Data Hiding on 3-D Triangle Meshes

François Cayre and Benoît Macq, *Senior Member, IEEE*

***Abstract*—In this paper, we present a new scheme for digital steganography of three–dimensional (3–D) triangle meshes. This scheme is robust against translation, rotation, and scaling operations. It is based on a substitutive procedure in the spatial domain. The key idea is to consider a triangle as a two-state geometrical object. We discuss its performance in terms of capacity, complexity, visibility, and security. We validate the use of a principal component analysis (PCA) to make our scheme signal-dependent in the line of second generation watermarking scheme. We also define a simple specific metric for distortion evaluation that has been validated by many tests. We conclude by giving some other solutions, including open steganographic schemes that could be derived from the basic ideas presented here.**

***Index Terms*—Binary MEP, fragile watermarking, geometrical state, mesh, steganography, triangle, TSPS.**

I. Introduction

STEGANOGRAPHY, the art of reliably hiding a message in another, has been widely used over the centuries for analog media but today is being for digital multimedia content. There are many applications for which steganography is a suitable solution, ranging from in-band captioning and side-information channeling to authentication and tamper proofing. Our main goal here is to present a new simple steganographic system designed for three-dimensional (3-D) triangle meshes.

Over the past few years, 3-D hardware has become much more affordable than ever, allowing the widespread use of 3-D meshes from CAM/CAD industry into video games and other end-user applications. Three-dimensional meshes have become of great interest since they are widely used. They are also a type of multimedia content, which in some cases has to be enhanced using steganographic techniques. Our main goal here is to present a new simple steganographic system designed for 3-D triangle meshes, extending and enriching one of the simplest technique called the triangle strip peeling sequence (TSPS). From a geometrical point of view, one could see this scheme as a quantization index modulation (QIM) scheme extended on a discrete partition of a physical measurement.

The basic idea behind the TSPS algorithm is the insertion of bits while moving on the mesh. Nevertheless, the original TSPS implementation [3] suffers from two majors drawbacks. It is not fully automatic as some kind of human intervention is needed. It is unsecure as TSPS is based on the modification of the topology so that an opponent easily locates the payload. We propose a new way of modifying the triangle characteristics in order to obtain a more secure algorithm and a fully automatic implementation. We also show how to achieve the reversibility of the method, which is often desirable since 3-D meshes are sometimes results of measures (e.g., scientific data or medical images) or specifications for industrial processes (CAD data).

Triangle meshes are a general representation of a 3-D visual object that are very well suited to communications. Even if the vizualization process uses a different representation (see NURBS [4] as an example) or if the object is acquired in a specific representation, it is possible to derive a triangulated representation of the object in any cases. Triangles offer many steganographic possibilities through modifications of their elementary features, vertices (geometry), and connectivity (topology). The 3-D object appears as a list of elementary connected objects (triangles), which is very different from the usual regular sampling of pixels in use in photographic and video imagery. Here, we first exploit the properties of the graph representing the mesh to find where to hide the bits. Then, as an isolated triangle may be seen as a two-state object, we flip its state according to the bit to be hidden in this location. Robustness is another difference with 2-D media. Global geometrical manipulations (scaling, translation, rotation) are not easily tackled when working on sampled images. Yet our scheme provides natural robustness to affine transformations.

II. Background

As usually denoted in steganography, the media hiding information is called the *cover* media, and the information to be hidden is referred to as the *payload*. From a protocol point of view, one distinguishes between schemes that need the original cover media to recover the payload and the *blind* schemes that do not need it. Blind schemes are of special interest as they provide automatic retrieving of the payload without any kind of assistance. From a signal processing point of view, the schemes are classified in *additive* or *substitutive* schemes. Additive schemes are related to the fact that the payload is first coded in a signal simply added to the cover media. The substitutive schemes flip binary features of the cover content in order to encode the payload (for example, one could modify the phase of a cover signal so that it is either 0 or π). Our scheme is a substitutive blind scheme in the spatial domain.

Several watermarking schemes for meshes have been proposed either in the spatial domain [3], [4] or in a transform domain [5], [16], [17]. Embedding in the spatial domain naturally leads to a relatively poor robustness, whereas capacity increases. In the general framework of watermarking, transform domains

Manuscript received February 4, 2002; revised December 3, 2002. This work was supported by the European Union IST-1999-10987 project CERTIMARK. The associate editor coordinating the review of this paper and approving it for publication was Prof. Edward J. Delp.

F. Cayre was with the Communications and Remote Sensing Laboratory, Université Catholique de Louvain, 1348 Louvain-la-Neuve, Belgium. He is now with the Département Traitement du Signal et des Images, ENST Paris, 75684 Paris, France (e-mail: cayre@tsi.enst.fr).

B. Macq is with the Communications and Remote Sensing Laboratory, Université Catholique de Louvain, 1348 Louvain-la-Neuve, Belgium (e-mail: macq@tele.ucl.ac.be.

Digital Object Identifier 10.1109/TSP.2003.809380

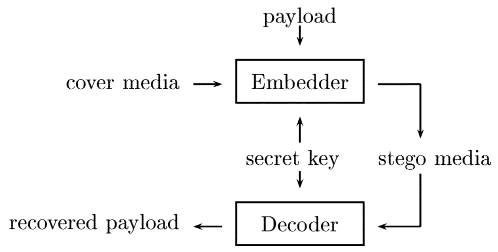


Fig. 1. Steganographic paradigm for secret-key schemes. The secret key is used to embed the payload into the cover media, and the decoder needs the secret key to extract the recovered payload from the stego media.

have shown to offer a better robustness. Since we are interested in maximizing capacity, we plan to embed the payload into the spatial domain.

We present in Fig. 1 the basics of a steganographic system. The system is build around two major blocks: the encoder and the decoder. The encoder takes as input the cover media, the payload, and the secret key. It produces a *stego*- media. The decoding block takes the stego-media and the secret key to extract the payload. From one block to another, the stego-media is supposed to go through an undefined communication channel, and the payload is supposed to be retrieved even after some degradations.

To evaluate a stego-system, one establishes results on imperceptibility, capacity, complexity, and security. In the case of well-known media such as sounds or photographic images, a simple measure is used for distortion: the signal-to-noise ratio (SNR). This measure is dedicated to regularly sampled signals, which is not the case of 3-D meshes. In our case, there are several proposals to address the issue of measuring distortion. The simplest is the Haussdorff distance, which is based on an infinite norm. Another proposal was made by Karni and Gotsman [14] to handle meshes with different topologies. Since our scheme does not change the topology, we will use an estimation of the Haussdorff distance to evaluate the distortion induced by the embedding process.

In the steganographic framework, one usually maximizes the size of the payload, whereas in watermarking, there is a tradeoff to commit between capacity and robustness. We assume no robustness requirements, except trivial operations such as rotation, scaling, and translation. Capacity is our main goal. We give an upper bound on the maximal capacity of our scheme.

