Measuring Errors for Massive Triangle Meshes

Anis Meftah [1], Arnaud Roquel [2], Frédéric Payan [3], Marc Antonini [4]

I3S laboratory UMR 6070, University of Nice-Sophia Antipolis and CNRS, France

[1] meftah@i3s.unice.fr
[2] roquel@i3s.unice.fr
[3] fpayan@i3s.unice.fr
[4] am@i3s.unice.fr

***Abstract*—Our proposal is a method for computing the distance between two surfaces modeled by massive triangle meshes which can not be both loaded entirely in memory. The method consists in loading at each step a small part of the two meshes and computing the symmetrical distance for these areas. These areas are chosen in such a way as the orthogonal projection, used to compute this distance, have to be in it. For this, one of the two meshes is simplified and then a correspondence between the simplified mesh and the triangles of the input meshes is done. The experiments show that the proposed method is very efficient in terms of memory cost, while producing results comparable to the existent tools for the small and medium size meshes. Moreover, the proposed method enables us to compute the distance for massive meshes.**

I. Introduction

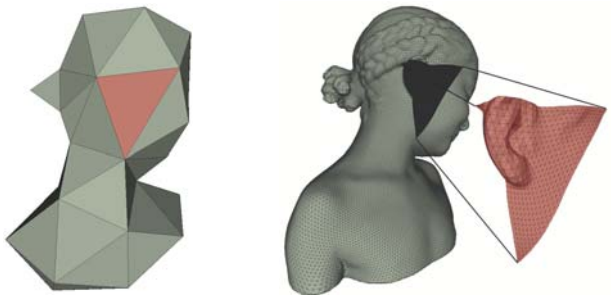
A. Context

The spectacular development of 3D scanners and CAD tools allowed the creation of more and more detailed 3D surface meshes. As an example, the Michelangelo project [1] produced densely sampled meshes from scanned statues. The problem arises when processing and visualizing of these massive data. Indeed, processing such objects requires their global load in memory, which is impossible due to their size. Many works have been proposed to process massive 3D meshes like [2], [3], [4] for compression, [5] for remeshing and [6], [7], [8], [9] for simplification. These treatments produce new meshes but introduce unavoidably a distortion (*remeshing, compression or simplification error*). To estimate such a distortion between the original mesh and the produced one, a metric is necessary, and the surface-to-surface (S2S) distance stimulated by the Hausdorff distance is often used. This S2S distance is one of the simplest approaches to provide a mean square error (MSE)-like measurement for surface meshes, since it allows to compare surfaces represented by meshes with different samplings.

B. Problem statement and proposed method

The most popular tools to estimate this distance are Mesh [10] and Metro [11]. But these programs do not work with massive meshes. This is why only visual comparisons are considered in some papers [5] for evaluating the efficiency of remeshing and compression algorithms.

MMSP'10, October 4-6, 2010, Saint-Malo, France.
978-1-4244-8112-5/10/$26.00



(a) The *reference* mesh B_{ir}. (b) L_i^{ir} extracted from the M_{ir}.

Fig. 1. B_{ir}, L_0 and M_{ir} illustrated on Bimba.

Recently, we proposed an algorithm for the computation of the S2S distance for massive semi-regular meshes [12]. But, this approach works only if one of the two meshes is semi-regularly sampled and if the other one is not massive (which happens in a context of semi-regular remeshing). This previous method loaded entirely the irregular mesh on memory and exploited the relevant multiresolution structure of the semi-regular mesh to load only a small area of it and then compute the distance for this region only.

Our proposal is a solution for computing the S2S distance for all triangle meshes, regardless of their sampling and their size. The idea is to exploit a coarse *reference* mesh for making a correspondence between the triangles of the two densely sampled meshes (by using an orthogonal projection on the coarse level). This reference mesh is obtained by simplifying one of the two input meshes. The indexing step will then enable the computation of the S2S distance on the fly, while having in memory only a small area of the two meshes.

