INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Watermarking 3D triangle meshes for authentication and integrity

François Cayre — Olivier Devillers — Francis Schmitt — Henri Maître

N° 5223

Juin 2004

Thèmes COG et SYM

ISSN 0249-6399 ISRN INRIA/RR--5223--FR+ENG

INRIA
RENNES

Watermarking 3D triangle meshes for authentication and integrity

François Cayre , Olivier Devillers , Francis Schmitt , Henri Maître

Thèmes COG et SYM — Systèmes cognitifs et Systèmes symboliques
Projets Temics, Geometrica

Rapport de recherche n° 5223 — Juin 2004 — 26 pages

Abstract: Most watermarking schemes for 3D meshes arise from the CAD community. The analysis of their watermarking performances are generally not easily tractable. In this paper, we describe a framework for fragile watermarking of 3D triangle meshes that enables accurate assessment of its performances. Experiments show the relevance of the method towards authentication and integrity of heritage data purposes.

One often argues that authentication may be performed using pure cryptographic primitives. We use watermarking because we want to allow some non malicious transformations of the mesh. Moreover, our technique enables visual inspection of the mesh for integrity assessment.

Our method is based on alteration of geometrical invariants along a special traversal of the mesh connectivity graph. We proove that our traversal takes linear time for maximizing the number of sites that can store a watermark bit. We deduce several watermarking properties of our scheme, including the minimum watermark segment (MWS), the security (in bits) of the embedding, the class of robustness and the probability of false alarm. To our knowledge, such an accurate characterization of a watermarking scheme for 3D triangle meshes is not trivial for other similar methods.

Key-words: fragile watermarking, 3D triangle mesh, authentication, integrity, geometric invariant

Affiliations: François Cayre (IRISA part of his work has been done at TSI) Olivier Devillers (INRIA Sophia-Antipolis) Francis Schmitt (TSI, Télécom Paris) Henri Maître (TSI). This work was funded in part by the ARC Télégéo grant (INRIA).

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 —Télécopie : +33 2 99 84 71 71

Tatouage de maillages triangulaires 3D pour l'intégrité et l'authentification

Résumé : Nous décrivons dans cet article un algorithme de tatouage fragile pour des maillages de surfaces par des triangles et une analyse rigoureuse des performances de ce schéma de tatouage. Cette étude montre la pertinence de cette approche pour du tatouage aux fins d'assurer l'authenticité et l'intégrité de l'objet.

Contrairement à une approche cryptographique, le tatouage va nous permettre d'autoriser quelques manipulations banales du maillage telles que rotation ou translation en continuant de garantir l'intégrité de l'objet d'un point de vue géométrique (et pas du fichier original).

Notre méthode est basée sur une modification légère de certains invariants géométriques au cours d'un parcours du graphe d'adjacence du maillage. Ce parcours prend un temps linéaire et permet de maximiser le nombre de sommets du maillage hébergeant un bit tatoué. Nous en déduisons des propriétés de notre technique telle que la plus petite surface tatouée, la sécurité (en bits) de l'insertion, la classe de robustesse et la probabilité de fausse alarme. À notre connaissance obtenir de telles garanties pour les autres procédés de tatouage des maillages 3D est difficile.

Mots-clés : tatouage fragile, maillages triangulaires 3D, authentification, integrité, invariant géométrique

1 Introduction

The rapid evolution of network bandwiths along with the development of near-optimal source coding algorithms [1][2][3] for 3D triangle meshes enable fast transmission of such data. Applications range from biomedical imaging to heritage and geological data representation. Therefore, there is a clear need for new methods of authenticating meshes. Fragile watermarking has already shown to be an effective tool for ensuring such an authentication of multimedia content (the watermark is supposed to disappear when the mesh undergoes an illegal manipulation). We propose in this paper a new framework to handle fragile watermarking for authentication (and integrity) purposes. The advantages of our framework are numerous, as it allows precise design of watermarking schemes and accurate assessment of their watermarking performances.

In the first part of this paper, we recall some background on meshes and we present the basics of watermarking with geometrical invariants (invariants are scalar values we will use for our watermarking space). We then explain a special (indexed) localization of the watermark. A localization is the way the numbers of the watermark bits are committed. There are three types of localizations : global, local, and indexed (this last one is dedicated to meshes).

In the second part, we describe our new framework for fragile watermarking. It is composed of a local watermarking procedure and an indexed localization of the hidden information. The class of robustness of the algorithm is deduced. We then develop a special traversal of the graph that maximizes the performances of the indexed localization in linear time. Next, we propose a statistical model for detecting the watermark with a user-defined probability of false alarm.

Finally, experimental results on heritage data show the power of our framework. We deduce the actual minimal size of the watermarked surface (MWS). We also discuss further optimization of the framework both in terms of MWS and robustness. We conclude with a sketch that enables visual inspection of the mesh integrity and performances optimization.

2 Background

Watermarking on sound, image and video always implicitly rely on the regular sampling of the data : every audio sample has two neighbors almost everywhere (except the first and last), every pixel has four (or eight, depending on the neighboring function) neighbors everywhere except on the border. Furthermore, the sampling

distance is constant. Conventional data is then represented on a regular sampling grid whereas meshes are defined on an irregular sampling grid : every vertex may have an arbitrary number of neighbors (its valence) at variable distance. We first define what kind of meshes we aim at watermarking (manifolds without degeneracies). Then, we give a quick overview of watermarking meshes with geometrical invariants in the literature. We do not discuss other watermarking schemes, since they are generally based on a special representation of the mesh that better suits robust watermarking purposes [4][5].

