell does not contain boundary points, we compute the barycenter of the points in this cell. Based on the above description, we propose a very simple algorithm (Algorithm 1) with a linear complexity to roughly simplify the surface.

**Algorithm 1** roughSimplification(s)

```
1: for each cell S_q ∈ S do
2:     if S_q contains boundary points then
3:         keep only boundary points;
4:     else
5:         replace all points by p_q;
6:     end if
7: end for
```

## 3.2 Elaborate simplification

### *3.2.1 Overview*

In this step, we focus on two main points to process the surface: curvature of the surface and point density. We

 ISBN 978-80-86943-75-6

process the surface directly in the 3D grid. As previously, the sparse 3D grid (equivalent to the point cloud) is divided according to a regular 3D grid $C$. The initial size of the cells of $C$ is large (defined by the user) and elaborate simplification will further subdivide cells of $C$ according to density and curvature criteria. If cells contains boundary points, they are processed based on the combination between boundary density and local curvature in these cells. Otherwise, subdivision is based on local curvature within each cell and adapted to the size of neighboring cells. After simplification, the distribution of points has to vary continuously; it must be constrained regularly from the exterior boundary to the inside of the surface. This constraint is introduced to avoid creating bad triangles (in the sense of Delaunay triangulation) in a further meshing step.

### 3.2.2 *Analysis*

Obviously, our rough preliminary simplification is too basic to reach high simplification rates. It is useful only to adjust the resolution or as a first decimation for huge point clouds (for which a more elaborate simplification cannot be applied directly because of time and space complexity issues). Hence, this preliminary step is optional.
In the case of complex surfaces with a high curvature, simplification must be based both on density and curvature criteria. For this reason, we develop an advanced algorithm to simplify the surface more elaborately. This algorithm is based on an octree subdivision of the surface adapted to its curvature, point density and to the border density. We will combine two subdivision criteria to simplify the surface: subdivision according to the boundary density and subdivision according to the curvature.

### 3.2.3 *Subdivision according to the boundary density*

An important issue is that point density should vary "smoothly" (in order to preserve the shape of triangles in a further meshing step). It must be constrained continuously on the surface and propagate regularly from the boundary to the inside of the surface. Therefore, in this paper we propose a method to simplify the surface inside its boundary. In order to subdivide cells according to the boundary density, we have to build a subdivision criterion. At first, we analyze the density of boundary points (number of boundary points in a cell) and their distribution. Our criterion is based on the size of a cell, the number of boundary points and the distance between them.

**a) Notation and formula construction**

We will use the following notations:

- $C_q$: a cell (size $s$) in the 3D grid,
- $Nbp$: the number of boundary points in $C_q$,
- $d_{max}$: the maximum distance between two boundary points in $C_q$,
- $L_s$: the level of subdivision of a cell (see figure 3),
- $s'$: the size of the smaller cells after each subdivision of $C_q$: $s' = \frac{s}{2^{L_s}}$.
- $p_i, p_j$: point $i^{th}$, point $j^{th}$ of $C_q$.

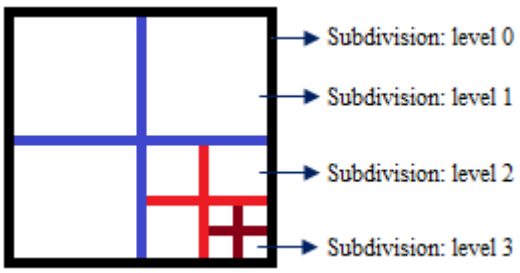


Figure 3: The level of subdivision in a cell.

In our context, data points are organized based on a 3D grid structure, each point in a cell has xyz coordinates and in the sequel, we will use the Euclidean distance to compute the distance between points. Hence the maximum distance between boundary points in a cell is given by:

$$d_{max} = \max_{i,j \in (1..Nbp); i \neq j} \left( \left\| p_i - p_j \right\| \right) \qquad (1)$$

**b) Boundary density criteria**

Subdivision according to boundary density is performed from cells containing boundary points (called first ring) towards the surface interior (ring by ring, starting from the boundary). In the sequel, we will denote by $r_i$ the $i^{th}$ ring of cells based on the 8-connectivity (hence, $r_1$ is the set of boundary cells). There is a relationship between the density of points and the distance between them in a cell. Obviously, as the density of boundary points in a cell increase, the distance between them will decrease. The formula: $D(density) = Np(number\,of\,points)/V(volume)$ can be applied to compute the density of points on a volume. In our case, we focused on the number of boundary points $Nbp$ in a cell and its size $s$ to calculate point density $PD$ of that cell ($PD = Nbp/s$). Hence our criterion is based on $PD$ and $d_{max}$:

$$\left(PD > threshold_{pd}\right) and \left(d_{max} > threshold_d\right) \qquad (2)$$