This paper is organized as follows. Section 2 introduces the notations and the S2S. Section 3 presents our proposal. Section 4 shows some experimental results in remeshing and compression domains. Section 5 concludes the paper.

II. Notations and background

A. Notations

Let us call M_{ir} a given irregular mesh, and M_{pr} the corresponding processed mesh (by compression, remeshing or simplification). B_{ir} defines a coarse mesh called the *reference* mesh (see figure 1(a)), obtained by simplifying M_{ir}. Let us also define L_i^{ir} the region of the mesh M_{ir} indexed by the

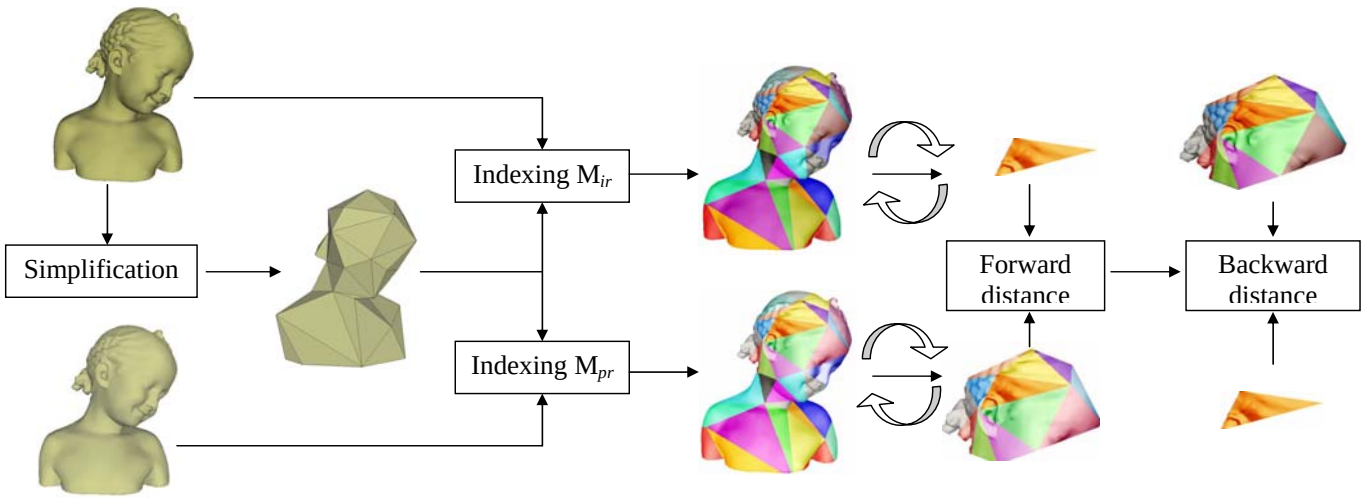


Fig. 2. Global scheme of our method.

i^{th} triangle of B_{ir}, and as well as, L_i^{pr} is the region of the mesh M_{pr} indexed by the i^{th} triangle of B_{ir} (see figure 1(b)). The indexation step will be detailed later. A fixed number of triangles L_i defines a group of triangles (GOT).

B. The surface-to-surface distance

The surface-to-surface distance generally used to evaluate remeshing or compression errors between two meshes depends on the distance between a point q and a set of n points $R = \{r_1, ..., r_n\}$, which is given by

$$d(q, R) = \min_{r \in R} \parallel q - r \parallel_2 . \quad (1)$$

Considering two sets of vertices belonging to two surfaces $Q = \{q_1, ..., q_m\}$ and $R = \{r_1, ..., r_n\}$, using the point-to-surface distance given by (1), we can compute the root mean square error ($RMSE$) between two surfaces defined by:

$$D_{rmse}(Q, R) = \sqrt{\frac{1}{\mid Q \mid} \int \int_{q \in Q} d(q, R)^2 dQ}. \quad (2)$$

$\mid Q \mid$ is the area of Q.