We estimate the complexity of our scheme by giving common processing times for the well-known *bunny* model. The time used to embed and to retrieve the payload is the same, as both operations are symmetrical. Finally, as the aim of steganography is to hide a secret payload, we are interested in the security level provided by our algorithm. We only give an estimate since a practical verification of theoretic paradigms is untractable in our case. Our scheme can easily be shown to be resistant against any exhaustive search.

III. Presentation of the Proposed Algorithm

We keep the basic TSPS idea of encoding a payload by moving over the mesh, which implies that we always see a triangle with its associated entry edge and two possible exit edges, as in Fig. 2. Our algorithm needs oriented triangles to proceed.

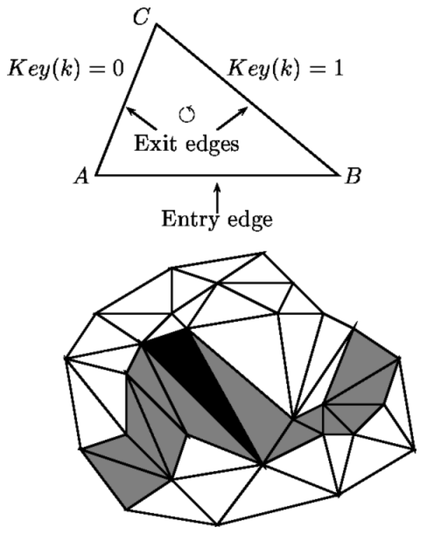


Fig. 2. Top: Triangle viewed from a topological setting: the entry edge (*AB*) corresponding to the current bit to insert and the two possible exit edges (*AC* and *BC*). Exit edges ordering is made with a clockwise criteria (the first one is *BC* here). Bottom: Original mesh with TSPS path (gray). One triangle carries one bit. The cell on which we will perform MEP is in black. Geometrical distortion: none. Topological singularity: stencil.

A. Overview

Our algorithm requires two steps: First, a list of triangles of the mesh that will contain the payload is established. This operation is driven by the secret key of the steganographic process. Second, each so-called *admissible* triangle of the list is modified or not according to the binary symbol it has to convey. This last operation is denoted as a macro embedding procedure (MEP) [3].

1) Listing Triangles to be Processed: The list of triangles is established according to the scheme of Fig. 2. A starting triangle is determined on the basis of a specific geometric characteristic (cf. infra). The next triangle in the list is either the first (its new entry edge being AC) or the second one (its new entry edge being BC) in clockwise order, depending on the bit value of the key. The length of the key must be as long as the list of admissible triangles required to convey the payload. The key may also be the seed of a binary pseudo-noise sequence generator. The path of visited triangles is called the *stencil*.

2) Macro Embedding Procedure (MEP): Each triangle is considered as a two-state object. We define the state of the triangle by the position of the orthogonal projection of the triangle summit C on the entry edge *AB*. We denote this position as $P(C)$. We divide the *AB* interval into two subsets S_0 and S_1. If $P(C) \in S_0$, then we consider that the triangle is in a "0" state; otherwise, $P(C) \in S_1$, and the triangle is in a "1" state. If the S_0 and S_1 subsets contain intervals defined as fractions of *AB*, states are invariant to an affine transform of the mesh. To set the triangle in the i ($i = 0$ or 1) state, two cases occur:

- $P(C) \in S_i$: No modifications need be processed.
- $P(C) \notin S_i$: C has to be shifted toward C' so that $P(C') \in S_i$.

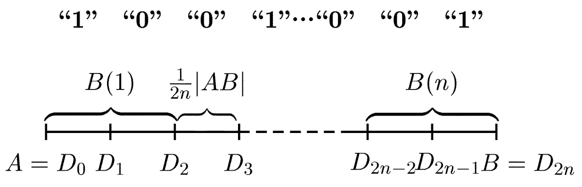


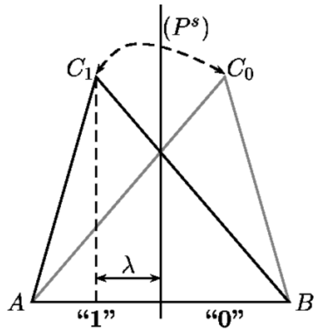
Fig. 3. Decomposition of the entry edge *AB* into two interleaved subsets S_0 and S_1, with the $2n$ binary values for every D_kD_{k+1}.

The $C \rightarrow C'$ mapping has to be reversible and invariant through affine transformations. Moreover, $|C - C'|$ has to be small enough to avoid visual degradation of the mesh but large enough to allow accurate payload detection. We choose to divide *AB*, our steganographic space, into subintervals and to use interval borders as a symmetry axis. The $C \rightarrow C'$ mapping is a symmetry accross the closest axis orthogonal to *AB* that intersects the border of the closest sub-interval belonging to S_i (see Figs. 4 and 5).

A string containing bits indicating whether a change was needed for each admissible triangle allows the data hider to retrieve an unmarked perfect copy of the original mesh. We call this string the erasing key. It can be seen as a hash of the mesh parameterized by the secret key and the payload. When the random selection of triangles leads to an admissible triangle, we can insert a bit of hidden information. The triangle can be geometrically modified or not, depending on the difference of the bit value to be hidden and the initial state of the triangle. If one wants to be able to recover the original state of the admissible triangle, it is necessary to store an erasing bit for every bit of the payload: The erasing bit is set to 1 if the state of the triangle changed and 0 otherwise. The erasing key is therefore constructed during embedding, and its size is exactly the same as the payload size.

B. Partition of the Entry Edge Into S_0 and S_1

A graphical summary of the partition is sketched in Fig. 3. The aim of this decomposition is to extend the QIM concept to 3-D triangle meshes [12], [13]. Our quantizer is implemented by dividing *AB* by $2n$. Compared with common QIM implementation, we chose to hide only one bit per vertex (C), although it is possible to hide more. Moreover, we were interested in keeping reversibility. In Fig. 3, we establish the main outline of our steganographic space. The D_k are the frontiers between two elementary binary domains. The $B(k)$ are the n different possible bit locations over the entry edge, each constituted of two elementary binary domains. The value $(1/2n)AB$ can be seen as the basic quantization step in the QIM method. The projection of C on *AB* will select both a $B(k)$ and a D_pD_{p+1}, which is the bit value of the triangle. The alternating numbering 0, 1 and1, 0 of the $B(k)$ is the most suited for robust steganography. The interleaved sets S_0 and S_1 are shown on the top of Fig. 3. The D_k points represent the borders between elementary binary domains. To every segment $B(k)$ is assigned a value from the two domains of which it is constituted. The aim of this regular decomposition is to provide us with an interleaved binary distribution over the entry edge, thus minimizing the distance between C and C' in case of a mapping. We extend this partition of the entry edge to the whole line defined by *A* and *B*. This way,



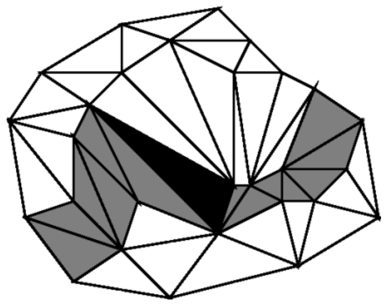


Fig. 4. Top: First-order MEP ($n = 1$). Two geometrical configurations. This case degenerates to a simple symmetry. Bottom: First-order MEP on black cell, same path. Does not show steering symbols. Geometric distortion: sometimes, strong. No topological singularity.

we can handle triangles for which the projection of C falls out of the segment *AB*.