In order to preserve the shape of the surface for a further triangular meshing step, the size of cells must vary smoothly. Therefore, for boundary cells (also called

 ISBN 978-80-86943-75-6

first ring cells), we state a specific subdivision criterion: if a cell $C_q$ (containing boundary points) satisfies the first condition (2), then we check the size of $C_q$. If the size is less than or equal than a threshold, we keep only boundary points; else, we keep boundary points and the barycenter of inner points in that cell. Otherwise, $C_q$ is subdivided (as an octree).
Starting from the second ring (which contains inner points of the surface), we subdivide cells both according to the local curvature and previous ring cell sizes. Cells are processed ring by ring from the outside to the inside of the surface. The cell size in ring $r_i$ is subdivided according to the sizes of neighboring cells of ring $r_{i-1}$ (the outside adjacent ring of $r_i$). It means that, if an inner cell satisfies the curvature criterion, we subdivide it according to the average subdivision level of all nearest neighboring cells. Let $C_q \in r_i$ and let $\{C_1^{i-1},\ldots,C_m^{i-1}\}$ be the set of neighboring cells in $r_{i-1}$, the subdivision level of $C_q$ is computed as:

$$size(C_q) = \frac{1}{m}\sum_{j=1}^{m} size(C_j^{i-1}) \tag{3}$$

In the end, the cell size varies smoothly; and if the curvature inside a cell is low, all points in this cell are replaced by one representative point. In next section, we build a flatness criteria in order to subdivide cells according to their curvature.

### *3.2.4 Subdivision according to the curvature*

Our goal is to preserve the shape of the surface after simplification. In this part we process the cells containing inner points, from the second ring to the inside of surface. For each cell we apply a principal component analysis (PCA) to estimate the average local curvature of the surface. We thus define a flatness criterion and subdivide cells accordingly.

#### a) PCA flatness criteria

PCA can be used as a useful statistical method to analyze data. This is a technique that can be applied to simplify a surface of 3D point clouds (see [Pauly02, Mederos04, Alexandra07, Zhe07, MZhang11]). In order to estimate the curvature/flatness of a cell, we compute the PCA of the vertices of the cell. The eigenvalues of the corresponding covariance matrix provide a curvature information and we define accordingly a flatness criterion. Cells that do not meet this flatness criterion are subdivided until either their size is lesser than a threshold or they satisfy the criterion.
We use the formula below to compute the covariance matrix for each cell:

$$C = \frac{1}{N}\sum_{i=1}^{N}(p_i-\bar{p})(p_i-\bar{p})^t; \tag{4}$$

Where:

- $N$: a set of points in each $C_q$,
- $\bar{p}$: barycenter of points in $C_q$,
- $\lambda_i$, $v_i$: the $i^{th}$ eigenvalue and $i^{th}$ eigenvector of $C$.

The eigenvectors of C provide information about the principal directions of a point set. More precisely, the eigenvectors provide main axes of the cloud, while eigenvalues provide its stretching along the corresponding axes. Hence, the eigenvector associated to the smallest eigenvalue provides an average normal vector while both other eigenvalues are related to principal curvatures.
Following the above analysis and applying the ideas introduced in [Pauly02, Mederos04, David08, MZhang11], let us sort eigenvalues: $\lambda_0 \leq \lambda_1 \leq \lambda_2$. If the value of $\lambda_0$ is very small or even equal 0, that means all the points in a cell are approximately on a plane (it satisfied the flatness criteria). In such a case, the average normal vector on a local surface within a cell can be determined based on the direction of $v_0$. The **flatness criterion** "$\partial$" below is considered as a condition to further subdivide cells (and hence to control the simplification of the surface):

$$\partial = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2} \tag{5}$$

For each point on the local surface, if their normal vectors are distributed isotropically, these points will lie on the same plane. This solution is given by Hugues Hoppe [Hoppe92] to compute the orientation of the tangent plane: for each data point $p_i$, a tangent plane is computed by least-squares approximation based on PCA of the $k$ nearest neighbor of $p_i$.
In our case, we use the flatness criteria(5) to estimate the local curvature in a cell. The minimum value of $\partial$ equal 0, while its maximum value equal $1/3$, and our flatness criteria is based on the range of these values. (see figure 4)

