3D SCAN-BASED WAVELET TRANSFORM FOR MULTIRESOLUTION MESHES

Akram ELKEFI [1,2] *, Marc ANTONINI* [1] *and Chokri BEN AMAR* [2]
Elkefi@i3s.unice.fr, am@i3s.unice.fr and Chokri.BenAmar@enis.rnu.tn
[1] I3S Laboratory, CNRS UMR 6070 and University of Nice-Sophia Antipolis, Sophia Antipolis, France
[2] REGIM Laboratory Research Unit 01/UR/11-02 and Ecole Nationale d'Ingénieurs de Sfax,Tunisia.

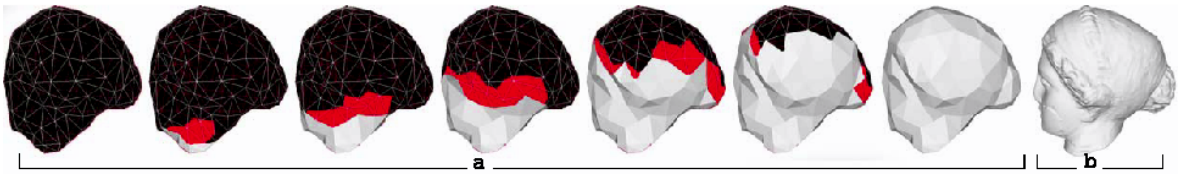

Figure 1: Visualisation of 3D Scan-based Wavelet Transform for Multiresolution Meshes: (a) The spiral acquisition sends a triangle and then its neighbours to initialize the transform. The next steps correspond to the spiral acquisition evolution. (b) Reconstruction with all resolution details.

ABSTRACT

In this paper we introduce a method allowing to perform a Scan-based Wavelet Transform of 3D semi-regular multiresolution meshes. This method consists in processing the datas progressively during the acquisition while reducing considerably memory usage. The experiments show that the method is very effective in terms of memory usage and access. Moreover the proposed algorithm allows to reduce the complexity of processing from $O(N^2)$ to $O(N)$.

1. INTRODUCTION

Meshes are a powerful tool to model complex 3D objects thanks to their double geometrical and combinatory nature (positions of vertices and connectivity). Although many alternatives exist for surface modelling, meshes are omnipresent today and considerable efforts are put in developing for the digital processing of geometry using primarily triangular meshes [7, 9]. The problem of Scan-based processing arises when compressing very large volumes of data using a minimum of memory resources [2, 5, 8]. Indeed, the wavelet transform of the 3D multiresolution meshes requires the acquisition and the complete loading of the object in memory before its processing. Knowing that the 3D meshes with a high degree of precision have enormous sizes exceeding several million points and that the processing systems have limited memory sizes, the difficulty of processing quickly arises related to this kind of data. In this paper, we are interested in compression using wavelets of very large meshes progressively during their acquisition. The wavelet transform of a signal is obtained by a convolution of the input signal with one or more finite length filters. The fact that these filters have a compact support makes it possible to calculate the wavelet coefficients using a limited number of signal samples. The calculation methods of the Scan-based wavelet transform use this property to calculate the high and low frequencies coefficients of the transform using a minimal memory quantity. Here, we propose a calculation algorithm of the Scan-based transform implemented by a lifting scheme. This method consists in carrying out a local processing of the object according to the considered 3D acquisition mode while forcing the memory cost to be minimal. The generated wavelet transform must be identical to the one we would obtain if we had the whole 3D object present in memory at once.

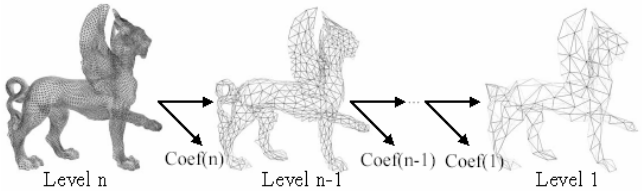

Figure 2: Example of multiresolution analysis on 3D semi-regular mesh. Wavelet coefficients represent the geometrical details.

