A predictive approach for a real-time remote visualization of large meshes

Justin-Hervé Noubissi, Christophe Guillet, Jean-Luc Martinez, Frédéric Merienne

A predictive approach for a real-time remote visualization of large meshes

Justin-Hervé NOUBISSI[1], Christophe GUILLET[2]
[1] Arts et Métiers ParisTech, CNRS, Le2I
Institut Image
Chalon-sur-Saône, France

Jean-Luc MARTINEZ[1], Frédéric MERIENNE[1]
[2] Université de Bourgogne, CNRS, Le2I
Institut Image
Chalon-sur-Saône, France

Abstract. **Remote access to large meshes is the subject of studies since several years. We propose in this paper a contribution to the problem of remote mesh viewing. We work on triangular meshes. After a study of existing methods of remote viewing, we propose a visualization approach based on a client-server architecture, in which almost all operations are performed on the server. Our approach includes three main steps: a first step of partitioning the original mesh, generating several fragments of the original mesh that can be supported by the supposed smaller Transfer Control Protocol (TCP) window size of the network; a second step called pre-simplification of the mesh partitioned, generating simplified models of fragments at different levels of detail, which aims to accelerate the visualization process when a client(that we also call remote user) requests a visualization of a specific area of interest; the final step involves the actual visualization of an area which interest the client, the latter having the possibility to visualize more accurately the area of interest, and less accurately the areas out of context. In this step, the reconstruction of the object taking into account the connectivity of fragments before simplifying a fragment is necessary.**

Keywords: mesh, sewing, partitioning, simplification, visualisation.

I. INTRODUCTION

The advent of 3D models leads nowadays to the need to exchange more and more. Thus, the increased requirements in terms of data volume, the fact that our digital model can be on remote locations, making it essential to further improve the performance of simplification techniques according two main axes: the degree of simplification and speed of simplification. If treatments are possible simplification in real time, it becomes possible to take into account the criteria snapshots (application context) and thus hope to obtain better distribution of the rate of simplification. Many approaches have been proposed, allowing the visualization of local or global mesh distance regardless of the TCP window size of the network. Others have focused on reducing the volume of data, but they are limited in terms of mesh quality and reduction rate. Thus, the real-time visualization of large meshes distance taking into the user's needs remains a need for concern. In this context, we propose to develop a new method of remote monitoring adapted to the user needs. Our visualization is made taking into account the TCP window size of the network, the user also has the possibility to view several parts of the object at different levels of detail: the more specific area of interest to a higher level of detail, and other areas out of context less precise, with lower levels of detail. Our approach preserves the connectivity within the meaning of fragments (fragments of a seam with different levels of detail and taking into account the connectivity of said fragments before partitioning is performed).

II. RELATED WORK

Extremely progressive development of networks and the Internet has pushed for several years researchers to find strategies to access and exchange of 3D geometric data remotely. So, many researchers have focused on the difficulty to transfer the data for visualization, data that are often very large.

[12] introduced the first system of progressive mesh (PM) for generating various levels of mesh simplification, which aims to reduce the volume of data at each step of the simplification. A PM is a linear sequence of increasingly coarse meshes built from an input mesh by repeatedly applying edge collapse operations. It provides a continuous resolution representation of an input mesh and is useful for efficient storage, rendering and transmission. [8] proposed a simplification method depends on the point of view, to display interactive objects of several million triangles. The authors propose to perform pre-processing (pre-simplification), building a tree of sub-meshes while optimizing disk access. During the exploration of the tree, only nodes required viewing are displayed. [13], [17], [1] propose a mesh cutting approach. The input mesh is partitioned into pieces small enough, which are then simplified individually. The partition boundaries are left untouched such that the simplified pieces can be stitched back together seamlessly. [4] proposed an approach based on client/server architecture for generating static levels of detail (LOD) of the simplification of a given mesh. With this approach we, only select the detail which corresponding to the client request for viewing. [9] proposed an approach based on a client / server architecture. To improve adaptive visualization and reduce transmission time, the system uses the cache and a prediction mechanism on the client side. They subdivide the view dependent tree into blocks which allow selective refinement and maintain on the server side a list of active odes for each client connected. The server responses the requests from the clients for sending the update operations needed to satisfy the visual query. [16] proposed a client/server framework for view dependent streaming of progressive meshes: the servers send the base mesh and the vertex hierarchy of 3D models to the client. The client creates further requests for selective refinement operations, and the server continues to send the rest level of detail of the models until the requests stop. However, the application cannot support interactive exploration of models, which involves selective refinement. [7] designed a client/server system for allowing an application to deliver a mesh progressively, which overcomes some limitations of

