Texture-Based Watermarking of 3D Video Objects

Emmanuel Garcia, Jean-Luc Dugelay, *Senior Member, IEEE*

***Abstract*— In this article, we describe a novel framework for watermarking 3D video objects via their texture information. Unlike classical algorithms dealing with 3D objects that operate on meshes in order to protect the object itself, the main goal of our work is to retrieve information originally hidden in the texture image of the object, from *resulting images or videos* having used the 3D synthetic object, thus protecting the visual representations of the object.**

After developing the theory and practical details of this 3D object watermarking scheme, we present the results of several experiments carried out in various conditions, ranging from ideal conditions (e.g. known rendering parameters) to more realistic conditions (e.g. unknown rendering parameters, but estimated from 2D view) or within the context of attacks (e.g. mesh reduction).

***Index Terms*— Watermarking, 3D objects, texture.**

I. Introduction

Synthetic images of virtual 3D worlds are easier to produce and look more realistic than ever. Many tools are already available to a large public allowing them to design such images and even animated sequences. Many computer applications, especially games and simulators, use 3D modeling and rendering to represent virtual worlds or to represent abstract data in an efficient and meaningful way [1].

Other applications mix synthetic images with real images in order to produce what is called *virtualized reality* [2]. This is especially the case in the field of medical images and surgery simulators, or in some movies where cartoon characters and other computer generated objects are embedded in real videos. Another application of interest is that of virtual teleconferencing where image and videos of participants would be replaced by their synthetic clones, allowing for the creation of a virtualized environment common to all and yet potentially customized for each person [3].

Considering all the progress made in these fields, it will be more and more difficult to make the distinction between reality and fiction. We could imagine a scenario where the 3D model of a real person would be used and manipulated to make a speech or perform some fictitious action. Provided it is done in a realistic manner, it would be impossible to know whether the performance is a simulation or that of the real person, which could lead to serious abuses.

These concerns are taken seriously by many people and labs who work out solutions to be able to protect multimedia documents in general and 3D objects in particular, and to be able to identify or authenticate them. The keyword *watermaking* is used to designate the emerging techniques that allow users to hide information in computer data much like an invisible signature, be it in images, multimedia documents, or more abstract data.

This work was partly supported by the EU Certimark Project and is now partly supported by the FR RNRT SEMANTIC-3D project

The purpose of this article is to introduce a new framework for watermarking a 3D video object based on the object's texture map instead of its geometrical data. For this reason it has the potential of protecting all the images derivated (i.e. synthesized) from a 3D object, after it was watermarked.

After presenting this framework in section II, we will discuss the links with the underlying still image watermarking problem in section III, then provide algorithmic details in IV, and finally present preliminary experiments in V. First, we must recall a few prerequisite notions of watermarking and 3D objects.

A. Basic review on watermarking

Using the special case of images, which was historically the first type of digital documents investigated for watermarking, we will recall some basic and useful watermarking notions.

First, image watermarking is a term designating a technique that aims at *embedding* and *hiding* secret information in an image according to an optional secret key. It is then possible to check whether secret information has been embedded in the image (i.e. whether the image is marked), or whether some given information (i.e. message) is actually embedded in the image, or to determine the actual information (i.e. mark) that was embedded in the image. To these correspond three extraction modes:

- *Blind*: the watermark can be extracted from the watermarked image and no other knowledge is needed.
- *Semi-blind*: one can check whether the watermark is embedded or not in the image provided one knows what mark one is looking for.
- *Non-blind*: the watermark can be extracted provided one knows not only the image that is being checked, but also the original image from which it is thought to be derived.

Then, there are two main classes of watermaking applications:

- *Integrity checking*: the aim is to be able to detect that a document has been tampered by inserting a *fragile* watermark that would disappear if the document is touched, and possibly tell what parts of the document are altered.
- *Authentication*: the aim is to be able to say that a given document originates from a certain source even after it has been manipulated and tampered, either in a voluntary attempt to remove the mark or in the course of usual operations (e.g. image compression).

Finally, a fundamental notion is that watermarking is a tradeoff between *capacity*, *visibility* and *robustness*. The capacity of a watermarking algorithm is the amount of information that can be hidden in the watermarked image. The term

visibility refers to image distortion and visual degradation due to the embedding of the watermark, and it is expected that a watermark be as invisible as possible. Robustness means the ability to recover the watermark even after the image has been manipulated and altered in a non-destructive manner (i.e. in a manner that preserves its semantic and authentic content, this notion being partly subjective). For a given algorithm, it is impossible to improve one criterion without degrading one of the other two. For a more extensive exposition of image watermarking, we refer the reader to [4].

B. Watermarking of 3D objects

1) Nature of 3D objects: Just as 3D objects may represent many different things and concepts and be used for many different purposes, there are many ways to represent them in a computer system. In its most common form, a 3D object is seen as a closed surface in a 3D space, possibly with a notion of inside and outside, and possibly associated with data that describes its appearance or properties, such as color, transparency, etc. These properties may vary over the surface and thus be stored in a map that is associated to the surface. There might also be properties that describe the inside of the object as a volume. However, in the context of multimedia applications, what usually matters is to give a visual representation of the object and for this what is needed is to know the aspect of its surface, which is often described as a mere color map (i.e. image) also called texture map or texture image. For realistic rendering, a mere color map is usually not enough and the texture map should also provide information about the reflectance properties of the surface (i.e. how it reflects or diffuses incident light) [5].

As for the geometry of the surface itself, it can be described in various ways. A common and flexible way is to describe it as a polygonal mesh (i.e. a set of 3D points linked by edges that form planar polygons). But there exist other ways, such as using an implicit formulation (i.e. an equation, like that of a sphere) or CSG (Constructive Solid Geometry), resulting in a tree of boolean operations of primitive objects, or octrees, or stitched NURBS (Non-Uniform Rational B-Spline) patches, etc.

Regardless of the spatial nature and description of 3D objects, they may also be subject to variations along time if we intend, for example, to create a video representing a moving or deforming 3D object, such as a virtual talking head.

2) 3D watermarking algorithms: The purpose of current 3D watermarking algorithms is first to be able to tell, when given a purely 3D geometrical description, whether the object it represents is protected or not by a watermark (using one of the three possible extraction modes, depending on the algorithm).

In this overview of 3D watermarking algorithms we will restrict ourselves to the case of 3D objects represented by a triangular mesh, i.e. a set of 3D points (vertices) linked by edges that form triangles composing the surface of an object.

In this context, the requirement of a 3D watermarking algorithm is to be able to hide data in such a mesh description without noticeably modifying the surface described by the mesh and to be able to retrieve the hidden data afterwards. This usually involves slightly modifying the coordinates of the vertices, or modifying the connectivity and number of triangles in more or less sophisticated ways.

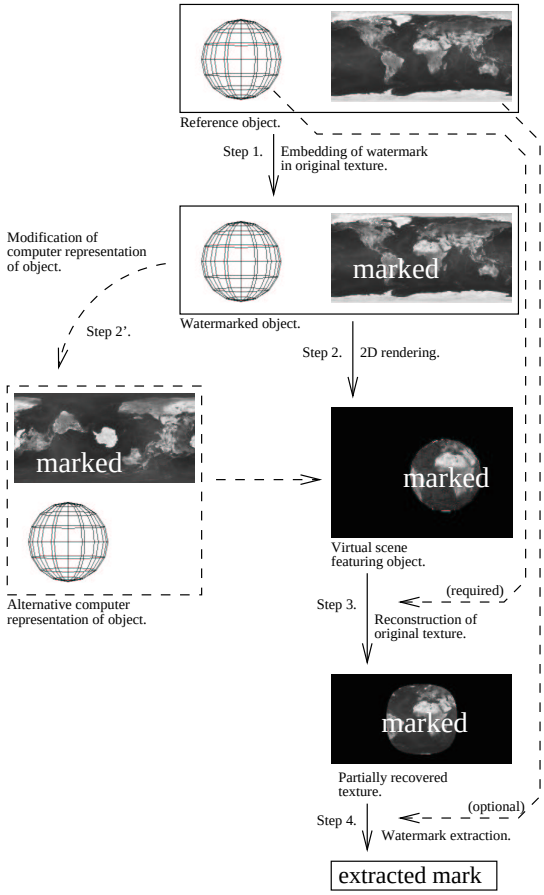


Fig. 1. General principle of texture-based 3D watermarking.

