Noname manuscript No.
(will be inserted by the editor)

Joint Reversible Watermarking and Progressive Compression of 3D Meshes

Ho LEE · Çağatay DIKICI · Guillaume LAVOUÉ · Florent DUPONT

Received: 16 Mar 2011

Abstract A new reversible 3D mesh watermarking scheme is proposed in conjunction with progressive compression. Progressive 3D mesh compression permits a progressive refinement of the model from a coarse to a fine representation by using different levels of detail (LoDs). A reversible watermark is embedded into all refinement levels such that (1) the refinement levels are copyright protected, and (2) an authorized user is able to reconstruct the original 3D model after watermark extraction, hence reversible. The progressive compression considers a connectivity-driven algorithm to choose the vertices that are to be refined for each LoD. The proposed watermarking algorithm modifies the geometry information of these vertices based on histogram bin shifting technique. An authorized user can extract the watermark in each LoD and recover the original 3D mesh, while an unauthorized user which has access to the decompression algorithm can only reconstruct a distorted version of the 3D model. Experimental results show that the proposed method is robust to several attack scenarios while maintaining a good compression ratio.

Keywords Reversible watermarking · 3D-mesh · Compression

H. Lee · F. Dupont
Université de Lyon, CNRS, Université Lyon 1, LIRIS, UMR5205, F-69622, France
E-mail: {name.surname}@liris.cnrs.fr

Ç. Dikici · G. Lavoué
Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621, France
E-mail: {name.surname}@liris.cnrs.fr

1 Introduction

The use of 3D content, mainly in the form of 3D meshes, has become widespread in numerous applications: computer aided design, scientific simulation or entertainment (video games, 3D movies). The increasing needs of precision and realism lead to an important increase in the complexity and size of these 3D data; besides, more and more applications now aim to run on very heterogeneous devices from high performance workstations to smart phones, and need to transmit these data over heterogeneous networks (including low bandwidth channels) to provide a remote access. For instance, in most of scientific visualization applications, the 3D data are produced in high performance computing centers and then analyzed remotely by teams of scientists and engineers who are geographically distant and who may have heterogeneous visualization devices. In such scenarios, the main bottlenecks are: (1) how to transmit efficiently this 3D content, (2) how to adapt the 3D content to the graphic capabilities of the user device, (3) how to protect this content during its transmission, and (4) how to reconstruct the original 3D content after the protection phase at the receiver.
Issues (1) and (2) can be resolved by the use of progressive compression techniques [16]; indeed, progressive compression allows to achieve high compression ratio (and thus fast transmission) and also to produce different levels of detail (LoD), allowing to adapt the complexity of the data to the remote device by stopping the transmission when a sufficient LoD is reached. The main idea is to transmit a simple coarse mesh (low-resolution), and a refinement sequence permitting to reconstruct incrementally the mesh during the transmission. Most of progressive compression techniques, such as [1], consist in iteratively decimating the mesh (vertex/edge suppressions), while storing the information

necessary to the process inversion, i.e. the refinement (vertex/edge insertions during the decompression).
Issue (3) concerns the intellectual property protection problem, indeed during its transmission or visualization the 3D content can be duplicated and redistributed by a pirate. Digital watermarking is considered as a good solution to this emerging problem [21]; it consists in hiding a secret information (the watermark) in the 3D content itself, usually by slightly modifying its geometry. Issue (4) can be resolved by the use of *reversible watermarking* scheme, where the slight modification introduced by watermarking is perfectly restored at the receiver side [24].
The main problem is how to combine compression and reversible watermarking? A simple solution consists in firstly inserting the watermark into the 3D model and secondly compressing it; however since the mesh is transmitted in a progressive way until a certain level of detail (not necessary the finest one), the mark have to be readable in all intermediate LoDs. In other words, the mark have to be robust to the iterative simplifications processed during the progressive encoding, unfortunately it is often not the case; to illustrate that, we have watermarked two 3D meshes, Horse (113K vertices) and Dragon (50K vertices), with the moment-based blind scheme from Wang et al. [22] which has shown to be one of the most robust techniques among the state of the art especially against simplification. Figure 1 illustrates the results of the watermark extraction in all levels of detail from the finest (level n) to the coarsest (around 1000 vertices). These levels of details have been obtained using the progressive compression algorithm from Alliez and Desbrun [1]. This figure illustrates clearly that after level $n-3$ (i.e. after 3 decimation iterations) the bit error rate grows very quickly (50% means a random extraction); hence under this scenario (watermarking then compression) the mark will be extractable only in the finest LoDs even when using a very robust technique.

To resolve this issue we propose a joint compression-watermarking scheme where the mark insertion is done within the progressive compression process, hence the watermark extraction can be done at each intermediate level of detail, even the coarsest ones. A *histogram bin*

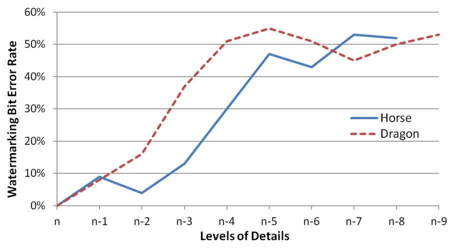

Fig. 1 Robustness of the watermarking algorithm from Wang et al. [22] under the progressive compression from Alliez and Desbrun [1].

shifting technique is used to embed the watermark bits. During the encoding, at each decimation, mesh vertices are separated into two sets (removed and remaining vertices), and a histogram is computed for each set based on the distribution of its vertex norms from the mesh gravity center. We split these histograms into distinct bins and modify these histograms by shifting the bins to embed one bit. To increase the watermark payload (i.e the number of embedded bits), we decompose the mesh into regions so as to insert one bit into each region. Fig. 2 illustrates the global principle of our algorithm. The mesh refinement as well as the watermark extraction are achieved progressively.

This paper is organized as follows: section 2 presents the related works while sections 3 and 4 respectively detail the two components of our joint scheme: progressive compression and reversible watermarking. Section 5 presents extensive experiments regarding payload, distortion, compression ratio and robustness of our joint scheme.