2. WAVELET ON 3D MESHES

2.1 Multiresolution analysis

The main idea of multiresolution analysis on 3D semi-regular mesh [Fig. 2] is the hierarchical representation of the mesh by a low resolution sub-band and a collection of geometrical details called the wavelet coefficients, necessary to find the original function. The wavelet transform, known for its strong decorrelation, is easier to implement by using a lifting scheme. Indeed, calculation is fast, effective and the inverse transform is directly deduced. The computation of the wavelet transform of the signal requires a certain number of filters calculated from the lifting scheme [3]. The input signal is subsampled and divided into k sets disjoined by means of a polyphase transform, k depending on the selected filter. For each set a filter is applied. These sub-sampled signals are predicted and then updated to reduce the difference between the original function and its predictions. For a multiresolution mesh, the application of the Lifting scheme is repeated $n-1$ times successively on the resulting signal, n being the resolution level. To rebuild the original signal, the signs and the order of the operations must be reversed [Fig. 3].

2.2 Butterfly Filter

2.2.1 Filter implementation

The signal entering the lifting scheme is divided into 4 channels. Each channel is filtered by the corresponding filter. The prediction of a point is a balanced sum of a certain vicinity quickly calculated on the whole set of triangles available at that time.

To apply the filter associated with channel i [Fig. 4], it should be centred on the point of the same channel type following the orientation axes n_1 and n_2. The topology of the object being random, the filter must follow the shape of the object while sticking to the mesh and being centred on the treated point. The extraction of the filtering points is possible thanks to a mesh scanning method using neighbours based on the vicinity knowledge.

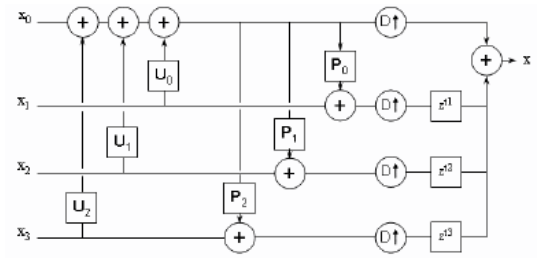

Figure 3: 4-channels Butterfly filter Lifting Scheme.

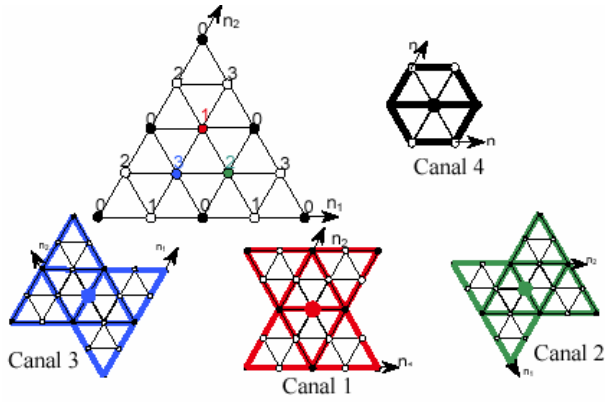

Figure 4: On each type of vertex, we apply the corresponding filter, as shown in Butterfly Lifting scheme.

2.2.2 Vicinity calculation

The step of the vicinity calculation is very expensive in number of operations. Indeed, when searching for all the neighbours of each triangle, two overlapping loops of length *N* are necessary, *N* being the number of triangles of a mesh. However, the mesh of a medium resolution object consists of more than one half a million triangles. We estimated that the number of operations necessary to determine all the vicinity of one triangle in the whole grid is *O=60* floating operations on average, and according to *BM2003*, one of the best benchmarking software, a *2Ghz Pentium IV* with *512 Mo* of Rambus memory cadenced at *800 MHz*, by neglecting the delay of memory access, would theoretically perform under optimal conditions, a maximum of $P_{IV}=10^9$ *Flops* (floating point operations per second). The function of vicinity would require at least four hours = $(N^2*O)/(P_{IV}*3600)$. Our procedure of vicinity search performs this calculation in less than one second, that is to say 10^4 times faster. Our method reduces this complexity from $O(N^2)$ to *O(N)*, (see[4] for more details).