This point is shared by all algorithms, but they differ in their robustness to various manipulations of marked 3D objects. Some algorithms have been made specifically robust to translation, rotation or scaling of the whole object [7], [8], adding noise to the 3D coordinates of the vertices (most algorithms), cropping of the object [9] or remeshing [10] (e.g. mesh simplification aimed at reducing the number of triangles to speed up manipulations and rendering).

As later described, the texture-based approach is inherently robust to all these attacks. However, it only has a meaning for textured objects and is irrelevant to some CAD (Computer Assisted Design) objects only described as purely geometrical shapes.

II. Proposed framework

A. General principle

Before discussing in detail the implementation of our texture-based 3D watermarking algorithm and the results of preliminary experiments, we will now present its general framework.

As shown in Fig. 1, given a known 3D object consisting of a geometric definition, a texture image and a texture mapping

function (not represented but implicit), we embed information in the object by watermarking its texture (step 1) using a robust watermarking algorithm designed for still images. This watermarked object can then be published for further use and representations in virtual scenes (step 2). Afterwards, we can check that the represented object is protected by reconstructing the watermarked texture (step 3) and by finally extracting the embedded watermark (step 4) from the recovered texture image.

Let us point out that this process does not depend on any modification of the computer representation of the released 3D object as long as the appearance of the rendered 3D object remains the same (step 2'). Such a modification of either the geometry or texture information may be performed to try to erase the mark or to simplify the object or for any other purpose. This aspect will be discussed later in Section IV-B.1.

As can also be seen, one view of a 3D object generally provides only a partial knowledge of the whole texture image. So it is better, when possible, to recover partial texture images from several 2D views of the 3D object (e.g. face and profiles of the model of a human face) and then to merge them into a more complete texture image. This might be relevant in the case of animated virtual scenes.

As indicated by the dashed lines, the knowledge of the original 3D object is needed for watermark extraction. The geometry and texture mapping function are needed for the texture recovery step, and the knowledge of the original texture may be needed in case we use a non-blind underlying still image watermarking algorithm. So, our algorithm is basically non-blind since it requires at least the knowledge of the original geometry and texture mapping function in order to represent the visible texture information in its original "frame of reference" (i.e. where it was actually watermarked).

Finally we also need to know the rendering parameters. These mainly consists of the way the 3D object was projected onto the 2D view (i.e. its location in space with respect to the virtual camera and the intrinsic parameters of the virtual camera). The knowledge of the lighting model and parameters may also be required. This is needed in order to undo the transformations undergone by the texture from its original state to the final 2D view. The ability to accurately compute or estimate these parameters directly from the 2D view is the cornerstone of the presented framework.

III. Parallel with still images

In order to further describe the textured-based 3D watermarking algorithm we will comment on its features in comparison with the well-known case of still image watermarking.

A. Mode of extraction

By analogy with still image watermarking, we can say that our 3D watermarking algorithm uses a non-blind extraction mode, as it is necessary to know the original object. This knowledge is needed to reconstruct the texture image in the original frame of reference (i.e. where it was marked using a still image watermarking algorithm) and for the subsequent extraction of the mark by the still image watermarking algorithm.

More precisely, what must be known is the geometry of the object and the texture mapping function that describes how the original texture image is mapped onto the surface of the object, so that it can be reversed. The knowledge of the original texture image is not required by the texture reconstruction process, but it might be required by the underlying still image watermarking algorithm if it uses a non-blind extraction mode. It might also be needed to compute the rendering parameters (especially the perspective projection) with a projective registration scheme, as explained later in Section IV-C.

B. Visiblity of the mark

In the case of still image watermarking, the visual impact of the watermark on the marked image should be as limited as possible. The visibility is often expressed numerically as the signal-to-noise ratio between the marked image and the non-marked image. In the case of textured 3D objects it is slightly different because the images that are actually seen are not the marked texture image of the object, but 2D projections of it. While the visibility of the mark in the texture image still has a meaning, we are primarily concerned with the visibility of the mark in the 2D views of the textured 3D object. As for the signal-to-noise ratio between identical 2D views respectively using the marked and non-marked texture image of the object, we have observed that it is mostly the same as the signal-to-noise ratio between the marked and non-marked texture images. However, the underlying still image watermarking algorithm could possibly be better adapted to the special purpose of watermarking 3D objects in terms of visibility in the 2D protected views since some parts of the texture image could be mapped onto large areas of the object, while other areas could be mapped onto small areas of the object, and that some areas of similar size in the texture image may not, on average, be seen as areas of similar size in a 2D view of the 3D object. This could potentially be used to locally adapt the watermarking strategy in the texture image according to how the texture image is mapped on the geometry of the object.

C. Types of attack

We end our comparison with the still image case by indicating how the several operations to extract a watermark from a 2D view of a marked 3D object can be related to well-known attacks of still images.

Between the insertion of a watermark in the texture of a 3D object and its extraction, the texture, and therefore the watermark that is embedded in it, may have been altered during any one of the three stages: publishing the object, using the object, and extracting the mark (Fig. 2).

1) Publishing the object: After the object has been watermarked and published for use, one might change its file format, involving modifications of the texture mapping (Fig. 2 (i)) and/or of the geometrical description (Fig. 2 (ii) and (iii)) as said previously. An important feature of our algorithm is that it is insensitive to such modifications as long as the appearance

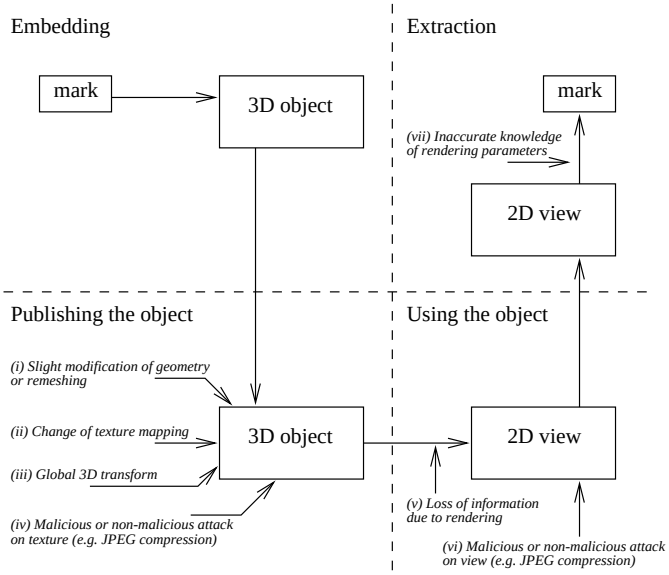


Fig. 2. Inventory of the possible impediments to watermarking recovery.

of the object does not change when visualized. However we mentioned that such modifications could introduce some noise (although probably invisible to a human eye) in the reconstructed texture, thus possibly lowering the performance of the subsequent watermark extraction.

2) Using the object: During the rendering of the object (Fig. 2 (v)) the texture information is reduced and altered. First, the whole surface of the object is generally not visible and thus the texture information is cropped. Then, the perspective projection induces an alteration similar to that induced by an image warping. In particular, when a part of the image is seen from a large distance, the corresponding texture information is visible only in its coarsest details and thus suffers an alteration similar to sub-sampling. And lastly, the texture information is modified, most likely in the low spatial frequencies, by the synthetic lighting conditions which may change the colors of the object. All these can be considered as the introduction of noise at various levels.

3) Texture reconstruction, watermark extraction: During this last stage the texture information is again affected by the warping (reverse projection and reverse texture mapping) of a discrete image but also by another kind of alteration which is much more severe. This alteration consists of local deformations of the recovered texture image with respect to the original texture image. It comes from the imprecision with which the perspective projection to be reversed is known (Fig. 2 (vii)). In the case of controlled experiments where this projection may be perfectly known, there would be no such local deformations. Only noise, cropping, and colorimetric alterations. However, we emphasize that in a realistic context, where the projection is not known a priori, the accuracy with which this projection is estimated is critical for the quality of the reconstructed texture image and for limiting local deformations to a minimum.

4) Comparison with still image attacks: We just mentioned all the alterations that could be induced in each stage of the 3D watermarking process. The alterations induced in the end-result reconstructed texture image on which the watermark extraction is actually performed can be compared to the following common attacks in still image watermarking.