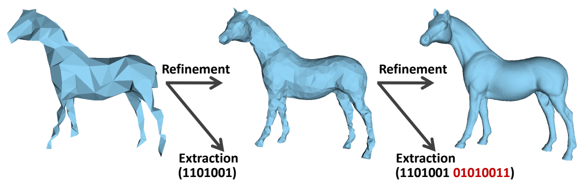

Fig. 2 An example of progressive decoding and watermark extraction.

2 Related Works

2.1 Progressive Compression

The concept of progressive mesh compression was introduced for the first time by Hoppe [9]; in his work, the mesh is iteratively simplified by performing a sequence of edge collapses (two connected vertices are merged together); to reverse these operations during decompression, the positions of the vertices to split are explicitly encoded hence this method is very costly. A bunch of methods were then introduced, like [19,15], to improve the compression ratio by applying the collapse/split operations on sets of independent vertices. Other methods are based on vertex removal, they consist in removing sequences of vertices and re-triangulating the holes left by the deletions at no cost [5,1]. All the progressive algorithms described above are connectivity-driven algorithms, meaning that the priority is given to the connectivity coding. Observing that the mesh's geometry data is larger than the connectivity data, more recent research has focused on geometry-driven algorithms like Gandoin and Devillers [7] and Peng and Kuo [16]; these methods outperform the other in term of lossless compression rate however they provide quite poor visual results at intermediate levels of detail. Very recently

Valette et al. [20] and Peng et al. [17] proposed two new progressive approaches. [20] is based on a reconstruction scheme, the algorithm starts from a coarse version of the original model which is refined progressively by inserting a vertex to the longest edge using edge split operation, aiming to generate uniformly sampled intermediate meshes. The algorithm from Peng et al. [17] aims at carefully preserving the relevant features at each intermediate level of detail using a hierarchical clustering.
Our joint watermarking-compression algorithm is based on the progressive compression from Alliez and Desbrun [1], later improved by Lee et al. [11]; this method has the benefit to be simple to implement, fast and to produce results competitive with the very recent state of the art in term of rate distortion curve. We have to precise moreover that our watermarking method could be integrated to most of the progressive compression schemes presented above.

2.2 3D Watermarking

Whereas the first method was introduced almost 15 years ago [14], 3D mesh watermarking is still in its early stage; indeed, even if a lot of methods have been introduced so far [21] only few of them are both robust (i.e. able to survive malicious attacks on geometry and connectivity) and blind (the original model is not needed at extraction) whereas these two characteristics are critical for copyright protection. Existing robust and blind methods are mostly based on robust shape descriptors: Zafeiriou et al. [27] consider the histogram of the vertex coordinate prediction errors, Cho et al. [3] use the histogram of the vertex norms, Wang et al. [22] rely on the volume moments of the shape and Wang and Hu [23] use the integral invariants. Some other robust methods embed the mark into a transform domain: Konstantinides et al. [10] embed the mark in the spheroidal harmonic coefficients while Liu et al. [12] consider a spectral transform. Praun et al. [18] use the multiresolution decomposition method of Hoppe [9] to find the salient spatial parts and to embed the watermark in these parts.
Our algorithm is blind and robust to geometry attacks; moreover it owns two important supplementary properties regarding these existing works; firstly it is *reversible*, i.e. once the mark is extracted it can be removed to retrieve the original non-distorted shape and secondly it is *joint* with compression (i.e. the mark insertion is embedded into the progressive compression algorithm). The next two subsections detail existing reversible and joint techniques; since there exist very few or even no 3D mesh techniques with these properties, the following subsections will largely focus on existing works for 2D image.

2.3 Reversible Watermarking

In reversible watermarking, the receiver should recover the original object after extracting the embedded watermark [24]. The reversibility is desired where the alteration of original object is not preferred. For 3D meshes, one method is based on prediction-error expansion [25]. Lu [13] presented an alternative reversible method based on vector quantization, where the reversibility is limited with the quantization scheme. Our proposed algorithm is based on histogram shifting of geometric information of LoDs.

2.4 Joint Compression and Watermarking (JCW)

The watermarked content should be compressed before transmission in order to minimize the transmission cost. Separate compression scheme of a watermarked content may (1) decrease the robustness of the watermark, or (2) increases the compression rate for a given distortion. In order to cope with these two inconvenient situations, one recent strategy is to apply watermarking and compression jointly. For the time being, JCW methods have been applied only on 2D data, especially in image watermarking and compression. DCT coefficients of JPEG image are jointly compressed and watermarked by [26], and [8] proposes a JCW method for wavelet coefficients of JPEG-2000. Very recently, Chen and Chen [2] proposed a watermarking scheme applying on a progressive mesh representation obtained through iterative simplification, however they did not apply practically the compression. Hence our method applies such joint compression and watermarking operation for the first time for 3D meshes, moreover the watermarking is reversible.

3 Progressive Compression

Our joint compression-watermarking method is based on the progressive algorithm of Alliez and Desbrun [1], since it is one of the most efficient connectivity-driven algorithm which produces very competitive results comparing to the most recent algorithms.

In this algorithm, levels of detail (LoDs) are generated by applying successively a pair of conquests which remove a set of vertices : decimation and cleaning conquests. Decimation conquest traverses the mesh in a deterministic way by means of patch-based traversal. A patch is a set of facets around a vertex. When the center vertex of the current patch has valence code inferior to 6, this vertex is removed and the patch is re-triangulated. The deterministic patch re-triangulation allows to preserve as mush as possible the regularity. In the same way, cleaning conquest removes vertices of valence 3. Fig. 3 illustrates the decimation and the cleaning conquest applied on a regular mesh.

For the geometry coding, firstly mesh vertices are uniformly quantized. Then, the authors use a local prediction which is the barycentric prediction. When a vertex is removed, the average position of its neighboring vertices is computed to predict its position and the difference is encoded. This prediction method performs particularly well for smooth surfaces. To further reduce geometry bit rates, they adopt also a local coordinates system to separate tangential and normal components. This geometry coding has been later improved by Lee et al. [11].

