*I.J. Image, Graphics and Signal Processing,* **2014, 2, 14-22**
Published Online January 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijigsp.2014.02.02

# Modified Streaming Format for Direct Access Triangular Data Structures

Khaled Abid
Ibn khaldoun University,
Tiaret, Algeria
E-mail: k_abid@esi.dz

Abdelkrim Mebarki
Université des sciences et de technologie d'Oran – Mohamed Boudiaf,
Oran, Algeria
E-mail: amebarki@visiondz.info

Wahid Hidouci [3]
Ecole nationale supérieure d'informatique,
Alger, Algeria
E-mail: w_hidouci@esi.dz

***Abstract*** — We define in this paper an extended solution to improve an Out-of-Core data structure which is the streaming format, by adding new information allowing to reduce file access cost, reducing the neighborhood access delay to constant time.

The original streaming format is conceived to manipulate huge triangular meshes. It assumes that the whole mesh cannot be loaded entirely into the main memory. That's why the authors did not include the neighborhood in the file structure.

However, almost all of the applications need the neighborhood information in the triangular structures. Using the original streaming format does not allow us to extract the neighborhood information easily. By adding the neighbor indices to the file in the same way as the original format, we can benefit from the streaming format, and at the same time, guarantee a constant time access to the neighborhood.

We have adapted our new structure so that it can allow us to apply our direct access algorithm to different parts of the structure without having to go through the entire file.

***Index Terms —*** Triangular data structure, Streaming format, Direct access structure

## I. INTRODUCTION

With the recent development in computer graphics and 3D vision, new methods were born for geometrical modeling.

The common principle between these methods consists in modeling an object as a set of geometrical primitives such as points, positioned in Euclidean three-dimensional space.

These structures -also known as geometrical structures- are the object of our study. The size of the triangulations handled in the various applications also do not cease increasing, so that their processing in real-time starts to really pose a problem with the standard machines, and even the structures of data called in-core, do not allow the processing of these volumes. In this article we propose a modification of the streaming format conceived to this end, including improvements which allow the access to neighborhood, and direct access to different parts of the file in real-time.

### *A. In-core Data Structures*

Several data structures were developed for the storage of geometrical information, provided with access and update methods (creation, modification) [13]. The basic idea is very simple: A triangulation or any other arrangement is generally represented in memory using two tables (often contiguous), one for the vertices, and the other for the topological simplices (vertices, edges, and faces).

Programmers aim often to conceive data structures which are efficient and simple [17], moreover, it must respect the requirements of the desired applications [2]. The 2D data structures in the literature use the three basic simplices which are edges, triangles, and vertices. In the edge-based data structures, we define several modes of representations whose basic object is either the edge, or the half-edge [1], [3], [2], [18], [10], [19], [14], [4], [24]. These structures can be used not only for the representation of the triangulations, but also for any polygonal model.

In the triangle-based data structures [6], [5], [16], the basic element is the triangle. The basic structure uses two tables:

- A table for vertices: to store geometric coordinates.
- A table for triangles to store references of the three vertices that define the triangle.

 

This minimal triangle-based structure requires 6n references. However, access to neighbors cannot be done in constant time. To improve the structure, and give access in constant time to the neighbors, the references of the three neighboring triangles can be added to the structure of the triangle, which increases the cost of storage to 12n references.

This representation scheme is the basis of our work. The algorithm proposed by Isenburg to construct the streaming file uses another data structure which is the triangle soup. This is because this structure allows the processing of the triangulation without charging the entire file in the main memory but it does not allow the addition of neighborhood information to the streaming file.

In the vertex-based data structures [12], the basic element is the vertex. The triangulation is represented as a graph of incidences between the vertices of the triangulation. The data structure is a list of vertices where each vertex keeps three references: its degree (the number of incident vertices), the list of its neighbors, and a mark indicating if this vertex is on the border or not. Considering that the average degree of a vertex in a triangulation of n points is 6, the global cost of such a structure is 7n references.

### *B. Out-Of-Core Data Structures*

The fast development of storage capacities led to a strong increase in the data volumes, in particular the geometrical data, where the meshes generated from digitalization by laser or modeling of complex scenes, can reach several gigabytes. This is not the case for external memory where the curve of growth is not accompanied by a proportional improvement in access time and capacity of central memory.

Unfortunately, traditional methods and algorithms (In-Core) require the entire mesh to be loaded in main memory. With huge size meshes (order of billion of triangles), indexing vertices is very costly or even impossible when it exceeds the addressable range of the computer main memory. The Out-of-Core algorithms and structures allow us to process 2D or 3D objects of any size on standard workstations.

The main idea of these algorithms is to arrange operations in such way to execute them on only a portion of the file. This file can be seen as a set of blocks, each one of them has a size that matches the main memory size. In other words, only the needed information for the current treatment is loaded into memory.

1. Triangle soup:

The basic principle is to include directly the coordinates of the vertices in the faces without enumeration. This allows us to treat the faces of a mesh independently, and avoid the step of indirection. But the updating of vertices and the access to its neighborhood is not as easy as in the indexed format. In this format each triangle is defined by its three vertices, each vertex by these coordinates. Although this format stores each vertex of the mesh several times (in average six times), it is still preferred in many applications such as rapid prototyping because no global indexing is required [14], [15], [24].