In practice, this distance is not symmetrical, *i.e.*, $D_{rmse}(Q, R) \neq d_{rmse}(R, Q)$. We will refer to $D_{rmse}(Q, R)$ as *forward distance* and to $D_{rmse}(R, Q)$ as *backward distance*. Finally, the so-called *surface-to-surface distance* (S2S) is given by the symmetrical distance D_s :

$$D_s(Q, R) = \max(D_{rmse}(Q, R), D_{rmse}(R, Q)). \quad (3)$$

In our case, the forward error is the difference between M_{ir} and M_{pr}. The backward distance is the difference between the M_{pr} and M_{ir}. For more details please read the paper [10].

As introduced before, the problem arises when this distance is computed for massive meshes. Indeed, the actual tools crash when dealing with massive models because the entire objects have to be loaded in memory.

III. Measuring the S2S Distance for Massive Meshes

The proposed method consists in computing the S2S distance while loading in memory only a small area of the two 3D objects. Figure 2 gives an overview of our method, which contains three principal steps:

- Simplification of M_{ir}: M_{ir} is simplified using QSLIM [13] to obtain the reference mesh B_{ir}. This mesh is used to make a correspondence between M_{ir} and M_{pr};
- Indexation of the triangles of M_{ir} and M_{pr} using B_{ir}: The correspondence is obtained by doing an orthogonal projection of the triangles of M_{ir} and M_{pr} on B_{ir}. The index of the base triangle of B_{ir} which contains the projected point is kept. Hence, we know for any triangle the area of the other mesh that has to be loaded to compute the distance for it.
- Computation of the distance on the fly: the first mesh is loaded L_i by L_i ($i = 0..n$ n being the number of triangles of B_{ir}). For a given L_i the distance is computed using the corresponding region on the second mesh.

A. Indexing

As explained in section II-B, the S2S distance is based on the computation of the minimum distance of a point to a surface. So, if the area where this minimum distance has to be computed is known, only this area would be loaded in memory for its computation. Thus, the goal of the indexing step is to determine approximately, for each triangle, which area has to be loaded for the computation of the S2S distance. For this, the orthogonal projection is done for each vertex belonging to M_{ir} and M_{pr} on the triangles of B_{ir}.

- If one or more orthogonal projected points exist, the point which minimizes the distance to the barycenter of the B_{ir} triangle is kept;
- Else, the distance between the processed vertex of M_{ir} and the barycenters of all the triangles of B_{ir} is com-

puted, and the index of the triangle minimizing this distance is kept.

Figure 2 shows the result of the indexing step on BIMBA using the base level B_{ir} (for more details see [12]).

B. Computation of the distance

Once the indexing step done, the forward distance (from M_{ir} to M_{pr}) is done by loading in memory each L_i^{ir}, one by one. For a given L_i^{ir}, the corresponding region L_i^{pr} plus its one-ring neighborhood [12] are loaded and the point-to-surface distances are estimated. For the backward distance, the inverse process is applied. For each area L_i^{pr}, we use the one-ring neighborhood of L_i^{ir} for the computation of the the point-to-surface distances. To make the process faster, the forward and backward distance are computed simultaneously by loading in memory at the same time the one-ring neighborhoods of L_i^{ir} and L_i^{pr}. This avoids useless hard disk accesses. Finally, the symmetrical distance is computed by applying Eq. (3).

C. General Algorithm

1: Simplify M_{ir} to generate B_{ir} with n triangles
2: **for** each vertex v_i of M_{ir} and M_{pr} **do**
3: Compute its distance to all barycenters of the triangles of B_{ir}
4: Keep the k^{th} nearest barycenters of the triangles of B_{ir} (k being a parameter chosen by the user to accelerate the process. It corresponds to the number of triangles used on the projection)
5: Compute the orthogonal projection on these k triangles
6: **if** there is one or more projected points inside B_{ir} **then**
7: associate to v_i the index of the triangle of B_{ir} which minimizes the distance between v_i and the projected points.
8: **else**
9: associate to v_i the index of the nearest triangle among the k kept triangles.
10: **end if**
11: **end for**
12: **for** $i = 0$ to n **do**
13: Load the GOT composed by the one-ring neighborhood of L_i^{ir}
14: Load the GOT composed by the one-ring neighborhood of L_i^{pr}
15: Compute the forward distance between L_i^{ir} and the one-ring neighborhood of L_i^{pr}
16: Compute the forward distance between L_i^{pr} and the one-ring neighborhood of L_i^{ir}
17: **end for**
18: Compute the symmetrical RMSE distance $D_s(M_{pr}, M_{ir})$ by applying Eq. (3).

