3D Object Processing:
Compression, Indexing and Watermarking

Atilla Baskurt, Mohamed Daoudi, Jean-Luc Dugelay
Contents

1 3D object Watermarking ........................................... 1
  1.1 Introduction .................................................... 1
  1.2 Basic review on watermarking ................................. 1
  1.3 Watermarking principles applied to 3D objects ............. 5
     1.3.1 Aims of watermarking .................................. 5
     1.3.2 Trade-off between capacity, visibility and robustness .. 6
     1.3.3 Embedding space and watermark detection requirements .7
     1.3.4 Manipulations and attacks on 3D objects ............... 8
  1.4 A guided tour of 3D watermarking algorithms ............. 13
     1.4.1 Data file organization .................................. 14
     1.4.2 Topological data ....................................... 15
     1.4.3 Geometrical data ....................................... 17
     1.4.4 Others ................................................... 30
  1.5 Concluding remarks ........................................... 31
  1.6 Questions and problems ....................................... 34
     1.6.1 Watermarking questions ................................ 34
     1.6.2 Watermarking problem .................................. 34
Bibliography ...................................................... 38
3D object Watermarking

1.1 Introduction

The digital revolution, the explosion of communication networks, and the increasingly growing passion of the general public for new information technologies led to exponential growth of multimedia document traffic (image, text, audio, video, 3D objects, etc.). This phenomenon is now so important that insuring protection and control of the exchanged data has become a major issue. Indeed, due to their digital nature, multimedia documents can be duplicated, modified, transformed, and shared very easily. In this context, it is important to develop systems for copyright protection and digital rights management (DRM) in general, copy or access control, and content authentication. Traditionally, such issues were handled by data encryption. However, once data were encrypted by an authorized user they could be distributed and manipulated. Watermarking is a promising alternative solution for reinforcing the security of multimedia documents. In this chapter, we will present an overview of the basic notions and principles of 3D object watermarking along with a review of state of the art algorithms.

1.2 Basic review on watermarking

In this section, we recall some basic watermarking notions, using the special case of images, which was historically the first type of digital documents investigated for watermarking (see Fig. 1.1).

Image watermarking is the technique that aims at imperceptibly embedding 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 watermarked), or whether some specific information (i.e. a message) is actually embedded in the image, or to determine the actual information that was embedded in the image (see Fig. 1.2). Thus a watermarking systems consists of two
basic modules, namely the watermark embedding (signing) module and the watermark detection/extraction (retrieving) module.

Watermarking techniques can be separated in two major categories with respect to the information conveyed by the watermark:

- **Zero bit**: watermarking techniques in this category can only verify whether the data are watermarked or not.

- **Multiple bit**: these techniques can encode a message consisting of a number of bits in the host data. In such systems, the image under investigation is first tested to verify whether it hosts a watermark or not. If the image is indeed watermarked, the embedded message is decoded.

With respect to the method used for watermark detection, techniques can be classified into the following categories:

- **Blind**: the only information required to extract the watermark from the image under investigation is the watermark secret key.

- **Non-blind**: the watermark can be extracted provided that one knows not only the image under test, but also the original image from which it is thought to be derived. Obviously, the requirement that the original image is available during watermark extraction implies serious limitations for the applicability of such algorithms in a number of scenarios. For example non-blind techniques
cannot be used for the automatic search over the Internet in order to detect illegal copies of images.

In a similar way, watermarking techniques can be distinguished into two major categories with respect to the way embedding is performed.

- **Blind embedding techniques.** Such techniques consider the host image as noise or interference. In most cases, these methods utilize knowledge of the host signal statistics.

- **Informed coding/embedding techniques.** These techniques exploit the fact that during embedding, not only the statistics of the host image but also the image itself is known and try to utilize this fact in order to improve watermark detection performance (e.g., through interference cancellation).

Image watermarking has a wide range of applications. A non-exhaustive list of such applications is provided below:

- **Integrity checking, content authentication:** the aim is to detect if an image has been tampered or not. This is usually achieved by inserting a fragile watermark that is destroyed (i.e., it becomes undetectable) if the image is modified. In some cases, content authentication watermarking techniques can provide information about the exact locations of the image that have been altered.

- **Traitor-tracing, transaction tracking:** the aim is to trace back a malicious user, e.g., a user that legally possesses an image but has distributed it to other users in an illegal way. This can be done by embedding a robust watermark identifying the customer in each legal image transaction. The term fingerprinting is often used to describe this application.
- **Owner identification, copyright protection**: the aim is to verify that a given document originates from a certain source even after it has been manipulated and tampered, either in an intentional attempt to remove the watermark or during usual operations (e.g. image compression).

- **Usage Control**: In this case the embedded information controls the terms of use of the digital content. As an example, the embedded information can be used along with compliant devices to prohibit unauthorized recording of a multimedia document (copy control), or playback of illegal copies (playback control).

Another fundamental notion in watermarking is the trade-off between capacity, visibility and robustness (see Fig. 1.3).

- The capacity of a multiple bit watermarking algorithm is the amount of information, i.e., the length of the message, that can be hidden in the watermarked image. It should be noted that, in most cases, data payload depends on the size of the host data. The more the available host samples, the more bits can be hidden. Thus, capacity is often given in terms of message bits per sample of the host image.

- The term visibility refers to image visual degradation due to the embedding of the watermark. Naturally, a watermark should be as invisible as possible and thus, the watermarking process should not introduce suspicious perceptible artifacts. In other words, a human observer should not be able to detect if some digital data has been watermarked or not. The visibility is often expressed quantitatively as the signal-to-noise ratio (SNR) between the marked image and the non-marked image, despite the poor performance of SNR in capturing the way humans perceive distortions.

- Robustness refers to 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 content, this notion being partly subjective). The alterations can be malicious or not: they can result from common image processing or transmission (i.e. filtering, lossy compression, noise addition) or from an attack attempting to remove the watermark (e.g. the Stirmark attack). Robustness is evaluated by measuring the survival rate of watermarks after attacks.

For a given algorithm, improvements in performance with respect to one aspect usually result in performance degradation with respect to one of the two others.

Numerous still image watermarking algorithms exist in the literature. These algorithms differ on various aspects:

- The selection of the locations in the host image where the watermark is to be embedded. This should be done carefully in order to minimize the distortion or to ensure that the existence of the watermark will remain secret. A
direct consequence of Kerckhoffs' principle is that the watermarking algorithm should be public and that the watermark should not be "detectable" in a straightforward manner in order to prevent its removal.

- The choice of the embedding domain: the embedding operation can be performed in the spatial domain, or in a suitable transform domain such as DCT, Mellin Fourier or Wavelet.
- The formatting of the message: some techniques allow any bitstream to be directly embedded as a watermark, others require a transformation of the message bits prior to the embedding process.
- Embedding procedure: selection of the rule that will be used for modifying the selected features of the host image so as to embed the watermark. Additive and multiplicative embedding rules are often used.
- Optimization of the watermark detector: designing the detector module in a way that optimizes watermark detection performance, especially in the case of attacks.

For a more detailed description of image watermarking principles and methods, we refer the reader to Cox et al. (2002); Katzenbeisser and Petitcolas (2000); Tefas et al. (2005).

1.3 Watermarking principles applied to 3D objects

1.3.1 Aims of watermarking

3D Watermarking is a hot topic in the watermarking community. Similar to image watermarking, 3D watermarking aims at hiding in an invisible way information inside a 3D object. This information can then be recovered at any time, even if the 3D object was altered by one or more nondestructive attacks, either malicious or not. 3D watermarking can be useful in several application, security-related ones being the most prominent. For example, users would like to check if the use of a given object is legal or not, to access additional information concerning the object.
3D OBJECT WATERMARKING

(e.g. for authentication or indexing), the owner (copyright), or even the buyer (e.g. for traitor tracing).