2. The hierarchical data structures (multi-resolution):

These structures represent triangulations according to spatial or topological subdivisions. These structures are also used to adjust the level of details [8] and for interactive visualizations. The basic common structures are trees (including B-trees [22] and its derivatives [23]). Cignoni et al.[7] proposed an adapted a version of the octree [20], [21] devoted to the generic Out-of-Core algorithms for meshes that they called *Octree Based External Memory Mesh*. This structure is based on a decomposition of the cube including the triangulation recursively until the desired size of the elementary cubes (in number of vertices per cube, which are indexed locally in the cube) is reached. These elementary cubes are the leaves of the tree, and are stored on the external drive and loaded into memory on demand. A variant of the octree representation has been adapted to the multi-resolution meshes [9], in which the internal nodes of the tree keep representatives scattered of the lower level vertices to allow construction at this coarse level. In these structures, the permanent memory occupancy is limited to the tree of search blocks, whereas the actual data are loaded as needed.

## II. STREAMING MESH

Isenburg et al. [11] proposed a new data structure they called" Streaming Mesh". This structure is ordered and designed to encode and describe incrementally the triangulations in order to minimize memory requirements, which allows the user to manage models of very large dimensions. The principle is that the streaming format explicitly indicates the first and the last time that a vertex is used by a triangle. Vertices and faces are so interlaced in the file; a vertex is inserted only when it is used for the first time; when a triangle refers a vertex for the last time, the vertex is finalized (e.g. via special symbol). Thus, the number of vertices to be kept in memory at any given time is reduced.

***Generating Streaming Meshes***

Our method is based on an indexed structure in input, which is the triangle-based data structure with two tables (Tables I and II):

TABLE I. VERTEX TABLE

| Vertex | X | Y |
|---|---|---|
| V1 | x1 | y1 |
| V2 | x2 | y2 |
| V3 | x3 | y3 |
| V4 | x4 | y4 |
| V5 | x5 | y5 |
| V6 | x6 | y6 |
| V7 | x7 | y7 |

 

TABLE II. INPUT TRIANGULATION TABLE: THE NUMBER MAX-INT IS THE MAXIMAL NATURAL INTERGER

| T | V1 | V2 | V3 | T1 | T2 | T3 |
|---|---|---|---|---|---|---|
| T1 | 2 | 3 | 6 | 2 | Max-Int | 4 |
| T2 | 3 | 6 | 7 | Max-Int | 3 | 1 |
| T3 | 3 | 7 | 4 | Max-Int | 6 | 2 |
| T4 | 1 | 2 | 3 | 1 | 5 | Max-Int |
| T5 | 1 | 3 | 5 | 6 | Max-Int | 4 |
| T6 | 3 | 5 | 4 | Max-Int | 3 | 5 |

To construct a Streaming file we must follow some steps in which we need three kinds of information: the vertex layout, the triangle layout, and the finalization information.

We add to each vertex V:

a. An ordered key Kv based on its incident triangles. The key Kv takes the smallest incident triangle index:

$$\{Kv = \min(v), v \in tt\},$$

This key is based on the indices of vertices and triangles in the two input arrays.

b. A degree d: the degree d is equal to the number of triangles incident to the vertex V:

$$d = |t : V \in t|$$

c. A Boolean value ∃ that indicates whether the vertex exists in the streaming file. This value is initially equal to False. Each vertex will be represented using (Kv, V, d, X, Y, ∃ ), Where *Kv* is the vertex key, *V* is the index, *d* is its degree and both *X* and I coordinates. Each triangle has as information (V1, V2, V3, T1, T2, T3, ∃ ), where: *V1*, *V2*, *V3* are the three vertices of the triangle; *T1*, *T2*, *T3* are the neighbors, ∃ is a Boolean value indicating whether the triangle is in the file streaming or not.

***Vertex finalization***

When a vertex is closed and referenced for the last time, it will be removed immediately from the memory and thus allows other vertices to use not only the memory but also the index of the closed vertex. We maintain for each active vertex a degree counter which is equal to its degree *d*, and each time a vertex is referenced by a triangle, the degree counter is decremented by one, and so on until it is equal to zero, in this case the vertex is closed (referenced for the last time) and removed from the main memory.

***Sequence of the algorithm***

1. Reading the triangulation: We use a procedure that reads the input file and fills two tables of vertices and triangles while adding to each table, the additional information mentioned above: For each vertex *V* : (The vertex coordinates *X* and *Y* , The vertex degree *d*, The key *Kv*, a Boolean value ∃ ). For each triangle *t*: (the three indices of its vertices *V1*, *V2*, *V3*, the three indices of its neighbors *T1*, *T2*, *T3* (if a vertex has no opposite triangle, it will be replaced by Max-integer), a Boolean value ∃ that indicates whether the triangle is inserted in the file streaming).