- Cropping: in our context the end-result reconstructed texture image is generally a cropped version of the original texture image due to the fact that a single 2D view of a 3D object generally doesn't display the whole texture.
- Colorimetric alterations: the synthetic lighting used for 2D rendering is likely to modify the colour of the object's texture, more or less uniformly.
- Resampling noise: it is generally due to a resampling of the image to increase or decrease its dimensions, and in our case it is due to the fact that the 3D object may be viewed as a small picture (sub-sampling) and that this small picture will lead to a large reconstructed texture image (over-sampling), the whole process generating resampling noise and a loss of resolution due to sub-sampling.
- Local geometric deformations: they consist in limited displacement/warping of areas of the image. In our case they may occur due to a lack of precision in the perspective projection's estimate or due to a noticeable modification of the object's geometry, so that the reversing of the perspective projection and/or texture mapping would be more or less erroneous.

Finally, we point out that the overall rendering and texture reconstruction process consists of a combination of several above-mentioned attacks. The resulting attack may be seen as a destructive attack in the context of still images, as the original image may be barely recognizable. However, in the context of 3D objects, the process of 2D rendering must be considered as a non-malicious operation as long as the object can be recognized in the 2D view. This is a difference of interpretation of the term "destructive" which, in the context of 3D objects, is much more demanding of still image watermaking algorithms.

IV. Algorithmic details on 3D/2D part

A. Texture reconstruction

In this section we detail the algorithm for reconstructing the texture image from a 2D view, which is the core of our framework since the other elements (2D rendering and still image watermarking algorithms) are supposed to already exist and need not be documented here. As shown in Fig. 3,

- let S be the set of 3D points of the surface of the 3D object,
- let I be a 2D view of the 3D object,
- let T be the texture image to recover,
- let $\mathcal{F} : S \rightarrow T$ be the reference texture mapping function,
- let $\mathcal{P}$ be the projection from the system of coordinates of the reference 3D object onto the image of the virtual camera.

The main problem is to accurately know the projection matrix $\mathcal{P}$ of the virtual camera. Assuming it is known, we recover the part of texture T visible in the image I in the following steps:

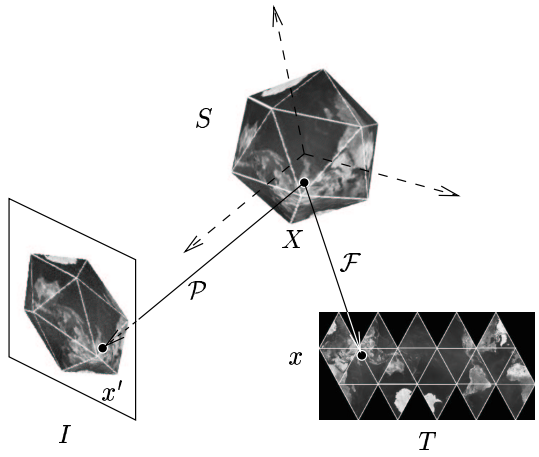


Fig. 3. Texture mapping and viewing of a 3D object.

1) consider each pixel x of T
2) compute $X = \mathcal{F}^{-1}(x)$ (if it exists)
3) compute the pixel $x' = \mathcal{P}(X)$ of the image I where X is projected
4) check that X is actually visible at pixel x' in image I
5) set the color of x to be that of x' in case X is visible in I at pixel x'

The fourth step is required because several points of the 3D object could project on the same pixel in the image I, but, if we assume the object is opaque, only one would be seen. To check whether a point X of the 3D object is visible in the image I, we first fill a Z-buffer associated with the image I.

Another problem that arises is that the computed x' generally has non-integer coordinates, whereas both the image I and its associated Z-buffer are only defined at pixels having integer coordinates. Thus, a strategy for comparing the depth of pixel x' with the values stored in the Z-buffer has to be found, and a choice for interpolating the image I at point x' has to be made.

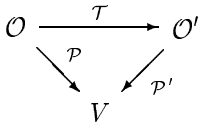
In both cases we considered the nearest neighbour (with integer coordinates) of pixel x'. For our experiments, we decided x' was visible when its depth did not differ from the depth of its nearest neighbour in the Z-buffer by more than a certain fraction, e.g. 1%.

As for interpolating the texture (i.e. colour) at pixel x' there is a wide range of formulas which are very well explained and assessed in [11]. The "nearest neighbour" algorithm is the most basic and the worst for most applications. However, in our case, what matters is that the amount of information present in the image I be used to the maximum extent for reconstructing T, and for making the extraction of the mark easier. It is not at all obvious that one interpolation scheme would retain more information than an other or would make the subsequent extraction of the watermark easier (we can especially remember that a visually good interpolation scheme always involves a low-pass filtering operation in one way or another, which can hardly be beneficial, even when not detrimental, to the preservation of embedded information such as a watermark). In fact we tested several interpolation schemes, including "nearest neighbour", "bilinear" and "cubic B-spline" and noticed no significant differences, if any, in the results of our experiments. We thus stuck to a "nearest neighbour" interpolation scheme.

B. Robustness properties

We pointed out that the algorithm was not affected, in principle, by a possible transformation of the computer representation of the object, as long as its visual representations did not change.

1) Resilience to projective transformation: In particular, when the geometry of the object undergoes a rotation, or even any affine transform, or more generally any projective transform, the set of possible visual representations via a perspective projection does not change, since such a projective transform can be accounted for in the perspective projection factor (i.e. $\mathcal{P}$). The following diagram



shows that if a watermarked object $\mathcal{O}$ undergoes a 3D projective transform $\mathcal{T}$ (which includes all types of affine transforms) before being rendered in a view V with a perspective projection $\mathcal{P}'$, what matters is to recover the perspective projection $\mathcal{P}$ according to which the original representation of the object $\mathcal{O}$ seems to be projected in the considered view V. In other words we aim to estimate the overall $\mathcal{P} = \mathcal{P}' \circ \mathcal{T}$, from the knowledge of only V and $\mathcal{O}$, and this problem remains exactly the same whether or not the geometrical representation of the object undergoes a projective transform before perspective projection.

2) Resilience to change of texture mapping: Another case of modification of object format is when the texture mapping is modified, as illustrated by the Earth globe example in Fig. 1, step 2'. Even if we use an intermediate computer representation of the object, the visual representation remains the same. We still use the original computer representation of the object (especially the original texture mapping function) to recover the original watermarked texture, or rather, to reconstruct the watermark texture in the original "frame of reference" where the still image watermarking algorithm was actually applied. Even though our algorithm is inherently resilient to a modification of the texture mapping of the watermarked object, the modification of the texture mapping implies the warping/resampling of the original texture image and thus introduces some resampling/interpolation noise.

3) Resilience to change of geometrical description: The last case of interest is when one changes the geometrical description of an object without significantly altering its shape. For example, an object described as a triangular mesh could be simplified by reducing the number of triangles. As long as the shape does not change (or only unnoticeably) it would make no difference for the watermark extraction since it is based on the knowledge of the original geometry. However if mesh simplifications are too drastic, the rendered shape would be quite different from the original watermarked shape. This would lead to local geometrical distortions in the reconstructed

texture with respect to the original texture because the projection of the original shape would not perfectly match the rendered view of the modified shape, i.e. the match X-x' in Fig. 3 would be biaised.

C. Projective registration

As underlined in the introduction, the main problem of our 3D watermarking framework is the necessity to know the 2D rendering parameters to reverse them for texture reconstruction. These rendering parameters are two-fold. First there are the lighting parameters modifying the original colours of the object. This has not yet been subject to experiment. So far we are confident that the underlying still image watermarking algorithm could easily cope with colorimetric alterations.

Then there are the projection parameters that consist of the 3D location of the 3D object with respect to the virtual camera (6 degrees of freedom) and in the intrinsic parameters of the virtual camera (5 degrees of freedom), though we do not make a distinction between these two kinds of parameters and designate them under the common term of projection parameters, or projection matrix, which is a 3×4 matrix defined up to a scale factor (thus having $3 \times 4 - 1 = 6 + 5$ degrees of freedom).

It is possible to carry out experiments with controlled rendering parameters (especially projection parameters) that would then be directly used for texture reconstruction. However, in a realistic context we would not know these parameters. They would have to be estimated as accurately as possible from the 2D view of the object and the knowledge of the 3D object, with no other a priori knowledge. Estimating the projection parameters involves what is called 3D/2D projective registration, that is, finding a 3×4 projection matrix that maps a set of 3D points onto a set of corresponding 2D points.

The problem of projective registration has been addressed in several other contexts, as in the case of medical imaging and robotics [12]. But then it is often a matter of registering pure geometrical features, e.g. points of maximum curvature or crest lines (which are hard to find in 2D images of textured objects, if possible at all), instead of textured patches of surface (which would be particularly adapted to our application).