previous work. [21] proposed an approach based on the compression of triangles. They decompose a mesh into a set of clusters, each containing a few thousand vertices and triangles.
Then, the compression and decompression are performed at the granularity of clusters. At runtime, if an application requires a triangle, the proposed technique first identifies a cluster called the triangle, without decompressing decompresses the cluster of other clusters, and returns the uncompressed data to the application. However, their method is not progressive, and therefore, the overall imaging of a mesh would require to fully load into memory. [5] and [2] propose an approach that consists on hierarchical splitting mesh into several fragments and compress each fragment separately. With this, the can decompress fragment separately, without waiting full decoding of entire mesh.

Previous work has allowed us to identify three main categories of approaches for remote viewing: mesh simplification as progressive mesh, compression and mesh partitioning. The first is to generate multiple layers of simplified representations of the original mesh, each level representing a crude form of the mesh at the previous level, with fewer vertices. Over the mesh is coarse, the less of vertices, and the amount of information to be displayed is small. This approach has thus to be able to accelerate time data visualization, because the more the amount of data is small, more time viewing is low. This approach, while interesting, suffers from the fact that it does not propose to view only a specific area of interest, forcing the transfer of the original model in its entirety. It thus has a further disadvantage, that of failure to take into account the TCP window size of the network. The second approach is data compression, proposes to encode geometry and topology of the original mesh to represent it with the least bit possible. Thus, the data volume is reduced, and the model can pass through a much smaller TCP window than the previous approach. A decoding phase is then carried out after transfer to reconstruct the object to view, which can be time consuming. A third approach is to partition the original mesh 3D model, generating several fragments representing parts of the large mesh. Thus, there is the possibility, in contrast to previous approaches, to only transfer over the network the area of interest of the user. Although this is a step forward, this approach seems limited, since the user does not see the vicinity of its area of interest or the entire object from which its area of interest. More, the partition being performed according to the number of fragments we want to obtain, and therefore, according to criteria that do not depend on the width of the windows TCP network, it may also have limits network.

In this context, we propose a client/server approach, in which almost all operations are performed on the server, and which taking into account the TCP window size of the network.

III. PROPOSED APPROACH

We propose a method divided into three steps: a first step of partitioning the original mesh, generating several fragments of the original mesh; and a second pre-simplification of the mesh partitioned, generating simplified models of fragments of different levels of detail; the final step involves the actual visualization of an area which interest the client, the latter having the possibility to visualize more accurately the area of interest, and less accurately the areas out of context. Figure 1 presents our architecture approach.

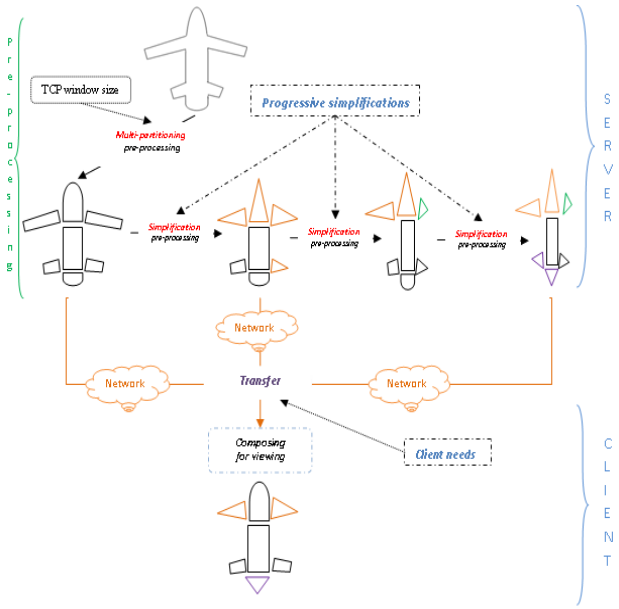

Figure 1 : architecture of proposed approach.