**Note:** The key *Kv* and the degree d are computed on-the-fly.

TABLE III.VERTEX TABLE AFTER READING

| Vertex | X | Y | *d* | $K_v$ | ∃ |
|---|---|---|---|---|---|
| V1 | x1 | y1 | 2 | 4 | F |
| V2 | x2 | y2 | 2 | 1 | F |
| V3 | x3 | y3 | 6 | 1 | F |
| V4 | x4 | y4 | 2 | 3 | F |
| V5 | x5 | y5 | 2 | 5 | F |
| V6 | x6 | y6 | 2 | 1 | F |
| V7 | x7 | y7 | 2 | 2 | F |

TABLE IV. TRIANGLE TABLE AFTER READING, THE NUMBER MAX-INT IS THEMAXIMAL NATURAL INTERGER =4294967295

| T | V1 | V2 | V3 | T1 | T2 | T3 | ∃ |
|---|---|---|---|---|---|---|---|
| T1 | 2 | 3 | 6 | 2 | Max-Int | 4 | F |
| T2 | 3 | 6 | 7 | Max-Int | 3 | 1 | F |
| T3 | 3 | 7 | 4 | Max-Int | 6 | 2 | F |
| T4 | 1 | 2 | 3 | 1 | 5 | Max-Int | F |
| T5 | 1 | 3 | 5 | 6 | Max-Int | 4 | F |
| T6 | 3 | 5 | 4 | Max-Int | 3 | 5 | F |

2. Spatial sorting: The easiest way to build a streaming file is to sort the elements along a spatial direction as the *X* and *Y* axis. We sort the vertices in lexicographic order, and then each vertex is re-indexed according to his order. In the case of our algorithm and after reading the input file, a lexicographic sorting is applied to the array elements of the vertices, from the smallest to the largest vertex according to their *X* and *Y* coordinates.
3. Updating triangle table: After sorting vertices and changing their indices, we update the vertices of the triangles in the tables V and VI. After sorting the vertex table in the previous example, the ordering of the vertices will be as follows: (2.6.1.3.7.5.4).

TABLE V. VERTEX TABLE AFTER SPATIAL SORTING

| Vertex | X | Y | *d* | $K_v$ | ∃ |
|---|---|---|---|---|---|
| V1 | x2 | y2 | 2 | 1 | F |
| V2 | x6 | y6 | 2 | 1 | F |
| V3 | x1 | y1 | 2 | 4 | F |
| V4 | x3 | y3 | 6 | 1 | F |
| V5 | x7 | y7 | 2 | 2 | F |
| V6 | x5 | y5 | 2 | 5 | F |
| V7 | x4 | y4 | 2 | 3 | F |

TABLE VI. TRIANGLE TABLE AFTER SPATIAL SORTING

| T | V1 | V2 | V3 | T1 | T2 | T3 | ∃ |
|---|---|---|---|---|---|---|---|
| T1 | 1 | 4 | 2 | 2 | Max-Int | 4 | F |
| T2 | 4 | 2 | 5 | Max-Int | 3 | 1 | F |
| T3 | 4 | 5 | 7 | Max-Int | 6 | 2 | F |
| T4 | 3 | 1 | 4 | 1 | 5 | Max-Int | F |
| T5 | 3 | 4 | 6 | 6 | Max-Int | 4 | F |
| T6 | 4 | 6 | 7 | Max-Int | 3 | 5 | F |

 

4. Complete streaming file: To build the streaming file, we take each time a vertex from the vertex table according to their orders and we apply the following operations:

***Selection and verification***

If the vertex does not exist in the streaming mesh:

1) We go to the smaller triangle that refers it using the key *Kv*.
2) We turn around the vertex by completing a list of incidents triangles on the vertex (only triangles that are not already inserted will be added to the list).
3) We sort the elements of the list from the smallest one to the largest one. If the vertex already exists: we move to the next vertex and resume the steps from the beginning.

***Interleaving vertices and triangles:***

For each triangle of the list:

1) We interleave vertices one by one starting with the smallest, if they are not interspersed. After each new vertex inserted in the streaming file, we must immediately interleave all the triangles that include the current vertex and the previous vertices.
2) Finally, we interleave the current triangle.

**Note:**

A vertex or a triangle must not be interleaved twice in the streaming mesh, here is a small example: While travelling through a streaming file, at a given moment, all active triangles and vertices can be seen as a triangle-based data structure with minimal representation.

The set of vertices of this structure is called the front Fi (the width front or simply the width = maxi |Fi| of the front, i.e. the maximum number of active vertices at the same time). However, to access the neighbors of a given triangle we must traverse all active triangles, which will take a long time to process the active part of our streaming mesh, especially if the width front is very large, and therefore the file processing time will be too long. To solve this problem we add neighborhood information to make access to neighbors in constant time. We have two ways to add the notion of neighborhood to the new structure:

1. Starting from the same principle of the triangle-based structure, using the notion of the neighbor triangle: with each triangle interleaved in the streaming file we add beside three fields, each field represents the neighbor triangle opposite to the vertex located in the same order. Otherwise (in border triangles), we put a special index e.g. max-integer (Figure2). For each triangle, we refer only to those neighbor triangles that are inserted before it in the streaming file. The disadvantage of this method is that:
   - The streaming file is larger.
   - There is useless information, such as special indices that tell us what a vertex has no opposite triangle is unnecessary and slows the course of the streaming file.
2. A second method consists of inserting after each triangle of the streaming file, its neighborhood information with the previous triangles (If a triangle has two neighbors that are already inserted in the file, we inserted neighborhood information for each incident triangle sequentially. otherwise we move to the next item that is a vertex or a triangle). So, to build our streaming file we interleave vertices, triangles, and neighborhood information in a single file (Figure2). This will allow us, at a given moment, to know if a triangle has neighbors or not, we go directly to the neighborhood information that comes after. This second method not only enables us to reduce the size of the file by eliminating useless information, but also to facilitate the reading of the file on the one hand and to accelerate the course of another share.