2.1 Dealing with meshes

Let $M(C,G)$ be an orientable manifold[1] triangle mesh with graph connectivity C and cartesian coordinates geometry $G(x,y,z)$. Let N (resp. E, F) be the number of vertices (resp. edges, triangles) in M. The geometry is defined with three vectors x,y,z that assign 3D cartesian coordinates to every vertex. One says that the mesh is composed of the geometry G defined over the arbitrary connectivity C. The Euler relation links N, E and F for triangle meshes. In the special case of manifold meshes, it writes [6] :

$$N + F - E = 2 - 2 \times g, \tag{1}$$

where g is the genus of the surface : a sphere has a 0 genus, a torus has a 1 genus, etc. One can also add other attributes to the mesh like texture coordinates, color, local refraction, etc. Processing of such data is out of the scope of this work. We only focus on watermarking the geometry of the mesh. We will generate a watermarked mesh M_w containing a watermarked message W with a secret key K :

$$M_w^K(C,G_w) = w(W,M(C,G),K), \tag{2}$$

where w stands for the watermarking process. The watermark decoder d will retrieve an estimated message $\hat{W}$ from M_w^K :

$$\hat{W} = d(M_w^K,K). \tag{3}$$

We will later discuss the length of K and propose a way to ensure with a certain amount of confidence that correct detection is performed, i.e. :

$$W = \hat{W}. \tag{4}$$

We use a special strategy for watermarking : the alteration of geometrical invariants. This approach leads to exact characterization of the watermark class of robustness. We recall below the basics of such a watermarking technique.

[1] A mesh is manifold when every edge belongs exactly to two triangles.

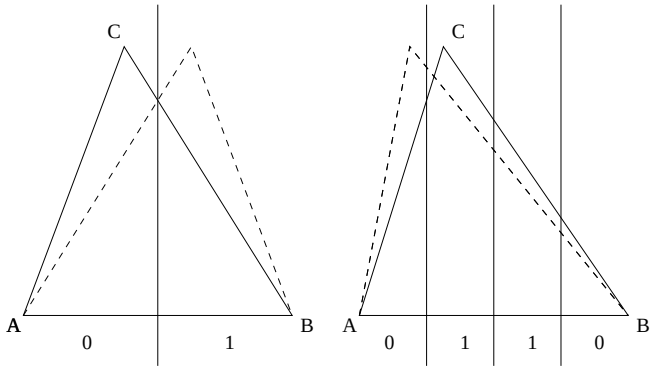

Figure 1: Embedding of a watermark bit through geometrical invariant in one triangle. The vertex C is projected onto (AB) to assess the value of the bit hidden in ABC. The invariant used is the ratio of the lengths of two colinear vectors : it is thus rotation, scaling and translation (RST) invariant. Alteration of the invariant is easily performed through a symetry of C with respect to the next symetry plane. Such a watermarking procedure is non linear, often called substitutive.

2.2 Watermarking with geometrical invariants

We call invariant a geometric scalar value that remains unchanged after a given set of geometric transforms. Modifying geometrical invariants relies on the implicit definition of a special geometrical configuration : one must be able to exactly recover the configuration in which the invariant is to be computed. In classical watermarking, it is related to channel estimation and resynchronization. We dedicate the alteration of geometrical invariant to fragile watermarking : attacks that change the connectivity C do not allow exact registration on the original connectivity. Such attacks include mesh simplification or decimation [7], remeshing [8] and refinement [9], among others.

Geometrical configuration in which invariants are computed may be an imaginary tetrahedron associated to every couple of adjacent triangles (see the TVR scheme in [10]), a group of four adjacent triangles (TSQ scheme in [10]), or even only one triangle as in [11]. A geometrical configuration is chosen from an embedding point of view : the alteration of the invariant should preferably be easily computable (otherwise one proceeds to an iterative magnitude-decreasing alteration until a certain condition is satisfied [12]). In this work, we build upon [11] to keep a 1-triangle geometrical configuration (see Fig. 1) that meets all of the following requirements.

In this paper, we consider that only robustness against rotation, scaling and translation (this set of transforms is denoted RST in the sequel), as well as cropping of the mesh is desirable. Our strategy is as follows : robustness against RST manipulations is provided through geometrical invariants and robustness against cropping relies on a special (indexed) localization of the watermark.

2.3 Indexed localization of the watermark

One important remark on watermarking with geometrical invariants deals with the way numbers are assigned to the recovered bits. Classical watermarking, in one way or another, always considered an implicit way of numbering the bits of the watermark.

2.3.1 Classical localizations

In a 2D image Least Significant Bits (LSB) steganography framework [13], the numbering of the bits is implicit : it is based on a canonical line/column traversal. An analogon for 3D triangle meshes is described in [11], although no bound on capacity can be easily derived (we give here an optimal mesh traversal that is a solution for this problem). In 2D watermarking with spatial repetition of the same watermark pattern, autocorrelation is used to resynchronize the watermark. The former is called *global* localization, the latter is called *local*.

2.3.2 Indexed localization

Another kind of localization may be employed with meshes : a so-called *indexed* localization (TSQ scheme in [10]). The key idea of such a localization is to *explicitly* write the number of the watermark bit along with its value. In such an indexed localization setting that we use below, a watermark message W actually writes :

$$W = (W_i^n, W_i^v) \quad \forall\, i \in [1; |W|], \tag{5}$$

where W^n (resp. W^v) denotes the number (resp. the value) of the watermark bits.

Let N_b be the actual number of bits necessary to hide W. An indexed localization introduces a significant overhead :

$$N_b = |W| \times (1 + \log_2(|W|)), \tag{6}$$

where $|.|$ denotes the length of a binary message. However, in the sequel we keep the usual definition of capacity : we aim at hiding typically $|W| = 64$ bits, and we thus need to hide $N_b = 64 \times (1 + \log_2(64)) = 448$ bits. Due to the overhead, such

a localization is naturally dedicated to fragile watermarking for high-resolution data (float or double type).

3 Framework

We are now able to fully describe this new framework. First, we show how to add an indexed localization of the watermark to the geometrical procedure of [11] depicted in Fig. 1. This is done through a kind of geometrical DMA (Division Multiple Access) coding : one geometrical channel is dedicated to the bit value encoding and another is devoted to the encoding of the bit numbers. By construction, these two channels do not interfere.