The fact that the creation of 3D models is, in many cases, a labor-intensive and costly procedure made the protection of such models an urgent necessity and attracted the interest of many engineers and researchers toward 3D watermarking. As a result, numerous watermarking algorithms for 3D objects have been proposed in the literature.

1.3.2 Trade-off between capacity, visibility and robustness

By analogy with still image watermarking, watermarking of 3D objects involves a trade-off between capacity, visibility and robustness:

1. Capacity: the amount of information that can be hidden in the 3D object. It should be noted that this amount is closely related to the complexity of the object (e.g. number of vertices, curvature variations) and to the application aimed by the watermarking algorithm (e.g. authentication vs copyright).

2. Visibility: the visual degradation of a 3D object due to watermarking. The visual impact of the watermark on the protected 3D object should be as limited as possible. In order to measure the imperceptibility of the embedded watermark in 3D objects represented as meshes one can use:

(a) Existing metrics

- Hausdorff distance:

\[
H_{\text{max}}(M_1, M_2) = \max \{ \max_{a \in M_1} \min_{b \in M_2} d(a, b), \max_{a \in M_2} \min_{b \in M_1} d(a, b) \} \tag{1.1}
\]

where \(M_1\) and \(M_2\) are the two meshes and \(d(a, b)\) is the Euclidean distance between \(a\) and \(b\) in the 3D space. This metric is usually called \((\text{maximum geometric error})\). Another definition of the Hausdorff distance called \((\text{mean geometric error})\) is defined as

\[
H_{\text{mean}}(M_1, M_2) = \frac{1}{A_{M_1} + A_{M_2}} \left( \int_{a \in M_1} \min_{b \in M_2} d(a, b) + \int_{b \in M_2} \min_{a \in M_1} d(a, b) \right) \tag{1.3}
\]

Hausdorff distance is sensitive to linear transformation and computationally expensive.

- Vertex Signal to Noise Ratio (VSNR):

\[
VSNR = \frac{1}{N} \sum_{i=0}^{N-1} d(a_i, b_i). \tag{1.5}
\]
3D OBJECT WATERMARKING

- Geometric Laplacian (GL)

\[ GL(p_i) = p_i - \frac{\sum_{j \in N(p_i)} d(p_i, p_j)^{-1} p_j}{\sum_{j \in N(p_i)} d(p_i, p_j)^{-1}} \]  

(1.6)

where \( N(p_i) \) is the neighborhood of the vertex \( p_i \) and \( d(p_i, p_j) \) the Euclidean distance between \( p_i \) et \( p_j \). GL differentiates random noise addition and a compressed version and captures the local smoothness of the mesh. But usually do not capture the perception of distortions.

(b) Image-based metrics

It consists in comparing 2D projections of the 3D objects. Lindstrom and Turk (2000) compute root mean square differences of pairs of corresponding 2D views. The main problem of this metric is that perceived degradation of still images is usually different from perceived degradation on the 3D object (experiments of E.Rogowitz and E.Rushmeier (2001)).

(c) Evaluation of 3D watermarking perceptual quality.

Gelasca et al. (2005) have published the first paper for this particular problem. They have proposed a roughness based metric validated through psycho-visual experiments.

3. Robustness: the ability to recover the watermark even if the watermarked 3D object has been manipulated. 3D watermarking algorithms exhibit different degrees of robustness to various manipulations of the host 3D objects. Most algorithms are robust to global translation, rotation or scaling of the object, and noise addition to the 3D coordinates of its vertices, whereas fewer algorithms are robust to more challenging attacks like cropping or remeshing (e.g. mesh simplification aimed at reducing the number of triangles to speed up manipulations and rendering).

1.3.3 Embedding space and watermark detection requirements

One of the main characteristics of a watermarking algorithm is the space or domain in which the watermark is embedded (i.e. workspace). This domain is closely related to the representation used for the object (e.g. mesh, Nurbs, etc.), but it may also be a transformed version of this representation. For instance, watermarking of objects represented by 3D meshes can be performed either by modifying the positions of mesh vertices or on the spectral decomposition of the mesh.

Similar to still image watermarking algorithms, 3D object watermarking algorithms can be distinguished into two different classes according to the extraction procedure.

- Blind algorithms: only the private key is needed for watermark extraction.
• Non-blind algorithms: to extract the watermark, one should possess not only the 3D object that is to check, but also the original 3D object from which it is believed to be derived.

### 1.3.4 Manipulations and attacks on 3D objects

**Attacks affecting the data organization within the file**

Mesh representations usually consist of a list of vertices and a list of triplets of these vertices, each defining a different triangle. The order of the vertices or triplets within these lists does not affect, in general, the shape of the represented 3D object. This order can thus be manipulated without modifying the 3D object itself (see Fig. 1.4). As it will be shown later on, some watermarking algorithms use data reordering in order to perform watermark embedding. However, most watermarking algorithms do not modify this order but they rely on the given order for the definition of the embedding positions during watermark insertion and extraction. Obviously, if the vertices or triangles are reordered after watermarking, the watermark extraction becomes impossible and thus this operation can be a relevant attack for this class of algorithms.

Similarly, such degrees of freedom in the organization of the data representing an object may exist with other representations such as NURBS or CSG trees for instance. Whenever a watermarking algorithm relies on a specific way of data organization within the file, a successful attack consists in modifying this organization without modifying the 3D object represented by the data.

**Attacks affecting the geometric representation**

A specific 3D object can be defined in a number of different ways within a given representation principle. For instance, the surface of a 3D object can be meshed in different ways (see Fig. 1.5). It is thus possible to alter the mesh representation of a 3D object into another one while preserving its shape (up to some precision level). This attack would be efficient against watermarking algorithms that rely on a given mesh representation to embed and extract the watermark.

The same remark applies to other representations: different NURBS parameterizations can be used to describe the same surface. Thus an algorithm that relies on a given NURBS representation is vulnerable to attacks that change the NURBS parametrization of the object.

**Noise on geometry**

Assuming that the watermark is encoded through minor modifications of the geometrical entities used to define a 3D object, e.g. by slightly altering the location of the vertices of a mesh, then the watermark may be affected by noise applied on the geometry (see Fig. 1.6). Such a noise could be introduced either maliciously to
Figure 1.4 Example of a manipulation that affects the data organization within a file representing a 3D object. Each triplet of real values defines a vertex whereas each triplet of integers corresponds to the vertex indices that define a triangular face. The integer 3 indicates the number of vertices per face.

weaken the watermark, or in the course of typical 3D object manipulations like lossy compression, or format conversion.

Global transformations

A watermarked object can be subject to an affine transform like translation, rotation, uniform or non-uniform scaling, or even a projective transform or another non-affine global transform (see Fig. 1.7). Some algorithms rely on the precise position and orientation of an object to extract the watermark. In this case, a global transformation applied, maliciously or not, to the object can hamper watermark detection unless a way is found to recover the object’s original reference frame used for watermarking.

Cropping of geometry

Similar to images, it is possible to remove part of an object’s geometry (see Fig. 1.8). In some cases this would yield a meaningless object, but in other cases the remaining part could still be of some value, whereas the watermark may be destroyed in the process. In fact, cropping would often destroy the structure of the watermark embedding space (e.g. the ordering of vertices), unless the watermarking algorithm is designed to be robust to cropping, which may involve hiding the watermark several times in different parts of the object.
Figure 1.5 Manipulations affecting the representation of the 3D object.

Figure 1.6 Random noise applied to the geometry of a 3D mesh with different magnitude.
Mesh simplification attacks

The goal of mesh simplification is to speed up manipulations and rendering of 3D objects. It consists in displaying the 3D polygonal mesh with fewer triangles while preserving the same shape (see Fig. 1.9). It is a challenging attack for 3D watermarking community.