***Triangles finalization***

The problem with both methods is still the indexing of triangles which may exceed the addressable range in the main memory when browsing the streaming file. To solve this problem we use the finalization to allow multiple triangles to use the same index. A triangle is finalized and referenced for the last time if all its neighboring triangles were interleaved.

To finalize a triangle t we use its degree which is equal to |t'|: t' neighbor of t.

A triangle can have at most three incident triangles. This degree will be used as a degree counter that is decremented each time the triangle is referenced by the neighborhood information. The streaming file will be as follows.Figure2:

## III. DIRECT ACESS ON A STREAM FILE

The use of neighborhood information reduces on a remarkable way the time required to traverse the neighborhood in the streaming file. However, to access the elements located in the middle of a triangulation we must always start with the first point of streaming and pass through all the elements of the triangulation, according to the sequential order of the file to the target. This is in large part due to the sequential structure on one hand, and because of the occupation of the same index by several vertices and faces in the other hand. We have reduced access time to the elements (vertices, faces or parts of the triangulation) to any position in the triangulation. The proposed method defines the elements involved in the reading part of the triangulation using zigzag[1] readings (see figure 3), to collect the missing components (vertices, faces, and neighborhood information).

To do this, we add an indirection stage at the streaming file, which directly allows to determine the indices of vertices and faces without going through the elements that dereference its. The index values are the same used to reference the elements (faces, vertices) when these elements pass into the current stream of memory.

[1] do (readings) trips / Returns to search for intercalated vertices and faces involved in the definition of the concerned part.

 

We envisaged three possible methods to access directly to a particular subdivision of a triangulation:

1. We designate *Si*, the vertex with the coordinates (*xi*; *yi*): we look for the vertex in question, and then we made trips/returns to find the indices of vertices which make up the faces located after the vertex *Si*.
2. A second method is to directly load the span (the maximum number of active vertices at the same time) witch precede the vertex Si, it allows us to avoid to trips/returns like as the previous method.
3. The third alternative is to change the streaming format by inserting with each vertex and each face, the corresponding index in the streaming file. So we have the index directly without having to read the file from the beginning.

**Note:** The first two methods suffer from two drawbacks: the time of traversing, especially in the first method with several trips/return, and the memory space required will go up to the double span space.

### *C. Indexing structure*

We modify our structure to be semi-indexed[1]. We added an indirection to vertices and faces of our file, which will enable us to know directly the indices of the vertices (faces) without making trips/returns to the faces (neighborhood information).

To apply this indirection to the file structure, we will further modify the construction scheme. For this we must:

- add before the coordinates of each new vertex intercalated in the streaming file, the index to which it is referenced by the first face that follows;
- add before the vertices of each new face intercalated in the streaming file, the index to which it is referenced by the first neighborhood information that follows.

### *D. Direct access to the triangulation*

To build a window with size Size_front[2] in the streaming file we chose to position the reading head at a line L witch represents the center line of the window. Then we decode the ”L-Size_front” and ”L+Size_front” (see Figure 3) to construct the two tables of vertices and faces containing the explicit representation of this part of the triangulation.

$$\texttt{Size\_front} = |F_L|^{3}.$$

***Position the reading head***

The reading head can be pointed at three different lines, which leads us to treat each of these three cases:

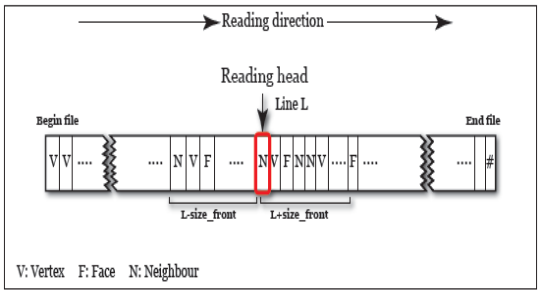


Figure 3. Position of the reding head on line streaming file: the length of Size_front depends on the forehead of the streaming file (i.e. loaded into memory part) in this point.

1. The reading head is on a vertex line of the form:

   *V i x y*, where:

- *V* means that the line is a vertex;
- *i* is the index of the vertex in the streaming file;
- *x* and y are the coordinates of the vertex *i*.

2. The reading head is positioned on a face line of the form:

   *f i V0 V1 V2,* where:

- *f* means that this line is a face;
- *i* is the index of this face in the streaming file;
- *V0*, *V1* and *V2*: are the indices of the vertices that make up the face *i*.