Next, we use algorithmic geometry to ensure optimal traversal of the mesh connectivity graph with respect to our geometrical embedding procedure. This may be seen as a complement of [11], as this traversal may be used in a global localization framework towards steganography purposes. However, we restrict the scope of this paper to fragile watermarking only. We proove that our traversal ends with $O(N)$ iterations and that it enables watermark embedding in exactly $N-2$ triangles.

Since we shall know the number of watermarked triangles, statistical modelling of the detection becomes possible, as well as assessment of various watermarking properties of our scheme. We assume classical Gaussian modelling for the watermark detector, and we let the probability of false alarm be $P_{fa} = 10^{-7}$. The same modelling will allow us to set the length of the keys that enables 24-bit security.

3.1 Local watermarking procedure

Our local watermarking procedure is twofold. The first subprocedure is dedicated to the embedding of the watermark bit value W^v and the other to its number W^n. Both subprocedures do not interfere geometrically. The subprocedure dedicated to the bit value is exactly the same as in [11]. From now on, a watermark site is a triangle ABC. C is the only vertex that undergoes watermark geometrical embedding : AB is the reference edge.

In order to bound the embedding distorsion, we commit the coordinates alteration of C only if the watermarked C is inside a sphere of radius r centered on the original C (we also discard this way numerical degeneracies). We have chosen :

$$r = 10^{-3} \times d_{bbox}, \tag{7}$$

where d_{bbox} denotes the diagonal of the bounding box. Clearly, one should also take into account local curvature to better respect CAD surfaces for example. However,

we only used basic distorsion control since it was sufficient for illustrating the ideas of this work.

3.1.1 Embedding watermark bit values W^b

In [11], a local geometrical procedure that enables embedding of one bit value in one triangle is described. The idea is as follows : a reference edge (AB) of the triangle is divided into 1 and 0 interleaved partitions, and the other vertex (C) of the triangle is projected onto the reference edge to read the bit value (see Fig. 1). To be able to process every triangle, the partition of the reference edge is repeated along the whole line (AB). The geometrical invariant used is the ratio of the lengths of two colinear segments. Substitutive embedding is committed through symetries along the next symetry planes. This way, the embedding of the bit value keeps the host triangle in the same original plane (but not the neighboring triangles). Furthermore, the distorsion is user-defined : one can tune the number of interleaved partitions to decrease the embedding distorsion. We will use 64 interleaved partitions in the sequel. Note that using symetries for embedding does not give any information to a pirate (see the security analysis below). Bit *value* embedding is performed *colinearly* to AB : the area of the triangle ABC remains unchanged (we use this special property in the next section).

3.1.2 Embedding watermark bit numbers W^n

In [10], the geometrical configuration of the TSQ scheme that enables indexed localization contains four triangles : one is dedicated to the watermark bit number, the three others carry hidden data (3 channels of bit values). However, we could not find in [10] a way to maximize the number of such 4-triangle geometrical configurations.

We now present an indexed localization of the watermark that can be easily plugged to the previous subprocedure dedicated to bit value encoding. We preserve the nice property of the previous subprocedure : embedding always keeps the host triangle in the same plane. As we will see, bit *numbers* embedding will be performed *perpendicularly* to AB. This is the reason why our two subprocedures do not interfere and are easily pluggable to one another. The geometrical invariant we use for bit numbers embedding is the ratio of the area of the imaginary square drawn from AB and the area S of the ABC triangle.

Let $K = K_1 \oplus K_2$ be a secret key made of two concatenated subkeys K_1 and K_2. Let also T_1 and T_2 be internal parameters of the algorithm that will be tuned later

on. We define the following secret watermark variable G_{ABC} in the triangle ABC :

$$G_{ABC} = \left(T_1 + \frac{K_1}{2^{|K_1|}}\right)^{\left(T_2 + \frac{K_2}{2^{|K_2|}}\right)} \times \frac{AB^2}{S}, \tag{8}$$

where the invariant is modulated with the secret key. Geometrically, we use the trivial relation linking the height H based on AB to S :

$$2 \times S = AB \times H \tag{9}$$

to perform an embedding that is perpendicular to AB by modifying only the height H (always in the triangle plane). A watermark bit number W^n embedded in ABC is decoded with :

$$W^n = \lfloor G_{ABC} \rfloor \mod |W|. \tag{10}$$

And the insertion is performed such that after embedding ABC verifies :

$$|G_{ABC} - \frac{1}{2} - \lfloor G_{ABC} \rfloor| < \epsilon, \tag{11}$$

where ϵ is a (potentially) user-defined decoding parameter for sensitiveness. This last equation is classical in watermarking : it often arise from robustness reasons. However, we only use this value of $\frac{1}{2}$ to :

- correctly distinguish between watermarked and non-watermarked triangles,
- correctly recover the orientation of the site (triangle) : which vertex was C *at embedding* (watermark resynchronization).

We will see later how to take advantage of this to model the statistical confidence in the detector output.

3.1.3 Equirepartition of watermark bits

To ensure that all bits of W are embedded the same number of times, we just use the simple scheme of marking bit number $j+1$ after bit number j (modulo $\|W\|$). However, it is possible to use a more sophisticated scheme to minimize mesh distorsion since the modification of a triangle depends on which bit of W is to be embedded. Such a sophisticated scheme would try to minimize the distorsion in every triangle by chosing the best bit of W to be embedded under the constraint of embedding every bit approximately the same number of times.

3.2 Optimal traversal

One important point on indexed localization techniques is that they *locally* enable watermark recovery. In particular, no specific traversal is needed at decoding to cope with mesh cropping. At decoding side, we proceed every triangle after one another, independently of any connectivity relationship (we could proceed them in random order). Decoding then takes $O(F)$ time, which is[2] $O(N)$ by Euler relation (Eq. 1). Such a constraint on decoding imposes a special traversal *during embedding* that maximizes the number of usable sites for watermarking. The purpose of this part is to propose an optimal traversal of the mesh from this point of view.

