Blind Robust Watermarking Schemes for Copyright Protection of 3D Mesh Objects

Stefanos Zafeiriou, Anastasios Tefas, *Member*, *IEEE*, and Ioannis Pitas, *Senior Member*, *IEEE*

Abstract—In this paper, two novel methods suitable for blind 3D mesh object watermarking applications are proposed. The first method is robust against 3D rotation, translation, and uniform scaling. The second one is robust against both geometric and mesh simplification attacks. A pseudorandom watermarking signal is cast in the 3D mesh object by deforming its vertices geometrically, without altering the vertex topology. Prior to watermark embedding and detection, the object is rotated and translated so that its center of mass and its principal component coincide with the origin and the *z*-axis of the Cartesian coordinate system. This geometrical transformation ensures watermark robustness to translation and rotation. Robustness to uniform scaling is achieved by restricting the vertex deformations to occur only along the r coordinate of the corresponding (r, θ, ϕ) spherical coordinate system. In the first method, a set of vertices that correspond to specific angles θ is used for watermark embedding. In the second method, the samples of the watermark sequence are embedded in a set of vertices that correspond to a range of angles in the θ domain in order to achieve robustness against mesh simplifications. Experimental results indicate the ability of the proposed method to deal with the aforementioned attacks.

Index Terms—3D mesh watermarking, blind watermarking, copyright protection.

1 Introduction

In the last decades, many new technologies became available for digital media representation, storage, and distribution. The danger of copying, tampering with, or transmitting copyrighted data without authorization (including 3D graphics models used in graphics arts, games, virtual reality, and digital terrain modeling) generated an increased demand for robust copyright protection methods. Consequently, the design of robust techniques for copyright protection and/or content authentication of multimedia data became an urgent necessity. One approach to this goal aims at generating and embedding an imperceptible signal (called a watermark) in the original data. The watermark can carry information about the data owner or an authorized user/distributor. Even though watermarking is a very active research field and its application to 2D still images and audio signals has been rather thoroughly studied, watermarking of 3D mesh objects has not been heavily researched. Digital watermarking of 3D objects remains a challenging problem. One of the reasons is the fact that there is no unique representation of 3D objects, e.g., there are 3D mesh objects, 3D objects represented using parametric surfaces such as 3D NURBS (Nonuniform Rational B-Spline surfaces), or 3D model data combined with texture information. The interested reader can refer to [1], [2], [3] for 3D NURBS graphic data watermarking and to [4] for texture-based watermarking of 3D objects.

• *The authors are with the Department of Informatics, Aristotle University of Thessaloniki, Box 451, 54124 Thessaloniki, Greece. E-mail: pitas@zeus.csd.auth.gr.*

Manuscript received 6 Oct. 2004; revised 1 Mar. 2005; accepted 2 Mar. 2005; published online 11 July 2005.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0124-1004.

In general, watermarking can be separated into two different classes according to the applications for which it was implemented:

- content authentication and tamper proofing,
- copyright protection.

In the first class, the objective is to check content authenticity or integrity and highlight any regions that have been tampered with. This goal has motivated research into fragile or semifragile watermarking technologies. The interested reader can refer to [5] for the authentication and tamper proofing of 3D mesh objects using fragile watermarking. In copyright protection applications, the embedded watermark should be perceptually invisible, statistically undetectable, and robust against various copyright attacks. This application type was more researched in the recent past [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17].

The watermarking systems can be separated into two different classes according to their detection procedure:

- blind detection watermarking systems,
- informed detection watermarking systems.

In blind detection watermarking systems, only the private key is needed for successful watermark detection. In informed detection watermarking systems, additional information concerning the original object, besides the knowledge of the private key, is needed for watermark detection. It is obvious that blind watermark detection is a major advantage due to the fact that neither original data knowledge nor time consuming search in the owners' database is needed to do watermarked object traffic monitoring over, e.g., the Internet. More details about the advantages of blind watermarking can be found in [18].