Readers are invited to refer to the compression chapter for more details about this attack.
Mesh smoothing

Meshes obtained from real world objects are often noisy. Mesh smoothing relocates mesh vertices to improve the mesh quality while keeping the mesh topology unchanged (see Fig. 1.10). Several techniques can be used for mesh smoothing, we can cite the umbrella-operator smoothing and the Taubin’s lambda-mu smoothing (see Fig. 1.11).

Generic attacks

Watermarking of 3D objects is subject to generic attacks on watermarking protocols. Such attacks include collusion and de-watermarking. De-watermarking attacks are methods that try to remove the watermark by taking advantage of the fact that
the watermarking algorithm (especially its embedding space and its weaknesses) is known.

On the other hand, a collusion attack involves several people that possess the same watermarked object, hosting different watermarks, and cooperate in order to remove or detect the watermark. In fact, by simply averaging the various watermarked instances of the object into a single object, the different watermarks might be averaged (due to their random nature) into a weak noise, while the object itself would remain intact. Another collusion attack consists in combining pieces from various watermarked instances of the same object so as to obtain a complete object featuring pieces from different watermarks. It would thus be impossible to detect any complete watermark in the resulting composite object.

1.4 A guided tour of 3D watermarking algorithms

In this section, we briefly describe the most cited algorithms for 3D models in order to provide a review of the state of the art in 3D watermarking algorithms.

The reviewed algorithms are classified according to two criteria.

The first criterion is the method used to embed the watermark in the object. According to this criterion, watermarking algorithms are classified to the following categories:

1. Data file organization: these algorithms encode information by modifying the organization of the data in the computer file associated with the 3D object.

2. Topological data: these algorithms, that operate on mesh data, use the topology of the 3D object, i.e., the connectivity of the mesh to embed data. The geometry of the mesh, i.e. positions of vertices, is not modified.
3. Geometrical data: these algorithms are based on slight modifications performed on the geometric data of the 3D object. This category is the largest one and has been organized by starting with algorithms that operate in the spatial domain followed by approaches using some multi-resolution representation and then techniques that work in the spectral domain.

The second criterion used for our classification of 3D object watermarking algorithms, is the representation used to define the 3D object. The following main classes are used in the classification presented below:

1. Mesh.
2. NURBS.
3. Others.

1.4.1 Data file organization

This category includes only three algorithms, each operating on a different 3D object representation scheme namely 3D meshes, NURBS and CSG trees.

Algorithm 1 Ichikawa et al. (2002): mesh, reordering of vertices or triangles

This algorithm encodes information in a mesh represented by a list of vertices and a list of triangles. To achieve this, it modifies in the computer file, the order of the triangles in the list of triangles, or the order of the triplet of vertices forming a given triangle. This algorithm does not modify neither the geometry nor the topology of the mesh.

Algorithm 2 Ohbuchi and Masuda (n.d.); Ohbuchi et al. (1999): NURBS, reparameterisation

These algorithms hide data in a NURBS curve or surface by changing the parameter involved in the NURBS parametrization through a homographic function. Such a change of parameter has three degrees of freedom that are reduced to one by imposing some constraints. The remaining degree of freedom is used to encode the hidden information. Changing the parameter also involves changing the nodes and weights accordingly in order to preserve the geometry. Thus, this algorithm does not modify the geometry.

Algorithm 3 Fornaro and Sanna (2000): CSG tree

This algorithm aims at hiding information in a 3D model described by a CSG tree. For this, a new kind of nodes for the CSG are defined: the watermark (control) nodes. Nodes in this new category are linked to the original CSG and contain the watermark information. To achieve invisible watermarking, control nodes are null volume objects (e.g. null-radius spheres). The information that is hidden in those
nodes is a hash value of the original 3D model, encrypted by the secret key of an asymmetric cryptographic algorithm. The targeted application is thus the authentication of the object that is achieved by comparing the computed hash value of the object under investigation against the decrypted hash value hidden in the new nodes.

1.4.2 Topological data

This class of algorithms uses the topology (connectivity) of the 3D object to insert the watermark. All algorithms reviewed in this section operate in 3D objects defined as a 3D mesh.

**Algorithm 4** Ohbuchi et al. (1997): mesh, “Triangle Strip Peeling Symbol sequence embedding” (TSPS)

This algorithm uses the topology of the polygonal 3D model but does not modify its geometry. Starting from an initial edge of the mesh, a strip of triangles is defined by successively attaching a new triangle to the current strip as follows: the last triangle of the current strip has two “free” edges, (i.e. edges that are not connected to the strip) each corresponding to a triangle that is not yet attached to the strip. The selection of the next triangle to be attached to the strip among the two possible choices is based on the next bit of the message to be hidden. Once the strip is defined, it is “peeled off” by duplicating all edges and vertices on the boundary of the strip except from the starting edge. Thus the strip is connected to the rest of the mesh only by the starting edge (see Fig. 1.12). This “peeling” of the strip changes the topology but is invisible since its boundary edges and vertices coincide with those defining the corresponding boundary on the remaining of the mesh. For message extraction, the starting edge of the strip is found and, as the strip is traversed, the embedded bits are extracted based on the strip’s path.

**Algorithm 5** Ohbuchi et al. (1997): mesh, “Polygon Stencil Pattern” (PSP)
Given a 3D object described by a mesh, a visual pattern (e.g., a letter) can be hidden in it by "peeling off" a triangle strip that has the shape of the pattern to be hidden (e.g., a strip that forms the letter to be hidden (see Fig. 1.13)). Just like the algorithm 4, the strip is "peeled off" by duplicating the edges and vertices that form the boundary of the strip. Thus, only the topology of the mesh is modified, but not the geometry. Furthermore, when the "peeled" triangle strip is displayed along with the rest of the mesh, no alterations are visible. This algorithm is somewhat robust to remeshing.

![Figure 1.13 3D object watermarked using the 'Polygon Stencil Pattern' technique.](image)

**Algorithm 6 Ohbuchi et al. (1997): mesh, “Mesh Density Pattern embedding”**

This algorithm locally modifies the density of triangles in a mesh so as to construct a visual pattern, observable when viewing the 3D object in wireframe mode (see Fig. 1.14). This visible watermark is robust to geometrical operations Rotation, Scaling and Translation (RST) and is resistant but not immune to polygonal simplification and other topology manipulations.

**Algorithm 7 Mao et al. (2001): mesh, triangle subdivision**

This algorithm re-triangulates a part of a triangle mesh and embeds the watermark into the positions of the newly added vertices. Triangles are chosen according to a secret key in such a way that the ratio of two line segments lying on the same straight line encodes the hidden information (see Fig. 1.15). The embedded watermark can be extracted only from the stego-model without using the original cover model. This algorithm achieves high capacity and is robust to affine transformations whereas it is easily destroyed by local deformations and topological alterations.
1.4.3 Geometrical data

The majority of the 3D watermarking algorithms insert the watermark by modifying the geometry of the 3D object. Most of techniques deal with mesh data. In this category, some of them operate in the spatial domain by modifying vertices, normals (direction, length) and geometrical invariants (i.e. length of a line, area of a polygon, etc.). Others, embed information in a transform domain: spectral decomposition, wavelet transform and spherical wavelet transform.

Watermarking algorithms operating on 3D models represented by NURBS, point sets or other forms of representation are less widespread.

In this section, we present a brief description of the most known techniques.
A- Mesh representation

AI- Spatial domain

Algorithm 8 Bors (2004a,b, 2006); Harte and Bors (2002): mesh, vertex displacement

This watermarking algorithm is based on the position of the vertices and not on their connectivity. It consists of two steps. In the first step, a list of vertices and their neighborhoods are selected from mesh areas consisting of small polygons and are ordered according to their Euclidean distance from the vertex that is the head of the list (i.e., the vertex with the smallest squared distance to its neighbors). In the second step, locations of selected vertices are changed according to their local moments and the information bit to be embedded (see Fig. 1.16). The watermark can be recovered after scaling, rotation or a combination of geometrical transformation. Robustness to various levels of noise added on vertices as well as to 3D object cropping was experimentally verified.