The key problem is to find a traversal that :

- do not erase what was written before (causality issue),
- ends with polynomial complexity (linear at best),
- reaches a predictable number of sites (for statistical modelling).

3.2.1 Causality issue

The causality issue is straightforward to understand : the mesh traversal cannot select a C vertex (supposed to undergo coordinates alteration) that already participated in marking a previous site. The two situations to be avoided are depicted in Fig. 2. First, a vertex that was previously selected as a C vertex cannot be selected anymore. Neither can a vertex that was previously playing a A or B role.

To avoid every risks of erasure, we dynamically mark the vertices with a *forbidden* flag during embedding when they have been used to embed a bit. A *forbidden* vertex cannot be used as C point in the future iterations (but it still can play a A or B role).

Similarly, marking a vertex as *forbidden* defines a *watermarked triangle site* because all its vertices are *forbidden* (the A and B vertices have been previously marked as *forbidden*). A manifold triangle mesh approximately contains one vertex for two triangles (by Euler relation and manifold hypothesis). We thus aim at watermarking approximately one out of two triangles ($N - 2$ sites).

3.2.2 Traversal algorithm

Our solution is inspired by [3] for implicit traversal of mesh connectivity for static compression purposes. The key idea is to grow and propagate edge cycles on the

[2] For a triangulated manifold mesh we have $2 \times E = 3 \times F$.

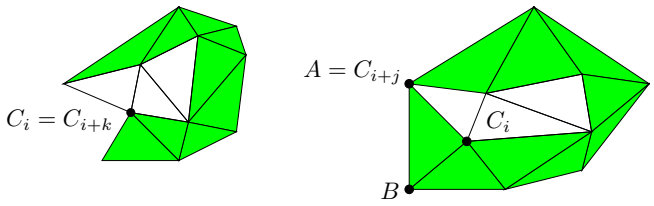

Figure 2: Causality issue in the traversal of the mesh connectivity graph during embedding. Left : vertex C_i was used at iteration i and further selected to be C_{i+k} at iteration $i+k$ thus erasing the information embedded in site i. Right : vertex C_{i+j} was previously playing the A role in the reference edge of iteration i (the same problem occurs for B). The information embedded at iteration i is erased because the reference edge would change at iteration $i+j$.

connectivity graph that finally conquers every vertices. We begin with the initialization of an edge AB (see Fig. 3). Vertices can be selected as C vertices (i.e. : in [3][1] they are *conquered*) iff :

- they are not marked as *forbidden* (causality issue),
- the reference edge of the site is composed of two forbidden edges (to maximize the number of sites).

3.2.3 Traversal discussion

Our algorithm uses a while loop. We must ensure it terminates, preferably in polynomial time. The first point is to prove that if all vertices are not forbidden, then there exists a non-forbidden vertex that belongs to a triangle with two forbidden vertices (this ensures our traversal terminates). Note that if the first point is true, then the traversal reaches exactly $N-2$ sites for watermark embedding (the two vertex coordinates of the initial edge can never be alterated). The second point is to find such a non-forbidden vertex efficiently.

Lemma 1 *At each iteration of the algorithm, there exists a non-forbidden vertex C of a triangle with two forbidden vertices.*

Proof: The first claim is that all forbidden vertices belong to a watermarked triangle. This is clearly true, since, except for the first two initialization vertices, when a vertex C is forbidden, we guarantee that the corresponding A and B vertices are already

```
Init first AB edge (mark A and B forbidden)

  While there exist a non-forbidden vertex

    Let ABC be an oriented triangle of M s.t.
      A and B are forbidden but not C

      Embed a watermark bit in ABC
      Mark C as forbidden

    EndLet
  EndWhile
```

Figure 3: Optimal traversal of the mesh with respect to our embedding procedure. The causality issue drives the traversal.

forbidden. The first watermarked triangle is generated after the first execution of the main loop.

Now, if all vertices are not forbidden, then let Q be such a non forbidden. From Q, we will find a vertex C satisfying the lemma conditions. If Q does not belong to a triangle with two forbidden vertices (see Fig. 4), then, we can draw a path between Q and a forbidden vertex, and this path has necessarily an edge between a non-forbidden vertex Q' and a forbidden vertex (call it B). If Q belongs to a triangle with two forbidden vertices, we let Q' to be equal to Q.

Then turning around B gives another path which contains (at least) one non-forbidden vertex (call it C) and (at least) two forbidden vertices (including B by previous claim that it belongs to a watermarked triangle). Thus this path contains an edge formed by a non-forbidden (C) and a forbidden vertex (call it A) and ABC is the searched triangle. ■

To implement the algorithm efficiently, we modify a little bit the algorithm, by maintaining a list of vertices belonging to at least one triangle with two forbidden vertices. When a vertex is forbidden, then all triangles incident to this vertex are examined and for triangles having only one non-forbidden vertex, this vertex is inserted in the list. Tagging the vertices as being in the list allows to avoid multiple insertions.

Lemma 2 *The algorithm takes linear time.*

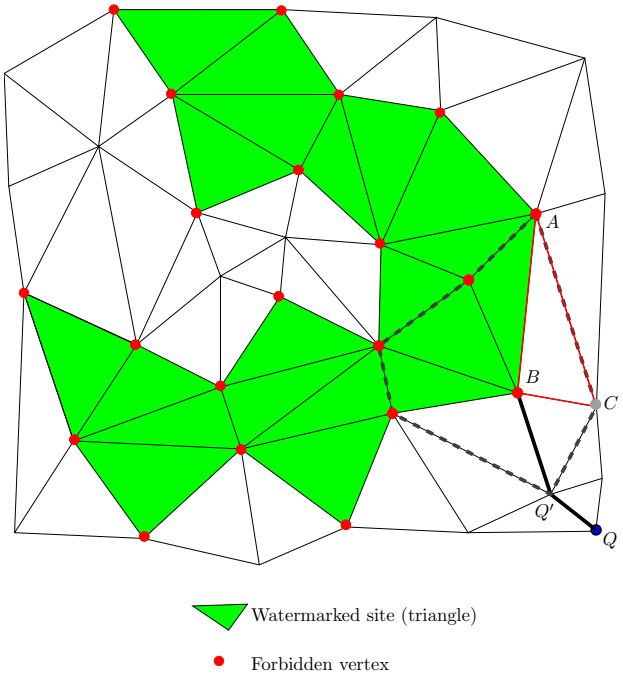

Figure 4: Proof of Lemma 1. At every iteration there exists a non-forbidden vertex that belongs to a triangle with two forbidden vertices.