A texture-based approach to registration is mentioned in [13] in the specific case of a human head model, where it is modeled as a cylinder. In the special case of human faces, results of face tracking independent of lighting conditions such as in [14] could maybe be reused and adapted. Still in the case of human faces, a gradient descent approach to estimating projection parameters (only the pose of the face actually) is proposed in [15]. A source of inspiration might also be found in [16] which deals with piecewise projective registration between successive images of a scene in a video sequence.

However, a dedicated algorithm to register a textured 3D object with a 2D view, possibly taking into account lighting conditions, has still to be specifically developed for our purpose. In Appendix II we present a simple projective registration algorithm that provides accurate results (mean distortion of about 1/20 pixel in the projected view with respect to the original projection) under the no-synthetic-lighting assumption.

V. Experimental results

We have carried out several experiments to provide a first assessment of the performances that can be expected from the presented 3D watermarking framework. The first of these experiments was conducted under minimal constraints and ideal conditions, while the others dealt with more realistic conditions.

In the following experiments we watermarked a 3D object using our 3D watermarking scheme and the underlying still image watermarking software called Eurémark [17] (see Appendix I for a short description) with a fixed visibility of 38dB. The visibility parameter being fixed, we assessed the tradeoff between capacity and robustness.

Obviously, other technologies designed for still image watermarking can be used within our framework of 3D video object watermarking based on texture. Nevertheless, the selected technology should include some relevant properties in terms of robustness, against photometric and possibly geometric attacks. As reported in [17], this is the case of the technology used in the following simulations. The watermarking algorithm is robust to any global affine transformation, local distortions [18] (e.g. Stirmark) and the main non destructive photometric attacks (e.g. typically, with a payload of 64 bits in a 512×512 image and a distortion of 38dB, the chosen technology is robust to a JPEG compression with a quality factor of as low as 25%).

A. Reference experiment

The conditions and parameters of this experiment, which we call the reference experiment, were completely controlled. First, there was no synthetic lighting (or, according to another point of view, there was only a unit ambient light) which means that the colours of the object, as described in its texture image, were projected directly in the 2D image plane with no alteration.

Then, the perspective projection that we used for 2D rendering was known during the texture reconstruction process. All this allowed for the best possible reversing of the 2D rendering process. Therefore it is hard to conceive that these results could be improved when assuming more realistic conditions (i.e. with more unknown parameters). They actually outline some of the limit performances to be expected from the presented 3D watermarking framework.

The only condition that may not have been ideal is the choice of the still image watermarking algorithm. Perhaps performances could be improved by using a still image watermarking better adapted to our specific purpose. For example, one can notice that some tuning (e.g. the proportion of oversampling vs. the duplication of the message to hide during the formatting step) designed within a context of still images may be reconsidered, customized and optimized with respect to the new context of 3D video object approach proposed in this article.

(a) Geometry. (b) Texture (512×512 pixels).

Fig. 4. 3D model of a real human head.

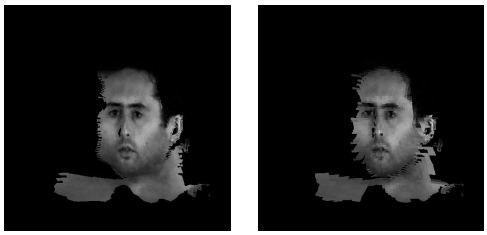
(a) (b)

Fig. 5. Two 2D views of the 3D face model under same point of view but with varying scale factor.

We will now present the results of this experiment applied to an object that was used throughout the other experiments, we will also test the performances of the still image watermarking algorithm in the absence of rendering and attacks, and finally perform an additional test on another object in order to check the consistency of these results.

1) First object: Fig. 4 shows the first 3D object used for the experiment, it is made up of a triangular mesh and a texture image and represents a realistic human head. We first watermarked the texture image with a 64-bit mark, and then viewed the 3D object under a fixed arbitrary point of view but with a varying scale factor (i.e. zoom). Fig. 5 shows two such views. Finally, we reconstructed the part of the original texture image that was viewable in those views. Fig. 6(a) and 6(b) show the texture images that could be reconstructed from the views shown in Fig. 5(a) and 5(b) respectively. It is to be noted that both views displayed the same part of the object, and thus, the same part of its texture. However, they displayed it with different resolutions, and the corresponding reconstructed texture images also had different resolutions: Fig. 6(b) displays less accurate details than Fig. 6(a).

Fig. 7 shows the number of erroneous bits in the recovered watermark from the view of Fig. 5 versus the scale factor (zoom). Actually, the x variable is not the scale factor of the



(a) (b)

Fig. 6. Texture images reconstructed from views of Fig. 5.

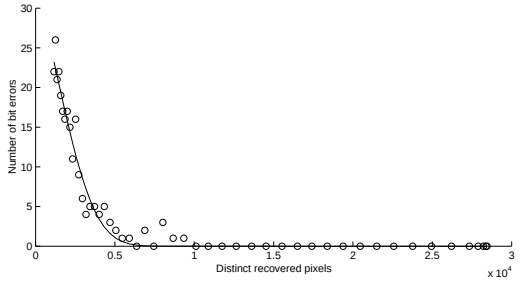


Fig. 7. Bit error rate versus number of recovered texture pixels from view of Fig. 5 with varying zoom factor and with a 64-bit mark.

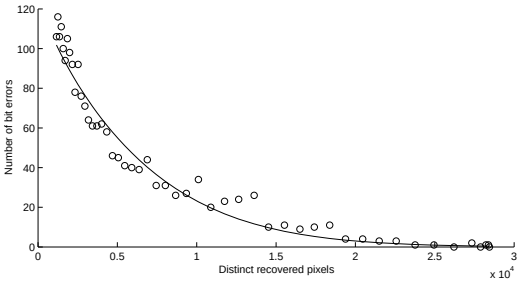


Fig. 8. Bit error rate versus number of recovered texture pixels from view of Fig. 5 with varying zoom factor and with a 256-bit mark.

2D view but rather the number of distinct texture pixels that could be recovered from the 2D view. The circles indicate experimental results while the continuous line represents a curve fitting of the form

$$\frac{y}{64} = \frac{1}{2} erfc(\alpha x^{\beta})$$

where $erfc : x \mapsto 1 - \int_0^x \frac{2}{\sqrt{\pi}} e^{-t^2} dt$ is the complementary error function. This choice is inspired by the formula $p = \frac{1}{2} erfc(SNR^{\frac{1}{2}})$ which is a well-known formula in the field of signal processing that relates the bit-error rate p to the signal-to-noise ratio (SNR) in the case of a basic noisy channel. In fact, it is logical that the bit error rate be close to $\frac{1}{2}$ when the available texture information is close to zero. It means that the bits of the so-called extracted watermark are determined completely at random when there is no texture information.

Fig. 8 shows the results of the same experiment, except that the size of the watermark was 256 bits. We used the same formula for curve-fitting, but then it was fitted to $\frac{y}{256}$. It can be noted that the bit-error rate converged more slowly to zero than with a 64-bit watermark.

Fig. 9 represents some performances of the algorithm in another way. Here we used the same view (that of Fig. 5(a)) but we varied the size of the watermark. As can be seen, a 200-bit mark can be hidden and recovered from the most magnified 2D view of the object that is considered in this experiment.

2) Bare performance of the still image watermarking algorithm: In order to see how the mere fact of projecting the object and undoing this projection affects the performance of the still image watermarking algorithm, we carried out an experiment that shows the performance of the watermarking algorithm without projecting and reconstructing the texture image. It consists in watermarking the texture image of Fig.

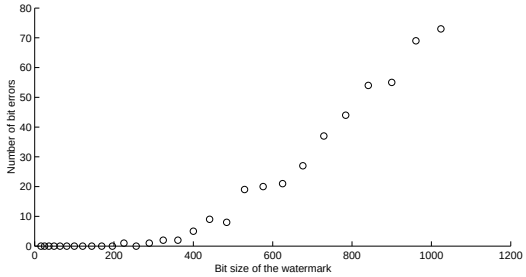


Fig. 9. Results obtained when using the view of Fig. 5(a), showing 28445 distinct texture pixels, and when varying the size of the watermark.