3. The reading head is positioned on a line of the form:

   *N F1 F2, where:*

- *N*: Indicates that this line represents a neighborhood information;
- *F1* and *F2*: represent the indices of the two neighboring faces.

### *E. Search of the elements of the triangulation*

Each line requires special processing to extract the relevant information that can be a vertex, a face or a neighborhood between two faces:

1. First case: This case corresponds to the introduction of a new vertex on the current flow, and is of the form: *V i x y*. All we needs to do in this case is to extract the coordinates x and y, and assign its directly to the vertex *Vi* in the vertex table.
2. Second case: In this case, the line is a declaration of a new face of the form *fi V0 V1 V2*. This means that the face *fi* is constructed by three vertices *V0*, *V1* and *V2*. Given the sequential nature of the streaming format, the three vertices are by default inserted before the face *fi*.

   To dereference the three vertices of the face *fi*:

   a. We check for each vertex in the vertex table. If some vertices are not yet included, we pass to the next step.
   b. If a vertex is not yet declared in the vertex table, it may be that there is a previously closed vertex on the streaming file (referenced for the last time) with the same index.
      i. We read the file in reverse order from the *L*.

[1] This refers to the structure in streaming format, but with the index of vertices and faces in moments of their passage in the active flow, which means a provisional indexing when passing in RAM.

[2] Size_front depends on the location where we want to open the window on the file

[3] $F_L$ is the front which corresponds to the center line L

 

ii. We support only lines that begin with the letter ”*V*” by comparing the index found *i* with the three vertices *V0*, *V1* and *V2*.

iii. Since we retrieve a vertex, we add its coordinates to the vertex table and did the same search for other vertices, until the three vertices are reported in the vertex table.

c. Then the face *fi* is filled in the table of faces by its three vertices *V0*, *V1* and *V2* (by updating the vertices indices following the vertex table and not the streaming file).

3. Third case: This is the case of a line declaring a neighborhood between two faces of the form:*N F1 F2*.

This means that the face *F2* is adjacent to the face *F1*. This third event brings both previous cases together because the neighborhood is called to two adjacent faces and therefore: the four vertices that compose them. The search of the vertices of both faces, if they have not yet been introduced into the vertex table follows the same detail scheme in the first two cases. The only improvement that can be made is due to the fact that the neighboring face is reconstructed by two vertices belonging to the first face, it remains to find only a single vertex (see Figure 4).

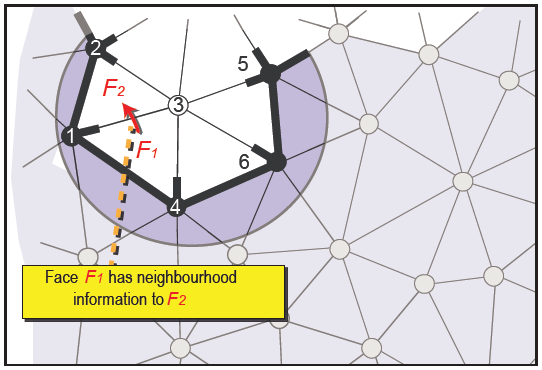


Figure 4. Opening a window in a streaming file: (1) Only the face F1 contains the neighborhood information to face F2 (2) research elements of the face F1 is limited to the search for vertex 4.

**Note:** The neighborhood between two faces in a streaming file takes always one direction (Figure4).
(If a face *F1* is neighborhood of *F2*) → (F2 is interlaced before F1).

### *F. Accessing multiple sliding windows*

In addition to its advantage of access to the elements of the triangulation, our method provides another opportunity for the users and applications. It can open multiple windows simultaneously (see Figure 5) on the surface of the triangulation.

The gain provided by the use of the direct access method in terms of time is considerable:

- The direct access algorithm reduces the cost to go to a point in the streaming file to Zero (instead of traversing the file from the beginning to the specific point).
- The multiplication of sliding windows divides the travel time of the entire file according to the number of windows:

Traversing time = Traversing time _ Number windows.

With these advantages, we gain a more flexible structure (easy to navigate), more rich representation (with more information) on one hand. We obtain also a faster scheme to traverse the data by supplying the original structure on other hand.

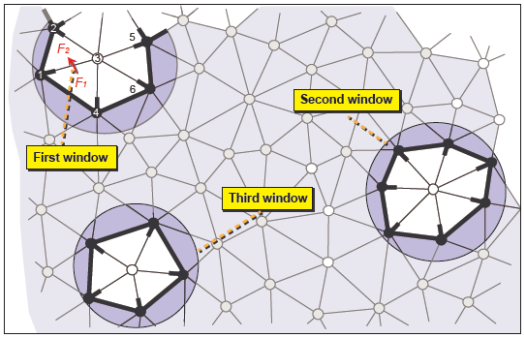


Figure 5. Opening of three slippery windows on the surface of a streaming file at the same time.

## IV. RESULTS

In this section we evaluate the efficiency of our approach with several triangulations. However, most of the geometric models are not available. For that purpose, we made our evaluation tests on Delaunay triangulations generated randomly.

Indeed, the problem we deal with regard to the handling of triangulations whose size exceeds the capacity of the main memory of standard computers, that is to say, we are interested in information exchange[5] between main memory and external memory, and not the use of information exchanged. In other words, when the user wants to load a part, it receives the response (if it exists) regardless of what he will do with afterwards.