![Figure 1.16](image)

Figure 1.16 $P_0$ and $P_w$ are respectively the original and watermarked vertices. The configuration (a) embed a bit '0' and the configuration (b) embed a bit '1'.


This non-blind watermarking algorithm operates in the spatial domain. It inserts the watermark signal $(W_{xi}, W_{yi}, W_{zi})$ into the vertex coordinates $(X_i, Y_i, Z_i)$ using the following equations:

\[
\begin{align*}
X'_i &= X_i + a \cdot M_x(p_i) \cdot W_{xi} \\
Y'_i &= Y_i + a \cdot M_y(p_i) \cdot W_{yi} \\
Z'_i &= Z_i + a \cdot M_z(p_i) \cdot W_{zi}
\end{align*}
\]
with the masking functions \((M_x(p_i), M_y(p_i), M_z(p_i))\) defined as the difference of coordinate values of each point with the points connected to it.

This algorithm is robust to additive noise, geometry compression performed by the MPEG-4 SNHC standard and mesh simplification.

**Algorithm 10** Barni et al. (2004): mesh, vertex displacement

This algorithm perturbs vertices positions of the 3D model according to a spherical pseudo-random bumped surface. The pseudo-random position and amplitude of the bumps encode the watermark. Watermark recovery is achieved via a standard correlation detector. This algorithm can achieve a good degree of robustness in case of 3D objects with a fairly large number of faces.

**Algorithm 11** Yeung and Yeo (1998), Yeo and Yeung (1999): mesh, fragile watermark

To check the integrity of a triangular mesh, Yeung and Yeo have developed a blind fragile watermarking algorithm that slightly moves each vertex.

The insertion process is as follows:

1. Compute a double position index \(L = (L_x, L_y)\) for each vertex \(v\)
   - Define the centroid \(s\) of the set of vertices adjacent to \(v\),
     \[
     s = \frac{1}{|N(v)|} \sum_{u \in N(v)} u
     \]  
   - Convert \(s\)'s coordinates into integers \((N_x, N_y, N_z)\).
   - Combine these integers to obtain \(L_x\) and \(L_y\). An example of combination is \(L_x = (N_x + N_y + N_z) \mod \text{XSIZE}\).

2. Compute a triple value index \(p(v) = (p_1, p_2, p_3)\) for each vertex \(v\)
   - Convert \(v\)'s coordinates into a triplet of integers.

3. Generate a binary sequence using a secret key \(K\) and values of indices \(p(v)\).
   Each \(p(v)\), given as input to a conversion table parameterized by \(K\), yields a binary value \(K(p(v))\).

4. Slightly modify the geometry of the model (i.e. the vertices \(v\)) so that the bit \(W(L(v))\) of the watermark \(W\) (defined as a black and white two-dimensional image) is equal to \(K(p(v))\).

The verification step consists in evaluating \(K(p(v))\) and \(W(L(v))\) for each vertex \(v\), and then computing a correlation score \(c\) between the two binary sequences:

\[
c = \frac{|\{v : K(p(v)) \neq W(L(v))\}|}{|\{v\}|}
\]  
(1.8)
If the object has not been modified, the correlation score $c$ is equal to 1. This method is not robust to vertex renumbering due to the definition of the centroid $s$ in the insertion step.

**Algorithm 12** Benedens (1999b); Benedens and Busch (2000): mesh, “Vertex Flood”

This high-capacity method hides a watermark into a 3D object represented by a set of triangles. The algorithm, which affects only the geometry of the object, modifies the distance between vertices of the model and the center of gravity of a given reference triangle for each selected interval. In the retrieval process, the mean distance of all vertices included in an interval can be used for decoding the embedded bits. This way can increase robustness to randomization of simple vertices.

**Algorithm 13** Yu et al. (2003), Zhi-qiang et al. (2003): mesh, vertex displacement

This algorithm is an extension of algorithm 12. It partitions the vertices of a mesh into $N$ sets, pseudo-randomly defined from a secret key. The watermark is encoded in the distance between a vertex and the barycenter of the set it belongs to as follows:

$$L_{wij}^w = L_{oij} + \alpha W_i U_{oij}$$

where $L_{oij}$ denotes the original vector from the center of the model to the $j^{th}$ vertex of the $i^{th}$ section and $U_{oij}$ denotes the vector whose direction is the same as that of $L_{oij}$ and its amplitude is the minimal length of the $j^{th}$ vertex’s 1-ring edge neighborhood. The watermark retrieval is achieved by first computing an estimation of the watermark. For this, the algorithm evaluates for each set the difference in length between the vectors of the original model that link the vertices to the center of the set and the vectors of the detected model that also link the vertices to the set center. Then correlation coefficient between the extracted watermark sequence and the designated watermark sequence can provide information about the watermark presence or not. Experiments show that this method is robust to many attacks, especially cropping and noise, and is made robust to remeshing by the inclusion of a registration/resampling step that aims at aligning the mesh under investigation with the original mesh before watermark extraction.

**Algorithm 14** Benedens (1999b); Benedens and Busch (2000): mesh, “Triangle Flood”

This is another high-capacity algorithm. The targeted applications are the same as for algorithm 12. The algorithm first generates an order for walking over the set of triangles, and then hides information in the heights of the successive triangles by slightly modifying the position of their vertices.

**Algorithm 15** Ohbuchi et al. (1997): mesh, “Triangle Similarity Quadruple” (TSQ)
Figure 1.17 Dimensionless quantities for a triangle: a/b and h/c.

This algorithm is based on the modification of the geometry of a triangular mesh. For each triangle a couple of dimensionless quantities are considered (see Fig. 1.17). These quantities may be for example the length ratio of two edges of a triangle (a/b) or the length ratio between one edge and the corresponding height (h/c). An integer value is encoded into a triangle by slightly modifying the associated quantities.

In fact the building block, Macro Embedding Primitive (MEP) for encoding information is a set of four adjacent triangles (quadruple) as shown in Fig. 1.18. The integer value hidden in one of the four triangles (marker) identifies the MEP. Two other triangles encode a part of the payload (data 1 and data 2). The last triangle encodes a sequence number, indicating how to assemble the payload pieces obtained from the various quadruples into the original complete message (subscript). The extraction of the watermark consists on traversing the triangular mesh and finding all the triangles with the marker. Then the two data symbols and the subscript are extracted from the triangles in the MEP.

Algorithm 16 Ohbuchi et al. (1997): mesh, “Tetrahedral Volume Ratio embedding” (TVR)

This algorithm hides a piece of information in a triangular mesh by modifying an affine invariant value of a pair of tetrahedrons. This affine invariant is the ratio of their volumes. A tetrahedron is simply defined by an edge and its two adjacent triangles.

Algorithm 17 Cayre and Macq (2003): mesh, variant of TSPS

This algorithm considers that each triangle is a two-state variable. The state of a triangle is determined by the position of the projection of one of its vertices on the opposite edge. This state can be modified by slightly moving the relevant vertex as shown in Fig. 1.19. The set of triangles that encode the binary message is defined as a strip of triangles by a procedure analogous to that of algorithm 4 (TSPS): an
Figure 1.18 Macro Embedding Primitive for the TSQ watermarking technique. The marker (M) is embedded by slightly changing the coordinate of vertices $v_1$, $v_2$ and $v_4$ such that \{${e_{14}}/{e_{24}}$, $h_4/{e_{12}}$\} = ${b/a, h/c}$ with $e_{ij}$ the distance between vertices $v_i$ and $v_j$. The subscript (S) is coded in the pair \{${e_{02}}/{e_{01}}$, $h_0/{e_{12}}$\} by slightly changing the coordinate of vertex $v_0$. Data1 ($D_1$) and data2 ($D_2$) are coded in pairs \{${e_{13}}/{e_{34}}$, $h_3/{e_{14}}$\} and \{${e_{45}}/{e_{25}}$, $h_5/{e_{24}}$\} by slightly changing the coordinate of vertices $v_3$ et $v_5$. The initial triangle is chosen and ones moves from one triangle to one of the two possible adjacent triangles, as dictated by a pseudo-random binary sequence. The difference with the TSPS algorithm is that in this algorithm the hidden data are not encoded in the shape of the strip of triangles. Furthermore, the triangle strip is not cut out of the object’s surface, thus the topology is not modified.