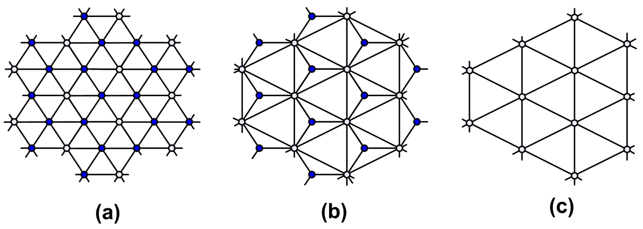

Fig. 3 A regular mesh (a) is simplified by the decimation conquest (b) and the cleaning conquest (c) in the algorithm [1].

4 Watermarking Method

To insert the watermark bits within the progressive compression algorithm, we consider a reversible watermarking technique. This kind of technique is often used in scenarios where the integrity of the original has to be guaranteed.

In a progressive compression approach, the model is successively reconstructed at the decoding side. Thus, it is proper and natural to extract also progressively the watermark instead of waiting the full transmission. This progressive extraction can be realized through a repeated watermark insertion into each level of detail during the encoding process.

However the iterative use of classical watermarking methods would deform increasingly and severely the mesh shape, presenting no interest to the user. This issue forces us to combine progressive compression algorithm with a reversible watermarking method, which permits to remove the deformation caused by the watermark insertion and to restore in a lossless way the input mesh at the end of transmission.

Among various existing reversible watermarking methods, we adopt the *histogram bin shifting* technique.

4.1 Watermark Primitive Selection

The objective of histogram bin shifting techniques is to insert the watermark bits by constructing two histograms of a characteristic feature presenting similar distributions and then by shifting slightly these histograms. The initial similarity of these histograms is crucial for this technique since the watermark insertion is possible only when the two histograms present a very close distribution. For instance, in [6], De Vleeschouwer et al. divide 2D image pixels into two pseudorandom sets and use luminance histograms.

To construct such histograms, two difficulties arise in our case: (1) a 3D mesh feature has to be carefully chosen in order to construct similar histograms even for the coarsest levels of detail and also to achieve a good robustness, and (2) 3D mesh has intrinsically irregular topological structure, thus classifying the mesh elements into two random-like sets is more difficult than in the case of 2D images.

To resolve the first issue, firstly we assume that the selected feature has to be related intrinsically to the shape of the 3D mesh. This is the reason why we choose to use distribution of distance between mesh vertices and mesh center as in the work of Cho et al. [3]. The second issue consists in separating the mesh vertices in a deterministic way so that the two sets S_1 and S_2 have the same configuration of vertices for both the embedding and the extraction. In addition, to obtain a good similarity between the two histograms, it is necessary to insure that vertices from the two sets are well spread over the mesh surface. As our watermarking method is carried out in combination with the progressive compression, a simple and reliable way is to consider, at each iteration, the sets of vertices which are removed (*blue* vertices in Fig. 3 form S_1) and which remain (*white* vertices in Fig. 3 form S_2).

4.2 Watermark Embedding

The first task of our embedding method is to convert the cartesian coordinates of each vertex $v_i = (x_i, y_i, z_i)$ of the current level of detail into spherical coordinates (r_i, θ_i, ϕ_i) using :

$$\begin{aligned} r_i &= \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2} \\ \theta_i &= \cos^{-1}(\frac{z_i - z_c}{r_i}) \\ \phi_i &= \tan^{-1}(\frac{y_i - y_c}{x_i - x_c}) \qquad 0 \le i \le V - 1 \end{aligned} \tag{1}$$

where, V is the total number of vertices of the current intermediate mesh, $V_c = (x_c, y_c, z_c)$ is the center of the mesh, and r_i corresponds to the norm of the vertex v_i.

Afterwards, vertices are divided into two sets S_1 and S_2 as aforementioned, and the distribution of vertex norms of the first set S_1 is divided into K distinct bins $B1_k$:

$$B1_k = \{r_i | r_{min} + \Delta k \le r_i < r_{min} + \Delta(k+1)\} \tag{2}$$

where $0 \le i \le |S_1| - 1$ and $0 \le k \le K - 1$. $|S_1|$ is the number of vertices in S_1, r_{min} and r_{max} are respectively

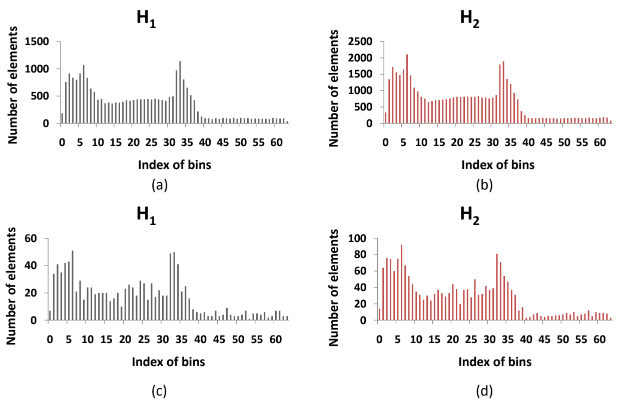

Fig. 4 Histograms of vertex norms of two vertex sets of the finest level (level n) and those of the level $n-7$ of the Rabbit model.

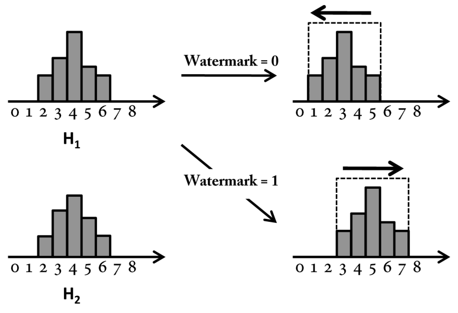

Fig. 5 The value of the center of mass CM_1 is modified to insert a bit. The modification is carried out by shifting the bins of the histogram H_1. Here we set the embedding level $L = 1$.

the minimum and the maximum of the distances of all vertices (i.e. vertices included in S_1 or in S_2). Δ is the size of each bin and also the unit distance used to divide the distributions of vertex norms :

$$\Delta = \frac{r_{max} - r_{min}}{K} \tag{3}$$

In the same way, the bins $B2_k$ of the second vertex set S_2 are obtained. Note that, the Δ and V_c of the finest level of detail are used for all intermediate meshes, and the decoder needs these values to extract correctly the watermark. Histogram H_1 (resp. H_2) is then produced by counting the number of vertices in each bin $B1_k$ (resp. $B2_k$). In the Fig. 4, (a) and (b) illustrate respectively H_1 and H_2 of the finest level (level n, 67039 vertices) and (c) and (d) illustrate respectively those of the level $n-7$ (2970 vertices) of the Rabbit model. We set the number of bins $K = 64$ for these examples.