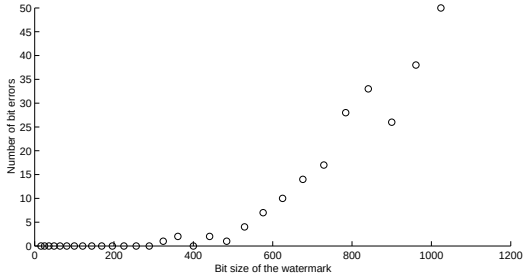


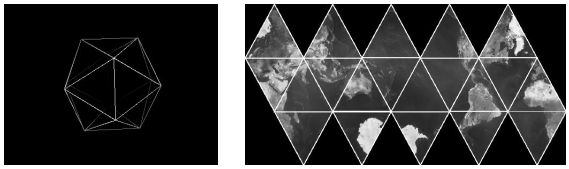
Fig. 10. Performance of the image watermarking algorithm after a cropping that keeps only the part of the texture visible in the 2D view.

4(b) and in performing the watermark extraction directly from this watermarked texture image, after retaining only the area that is visible in the view showed on Fig. 5(a), i.e. after applying a mask that has the shape of the reconstructed areas of texture in Fig. 6(a) to the watermarked texture. In this case, this represents a cropping to about 20% of the original area of the texture image.

Fig. 10 shows results that can be directly compared with those of Fig. 9 since the image used for watermark extraction had same shape and size (illustrated in Fig. 6). We can see that the projecting and "unprojecting" of the texture image slightly degrades the results of the still image watermark extraction process even when, as here, we projected the object with a large scale factor so that the texture was seen with a high resolution.

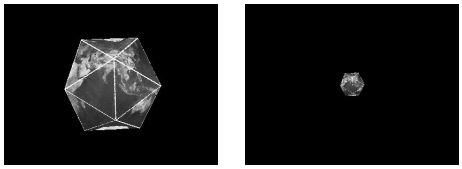
3) Second 3D object: Fig. 11 shows a second object used in this experiment. It is a polyhedral representation of the Earth. Then, Fig. 12 shows two 2D views of this object, under the same point of view but with different scaling factors (zoom), and Fig. 13 shows the two texture images that were reconstructed from these two views respectively. As in the case of the human head model, the two reconstructed textures have the same shape in both cases because the same part of the object was visible in both 2D views, even if the scaling factor was different. Here too, the image of Fig. 13(b) has a much lower resolution than the image of Fig. 13(a) because it was reconstructed from a 2D view where the object's texture is seen with a lower resolution (the object appears very small).

Fig. 14 and 15 show the results of recovering a 64-bit mark and a 256-bit mark respectively, from views identical to those of Fig. 12 but with a varying scaling factor. As with the first object (human head), we plotted the number of erroneous bits in the recovered watermark versus the number of distinct



(a) Geometry. (b) Texture (600×300 pixels).

Fig. 11. 3D model of a Earth-icosahedron.



(a) (b)

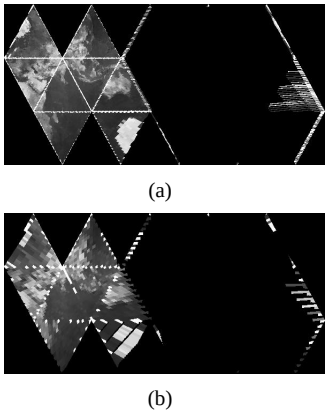
Fig. 12. Two 2D views of the Earth object under same point of view but with varying scale factor.

visible texture pixels, and we performed the same curve-fitting. Comparing those results with the case of the human head, we can see that they are not as good: 20000 distinct texture pixels are needed to recover a 64-bit mark whereas only 10000 were required in the the previous experiment (human head).

We did not investigate the causes of these differences very deeply, we only carried out the same experiment as in Section V-A.2 on this second object, to check the performances of the image watermarking algorithm on the texture image cropped to the shape shown in Fig. 13. Thus, we also noticed that the performances were lower than in Section V-A.2. This in turn could be explained by the fact that the performances of still image watermarking algorithms (including the one we used) are usually highly dependent on the content of the images. In our special case, the shape of the crop appearing in the reconstructed texture image may also play a major role.

B. Impact of imperfect projective registration

As already mentionned, the first experiment was carried out in most ideal conditions. The most significant assumption was that the perspective projection with which the 3D object was rendered in the 2D image was known. This assumption is



(a)

(b)

Fig. 13. Texture images reconstructed from views of Fig. 12.

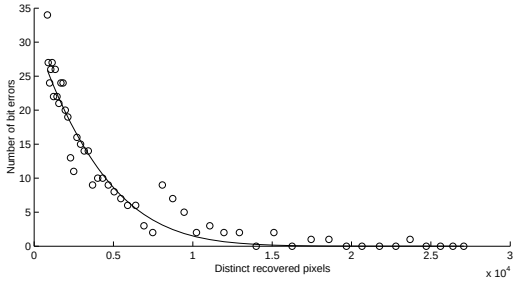


Fig. 14. Bit error rate versus number of recovered texture pixels from view of Fig. 12 with varying zoom factor and with a 64-bit mark.

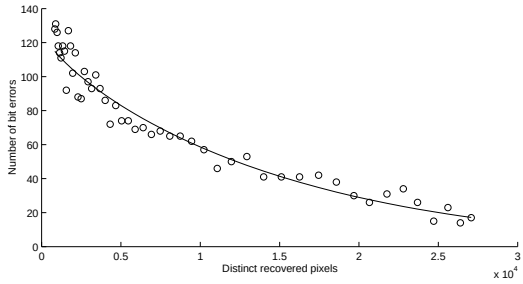


Fig. 15. Bit error rate versus number of recovered texture pixels from view of Fig. 12 with varying zoom factor and with a 256-bit mark.

what makes the previous experiment a bit unrealistic, even though it has the merits of showing the performances towards which we could tend if we were to develop an efficient 3D/2D projective registration algorithm, i.e. an algorithm that allows to accurately estimate the perspective projection between a 3D object and a perspective 2D view of it.

The other rendering parameters are mainly the lighting conditions but even if they are unknown and not cancelled when reconstructing an object's texture from a 2D view, it would have little consequence on the watermark recovery because any competitive still image watermarking algorithm is usually resilient to color alterations.

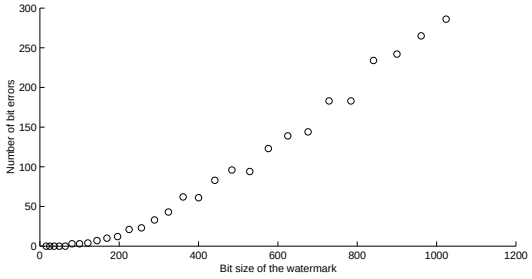
In order to assess the impact of an imperfect estimation of the rendering perspective projection on the texture reconstruction and watermark recovery process, we developped a simple projective registration algorithm (cf. Appendix II). This algorithm, which is currently designed to work in the absence of lighting alterations, could almost recover exactly the exact perspective projection used for rendering (the mean displacement of viewed pixels being around 1/20 pixel). Using this accurately recovered perspective projection for texture reconstruction, the results were the same as in the reference experiment without any noticeable degradation.

However, this projective registration algorithm cannot handle synthetic lighting (i.e. color alteration of the texture) and we do not currently know what accuracy is achievable in terms of projective registration in the presence of synthetic lighting. The following experiment hints at how performances are lowered when the perspective projection is not accurately estimated.

For this experiment we thus used the same protocol as for the previous one, except that the perspective projection used

(a) 2D view showing 28445 distinct pixels of texture.

(b) Reconstructed texture image.



(c) Bit error rate versus size of watermark.

Fig. 16. Watermarking experiment using a fixed view of the human head model, a varying watermark size, and an inaccurate perspective projection for texture reconstruction.

for texture reconstruction was the one used for rendering but shifted one pixel to the right in the 2D view.

Fig. 16 shows the 2D view, the corresponding reconstructed texture image, and the results of extracting a varying size watermark when using the above-mentioned inaccurate "estimate" of the perspective projection for texture reconstruction (shifted one pixel to the right).

This is to be compared to the results of Fig. 9. We can see that in this case, an inaccurate projection estimation leading to a mean translation of one pixel in the perspective view decreases the capacity of the overall 3D watermarking algorithm by a factor between 3 and 4 (instead of retrieving 200 bits without error we can only retrieve around 60 bits). Again these results emphasize the need for an *accurate* projective registration algorithm.

C. Impact of mesh simplification

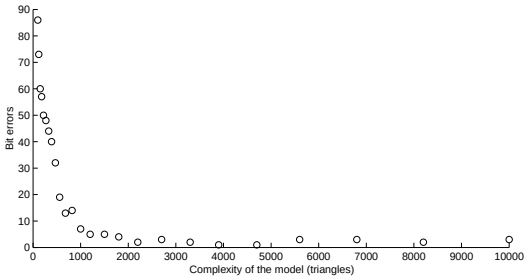
The purpose of this experiment is to support the claim that our watermarking scheme is resilient to modifications of the computer representation of a 3D object, and, in this example, to mesh reduction.