**Note:** Watermarking algorithms that operate in the spatial domain via slight modification of the vertices coordinates generally are not robust to noise addition.

**Algorithm 18 Wagner (2000): mesh, length of normals**

This algorithm embeds the watermark in the length of the “normals” $n_i$ defined on each vertex. The normal at a vertex is defined as the norm of the vector between the barycenter of the vertex’s neighbors and the vertex itself. Modifying these normals by replacing some bits of $n_i$ with the bits of the watermark implies moving the vertices. The watermark extraction can be achieved by calculating the value of the normal vector length and extracting the appropriate bits in which the watermark is hidden. The use of a norm invariant to affine transformations yields a watermark that is robust to affine transforms of the 3D object. The paper of Maret and Ebrahimi (2004) proposes an extension that increases the embedding capacity.
Figure 1.19 A bit '1' is encoded in both cases (a) and (b).

**Algorithm 19** Benedens (1999a,c); Benedens and Busch (2000): mesh, direction of normals

The watermark embedding space used in this algorithm is the direction of the normals of the object triangles. The unit sphere, representing the set of all oriented directions, is divided into regions. The normal of each triangle belongs to one of these regions. The dispersion of the set of normals belonging to a given region of the unit sphere is actually used to encode a bit. To encode a 0 the dispersion of normals in a given region is decreased. To encode a 1 it is increased. This is achieved by altering the normals, and ultimately the vertices of the object. The regions of the unit sphere that are chosen to encode a bit are determined by a secret key.

**Algorithm 20** Kwon et al. (2003): mesh, direction of normals.

This algorithm is an improvement of algorithm 19. The embedding space is the histogram of the normals’ direction on the unit sphere. The algorithm achieves robustness against cropping by hiding the watermark in several parts of the object. Furthermore, the authors claim that the algorithm is robust to remeshing and isometries.

**Algorithm 21** Song and Cho (2004); Song et al. (2002): mesh, cylindrical depth map

This algorithm computes a cylindrical depth map associated to a mesh (see Fig. 1.20). The cylindrical frame of reference is aligned on the principal axis of inertia of the 3D object. The depth map is watermarked using a 2D image watermarking algorithm and the modifications of the depth map are applied back to the original mesh object.

**Algorithm 22** Kalivas et al. (2003): mesh, principal component analysis.
This algorithm is based on the still image watermarking technique originally proposed by Tefas and Pitas (2001). Robustness to translation is achieved by translating the model so that its center of mass falls on the origin of the coordinate system. To achieve robustness against rotation, the model is rotated so that the principal component of the vertices (the eigenvector that corresponds to the greatest eigenvalue of the covariance matrix of vertex coordinates) coincides with the z axis. These operations are performed both before watermark embedding and detection. Subsequently, the model vertices are represented in spherical coordinates \((r; \theta, \phi)\), ordered according to their \(\theta\) value and their \(r\) value (distance from the center of mass) is modified according to the watermark bits. Essentially this procedure corresponds to performing the embedding on the 1D signal \(r(\theta)\). The fact that only the \(r\) value is changed by the embedding process, makes the method robust to uniform scaling.

**Algorithm 23** Zafeiriou et al. (2005): mesh, SPOA (Sectional Principal Object Axis watermarking).

This algorithm is an improvement of the algorithm 22. Instead of using one vertex for the embedding of a bit of the watermark sequence, this method embeds a watermark bit in multiple vertices. More specifically, the model is translated and rotated so that its center of mass coincides with the origin of the coordinate system and the principal component is aligned with the z axis. Then vertices are transformed to spherical coordinates \((r, \theta, \phi)\) and the range of \(\theta\) values is split into a number of consecutive non-overlapping intervals. The vertices whose \(\theta\) values lie in two consecutive intervals are used to embed one watermark bit. This is achieved by modifying the \(r\) values so as to enforce certain constraints. Detection is performed by checking the validity of these constraints on the model under investigation. The proposed algorithm is robust to rotation, translation, uniform scaling, noise addition and mesh simplification but is vulnerable to cropping.

**Algorithm 24** Koh and Chen (1999): mesh, progressive transmission
This algorithm aims at watermarking the stream of data used to send a mesh in a progressive manner over a network. Each vertex is transmitted one by one, in a “progressive” order. The watermark is in fact embedded in the 1D signal consisting of the sequence of transmitted vertices. It is possible to add a pseudo random sequence to the 1D signal to get it watermarked. The watermark is detected upon reception of the data stream. The correlation coefficient between the embedded watermark and the transmitted signal informs if the stream is watermarked or not (i.e., it is a zero-bit watermarking algorithm).

A2: Transform domain

Algorithm 25 Kanai et al. (1998): mesh, wavelet decomposition

This algorithm performs a wavelet decomposition of the mesh. The watermark is then embedded in the wavelet coefficients, and the watermarked mesh is obtained by reconstruction from the modified wavelet coefficients. Only the wavelet coefficients of significant magnitude are modified. The algorithm is based on the notion of spread spectrum (see Cox et al. (1997)).

Algorithm 26 Uccheddu et al. (2004): mesh, wavelet decomposition.

This blind watermarking algorithm is based on wavelet decomposition of semi-regular meshes. The watermark is embedded in the wavelet coefficients of a suitable resolution level. Watermark detection is accomplished by computing the correlation coefficient between the watermark and the mesh under inspection. Robustness against geometric transformations (rotation, translation, uniform scaling) is achieved by embedding the watermark on a normalized version of the mesh, obtained by means of PCA (Principal Component Analysis). This technique has been extended to irregularly subdivided 3D triangular meshes (Valette et al. (1999)) by using lazy wavelet defined on arbitrary connectivities, but the algorithm still fails after connectivity attacks.

Algorithm 27 Praun et al. (1999): mesh, wavelet decomposition.

This algorithm embeds a watermark in wavelet coefficients computed by the progressive representation of a mesh introduced by Hoppe (1996) and associates a function to each vertex duplication operation. This function defines a deformation of the object by moving the vertices neighboring the duplicated vertex. For inserting the watermark, the duplication operations that result in the biggest modifications of the object’s shape are chosen, and the associated deformation functions are used and weighted by the watermark information to slightly deform the object. The extraction of the watermark is done in a non-blind manner after registering and resampling the tested object with respect to the original object.

Algorithm 28 Yin et al. (2001): mesh, wavelet decomposition.
The authors of this algorithm aim at generalizing the idea of algorithm 27 and applying it on other multi-resolution representations of a triangular mesh. The underlying principle is however somewhat different: a multi-resolution representation of the mesh is constructed and then the watermark is encoded by modifying the shape (via small displacements on a subset of the vertices) of a certain, properly chosen, resolution level (instead of using several resolution levels as in algorithm 27). The watermark extraction is performed after registering and resampling the watermarked object with respect to the original object, so as to be robust to attacks such as format conversion.