The algorithms for 3D mesh watermarking can be separated in those using embedding in the spatial domain [6], [7], [8], [9], [10], [11], [12] and those using transform embedding domain [13], [14], [15], [16], [17]. The first watermarking algorithms for 3D mesh objects were proposed in [9] and [10]. In those papers, general principles for embedding watermarks by altering the geometry or the topology of the triangles or the polygons of the 3D mesh object were introduced. The presented algorithms use informed detection and fail under remeshing attacks.

The fist watermarking technique for copyright protection that could handle mesh simplifications was proposed in [6] using informed detection. In [11], an attempt to create a 3D mesh blind watermarking system was made. The proposed watermarking scheme is a combination of three algorithms and can resist both affine transformations and mesh simplifications. However, it is not blind, due to the informed detection used in one of them in order to compensate for all affine transformations.

A watermarking system along with a method for mesh registration that needs the original 3D mesh object was proposed in [12]. This method proved to have satisfactory results against attacks such as Gaussian noise addition, surface subdivision, affine transformations, and mesh simplification.

An algorithm for watermarking 3D mesh objects using blind detection was proposed in [8]. In the first step of the algorithm, a chain of vertices and their neighborhood vertices are selected from the 3D mesh object and the vertices are ordered according to some distance metric. The vertices to be watermarked are picked from this ordering according to the watermark key. The neighborhood of the vertices to be watermarked should fulfill some criterion that employs local mesh variations in order to ensure low watermark visibility. The watermark is robust against rotation, translation, uniform scaling, and cropping. However, results against more sophisticated attacks, such as mesh simplification, were not presented in [8].

In the category of transform domain watermarking systems fall the methods proposed in [13], [14], [15], [16], [17]. In [13], the watermark is embedded using spread spectrum watermarking techniques. The algorithm is robust against mesh smoothing, random noise addition, and mesh simplification using informed detection. The multiresolution mesh decomposition [19] was used in [14] in order to embed the watermark in the wavelet domain. The watermark can resist affine transformations, partial cropping, and random noise addition to vertex coordinates. The main drawbacks of this method are the nonblind detection and the fact that the mesh must have a specific connectivity [19].

In [15], the watermark is embedded in the mesh spectral domain presented in [20]. The algorithm is robust against remeshing attacks, mesh smoothing, noise addition, and cropping using informed detection. Another robust watermarking algorithm that transforms the mesh into an image and then embeds the watermark using image-based watermarking transform domain techniques was proposed in [17]. The algorithm is robust against translation, rotation, uniform scaling, mesh simplification, and Gaussian noise addition attacks, but requires information of the original object in order to detect the watermark.

It is obvious that the vast majority of the 3D mesh object watermarking systems use informed detection. The additional information (in most cases, the original 3D mesh object itself) needed in the detection stage is used, primarily, either for object registration (in order to compensate for affine transformations or for resampling by regaining the initial connectivity [11], [12]) or for watermark extraction since it is performed using a kind of difference between the original and the watermarked object [6], [14], [15], [16] or for both [13].

In this paper, two novel blind watermarking schemes for 3D mesh objects of arbitrary topology are proposed. The first watermarking method, the so-called Principal Object Axis watermarking (POA) scheme, is robust against rotation, translation, and uniform scaling and it is an extension of the method proposed in [21]. The second method, the so-called Sectional Principal Object Axis watermarking (SPOA) scheme, which is an improvement of the first scheme, is additionally robust against mesh simplification. The low computational complexity of both watermark embedding and detection and the blind watermark detection used make them suitable for 3D model traffic monitoring applications for copyright protection.

The paper is organized as follows: In Section 2, the POA watermarking method will be described. SPOA watermarking procedure will be discussed in Section 3. The metrics used to evaluate the watermarking performance and experimental performance verification are reported in Section 4. Conclusions are drawn in Section 5.