Proof: Basically the algorithm contains two nested loops, one "for all vertices" and one "for all edges incident to the vertex". The previous lemma demonstrates that at every iteration of the algorithm there is a suitable vertex for embedding. This while loop turns in fact into a for loop. The second loop iterates on all edges incident to the vertex to dynamically fill in the active list. The overall complexity is thus the sum of the degree of all vertices, which is less than 6 times the number of vertices by Euler relation (Eq. 1). ■

We have prooved that our traversal reaches exactly $N-2$ valid sites for watermarking, in $O(N)$ time. In practice, we watermark only the sites that do not introduce a too big distorsion of the mesh. Thus this number of $N-2$ is not reached, but experiments show that the number of unmarked sites remains low (see line "waste" in Fig. 7).

3.3 Statistical modelling of the detector

From the previous statements, we ensure that about N triangles are watermarked. It becomes possible to statistically model the confidence of the watermark decoding. This is the goal of this section. To our knowledge, only few methods allow such a characterization, see [4] and [14] for example. We recall our decoding strategy : take the triangles one after the other to retrieve the watermark. No particular traversal is needed at decoding.

The decoding process consists in testing every three possible circular permutation for ABC in the examinated triangle to check wether Eq. 11 holds with secret key K :

- if Eq. 11 does not hold for any permutation of ABC, then the triangle is not watermarked,
- if it holds for one permutation, decode the number W^n of the hidden bit (Eq. 10), and the bit value W^v (Fig. 1).
- if it holds for more than one permutation, we take every decoding into account : false decodings will be considered as noise in statistical modelling.

3.3.1 Generating a reference distribution

At the end of the decoding process, we have an array of size $|W|$ composed of two fields : the number of 1-value detected, and the number of 0. If the mesh has been watermarked with secret key K and not tampered with, then the distribution of 1-

and 0-values will be highly perturbed by the embedding process. We have now to estimate on the suspect mesh what is statistically a non-watermarked mesh with K to perform blind decoding (the original mesh shall not be needed).

We found two ways to estimate what is a non-watermarked mesh with K :

- by decoding with false (probably reserved) keys, i.e. : other keys than K should estimate pretty well how this particular mesh behaves when it is not watermarked with K,
- by testing other values than $\frac{1}{2}$ in Eq. 11, i.e. : if we look in another watermarking space that is known not be used, this should also estimate what a reference mesh (non-watermarked with K) statistically is.

Obviously, the first solution is a security hole : an attacker could possibly get the reserved keys and optimize its attacks with respect to these particular keys. In this work, we shall use the second solution : we tested distributions centered around 0.1, 0.2, 0.3, 0.4, 0.6, 0.7, 0.8 and 0.9 (instead of $0.5 = \frac{1}{2}$ in Eq. 11).

To obtain a 0-centered distribution, we worked on the *signed* difference of the numbers of 1 and 0 read. Such a reference distribution is depicted on Fig. 5. We can generate as much samples as needed to perform accurate parameters estimation. We used Gaussian $\mathcal{N}(\mu, \sigma)$ modelling to describe the distribution of a non-watermarked mesh with K (see Fig. 5). Under an equirepartition assumption, we can define a false alarm probability $P_{fa} = 10^{-7}$. This gives us a threshold T_{hr} on the signed difference between the number of 1 and 0 that set the limit between watermarked and non-watermarked meshes. We employed a hard decision for decoding : the watermark bit with the lowest signed difference (in magnitude) engages the validity decision of the whole watermark recovery.

3.3.2 Tuning internal parameters

At this point, internal parameters T_1 and T_2 (Eq. 8) have been experimentally tuned to 10 and 4, respectively. We found that these parameters give a good equirepartition of original W^n, thus justifying the bits equirepartition hypothesis. To set up $|K_1|$ and $|K_2|$, we considered two contradictory constraints :

- The lengths of the subkeys must be big enough to ensure proper security of the embedding under a $P_{fa} = 10^{-7}$ constraint,
- The lengths of the subkeys must be small enough to ensure proper discrimination between two next keys, i.e. : the statistical imperceptibility (security) of the watermark shall be of size $|K|$ bits.

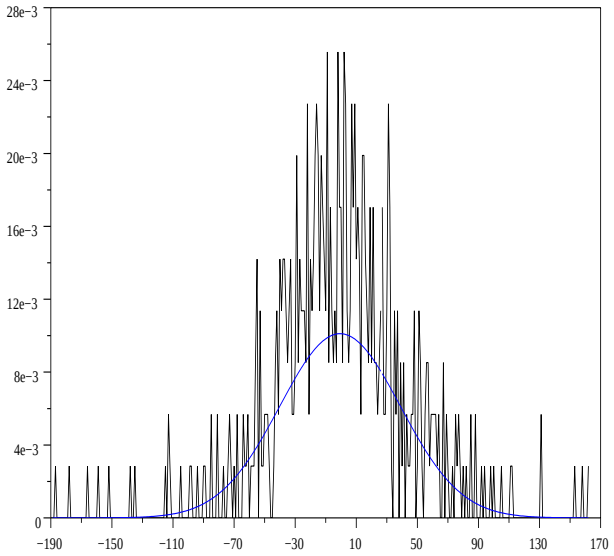

Figure 5: Reference distribution for the watermark detector. An arbitrary number of samples can be generated to estimate the (μ, σ) parameters of a Gaussian distribution that characterizes a non-watermarked mesh.