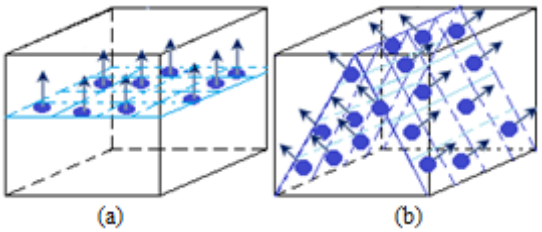


Figure 4: Estimation of the curvature in a cell: (a) The points are approximately on a plane within a cell ($\lambda_0$ is very small, $\lambda_1$ and $\lambda_2$ are large); (b) $\lambda_0$ is large or ($\lambda_0 \simeq \lambda_1 \simeq \lambda_2 \simeq 1$) or ($\partial \simeq 1/3$) $\Rightarrow$ this cell is subdivided.

The curvature in a cell is first determined by computing $\partial$. Then, $\partial$ is compared with a threshold value from the user. If $\partial \leq threshold_\partial$, we replace all points in this cell by one representative point. This way can simplify

 ISBN 978-80-86943-75-6

the surface efficiently and the ratio of simplification is very high (if the points in that cell are approximately on a plane). However, the density of points could vary irregularly after a large number of points have been removed. For this reason, we have to combine with the computation of point density and size of cells to constrain the distribution of points on the surface to be as regular as possible.

### 3.2.5 *Algorithm*

According to the previous analysis, we now define our simplification algorithm. Our algorithm covers cells ring by ring (starting from boundary cells), each ring is processed clockwise (see figure 5).
We start from the first ring, blue color (i.e. the ring of boundary points). In this ring, we begin with the left-most cell (1) and follow the clockwise direction to compute, subdivide and simplify each cell. From the second ring (yellow color), we also begin with the left-most cell (2) and so forth for following rings (third - green, fourth - pink, etc). The algorithms below are used to simplify the surface: algorithm 2 is used to process the cells containing boundary points in the first ring.

**Algorithm 2** SimplifyBoundaryCells(s)

```
1:  Nbp = 0, L_s = 0; //start from the left-most cell, follow the clockwise direcrion.
2:  for each boundary cell C_q(size s) ∈ S do
3:      compute Nbp, d_max;
4:      if C_q satisfy the density criteria(2) then
5:          if size s ≤ threshold_s then
6:              keep only boundary points;
7:          else
8:              replace all points by boundary points and the barycenter of inner points;
9:          end if
10:     else //subdivide C_q by L_s.
11:         L_s = L_s + 1;
12:         s' = s/(pow(2, Ls));
13:         for each C_q(s') ∈ C_q(s) do
14:             if C_q(s') contains boundary points then
15:                 SimplifyBoundaryCells(s');
16:             else
17:                 SimplifyInnerCells(s');
18:             end if
19:         end for
20:     end if
21: end for
```

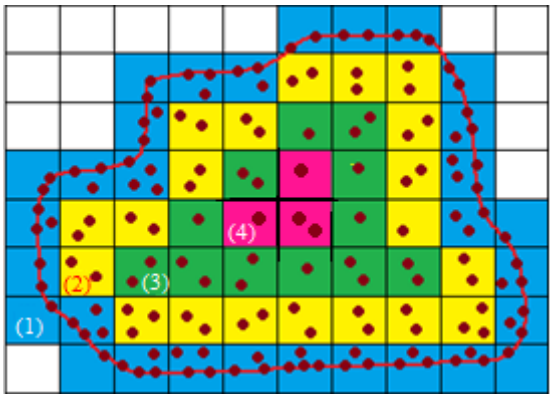


Figure 5: Illustration of the elaborate algorithm.

Algorithm 3 is used to process the cells containing inner points from the second ring to the inside of surface.