A. *Partitioning*

Mesh partitioning, also sometimes calls mesh segmentation or mesh decomposition, has became an essential problem in many domains like modelling [10], simplification [6], skeleton extraction [15], compression [14]. Our definition about mesh partition is therefore:

Mesh partitioning $\aleph$: Let M be a 3D boundary-mesh, and E the set of faces of the mesh M. A partitioning $\aleph$ of M is the set of sub-meshes $\aleph = \{M_0, M_1, ..., M_{p-1}\}$ induced by the partition of E into p sub-sets intersecting pairs. An example of mesh partitioning can be seen in Figure 2 .

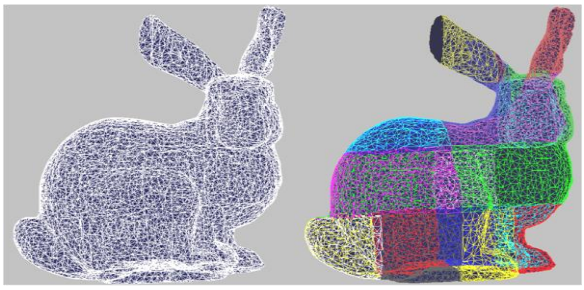

Figure 2 : example of mesh partitioning

All the approaches existing in literature about mesh partitioning do not taking in count the TCP window size of the network, which is a condition in our context, because we work on large meshes transmission through the network. We

propose an approach that based on the region growing method.

The split process that we propose is done on the server, before the client request. Assume that it is only one part of the initial mesh which interests the client. Therefore, it becomes expensive in term of time and memory to transmit the entire object with all the details, even on areas that particularly not interest him. What's more, if we consider that we have an initial mesh M, and that the file which support the mesh M cannot be transferred over the network because of its large size (we note size_file(M) the size of file that support the mesh M), it becomes necessary to partition M to allow to transfer only the region of interest of the client in the necessary details, other parts that can be transferred to lower detail, which would reduce the waiting time of the remote user, and take into account the size of the TCP network. This operation is done on the server, before the client request.

All vertices must be ordered. Let ε be the supposed smaller TCP window size. We consider that we have a mesh *M*, and that the file which support the mesh *M* can not be transferred over the network because of its large size (we note *size_file(M)* the size of file that support the mesh *M*). So, we want to partition M. Let *B* be the bounding box of *M*. We call $s_{i,1\le i\le k}$ the different vertices of *M*, where k is the number of all vertices of *M*. So, we define $S=\{s_1,s_2,...,s_k\}$ as the set of all vertices of the *M*. We also call $t_{i,1\le i\le n}=(s_{i1}s_{i2}s_{i3})$ a triangle *i* formed by three vertices, with n representing the number of all triangles of mesh *M*. $B=\{t_i / s_{i1},s_{i2},s_{i3}\angle B\}$ defines the bounding box of *M* which contains all the triangles of *M* ($s\angle B$ means that is inside or at the border of *B)*.

For splitting our mesh *M*, we first divide B in the middle into two boxes, and we get two new sub-bounding boxes B_1 and B_2, such as $B_1\cup B_2=B$. Axis of partitioning is chosen so that when the threshold for partitioning is not met, the two sub-bounding boxes must contain triangles. So, we obtain two sub-meshes M_1 bounded by B_1 and M_2 bounded by B_2. If $size\_file(M_1)\le\varepsilon$ and $size\_file(M_2)\le\varepsilon$, we stop the partitioning; else we repeat the fragmentation of each B_i *(i=1,2)* until $size\_file(M_i)\le\varepsilon$.

An example of our fragmentation principle in 2D can be visualized in Figure 3.