IV. BINARY MEP

In this section, we detail some features of the MEP. This MEP has to be considered as an example, where other physical changes are possible.

A. MEP: Projection of the Summit on the Quantized Base

Let us first define the planes ($P^s_{k,n}$), $1 \leq k \leq n$, which are all orthogonal to *AB*:

$$M \in \left(P^s_{k,n}\right) \Leftrightarrow \overrightarrow{MD_{2k-1}} \cdot \overrightarrow{AB} = 0.$$

All these planes are placed on a regular grid of *AB*. They cut every $B(k)$ into two intervals: [$D_{p-1}D_p$] on the left side and [D_pD_{p+1}] on the right side. The aim of the regular decomposition is to provide a set of elementary domains over the entry edge. Each domain D_pD_{p+1} is defining a place where the bit inserted within the triangle is said to be "1" or "0" (see Fig. 3).

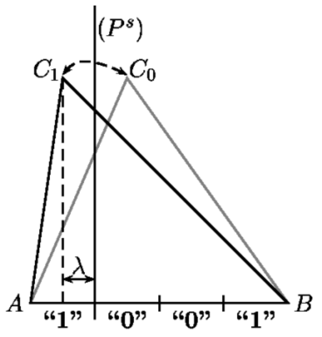
We then compute all the distances λ_k, $1 \leq k \leq n$, from C to all ($P^s_{k,n}$), and we let

$$\lambda = \min_k \{\lambda_k\}.$$

We denote (P^s) the corresponding ($P^s_{k,n}$). We thus can state the boundaries of λ:

$$0 \leq \lambda \leq \frac{1}{2n}|AB|. \tag{1}$$

From now on, we know on which $B(k)$ to focus. The link between λ and $B(k)$ is straightforward, as the projection of C on *AB* now selects both a $B(k)$ and a $D_{2p}D_{2p+1}$ (or a



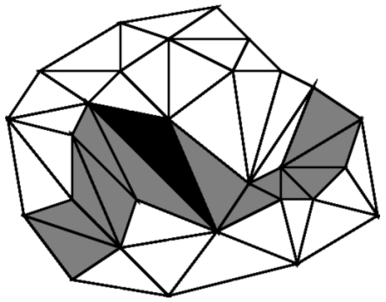


Fig. 5. Top: Second-order MEP ($n = 2$). Four geometrical configurations: Geometrical distortion decreases. Bottom: Second-order MEP on black cell, same path. Does not show steering symbols. Geometric distortion: Sometimes (generally does not happen on the same triangles as in the first order case), smoother. No topological singularity, better visual quality.

$D_{2p+1}D_{2p+2}$). λ represents the minimal amount of geometrical distortion to be introduced for a change of state.

Let us note that n parameterizes the smoothness of the algorithm. As n increases, λ decreases so that the amount of distortion to be introduced gets smaller. On the other hand, the number of frontiers between domains increases, which can cause bit retrieval errors due to the limited machine precision. In our graphical simulations, we only considered the cases $n = 1$ and $n = 2$ (see Figs. 4 and 5), although the MEP algorithm does not imposes no restrictions on the order.

Usually, watermarking schemes are divided into two categories: additive or substitutive. The method we propose here could be additive (λ is just the amount of minimal distortion to be introduced in the scope of changing the triangle state), but we preferred a substitutive version, which is simple to implement and fully reversible (given an *erasing key*).

Keeping this frame in mind, we define our MEP to be the *symmetry* of C with respect to (P^s), hence, the superscript s. If the triangle is in the correct state, no geometrical distortion is introduced; otherwise, only a (potentially) small reversible shift is performed.

B. Erasing Key

As explained in Section III-A2, we build at every iteration the erasing key that should be kept private. For every payload bit, we define its corresponding erasing bit. The erasing bit k is at "1" if the embedding process of the payload bit number k performs a binary-MEP and "0" if it keeps the triangle in its original state. The erasing key has the same length as the payload size since exactly one triangle is chosen to be inserted into each bit of the payload. It is just the bit-wise difference between the payload and the mesh along the stencil defined by the secret key. By this way, exact reversibility is *only* granted to the erasing and secret keys' holder.

C. Simulations

We present here simulations of the two first orders of the binary MEP. Even if n could be arbitrarily high, we will generally not consider high orders of the MEP. Tests have been performed with order 8.

1) Simplest: First-Order MEP: Let (P^s) be the symmetric plane for the entry edge *AB* (see Fig. 4). The point C is called C_0 if we embedded a "0"; else, C_1. We build a binary-MEP by moving C symmetrically with respect to (P^s) from one side of (P^s) to another when needed. This is a simple way to make a reversible bit commitment. This MEP has only two geometrical configurations. Thus, geometrical distortions are quite strong. We give a simulation on a two-dimensional (2-D) mesh of the MEP in Fig. 4. For a better understanding, the black cell on which we perform the MEP is always in the reference plane.

2) Smoother: Second-Order MEP: We move C with respect to its closest plane (P^s), which is always closer to C than in the previous case. This MEP has four geometrical configurations and still hides one bit. It is smoother because distortion is always less important than in the previous MEP, due to the higher number of geometrical configurations. We show in Fig. 5 a simulation of this MEP on the black cell. Geometrical distortion gets smaller as n increases.To choose the next triangle to be added to the stencil, simply read the value of the bit k of the key and deduce the *key-edge*. This exit edge becomes the next entry edge of the next triangle. The edge that was not selected with the key is called the *steering edge*. We show in Fig. 6 the visual effect of the MEP transformation when the MEP order is 4. We choose colors to indicate the kind of transformation of the triangles, blue being the largest ones (see Fig. 7, for the pseudo-code of the algorithm, where binary-MEP operations are performed only on admissible triangles.).

V. Moving on the Mesh

Let us recall the general mainframe of the method: The bits are hidden one after the other in a list of triangles with a geometrical MEP. So far, we only mentioned that the secret key drives the stencil through the mesh. This section essentially deals with the way of moving over the mesh. We first explain why we no longer need peeling. Finally, we address the problem of finding the initial cell, which is the *synchronization problem*.