**Algorithm 3** SimplifyInnerCells(s)

```
1:  L_s = 0; //start from the left-most cell, follow the clockwise direcrion.
2:  for each inner cell C_q (size s) ∈ S do
3:      compute the covariance matrix of points in C_q;
4:      if C_q satisfy the flatness criteria(5) then
5:          subdivide C_q based on (3);
6:          replace all points by the barycenter in each sub-cell;
7:      else //subdivide C_q by L_s.
8:          L_s = L_s + 1;
9:          s' = s/(pow(2, Ls));
10:         SimplifyInnerCells(s');
11:     end if
12: end for
```

For each inner cell, we compute the curvature criterion (5). If it satisfies the threshold, we first subdivide this cell based on (3); then, replace all points in each sub-cell by their barycenter. Otherwise, we subdivide this cell and repeat the process until all conditions of the criterion are satisfied.
In this step, our computing experiences have seen that the processing time mostly depends on values of $\partial$; before and after combining with step one (rough simplification) (see table 2), and less depends on $s$ (size of a cell). Normally, the number of points in a cluster (using PCA) is around from 30 points [Carsten04, RenFang08, Morales10]. In our case, the curvature within a cell of a geologic surface is low and the 3D points are sparse. Therefore, we choose $s \leq 10$ (that is initial cells containing at most 100 voxels) and many values of $\partial$ to implement. As a result, the time is affected if the number of points in a cell greater than 36 or $\partial$ close to 0 and before combining with step one. We keep the boundary and combine two steps (rough and elaborate) to simplify a surface; thus, the surface is simplified com-

 ISBN 978-80-86943-75-6

pletely, the initial shapes of the surface are preserved, and the time is controlled.

## 4 RESULTS

In this section, we present some of our results. For the step one (rough simplification), the computation are very fast. The algorithm has been tested on many surfaces with different number of points to compare the running time and simplification rate with an existing method (cluster vertices) [Pauly02]. The results are presented in table 1, the running time of our method is faster than the clustering method, while the simplification rate is slightly lower (depending on the initial shapes of input surface) because we kept the boundary points.

| Input points | Our method | | Cluster method | |
|---|---|---|---|---|
| | Output points/s.rate | time (ms) | Output points/s.rate | time (ms) |
| 32402 | 1881/94% | 36 | 1075/96% | 303 |
| 68956 | 3695/95% | 53 | 2432/96% | 544 |
| 148317 | 6368/96% | 98 | 4675/97% | 1149 |
| 346796 | 13030/96% | 206 | 11068/97% | 2766 |
| 664582 | 22388/96.6% | 377 | 19872/97% | 5739 |
| 1006712 | 67360/93% | 651 | 28850/97% | 8501 |

Table 1: The comparison between our method (rough simplification) and clustering method. We use the same size of a neighboring distance between the points, and run on the same computer (s.rate: simplification rate; ms: milisecond).

In this step, the simplification rate is controlled by the cell size. In our method, although boundary points are kept to preserve a part of the shape of surface, this approach does not take into account the curvature of the surface and hence is too rough to be applied with high simplification rates. If we use a larger size of cells to simplify, the received results are not accurate (see figure 7). Therefore, this step can only be applied to simplify a simple surface of 3D points or to adjust the resolution of a 3D point cloud by using a small size of cell. In the clustering method, all points of the surface (boundary points and inner points) are simplified; the shape of the output surface is not well preserved (see figure 8).
In step two (elaborate simplification), we have tested our approach on different surfaces with different numbers of 3D points and different values for $\partial$. The results are detailed in table 2. We provide the values of $\partial$ in order to show that: if the value of $\partial$ is close to 0, the obtained surface is smooth, close to the initial surface (small simplification rate) and the processing time is low; otherwise, if the value of $\partial$ is close to 1/3, the obtained surface is far from the original one (higher simplification rate) and the running time is higher. However, we have maintained boundary points, and constrained the point distribution from the boundary to the inside of the surface. Therefore, we have obtained the output surfaces preserving the initial geometry of the surface (see figure 10). Figure 9 shows the result of the point distribution constrained from the boundary to the inside of the surface. As a result, a good triangular surface can be obtained in a further meshing step.

| P.input (kb) | Values of $\partial$ | Time1 (ms) | Time2 (ms) | P.output (s.rate) |
|---|---|---|---|---|
| 60511 (976) | $\partial \leq 0.03$ | 5271 | 3231 | 9879/84% |
| | $\partial \leq 0.12$ | 5026 | 2958 | 9377/84.5% |
| | $\partial \leq 0.20$ | 3776 | 2910 | 6786/89% |
| 148317 (2461) | $\partial \leq 0.03$ | 22106 | 14194 | 21122/86% |
| | $\partial \leq 0.12$ | 21167 | 13825 | 20820/86% |
| | $\partial \leq 0.20$ | 15896 | 12079 | 18916/87% |
| 346796 (5727) | $\partial \leq 0.03$ | 114795 | 111362 | 56448/84% |
| | $\partial \leq 0.12$ | 111289 | 107309 | 52187/85% |
| | $\partial \leq 0.20$ | 110623 | 101544 | 50112/86% |
| 866639 (14500) | $\partial \leq 0.03$ | 832865 | 191153 | 147328/83% |
| | $\partial \leq 0.12$ | 786980 | 185491 | 138622/84% |
| | $\partial \leq 0.20$ | 581159 | 166116 | 112633/86% |

Table 2: The running time of step two before (Time1) and after (Time2) combining with step one; the simplification rate (s.rate) after using the same size of cells; different values of $\partial$ (kb: kilobyte; ms: millisecond).