2 3D Watermarking Using the Principal Object Axis (POA)

2.1 Preprocessing

A 3D mesh object is comprised of a set of vertices $\mathbf{V}^c$ (in Cartesian coordinates) and a set of connections between these vertices. Let $\mathbf{u}_i^c$ denote the ith vertex, $\mathbf{u}_i^c = (x_i, y_i, z_i)$. The representation of the vertex $\mathbf{u}_i^c$ in spherical coordinates is $\mathbf{u}_i^s = (r_i, \theta_i, \phi_i)$. The set of all vertices of the 3D mesh object in spherical coordinates will be denoted as $\mathbf{V}^s$. In the following, $N(\mathbf{X})$ denotes the cardinality of a set $\mathbf{X}$.

The first step before both the watermark embedding and detection procedures is a 3D mesh object transformation. Its objective is to obtain invariance against 3D translation and rotation. A description of each transform step follows.

- **Mass Center Translation**. The object is translated so that its center of mass is the center of the coordinate system axes. Let $x_i^{'}$, $y_i^{'}$, and $z_i^{'}$ be the coordinates of the translated vertex $\mathbf{u}_i^c$ and k_x, k_y, and k_z be the coordinates of the center of mass $\mathbf{k}^c$:

$$\mathbf{k}^c = \frac{1}{N(\mathbf{V}^c)} \sum_{i=0}^{N(\mathbf{V}^c)} \mathbf{u}_i^c. \tag{1}$$

- **Principal axis alignment**. The 3D mesh object is rotated so that its principal component axis of its vertices coincides with the z axis. This axis is the

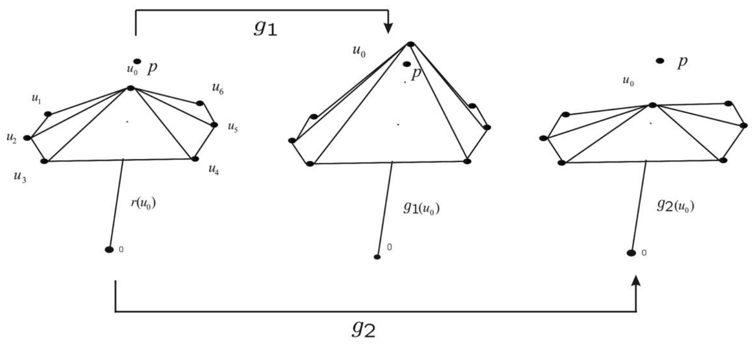


Fig. 1. The original vertex $\mathbf{u}_0$ and its neighboring vertices. A watermark sample is embedded in vertex $\mathbf{u}_0$.

eigenvector that corresponds to the greatest eigenvalue of the covariance matrix $\mathbf{C}$ of the vertex coordinates [21]. Thus, robustness against rotation of the watermarked 3D mesh object is achieved.

- **Conversion to Spherical Coordinates**. The 3D mesh object is converted to spherical coordinates in order to achieve robustness against scaling:

$$\begin{aligned} r_i &= r(\mathbf{u}_i^s) = \sqrt{x_i''^2 + y_i''^2 + z_i''^2}, \\ \theta_i &= \theta(\mathbf{u}_i^s) = arccos\left(\frac{z_i''}{r_i}\right), \\ \phi_i &= \phi(\mathbf{u}_i^s) = arctan\left(\frac{y_i''}{x_i''}\right), \end{aligned} \qquad (2)$$

where x_i'', y_i'', z_i'' are the vertex coordinates after the model rotation, $r_i \in [0, \infty)$, $\phi_i \in [0, 2\pi)$, and $\theta_i \in [0, \pi]$. The domain $\mathbf{\Theta}$ of θ angles is defined as $\mathbf{\Theta} = \{\theta_j : \exists \mathbf{u}_i^s \in \mathbf{V}^s, \theta(\mathbf{u}_j^s) = \theta_j\}$.