A. Avoiding Peeling

First, we precise the method we use to move across the cells. In our case, the criteria for choosing the next cell to add to the stencil is not based on topological peeling. Our rule is geometrical: Using the clockwise distinction between the two exit edges, we always get out of the cell using the exit selected by the secret key. This way, we no longer need to peel off the stencil, which is more secure.

When selecting the next triangle, we use a general *rule* that may be overridden by an exception case detailed. This rule is to

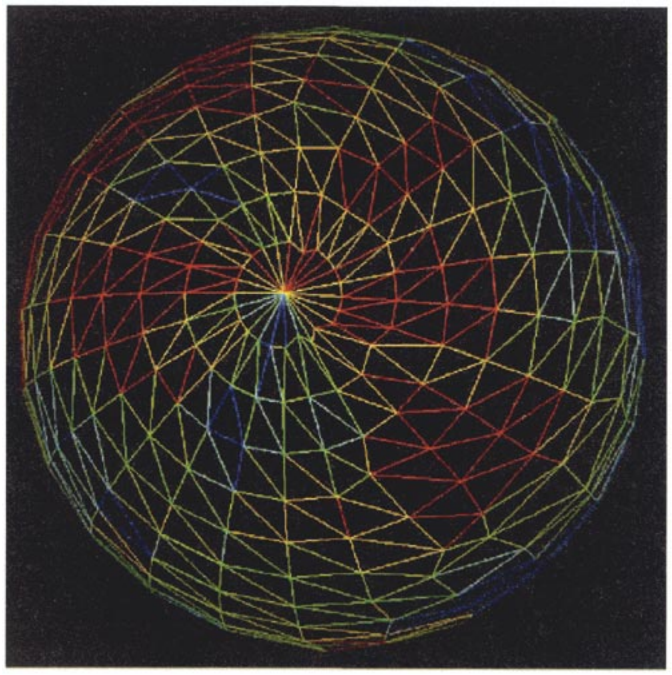


Fig. 6. Stego-sphere with 256 bits embedded (48.3% of total capacity). The order of the MEP is 4. This trivial example is provided to illustrate the effect of the MEP on real meshes. The cover mesh was a regularly sampled sphere. A coloring of the triangle was performed with respect to the number of times it was used for a bit commitment by the algorithm. The maximum value is 3, blue, and the minimum is 0, red. One could see this coloring as the local degree of presence of the information: Satured places are in blue.

always get out of the current triangle using the exit edge corresponding to the current secret key bit. If the secret key bit is equal to 1, we use the first edge, and if it is equal to 0, we use the second edge. It is the geometrical replacement to the topological rule in the original TSPS method [3] (i.e., *stay on the stencil*).

B. Automatic Steering

To address the issues raised by the topological singularities of the mesh (holes, etc…), we use the so-called steering edge (which is the equivalent of the steering symbols in [3]). This is a way to keep the stencil as long as possible; however, multiple stencils achieve the same results. We automatically insert steering symbols when needed; therefore, the stencil is uninterrupted. This is the only exception to the general rule. In our setting, steering corresponds only to topological singularities of the cover mesh. We limited our use of the steering-edge to simple cases, but virtually all exceptions can be handled this way. Our implementation can process both manifold and non-manifold meshes.

C. Admissibility and Upper Bound for Capacity

In the classic TSPS algorithm, the stencil is strictly topologically separated from the mesh, whereas we materialize the frontier of the stencil by creating a list of the already-used points. At every iteration, we add to this points list the vertex C used to hide a bit, which is either modified by an MEP or not. This list contains the points that correspond to a hidden bit of information. They cannot be moved again during the embedding process. If later on during embedding the secret key produces a path that

```
Hide ( mesh, payload[nBits],

       key[nBits], initCellAndEdge )


While ( i < nBits ) do

  If ( currentCellIsAdmissible )


#EMBED PAYLOAD

#    If ( readCellState != payload[i] )

#      MEP ( currentCell )

#      eraseKey[i] = 1

#    End If


#RETRIEVE PAYLOAD

#    payload[i] = readCellState


    exitEdgeNumber = key[i]

    i += 1

  Else

    exitEdgeNumber = getSteeringEdge

  End If


  nStencil += 1

End While

End Hide
```

Fig. 7. Pseudo-code of the algorithm. Binary-MEP operations are performed only on admissible triangles. We show how to build the erasing key dynamically.

selects once again a forbidden point in this list, the triangle is not admissible. In this case, we simply go on through the mesh with respect to the key without considering the triangle for embedding or decoding. Since both sides of the algorithm are symmetrical, we maintain this list both at the embedding and decoding side. This way, we protect the stencil by materializing it into a dynamically created forbidden points list.

We can then deduce the upper bound for capacity. If we let the algorithm run long enough, it will add every point of the mesh into the forbidden points list. The final number of points in the forbidden points list is the number of points in the mesh. Since every point inside this list corresponds to a bit of information,

Results (performed on 11 models)					
Cover	**verts**	R_{fill}	R_{cod}	R_{dist}	**Kbits**
horse	48485	*84.5%*	6.5	0.15%	**40**
buddha	32328	*79.2%*	7.1	0.19%	**25**
face	26460	*77.4%*	7.3	0.19%	**20**
bunny	14007	*87.7%*	8.4	0.22%	**12**
inopl	9831	*83.3%*	7.5	0.23%	**8**
venus	8016	*89.4%*	8.1	0.22%	**7**
hand	6650	*61.6%*	6.3	0.21%	**4**
woman	7723	*53.0%*	6.2	0.23%	**4**
anttoon	7082	*57.8%*	5.9	0.18%	**4**
skull	11051	*37.1%*	7.8	0.24%	**4**
pelvis	4189	*48.9%*	7.6	0.25%	**2**

Fig. 8. Results for various models. For testings, we chose the fourth MEP order. In practice, MEP order has shown to be of no significance regarding the bit error rate. No errors were found in the recovered payloads.

the theoretical upper bound for the capacity of our scheme is the number of points inside the cover mesh. Let N_{bits} be the number of bits actually embedded, and let N_{verts} be the number of vertices in the mesh. We can state the following constraint on our scheme:

$$N_{\text{bits}} \leq N_{\text{verts}}. \tag{2}$$

In order to quantify the remaining space, we then define the so-called filling rate:

$$R_{\text{fill}} = \frac{N_{\text{bits}}}{N_{\text{verts}}} \tag{3}$$

We provide typical results in Fig. 8. Those results are discussed in Section VII. A visual example is given in Fig. 9, where the capacity is 87.7% of the total capacity. One could use this filling rate beforehand to set up the algorithm with the desired capacity. We keep this simple measure as the capacity of our algorithm. The only issue remaining now is synchronization, thanks to the fact that the algorithm produces no topological singularity to facilitate the recovery of the original stencil.

