Computing and Informatics, Vol. 37, 2018, 367–384, doi: 10.4149/cai_2018_2_367

XOR-BASED COMPACT TRIANGULATIONS

Abdelkrim Mebarki

Département d'Informatique
Faculté des Mathématiques et Informatique
Université des Sciences et de la Technologie d'Oran – Mohamed Boudiaf
USTO-MB, BP 1505 Oran El M'naouer, 31000, Oran Algérie
e-mail: abdelkrim.mebarki@univ-usto.dz

Abstract. Media, image processing, and geometric-based systems and applications need data structures to model and represent different geometric entities and objects. These data structures have to be time efficient and compact in term of space. Many structures in use are proposed to satisfy those constraints. This paper introduces a novel compact data structure inspired by the XOR-linked lists. The subject of this paper concerns the triangular data structures. Nevertheless, the underlying idea could be used for any other geometrical subdivision. The ability of the bitwise XOR operator to reduce the number of references is used to model triangle and vertex references. The use of the XOR combined references needs to define a context from which the triangle is accessed. The direct access to any triangle is not possible using only the XOR-linked scheme. To allow the direct access, additional information are added to the structure. This additional information permits a constant time access to any element of the triangulation using a local resolution scheme. This information represents an additional cost to the triangulation, but the gain is still maintained. This cost is reduced by including this additional information to a local sub-triangulation and not to each triangle. Sub-triangulations are calculated implicitly according to the catalog-based structure. This approach could be easily extended to other representation models, such as vertex-based structures or edge-based structures. The obtained results are very interesting since the theoretical gain is estimated to 38 % and the practical gain obtained from sample benches is about 34 %.

Keywords: XOR operator, XOR-linked list, XOR-based representation, triangular data structure, catalog-based structure

1 INTRODUCTION

Triangulations are widely used in almost all of the computer graphic applications. Several data structures are proposed to represent such subdivisions [1, 2]: vertex-based structures, edge-based structures, and triangle-based structures. All of these structures use an explicit indirection scheme to access to their elements. When dealing with huge triangulations, we need a large time to rearrange the structure in order to reduce the memory cost.

The trivial way to do this is either to compress the structure, or use an Out-of-Core structure. However, we need sometimes to still work In-Core in constant time, with not very huge triangulations, that is why we need compact data structures [3, 4] to delay as far as possible the swap between the main memory and the hard disk.

In this paper, a compact data structure is proposed that uses the bitwise XOR Operator to reduce the amount of references in the memory space. This idea is either applicable on vertex-based structures, edge-based structures, or triangle-based structures, and can be extended to 3D triangulations or any other subdivision.

1.1 Triangle-Based Representation

To illustrate the XOR-combined references idea, we use the explicit triangle-based representation (see Figure 1).

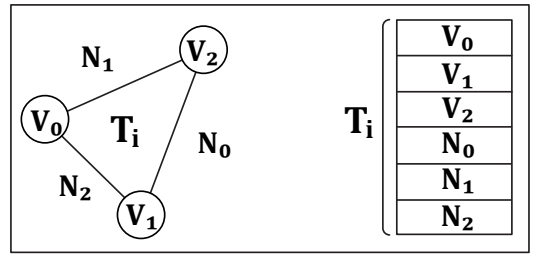

Figure 1. The explicit triangle-based representation. Each triangle maintains three references to its vertices and three references to its neighbors.

This representation [5] considers the triangle as the elementary object to represent the whole structure. Each triangle maintains references to its three vertices, and to its three neighbors. The whole triangulation is then represented using a table of triangles. Each entry in this table contains six references, three of them for the three triangle vertices, and the other three references indicate the three triangle neighbors. Since the number of triangles for a triangulation of n vertices is limited by $2n$, the global storage cost of such a triangulation is about $6n$ references.

The vertices in each triangle are indexed with 0, 1, and 2 in counterclockwise order. The neighbors are indexed in such a way that the neighbor indexed by i is opposite to the vertex with the same index.

For further details in this paper, two functions have to be defined to handle the item indices: $cw(i)$ and $ccw(i)$ which, given the index of a vertex in a triangle, compute the index of the next vertex of the same face in clockwise or counterclockwise order. Thus, for example the neighbor $cw(i)$ is the neighbor of the triangle which is next to neighbor i turning clockwise around the triangle. The triangle neighbor $cw(i)$ is also the first triangle encountered after the triangle when turning clockwise around vertex i of the triangle. In other words, the ccw and cw functions allow to enumerate the indices in cyclic way from 0 to 2 for the first function, and from 2 to 0 for the second function.

The remaining of this paper is organized as follows: Section 3 explains the major contribution of this paper. The Section 4 details the basic XOR linked date structure. In Section 5, the resolution scheme used to directly access triangles is presented. Section 6 presents some ideas to extend the XOR linked model to other representations. In Section 7, some sample benched results are presented to demonstrate the practical gain of this structure.

2 STATE OF THE ART

Data structures, and especially geometric data structures are used in many fields ranging from computer-aided design to finite elements [6].