2.2 Watermark Generation

The watermark generation procedure aims at assigning every vertex $\mathbf{u}_i^s \in \mathbf{V}^s$ of the 3D mesh object with a label $l(\mathbf{u}_i^s) \in \{-1, 0, 1, 2\}$ using two pseudorandom number generators. The pseudorandom number generators are used for creating a sequence of N_w numbers $\theta_i^w \in [0, \pi]$, $i = 1, \ldots, N_w$ and a watermark sequence $w_i, \in \{-1, 1\}$, $i = 1, \ldots, N_w$ of length N_w, based on the owner's private key. The sequence of θ_i^w is used for locating the vertices that the watermark samples will be embedded into. The sequence w_i indicates how the watermark will be embedded in these vertices (how the vertex will be labeled). Let $\mathbf{\Theta}^w = \{\theta_i^w\}$ be the set of all θ_i^w belonging to a certain watermark sequence.

2.3 Watermark Embedding

Here, it is assumed that the 3D mesh object has enough vertices for watermark embedding. In order to embed the watermark sample w_i (assign a label l), the vertex $\mathbf{u}_i^s$ whose angle θ_i is closest to θ_i^w is found. Then, the watermark embedding is performed by altering the r component of the vertex $\mathbf{u}_i^s \in \mathbf{V}^s$ according to:

$$r^w(\mathbf{u}_i^s) = \begin{cases} r(\mathbf{u}_i^s) & \text{if } l(\mathbf{u}_i^s) = 0, 2 \\ g_1(\mathbf{u}_i^s) & \text{if } l(\mathbf{u}_i^s) = 1 \\ g_2(\mathbf{u}_i^s) & \text{if } l(\mathbf{u}_i^s) = -1, \end{cases} \qquad (3)$$

where the vertex label $l(\mathbf{u}_i^s)$ comes from the corresponding watermark sample and by assigning a label $l(\mathbf{u}_i^s) = w_i$ for each watermarked vertex. Originally, all vertices $\mathbf{u}_i^s$ are labeled $l(\mathbf{u}_i^s) = 0,\ i = 1, .., N(\mathbf{V}^s)$.

The embedding functions, g_1, g_2, and the appropriate detection function can be designed giving different watermarking schemes. The functions that are used in this method are based on the values of the neighboring surface vertices of the vertex to be modified and are given by:

$$g_1(\mathbf{u}_i^s) = f(a_1)H(\mathbf{u}_i^s), \qquad (4)$$

$$g_2(\mathbf{u}_i^s) = f(a_2)H(\mathbf{u}_i^s), \qquad (5)$$

where a_1, a_2 are suitably chosen constants and $H(\mathbf{u}_i^s)$ is a local neighborhood operation of the vertices around $\mathbf{u}_i^s$, and $f(a)$ is a polynomial of a. A discussion about the function H is provided in Section 2.6. Moreover, the values of a_1, a_2 are chosen so that $a_1 > 0$ and $a_2 < 0$. Different values of a_1 and a_2 are used in order to provide a kind of visual masking where different thresholds are applied to a_1 and a_2 according to the neighborhood region. That is, different values of a_1 and a_2 should be considered for curved and hollow regions in order to prevent the generation of visual artifacts after embedding.

Except for using proper thresholds for a_1 and a_2, other masking procedures can be applied as well [8]. That is, the curvature of a specific neighborhood can be measured using local variance of the neighboring vertices. Then, the vertices in these areas are avoided during the embedding procedure.

The signs of a_1 and a_2 are used for the detection function and their values determine the watermark power. A watermark sample added in the vertex $\mathbf{u}_0$ using (4) and (5) is shown in Fig. 1. The operator H is used in order to estimate the point $\mathbf{p}$. The point $\mathbf{p}$ has the same θ and ϕ components as the vertex $\mathbf{u}_0$. Then, the vertex $\mathbf{u}_0$ is moved in the direction of ray cast from $(0, 0, 0)$ to $\mathbf{u}_0$ above or below the point $\mathbf{p}$. The pseudocode of the POA watermarking algorithm can be found in Appendix A.

All vertices that consist of the neighborhood around a vertex which is watermarked using g_1 or g_2 are assigned the label 2. Thus, they are not altered by (3).