IV. EXPERIMENTAL RESULTS

A. Remeshing Errors

Table I compares errors computed with MESH and the proposed method for different remeshing algorithms and different models. HORSE and FELINE are semi-regular meshes remeshed with MAPS [14]. VENUS is remeshed using the normal remeshing algorithm [15]. SCREW is remeshed in our lab [16]. ROCKER-ARM and RAMESSESS5 are remeshed using the TRIREME software [17]. We observe that the errors computed with our method, which loads only a small part of the two meshes (4.8% for SCREWDRIVER and 2.4% for ROCKER-ARM) and MESH are comparable. For all experiments, the bias between the two methods does not exceed 2%. This bias could be eliminated by loading more neighbors, but it will increase the memory requirements. Therefore, we prefer keeping this configuration, which is a good tradeoff between precision and memory cost. Moreover, this bias does not affect the PSNR results as shown later.

Table 2 shows some results obtained with the proposed method for densely sampled meshes: RAMESSESS7, MOULD and BIMBA which are formed by approximately 4 and 7 millions triangles. For these huge meshes the software MESH does not give results because it crashes when trying to load the entire mesh (Out Of Memory error).

B. Compression Errors

The Peak Signal to Noise Ratio (PSNR) is often used in the image coding field to evaluate the coding performances. In geometry compression, the PSNR depends on the S2S distance D_s between a given mesh M and its quantized version $\hat{M}$. This is given by:

$$PSNR = 20 * \log_{10}(\frac{BB}{D_s(M,\hat{M})}),$$

where BB is the length of the Bounding Box diagonal of M.

Fig. 3 shows the PSNR curves for the semi-regular VENUS compressed with the coder of [18]. It proves that the bias between our method and MESH has no impact on the PSNR measurements. Indeed, the two curves obtained respectively by our method and MESH are perfectly superposed.

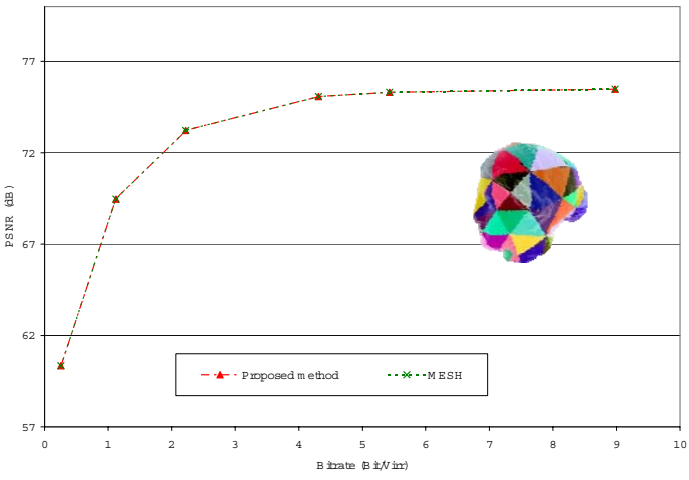


Fig. 3. Measuring compression errors using the proposed method and MESH for VENUS.