We can observe in this example that the histograms H_1 and H_2 are very similar even for a coarse level of detail hence it demonstrates the efficiency of our selected primitive and of our method to create the two sets of vertices.

Then, we calculate the centers of mass CM_1 of H_1 and CM_2 of H_2 as follows :

$$CM_1 = \sum_{k=0}^{K-1} \frac{k|B1_k|}{|S_1|}, \quad CM_2 = \sum_{k=0}^{K-1} \frac{k|B2_k|}{|S_2|} \tag{4}$$

where $|B1_k|$ and $|B2_k|$ are the numbers of elements in the bin $B1_k$ and $B2_k$.

These two values are generally very close to each other. For example, $CM_1 = 21.19$ and $CM_2 = 21.07$ for the finest LoD of the Rabbit model (Fig. 4.a and Fig. 4.b) and $CM_1 = 21.6$ and $CM_2 = 20.8$ for the level $n-7$ (Fig. 4.c and Fig. 4.d).

To embed a watermark bit, we modify the relative ordering of CM_1 and CM_2. More concretely, we modify CM_1 by shifting the bins of the histogram H_1 so that the sign of the difference between the modified CM_1^w and CM_2 takes a specific value, according to the watermark bit to insert:

$$CM_1^w - CM_2 = \begin{cases} < 0 \text{ if } w = 0 \\ \geq 0 \text{ if } w = 1 \end{cases} \tag{5}$$

Fig. 5 illustrates the histogram shifting operation. More precisely, this operation consists in subtracting (resp. adding) Δ from (resp. to) the vertex norms of the set S_1 if the embedded bit is 0 (resp. 1).

$$\forall v_i \in S_1, \begin{array}{l} r_i = r_i - \Delta L \text{ if } w = 0 \\ r_i = r_i + \Delta L \text{ if } w = 1 \end{array} \tag{6}$$

In this equation, L designates the embedding level (set to 1 in Fig. 5). It corresponds to the number of shifted bins and it is also related to the degree of variation of CM_1 (i.e. the strength of the watermark). The histogram shifting modifies the center of mass as follows:

$$CM_1^w = \begin{cases} CM_1 - L \text{ if } w = 0 \\ CM_1 + L \text{ if } w = 1 \end{cases} \tag{7}$$

At the receiver side, the bit is retrieved from the sign of $CM_1^w - CM_2$.

In the proposed method, the watermark insertion is possible only when the absolute difference between the two initial center of mass is inferior to the embedding level ($|CM_1 - CM_2| < L$). Otherwise, it is a special case where the embedding is not possible. This special case is discussed in detail in Section 4.4.

Note that we only modify the histogram of the set S_1 which is composed of vertices which will be removed at the upcoming decimation. The reason is that in progressive mesh transmission schemes, the preservation of the original shape in all LoDs is important. So, by not modifying vertices in S_2, vertices in all LoDs conserve their initial position, optimizing the geometric quality.

4.3 Joint Watermark Extraction and 3D Mesh Restoration

To extract the watermark, we need the position of V_c and Δ used for the embedding process. Therefore, we need to store these 4 float data for the correct watermark extraction, they constitute the watermark key. The extraction process is similar to the embedding one. Fig. 6 illustrates one iteration of the decoding (i.e. mesh refinement) and watermark extraction. Firstly, the current intermediate mesh (Fig. 6.a) is refined with insertion of a set of vertices in red in (Fig. 6.b). This mesh contains a visible distortion since the positions of the inserted vertices were changed during the watermark insertion at the encoding. Then, the two histograms are constructed from the two sets of vertices (resp. black and red in the figure).

The comparison between the two centers of mass CM_1^w and CM_2 of these histograms allows extracting a watermark bit w' by calculating the sign of their difference:

$$w' = \begin{cases} 0 \text{ if } CM_1^w - CM_2 < 0 \\ 1 \text{ otherwise} \end{cases} \tag{8}$$

Afterwards, the mesh geometry is corrected by performing the opposite operation of the embedding process (Fig. 6.c):

$$\forall v_i \in S_1, \begin{array}{l} r_i = r_i + \Delta L \text{ if } w' = 0 \\ r_i = r_i - \Delta L \text{ if } w' = 1 \end{array} \tag{9}$$

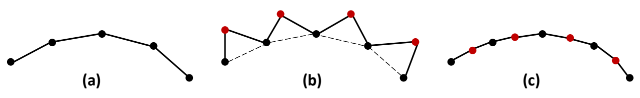

Fig. 6 A step of mesh refinement and watermark extraction. An intermediate mesh (a) is refined (b). After the extraction, the distortion induced by the watermark embedding is removed (c).

4.4 Handling of Special Cases

In the proposed method, a special case occurs when $|CM_1 - CM_2| \geq L$. In this case, the two sets of vertices cannot carry a bit of information, since we cannot adjust the sign of the difference of the two centers of mass according to the watermark bit using the histogram shifting (Eq. 5 and Eq. 7). However, even in this situation, we have to modify CM_1 by shifting the histogram H_1 in the direction which makes the difference become larger, in order not to interfere with the watermark extraction of the normal cases.

The decoder can thus detect this special case by comparing CM_1^w and CM_2:

$$\begin{cases} \text{normal case, if } |CM_1^w - CM_2| < 2L \\ \text{special case, if } |CM_1^w - CM_2| \geq 2L \end{cases} \tag{10}$$

In the case of 2D images [6], the problem of *Salt-and-Pepper* occurs since the highest and lowest bins are shifted to the other side due to the limited and fixed number of bins (number of pixel intensity (256) in [6]), causing a significant visual distortion. In our case, this issue can be easily resolved by generating empty bins at the two extreme sides. Usually the center of gravity of the mesh and the closest vertex are enough distant to create empty bins. If the distance between the center of gravity and the closest vertex is too small, the center of the gravity is moved to the location which enables the creation of empty bins.