D. Topological Boundaries: Initial and the Final Cells

The last issue we have to address is to uniquely determine the initial cell and edge. Since the information is no longer topologically separated from the mesh, we have to find the initial cell using some local information. This is a synchronization problem. This information has to be local because the initial cell is essentially local and geometrical because there is no longer any peeling. A content-based approach has been followed in [20]; we will use this path as another possible synchronization procedure. In the literature [10], some solutions are proposed and divided into geometrical and topological approaches. One may select as the initial cell the triangle of lowest or greatest area (geometrical synchronization setting) or ask the initial cell to have a certain property in terms of connectivity such that the subset of matching cells is very small. A content-based proposal can be found in [20], and we will follow this path as another possible synchronization procedure. We used two different techniques. The first one is based on triangle areas as we selected the triangle of smallest area for geometrical distortion reasons. The second one is based on a principal component analysis (PCA) that gives three principal axes centered on the gravity center whose intersections with the mesh leads to a small (6) number of possible initial cells. The former is very quick, compared with the later, but we check the speed of the *raw* method by imposing the index of the initial cell at embedding and retrieving (arbitrary initial cell). Performing a PCA is a way to link the payload to the content of the mesh. Such an operation can be performed in other concerns [25], like mesh indexation, for example.

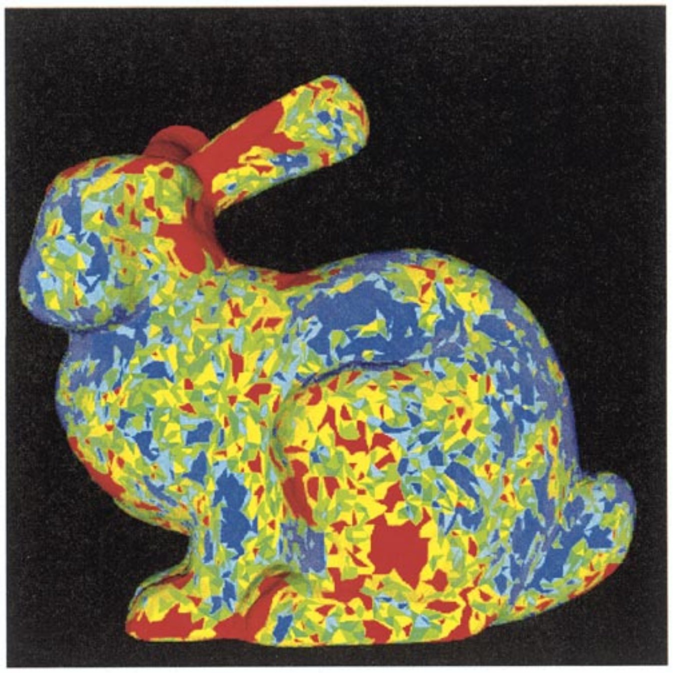
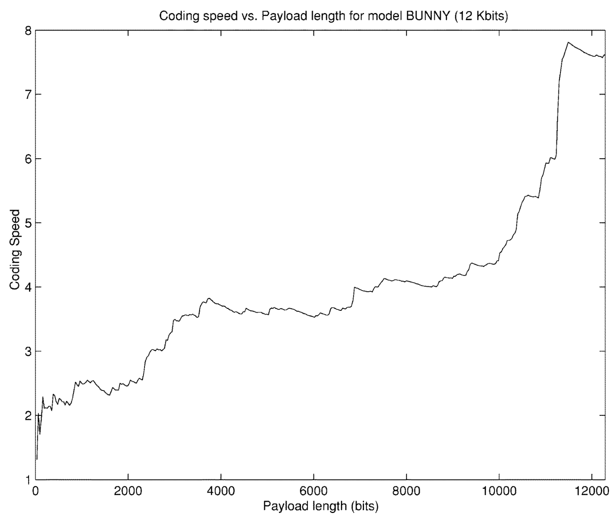


Fig. 9. Stego mesh for model **bunny** (12 Kbits, 87.7% of total capacity).

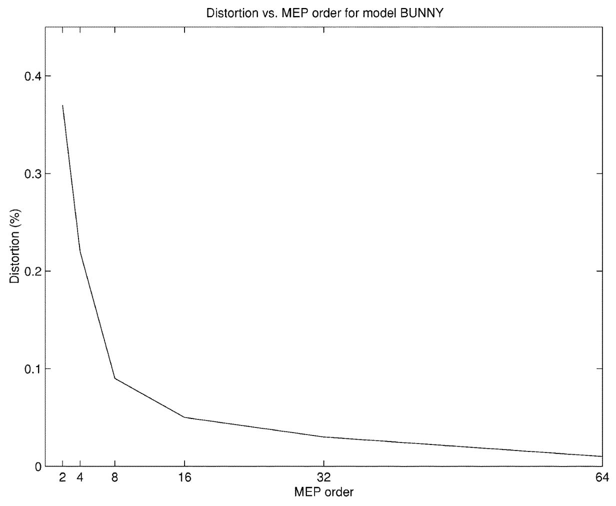
Finding the initial edge when the initial cell is known is quite easy. A local ordering of the edges inside a cell is straightforward: We take the largest edge of the initial cell for the very first entry edge of the geometrical stencil. We just described several ways of finding the initial cell, which is of greatest interest, but in the original TSPS implementation, the final cell was found topologically (when the stencil stops). In our case, we have to know when to stop. Two ways are possible, at least. One knows the number of bits to be retrieved, or one adds a *lead-out* sequence. For simplicity, we assume the size of the payload is known.

E. Security

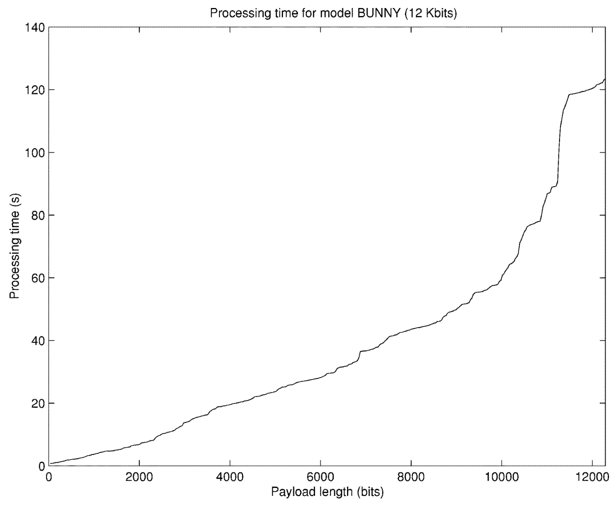
There are already results for validating the level of security offered by an algorithm. Unfortunately, we found them un-



(a)



(b)



(c)

Fig. 10. Figures for the embedding of 12 Kbits inside model **bunny**. (a) Coding speed (ratio between total stencil length and payload length) is very much related to processing time and complexity. (b) Distortion rate: exponentially decreasing with MEP order. (c) Processing time: exponentially growing with payload length.

tractable to evaluate security in practice due to the large amount of needed processing power. However, we can still estimate how secure our stego-system is. The two challenges an attacker has to address are finding the initial cell and getting the path over the mesh. Just by looking at these two issues, one can state that this scheme is secure in the cryptographic sense. It is resistant against exhaustive search. This means that it is much easier to remove the payload than to read it without the key. We believe that retrieving the message without the key is virtually impossible. This problem is NP-hard with respect to the number of cells in the mesh. Moreover, we embed the payload in an undeterministic way, which makes any steganalysis difficult [23], [24] or impossible.