Given these considerations, we have adopted a method of achieving consists of representing each triangulation in two tables (vertex table and faces face), as long as the other functions (visualization, simplification, compression, etc) didn’t interest us in the context of this study. As for new streaming files structures, they are checked through inverse algorithms[6].

Our results are obtained on a 64 bit machine, Processor Pentium (R) Dual-Core T4500@2.30 GHz with 6 GB of RAM. The table VII represents the different results obtained by traversing two types of streaming files of different sizes.

[5] In the form of vertices and faces.

[6] algorithms that generate the indexed triangular data structures from streaming files.

 

The tests are applied to triangulations (from 100,000 to 700,000 points). For all streaming files, we can see that there's not a big difference for simple reading between both structures: our structure is about 60% by late contributions to the reading of the original structure. On the other hand, if we take into consideration the traversing in neighboring of the parts charged in memory for the original streaming format, we see that the time required for this operation through the entire file is very large compared to our structure. And even the representation (under a graphical format or through a table) of the gain provided by our method cannot be exact because of the exponential explosion of the time of the traversing neighborhood in the original structure.

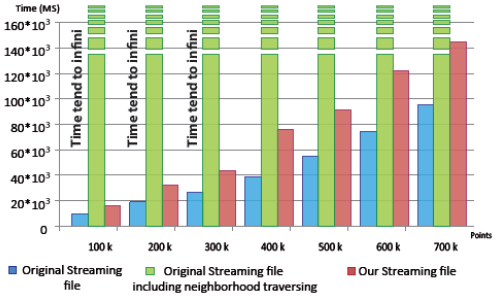


Figure 6. Graphical representation of reading time of the both streaming file structures. Blue bars represent the reading time of the original structure. Green bars represent the reading time of the basic structure including the neighborhood traversing. The red rectangles represent the reading time of our new structure including neighborhood information.

As far as the occupied memory space by our structure, the flowchart 7 shows that the size of the structure is equal to twice the size of the original structure for each triangulation. Since the streaming format is a data structure designed for the handling of triangulations outside the main memory, then the file size is not a problem in itself, since it is supposed to load only the front of the structure in main memory.

For our second contribution which is the direct access to the streaming file, the original structure does not allow this issue. Table VIII represents the different results obtained by opening a windows of different sizes (100, 1,000 and 10,000 points) at the center of three streaming file with: 100,000, 200,000 and 300,000 points.

TABLE VII. NECESSARY TIME FOR TRAVERSING STREAMING FILES. THE COORDINATES ARE IN THE FLOAT FORM AND REPRESENTED IN 4 BYTES EACH. THE GAIN IN THE TABLE IS REPRESENTED COMPARED TO THE COST (TIME IN MS) OF READING STREAMING FILES.

| Number of points | 100K | 200K | 300K | 400K | 500K | 600K | 700K |
|---|---|---|---|---|---|---|---|
| Original Streaming | 9658 | 19878 | 26472 | 38668 | 54697 | 74478 | 95261 |
| Our structure | 15896 | 32644 | 43457 | 75707 | 91453 | 121784 | 144700 |

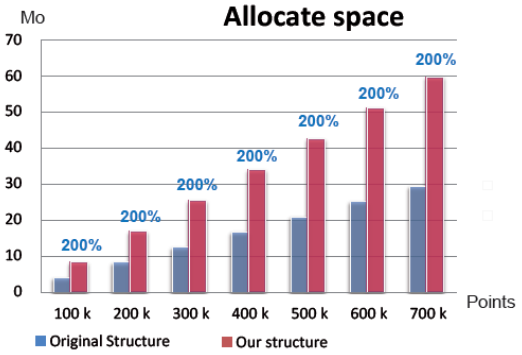


Figure 7. The memory space required to represent the two streaming file structures.

TABLE VIII. REPRESENTATION OF NECESSARY COSTS TO OPEN WINDOWS OF VARIOUS SIZES ON STREAMING FILES OF SEVERAL SIZES. THE TESTS CONSIST OF OPENING WINDOWS AT THE CENTER OF TRIANGULATIONS.

| Window | | 100k | 200k | 300k |
|---|---|---|---|---|
| 100 points | median Line | 294900 | 594600 | 894300 |
| | Cost (MS) | 2027 | 2640 | 1844 |
| 1000 points | median Line | 292200 | 591900 | 891600 |
| | Cost (MS) | 9670 | 11161 | 11242 |
| 10000 points | median Line | 265230 | 564930 | 864630 |
| | Cost (MS) | 10995 | 12605 | 11590 |

## V. CONCLUSION

We have proposed in this paper a new Out-of-Core solution improving a processing method which is the streaming format; this format allows us to treat the triangulation sequentially from the beginning with a capacity equal to the maximum length of active vertices. We have changed the streaming format by adding new information used to browse the part loaded into memory more quickly and make access to the neighborhood in constant time. Another contribution is the direct access, our format allows it by proposing a direct localization on the steam with a retrieving of the entire needed information.