4.5 Complete Reversibility and Prediction Method for Geometry

In our algorithm an initial and global quantization is performed at the beginning. Therefore, all vertices have quantized positions and these positions are restored in our reversible scheme.

Unfortunately, the reversibility is not guaranteed for all vertices. An example is shown in Fig. 7. During the embedding, an initial vertex V_i is moved by adding Δ to its distance from the center of gravity, V_c. Then it is moved to the nearest quantized position, V_w (watermarked position). At the extraction, the vertex V_w is displaced this time by subtracting Δ and then it is moved to the nearest quantized position, V_e (extracted position). Positions of V_i and V_e are different in this case, since the displacement is not performed in the exactly same direction between the embedding and the extraction.

To avoid accumulation of geometric error during the iterative refinement and watermark removal, the extracted positions are precomputed during the encoding process and they are used for the geometry coding.

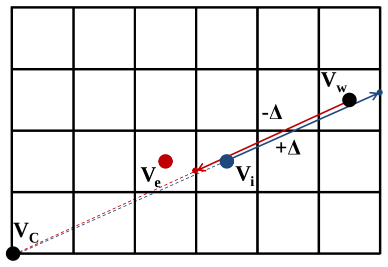

Fig. 7 An example of violation of reversibility. The initial position V_i is moved to the watermarked position V_w during the embedding. At the extraction, V_w is moved in a slightly different opposite direction. The resulting extracted position V_e is different from V_i in this case.

We offer also the possibility of a complete reversibility, by encoding in the compressed stream, the difference between V_e and V_i. In the example of Fig. 7, we encode a vector (1, 0) to move V_e to the initial position V_i. Such violation of reversibility occurs rarely, hence only a very small extra coding cost is needed for this option (see experiments).

The watermarking insertion usually affects the coding rates, since the mesh geometry is deformed during the watermark embedding which precedes the encoding; the geometry of the current mesh (Fig. 6.c) is modified to embed a watermark bit (Fig. 6.b) and then these modified vertices are encoded to allow the decoder to extract the watermark.

To reduce the deterioration of the coding performance, we propose a new prediction method, as illustrated in Fig. 8.

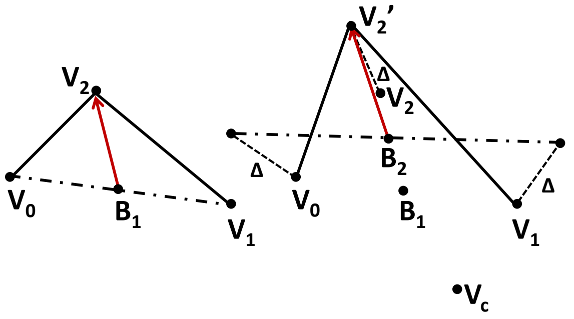

Fig. 8 The original prediction method (left) [1] does not perform well in our joint system (right). Our proposed prediction method takes into account the displacement of the vertex (V_2') to encode efficiently the geometry.

Fig. 8.a shows the prediction method used in the original geometry coder [1]. Position of the vertex to be removed V_2 is predicted as the average position B_1 of its neighboring vertices V_0 and V_1. In our case (Fig. 8.b), V_2 is moved to V_2' by the watermarking insertion hence B_1 becomes a very bad prediction of its position. To resolve that, we predict it using B_2 which is the average position of V_0 and V_1 after a displacement with the same amplitude and same sign from the center of gravity V_c. By taking into account this displacement, the geometry in encoded more efficiently.

One important issue is to know the displacement sign of each region at the receiver side. This issue is resolved by encoding the three first visited vertices without any prediction so that the decoder deduces the direction from comparison between these vertices and the configuration of their neighboring vertices.

4.6 Increase of Watermark Payload

The proposed method can embed only one bit in each level of detail. This watermarking payload (number of embedded bits) is usually not enough for common applications. To increase the payload, we first decompose the mesh of each LoD by grouping vertices into regions so as to embed a bit into each region. The decomposition has to be processed in a deterministic way so that both the encoder and the decoder have the same region configuration. Our simple and fast way is to decompose first the coarsest mesh into R regions, and then to grow these regions simultaneously with the mesh refinement. More concretely, every time a vertex is inserted, we assign the index of the most frequent region among its neighboring vertices. This mesh decomposition allows increasing the number of inserted bits.

We observe that the number of vertices in a region has to be high enough to be able to carry a bit of information, and it is often higher than necessary in the finest resolutions of large meshes. To increase further the embedding payload, we divide a region into two sub-regions if its number of vertices is higher than a user-defined parameter $N_{division}$.

On the contrary, when a bit is inserted into a region with very few vertices, this watermark bit is naturally more sensitive; moreover, when the initial position of a vertex is not restored correctly due to the reversibility violation problem (cf. Section 4.5), this vertex can be assigned to an other bin, disturbing the correct watermark extraction for a region with few elements. For those reasons, when the number of elements of a region is inferior to N_{min}, this region is not used for the watermarking.

5 Experimental Results

The proposed method has been implemented in C++. Several models are used for the evaluation and four of them are shown in Fig. 9: Horse (19 851 vertices), Bunny (34 835 vertices), Dragon (50 000 vertices) and Venus (100 759 vertices). The watermark bits are generated randomly. The following parameter settings are used in our experiments, if any modification is mentioned; the number of bins $K = 256$, the number of regions $R = 20$, the embedding level $L = 2$, the minimum number of elements to use the region for the watermarking $N_{min} = 50$, and the region division and the complete reversibility options are disabled.

5.1 Baseline Evaluation

Fig. 10 illustrates the meshes obtained after the full reconstruction without using the complete reversibility with an embedding level $L = 6$. We can see that the distortion is quite imperceptible, even if some vertices violate the reversibility during the watermark extraction (see Section. 4.5).