**Algorithm 29** Jin et al. (2004): mesh, spherical wavelet transform.

The basic idea of this non-blind algorithm is to decompose the original mesh into some detailed parts and an approximation part by using spherical wavelet transformation. The watermark is embedded into both the detailed and the approximation parts. The embedding process starts with global spherical parametrization, spherical uniform sampling of the host mesh, and spherical wavelet forward transform. Once in the wavelet domain, watermark embedding is performed, the inverse transform is applied and the watermarked mesh is resampled to recover the connectivity of the original model. The watermark detection process includes alignment of the watermarked mesh with the original one, global spherical sampling, spherical wavelet forward transform on both the original and the watermarked model and finally the watermark extraction through comparison of these two meshes. This algorithm is robust to reordering of vertices, mesh simplification and noise addition.

**Note:** Watermarking algorithms operating in the wavelet domain offer a good control of the local distortion caused by the embedding process.

**Algorithm 30** Ohbuchi et al. (2002): mesh, spectral decomposition

This algorithm operates in the mesh “spectral domain”, obtained after diagonalizing the combinatorial Laplacian matrix of the mesh. Diagonalizing a large matrix (whose dimension equals the number of vertices) is a very heavy and numerically unstable operation. That is why the mesh is first divided into smaller regions which are processed independently. The watermark message is duplicated many times and is used to modulate the amplitude of the spectral coefficients. To extract a watermark, the algorithm compares the shape of the reference (i.e. original) mesh with the watermarked (and possibly attacked) mesh in the mesh spectral domain. The extraction requires the information on how the original mesh is partitioned. This non-blind algorithm is robust to several attacks. Prior to detection, a registration/resampling step with respect to the original model is performed to ensure robustness to remeshing.

**Algorithm 31** Cayre et al. (2003): mesh, spectral decomposition

This algorithm is very similar to algorithm 30. The Laplacian matrix used to obtain the frequency decomposition is the one originally proposed by Taubin et al.
(1996) and not the combinatorial Laplacian used in algorithm 30. The difference with respect to algorithm 30 is that each region subject to spectral decomposition is first resampled to obtain a given regular connectivity graph for the vertices (typically, the valence of each vertex is 6 except at edges). Since this connectivity graph is known in advance, the frequency decomposition matrix can be pre-computed. Another difference with algorithm 30 is that it does not perform a registration/resampling step before extracting the watermark. This makes the algorithm non-robust to remeshing, but this is only a “strategic” choice, unrelated to the embedding technique used by the algorithm.

**Algorithm 32** Alface and Macq (2006): mesh, spectral decomposition

This blind algorithm proceeds by first partitioning the mesh using a geodesic Delaunay triangulation of a number of feature points automatically selected through a multi-scale estimation of the curvature tensor field. Then, each of the geodesic triangle patches is parameterized and remeshed so as to generate a robust base mesh. Finally, the remeshed patches are watermarked in the mesh spectral domain following the work of Cayre et al. (2003) (algorithm 31). This algorithm is robust to affine transformations, cropping and connectivity attacks.

**Algorithm 33** Wu and Kobbelt (2005): mesh, spectral decomposition

This algorithm is very similar to algorithm 30. The major difference is that it uses a set of geometry dependent orthogonal basis functions derived from Radial Basis Functions (Carr et al. (2001), Ohtake et al. (2004)) to span the spectral space rather than using Laplace basis functions which emerge from the Laplacien matrix. In that way, this method runs faster and thus can efficiently watermark very large meshes.

**Algorithm 34** Murotani and Sugihara (2003): mesh, Singular Spectrum Analysis (SSA).

This algorithm adds a watermark into a 3D polygonal mesh in the spectral domain. The 3D polygonal model is considered as a sequence of vertices called a vertex series. The spectra of the vertex series are computed using SSA (Galka (2001), Golyandina et al. (2001)) for the trajectory matrix derived from the vertex series. The watermark is added in the spectral domain by modifying the singular values. The watermark can be extracted by comparing in the spectral domain the singular values of the watermarked and the original data. This non-blind watermarking algorithm is robust against similarity transforms and moderate noise added to vertex coordinates. An improvement of this algorithm is proposed in Murotani and Sugihara (2004) to achieve robustness against random noise.

**Note:** Spectral decomposition provides very good robustness against attacks. Spectral decomposition watermarking algorithms are promising. The main weakness is the computational complexity (for example, the diagonalization of a large matrix whose dimension equals the number of vertices is a very heavy and numerically unstable operation).
Algorithm 35 Daras et al. (2004); Jin et al. (2004): mesh, generalized radon transform.

Watermarks generated by this algorithm are to be used for content-based indexing and retrieval of 3D models stored in a database. The proposed approach is based on the use of a Generalized Radon Transform. More precisely a Cylindrical Integration Transform (CIT) is applied to the 3D model to produce descriptor vectors. Each descriptor vector corresponds to a cylinder and furthermore to a set of vertices lying inside the cylinder. The watermark is embedded via a modification of the location of these vertices according to a unique sequence of bits which is used as identifier linking each model to its descriptor vector.

Algorithm 36 Jeonghee et al. (2003): mesh, Discrete Cosine Transform

This algorithm operates in the DCT domain. The 3D mesh is traversed to generate triangle strips and transform their vertex coordinates into frequency coefficients in the DCT domain. The watermark is then embedded into the mid-frequency band of AC coefficients for robustness and imperceptibility. The extraction of the watermark is performed via a comparison of the mid-frequency coefficients of the original and watermarked 3D model. Experiments show that the inserted watermarks survive various attacks, such as additive noise, geometry compression, affine transformation and multiple watermarking.

B- Other representations

Algorithm 37 Benedens (2000): NURBS

This algorithm uses the same principle as algorithm 16 while introducing some small improvements. Its most interesting feature is that it uses a mesh watermarking algorithm to watermark 3D objects represented by NURBS. For this, the NURBS surface is first tessellated, then the mesh watermarking algorithm is applied to the obtained mesh, and finally the deformations induced on the mesh by the watermark are applied to the original NURBS coefficients.

Algorithm 38 Lee et al. (2002): NURBS, orthographic depth maps

This algorithm computes three orthographic depth maps of a NURBS object (see Fig. 1.21) and then watermarks them using some 2D image watermarking algorithm. The modifications of the depth maps induced by the 2D watermarking process are applied back to the original NURBS control points. The selection of the reference frame used to compute the depth maps is controlled by a secret key.


1These names come from the historical use of DCT for analyzing electric circuits with direct- and Alternating Currents.
This NURBS surfaces watermarking algorithm operates on 2D-DCT coefficients by means of a spread spectrum procedure. First three virtual images are computed from the NURBS representation: the pixel values in these images are the coordinates of the NURBS control points. A 2D-DCT is applied to each image to obtain a vector of characteristics. Then, the secret information (key) and public information (logo) are combined by means of a Code Division Multiple Access (CDMA) technique to provide the watermark which is subsequently embedded into the vector of characteristic by means of a weighted addition. Watermark detection is achieved by means of matched filters.

**Algorithm 40** Louizis et al. (2002): voxels, watermarks of specific spatial structure.

This algorithm is based on the image watermarking technique proposed by Tefas and Pitas (2001) and, unlike most algorithms, operates on volumetric (voxel-based) models. A watermark signal of specific structure (having a self-similar nature) is embedded in the spatial domain and enables blind, fast and robust progressive watermark detection even after geometric distortions of the watermarked volume. The term progressive watermark detection means that under certain conditions the watermark detection procedure does not have to be performed on the entire volume; the volume is scanned sequentially until a decision on the existence of a watermark can be reached.