Indeed, instead of seeking the neighbours in the whole grid, we use a procedure that determines the smallest surface containing them. This procedure is based on the following properties:

P1- The neighbours of a central triangle T_c are among the sons of the father of T_c.

P2- Among the sons of the father of a triangle *T*, only the one which is central is a neighbour. The other neighbours of *T* are among the sons of the neighbours of the father of *T*.

P3- If two triangles T_1 and T_2 are close then one of them is central. This knowledge facilitates the search of the other neighbours according to propriety 1.

P4- Two close triangles certainly have a common relative on a higher level.

We use this hierarchisation when acquiring the object by adopting a data structure of a tree with return where all the branches point towards others to take into account the topology, the vicinity and the relationship at the same time. Indeed, a triangle present in memory must point to his father, sons, neighbours, level of resolution and state (see 3.3). We adopt an architecture of four dimensions pointers to satisfy all these constraints. This data structure gathers the concept of multiresolution and vicinity and it allows us a direct access to an element of the grid without the expensive procedure of the course and search in a tree: Let T_1, T_2, T_3 and T_4 be four sons of T_5. If T_5 belongs to a level *i* of resolution then $T_1,..,T_4$ belong to the level *i+1*. We store T_5 in the position *[Level=i][Number=j]* and $T_1,..,T_4$ in the positions *[i+1][j+0..4]* respectively. Thereafter the son of a triangle *[i][j]* are *{[i+1][4j]..[i+1][4j+3]}* and the father of a triangle *[i][j]* is *[i-1][E(j/4)]*, with *E(j)* is the integer part of *j*. An interval of search for the vicinity of *[i][j]* is *{ [i][(4E(j/4))+4] ... [i][(4E(j/4))+16] }*, which is less than *12* triangles to be tested instead of *N*.

3. Implementation of the Scan-based transform

3.1 Problems

To be able to process an object during its acquisition, it is necessary to divide it into regions and send a particular region whenever it is necessary for the transform. In the case of *2D* images, the decomposition of an object is simple. It is sufficient to divide the image into lines or blocks. For video, the unit is a whole image [2]. In these two cases, division is more obvious than for a *3D* semi-regular object.

The idea we propose in the case of semi- regular meshes is to use a low frequency triangle as a unit data. Therefore, it is necessary to have a tool allowing an oriented displacement on the irregular mesh so that a complete scan of the entire object does not leave untreated parts, regardless of the topology of the object. Here, we propose to use a spiral scanning (see 3.2). The choice of the scanning way influences directly the maximum size of used memory.

3.2 Spiral acquisition

When applying the Scan-based filtering, we noted that the method of spiral acquisition [Fig. 5] is the least expensive in memory. It ensures at any time t, the availability of the whole vicinity of a triangle partially processed, necessary to finish its processing and to remove it from the memory. To allow the detection of a path on the mesh, we build a list of marks which are used as references and which are updated during the processing.

The first step is to follow a reference list L_0 = {a, c, b} [Fig.5], to acquire triangles A_i, B_i and C_i, coloured respectively in Yellow (neighbours of vertex *a*), Green (vertex *c*) and Blue (vertex *b*). This represents a full round of initialization. When a new triangle is detected, we send it to the filtering procedure and we update the list of references. The second round has the newly created list $L_1=\{r_0, r_1,.., r_i\}$ as reference. Repeating the previous step, we process the whole mesh with the desired spiral approach. The end of the transmission corresponds to a new list of empty reference. The real case has a large number of particular cases, considering

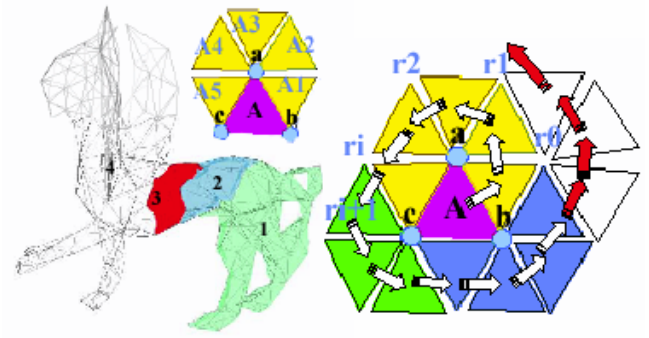

Figure 5: The mesh scanning method follows a spiral path to process the whole mesh with the desired approach.

the irregularity of processed surfaces. We developed a structure of "*states*" for managing these cases, thus trying to overcome the problem by detecting the current configuration and its associated procedure of resolution.