We also illustrate a step of vertices insertion and watermark extraction in Fig. 11. We insert a set of watermarked vertices to an intermediate mesh of level $n-4$ of the Horse model (a) to obtain a higher level of detail

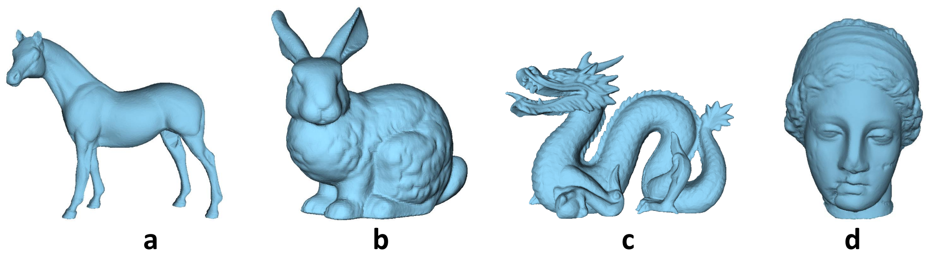

Fig. 9 The original meshes: (a) Horse, (b) Bunny, (c) Dragon and (d) Venus.

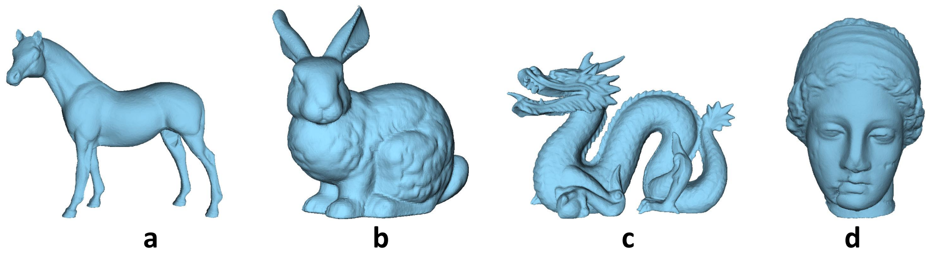

Fig. 10 The reconstructed meshes: (a) Horse, (b) Bunny, (c) Dragon and (d) Venus. Here, we set $L = 6$.

(b) during the decoding process. Then, the watermark is retrieved and the mesh geometry is corrected (c). During the progressive transmission, these two operations are performed together so that the intermediate meshes which are deformed by the watermark insertion are not visualized at the receiver side.

Table 1 shows the compression performance of the original algorithm of Alliez and Desbrun (AD 2001) [1] and the proposed method for $L = 2$ and $L = 6$. All meshes are quantized using 12 bits. Distortions between the original models and the reconstructed models at the finest level of detail are measured by Metro tool [4] in terms of maximum root mean square error (MRMS) with respect to the bounding box. We list also the processing time of the joint compression and watermark embedding (in seconds) and the payload. All the tests are achieved on a laptop running on an Intel Core 2 Duo T9600 2.80 GHz processor with 4 GB memory.

Our quasi-reversible method encodes the tested meshes with similar coding costs to the original compression method (AD 2001). The coding overheads induced by the introduction of the watermarking are only 0.56 bits-per-vertex (bpv) with $L = 2$ and 1.10 bpv with $L = 6$ on average, which demonstrates the efficiency of the proposed geometry prediction method. This coding overhead is mostly caused by the geometry deformation induced by the watermark embedding, and it is barely affected by the number of inserted bits. Also, the complete reversibility can be achieved with an additional coding cost of 0.15 bpv with $L = 2$ and 0.45 bpv with $L = 6$ on average. The distortion values of our complete-reversible me-thod correspond to the error induced by the initial quantization and we can see that the reconstructed meshes with our quasi-reversible method possess almost the same distortions.

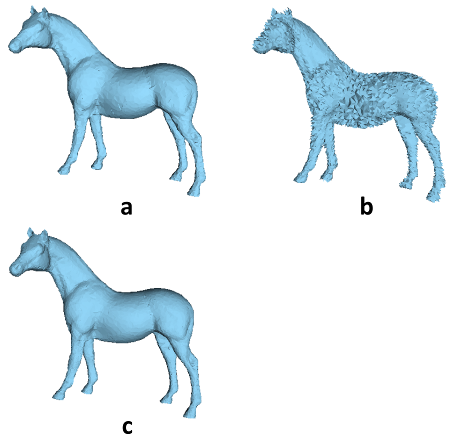

Fig. 11 A step of vertices insertion and geometry correction of an intermediate LoD: (a) Level $n - 4$ of the Horse model. (b) A set of watermarked vertices is inserted for level $n - 3$. (c) Reversible watermark extraction corrects the deformation induced by the watermark embedding. Here we set $L = 6$.

Fig. 12 illustrates the embedding payload of the proposed method for variable numbers of regions, R. The region division option is disabled in Fig. 12.a and it is activated with $N_{division} = 500$ in Fig. 12.b. We can remark that the payload is generally high for all tested models and it is relatively proportional to the size of the meshes. The reason is that large meshes have generally a higher number of LoDs and their regions are composed of more elements. The number of LoDs is between 10 and 14 for the tested models. We have also applied our method on a less dense mesh (Fandisk, 6

	L	Horse	Bunny	Dragon	Venus
AD 2001		20.89	18.55	19.94	17.30
Quasi-reversible	2	21.18	19.47	20.34	17.95
	6	21.65	20.27	20.48	18.68
MRMS (10^{-5})	2	3.71	3.65	4.00	3.36
	6	4.47	4.25	4.72	3.73
Complete-reversible	2	21.33	19.61	20.52	18.05
	6	22.23	20.72	20.97	18.97
MRMS (10^{-5})		3.70	3.46	3.79	3.24
Processing time (s)		1.37	2.43	3.82	7.57
Payload (bits)	2	93	116	110	171
	6	106	131	137	183

Table 1 Compression rates of the original method (AD 2001) [1], our method with quasi reversibility and with complete reversibility. The numbers are in bits-per-vertex (bpv). Distortions, processing times and payloads are also listed.

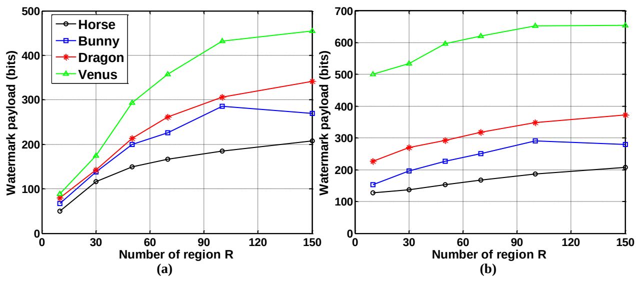

Fig. 12 Watermark payload versus number of regions. Initial number of regions is preserved during the whole transmission in (a) and large regions are divided in (b) with threshold $N_{division} = 500$.