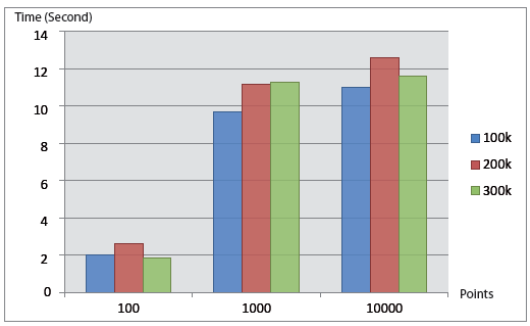


Figure 8. Graphical representation of necessary costs to open windows in the middle of streaming files. The blue rectangles represent the costs (time in milliseconds) for opening windows in streaming files with 100k points. The same case (for red and green rectangles) applies to files 200k and 300k points.

 

REFERENCES


[1] Bruce G. Baumgart. Winged edge polyhedron representation. Technical report, Stanford University, Stanford, CA, USA, 1972.

[2] Bruce G. Baumgart. Winged-edge polyhedron representation for computer vision. In National Computer Conference, May 1975.

[3] Bruce Guenther Baumgart. Geometric Modeling for Computer Vision. PhD thesis, Stanford University, USA, August 1974.

[4] Swen Campagna, Leif Kobbelt, and Hans-Peter Seidel. Directed edges a scalable representation for triangle meshes. Journal of Graphic Tools, 3(4):1–11, 1998.

[5] Luca Castelli Aleardi. Repr ésentations Compactes de Structures de Donn ées G éom étriques. PhD thesis, Ecole Polytechnique, Palaiseau, France, December 2006.

[6] Cgal: Computational geometry algorithms library. www.cgal.org.

[7] Paolo Cignoni, Claudio Montani, Claudio Rocchini, and Roberto Scopigno. External memory management and simplification of huge meshes. IEEE Transactions on Visualization and Computer Graphics, 9(4):525–537, 2003.

[8] Leila De Floriani, Leif Kobbelt, and Enrico Puppo. A survey on data structures for level-of-detail models. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization, pages 49–74. Springer, Berlin, Heidelberg, 2005.

[9] Michael Garland. A multiresolution representation for massive meshes. IEEE Transactions on Visualization and Computer Graphics, 11(2):139–148, 2005. Student Member-Eric Shaffer.

[10] Leo J. Guibas and Jorge Stolfi. Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. In STOC '83: Proceedings of the fifteenth annual ACM symposium on Theory of computing, pages 221–234, New York, NY, USA, 1983. ACM Press.

[11] Martin Isenburg and Peter Lindstrom. Streaming meshes. In Proceedings of the 16th IEEE Visualization Conference (VIS 2005), 23-28 October 2005, Minneapolis, MN, USA, page 30. IEEE Computer Society, 2005.

[12] Marcelo Kallmann and Daniel Thalmann. Star vertices: A compact representation for planar meshes with adjacency information. Journal of Graphics Tools, 6(1):7–18, 2001.

[13] Michael J. Laszlo. Computational Geometry and Computer Graphics in C++. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[14] P. Lienhardt. Subdivisions of n-dimensional spaces and n dimensional generalized maps. In SCG '89: Proceedings of the Fifth Annual Symposium on Computational geometry, pages 228–236, New York, NY, USA, 1989. ACM Press.

[15] Peter Lindstrom. Out-of-core simplification of large polygonal models. In SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pages 259–262, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[16] Abdelkrim Mebarki. Implantation de structures de donn ées compactes pour les triangulations. PhD thesis, Universit é de Nice-Sophia Antipolis, France, April 2008.

[17] Dinesh P. Mehta and Sartaj Sahni. Handbook Of Data Structures And Applications. Chapman & Hall/Crc Computer and Information Science. Chapman & Hall/CRC, 2004.

[18] David E. Muller and Franco P. Preparata. Finding the intersection of two convex polyhedra. Theor. Comput. Sci., 7:217–236, 1978.

[19] Franco P. Preparata and Michael I. Shamos. Computational geometry: an introduction. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[20] Hanan Samet. Applications of Spatial Data Structures: Computer Graphics, Image processing, and GIS. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[21] Hanan Samet. The Design and Analysis of Spatial Data Structures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[22] Robert Sedgewick. Algorithms. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[23] C. Silva, Y. Chiang, J. El-Sana, and P. Lindstrom. Out-of-core algorithms for scientific visualization and computer graphics. In IEEE Visualization conference'02. Boston, Massachusets. Course Notes, October, 2002.

[24] Jianhua Wu and Leif Kobbelt. A stream algorithm for the decimation of massive meshes. In Graphics Interface, pages 185–192. CIPS, Canadian Human-Computer Commnication Society, A K Peters, June 2003. ISBN 1-56881-207-8, ISSN 0713-5424.



**Khaled Abid** was born in Oran, Algeria. He is now a PhD student at "Tiaret, Ibn khaldoun University". He has a degree of "Ingénieur d'Etat" from "Université d'Oran", Algeria.