Obviously, the second constraint is stronger than the first. We then drive manually the choice of $|K_1|$ and $|K_2|$ to first satisfy the second constraint, and we check wether the first constraint is met.

Modelling of the second constraint is as follows : suppose the mesh is watermarked with secret K. If $|K|$ is badly chosen, Eq. 8 may lead to the same numerical result both with K and the next key in Hamming distance, namely $K_1 + 1 \oplus K_2$ (or $K_1 - 1 \oplus K_2$, $K_1 \oplus K_2 + 1$ and $K_1 \oplus K_2 - 1$). We have tuned $|K_1|$ and $|K_2|$ such that two next keys are well discriminated. We found using $|K_1| = 14$ bits and $|K_2| = 10$ bits is a good choice that meets both constraints (see detector output in Fig. 6). The payload may be classically encrypted with a 64-bit crytosystem, but the invisibility of the watermark is only of 24-bit security (oracle attack with watermarked meshes only).

4 Results

We give now some results on our watermarking scheme. We used heritage data to perform the testings [15]. We first illustrate the watermark localization on a watermarked mesh. Then we describe the experiments we tested our scheme against. Finally, we summarize the characteristics of our watermarking scheme in terms of accurate watermarking specifications : smallest protected surface (Minimum Watermark Segment, MWS), robustness class, embedding security and statistical modelling of the detector. We first verify that only the correct key leads to positive detector response (see Fig. 6).

4.1 Experiments protocol

Our fragile watermark is supposed to resist only RST transformations and cropping. Every test was performed with 10 keys. To test the first constraint, we proceed to a random parameters RST manipulation (i.e. : a combination of a random scaling, a random rotation and a random translation). The second constraint on cropping will actually allow us to estimate the size of the minimal amount of surface (MWS) that is needed to retrieve the watermark. The MWS will be given in the form of a number of (connected) triangles.

Still, we want to verify that even slight modifications of the mesh geometry erase the watermark. We thus tested our scheme against noise addition (with a magnitude of $10^{-2} \times r$, see Eq. 7), MPEG-4 3D Mesh Coding compression with standard 12

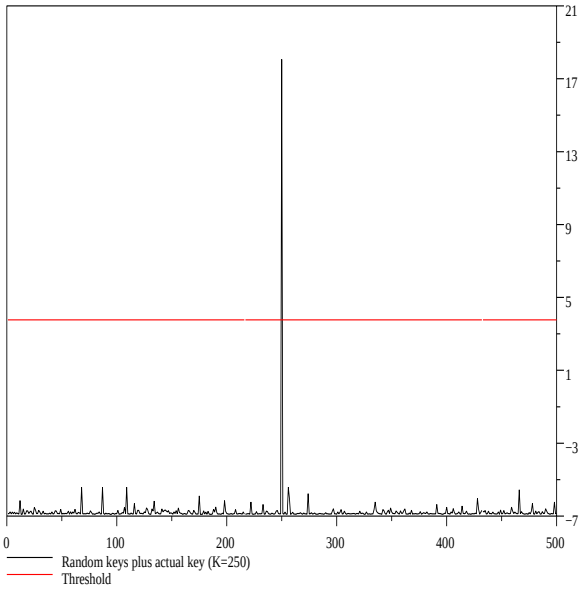

Figure 6: Detector output for various random keys plus the correct key $K = 250$. The threshold is 3.7665. Correct key detector output is 18.0722. The four next numerical keys ($K_1 + 1 \oplus K_2$, $K_1 - 1 \oplus K_2$, $K_1 \oplus K_2 + 1$ and $K_1 \oplus K_2 - 1$) have been included in the experiment.

bits error prediction quantization [16][17] and Laplacian smoothing [18] with only 2 iterations (denoted as LS in Fig. 7).

4.2 Experiments

We depict in Fig. 7 the results we obtained (10 keys each time) for various meshes. We report various parameters of the meshes like N and F, and the genus g. We verify that our traversal is not limited to 0-genus surfaces. In regard to N, we give the number of sites used at embedding. We can then assess the proportion of vertices wasted for distorsion reasons : it is always limited, when not negligible. Since we ensure equirepartition of the watermark bits, we can evaluate the number of times a bit is repeated (in sites/bit). The results demonstrate the accuracy of our scheme against its specified class of robustness (*only* RST transformations and cropping).

Model	Horse	Big_1	Big_2	African	Twins
N	48k	22k	46k	58k	83k
F	97k	43k	92k	115k	166k
g	0	0	0	2	1
Sites	47131	21683	45726	57611	83212
Waste	2.79%	0.43%	0.39%	0.05%	0.03%
Sites/bit	736.4	338.8	714.1	900.2	1300.2
MWS	600	580	590	560	550
RST	Y	Y	Y	Y	Y
Noise	N	N	N	N	N
MPEG-4	N	N	N	N	N
LS	N	N	N	N	N

Figure 7: Watermarking experiments using 5 meshes. Cropping allows us to estimate the size of the MWS. Our watermarking scheme is crop and RST resistant. Every other manipulation erases the watermark.

To evaluate the size of the MWS, we progressively limited the number of triangles we allowed for detection. We give here a lower bound for every mesh of the MWS size. However, the MWS does not depend on the mesh, but on the required probability of false alarm. The first mesh (horse) needs more triangles to recover the watermark, and we claim that the actual size of the MWS is 600 triangles for our watermarking scheme. This leads us to 0.11 bit/triangle mean local watermark density (or 0.22

bit/vertex by Euler relation). This still remains an acceptable performance, since the indexed localization ($N_b = 448$ bits, see Eq. 6) makes us actually hide 0.75 bit/triangle (or 1.5 bit/vertex). For the worst model (horse), the size of the MWS is 0.62% of F.