475 vertices). The number of LoDs for this model is 7, and we can embed 52 bits with $L = 6$. The payload is sufficient even for the models with few vertices.

In the proposed method, the mesh refinement can be achieved without using the watermark extraction. Thus, according to the permission level of each user, we can provide different quality of the mesh geometry by activating or not the watermark extraction. For instance, we can disable the watermark extraction and geometry correction for the users with limited authorization so that they obtain significantly deformed reconstructed meshes, as illustrated in Fig. 13.

Contrary to an encryption algorithm which forbids unauthorized users to obtain the reconstructed mesh, our method allows them progressive visualization of the distorted version of the model.

5.2 Robustness Evaluation

For the robustness evaluation, we set $N_{min} = 200$ and $L = 6$. The robustness of the embedded watermark is evaluated in terms of BER (bit error rate) and correlation between the inserted watermark bit string $\{w_n\}$ and the extracted one $\{w'_n\}$ as given by the following equation:

$$Corr = \frac{\sum_{n=0}^{N-1}(w_n - \overline{w})(w'_n - \overline{w}')}{\sqrt{\sum_{n=0}^{N-1}(w_n - \overline{w})\sum_{n=0}^{N-1}(w'_n - \overline{w}')}} \quad (11)$$

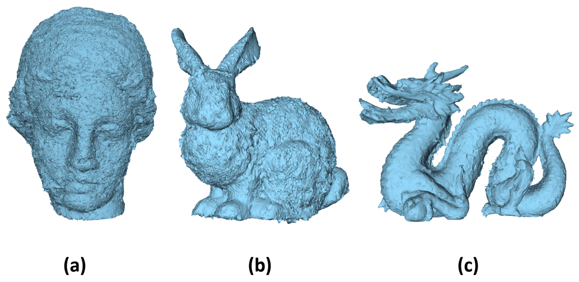

Fig. 13 The reconstructed meshes without the watermark extraction and geometry correction: (a) Venus, (b) Bunny, (c) Dragon.

Model	MRMS (10^{-3})	BER	Corr	Payload
Horse	1.03	0.01	0.98	51
Bunny	1.52	0.01	0.98	72
Dragon	2.42	0.34	0.32	83
Venus	1.77	0.07	0.85	128

Table 2 Robustness against the channel noise which is simulated by random noise addition attack to every inserted vertex.

where $\overline{w}$ and $\overline{w}'$ are respectively the average of $\{w_n\}$ and $\{w'_n\}$. The results are obtained as the average of five experiments using different watermark bits.

5.2.1 Robustness against channel noise

During the transmission over the network, errors can be introduced due to noise on the channel. These errors can perturb the decoder so that the positions of the inserted vertices are modified. We evaluate the robustness against this attack by considering it as a random noise addition attack applied to every inserted vertex at each refinement operation.

Table 2 presents the robustness results. The amplitude of noise addition is set to 0.5% of the average distance from the vertices to the mesh center. Our algorithm is fairly robust against such attack except the Dragon model, since its shape complexity is relatively high regarding its number of elements.

5.2.2 Robustness against intentional attacks

During the transmission, any intermediate mesh can be stolen and modified by a pirate to remove the watermark. Thus, the protection of all intermediate meshes is crucial. In the proposed method, the extraction of the watermark bits from any level of detail (to prove its ownership) is possible, simply by inserting the vertices of the next resolution level. The finest level can also be protected by creating and by encoding a supplementary level at the encoding process using a mesh subdivision method. This supplementary level is not transmitted to the client and it is used only for the protection of the finest level.

We give here the robustness evaluation against only common geometric attacks. As our algorithm is based

Model	MRMS (10^{-3})	BER			Corr		
		Our	Cho	Wang	Our	Cho	Wang
Horse	3.82	0.04			0.91		
Bunny	3.52	0.03	0.17	0.11	0.93	0.69	0.77
Dragon	4.17	0.04		0.19	0.86		0.61
Venus	4.00	0.07		0.11	0.91		0.78

Table 3 Robustness against random noise addition with an amplitude of 0.5%.

Model	MRMS (10^{-2})	BER			Corr		
		Our	Cho	Wang	Our	Cho	Wang
Horse	2.22	0.02			0.96		
Bunny	2.35	0.03	0.16	0.19	0.93	0.69	0.62
Dragon*	2.66	0.33		0.24	0.31		0.52
Venus	2.59	0.01		0.04	0.99		0.92

Table 4 Robustness against Laplacian smoothing with 30 iterations and a deformation factor $\lambda = 0.03$. For the Dragon model 50 iterations are used.

Model	MRMS (10^{-3})	BER			Corr		
		Our	Cho	Wang	Our	Cho	Wang
Horse	3.18	0.03			0.93		
Bunny	3.33	0.05	0.47	0.15	0.91	0.07	0.70
Dragon	4.25	0.35		0.39	0.26		0.23
Venus	4.18	0.11		0.11	0.77		0.79

Table 5 Robustness against uniform 7-bit quantization of the mesh vertices.

on a connectivity-based compression technique, the refinement is not possible when a change of connectivity occurs. Hence our algorithm is not robust to connectivity change.

In Tables 3, 4 and 5 we compare the robustness evaluation results under random noise addition, smoothing and uniform quantization of the mesh coordinates with the algorithm of Cho et al. [3] and Wang et al. [22]. Their results have been taken from [22].

These comparisons have to be seen on a qualitative basis since there are many differences between these algorithms and ours:

– The payloads are not the same (45 to 75 bits for [22], 67 bits for [3] and 51 to 128 bits for us).
– The attacks are applied iteratively to each intermediate mesh, except the finest one, for our method while these attacks are applied only once for the other algorithms.
– Our method needs the center of gravity V_c and Δ for the watermark extraction, while the other methods do not require any supplementary information.