## 5 EVALUATION

Our method has two advantages compared to existing methods. First, we use a cell to gather and compute the points in a local neighborhood to simplify the surface. By using a cell, there are no outside points between the cells; only one loop is used to consider all points of the surface. On the contrary, the other methods [Pauly02, Zhang10, Morales10] use a sphere or a cylinder (both are the same) to compute the neighboring points within a threshold value of a radius $r$ (see figure 6). Therefore, after each iterative operation, they have to process the points outside of these sphere/cylinder. The second advantage is that searching to compute a neighboring point within a cell is faster than within a sphere [Matthew96]. Our approach also takes advantage of the fact that our data are already organized in a sparse numeric volume, and hence we don't need to lose time and memory space to build accelerating data structure for k_neighbors computation (such as kd-trees or octrees).

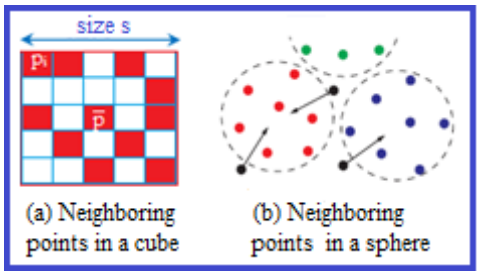


Figure 6: Determining of a neighboring point.

 ISBN 978-80-86943-75-6

## 6 CONCLUSION

In this paper, we have presented a method to simplify an elevation surface defined by a 3D point cloud. It is a part of our research in the field of geometric modeling of oil reservoir. The input data are a mass of 3D point clouds, and the number of points can reach millions of points. Therefore, our first approach focuses on data processing and surface simplification. Successively, we succeed in boundary extraction and simplification of the surface, while preserving the original shape of the surface as expected [Sinh12]. The surface simplification of 3D point clouds using PCA can normally yield an expensive computation. In our case, the input data are stored in the 3D grid volume, implicitely containing the neighborhood information for each point. We have taken this advantage; combined two steps for rough and elaborate simplification; and two ways of subdivision by using a cell to grow and simplify the surface. The output surface preserves the initial shape of the input surface, the point density and the point distribution are kept regularly, constrained from the boundary to the inside of surface. This good distribution of points is an advantage to obtain a good triangulation of the point clouds. Obtaining this triangulation by a fast method corresponds to our forthcoming work.

## 7 ACKNOWLEDGMENTS

The work reported in this article is a part of a PhD thesis, in which the finance is supported by the cooperation between Vietnam's government (MOET) and France's government (Campus France). We would like to thank all their valuable helps. We would also thank the reviewers for their valuable comments.

## 8 REFERENCES

[Sinh12] Van-Sinh NGUYEN, Alexandra BAC, Marc DANIEL, "Boundary Extraction and Simplification of a Surface Defined by a Sparse 3D Volume", *Proceeding of the third international symposium on information and communication technology SoICT 2012, Pages. 115-124, ACM-ISBN: 978-1-4503-1232-5*, August 23-24, Vietnam, 2012.

[Van06] Nam-Van TRAN, "Traitement de surfaces triangulées pour la construction des modèles geologique structuraux", *PhD Thesis*, Université de la Méditerranée, 2008.

[Garland99] Michael Garland, "Quadric-Based Polygonal Surface Simplification", *PhD Thesis*, Carnegie Mellon University, 1999.

[Philippe09] Philippe Verney, "Interprétation géologique de données sismiques par une méthode supervisée basée sur la vision cognitive", *PhD Thesis*, École Nationale Supérieure des Mines de Paris, 2009.

[Carsten04] Carsten Moenning, Neil A. Dodgson, "Intrinsic point cloud simplification", *International Conference Graphicon '14*, Moscow, Russia, 2004.