The watermark embedding procedure is an iterative procedure that finishes after N_w steps or after all vertices of the 3D mesh object have been used in this procedure (i.e., they have been assigned with a label -1, 1, and 2). If a vertex selected for watermarking belongs in the neighborhood of a previously marked vertex, the θ sequence is advanced and another vertex is selected.

2.4 Robustness to Uniform Scaling

In order for the watermark procedure to be scale invariant, the H operator in (4) and (5) should possess the property:

$$H(\mathbf{u}_i^s) = \gamma H(\mathbf{v}_i^s), \tag{6}$$

where $\mathbf{u}_i^c = \gamma \mathbf{v}_i^c$ in the corresponding Cartesian coordinates and γ is a scalar that corresponds to the scaling factor $\gamma > 0$. Thus, for a scaled version of the 3D mesh object, it is valid that

$$\operatorname{sign}(r(\mathbf{u}_i^s) - H(\mathbf{u}_i^s)) = \operatorname{sign}(r(\mathbf{v}_i^s) - H(\mathbf{v}_i^s)). \tag{7}$$

2.5 Watermark Detection

In the watermark detection procedure, the 3D mesh object under investigation is transformed according to the transformation presented in Section 2.1. In order to cope with object transposition in the principal object axis, the detection is held twice, one for each transposition. After the geometric transformations, the watermark sequence and the angles θ_i^w are generated, using the owner's key, in order to label each vertex of the 3D mesh object $\mathbf{u}_i^s$ with a label $l(\mathbf{u}_i^s) \in \{-1, 0, 1, 2\}$, as described in Section 2.3. Let the set $\mathbf{L}_w = \{\mathbf{u}_j^s \in \mathbf{V}_w^s : l(\mathbf{u}_j^s) \in \{-1, 1\}\}$. The resulting detection function using (3), (4), and (5) for every $\mathbf{u}_i^s \in \mathbf{L}_w$ is:

$$d(\mathbf{u}_i^s) \triangleq \begin{cases} 1 & \text{if } r(\mathbf{u}_i^s) - H(\mathbf{u}_i^s) > 0 \\ -1 & \text{if } r(\mathbf{u}_i^s) - H(\mathbf{u}_i^s) < 0. \end{cases} \tag{8}$$

Based on the watermark sequence and the detection signal d, it is decided whether the watermark under investigation is embedded in the 3D mesh object or not. The detection is based on the value by value comparison of the $d(\mathbf{u}_i^s)$ with $l(\mathbf{u}_i^s) \in \{-1, 1\}$:

$$e_w(\mathbf{u}_i^s) = \begin{cases} 1 & \text{if } l(\mathbf{u}_i^s) \neq d(\mathbf{u}_i^s) \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

The false detection signal is equal to 1 if a watermarked vertex is falsely detected and 0 otherwise. The detection ratio is defined as the ratio of the correctly detected vertices to the sum of the watermarked vertices in the 3D mesh object:

$$D_w \triangleq \frac{1}{N(\mathbf{L}_w)} \sum_{\mathbf{u}_i^s \in \mathbf{L}_w} (1 - e_w(\mathbf{u}_i^s)). \tag{10}$$

The embedding functions are designed in such a way that the probability p of a vertex being detected as signed with g_1 or g_2, for an unwatermarked 3D mesh object, is 0.5. The watermark decision is taken by comparing D_w with a predefined threshold T. The threshold value determines the minimum acceptable level of watermark detection.

2.6 The Neighborhood Operator

The neighborhood operator H used in (4) and (5) plays a very important role in the watermarking procedure. In the POA watermarking procedure, H was used for locating the watermarked vertices and in the SPOA (which will be discussed in the next section) method for forming the random variable d_r (16). Here, some implementations of this operator are shown and their advantages and disadvantages are discussed.

A first operator H that could be used is the arithmetic mean of the r component:

$$H(\mathbf{u}_i^s) = \frac{1}{n} \sum_{j=1}^{n} r(\mathbf{v}_j^s), \tag{11}$$

where $\{\mathbf{v}_j^s\}$ is a local neighborhood of $\mathbf{u}_i^s$ and $n = N(\{\mathbf{v}_j^s\})$. The original vertex $\mathbf{u}_i^s$ does not belong to the neighborhood $\{\mathbf{v}_j^s\}$.