The basic work has targeted the design of data structures [7] that are robust, fast and simple, while respecting the requirements of the different types and aims of the application [1]. This compromise is not always easy to manage, because applications are not always categorized. The efficiency in terms of execution time is required for real-time applications where the need to reduce the space used in memory usually happens in the background, while the economy in terms of memory footprint arises for handling large models, and induced in most cases a decrease of efficiency and loss of simplicity.

We will see in this part of the paper a description of the various solutions that have been proposed for the realization of geometric data structures to meet the various requirements imposed by the variety of applications ranging from simple and explicit data structures designed for small data sets, to compact or succinct data structures. We will also see the outlines of compression algorithms and data structures proposed for applications using auxiliary memories.

The most used structures in the state of the art are the primitive-based representations. Also known as index models or array-based models [8]. Since the triangulation can be seen as a set of vertices, edges, or faces, we can set these structures in three categories: edge-based structures, triangle-based structures, and vertex-based structures.

The first one deals with modes of representation that have been proposed for polygonal models in general, not just the triangulations. The basic object can be the edge or the half-edge. These structures have been developed around the modeling of solids and surfaces [9]. In such structures, the topological information is mainly

contained in the edges of the object. A very large literature is available for this set [10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

In this type of representations, the basic concept is the triangle. The basic structure uses two tables: one for the vertices where the geometric coordinates are stored, and the second one is for the triangles, where are stored the references of three vertices that define each triangle. This structure requires at least $6n$ references for a triangulation of n points.

This type of representation is widely used in practice [20, 21, 22], as it allows a minimum storage compared to other strategies, while maintaining a constant time response to incident requests.

In the vertex-based structures [23], the triangulation is represented as a graph of incidence between the nodes (vertices) of the triangulation. The structure is a list of vertices, where each vertex maintains its *degree* (the number of neighboring vertices), the list of these neighbors, and a mark indicating if the vertex is a vertex on the boundary or not. Given that the average degree of a vertex in a triangulation of n points is equal to 6, the cost of such a structure is $7n$ references [24, 25].

Compressing meshes (or triangulations) comprises encoding triangulation eliminating maximum redundancy by minimizing the entropy of the structure. The data structure after compression is unusable, and to be able to manipulate the structure or access items, you must decompress the entire structure and rebuild the explicit structure. For a detailed and comprehensive study of mesh compression techniques, a wide range of publications exists [26, 27, 28, 29, 30, 31, 32, 33].

2.1 Dealing with Huge Triangulations

The treatment of huge meshes of the order of billions of triangles is very limited by the above cited representations. Indeed, the indirection required to access vertices from a given face can be very costly when the index is very wide, even impossible when it exceeds the addressable range in the memory of the machine. Hence the importance of the *Out of Core* algorithms.

An intuitive solution is to not list the vertices, and to include directly the coordinates of the vertices in the faces. This solution that is called *Triangle Soup* would enable the faces of a mesh independently and avoid the step of indirection. But vertex update and neighborhood span is not as easy as in the indexed format. This technique was introduced for several types of applications [34, 35, 36]. The explicit representation of the parts of this mesh in working memory is usually indexed.

Another type of Out-of-Core approaches is the multi-resolution structures [37, 38, 39, 40, 41]. These structures allow access to the grid by induction on spatial subdivisions. These structures are also used to adjust the level of detail of the explicit structures. The involved common basic models are the trees (especially B-trees and its derivatives [42]).

For a data structure suitable for processing mesh, where this is done sequentially, Isenburg et al. [43] have proposed an ordered structure that they called *Streaming Mesh*. The goal is to have a format that allows switch the flow of mesh in the working

memory to process it without the need to load the resident parts in the auxiliary memory. To do this, the vertices and the faces of the mesh are inserted into the same structure. Gradually, as the faces scrolled in the structure, the vertices are introduced in the flow, or finalized when they are no longer referenced by any face. This format was used to compress large meshes [44, 45, 46], for the simplification of meshes [47], for the construction of the Delaunay triangulation [48]. In [49], this format is modified to allow direct access to neighboring information.

2.2 Compact Data Structures

Between explicit data structures, and the coding of triangulations, there is a third set of structures called *Compact Data Structures*. This type of structure has two objectives:

- Conceive a structure that is locally accessible: That means that we can access to its components in constant time.
- At the same time, we have to minimize the space occupied by the structure, so that it holds in working memory, and consequently to delay as long as possible the use of the transfer with the auxiliary memory.

The first compact data structure was proposed by Kallmann and Thalmann, they called it *Star Vertices* [24]. It is a vertex-based representation: each vertex handles a list of all of its adjacent vertices (the vertex stores the size of this list), resulting in $6n$ references plus n integers (sizes of lists) to represent the whole triangulation. However, the internal structure has no longer an explicit representation of faces, and queries cost time is proportional to the degree of the involved vertex.

Blandford et al. [50, 4, 51] proposed a compact data structure for representing simplicial meshes, requiring in practice 40 bpt [1]. The representation does admit an edge-based or vertex-based representation, providing basic update operations and standard local navigation between triangles (performing these operations takes $O(1)$ time for the case of meshes with bounded vertex degree). To gain in memory, difference vertex labels are used instead of real pointers, and a preprocessing step consisting of relabelling vertices, for reducing the differences, is needed. This approach takes advantage of properties of graphs with small separators and require some assumptions on the input data.