We used the same protocol as in the reference experiment with the human head model. The object's original mesh consisted of 10000 triangles. This is the mesh that was used to reconstruct the texture. But before creating the 2D view of the object, we have simplified its mesh to various numbers of triangles and down to as low as 100 triangles [1]. This corresponds to the scenario where the owner of the object knows its 10000-triangle representation and the user of the object might simplify the mesh before rendering, but in the end the owner does not know what mesh simplifications have been

[1]We used Michael Garland's QSlim mesh simplification software [19] to perform mesh simplifications.

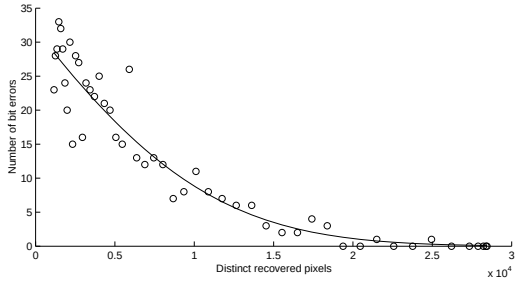
(a) 2D view of the human head model simplified down to 100 triangles and showing 27975 distinct pixels of texture.

(b) Texture image reconstructed from the 100-triangle model view, using the 10000-triangle 3D model.

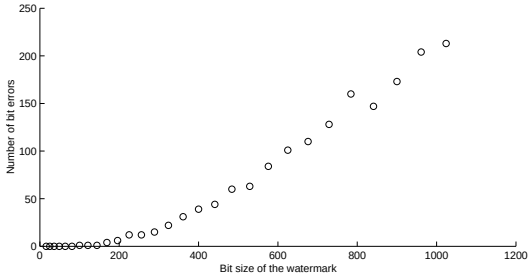


(c) Bit error rate versus mesh complexity for a 256-bit mark.

Fig. 17. A view of the simplified head model and results of watermark recovery for various mesh reduction rates.



(a) Same experiment as in Fig. 7 except the 2D view is subject to a 75%-quality JPEG encoding.



(b) Same experiment as in Fig. 9 except the 2D view is subject to a 75%-quality JPEG encoding.

Fig. 18. Impact of a JPEG compression applied to the 2D view used for watermark recovery.

performed on the object and can only use its original 10000-triangle description of the object to extract the watermark from the 2D view of the simplified object. However, as said earlier, if the 2D view of the object is the same, or mostly the same, whether the mesh is simplified or not, this should make no, or little, difference in the watermark extraction process.

Fig. 17 shows a 2D view of a simplified version of the human head, along with the corresponding reconstructed texture image (that was reconstructed using the knowledge of the 10000-triangle model) and the watermarking extraction results for a varying mesh complexity and a 256-bit mark. We can see that between 10000 and 1000 triangles, the number of errors in the recovered watermark fluctuates around 0 and that it increases dramatically as the mesh complexity decreases towards 100. In fact, we can see that the example reconstructed texture image shown is a slightly distorted version of the original texture image.

Let us note here that since other 3D watermarking algorithms usually modify the geometry in a visually imperceptible way, it is possible to use these watermarking algorithms after having applied our watermarking algorithm, since the former will not have any effect on the latter. The other way around is also true since whether or not we watermark the texture image, it does not make any difference for geometry-based 3D watermarking algorithms. Thus, the object could be simultaneously protected in various ways, both its computer representation as a geometry and texture, and its possible representations as 2D images.

Finally, let us point out that when the computer representation is modified to such an extent that it appears to be distorted, the watermark may not be recoverable (as in the case of a mesh simplification to 100 triangles) but in this case the object does not look the same anyway, so we cannot really talk about an object that is the same as the original object. It is as if it has undergone a more or less destructive attack, in which case watermarking ceases to be relevant.

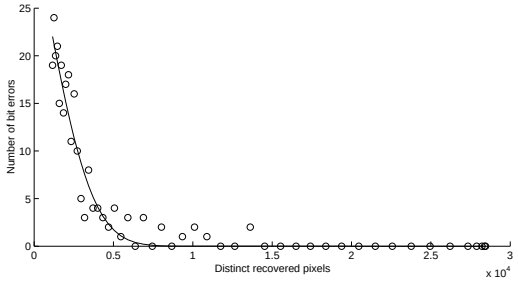
D. Impact of JPEG compression on 2D view

We performed an experiment to underline another kind of problem that may emerge when applying our algorithm in a realistic context. In fact, it is often the case that 2D images are compressed in order to save space or to reduce transmission times when they are to be viewed over a network. This is much more so when they are part of a video sequence (e.g. encoded following the MPEG-2 standard). Such compression not only degrades the visual quality of the image but is also likely to make a watermark extraction more difficult.

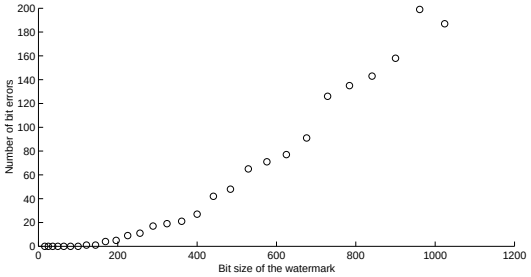
Fig. 18(a) and 18(b) are to be compared to Fig. 7 and 9 respectively. The latter experiments differ from the former only in the fact that the 2D view from which the watermark was extracted was subject to a JPEG compression with a quality factor of 75% (default normal usage).

We can see that the 75%-quality JPEG compression significantly decreases the robustness/capacity of the overall 3D watermarking scheme, whereas the underlying still image watermarking algorithm that we used is normally robust to this kind of compression. However, the still image watermarking algorithm operates in the “frame of reference of the texture image” whereas the JPEG compression that we consider operates in the “frame of reference of the 2D view”.

As in the previous experiment (mesh reduction) we can argue that if an image is heavily altered, as would be the case when a small view of a 3D object is subject to JPEG compression, it is no longer relevant to protect that view in the



(a) Same experiment as in Fig. 7 except the watermarked texture image is subject to a 75%-quality JPEG encoding.



(b) Same experiment as in Fig. 9 except the watermarked texture image is subject to a 75%-quality JPEG encoding.

Fig. 19. Impact of a JPEG compression applied to the watermarked texture image of the 3D object.

first place, since it can then be barely recognized, or mistaken, as a faithful representation of the original object. However this argument does not really hold for the results of Fig. 18(b) which were obtained using a large image of the human head model, and for which the effects of the compression should not be very visible. So, we must consider that these results better outline the limits of our scheme, at least if we are concerned with JPEG compression of 2D views.

E. Impact of JPEG compression on texture image

This last experiment is aimed at showing how common operations performed on the texture image can alter the whole watermarking process. In this case we assessed the impact of a JPEG compression of quality 75% applied to the texture image of the 3D object.

The results are shown in Fig. 19(a) and 19(b) and are to be compared with Fig. 7 and Fig. 9 respectively (no JPEG compression).

In order to compare this with the robustness of the still image watermarking algorithm to JPEG outside the context of 3D objects, we carried out the same experiment as in Section V-A.2 (Fig. 10) except that after watermarking and before cropping, we applied a 75%-quality JPEG compression.

The results of Fig. 10 are only slightly worse than those of Fig. 20, which means that what prevails here is the JPEG compression and not whether the still image watermarking algorithm is applied to a reconstructed texture or not.

We also see, by comparing Fig. 10 with Fig. 7, that the JPEG compression on the original texture image had little impact on the performances of our framework in the case of a “small” 64-bit mark. The JPEG compression had much more impact when applied to the 2D view, as reported in the previous experiment.

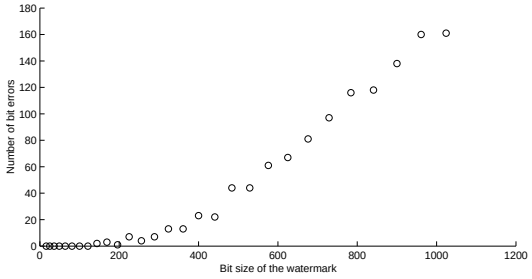


Fig. 20. Performance of the image watermarking algorithm after 75%-quality JPEG compression and a cropping keeping only the part of the texture visible in the 2D view. To be compared with Fig. 10.