**Algorithm 41** Ohbuchi et al. (2004): point set

This non-blind algorithm is applicable to 3D objects defined as a point cloud, i.e., a set of points that are not connected by edges. It follows the same frequency-domain shape modifications approach as the 3D mesh-watermarking algorithm by Ohbuchi et al. (algorithm 30). For the watermark embedding, the algorithm creates a disjoint set of clusters from the point set, generates a mesh for each cluster of points and applies mesh spectral analysis on each mesh. Then it modifies the mesh spectral coefficients according to the information bit string to be embedded, performs inverse
transform of the coefficients back into the vertex coordinates domain and finally discards the vertex connectivity of the mesh to obtain a watermarked point set. For the watermark extraction, the watermarked point set is aligned with the original one, and its geometry is resampled using the original one. Then the clusters on the reference point set are re-created and the clustering is transferred onto the resampled watermarked one. Subsequently a mesh is created for each cluster, mesh spectral analysis is performed on both the original and watermarked point sets and the embedded bit string is extracted by comparing the coefficients of the corresponding clusters.

1.4.4 Others

Algorithm 42 Ohbuchi et al. (1998a,b): miscellaneous, attributes

These papers refer to techniques that achieve watermarking of 3D objects through the modification of certain attributes attached to its geometry, such as texture coordinates or color/opacity parameters associated with each vertex, line or face. In Ohbuchi et al. (1998a) preliminary results are presented for a watermarking algorithm that operates in the texture coordinates. Other attributes mentioned for possible watermark embedding are VRML scene parameters, or 3D animation tables for deformable models.

Algorithm 43 Hartung et al. (1998): miscellaneous, animation parameters

This algorithm belongs only marginally to the topic covered by this chapter since it involves synthetic video sequences of 3D face models, animated from a stream of MPEG-4 facial animation parameters. What is watermarked here is not the 3D object but the parameter stream. For embedding of watermark data a spread spectrum approach was adopted.

Algorithm 44 Garcia and Dugelay (2003): miscellaneous, Asymmetric 3D/2D watermarking procedure.

This algorithm for watermarking 3D textured objects is based on texture map of the object instead of its geometrical data. The main goal of the algorithm is to hide information in the texture image in such a way that one can retrieve hidden information from images or videos generated from the 3D synthetic object, thus protecting the visual representations of the object.

Given a 3D object consisting of its geometry, a texture image and a texture mapping function, information is embedded in the object by watermarking its texture image via a still image watermarking algorithm. Watermark detection is performed on views (i.e. 2-D projections) of the object and is achieved by reconstructing the watermarked texture and extracting the watermark from the recovered texture image.

Similar to algorithm 44, the main goal of this framework is to retrieve the watermark from images or videos resulting from the 3D synthetic object, thus protecting the visual representation of the object. However, this framework is based on the object’s apparent contour instead of its texture map. Thanks to this point, this approach can be used to protect the images derived (i.e. projected) from a watermarked 3D object with or without texture.

Given a 3D object, its 3D silhouette is extracted, sampled, and watermarked using a robust algorithm designed for 3D polygonal lines in order to get the watermarked 3D object. This object can then be used in virtual scenes or hybrid/natural synthetic videos. To check from a certain view if an object is protected or not, the 2D contour that defines the boundary of the object’s projection is extracted, sampled and the watermark presence is detected.

Fig. 1.22 and Fig. 1.23 summarize all the described techniques included in this chapter.

1.5 Concluding remarks

During the last few years, watermarking of 3D objects has attracted a considerable amount of interest within the watermarking community, although not as much as watermarking of other types of multimedia data like images, audio or video. Although 3D object watermarking algorithms have many things in common with algorithms developed for other types of media, the particularities of 3D data, e.g. the fact that no natural ordering (and thus no globally accepted traversal scheme) can be devised for points in the 3D space or the fact that, unlike other media, watermark imperceptibility should not be judged directly on the 3D object but rather on its projections (images, videos) on the plane, make 3D watermarking a rather distinct and definitely interesting problem. Despite the significant amount of work conducted so far in this area, (especially on mesh models), numerous open issues exist and the problem is far from being considered solved. Obviously, it is rather optimistic to consider that a 3D object watermarking technique that will be robust to all envisioned attacks (including the ones that will be designed in the future) will be introduced in the near future. However, watermarking can be indeed a powerful digital rights management tool provided that the devised techniques are constructed having in mind the needs and challenges of specific applications and environments. The combination of watermarking with other related techniques like encryption or perceptual hashing is also a promising direction toward the successful application of 3D object watermarking in real-world scenarios. The introduction of universally accepted benchmarking procedures, test sets and performance metrics can also help in this direction.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Representation format</th>
<th>Extraction mode</th>
<th>Embedding domain</th>
<th>Robust to remeshing</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mesh</td>
<td>Blind</td>
<td>Computer data</td>
<td>No</td>
<td>reordering of vertices or triangles</td>
</tr>
<tr>
<td>2</td>
<td>NURBS</td>
<td>Non-blind</td>
<td>Computer data</td>
<td>-</td>
<td>reparametrisation</td>
</tr>
<tr>
<td>3</td>
<td>CSG</td>
<td>Blind</td>
<td>Topological data</td>
<td>No</td>
<td>Triangle Strip Peeling Symbol</td>
</tr>
<tr>
<td>4</td>
<td>Mesh</td>
<td>Blind</td>
<td>Topological data</td>
<td>Yes</td>
<td>Polygonal Stencil Pattern</td>
</tr>
<tr>
<td>5</td>
<td>Mesh</td>
<td>Blind</td>
<td>Topological data</td>
<td>Yes</td>
<td>Mesh Density Pattern</td>
</tr>
<tr>
<td>6</td>
<td>Mesh</td>
<td>Blind</td>
<td>Topological data</td>
<td>No</td>
<td>triangle subdivision</td>
</tr>
<tr>
<td>7</td>
<td>Mesh</td>
<td>Blind</td>
<td>Topological data</td>
<td>Yes</td>
<td>Triangle Strip Peeling Symbol</td>
</tr>
<tr>
<td>8</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>vertex displacement</td>
</tr>
<tr>
<td>9</td>
<td>Mesh</td>
<td>Non-Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>vertex displacement</td>
</tr>
<tr>
<td>10</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>vertex displacement</td>
</tr>
<tr>
<td>11</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>fragile watermark</td>
</tr>
<tr>
<td>12</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>Vertex Flood</td>
</tr>
<tr>
<td>13</td>
<td>Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>vertex displacement</td>
</tr>
<tr>
<td>14</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>Triangle Flood</td>
</tr>
<tr>
<td>15</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>Triangle Similarity Quadruple</td>
</tr>
<tr>
<td>16</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>Tetrahedral Volume Ratio</td>
</tr>
<tr>
<td>17</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>Triangle Strip Peeling Symbol</td>
</tr>
<tr>
<td>18</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>length of normals</td>
</tr>
<tr>
<td>19</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>repartition of normals</td>
</tr>
<tr>
<td>20</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>repartition of normals</td>
</tr>
<tr>
<td>21</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>cylindrical depth map</td>
</tr>
<tr>
<td>22</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>principal component analysis</td>
</tr>
<tr>
<td>23</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>Sectional Principal Object Axis</td>
</tr>
<tr>
<td>24</td>
<td>Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>progressive transmission</td>
</tr>
</tbody>
</table>