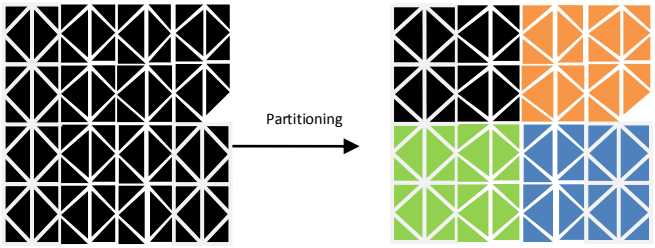

Figure 3 : example of our mesh partitioning in 2D.

So, at the end of our fragmentation, we have:

$size\_file(M_i)\le\varepsilon$ $(1\le i\le p)$, where p represents the number of all last fragments that we have obtained. Now, we are sure that our all fragments files can be transferred over the network. Moreover, this operation of fragmentation is important because it would enable us at some point, depending on client needs, to select only part of the mesh to allow quick viewing (less time consuming).

Quick view is one of the fundamental objectives, so we can also further accelerate the visualization process by displaying only the mesh at the levels of details the client needs. This process of reducing the details of a mesh, introduces what we call mesh simplification.

B. Simplification

The mesh simplification is an operation that aims to define from a base mesh M_1, another resulting mesh M_2, such that $card(M_1)>card(M_2)$, where $Card(M_1)$ and $card(M_2)$ respectively correspond to the number of faces associated with the base mesh M_1 and the resulting mesh M_2.

We need this operation for generating many levels of details for each fragment that we have obtained after partitioning; and fewer details; the size file supporting the mesh is reduced.

We can classify simplifications methods that exist in three mains categories:

- Vertex clustering [18]. This method involves placing a grid of cells on the mesh and merges all the points contained within the same cell. This method is very fast, but the visual appearance of the final mesh is not relatively good. The model topology is not preserved.
- Edge collapse [11] uses an iterative selection of edges to be removed. At each step, the two vertices of the selected edge are joined to form a single vertex.
- Vertex decimation [20], [3], [19]: The idea is to remove a vertex and all faces that using this vertex, and then re-triangulate the hole created. This approach is commonly used when we want to preserve the mesh topology. However, this method preserves topology and is limited to manifold meshes.

In our context, we want to transmit fragments preserving their geometry and topology, even after several levels of simplification. To do this, we opt for using the vertex decimation method, which is appropriate, as we work on manifold meshes.

Our simplification is done on the server, before the client request. We want to simplify our various fragments to generate these fragments with different levels of detail, allowing the client to have a more accurate view of its area of interest and a less precise view of the surrounding areas. We wish to keep the geometry. For this reason, it is important to establish a principle of reducing the number of mesh faces at each stage of simplification. We also want to keep the topology of the mesh. We opted for the vertex decimation method. The selection criterion for removing a vertex at every stage of the decimation is based on the evaluation of discrete

Gaussian curvature of the surface described by the polyhedron around each vertex.

The formula for evaluating the discrete Gaussian curvature is given by: $A = 2\pi - \sum_i \alpha_i$, where α_i represent the angles of adjacent faces to the vertex i. So, at each stage of the vertex decimation, we remove the vertex with the minimum radius of curvature. For preserving the mesh connectivity within the meaning of the fragments, we propose a first approach which consists on not delete the vertices located on the border of the fragment through the simplification process. Figure 4 shows the set of vertices non-removable in this case.

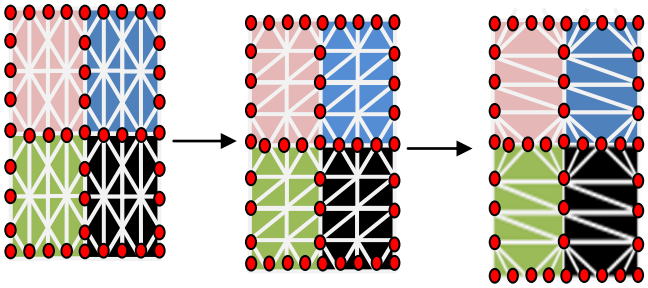

Figure 4 : generation of two levels of detail for the simplification of fragments, seen in 2D. Left, a partitioned mesh (LOD 0); in the center a first simplification of fragments (LOD 1); right, a second simplification of fragments (LOD 2). The points in red represent the non-candidates vertices for deletion throughout the vertex decimation.