VI. Algorithm and Accuracy Measures

In this section, we focus on the algorithm, and we deduce two measures to evaluate our implementation. They are related to the amount of distortion caused by the MEP (geometrical part of the algorithm) and to the coding speed (topological part of the algorithm).

A. Algorithm

We have mainly focused on the separate aspects of this steganographic scheme in order to sharpen the definitely strong possibilities of steganography on connected triangles. We present the pseudo-code algorithm in Fig. 7. As the pseudo-macros `currentCellIsAdmissible` and `getSteeringEdge` are not of much interest (they correspond to forbidden points management and topological exceptions handling), they will no longer be taken into account here. This way, one can observe that the way we use to move on the mesh leads to common parts for implementation. As a matter of fact, when we are on a triangle, we know it is admissible, and we only have to choose the access type to the information inside the cell: *embed*, or *retrieve*, or even possibly *erase*.

In the scope of this presentation, we do not present some of the refinements that may improve this basic implementation. For example, one could add *lead-in* and *lead-out* sequences to better control the retrieval step. We mainly focus on the algorithm machinery since optimization from previous work [10] can directly be transposed in our case.

B. Accuracy Measures

In the still-image case, one may generally want to evaluate the distortion of the algorithm on the original content. This is usually done with SNR computations in the still image case [21], and some proposals have been made for meshes mostly based on the Haussdorff distance‘ and some others on Laplacian. The measure we define for estimating the distortion induced by our algorithm is very simple: During embedding, we save the value of the maximal $\lambda_{\max}$, and we divide it by the length of the largest edge in the mesh $L_{\max}$. This is some sort of normalized Hausdorff distance, but it is not at all related to any psycho-visual or perceptual statement. We let R_{dist} be the distortion rate

$$R_{\text{dist}} = \frac{\lambda_{\max}}{L_{\max}}. \quad (4)$$

If we had taken only $\lambda_{\max}$ as a measure, it would have been in fact the Hausdorff distance. Since we want to compare distortion

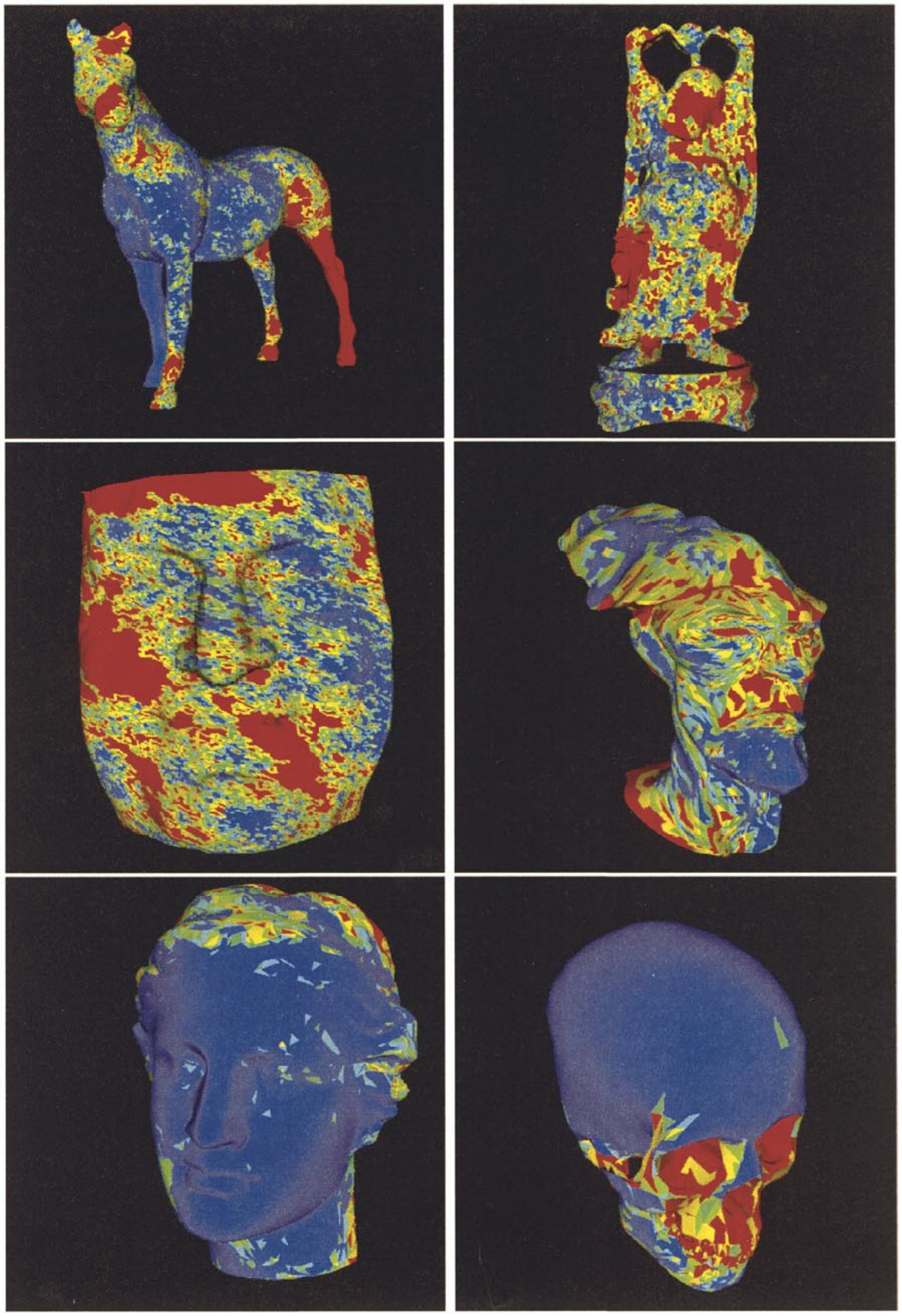


Fig. 11. Stego meshes for models (top left) **horse** (40 Kbits), (top right) **buddha** (25 Kbits), (middle left) **face** (20 Kbits), (middle right) **inopl** (8 Kbits), (bottom left) **venus** (7 Kbits), and (bottom right) **skull** (4 Kbits). Saturated regions are in blue and information-free regions in red.