4.3 Algorithm summary

We have described a technical framework that enables precise design of fragile watermarking schemes for 3D triangles meshes. Both fragile watermarking setting and high-resolution data allow to use watermarking with geometrical invariants. However, maximizing the number of geometrical configurations (sites) that can embed a watermark bit is not trivial. We have proposed an optimal traversal with respect to our embedding procedure that makes the whole algorithm (decoding and embedding) terminate in $O(N)$. We have deduced a statistical modelling of our watermark decoder in order to bound the confidence in its output. Moreover, we have set up the length of the keys such that two next numerical keys are well discriminated. Cropping attacks allowed us to estimate the size of the MWS, thus leading to the mean watermark local density. We sum up the features shared by our algorithm in Fig. 8.

Complexity	$O(N)$ time
Watermark size (actual)	64 bits (448)
Robustness class	RST & crop
MWS	600 triangles
Watermark security	24 bits
Watermark density (actual)	0.22 bit/vertex (1.75)
False alarm probability	$P_{fa} = 10^{-7}$

Figure 8: Summary of the features shared by our algorithm. Robustness class is perfectly defined.

5 Further improvements

At present, we build on the flexibility of our framework to sketch other watermarking schemes designed towards watermarking performances optimization. Our fragile watermarking framework may be divided into two parts :

- an indexed localization embedding with respect to two precedently used vertices,
- an optimal traversal of the mesh connectivity graph that maximizes in $O(N)$ the number of sites valid for watermarking with respect to the causality issue.

First, we explain how to take advantage of our raw watermarking scheme to address integrity assessment of the mesh. Next, using the same optimal traversal, we will discuss another embedding procedure that would enable watermarking every triangle. Finally, we draw some considerations about watermark robustness against coordinates quantization.

5.1 Mesh integrity

We show on Fig. 9 the localization of the sites that the detector identifies as being watermarked. As mentionned before, nearly one out of two triangles contains a watermark bit. In Fig. 9, green triangles are those for which one of the three possible circular permutation over ABC makes Eq. 11 hold. From our traversal, it follows that not all triangles sharing the same vertex can carry a watermark bit. However, this case occurs in Fig. 9 : this is because Eq. 11 sometimes holds in triangles by itself (counting for noise in the statistical modelling). Mesh integrity could then be assessed in two steps :

- compute the ratio r_{Int} of watermarked sites over $N-2$,
- if it is too low, proceed to visual inspection.

For example, if r_{Int} is lower than 90% display watermarked and non-watermarked sites as in Fig. 9. If only a little part of the mesh has been tampered with, then there will be a connected set of triangles were Eq. 11 never holds. Such a tampered area will then appear in red to the human inspector.

To avoid such a human actor in assessing mesh integrity, one could employ morphological dilation and erosion over the binary watermarked or non-watermarked vertex space. Accurate assessment of the mesh integrity would then return a list of the size of the non-watermarked areas (in triangles or vertices). If the smallest non-watermarked area is inferior to a certain integrity threshold, then the mesh is declared to have been tampered with, and visual inspection may help locate the attack (and therefore evaluate its semantics).

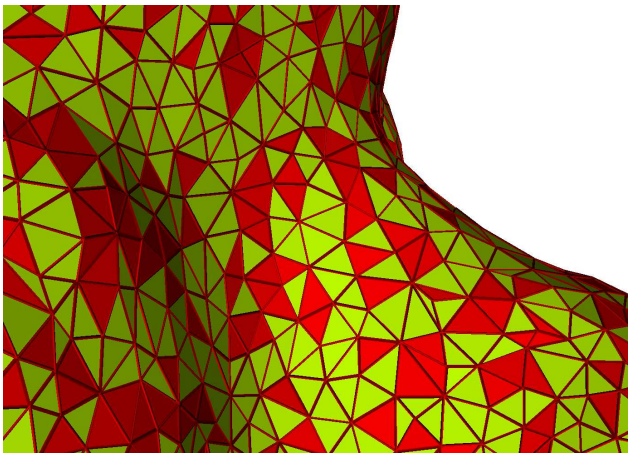

Figure 9: Localization of the watermark at the detection stage. Watermarked triangles (sites) are in green, non-watermarked triangles are in red. Approximately one out of two triangles carries a watermark bit.

5.2 MWS optimization

We now explain how to take advantage of our optimal traversal to embed a watermark bit in every triangle. This would increase the local watermark density by decreasing the number of required triangles to recover the watermark (MWS).

5.2.1 Embedding procedure limitation

When we commit geometrical embedding, it is always with respect to exactly two vertices that are already used : we embed a watermark bit in exactly one triangle at every iteration of our mesh traversal. This maximizes the number of sites for our method, but only reaches approximately one out of two triangles. We will see that our traversal is *really* optimal, as it allows with no change designing of other indexed localization embedding procedures.

To illustrate our concern, let the iteration of our traversal depicted in Fig. 4 be followed by the iteration in Fig. 10. It is obvious from Fig. 10 that alteration of C coordinates engages not only ABC but four triangles (denoted 1, 2, 3, 4 in Fig. 10, including ABC). We then strive to another slightly different embedding algorithm that relies on exactly the same active list implementation. The first ingredient consists in being able to embed two bits in two triangles ABC and $A'B'C$ incident to the same vertex C. Indeed, hiding a given bit in ABC corresponds to some allowed