F. Summary of all experiments

Table I gives a summary of the described experiments. There is a reference experiment, and others which are variations of this reference experiment to assess the impact of one or the other of the negative factors mentioned on Fig. 2.

What emerges from the results that can be compared (Fig. 9, 16(c), 18(b), 19(b)) is that the negative impact of a one-pixel inaccuracy of the projective registration is more severe than, but still comparable to, the impact of common operations such as 75%-quality JPEG encoding applied to either the texture image or the 2D view. So, while it is worthwhile finding efficient registration algorithms adapted to our purpose, it is possible that in the end the performance of the overall 3D watermarking scheme will be limited by common operations such as JPEG encoding.

VI. Concluding remarks

In this article we proposed an alternative framework for watermarking 3D objects. Whereas all currently known watermarking algorithms dealing with 3D objects seek to hide data in the 3D geometrical description of a 3D object, we pointed towards a dual direction, namely that data can be hidden in the texture map of a textured 3D object. The concerns are also different in that the former type of algorithms aims at protecting the computer description of a 3D object regardless of its visual representations, whereas our research was driven by the will to protect the subsequent visual representations of a textured 3D object in images or videos after it has been marked.

We drew attention on the fact that the presented scheme was inherently robust to modifications of the computer description of a 3D object as long as its appearance does not change significantly, and this was proven by experiments where a watermarked object was subject to mesh simplification before rendering. We also drew attention on the fact that currently the main limiting factor is the accuracy with which a watermarked 3D object can be registered with a 2D view. Good projective registration algorithms adapted for our purpose and able to cope with lighting conditions have still to be investigated and developed.

The presented technique, mainly intended to protect 2D representations of a textured 3D object, could also be used to protect the 3D object itself, much like geometry-based watermarking algorithms. This application has been left aside

TABLE I

SUMMARY OF EXPERIMENTS.

Section	Description of experiment	Figures
V-A.1	*Reference*. "Reference" experiment, watermarking a human head 3D model, projecting it in a 2D view with unit ambient light, reconstructing the texture image from the 2D view with completely known rendering parameters, and extracting the watermark therefrom.	7, 8, 9
V-A.2	*Bare performance.* Check the bare performance of the underlying still image watermarking algorithm with respect to capacity/robustness when the watermarked texture image is directly used for watermark extraction (i.e. bypassing mapping on 3D object, rendering on 2D view, and reconstructing from 2D view) except that it is subject to a crop similar to that of the previous experiment.	10
V-A.3	*Other object.* Same as first experiment but using a different 3D object: a polyhedral representation of the Earth.	14, 15
V-B	*Inaccurate projection.* Same as first experiment but the texture image is reconstructed from the 2D view using an inaccurate projection (obtained by shifting the actual one by one pixel to the right in the projected view).	16
V-C	*Mesh reduction*. Same as first experiment but simplifying the mesh of the object after watermarking and before rendering.	17
V-D	*JPEG view.* Same as first experiment but applying a 75%-quality JPEG compression to the 2D view before texture reconstruction and watermark extraction.	18
V-E	*JPEG texture.* Same as first experiment but applying a 75%-quality JPEG compression to the texture image after watermarking and before 2D rendering.	19
V-E	*Bare performance JPEG*. Check the performance of the still image watermarking algorithm with 75%-quality JPEG compression, and cropping to the area of the texture visible in the 2D view, but using directly the altered texture image without mapping, projecting and reconstructing it.	20

for the time being as we focused on protecting 2D views of a 3D object, but would be worth investigating and comparing with geometry-based algorithms as regards the robustness of the watermark to manipulations of the geometry; if the texture was left untouched, and only the texture mapping function changed to accomodate changes of geometry, then there would be no problem at all to extract the watermark from the texture image. Otherwise, if the texture image was warped in another frame of reference, it would all come down to registering, in 3D, the original watermarked object with a manipulated version of it, just like in the presented framework it was mainly a matter of registering a 2D view with the original 3D object. Such a technique would expectedly be at least as robust to modifications of the computer representation of the object as it is now when applied to 2D views.

Finally, let us point out that the features of both 3D watermarking approaches, texture- and geometry-based, could easily be combined in a unified watermarking scheme as either approach does not interfere, or little, with each other.

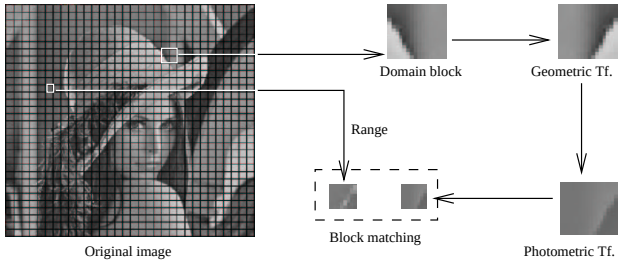


Fig. 21. Image fractal coding.

APPENDIX I
THE EURÉMARK ALGORITHM

The Eurémark still image watermarking software used in our simulations is a blind watermarking algorithm that can be summarized as follows, and which is documented in [20].

The considered approach is inspired from fractal image coding theory, in particular the notion of self-similarity, illustrated in Fig. 21. The main idea is to use some invariance properties of fractal coding, such as invariance by affine (geometric and photometric) transformations, to ensure watermark robustness.

A. Embedding

The watermark embedding process can be described in the following three steps: formatting and encryption of the message to hide, cover generation, merging of the watermark and the cover.

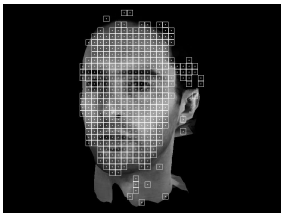
1) Formatting and encryption of the watermark: The message bits to be hidden are redundantly distributed: by over-sampling and duplication of the message to obtain a watermark of the size of the image. This redundancy is necessary for a good robustness. Finally, the watermark is globally encrypted using an XOR with a pseudo-random noise generated from a secret key, yielding the encrypted watermark W. The XOR operation allows, on one hand, to secure the hidden message, and on the other hand, to remove repetitive patterns reducing in this way the psycho-visual impact of the watermark embedding. In order to improve robustness against photometric attacks, error correcting codes can be added to the message prior to over-sampling and duplication of the message.

2) Cover generation: First, a "fractal approximation" I_{approx} is computed from the original image $I_{original}$. The cover I_{cover} corresponds to the error image, that is the signed difference between the original image and its fractal approximation.

$$I_{cover} = I_{original} - I_{approx}$$

3) Merging of the watermark and the cover: The last step of the embedding process consists in modulating the cover I_{cover} with W. The modulation consists in zeroing some of the cover pixels depending on their sign and the corresponding watermark bit to hide. For visibility reasons, only the low valued pixels in I_{cover} will receive a mark. Finally, the modulated cover $\hat{I}_{cover}$ is added to the fractal approximation I_{approx} to produce the watermarked image $I_{watermarked}$.

$$I_{watermarked} = I_{approx} + \hat{I}_{cover}$$



(a) Given 2D view.

(b) Given 3D object (only the texture is shown here but the geometry is also known and there is a one-to-one correspondance between texture and geometry via the also-known texture mapping function).

Fig. 22. Given data for the 3D/2D projective registration problem.

B. Extraction

The watermark extraction algorithm is similar to the embedding algorithm (i.e. dual operations) as is its complexity. First a fractal approximation is calculated from the watermarked image, which generates a cover close to the original one. Finally the cover is decoded according to the modulation rules (e.g. a positive pixel is supposed to carry a one-valued bit and a negative pixel a zero-valued bit). The crucial point is that most geometric transformations on the watermarked image are also transferred to the cover: the mark is not lost but the noise has to be correctly positioned with respect to the cover before applying XOR. Therefore, some additional bits called "resynchronisation bits" are added to the useful message bits in order to allow a self and blind resynchronisation of samples via two procedures: one for global geometric distortion (rotation and rescaling) based on FFT properties of periodic signals and Hough transform, one for local geometric distortion based on block-matching. The watermark can then be decrypted and the message rebuilt.

APPENDIX II
A SIMPLE 3D/2D PROJECTIVE REGISTRATION ALGORITHM

The fundamental element in the proposed 3D watermarking framework is the ability to recover the perspective projection parameters from a given perspective 2D view of a given textured 3D object with a very high accuracy. Although work related to this problem exists (as explained in Section IV-C), and although our problem may seem comparatively simple, we found no algorithm that actually tackles this specific problem in a simple manner. We thus propose an idea for such a 3D/2D registration algorithm.