Another simple operator H is the median of the neighborhood $\{\mathbf{v}_j^s\}$. The local neighborhood used can be defined using vertex connectivity information or some distance metric. If connectivity information is used, then the neighborhood can be found very quickly; otherwise, extra computational time is required (e.g., for finding the k nearest vertices of vertex $\mathbf{u}_i^s$ using Euclidean or other distance metrics). In the case where connectivity information is not taken into consideration for defining the vertex neighborhood, the resulting watermarking method is more robust against connectivity attacks and is suitable for watermarking 3D point clouds (connectivity information is no longer necessary). A family of more sophisticated operators H can be constructed by building a parametric surface using the neighborhood vertices as control points. Such kinds of surfaces are the tensor product Bezier surfaces [22]. The $H(\mathbf{u}_i^s)$ of a vertex $\mathbf{u}_i^s$ can be calculated from the intersection of the ray (line) that is cast from $O = (0, 0, 0)$ to the vertex $\mathbf{u}_i^c$ and the parametric surface $\mathbf{B}(s, t)$, defined as:

$$\mathbf{B}(s,t) = \sum_{i=1}^{m} \sum_{j=1}^{n} b_{i,m}(s) b_{j,n}(t) \mathbf{v}_{ij}^c, \tag{12}$$

where $\{\mathbf{v}_{ij}^c\}$ is the neighborhood of the vertex $\mathbf{u}_i^c$, $b_{i,m}(s)$ and $b_{j,n}(t)$ are two Bernstein Polynomials given by:

$$b_{l,k}(\tau) = \frac{k!}{(k-l)!l!} \tau^k (1-\tau)^{k-l}, \; 0 < \tau < 1. \tag{13}$$

Let $\mathbf{p}^c$ be the point where $\mathbf{B}(s,t)$ and the ray cast from $(0, 0, 0)$ to the vertex $\mathbf{u}_i^c$ intersect, then $\theta(\mathbf{u}_i^s) = \theta(\mathbf{p}^s)$, $\phi(\mathbf{u}_i^s) = \phi(\mathbf{p}^s)$, where $\mathbf{p}^s$ and $\mathbf{u}_i^s$ are the corresponding vertices of $\mathbf{p}^c$ and $\mathbf{u}_i^c$ in spherical coordinates. H is chosen to be:

$$H(\mathbf{u}_i^s) = r(\mathbf{p}^s). \tag{14}$$

The point $\mathbf{p}^c$ can be found using a very efficient method called Bezier clipping [23]. In the case where the ray intersects the patch in two points $\mathbf{p}_1^s$ and $\mathbf{p}_2^s$, then $\mathbf{p}^s$ is the closest point to $\mathbf{u}_k^s$. Another parametric family of surfaces that could be used for forming the operator H is the NURBS family.

It can be easily proven that the neighborhood operators, H, described in (11) and (14) possess the property given in (6) in order to produce scale invariant watermarks. In order to build the control points for the Bezier surface, the neighborhood vertices is one way to project the vertices in x, y plane (make the z coordinate 0). The vertices are sorted in the ascending order of x. After they are separated in n sets, each set contains m elements. Each of the n sets is successively sorted in ascending order with respect to y coordinates. In order to construct the initial neighborhood, the connectivity can be used and then the points inside the Bezier tensor product surface can be ordered. Fig. 2 shows how the operator H works for a tensor product surface of