Devillers et al. have proposed an optimal way of representing a triangulation of n points using $3.24n$ bits [52, 53], with an additional storage cost which is asymptotically negligible (in the case of a triangulation of a topological sphere; for the triangulation bounded by a polygon of arbitrary size the cost is 2.17 bpt. The idea is to gather triangles in tiny patches of size between $\frac{\log n}{12}$ and $\frac{\log n}{4}$, and to introduce a graph of patches to describe adjacency relations between them. Each patch is then represented by a reference to a catalog, consisting of all different tiny patches

[1] bits per triangle

of size less than $\frac{\log n}{4}$. The whole size of all references to the catalog gives the dominant term of 3.24 bpv [2], while the representation of the graph of patches requires a negligible amount of space.

In [54], the authors proposed some catalogs and evaluate in detail the amount of storage needed for representing triangulations using this approach. The implementation showed that the expected improvements are indeed obtained in practice.

In [55], Aleardi et al. proposed a new way of designing compact data structures which can be dynamically maintained. They described a new class of data structures, called *Editable SQuad (ESQ)*, offering the same navigational and storage performance as previous works, while supporting local editing in amortized constant time.

The proposed data structures are simple to implement and provided with an analysis of worst case storage bounds. The simplest solution uses 6 rpv [3], while supporting updates operations in $O(1)$ amortized time: this is obtained with a reordering of input data which allows to encode the map from triangles to vertices. The most compact data structure makes use of a grouping strategy between adjacent triangles, and uses only 4.8 rpv, while still supporting efficient navigation and update operations.

In [56], Gurung et al. proposed a data structure for representing the connectivity of manifold triangle meshes that they called *LR (Laced Ring)*. This structure provides the option to store on average either 1.08 rpt [4] or 26.2 bpt. Its construction, from an input mesh that supports constant-time adjacency queries, has linear space and time complexity, and involves ordering most vertices along a nearly-Hamiltonian cycle.

3 CONTRIBUTION

In this paper, we present a novel compact data structure for 2D triangulations based on the bitwise XOR operator [57].

The underlying idea concerns the indexed structures. The XOR-scheme is used to reduce the number of explicit references by combining them two by two.

A bitwise operation operates on binary numerals at the level of their individual bits. It is a fast, primitive action directly supported by the processor, and is used to manipulate values for comparisons and calculations.

A bitwise XOR takes two bit patterns of equal length and performs the logical *exclusiveOR* operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1. Otherwise, we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same.

[2] bits per vertex
[3] references per vertex
[4] references per triangle

The basic idea of the proposed structure is to combine references in the same way as the XOR-linked lists.

The XOR-linked list merges the next and the previous references into one single field using the bitwise XOR operator (see Figure 2). The access to a given element in the linked list is only sequential, that means that we access each element either from its predecessor neighbor, or from its successor neighbor. The obtained gain by replacing the explicit next and previous references with their XOR-combined reference is 50 %, since these two references are replaced by only one reference.

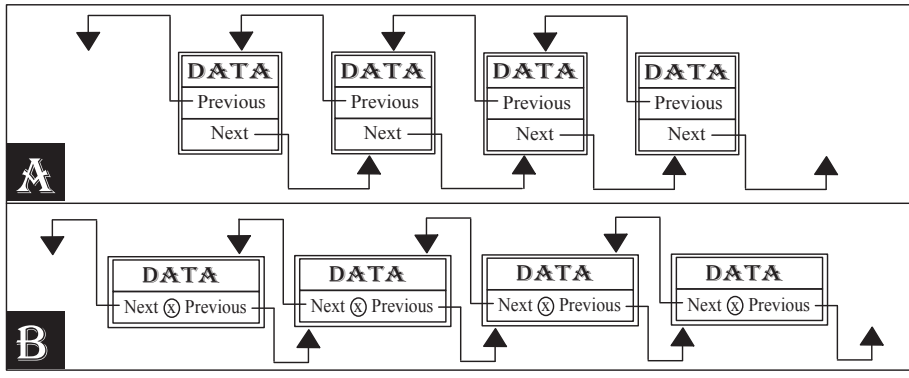

Figure 2. A) The explicit representation of liked list. B) The XOR-linked list: The implicit representation of a linked list using the XOR operator to join the next and previous pointers.

The resolution of the references (that means retrieving the real previous and next references) is done on the fly each time we access to an element. Since the access to the element is sequential, each element is accessed either from its predecessor or successor in the list. So, in the first case, we retrieve the successor reference from the predecessor one and the XOR combined references, and in the second time, we retrieve the predecessor reference from the successor one and the XOR combined references.

4 THE BASIC XOR-BASED TRIANGULAR STRUCTURE

When walking in a triangulation, we access each triangle from its neighborhood. This context defines for each accessed triangle at least one of its neighbor references, and at least two of its three vertex references.

Let us consider the triangle T_i. Where: V_0, V_1, V_2 are its vertices; N_0, N_1, N_2 are its neighbors.