results between several meshes, we choose to divide $\lambda_{\max}$ by the length of the largest edge in the mesh as a normalization. This measure only holds since the topology is not changed by the embedding process. However, the results will emphasize the effect of MEP order over distortion. Similarly, we define another measure for the topological part of the method: the so-called coding speed R_{cod}. We then let

$$R_{\text{cod}} = \frac{N_{\text{Stencil}}}{N_{\text{bits}}} \tag{5}$$

where N_{Stencil} is the number of triangles in the final stencil when considering the embedding of N_{bits} bits. The final stencil contains both triangles that convey the information and triangles that links the previous ones one another. This will lead to an accurate measure of coding speed, as there are in the stencil as many admissible triangles as there are N_{bits} to embed. The coding speed can also be seen as an empirical measure of the complexity of the algorithm, as it is very closely related to processing time. The search for admissible triangles is clearly a random process, and we will therefore have to keep an empirical measure of its behavior. During this random search for the next admissible triangle, the stencil may need to visit more triangles than the number of bits to embed. This is due to the fact that the number of forbidden points increases as the payload is embedded. This way, the coding speed can be greater than 1 as the filling rate gets close to 1.

VII. Results and Future Improvements

A. Practical Implementation

Here, we give some results that we obtained with some models (the bench set was constituted of 11 meshes); they are expressed with the two rates defined previously: the so-called *distortion rate* and the *coding speed*. Coding speed, distortion rate, and processing time are detailed in Fig. 10 for the case of the bunny model. Visual results of the watermarking are shown in examples of Fig. 11. The former is related to the geometrical distortion we introduced into the mesh, and the later deals with the coding efficiency of the algorithm. By looking at Fig. 8, one can see that the achieved filling rate is quite good in practice: Our way of moving over the mesh is sufficient. Furthermore, the order of the MEP used for embedding performs little distortion, but we still give full control over this parameter to the user. Despite these good results, the coding speed is quite high, which means that security is improved but, in addition, that processing time is far from optimum. This lack of control lies in the randomness of our moving strategy.

We used a PC-based 800-MHz SMP workstation running the Visualization ToolKit (VTK) under Linux to perform the tests on the models. Synchronization (finding the initial cell) may last 25 s in the case of a PCA synchronization for the *bunny* model. Again, we are only interested in evaluating the raw scheme. The code was written in C++, and the processing time is the same for embedding and decoding: It is comparable with other schemes for 3-D meshes.

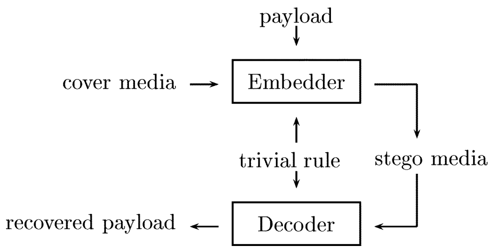


Fig. 12. Open steganographic paradigm for no-key schemes. A trivial rule is used to generate the key both at embedding and decoding. We embed the payload into the cover media, and the decoder only needs the rule for moving on the mesh to extract the recovered payload from the stego media.

B. Improvements

We present here several improvements one could add in order to better custom design his steganographic scheme using this method. The two main drawbacks of this method are its lack of local geometrical robustness, as well as its spatial localization. Moreover, when considering small triangles, machine precision errors can occur. The improvements presented here are various suggestions to help in the design of a customized scheme.

1) Multiple Stencils: We described the basic method for the generation of one stencil. We saw that the coding speed was not optimal, resulting in time-consuming embedding and decoding steps. To speed up the process, one could use multiple stencils. This is a way of improving the overall processing time. The forbidden point list must be the same for all stencils. One decides to start with another stencil when too many successive triangles are not admissible. For our current implementation, we let the algorithm generate up to four stencils. Considering a pure PCA synchronization, one could take advantage of the six channels it initiates.

Using multiple stencils allows a decrease in processing time, at least for the beginning of the process. At the end of the process, when reaching capacity limit, the problem remains the same, and the process is still time consuming. This improvement is of topological concern.

2) Increasing Capacity: Our scheme hides one bit per vertex. It is derived from the QIM concept. Still, it is possible to hide more than one sole bit per vertex. Using another decomposition of the entry edge, it enables the hiding of a certain amount of bits per vertex. For example, one could use a more elaborated version of the decomposition of every $B(k)$. Since we only hide one bit, we need to divide the $B(k)$ into two parts. Hiding k bits per vertex would require the division of the $B(k)$ into 2^k intervals. Since we can parameterize the order of the MEP with n, the definitive capacity limit is reached when machine-precision errors occur. As stated before, we did not implement such decompositions of the entry edge, but we plan to consider such issues for future work. It would result in generalized a QIM method for 3-D meshes. This improvement is of geometrical concern.

3) Key and the Open Steganographic Channel: Our method is designed to fit with the classical secret key paradigm. However, depending on the application, one could find it useful to make the information public. We then have to fit another paradigm: the open steganographic paradigm: see Fig. 12. Since the key represents the directions we want the stencil to follow successively, we have a way of making a simple open steganographic channel. If we take a simple rule (such as an alternate "0" and "1" pattern for the so-called key), we are able to make information public, as in other works [10].

Another choice could be a uniform string of "1" (resp. "0") for the key, thus letting the topological part of the algorithm find the admissible triangles to store the bits in (forbidden points management and admissibility criteria). This could translate into a simple rule: *always try to get out of the current triangle using the first (resp. second) exit edge*.

Moreover, when adding *lead-in* sequences [10], one could choose to make them select different areas of the mesh so that multiple public payloads are allowed from the same initial cell. Such added sequences would turn into a table of content (TOC) of the public information.

VIII. Conclusion

We developed a new spatial steganographic scheme for 3-D triangle meshes by dividing our approach into geometrical and topological considerations. The main difference with previous works [3], [10] is that these two issues are clearly separate from each other. This allows fine parameterization of the algorithm on both geometrical (order of the binary MEP) and topological aspects (use of rules). Security was also improved (no topological singularity is generated [3], and the retrieval process might be private and not only public [10]) through the use of a key to set up the stencil successive directions to follow.

This method is robust against translation, rotation, and scaling but offers poor robustness when performing local mesh manipulations like simplification or remeshing because it is triangle-realization dependent. To address the issue of finding the initial triangle, we proposed the use of a PCA to link the content of the mesh to the payload itself.