Figure 1.22 List of 3D watermarking algorithms.
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Representation format</th>
<th>Extraction mode</th>
<th>Embedding domain</th>
<th>Robust to remeshing</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>wavelet decomposition</td>
<td></td>
</tr>
<tr>
<td>26 Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>wavelet decomposition</td>
<td></td>
</tr>
<tr>
<td>27 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>wavelet decomposition</td>
<td></td>
</tr>
<tr>
<td>28 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>multi-resolution representation</td>
<td></td>
</tr>
<tr>
<td>29 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>spherical wavelet transform</td>
<td></td>
</tr>
<tr>
<td>30 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>spectral decomposition</td>
<td></td>
</tr>
<tr>
<td>31 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>spectral decomposition</td>
<td></td>
</tr>
<tr>
<td>32 Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>spectral decomposition</td>
<td></td>
</tr>
<tr>
<td>33 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>spectral decomposition</td>
<td></td>
</tr>
<tr>
<td>34 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>Singular Spectrum Analysis</td>
<td></td>
</tr>
<tr>
<td>35 Mesh</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>No</td>
<td>generalized radon transform</td>
<td></td>
</tr>
<tr>
<td>36 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>Yes</td>
<td>Discrete Cosine Transform</td>
<td></td>
</tr>
<tr>
<td>37 NURBS</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>38 NURBS</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>-</td>
<td>orthographic depth map</td>
<td></td>
</tr>
<tr>
<td>39 NURBS</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>-</td>
<td>spread spectrum</td>
<td></td>
</tr>
<tr>
<td>40 Voxels</td>
<td>Blind</td>
<td>Geometrical data</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>41 Point set</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>42 -</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>other attributes</td>
<td></td>
</tr>
<tr>
<td>43 -</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>animation parameter</td>
<td></td>
</tr>
<tr>
<td>44 -</td>
<td>Non-blind</td>
<td>-</td>
<td>-</td>
<td>asymmetrical approach</td>
<td></td>
</tr>
<tr>
<td>45 Mesh</td>
<td>Non-blind</td>
<td>Geometrical data</td>
<td>-</td>
<td>asymmetrical approach</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1.23 (Fig. 1.22 cont.) List of 3D watermarking algorithms.
1.6 Questions and problems

1.6.1 Watermarking questions

- What are the advantages of watermarking in different domains e.g. DCT (Discrete Cosine Transform) versus DWT (Discrete Wavelet Transform) ?

- Describe the characteristics and differences of the following watermark categories:
  - robust watermarks.
  - fingerprints.
  - fragile watermarks.

- What are the differences between:
  - de-synchronization and removal attacks.
  - blind and non blind watermark extraction.

- Give an example of a 3D watermarking algorithm that operates in:
  1. Computer data (organization of the data within the file).
  2. Topological data.
  3. Geometrical data.

- Explain the trade-off between capacity, visibility and robustness in the case of 3D watermarking.

- What are the possible applications of 3D object watermarking ?

- Give some examples of manipulations and attacks on 3D objects.

1.6.2 Watermarking problem

A polyhedron can be defined by a set of vertices and a set of polygonal faces constructed by connecting these vertices. Such data can be represented as a computer file formatted as follows:

\[
\begin{align*}
N_p^p_{X_1, Y_1, Z_1} \\
\ldots \\
N_p^i_{X_i, Y_i, Z_i} \\
\ldots \\
N_p^{N_f}_{X_{N_p}, Y_{N_p}, Z_{N_p}}
\end{align*}
\]
**Figure 1.24** Example of a cube.

\[ N_{p1}, \, n_{1,1}, \, \ldots, \, n_{1,k}, \ldots, n_{1,N_{p1}} \]
\[ \ldots \]
\[ N_{pj}, \, n_{j,1}, \, \ldots, \, n_{j,k}, \ldots, n_{j,N_{pj}} \]
\[ \ldots \]
\[ N_{pN_f}, \, n_{N_f,1}, \ldots, n_{N_f,k}, \ldots, n_{N_f,N_{pN_f}} \]

- \( N_p \) is the number of vertices, and is followed by an ordered list of \( N_p \) triplets of coordinates.
- The triplet \( X_i, Y_i, Z_i, 1 \leq i \leq N_p \), represents the \( 3D \) coordinates of the \( i \)-th vertex. In our format, these coordinates are integer values.
- \( N_f \) is the number of polygonal faces of the polyhedron, and is followed by \( N_f \) lines, each describing a different face.
- A line of the form \( N_{pj}, \, n_{j,1}, \ldots, n_{j,k}, \ldots, n_{j,N_{pj}}, 1 \leq j \leq N_f \), describes the \( j \)-th polygonal face. \( N_{pj} \) is the number of vertices of the face and it is followed by an ordered list of the indices \( (n_{j,1} \text{ to } n_{j,N_{pj}}) \) of the vertices that define the boundary of this face.

As an illustration, the cube of figure 1.24 can be represented by the following file:

```
8
-500, -500, -500
```
1. We want to hide information in the file that represents a polyhedron in such a way that the shape of the polyhedron is modified as little as possible. To do so, we change the LSB (least significant bit) of the numbers $X_i$, $Y_i$, $Z_i$.

(a) In the general case, what amount of data (i.e. how many bits) can we hide with this method?

(b) In the case of the cube given as example, does the insertion of data in the LSB significantly modify its shape? (compute an order of magnitude of the ratio between the maximal displacement of a vertex and the size of the cube).

(c) To increase the amount of hidden information while preserving the shape of the object, could we modify the LSB of $N_p$, $N_f$, $N_{p_j}$, $n_{j,k}$ too?

2. Now we assume that after having watermarked an object with the previous method, the file is modified by an attacker who arbitrarily renumbers the vertices. Thus, a permutation $s$ is applied to the vertices: vertex number $i$ becomes vertex number $s(i)$. In practice, this means that the line describing vertex number $i$ is moved to the line number $s(i)$ in the list of vertices, and that $i$ is replaced by $s(i)$ whenever it appears among the $n_{j,k}$ that describe the vertices of a face. This renumbering of the vertices does not change the shape of the object.

(a) Blind extraction after renumbering attack: is it possible to recover the information hidden in the file after the vertices have been renumbered by an unknown permutation, and knowing only the watermarked and renumbered file?

(b) Non-blind extraction after renumbering attack: is it possible to recover the hidden information if, along with the watermarked and renumbered file (see previous question), we also know the original unwatermarked file?
3. Suppose that we use another watermarking scheme that encodes hidden information by modifying the order of the list of faces, an operation that does not change the shape of the described object. To do this, we first choose a reference order of the faces by sorting them, more precisely, by sorting the lists \( N_{p_j}, n_{j,1}, \ldots, n_{j,k}, \ldots, n_{j,N_{p_j}} \) \((1 \leq j \leq N_f)\) in lexicographic order. By doing so for the cube example we obtain:

4, 1, 2, 3, 4
4, 1, 2, 7, 8
4, 1, 4, 5, 8
4, 2, 3, 6, 7
4, 4, 5, 6, 3
4, 5, 6, 7, 8

Then, we number the \( N_f! \) possible permutations of this list of faces choosing an arbitrary convention, and we encode a number \( N \) in the file that represents the 3D object by ordering the list of faces using the \( N \)-th permutation of the reference order.

(a) How do we extract the hidden information (the number \( N \)) from the watermarked file? (describe only the general idea in a few lines, not the complete algorithm).

(b) In the general case, what is the capacity in bits of this watermarking algorithm?

(c) Is it possible to recover the hidden information in blind mode (i.e. knowing only the watermarked file) if the vertices have been renumbered in an unknown way by an attacker after watermarking?

(d) What is your answer to your previous question in case of non-blind detection i.e., when both the attacked watermarked file and the original (unwatermarked) file are available?

4. We use the same watermarking principle as in question 3 but this time we add a preliminary step: first we renumber the vertices so that they are ordered in lexicographic order with respect to their coordinates \( X_i, Y_i, Z_i \), and then we watermark the file by modifying the order of the faces as before.

(a) Assume that an attacker renumbers the vertices after watermarking. How is it possible to recover the hidden information in blind-mode (from the attacked watermarked file only).

(b) Is this watermarking algorithm robust to noise addition on the coordinates of the vertices?
Bibliography


Cox IJ, Miller ML and Bloom JA 2002 *Digital Watermarking.* Morgan Kaufmann.


Murotani K and Sugihara K 2004 Generalized ssa and its applications to watermarking 3d polygonal meshes.


Praun E, Hoppe H and Finkelstein A 1999 Robust mesh watermarking ACM SIGGRAPH, pp. 49 – 56, Los Angeles, California.


Wagner MG 2000 Robust watermarking of polygonal meshes *Geometric Modeling and Processing*, Hong Kong, China.