The XOR-based representation of the triangle T_0 is defined by:

$$T_0 \begin{cases} V_0 \oplus V_1 \oplus V_2, \\ N_0 \oplus N_1, \\ N_0 \oplus N_2. \end{cases}$$

As we can remark, the number of references used to represent each triangle in this XOR-based representation is only three, instead of six references in the explicit triangle-based representation. The gain obtained is then equal to 50 %.

When accessing this triangle, coming from a neighbor N_i sharing the two vertices $V_{cw(i)}$ and $V_{ccw(i)}$, the above XOR-based representation allows us to retrieve all of the three vertex references and all of the three neighbor references as follows:

$$\left.\begin{array}{ll} V_0 \oplus V_1 \oplus V_2, & V_{cw(i)}, V_{ccw(i)} \\ N_0 \oplus N_1, & N_i \\ N_0 \oplus N_2, & \end{array}\right\} \Longrightarrow \begin{array}{l} V_i, V_{cw(i)}, V_{ccw(i)}, \\ N_i, N_{cw(i)}, N_{ccw(i)}. \end{array}$$

In order to affect the indices to the vertices and neighbors, a conventional order can be adopted: The vertices are sorted, and the indices are assigned from 0 to 2 in the same order for all the triangles of the subdivision.

The main disadvantage that we have here is the lack of direct access to triangles. The basic scheme presented here needs to have a context each time we access to a triangle. This context is defined as follows:

- One of its neighbors.
- Two of its three vertices.

If we want to enumerate all of the triangles (or read all of them from the triangle container for example), we lack this context. The only way to enumerate all of the triangles is to walk in the triangulation spanning all of its simplices.

5 THE RESOLUTION SCHEME

To be able to directly access to any triangle, we define a Local Resolution Scheme. The Local Resolution Scheme is an algorithm that can extract all of the triangle references from a restricted local neighboring sub-triangulation. That means that we need not to browse the whole triangulation to extract a given triangle references, just a local restricted neighborhood is sufficient to retrieve these references.

To establish such a scheme, we consider a local neighborhood according to a subdivision of the triangulation into packages likewise the catalog-based data structure [54] using only two packages: quadrangles and pentagons. Although we can go further by defining largest catalogs, the quadrangles and pentagons are widely sufficient.

As demonstrated in [54], any triangulation can be represented as a decomposition of sub-triangulations using only these catalog packages.

The global structure remains triangle-based, and the mentioned subdivision is only virtual. We need only to associate triangles belonging to packages between them, to be able to retrieve all of the lacking references.

5.1 Rearranging the Triangulation

In order to minimize additional costs, the triangles of the original triangulation have to be rearranged in such way that each package has its triangles stored in a contiguous area in the memory (that means that its triangles have sequential indices in the triangle container).

If we take the advantage of the memory size word that are at least four aligned bytes (for a 32 bits machine), we can remark that the two least significant bits are useless in the reference. We can squat these bits to store the indices of the triangles into their belonging packages: The triangles of a given quadrangle are indexed 0 and 1, and the triangles of a pentagon are indexed 0, 1, and 2.

Now, all of the triangles are stored in the memory, and all of the package triangles are stored consequently.

Thus, when we access to each triangle directly in the container, we consult its index, and by consulting its direct contiguous neighborhood indices, we can easily determine to which kind of package it belongs, and who are its neighbors in this package.

5.2 Resolution Scheme into a Quadrangle

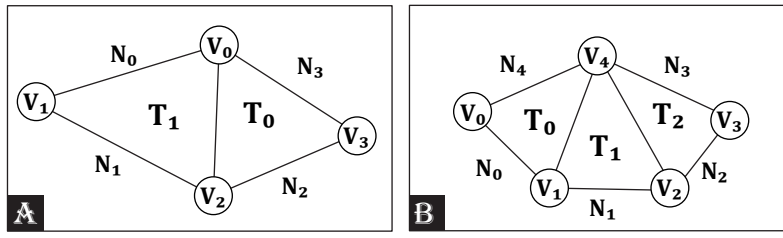

Figure 3. a) Two triangles grouped into one quadrangle patch, b) three triangles grouped into one pentagon patch

Let us suppose that we need to access to a specified triangle in a given quadrangle, the associated resolution scheme of the two triangles has to allow us to calculate all of the references. Supposing we have the two adjacent triangles T_0 and T_1 those belong to the same quadrangle (see Figure 3). The triangle T_0 is represented as follows:

- *vertices*: V_0, V_2, V_3,
- *neighbors*: N_2, N_3, T_1.

The triangle T_1 is represented as follows:

- *vertices*: V_0, V_1, V_2,
- *neighbors*: N_1, T_0, N_0.

The XOR-schemes of the two triangles are:

$$T_0 \begin{cases} V_0 \oplus V_2 \oplus V_3, \\ N_2 \oplus N_3, \\ N_2 \oplus T_1, \end{cases}$$

$$T_1 \begin{cases} V_0 \oplus V_1 \oplus V_2, \\ N_1 \oplus T_0, \\ N_1 \oplus N_0. \end{cases}$$

Since we have the two triangle references T_0 and T_1, we can resolve all of the other two triangle references using this scheme system and two additional fields: the V_0 and V_2 references. The resolution can be done as follows:

$$\left.\begin{array}{l} V_0 \oplus V_2 \oplus V_3, \\ V_0 \oplus V_1 \oplus V_2, \\ V_0, V_2, \end{array}\right\} \Longrightarrow V_0, V_1, V_2, V_3,$$

$$\left.\begin{array}{l} T_0, T_1, \\ N_2 \oplus N_3, N_2 \oplus T_1, \\ N_1 \oplus T_0, N_1 \oplus N_0, \end{array}\right\} \Longrightarrow N_0, N_1, N_2, N_3.$$

The indices of the vertices and neighbors in each triangle can be established by sorting the vertices, and assigning the indices according to the vertex orders as described in the convention cited in Section 4.