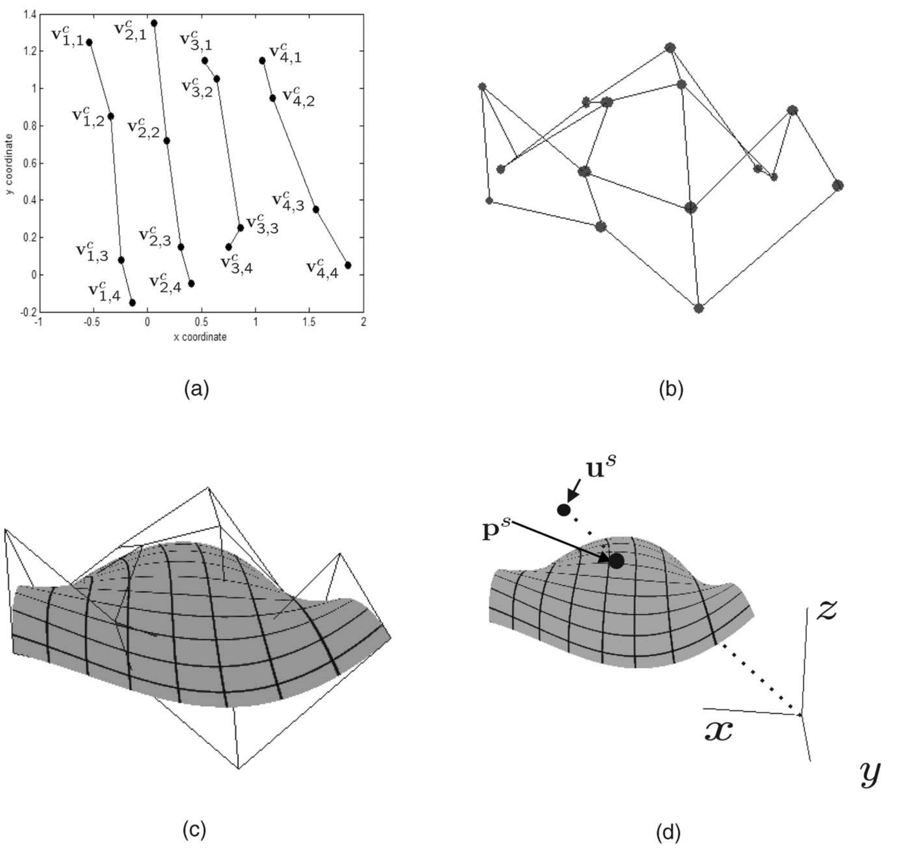


Fig. 2. (a) The neighborhood vertices are projected to (x, y) and ordered. (b) The patch in $3D$. (c) The patch along with the Bezier surface. (d) The ray cast from $(0, 0, 0)$ to the vertex $\mathbf{u}^s$ intersects the Bezier surface.

4×4 control points. The vertices that comprise the neighborhood of $\mathbf{u}^s$ correspond to connectivity down to depth 3.

The vertex prediction operator H can also be built using quadratic surfaces. Quadratic surfaces have the advantage that they do not need some ordering as Bezier surfaces. Another advantage is that they do not need some special method in order to find the intersection point with the ray cast from $(0, 0, 0)$ to the vertex $\mathbf{u}^s$. The predicted vertex can be easily found by just solving a quadratic equation. For every point $\mathbf{w}^c_i = (x_i, y_i, z_i)$ that belongs to the surface, it is valid that:

$$\begin{aligned} &\alpha x_i^2 + \beta y_i^2 + \gamma z_i^2 + 2\delta y_i x_i + 2\epsilon z_i x_i + 2\zeta x_i y_i \\ &+ 2\mu x_i + 2\nu y_i + 2\eta z_i + \rho = 0. \end{aligned} \tag{15}$$

Using the neighborhood vertices, the set of the parameters $[\alpha, \beta, \gamma, \delta, \epsilon, \zeta, \mu, \nu, \eta, \rho]$ of (15) can be calculated. The point $\mathbf{q}^s$ that is the intersection of the ray cast from $(0, 0, 0)$ to the vertex $\mathbf{u}^c_i$ can be calculated by (15), using the fact that $\theta(\mathbf{u}^s_i) = \theta(\mathbf{q}^s)$, $\phi(\mathbf{u}^s_i) = \phi(\mathbf{q}^s)$.