K. Abid is now preparing his PhD thesis, on External Memory Triangular Data Structures.

**Abdelkrim Mebarki** was born in Oran, Algeria. He is now an assistant professor at "Universit é des sciences et de la technologie d'Oran – Mohamed Boudiaf". A. Mebarki has his PhD degree from the University of Nice-Sophia Antiplis, France, in 2008. He has a Master degree from the same university, and the degree of "Ingénieur d'Etat" from "Universit é des sciences et de la technologie d'Oran – Mohamed Boudiaf".

A. Mebarki is now assistant researcher, in SIMPA laboratory, working in "Image-Vision" team. His research

works include Scientific Visualization, NPR image processing, and Triangular Data Structures.

**Walid Hidouci** is a professor at " Ecole nationale supérieure d'informatique", Algeria. K. Hidouci is the author of many publications and communications dealing databases and networks.

He has a PhD and a master degree from " Ecole nationale supérieure d'informatique", the degree of "Ingenieur d'Etat" from "USTHB, Algiers".

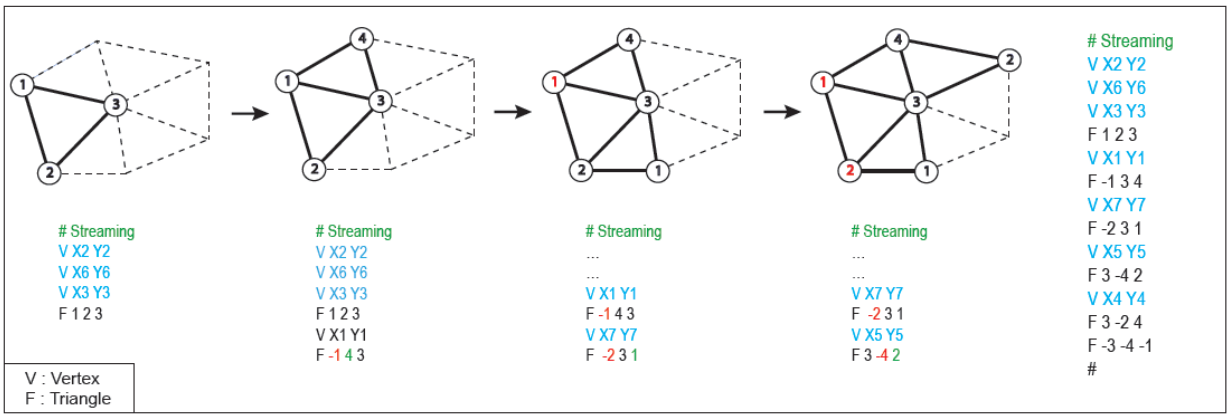


Figure 1. The construction steps of the streaming file

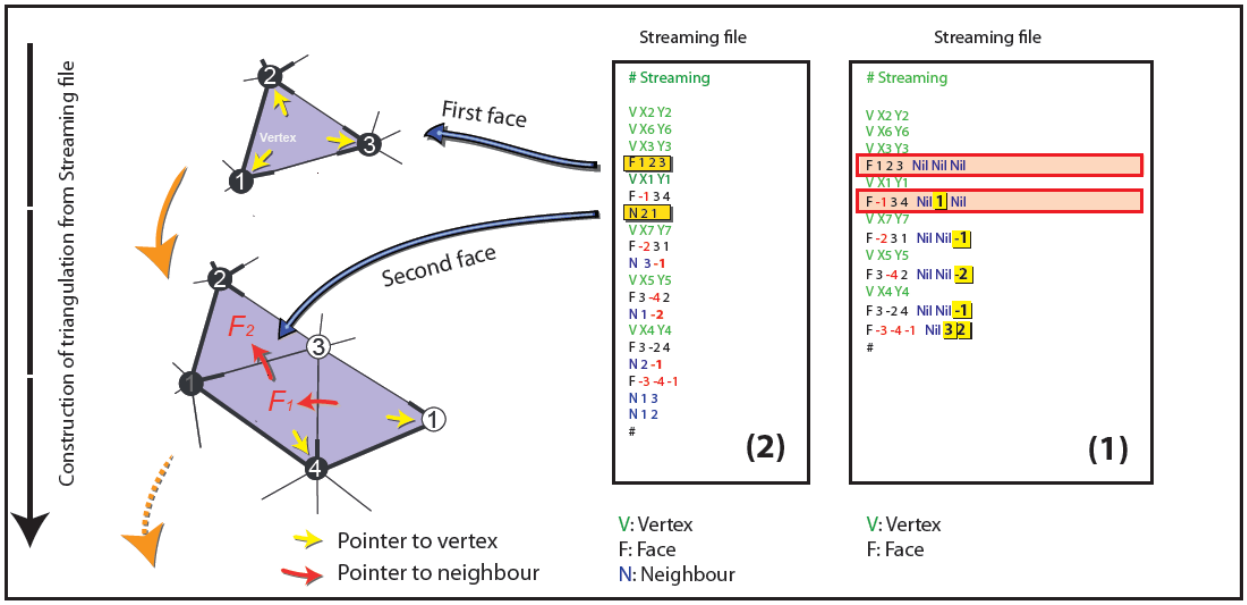


Figure 2. The final result of both method:
1) The first method: Neighborhood information inserted with triangles
2) The second method: Neighborhood information inserted After triangles.

 