5.3 Resolution Scheme into a Pentagon

Let us suppose now that we need to access to a specified triangle in a given pentagon, the associated resolution scheme of the three triangles has to allow us to calculate all of the references. Supposing we have three adjacent triangles T_0, T_1, and T_2 belonging to the same pentagon (see Figure 3), the triangle T_0 is represented as follows:

- *vertices*: V_0, V_1, V_4,
- *neighbors*: T_1, N_4, N_0.

The triangle T_1 is represented as follows:

- *vertices*: V_1, V_2, V_4,
- *neighbors*: T_2, T_0, N_1.

The triangle T_2 is represented as follows:

- *vertices*: V_2, V_3, V_4,
- *neighbors*: N_3, T_1, N_2.

The XOR-schemes of the three triangles are the following:

$$T_0 \begin{cases} V_0 \oplus V_1 \oplus V_4, \\ T_1 \oplus N_4, \\ T_1 \oplus N_0, \end{cases}$$

$$T_1 \begin{cases} V_1 \oplus V_2 \oplus V_4, \\ T_2 \oplus T_0, \\ T_2 \oplus N_1, \end{cases}$$

$$T_1 \begin{cases} V_2 \oplus V_3 \oplus V_4, \\ N_3 \oplus T_1, \\ N_3 \oplus N_2. \end{cases}$$

Using the same approach, we can retrieve all of the triangle references into a pentagon, using their XOR schemes and two additional words: the V_1 and V_4 references. These words that are the middle vertices of the pentagon are considered as the key gate of the package, that represent the local context allowing us to retrieve all of the package references. The resolution is as follows:

$$\left.\begin{array}{l} V_0 \oplus V_1 \oplus V_4, \\ V_1 \oplus V_2 \oplus V_4, \\ V_2 \oplus V_3 \oplus V_4, \\ V_1, V_4, \end{array}\right\} \Longrightarrow V_0, V_1, V_2, V_3, V_4,$$

$$\left.\begin{array}{l} T_0, T_1, T_2, \\ T_1 \oplus N_4, T_1 \oplus N_0, \\ T_2 \oplus T_0, T_2 \oplus N_1, \\ N_3 \oplus T_1, N_3 \oplus N_2, \end{array}\right\} \Longrightarrow N_0, N_1, N_2, N_3, N_4.$$

Using the same convention as in Section 4, the indices of the vertices and faces in each triangle can be established by sorting the vertices, and assigning the indices according to the vertex orders.

5.4 The Estimated Gain

The global gain in the case of the basic XOR-based structure is equal to 50 % since we represent each triangle using three references instead of six. However, in the proposed XOR-based structure allowing direct access to triangles, the gain is reduced.

For each two triangles grouped into a quadrangle, we add two additional references and for each three triangles grouped into a pentagon, we add two additional references.

Theoretically, the maximum gain we can obtain corresponds to the case where all the triangles are grouped into pentagons. In this case, and for a triangulation of n vertices, we can obtain $2n$ triangles, and thus $\frac{2n}{3}$ pentagons.

In this configuration, the global cost is equal to $\frac{22n}{3}$ references instead of $12n$ in the explicit triangle-based original structure. The gain is then equal to 38 %.

The worst gain we can obtain is when all of the triangles are grouped into quadrangles. In this case, the global cost is equal to $8n$. The gain is then equal to 33 %.

6 EXTENSION TO VEREX-BASED AND EDGE-BASED STRUCTURES

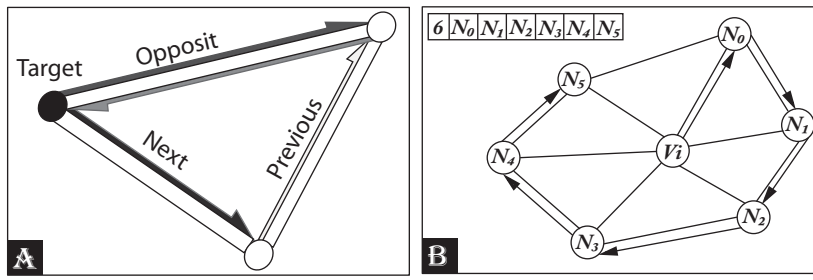

Figure 4. a) Half-edge-based representation, b) vertex-based representation

The XOR-linked representation is not restricted to triangle-based structures. It is also applicable to vertex-based and edge-based structures (see Figure 4).