[RenFang08] Ren-fang WANG, Wen-zhi CHEN, San-yuan ZHANG, Yin ZHANG, Xiu-zi YE, "Similarity-based denoising of point-sampled surfaces", *Journal of Zhejiang University SCIENCE A, Volume. 9, Number. 6, Pages. 807-815*, 2008.

[Alexandra07] A.Bac, V.Tran Nam, M.Daniel, "A hybrid simplification algorithm for trianglar mesh", *Graphic Conference 2007, Pages. 17-24*, Moscow.

[Alexa01] M.Alexa, J.Behr, D.Cohen-Or, S.Fleishman, D.Levin, C.T.Silva, "Point Set Surfaces", *Proceedings of the conference on Visualization '01*, San Diego, CA, USA - October 2001.

[Cignoni98] P.Cignoni, C.Rocchini, R.Scopigno, "Metro: Measuring error on simplified surfaces", *The Eurographics Association 1998, Volume. 17, Number. 2*, June 1998.

[Matthew96] Matthew T. Dickerson, David Eppstein, "Algorithms for proximity problems in higher dimensions", *Journal Computational Geometry, Theory and ApplicationsPages, Volume. 5, Pages. 277-291*, 1996.

[Pauly02] M.Pauly, M.Gross, L.P.Kobbelt, "Efficient Simplification of Point-Sampled Surfaces", *Visualization, 2002. VIS IEEE, ISBN: 0-7803-7498-3, Pages. 163 - 170*, Boston, MA, USA, 2002.

[Moenning03] Carsten Moenning and Neil A. Dodgson, "A New Point Cloud Simplification Algorithm", *In Proceedings 3rd IASTED Conference on Visualization, Imaging and Image Processing, Pages. 1027-1033*, Spain, 8-10 Sep 2003.

[Mederos04] B.Mederos, L.Velho, L.H.Figueiredo, "Smooth Surface Reconstruction from Noisy Clouds"*Journal of the Brazilian Computer Society, Volume. 9, Number. 3, Pages. 52-66, ISSN: 0104-6500*, Campinas Brasil, Apr. 2004.

[Zhang10] Y.J.Zhang, L.L.Ge, "A Robust and Efficient Method for Direct Projection on Point-sampled Surface", *International Journal of Precision Engineering and Manufacturing, Volume. 11, Number. 1, Pages. 145-155, DOI: 10.1007/s12541-010-0018-z*, 2010.

[Morales10] R.Morales, Y.Wang, Z.Zhang, "Unstructured Point Cloud Surface Denoising and Decimation Using Distance RBF K-rearest Neighbor Kernel", *Proceedings of the Advances in multimedia information processing, ISBN: 3-642-15695-9 978-3-642-15695-3*, China, 2010.

[HaoSong09] Hao Song, Hsi-Yung Feng, "A progressive point cloud simplification algorithm with


Figure 7: Rough simplification: the shape of the initial surface is not preserved and received results are not accurate using a large cell-size (c).

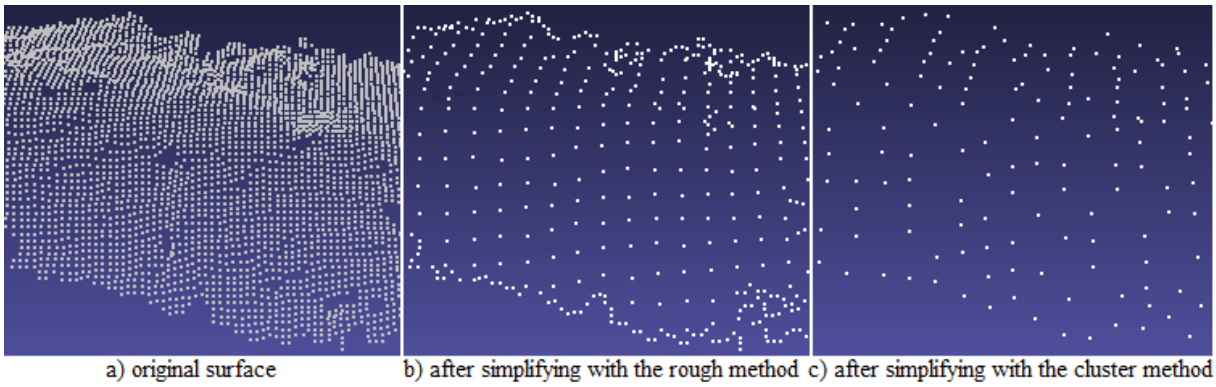


Figure 8: Shape comparison (computing the approximate error) by using the rough simplification method (Max: 0.014235, Mean: 0.000486) and the cluster method (Max: 0.029596, Mean: 0.000994), with the same size of neighboring distance.