Acknowledgment

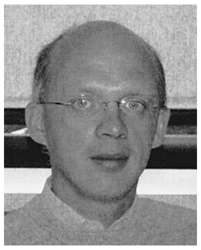
The models we used are taken from the Stanford Library and from various places over the Internet. The authors would like to acknowledge the suggestions of F. Schmitt and H. Maître from École Nationale Supérieure des Télécommunications, Paris, France.

References

[1] "Benchmark Requirements," European Union, EU IST-1999-10 987 CERTIMARK Project, 2000.
[2] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models through geometric and topological modifications," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 551–559, Apr. 1999.
[3] ——, "Watermarking three-dimensional models," in *Proc. ACM Multimedia*, 1997, pp. 261–272.
[4] ——, "A shape-preserving data embedding algorithm for NURBS curves and surfaces," in *Proc. Comput. Graph. Int.*, June 4–11, 1999.
[5] R. Ohbuchi, S. Takahashi, T. Miyasawa, and A. Mukaiyama, "Watermarking 3-D polygonal meshes in the mesh spectral domain," in *Proc. Graphics Interface*, Ottawa, ON, Canada, June 2001, pp. 9–17.
[6] M. G. Wagner, "Robust watermarking of polygonal meshes," in *Proc. Geometric Modeling Processing*, Hong Kong, Apr. 2001, pp. 201–208.
[7] B.-L. Yeo and M. M. Yeung, "Watermarking 3-D objects for verification," *IEEE Comput. Graph. Applicat.*, pp. 36–45, Jan./Feb. 1999.
[8] O. Benedens, "Watermarking of 3-D polygon based models with robustness against mesh simplification," *Proc. SPIE: Security Watermarking Multimedia Contents*, pp. 329–340, 1999.
[9] ——, "Geometry-based watermarking of 3-D models," *IEEE Comput. Graph., Special Issue on Image Security*, pp. 46–55, Jan./Feb. 1999.
[10] ——, "Two high capacity methods for embedding public watermarks into 3-D polygonal models," in *Proc. Multimedia Security Workshop ACM Multimedia*, 1999, pp. 95–99.
[11] O. Benedens and C. Busch, "Toward blind detection of robust watermarks in polygonal models," in *Proc. EUROGRAPHICS Comput. Graph. Forum*, vol. 19, 2000, pp. C199–C208.
[12] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Trans. Inform. Theory*, vol. 47, pp. 1423–1443, May 2001.
[13] M. Ramkumar and A. N. Akansu, "Self-nois suppression schemes for blind image steganography," in *Proc. SPIE Special Session Image Security*, vol. 3845, Boston, MA, Sept., pp. 55–65.
[14] Z. Karni and C. Gotsman, "Spectral compression of mesh geometry," in *Proc. SIGGRAPH*, 2000, pp. 279–286.
[15] H. Date, S. Kanai, and T. Kishinami, "Digital watermarking for 3-D polygonal model based on wavelet transform," in *Proc. ASME Des. Eng. Techn. Conf.*, Sept. 12–15, 1999.
[16] S. Kanai, H. Date, and T. Kishinami, "Digital watermarking for 3-D polygons using multiresolution wavelet decomposition," in *Proc. Sixth Int. Workshop Geometric Modeling: Fundamentals Applicat.*, Sept. 7–9, 1998.
[17] E. Praun, H. Hoppe, and A. Finkelstein, "Robust mesh watermarking," in *Proc. SIGGRAPH*, 1999, pp. 69–76.
[18] I. Guskov, W. Sweldens, and P. Schroeder, "Multiresolution signal processing for meshes," in *Proc. SIGGRAPH*, 1999, pp. 325–334.
[19] F. Petitcolas, R. Anderson, and M. Kuhn, "Attacks on copyright marking systems," in *Proc. Inform. Hiding 2nd Int. Workshop*, Apr. 15–17, 1998.
[20] P. Bas, "Méthodes de tatouage d'images fondées sur le contenu," Ph.D. dissertation (in French), Institut Nat. Polytechn. Grenoble, Lab. Images Signaux, Grfenoble, France, 2000.
[21] J.-F. Delaigle, "Protection of intellectual property by perceptual watermarking," Ph.D. dissertation, Commun. Remote Sensing Lab., Univ. Catholique Louvain, Louvain-la-Neuve, Belgium, 2000.
[22] S. Baudry, J.-F. Delaigle, B. Sankur, B. Macq, and H. Maître, "Analyzes of error correction strategies for typical communication channels in watermarking," in *Proc. Special Section Signal Process. Inform. Theoretic Aspects Digital Watermarking, Signal Process.*, June 2001, pp. 1239–1250.
[23] C. Cachin, "An information-theoretic model for steganography," in *Proc. 2nd Inform. Hiding Workshop*, Portland, OR, April 1998, pp. 306–318.
[24] J. Z. Zollner *et al.*, "Modeling the security of steganographic systems," in *Proc. 2nd Inform. Hiding Workshop*, Portland, OR, Apr. 1998, pp. 345–355.
[25] D. V. Vranic, D. Saupe, and J. Richter, "Tools for 3-D-object retrieval: Karhunen-Loeve transform and sperical harmonics," in *Proc. IEEE Workshop Multimedia Signal Process.*, Cannes, France, Oct. 2001, pp. 293–298.

François Cayre was born in 1977. He received the Informatics Engineering and the DEA (in systems control) degrees from the Universitée de Technologie de Compiégne, Compiégne, France, in 2000. He is currently pursuing the Ph.D. degree under the supervision of Prof. B. Macq of the Communications and Remote Sensing Laboratory, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, and Prof. H. Maître of the Département Traitement du Signal et des Images, ENST Paris, France.

He has been involved in the European Projects CERTIMARK (IST) and BRIC (ITEA). He is also involved in French research networks on watermarking and digital geometry communications.



Benoît Macq (SM'01) was born in 1961. He received the Electrical Engineering and Ph.D. degrees from the Université Catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 1984 and 1989, respectively. He did his Ph.D. dissertation on perceptual coding for digital television under the supervision of Prof. P. Delogne at UCL.

He has been a Professor with the Telecommunications Laboratory of the same university since 1996. From 1992 to 1996, he was a senior researcher with the Belgian NSF at UCL and invited assistant professor with the Telecommunications Laboratory. In 1990 and 1991, he worked on network planning for Tractebel S.A., Brussels, Belgium. He has been visiting scientist at Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 1995 and at the Massachussets Institute of Technology, Cambridge, during the summer of 1999. He has been Visiting Professor at the Ecole Nationale Supérieure des Télécommunications, Paris, France, and at the Université de Nice, Sophia-Antipolis, France, from 1999 until 2000. His main research interests are image compression, image watermarking, and image analysis for medical and immersive communications. He was a Guest Editor for the journal *Signal Processing*.

Dr. Macq served as a Guest Editor for the PROCEEDINGS OF THE IEEE. He is member of the program committee of several IEEE and SPIE conferences.