6.1 Vertex-Based Representation

In the vertex-based structure, the triangulation is represented as a set of vertices. Each vertex is associated to a list of its incident (neighbors) vertices. The whole triangulation is then a set of lists: Each list corresponds to a vertex, including its degree (number of its neighbors), and the list of these neighbor references. The global cost for this representation is equal to $6n$ (where n is the number of vertices). Using directly the basic XOR-linked list to implement the vertex lists, the global gain cost is about 50 %.

6.2 Edge-Based Representation

Let us consider the basic half-edge scheme, where each half-edge stores four references:

- the reference of the target or the departure vertex;
- the reference of the opposite half-edge;
- the reference of the previous half-edge in the same face;
- the reference of the next half-edge in the same face.

This representation is similar to the XOR-linked list, and could be converted to a XOR-based scheme easily. The *next* and *previous* references would be replaced by the result of their XOR combination ($previous \oplus next$).

7 EXPERIMENTAL RESULTS

In order to evaluate the practical impact of the proposed structure, the XOR-based scheme is used to represent a sample Delaunay triangulation in static mode. That means that the Delaunay triangulation is constructed, and then converted from the explicit triangle-based representation to the XOR-based representation. The obtained results are shown in Table 1:

Number of points	100 000 points	1 000 000 points
Number of triangles	199 298	1 997 498
Number of quadrangles	98 764	997 270
Number of pentagons	590	986
Gain in Memory	33.38 %	33.34 %
Browsing time in Explicit Structure	1.661 s	16.476 s
Browsing time in XOR-Based Structure	1.966 s	18.152 s
Loss in Time	15.51 %	9.23 %

Table 1. Number of packages, gain in memory and browsing time for two XOR-based triangulations

As we can remark, the practical results are close to the theoretical expectations. We gain a third of the memory space compared to the explicit triangle-based structure. The loss in execution time is not very significant, since we lose less than 15 % of time. This execution time is calculated by browsing the whole triangulation and outputting all of the triangulation elements into an external file.

8 CONCLUSION

This paper has presented a novel approach for geometrical data structures inspired by the XOR-linked lists. The basic idea is to combine different references using the XOR operator to reduce by two the number of references. Unfortunately, this ratio is not reached if we need to maintain a direct access to each element of the structure. To guarantee the direct access, a local resolution scheme is defined using additional information. With this additional information, the structure remains beneficial and useful. The estimated and the obtained practical gain could be improved, if this method is combined with other compact data structures.

Acknowledgements

I would like to thank Olivier Devillers and Sylvain Pion from INRIA, France, for the very interesting, helpful and fruitful discussions which have oriented me in the elaboration of this paper.

REFERENCES

[1] Goodrich, M. T.—Ramaiyer, K.: Geometric Data Structures. In: Sack, J.-R., Urrutia, J. (Eds.): Handbook of Computational Geometry. Chapter 10. Elsevier Science, 2000, pp. 463–489.

[2] Mebarki, A.: Les Structures de Données Triangulaires Compactes: Théorie et Implémentation. Editions Universitaires Européennes, 2013 (in French).

[3] Mebarki, A.: Implantation de Structures de Données Compactes pour les Triangulations. Ph.D. thesis, Université de Nice-Sophia Antipolis, France, April 2008 (in French).

[4] Blandford, D. K.—Blelloch, G. E.—Cardoze, D. E.—Kadow, C.: Compact Representations of Simplicial Meshes in Two and Three Dimensions. International Journal of Computatinal Geometry and Applications, Vol. 15, 2005, No. 1, pp. 3–24, doi: 10.1142/S0218195905001580.

[5] Boissonnat, J.-D.—Devillers, O.—Pion, S.—Teillaud, M.—Yvinec, M.: Triangulations in CGAL. Computational Geometry, Vol. 22, 2002, No. 1-3, pp. 5–19, doi: 10.1016/S0925-7721(01)00054-2.

[6] Zachmann, G.—Langetepe, E.: Geometric Data Structures for Computer Graphics. Proceedings of ACM SIGGRAPH, July 27–31, 2003.

[7] Mehta, D. P.—Sahni, S.: Handbook of Data Structures and Applications. Chapman & Hall/CRC Computer and Information Science Series. Chapman & Hall/CRC, 2004, doi: 10.1201/9781420035179.

[8] Castelli Aleardi, L.—Devillers, O.: Explicit Array-Based Compact Data Structures for Planar and Surface Meshes. XIV Spanish Meeting on Computational Geometry, Alcala de Henares, Spain, 2011, Special edition for Ferran Hurtado Birthday.

[9] Mäntylä, M.: An Introduction to Solid Modeling. Computer Science Press, Inc., New York, NY, USA, 1987.

[10] Baumgart, B. G.: Winged Edge Polyhedron Representation. Technical report, Stanford University, Stanford, CA, USA, 1972, doi: 10.21236/AD0755141.

[11] Baumgart, B. G.: Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975.

[12] Baumgart, B. G.: Geometric Modeling for Computer Vision. Ph.D. thesis, Stanford University, USA, August 1974.

[13] Preparata, F. P.—Shamos, M. I.: Computational Geometry: An Introduction. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[14] Guibas, L. J.—Stolfi, J.: Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC '83), ACM Press, New York, NY, USA, 1983, pp. 221–234, doi: 10.1145/800061.808751.