3.3 Wavelet transform method

3.3.1 Different steps

The diagram of *Fig. 6* presents the evolution of the system in time. The four blocks represent the essential functions of the method:

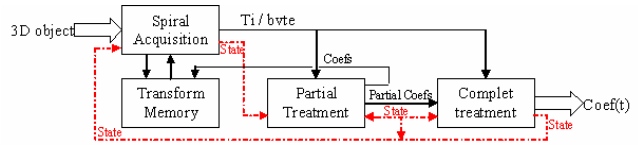

Figure 6: Proposed transform scheme.

- Spiral Acquisition: The input being the 3D object, spiral acquisition is responsible for sending triangles to the other blocks. The output of the method is strongly related to data gathering manner.
- Memory: It is the memory reserved by the operating system for the transform. This block is continuously communicating with the two remaining blocks.
- Partial Processing: The mesh being multiresolution, it is necessary to apply the filter to each level of resolution independently. The complete processing of a low frequency triangle implies the complete processing of all of its tree of descents, but its partial processing implies the complete processing of all of its descents except those on the edges of the processed triangle. These *"edge triangles"* will be stored waiting for all the levels of resolution until the reception of the necessary neighbour to supplement their processing. The partial processing is an intermediate phase. Indeed, to be able to entirely compute the wavelet transform of a triangle, its vicinity should be known, which is not possible because we do not always have the necessary vertices in memory. Thus, a partial phase is necessary to benefit from the presence of some neighbours of a triangle being processed, by making precalculations. Each partially processed triangle is marked *"Partial State"* and then kept in memory on "*Standby"* for the continuation of its processing. Each time a new triangle is received in memory, it starts again the processing of the concerned partially process triangle and it begins its own partial transform.
- Total Processing: When a triangle is not on standby any more, it is send to the block of total processing which completes its transform, sends its coefficients and then erases it from memory. When all the vicinity of a triangle is present in memory, the complete processing unit computes the transform directly.

3.3.2 Steps Cooperation

Considering that every partially processed triangle is kept in memory until the end of its processing, we should acquire needed triangles in priority to complete the processing of those marked *"Partial State"* in order to avoid increasing of memory usage. The method of spiral reading suggested in paragraph *3.2* ensures that each time a triangle is acquired, the processing of at least a triangle is completed and the corresponding triangle is thus removed from memory. We think that this method influences directly the size of the buffer and is effective to reduce the maximum memory usage. Indeed, our method ensures that at each spiral scanning a certain number of triangles is completely processed and erased from memory and that the new ones are partially processed.

The acquisition of a triangle starts the calculation of the vicinity, the total transform if possible, the update of the partial coefficients of the other triangles and the erasure of the triangle if it is completely processed. The calculation of the vicinity must take into account the modification of the structure of the object kept in memory after each acquisition. The triangles at the edges are processed separately, and every information of the vicinity which is not necessary after processing, partial or total, must be erased to optimize memory cost. The used data structure must separate each type of triangles to allow the possibility of freeing memory.

3.3.3 Memory State

Figure [5] represents an example of the state of the memory during a step of the transform: The part in black *(4)* is not scanned yet, the one in green *(1)* is already treated and erased from memory. The blue *(2)* is in the course of total processing and will be erased before the following spiral turn. The red part *(3) is* partially processed and kept in memory. It is necessary for the processing of *(2)*. At any time, only *(3)* is kept in memory, as opposed to the traditional method which requires the presence of the totality of the object in memory before its processing.

3.3.4 Multiresolution treatment

The filtering starts with the prediction of the higher resolution level, then the update of the coarser level and so on. If a detailed level triangle is in a waiting state, it implies the setting on standby of all the triangles which are in relation with it. This must be marked and managed so that the total processing of a triangle starts the resolution of all the triangles waiting for it, otherwise, the process remains blocked. These releases are ensured by the variable *"State"*.