Model	Number of triangles	MESH software	Proposed method	Bias	Size of $M_{ir} + M_{pr}$ (Kb)	Memory used (Kb)
VENUS	81,920	$1.679 * 10^{-4}$	$1.679 * 10^{-4}$	0.0006%	6691	363 (5.4%)
HORSE	225,280	$1.004 * 10^{-4}$	$1.024 * 10^{-4}$	1.95%	12374	713 (5.7%)
SCREWDRIVER	286,720	$8.821 * 10^{-5}$	$8.842 * 10^{-4}$	0.23%	14429	695 (4.8%)
RAMESSES5	421,888	$4.410 * 10^{-5}$	$4.410 * 10^{-5}$	0.002%	37365	1783 (4.7%)
ROCKER-ARM	704,512	$6.136 * 10^{-5}$	$6.209 * 10^{-5}$	1.1%	32262	1411 (2.4%)

TABLE I
REMESHING ERRORS COMPUTED WITH MESH AND THE PROPOSED METHOD.

Model	Number of triangles	The proposed method	Max of L_0 neighbors	size of $M_{ir} + M_{pr}$ (Kb)	Needed memory (Kb)	Memory used
MOULD	3,891,200	$5.589 * 10^{-5}$	19	171614	8151	4.75%
BIMBA	4,194,304	$2.439 * 10^{-5}$	16	181857	8774	4.8%
RAMESSES7	6,750,208	$4.409 * 10^{-5}$	18	292692	9268	3.16%

TABLE II
MEASURING ERRORS FOR HUGE MESHES.

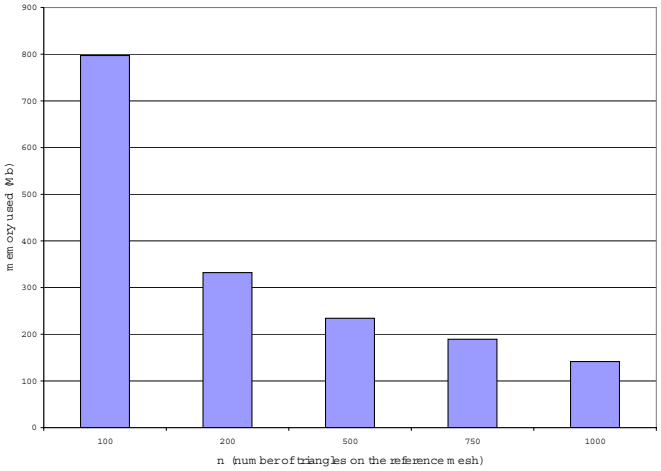


Fig. 4. Maximum size of the memory used during computation *Vs* the number of triangles of B_{ir} (n) for RAMESSES7.

C. Memory Usage

Fig. 4 shows the maximum size of memory used when processing RAMESSES7, for different values of n (number of triangles of B_{ir}). As expected, when n increases, the size of memory decreases. This is due to the fact that the required memory is proportional to the size of L_i which is smaller when n is greater. On the other hand, the processing time is higher since more hard disk accesses are required.
The size S of the loaded data (from the two meshes) is given by the following formula:

$$S = \max_{i=1,\ldots,n}(Nb_i) * size(L_i^{ir}) + \max_{i=1,\ldots,n}(Nb_i) * size(L_i^{pr}),$$

where $\max(Nb_i)$ is the maximum number of the one-ring neighborhood triangles for the n triangles of B_{ir}. To compute the size of a given L_i, the size of geometry G and connectivity T is needed.

$$T = 3 * 4 * N \ bytes,$$

where N is the number of triangles of the given mesh; "4" corresponds to the size in bytes of the index of the vertex coded in *int*.
Assuming that, the number of vertices is the half of the number of triangles, G is equal to:

$$G = 4 * \tfrac{N}{2} \ bytes.$$

"4" corresponds to the size in bytes of *float* (single-precision 32-bit floating).

$$size(L_i) = \tfrac{T+G}{n} = \tfrac{14*N}{n} \ bytes$$

Finally,

$$S = \max_{i=1,\ldots,n}(Nb_i) * \tfrac{14*(N_{ir}+N_{pr})}{n} \ bytes.$$

This function is characteristic of hyperbolic curves: this confirms Fig. 4.