[15] Weiler, K.: Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. IEEE Computer Graphics and Applications, Vol. 5, 1985, No. 1, pp. 21–40, doi: 10.1109/MCG.1985.276271.

[16] Lienhardt, P.: Subdivisions of n-Dimensional Spaces and n-Dimensional Generalized Maps. Proceedings of the Fifth Annual Symposium on Computational Geometry (SCG '89), ACM Press, New York, NY, USA, 1989, pp. 228–236, doi: 10.1145/73833.73859.

[17] Halbwachs, Y.—Hjelle, Ø.: Generalized Maps in Geological Modeling: Object-Oriented Design of Topological Kernels. In: Langtangen, H. P., Bruaset, A. M., Quak, E. (Eds.): Advances in Software Tools for Scientific Computing. Springer-Verlag, Lecture Notes in Computational Science and Engineering, Vol. 10, 1999, pp. 339–356.

[18] Hjelle, Ø.—Dæhlen, M.: Triangulations and Applications. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] Campagna, S.—Kobbelt, L.—Seidel, H.-P.: Directed Edges – A Scalable Representation for Triangle Meshes. Journal of Graphic Tools, Vol. 3, 1998, No. 4, pp. 1–11, doi: 10.1080/10867651.1998.10487494.

[20] Shewchuk, J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In: Lin, M. C., Manocha, D. (Eds.): Applied Computational Geometry Towards Geometric Engineering. Springer-Verlag, Lecture Notes in Computer Science, Vol. 1148, 1996, pp. 203–222, doi: 10.1007/BFb0014497.

[21] TRIANGLE: Mesh Generator and Delaunay Triangulator. 2014.

[22] CGAL: Computational Geometry Algorithms Library. `www.cgal.org`.

[23] Cline, A. K.—Renka, R. J.: A Storage-Efficient Method for Construction of a Thiessen Triangulation. Rocky Mountain Journal of Mathematics, Vol. 14, 1984, No. 1, pp. 119–140.

[24] Kallmann, M.—Thalmann, D.: Star Vertices: A Compact Representation for Planar Meshes with Adjacency Information. Journal of Graphics Tools, Vol. 6, 2001, No. 1, pp. 7–18, doi: 10.1080/10867651.2001.10487533.

[25] Kallmann, M.: Object Interaction in Real-Time Virtual Environments. Ph.D. thesis, Swiss Federal Institute of Technology (EPFL), January 2001, Thesis number 2347.

[26] Alliez, P.—Gotsman, C.: Recent Advances in Compression of 3D Meshes. In: Dodgson, N. A., Floater, M. S., Sabin, M. A. (Eds.): Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization. Springer, Berlin, Heidelberg, 2005, pp. 3–26.

[27] Gumhold, S.: Mesh Compression. Ph.D. thesis, University of Tübingen, July 2000.

[28] Gotsman, C.—Gumhold, S.—Kobbelt, L.: Simplification and Compression of 3D Meshes. Proceedings of the European Summer School on Principles of Multiresolution in Geometric Modelling (PRIMUS), August 2001, pp. 319–361.

[29] Isenburg, M.: Compression and Streaming of Polygon Meshes. Ph.D. thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2005.

[30] Lewiner, T.: Mesh Compression from Geometry. Ph.D. thesis, Université de Pierre et Marie Curie (Paris VI), Paris, December 2005.

[31] Peng, J.—Kim, C.-S.—Jay Kuo, C. C.: Technologies for 3D Mesh Compression: A Survey. Journal of Visual Communication and Image Representation, Vol. 16, 2005, No. 6, pp. 688–733.

[32] Rossignac, J.: 3D Mesh Compression. Technical Report GIT-GVU-03-21, Georgia Institute of Technology, Atlanta, GA, USA, 2003.

[33] Rossignac, J.: 3D Mesh Compression. In: Hansen, C., Johnson, C. R. (Eds.): Visualization Handbook. Academic Press, Inc., Orlando, FL, USA, 2004, pp. 339–356.

[34] Lindstrom, P.: Out-of-Core Simplification of Large Polygonal Models. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00), ACM Press/Addison-Wesley Publishing Co, New York, NY, USA, 2000, pp. 259–262, doi: 10.1145/344779.344912.

[35] Lindstrom, P.—Silva, C. T.: A Memory Insensitive Technique for Large Model Simplification. Proceedings of the Conference on Visualization (VIS '01), IEEE Computer Society, Washington, DC, USA, 2001, pp. 121–126, doi: 10.1109/VISUAL.2001.964502.

[36] Wu, J.—Kobbelt, L.: A Stream Algorithm for the Decimation of Massive Meshes. Proceedings of Graphics Interface 2003, CIPS, Canadian Human-Computer Communication Society and A. K. Peters Ltd., June 2003, pp. 185–192. ISBN 1-56881-207-8, ISSN 0713-5424.

[37] De Floriani, L.—Kobbelt, L.—Puppo, E.: A Survey on Data Structures for Level-of-Detail Models. In: Dodgson, N. A., Floater, M. S., Sabin, M. A. (Eds.): Advances in Multiresolution for Geometric Modelling. Mathematics and Visualization Series, Springer, Berlin, Heidelberg, 2005, pp. 49–74.