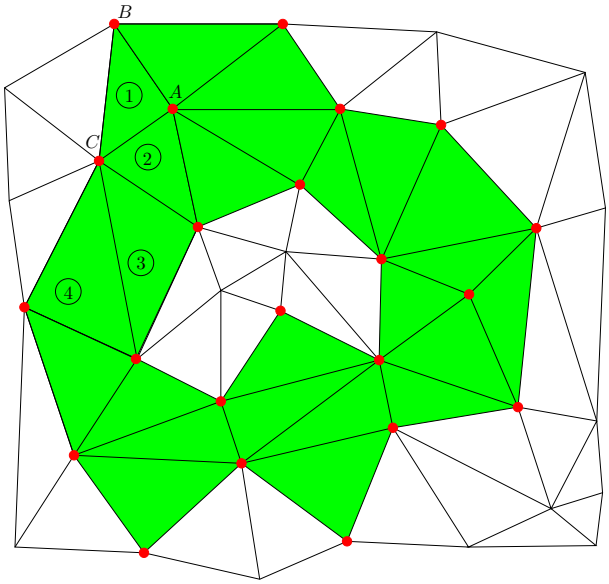

Figure 10: With respect to the previous iteration in Fig. 4, the iteration depicted here modifies 4 triangles. Up to now, we miss the opportunity to embed a bit in 3 of them.

```
Init first AB edge (mark A and B forbidden)

  While there exist a non-forbidden vertex

    If there exist ABC and A'B'C two oriented
       triangles of M s.t. A,B,A' and B' are
       forbidden but not C

        Watermark triangles ABC and A'B'C
        Mark C as forbidden

    Otherwise
        Watermark only one triangle ABC with
        A and B forbidden but not C
```

Figure 11: Embedding procedure constraints with respect to our optimal traversal. Optimal embedding require to watermark every triangle in U for which C is the last non-forbidden vertex. We keep exactly the same active list implementation.

position for C in the plane of ABC, that is some cylindrical region R in 3D space (by symetry of revolution around AB). Similarly, we can construct the allowed region R' for hiding another bit in $A'B'C$. If the quantization step of the watermarking space is small enough, R and R' necessarily intersect and thus the two bits can be embedded simultaneously.

The second ingredient is a slight modification of algorithm of Fig. 3 (depicted in Fig. 11) : instead of watermarking any non forbidden vertex in the active list, we will give preference to vertices incident to at least two triangles with two forbidden vertices, if there are some. In that way, at few steps we will hide only one bit and in other cases we will hide two bits simultaneously.

6 Conclusion

We have presented a new fragile watermarking framework for 3D triangle meshes that enables precise and flexible design of watermarking schemes running in $O(N)$ time. We proposed a scheme dedicated to authentication and integrity of high-resolution meshes purposes. Its watermarking specifications are discussed in terms of MWS

(Minimum Watermark Segment), class of robustness and probability of false alarm. Such an analysis was possible because of our optimal mesh traversal and the use of an indexed watermark localization. Furthermore, we finally sketched new schemes in order to optimize integrity assessment, MWS size and watermark robustness against coordinates quantization.

Acknoledgements

This work was completed thanks to fruitfull discussions inside INRIA ARC TéléGéo. Experiment stereovision heritage meshes from Carlos Hernandez [15] are available at : `http://perso.enst.fr/~hernand/repository/` .

References

[1] P. Alliez and M. Desbrun, "Progressive compression for lossless transmission of triangle meshes," in *Proceedings of SIGGRAPH 2001*, 2001, pp. 198–205.

[2] S. Valette, H.-Y. Kim, H.-Y. Yung, I. Magnin, and R. Prost, "A multiresolution wavelet scheme for irregularly subdivided 3d triangular mesh," in *Proc. of IEEE Intl. Conf. on Image Processing ICIP'99, Oct. 25-28, Kobe, Japan*, 1999, vol. 1, pp. 171–174.

[3] C. Touma and C. Gotsman, "Triangle mesh compression," in *Proceedings of Graphics Interface*, 1998, pp. 26–34.

[4] E. Praun, H. Hoppe, and A. Finkelstein, "Robust mesh watermarking," in *Proceedings of SIGGRAPH*, 1999, pp. 49–56.

[5] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukayiama, "Watermarking 3d polygonal meshes in the mesh spectral domain," in *Proceedings of Graphics Interface*, Jun 2001, pp. 9–17.

[6] J.-D. Boissonnat and M. Yvinec, *Géométrie Algorithmique*, Édiscience international, 1995.

[7] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics," in *Proccedings of SIGGRAPH*, 1997, pp. 209–216.

[8] L. Kobbelt, J. Vorsatz, U. Labsik, and H.-P. Seidel, "A shrink wrapping approach to remeshing polygonal surfaces," in *Proc. Eurographics*, 1999, pp. 119–130.

[9] C. Loop, "Smooth subdivision surfaces based on triangles," M.S. thesis, Dept. of Mathematics, Univ. Utah, 1987.

[10] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models," in *Proceedings of ACM International Conference on Multimedia*, Nov 1997, pp. 261–272.

[11] F. Cayre and B. Macq, "Data hiding on 3d triangle meshes," in *IEEE Trans. on Signal Processing*, 2003, vol. 51, pp. 939–949.

[12] B.-L. Yeo and M.M. Yeung, "Watermarking 3d objects for verification," *IEEE Computer Graphics and Applications*, vol. 19, no. 1, pp. 46–55, 1999.

[13] N. Johnson and S. Jajodia, "Exploring steganography : seeing the unseen," *IEEE Computer*, vol. 31, no. 2, pp. 26–34, Feb 1998.

[14] O. Benedens and C. Busch, "Towards blind detection of robust watermarks in polygonal models," in *Proceedings of EUROGRAPHICS*, 2000, vol. 19, pp. 199–208.

[15] C. Hernandez and F. Schmitt, "Multi-stereo 3d object reconstruction," in *Proc. 1st Intl. Symposium on 3D Data Processing Visualization and Transmission (3DPVT)*, 2002.

[16] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," in *ACM Transactions on Graphics*, Apr 1998, vol. 17(2), pp. 84–115.

[17] A. Guéziec, G. Taubin, F. Lazarus, and W. Horn, "Cutting and stitching : Converting sets of polygons to manifold," *IEEE Trans. on Visualization and Computer Graphics*, vol. 7, no. 2, pp. 136–151, Apr-Jun 2001.

[18] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of SIGGRAPH*, 1995, pp. 351–358.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399