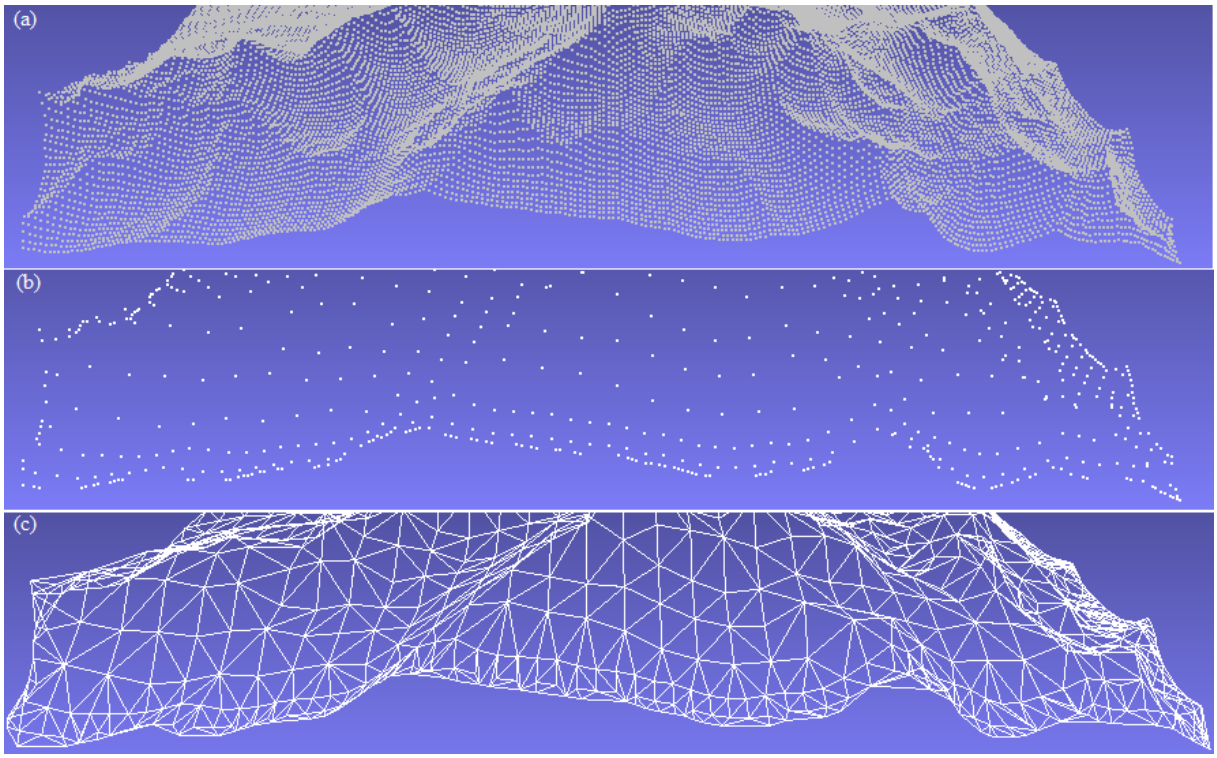


Figure 9: a) An elevation surface of 3D data points (66049 points); b) After simplifying by using the elaborate method (cell-size=8, $\partial \leq 0.09$, remaining points: 1840), the point distribution are constrained from the boundary to the inside of the surface; c) A good triangular surface can be obtained in a further meshing step (the approximate error between (a) and (c) is Max: 0.061598; Mean: 0.035884)