4. MEMORY GAIN

The semi-regular mesh of a medium resolution object consists of more than one million triangles out of which a few hundred represent the low frequency coarse mesh. This coarse mesh gives us an idea of the general shape. Let N be the number of triangles of this coarse mesh, P its number of points and M_i the number of triangles present in memory while processing a given point i. At the beginning of the processing no triangle is present in memory and at the end, all the triangles were processed and erased from memory $(M_0=0$ *and* $M_P=0)$. Let N_{max} be the maximum number of triangles contiguous in one point of the mesh. When processing a point P_i, the memory contains M_i triangles. The following iteration corresponds to an addition of I_{i+1} triangles with $I_{i+1} \in \{0,...,N_{max}\}$ and of a removal of O_{i+1} triangles with $O_{i+1} \in \{1,...,N_{max}-3\}$ triangles, for $i \in \{1,...,P\}$. The sum of all the additions of triangles is necessarily equal to the total number of triangles and similarly for the removal. The maximum value M_{max} that can reach M_i represents the maximum size of the memory used for the processing of the object. A first result is that $M_{max}<E[M]+2\,\sigma_M$ with E[M] the average of M_i and σ_M its standard deviation. It is then possible to write that:

$$M_i = \sum_{j=1}^{i}(I_j - O_j) < M_{\max},\ for\ i \in 0..P \qquad (1)$$

$D_j=(I_j-O_j)$ represents the modification of the system after each processing of a point, then $D_j \in X=\{3-N_{max},..,N_{max}\}$. In experiments, it is possible to notice that D_j, I_j and O_j follow the laws of probability described by the curves of figure *7*. In the worst of the cases, M_i reaches its maximum when the I_i 's are arranged in a descending order and the O_i 's in ascending order. Thus, the maximum will be reached when D_i is first negative and M_i is first decreasing. This case has a very low probability but it represents the worst case. Thus,

$$M_{\max} = \sum_{j=1}^{\max} D_j \quad with\ D_j > 0 \qquad (2)$$

Let $Pr(D_j)$ be the probability that the difference between the ad-

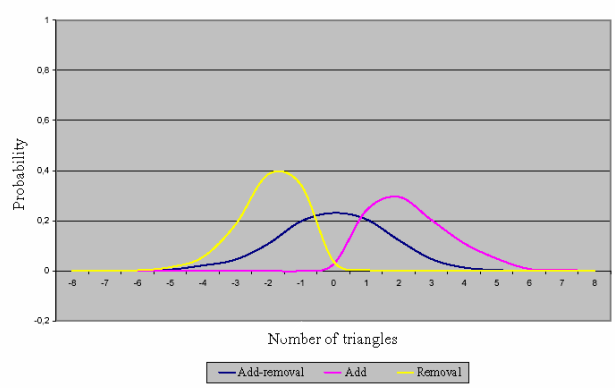

Figure 7: Probability curve Pr(d) of addition/removal estimated on typical 3D objects.

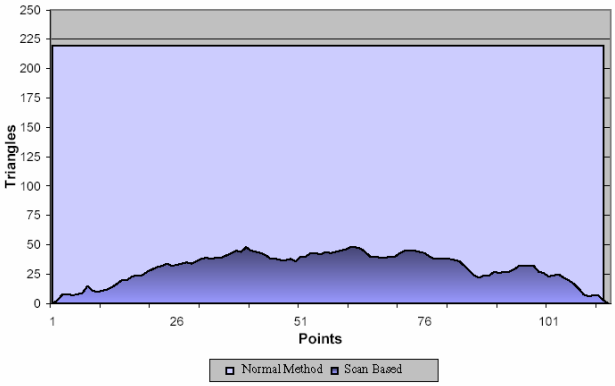

Figure 8: Memory evolution using the Scan-based transformation on horse object compared to the normal method.