This method of simplification preserving the boundaries of the fragments could be an interesting approach that we ensure the faithful reproduction of the semantics of the grid after transfer of the various fragments across the network. However, the fact that it does not allow to include in the process of simplifying the vertices on the border of the fragments leads to a limitation of the rate of simplification of the fragments. More, as we can see in Figure 4, this could generate degenerate triangles. For solving this, we propose another simplification approach, which include the boundaries of the fragments.

We propose to add the following conditions:

1) Each vertex must have a list of integers containing the levels of details of all simplification which it belongs.

2) Let us consider a set of fragments $\Sigma = \{F_i\}_{1 \le i \le p}$, where Fi represent the fragment i and p the number of fragments of the mesh after the partitioning. We also consider $S = \{s_j\}_{1 \le j \le n}$ as the set of all vertices of the mesh, where n represent the number of all vertices of the mesh.

We define the following stages:

a) *Each vertex must have indices of the fragments that contain it.* To do this, we define the application $\phi : \Sigma \to S$ by: $\forall s \in S, \exists \wp_s \subseteq \Sigma / \forall (F_i)_{1 \le i \le |\wp_s|} \in \wp_s, F_i \supset s.$

Notation: $|\wp_s|$ represents the cardinal of $\wp_s$.

b) Our criteria of not removing a vertex on the border of the fragment:

$\forall s \in S$, *s cannot be removed on the border if at least one of the followings is true:*

- ✓ *s is on the border of a fragment with* $|\wp_s| > 2$,
- ✓ *s is an external vertex and* $|\wp_s| \ge 2$,

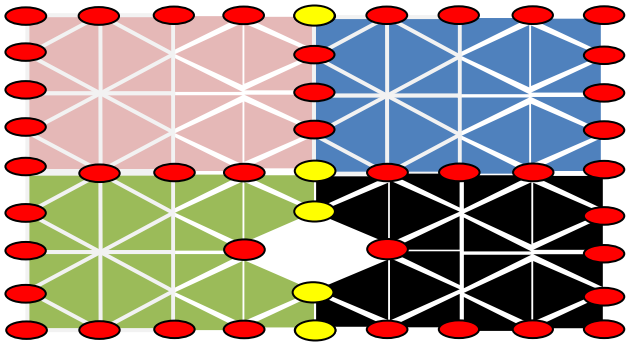

Figure 5 : example of presentation of some non-removable vertices on the border of some fragments. The red points represent the vertices on the border of fragments that can be removed, the yellow points represent the vertex on the border of fragments, but not removable.

c) *Preserving the connection of fragments after simplification.* When we include the vertices located on the border of the fragments while simplifying, the connection between the fragments can be lost. To maintain this connection, it is important to define a relationship between the fragment that we want to simplify and the simplified fragment, including the neighbouring fragments of the fragment to simplify. Let S_{bF} be the set of vertices of the border of the fragment F, and $\wp_s$ is the set of fragments that contain the vertex s.

***Neighbourhood*:** two vertices s_1 and s_2 are called *neighbours* if there is an edge that connects s_1 to s_2. We note $s_1 \Diamond s_2$.

$\forall s \in S_{bF}$ and $\forall F_i \in \wp_s$, if *s can be removed on the border, then* $\forall s_{j(1 \le j \le |\wp|)} / s \Diamond s_j$, if $F_i \supset s$ and $F_j \not\supset s_j$, then add F_i to $\wp_{s_j}$ before removing s.

d) *Creation of the fragment connection between two fragments after simplification. At each step of simplification*

- Create sub-lists of vertices on the border which have at least two indices of fragments in common, and had a removable vertex as common neighbour before removing this vertex,
- Create polygons for each sub-list,
- Triangulate all polygons created.

An example of creating a fragment of connection is shown in Figure 6.

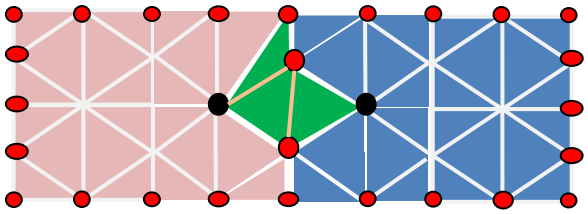

Figure 6 : example of creating a fragment of connection. A simplified model of a fragment in pink color, a simplification of another fragment in blue. Black points represent the new vertices on the border of fragments after simplification, green fragment represent the connection between two fragments pink and blue after simplification.