[38] Cignoni, P.—Montani, C.—Rocchini, C.—Scopigno, R.: External Memory Management and Simplification of Huge Meshes. IEEE Transactions on Visualization and Computer Graphics, Vol. 9, 2003, No. 4, pp. 525–537, doi: 10.1109/TVCG.2003.1260746.

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

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

[41] Shaffer, E.—Garland, M.: A Multiresolution Representation for Massive Meshes. IEEE Transactions on Visualization and Computer Graphics, Vol. 11, 2005, No. 2, pp. 139–148, doi: 10.1109/TVCG.2005.18.

[42] Silva, C. T.—Chiang, Y.-J.—El-Sana, J.—Lindstrom, P.: Out-of-Core Algorithms for Scientific Visualization and Computer Graphics. IEEE Visualization Conference, 2002, Boston, Massachusets, Course Notes.

[43] Isenburg, M.—Lindstrom, P.: Streaming Meshes. Proceedings of the 16th IEEE Visualization Conference (VIS 2005), October 23–28, 2005, Minneapolis, MN, USA, IEEE Computer Society, 2005.

[44] Isenburg, M.—Lindstrom, P.—Snoeyink, J.: Streaming Compression of Triangle Meshes. Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP '05), Eurographics Association, Aire-la-Ville, Switzerland, 2005, Art. No. 111, doi: 10.1145/1187112.1187276.

[45] Isenburg, M.—Gumhold, S.: Out-of-Core Compression for Gigantic Polygon Meshes. ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2003, Vol. 22, 2003, No. 3, pp. 935–942, doi: 10.1145/1201775.882366.

[46] Isenburg, M.—Lindstrom, P.—Gumhold, S.—Shewchuk, J.: Streaming Compression of Tetrahedral Volume Meshes. Proceedings of Graphics Interface 2006 (GI '06), Canadian Information Processing Society, Toronto, Ontario, Canada, 2006, pp. 115–121.

[47] Isenburg, M.—Lindstrom, P.—Gumhold, S.—Snoeyink, J.: Large Mesh Simplification Using Processing Sequences. Proceedings of the 14th IEEE Visualization 2003 (VIS '03), IEEE Computer Society, Washington, DC, USA, 2003, pp. 465–472, doi: 10.1109/VISUAL.2003.1250408.

[48] Isenburg, M.—Liu, Y.—Shewchuk, J.—Snoeyink, J.: Streaming Computation of Delaunay Triangulations. ACM SIGGRAPH 2006 Papers (SIGGRAPH '06), ACM Press, New York, NY, USA, 2006, pp. 1049–1056, doi: 10.1145/1179352.1141992.

[49] Abid, K.—Mebarki, A.—Hidouci, W. K.: Modified Streaming Format for Direct Access Triangular Data Structures. International Journal of Image, Graphics, and Signal Processing (IJIGSP), Vol. 6, 2014, No. 2, pp. 14–22.

[50] Blandford, D. K.—Blelloch, G. E.—Cardoze, D. E.—Kadow, C.: Compact Representations of Simplicial Meshes in Two and Three Dimensions. The 12th International Meshing Roundtable, 2003, pp. 135–146.

[51] Blandford, D. K.: Compact Data Structures with Fast Queries. Ph.D. thesis, Carnegie Mellon University, USA, February 2006.

[52] Castelli Aleardi, L.—Devillers, O.—Schaeffer, G.: Succinct Representation of Triangulations with a Boundary. Proceedings of Workshop on Algorithms and Data Structures (WADS 2005). Springer, Lecture Notes in Computer Science, Vol. 3608, 2005, pp. 134–145.

[53] Castelli Aleardi, L.—Devillers, O.—Schaeffer, G.: Succinct Representations of Planar Maps. Theoretical Computer Science, Vol. 408, 2008, No. 2-3, pp. 174–187, doi: 10.1016/j.tcs.2008.08.016.

[54] Castelli Aleardi, L.—Devillers, O.—Mebarki, A.: Catalog-Based Representation of 2D Triangulations. International Journal of Computational Geometry and Applications, Vol. 21, 2011, No. 4, pp. 393–402.

[55] Castelli Aleardi, L.—Devillers, O.—Rossignac, J.: ESQ: Editable SQuad Representation for Triangle Meshes. 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI 2012), Ouro Preto, Brazil, August 22–25, 2012, pp. 110–117.

[56] Gurung, T.—Luffel, M.—Lindstrom, P.—Rossignac, J.: LR: Compact Connectivity Representation for Triangle Meshes. ACM Transactions on Graphics, Vol. 30, 2011, No. 4, pp. 67:1–67:8.

[57] ISO/IEC 14882. Programming Languages C++. International Organisation for Standardisation, Geneva, Switzerland, 2003.

Abdelkrim Mebarki is currently Assistant Professor at the Science and Technology University of Oran (Mohamed Boudiaf). He received his Ph.D. degree from the University of Nice Sophia Antipolis (France) in 2008. He has his Master degree from the same university and the degree of state engineer from the Science and Technology University of Oran (Mohamed Boudiaf). He is now Assistant Researcher Professor in the same university. His research works include scientific visualization and triangular data structures.