 ISBN 978-80-86943-75-6

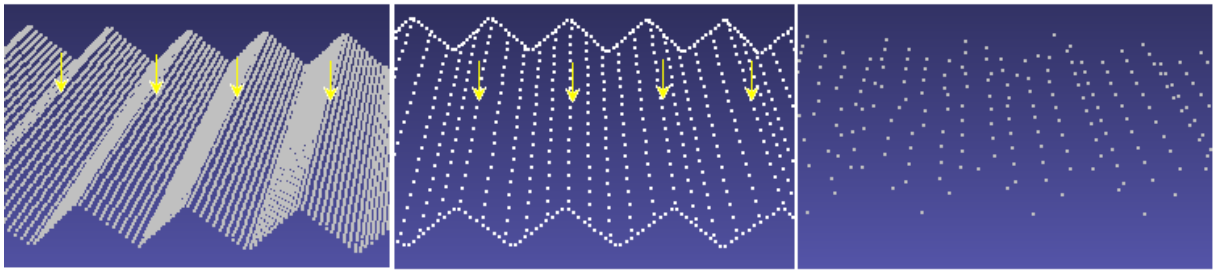


Figure 10: Comparison the shape of the surface between two methods by using the same size of neighboring distance. (a) An input surface of 23559 points; (b) After using elaborate simplification method (time: 858 msec, remaining points: 2305), the curvature (sharp lines: yellow arrows) of the surface is maintained; (c) After using clustering method (time: 255 msec, remaining points: 801), the curvature of the surface is not preserved.

preserved sharp edge data", *Technology - INT J ADV MANUF TECHNOL, Volume. 45, Number. 5, Pages. 583-592,*, 2009.

[Hoppe92] Hugues Hoppe, Tony DeRose, Tom Duchampy, John McDonaldz, Werner Stuetzlez, "Surface reconstruction from unorganized points", *Proceeding SIGGRAPH '92 Proceedings of the 19th annual conference on Computer graphics and interactive techniques, Pages, 71-78, Volume 26 Issue 2*, USA, 1992.

[Yoo09] D.J.Yoo, H.H.Kwon, "Shape Reconstruction, Shape Manipulation, and Direct Generation of Input Data from Point Clouds for Rapid Prototyping", *International journal of precision engineering and manufacturing, ISSN: 2005-4602, Volume. 10, Number. 1, Pages. 103-113*, 2009.

[Frey07] B.J. Frey, D.Dueck, "Clustering by Passing Messages Between Data Points", *Volume. 315, Number. 5814, Pages. 972-976*, 2007.

[Mario09] Mario Richtsfeld, Markus Vincze, "Point Cloud Segmentation Based on Radial Reflection", *CAIP '09 Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns, ISBN: 978-3-642-03766-5, Volume. 5702, Pages. 955-962*, Berlin, 2009.

[Mederos03] B.Mederos, L.Velho, L.H.Figueiredo, "Robust Smoothing of Noisy Point Clouds", *Conference on Geometric Design and Computing, ACM Trans on Graphics 22, Pages. 4-32*, 2003.

[MZhang11] M.Zhang, N.Anwer, L.Mathieu, H.B.Zhao, "A Discrete Geometry Framework for Geometrical Product Specifications", *CIRP Design Conference, Pages. 142-148*, South Korea, March 2011.

[Franc01] M.Franc, V.Skala, "Triangular Mesh Decimation in Parallel Environment", *EUROGRAPHICS Workshop on Computer Graphics and Visualization, Pages. 39-52, ISBN: 84-8458-025-3*, Girona, Spain, 2001.

[Zhe07] Ying-Zhe Lue, Yi-Hsing Tseng, "Surface Reconstruction from LiDAR Point Cloud Data with a Surface Growing Algorithm", *Proceedings of the 28$^{th}$ Asia Conference on Remote Sensing, Kuala Lumpur, Malaysia*, 2007.

[Tamal11] Tamal.K.D, Ramsay.D, L.Wang, "Localized Cocone surface reconstruction", *Computers Graphics, Volume. 35, Issue. 3, Pages. 483-491, Shape Modeling International (SMI)*, 2011.

[Jae05] Jae-Young.S, Sang-Uk.L, Chang-Su.K, "Construction of Regular 3D Point Clouds Using Octree Partitioning and Resampling", *Circuits and Systems. ISCAS 2005. IEEE International Symposium, Volume. 2, Pages. 956 - 959*, 2005.

[Xiaohui07] Xiaohui Du, Baocai Yin, Dehui Kong, "Adaptive out-of-core simplification of large point clouds", *Multimedia and Expo, 2007 IEEE International Conference, Pages. 1439-1442, Print ISBN: 1-4244-1016-9*, Beijing July 2007.

[David08] David Belton, "Improving and Extending The Information on Principal Component Analysis for Local Neighborhoods in 3D Point Clouds", *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume. XXXVII, Part. B5: 477 ff*, Beijing 2008.

[Yu06] Zhiwen Yu, Hau-san Wong, "An efficient local clustering approach for simplification of 3D point-based computer graphics models", *IEEE International Conference on Multimedia and Expo, ISBN: 1-4244-0366-7*, Toronto, Canada 2006.