C. *Real-time visualization*

When a client requests a viewing area of interest, we display his area of interest with the requested details (fine mesh), and others areas out the context being displayed at levels less detailed.

IV. RESULTS

Figure 7 presents a result of partitioning of a hollow pyramid which has 15360 faces initially. The size of initial image is 2186 Ko and she has 8632 vertices.

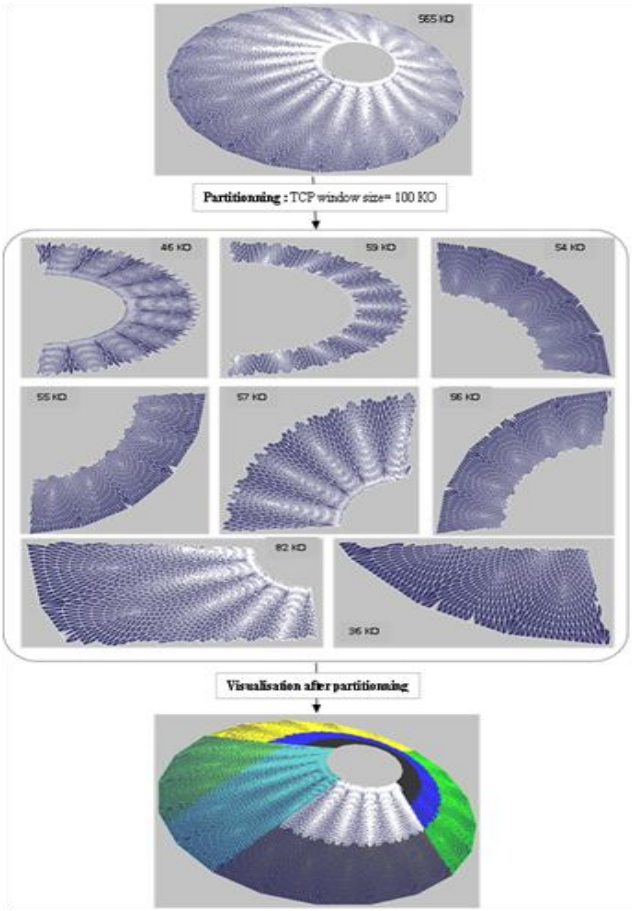

Figure 7 : Example of result of our partitioning approach testing with a hollow pyramid which has 15360 faces initially.

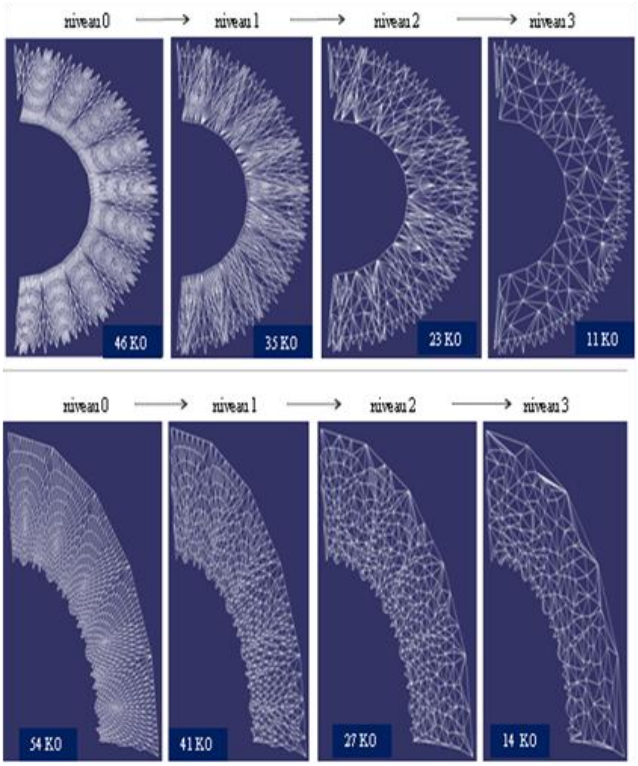

Figure 8 : example of two fragments simplification with maximum radius of curvature = 120°.