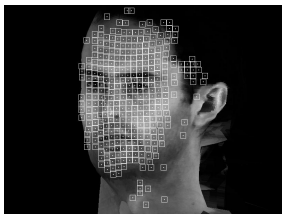
This algorithm consists in iterating a few times a projection refinement process starting from a rough initial estimate.

This process aims at finding point correspondances between 2D points of the object's given 2D view (Fig. 22(a)) and 3D points of the 3D object's surface (represented in Fig. 22(b) as the texture image) so that the projection matrix $\mathcal{P}$ can be computed from these pairs by solving a mere linear system.

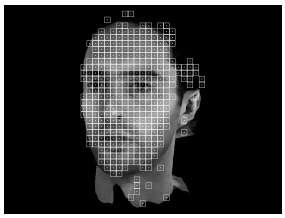
It seems rather difficult to find these pairs by directly matching points in the 2D view to points in the texture image using 2D non-rigid registration.

(a) Approximate view.

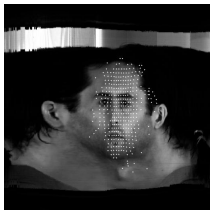
(b) Target view reconstructed from blocks of approximate view.



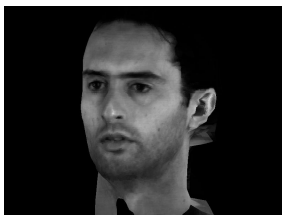
(c) Blocks found in approximate view.



(d) Given blocks in target view.



(e) Location of block centers on texture image.



(f) New approximate view.

Fig. 23. Finding 3D/2D pairs through block-matching.

Our approach rather consists in rendering the model using an approximate of the perspective projection (Fig. 23(a)) and in performing a simple block-matching between this approximate view and the given target 2D view (Fig. 23(c) and 23(d)) We thus find point correspondances between the target 2D view and the approximate 2D view.

This approximate 2D view is in a known one-to-one correspondance with the object's surface (Fig. 23(c) and 23(e)) because it was rendered using a known estimate of the projection matrix.

This yields by transitivity the needed point correspondances between the target 2D view and the 3D object's surface (Fig. 23(d) and 23(e)).

Fig. 23(b) shows the target 2D view reconstructed from blocks of the approximate 2D view and Fig. 23(f) shows the improved approximate 2D view using the projection matrix computed from the newly computed pairs.

In the given example, the initial estimate of the projection was computed from 6 manually selected pairs and after 4 iterations the mean distortion between the real and the estimated projections was 1/20 pixel in the projected image, which, as far as watermark extraction is concerned, is similar to no distortion at all. This algorithm, which presents some analogies with stereovision (especially in the matching of two different views of the same object) and with camera calibration, cannot yet cope with colorimetric differences between the two matched images, although this would be a requirement in the end.

REFERENCES

[1] P. Abel, D. Loisel, J.-P. Paris, and C. R. Dos Santos, "Automated construction of dynamic 3d metaphoric worlds for network management information visualization," in *Proc. of SPIE Electronic Imaging*, San Jose, California, Jan. 2000.

[2] I. Kitahara, H. Saito, S. Akimichi, T. Ono, Y. Ohta, and T. Kanade, "Large-scale virtualized reality," in *Proc. of CVPR*, 2001.

[3] J.-L. Dugelay, K. Fintzel, and S. Valente, "Synthetic natural hybrid video processings for virtual teleconferencing systems," in *Picture Coding Symposium*, Portland, Oregon, Apr. 1999.

[4] S. Katzenbeisser and F. A. P. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, 2000.

[5] S. R. Marschner, B. Guenter, and S. Raghupathy, "Modeling and rendering for realistic facial animation," in *Proc. of 11th Eurographics Workshop on Rendering*, Brno, Czech Republic, June 2000.

[6] C. Mallauran, "3d video object watermarking," Sept. 2001.

[7] O. Benedens, "Two high capacity methods for embedding public watermarks into 3d polygonal models," in *Proc. of ACM Multimedia and Security Workshop*, Orlando, Florida, 1999, pp. 95–99.

[8] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama, "Watermarking 3d polygonal meshes in the mesh spectral domain," in *Proc. of Graphics Interface*, 2001.

[9] E. Praun, H. Hoppe, and A. Finkelstein, "Robust mesh watermarking," in *ACM Siggraph 99 Conference Proceedings*, Los Angeles, California, Aug. 1999.

[10] O. Benedens, "Watermarking of 3d polygon based models with bobustness against mesh simplification," in *Proc. of SPIE Security and Watermarking of Multimedia Content*, 1999, pp. 329–340.

[11] P. Thévenaz, T. Blu, and M. Unser, "Image interpolation and resampling," in *Handbook of Medical Imaging, Processing and Analysis*, I. Bankman, Ed. San Diego CA, USA: Academic Press, 2000, ch. 25, pp. 393–420.

[12] J. Feldmar, N. Ayache, and F. Betting, "3D-2D projective registration of free-form curves and surfaces," *Journal of Computer Vision and Image Understanding*, vol. 65, no. 3, pp. 403–424, 1997.

[13] M. La Cascia, S. Sclaroff, and V. Athitsos, "Fast, reliable head tracking under varying illumination: An approach based on registration of texture-mapped 3d models," *IEEE trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 4, pp. 322–326, Apr. 2000.

[14] S. Valente and J.-L. Dugelay, "A visual analysis/synthesis feedback loop for accurate face tracking," Jan. 2001.

[15] F. Pighin, R. Szeliski, and D. Salesin, "Resynthesizing facial animation through 3d model-based tracking," in *International Conference on Computer Vision*, 1999.

[16] M. Gleicher, "Projective registration with difference decomposition," in *Proc. of CVPR*, June 1997.

[17] J.-L. Dugelay and C. Rey, "Image watermarking for owner and content authentication," in *ACM Multimedia*, Los Angeles, California, Nov. 2002.

[18] J.-L. Dugelay and F. A. P. Petitcolas, "Image watermarking: possible counterattacks against random geometric distortions," in *Proc. of SPIE Security and Watermarking of Multimedia Contents II*, San Jose, California, Jan. 2000.

[19] M. Garland, "Qslim mesh simplification software." [Online]. Available: http://graphics.cs.uiuc.edu/~garland/software/qslim.html

[20] J.-L. Dugelay and C. Rey, "Method of marking a multimedia document having improved robustness," French Pending Patent EUP 99 480 075.3 (EURECOM 14 EP), May, 2001.

Jean-Luc Dugelay (Ph.D. 92, IEEE M94-SM02) joined the Institut Eurécom (Sophia Antipolis) in 1992, where he is currently a Professor in charge of image and video research and teaching activities inside the Department of Multimedia Communications. His research interests are in the area of multimedia signal processing and communications; including security imaging (i.e., watermarking and biometrics), image/video coding, facial image analysis, virtual imaging, face cloning and talking heads. He contributed to the first book on watermarking (Information hiding techniques for steganography and Digital watermarking, Artech House 1999). He is an author or coauthor of more than 65 publications that have appeared as journal papers or proceeding articles, 3 book chapters, and 3 international patents. He gave several tutorials on digital watermarking (co-authored with F. Petitcolas from Microsoft Research, Cambridge) at major conferences (ACM Multimedia, October 2000, Los Angeles, and Second IEEE Pacific-Rim Conference on Multimedia, October 2001, Beijing). He has been an invited speaker and/or member of the program committee of several scientific conferences and workshops. He was technical co-chair and organizer of the fourth workshop on Multimedia Signal Processing, Cannes, October 2001. His group is involved in several national and European projects related to digital watermarking (RNRT Aquamars and Semantic-3D, IST Certimark). Jean-Luc Dugelay is currently an Associate Editor for the IEEE Transactions on Multimedia, the IEEE Transactions on Image Processing, the EURASIP Journal on Applied Signal Processing and the Kluwer Multimedia Tools and Applications. He is a member of the IEEE Signal Processing Society, Image and Multidimensional Signal Processing Technical Committee (IEEE IMDSP TC), Multimedia Signal Processing Technical Committee (IEEE MMSP TC), and ICME Steering Committee. Jean-Luc Dugelay serves as a Consultant for several major companies; in particular, France Telecom R&D and STMicroelectronics.

Emmanuel Garcia graduated from the Ecole Polytechnique, Paris, France, in 1998 and from the Ecole Nationale Supérieure des Télécommunications, Paris, France, in 2000, and is currently a Ph.D. student in the Multimedia Department of the Institut Eurécom, France. His research interests include 3D facial animation and modeling, and 3D object watermarking, in the context of virtual talking heads.