V. CONCLUSION

In this paper we proposed a method to measure the distance between massive 3D meshes. The proposed method consists in computing locally this distance while loading the meshes region by region. First, one of the two meshes is simplified. Then a correspondence between the simplified mesh and the triangles of the input meshes is done. This relationship between the two meshes enables the computation of the error locally by loading only a small part of the two meshes. Experimental results show that the distances computed with our method are comparable to those obtained when loading the entire meshes in memory. The proposed method also able to measure the error for massive meshes up to 7 million triangles with a small memory size which is not possible with the classical tools.

REFERENCES

[1] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital michelangelo project: 3D scanning of large statues," in *Proceedings of ACM SIGGRAPH 2000*, Jul. 2000, pp. 131–144.

[2] M. Isenburg and S. Gumhold, "Out-of-core compression for gigantic polygon meshes," in *SIGGRAPH'03 Conference Proceedings*, 2003.

[3] M. Isenburg and P. Lindstrom, "Streaming meshes," *Visualization Conference, IEEE*, vol. 0, p. 30, 2005.

[4] A. Meftah, M. Antonini, A. Elkefi, and C. Ben Amar, "Low memory cost scan-based wavelet transform for 3D multiresolution meshes using the unlifted butterfly filter," in *International Symposium on Image/Video Communication over fixed and mobile networks*, Tunisie, 2006, IEEE.

[5] M. Ahn, I. Guskov, and S. Lee, "Out-of-core remeshing of large polygonal meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 5, pp. 1221–1228, 2006.

[6] P. Lindstrom, "Out-of-core simplification of large polygonal models," in *SIGGRAPH '00: Proceedings of the conference on Computer graphics and interactive techniques*, USA, 2000.

[7] E. Shaffer and M. Garland, "Efficient adaptive simplification of massive meshes," in *VIS '01: Proceedings of the conference on Visualization '01*. USA: IEEE Computer Society, 2001, pp. 127–134.

[8] M. Isenburg, P. Lindstrom, S. Gumhold, and J. Snoeyink, "Large mesh simplification using processing sequences," in *VIS '03: Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, USA, 2003.

[9] J. Wu and L. Kobbelt, "A stream algorithm for the decimation of massive meshes," in *Graphics Interface*, 06 2003, p. 185192.

[10] N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "Mesh: Measuring errors between surfaces using the hausdorff distance," in *Proc. of the IEEE International Conference in Multimedia and Expo (ICME) 2002*, vol. 1, Lausanne, Switzerland, Aug. 2002, pp. 705–708.

[11] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, 1998.

[12] A. Roquel, A. Meftah, F. Payan, and M. Antonini, "Measuring errors for huge semi-regular meshes," in *Proceedings of IS-T / SPIE Electronic Imaging "Three-Dimensional Image Processing (3DIP) and Applications"*, vol. 7526, no. 1. SPIE, 2010.

[13] M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, USA, 1997.

[14] A. W. F. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin, "Maps: multiresolution adaptive parameterization of surfaces," in *SIGGRAPH '98: Proceedings of the conference on Computer graphics and interactive techniques*. USA: ACM, 1998.

[15] I. Guskov, K. Vidimče, W. Sweldens, and P. Schröder, "Normal meshes," in *SIGGRAPH '00: Proceedings of the conference on Computer graphics and interactive techniques*, USA, 2000, pp. 95–102.

[16] A. Kammoun, F. Payan, and M. Antonini, "A feature-preserving remeshing scheme for surface meshes," I3S-CNRS, Equipe IMAGES, Pole SIS, Tech. Rep. I3S/RR-2009-03-FR, March 2009.

[17] I. Guskov, "Manifold-based approach to semi-regular remeshing," *Graphical Models*, vol. 69, no. 1, pp. 1–18, 2007.

[18] F. Payan and M. Antonini, "An efficient bit allocation for compressing normal meshes with an error-driven quantization," *Computer-Aided Geometric Design, Special Issue on Geometric Mesh Processing*, vol. 22, pp. 466–486, July 2005.