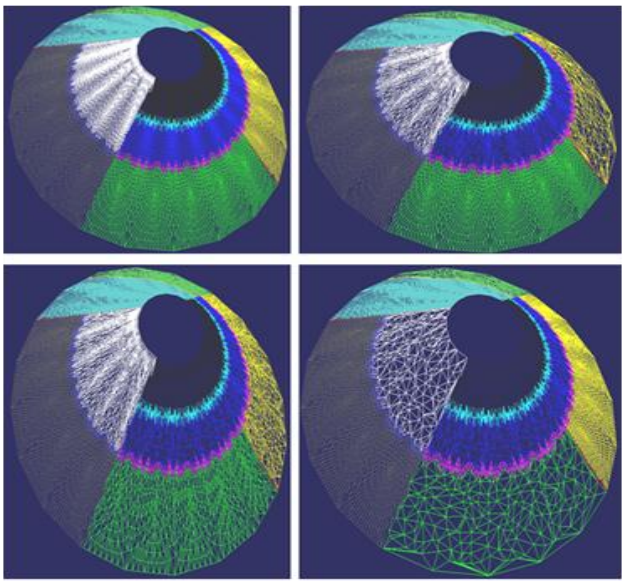

Figure 9 : visualization after the simplification of some fragments and the mesh reconstruction. We can see the different mesh connection of the fragments. The last object in the lower right has 7672 faces and 4992 vertices.

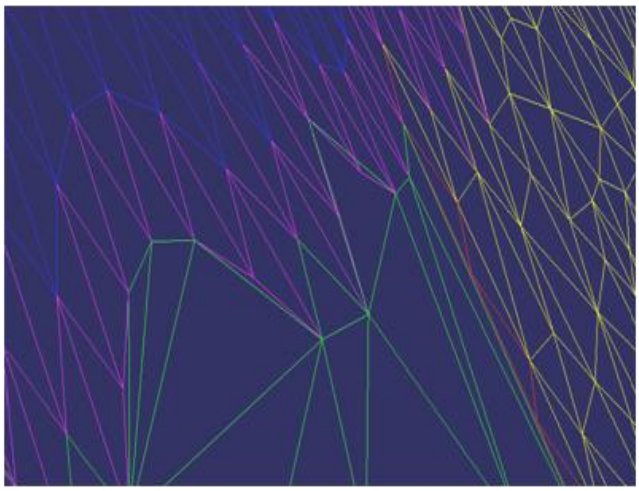

Figure 10 : example of mesh sewing after simplification of fragments with their connection mesh.

TABLE 1. COMPARISON OF DIFFERENT APPROACHES

Methods	**Approaches**				
	[5], [2]	***[21]***	***[16]***	***[9]***	***Proposed approach***
Partitioning pre-processing	✓	✓		✓	✓
Simplification pre-processing			✓	✓	✓
Progressive simplification			✓	✓	✓
Local visualization				✓	✓
Global visualization			✓	✓	✓
Taking into account the TCP window size					✓
Compression	✓	✓			
Decompression	✓	✓			
Local transfer	✓	✓		✓	✓
Global transfer	✓	✓	✓	✓	✓

V. CONCLUSION

In this paper, through a study of approaches to viewing real-time remote existing simplification, we proposed an approach to real-time remote visualization on of large volumes of triangular meshes, taking into account the size of the window TCP network. Our approach is based on a "pre-partitioning" mesh on the server, before any request for viewing an area of interest from the client, which generates fragments of the original mesh depending on the TCP window size of the network. We also generate “pre-simplified” meshes of the partitioned mesh on the server, which could accelerate the time of viewing an area which interest a client, and therefore, reduce the waiting time. We have also proposed an algorithm for connecting different fragments after their simplification.