We can see that even if the attacks are applied repeatedly in each level of details, our method demonstrates very good robustness performances regarding the algorithms from Cho et al. [3] and Wang et al. [22]. Bit error rates are very low, between 5% and 10%, in almost all cases.

6 Conclusion

In this paper, we have presented joint reversible watermarking and progressive compression of 3D meshes. Each LoD is compressed and watermarked by modifying the geometry of refined vertices with respect to the center of mass of the original 3D mesh. The watermark process is reversible in the sense that the geometrical modifications introduced by the embedding processing can be removed after watermark extraction. The proposed method is robust against channel noise and intentional attacks while having slightly additional compression rate cost. For future works, we plan to extend the robustness of our method to connectivity attacks.

Acknowledgements We gratefully acknowledge Kai Wang for his help on robustness of watermarking schemes under progressive compression attack. This work has been supported by French National Research Agency (ANR) through COSINUS program (project COLLAVIZ n°ANR-08-COSI-003).

References

1. Alliez, P., Desbrun, M.: Progressive encoding for lossless transmission of 3D meshes. In: ACM SIGGRAPH, pp. 198–205 (2001)
2. Chen, H.K., Chen, Y.H.: Progressive watermarking on 3D meshes. In: 2010 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), pp. 1–7. IEEE (2010)
3. Cho, J., Prost, R., Jung, H.: An oblivious watermarking for 3-D polygonal meshes using distribution of vertex norms. IEEE Transactions on Signal Processing **55**(1), 142–155 (2006)
4. Cignoni, P., Rocchini, C., Scopigno, R.: Metro: Measuring Error on Simplified Surfaces. Computer Graphics Forum **17**(2), 167–174 (1998)
5. Cohen-Or, D., Levin, D., Remez, O.: Progressive compression of arbitrary triangular meshes. In: Visualization, pp. 67–72 (1999)
6. De Vleeschouwer, C., Delaigle, J.F., Macq, B.: Circular interpretation of bijective transformations in lossless watermarking for media asset management. IEEE Transactions on Multimedia **5**(1), 97–105 (2003)
7. Gandoin, P.M., Devillers, O.: Progressive lossless compression of arbitrary simplicial complexes. In: ACM SIGGRAPH, pp. 372–379 (2002)
8. Goudia, D., Chaumont, M., Puech, W., Said, N.H.: A joint JPEG2000 compression and watermarking system using a TCQ-based quantization scheme. In: Visual Information Processing and Communication II (VIPC 2011), vol. 7882 (2011)
9. Hoppe, H.: Progressive mesh. In: ACM SIGGRAPH, vol. 96, pp. 99–108 (1996)
10. Konstantinides, J.M., Mademlis, A., Daras, P., Mitkas, P.A., Strintzis, M.G.: Blind Robust 3-D Mesh Watermarking Based on Oblate Spheroidal Harmonics. IEEE Transactions on Multimedia **11**(1), 23–38 (2009)
11. Lee, H., Lavoué, G., Dupont, F.: Adaptive coarse-to-fine quantization for optimizing rate-distortion of progressive mesh compression. In: Vision, Modeling, and Visualization Workshop, pp. 73–81 (2009)

12. Liu, Y., Prabhakaran, B., Guo, X.: A robust spectral approach for blind watermarking of manifold surfaces. In: Proceedings of the 10th ACM workshop on Multimedia and security, pp. 43–52. ACM (2008)
13. Lu, Z., Li, Z.: High Capacity Reversible Data Hiding for 3D Meshes in the PVQ Domain. In: Proceedings of the 6th International Workshop on Digital Watermarking, pp. 233–243 (2008)
14. Ohbuchi, R., Masuda, H., Aono, M.: Watermarking three-dimensional polygonal meshes. In: ACM Multimedia, pp. 261–272 (1997)
15. Pajarola, R., Rossignac, J.: Compressed Progressive Meshes. IEEE Transactions on Visualization and Computer Graphics **6**, 79–93 (2000)
16. Peng, J., Kim, C.S., Kuo, C.C.J.: Technologies for 3D mesh compression: A survey. Journal of Visual Communication and Image Representation **16**(6), 688–733 (2005)
17. Peng, J., Kuo, Y., Eckstein, I., Gopi, M.: Feature Oriented Progressive Lossless Mesh Coding. Computer Graphics Forum **29**(7), 2029–2038 (2010)
18. Praun, E., Hoppe, H., Finkelstein, A.: Robust mesh watermarking. In: ACM SIGGRAPH, pp. 49–56 (1999)
19. Taubin, G., Guéziec, A., Horn, W., Lazarus, F.: Progressive forest split compression. In: ACM SIGGRAPH, pp. 123–132 (1998)
20. Valette, S., Chaine, R., Prost, R.: Progressive lossless mesh compression via incremental parametric refinement. In: Proceedings of the Symposium on Geometry Processing, vol. 28, pp. 1301–1310 (2009)
21. Wang, K., Lavoue, G., Denis, F., Baskurt, A.: A Comprehensive Survey on Three-Dimensional Mesh Watermarking. IEEE Transactions on Multimedia **10**(8), 1513–1527 (2008)
22. Wang, K., Lavoué, G., Denis, F., Baskurt, A.: Robust and blind mesh watermarking based on volume moments. Computers & Graphics **35**(1), 1–19 (2011)
23. Wang, Y., Hu, S.: A New Watermarking Method for 3D Models Based on Integral Invariants. IEEE Transactions on Visualization and Computer Graphics **15**(2), 1 (2009)
24. Willems, F.M.J., Kalker, T.: Coding theorems for reversible embedding. In: Proc. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 66, pp. 61–76 (2003)
25. Wu, H., Dugelay, J.: Reversible watermarking of 3D mesh models by prediction-error expansion. In: 2008 IEEE 10th Workshop on Multimedia Signal Processing, pp. 797 –802 (2008)
26. Yang, E., Wu, G.: Joint compression and blind watermarking: A case study in the jpeg-compatible scenario. In: Proc. of the 43th Allerton Conference on Communications, Control, and Computing (2008)
27. Zafeiriou, S., Tefas, A., Pitas, I.: Blind robust watermarking schemes for copyright protection of 3D mesh objects. IEEE Transactions on Visualization and Computer Graphics **11**(5), 596–607 (2005)