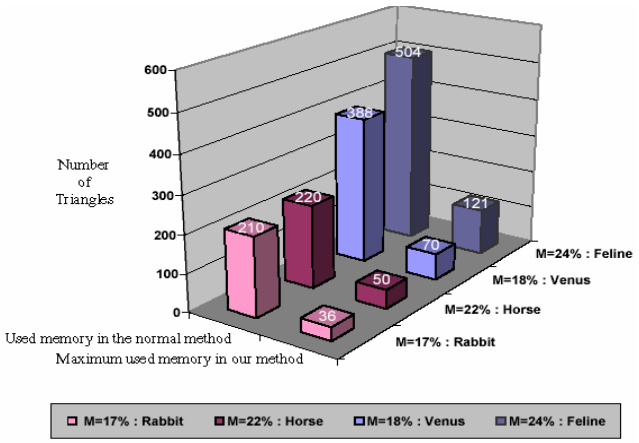

Figure 9: Comparison between the normal transform and our transform, in term of memory usage for various objects.

Figure 10: FELINE, an example of a 3D Multiresolution object.

dition I_i and the removal O_i at iteration j is equal to D_j triangles. This probability follows the curve of figure 7. Then we can write:

$$M_{\max} = P \sum_{j=1}^{N_{\max}} D_j \ \Pr(D_j) \ \ with \ D_j \in X^+ \qquad (3)$$

Considering these probabilities, we built a synthetic object of *100* points and *196* coarse triangles. Equation *(3)* gives us $M_{max}=66$ triangles, which is *33.7 %* of the memory used with the traditional transform. Thus *33.7 %* of the mesh represent a maximum of necessary memory to process any object. This maximum represents an extreme case. We tested the algorithm on several objects which include the majority of the cases of the possible surface irregularities: *Figure 8* represents the evolution of memory usage during the processing of the horse object and *Figure 9* represents a comparison between the normal transform and the proposed method, in term of memory usage for various objects. In practice, we found an average of *20.25 %* of the object size.

5. Conclusion

In this paper we introduce a method allowing to compute the Scan-based wavelet transform of *3D* semi-regular multiresolution meshes. This method consists in processing the data progressively during the acquisition while reducing considerably the memory usage. The computed wavelet transform is identical to the one obtained if we had the knowledge of the totality of the *3D* object. The experiments show that our method is very effective in term of memory cost and access; about *20.25 %* of the size of the object and under a maximum of *33.7 %.* Moreover the proposed algorithm allows to reduce the complexity of the processing from $O(N^2)$ to $O(N)$.

References

[1] F.Payan and M. Antonini *"Multiresolution 3D mesh Compression."*,IEEE ICIP Semptembre 2002,Rochester, New York

[2] C.Parisot,M. Antonini and M. Barlaud *"3D Scan-based wavelet transform for video coding."* , EURASIP, Applied Signal Processing, Multimedia Signal Processing,2003, pages 521-528.

[3] A.Khodakvosky, P.Schroder and W. Sweldens *"Progressive geometry compression"*, SIGGRAPH'2000, Computer Graphics Proceedings, ACM Press/ACM SIGGRAPH / Addison Wesley Longman, Pages 271-278

[4] A.Elkefi, M. Antonini and C. BenAmar *"Compression de maillages 3D multirésolution"* Rapport Interne I3S number ISRN I3S/RR-2003-30-FR, November 2003.

[5] M.Isenburg and S.Gumholh *"Out-of-core compression for gigantic meshes"*, SIG-GRAPH, SIGGRAPH'03,2003,Pages 1-8

[6] H.Hoppe *"progressive meshes"*, Proceedings ACM SIGGRAPH'96, August 1996, Pages 99-108.

[7] P.Alliez and C.Gotsman *"Recent Advances in Compression of 3D Meshes"*, Pro-ceedings of the Symposium on Multiresolution in Geometric Modeling, 2003

[8] C.Chrysafis and A.Ortega *"Line Based Reduced Memory Wavelet Image Compression"*, IEEE Transactions on Image Processing, march 2000, pages 378-389

[9] P. Alliez and M. Desbrum *"Valence-Driven Connectivity for 3D Meshes"*, C.EuroGraphics'01, Volume 20(2001), Number3