In the future, assuming for example that the initial model is a coarse level of detail, and the remote user may want a much more detailed mesh. It should therefore, propose a strategy for refinement of our mesh. More, today, in our visualization process, we transmit all the fragments or the part of object that the user wants to visualize. Thus, we want to develop a process which allow, in the process of transmission for viewing, to transmit only vertices or/and triangles that are not yet transmitted. This will reduce more the user waiting time for viewing.

ACKNOWLEDGEMENT

This research study was realized in the framework of the Pestiv-3D project, with the partnership of Arts et Métiers ParisTech, Grooviz, Eurocopter and Cimpa-AirBus.

REFERENCES

[1] Bernardini f., Martin i., Mittleman j., Rushmeier h., and Taubin g. 2002. Building a digital model of Michelangelo’s Florentine Pieta. IEEE Computer Graphics and Applications 22, 1, 59–67.

[2] Clement Courbet, Celine Hudelot. Random Accessible Hierarchical Mesh compression for Interactive Visualization. SGP '09 Proceedings of the Symposium on Geometry Processing. 2009.

[3] Ciampalini, A., Cignoni, P., Montani, C. and Scopigno R. “Multiresolution decimation based on global error”. Technical report, Centre National de la Recherche Scientifique, Paris, France, 1996.

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

[5] S. Choe, J. Kim, H. Lee, Seidel: Random accessible mesh compression using mesh chartification. In IEEE Transactions on Visualization and Computer Graphics (2009).

[6] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. ACM Transactions on Graphics (SIGGRAPH), 23(3):905–914, 2004.

[7] Danovaro, E., De Floriani, L., Magillo, P., Puppo, E., Sobrero, D., 2006. Level-ofdetail for data analysis and exploration: a historical overview and some new perspectives. Computers & Graphics 30 (3), 334–344.

[8] El-Sana J., CHIANG Y.-J.. External memory viewdependent simplification. Computer Graphics Forum 19, 3, 139–150.

[9] El-Sana J., Sokolovsky N., 2003. View-dependent rendering for large polygonal models over networks. International Journal of Image and Graphics 3 (2), 265–290.

[10] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin.Modeling by example. ACM Trans. Graph. (SIGGRAPH), 23(3):652–663, 2004.

[11] Garland, M. and Heckbert, P. “Surface simplification using quadric error metrics”. In SIGGRAPH ’97: 24th annual conference on Computer graphics and interactive techniques, pp. 209–216. 1997.

[12] Hugues Hoppe. 1996. Progressive meshes. In Proc. of ACM SIGGRAPH, 99– 108.

[13] HOPPE, H. 1998. Smooth view-dependent level-of-detail control and its application to terrain rendering. In Visualization’98 Proceedings, 35–42.

[14] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In Proceedings of SIGGRAPH *2000*, pages 279– 286. ACM SIGGRAPH, 2000.

[15] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM* Trans. Graph. SIGGRAPH), 22(3):954–961, 2003.

[16] Kim J., Lee S., Kobbelt L., 2004. View-dependent streaming of progressive meshes. In: 2004 International conference on shape

modeling and applications (SMI 2004). 7–9 June, Genova, Italy. IEEE Computer Society, Silver Spring, MD, pp. 209–220.

[17] Prince C. 2000. Progressive Meshes for Large Models of Arbitrary Topology. Master's thesis, University of Washington.

[18] Rossignac, J. and Borrel, P. "Multiresolution 3d approximations for rendering complex scenes". In B. Falcidieno and T. Kunii, eds, Modeling in Computer Graphics: Methods and Applications, pp. 455–465. Springer- Verlag, 1993.

[19] Schroeder, W. J. "A topology modifying progressive decimation algorithm". In VIS '97: 8th conference on Visualization '97, pp. 205–ff. IEEE Computer Society Press. Los Alamitos, CA, USA. 1997.

[20] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen: Decimation of triangle meshes. In Edwin E. Catmull, editor, Proceedings of the 19th Annual ACM Conference on Computer Graphics and Interactive Techniques, pages 65-70, New York, NY, USA, July 1992. ACM Press.

[21] Yoon S., Lindstrom P.: Random-accessible compressed triangle meshes. IEEE Trans. On Visualization and Computer Graphics 13, 6 (2007), 16–1543.