3 Sectional Principal Object Axis (SPOA) Watermarking

Mesh simplification routines reduce the mesh size for faster processing and rendering, while, at the same time, maintain the perceived object shape and its visual quality. The interested reader can refer to [24], [25], [26] for efficient mesh simplification algorithms. Examples of the object Stanford Bunny [27], with 34,834 vertices and 69,451 triangles, simplified using the algorithm in [24], are depicted in Fig. 3.

Mesh simplification attacks frequently erase some vertices that are used in the embedding procedure and/or may alter the principal object axis as well. Such alterations can cause watermark synchronization loss and, thus, false watermark rejection. The previously described watermarking scheme has been proven to be sensitive against mesh simplifications. This is due to the strong dependency of the principal component axis orientation with the object vertices and to the fact that a watermark sample is embedded in only one vertex. Thus, in many cases, blind detection will fail. In a nonblind fashion, mesh simplification and cropping will be handled if the center of the mass of the original 3D mesh object and the principal object axis were available at the watermark detection stage. However, in this case, the watermark detection becomes informed, which is a serious drawback of a watermarking procedure.

In order to achieve robustness against mesh simplification, a set of vertices that correspond to a range of θ angles $\mathbf{\Theta}_j \subset \mathbf{\Theta}$ is selected and the r components of these vertices are used as watermark embedding primitive. Let the set $\mathbf{I}(\mathbf{\Theta}_j) = \{\mathbf{u}^s_i : \mathbf{u}^s_i \in \mathbf{V}^s, \theta(\mathbf{u}^s_i) \in \mathbf{\Theta}_j\}$. For each vertex in $\mathbf{I}(\mathbf{\Theta}_j)$, the difference $d_r(\mathbf{u}^s_i)$ is formed :

$$d_r(\mathbf{u}^s_i) = r(\mathbf{u}^s_i) - H(\mathbf{u}^s_i), \tag{16}$$

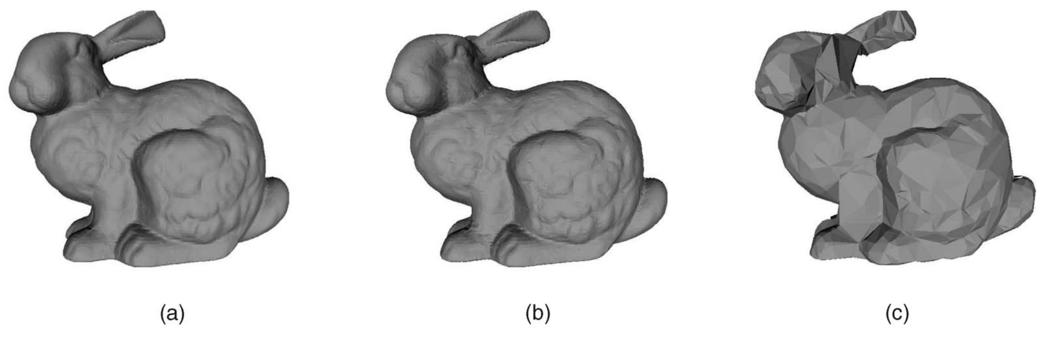


Fig. 3. (a) Stanford Bunny object. (b) After 40 percent vertex. (c) After 90 percent vertex decimation.

where H is a local neighborhood operation of the vertices, described in Section 2.6, around $\mathbf{u}_i^s$ and the $H(\mathbf{u}_i^s)$ is considered an approximation function of the $r(\mathbf{u}_i^s)$ that depends on the neighborhood of $\mathbf{u}_i^s$.

The operator H is chosen under the assumption that $d_r(\mathbf{u}_i^s)$ follows a Gaussian distribution with variance σ^2 and zero mean. The verification of the statement that d_r follows a Gaussian distribution has been done only experimentally with the use of the Kolmogoroff-Smirnoff test [28]. Since d_r is assumed to follow a Gaussian distribution with zero mean and variance σ^2, the so-called left and right variance estimators are defined as follows:

$$\hat{\sigma}_l^2 = \frac{1}{N(\{d_r : d_r < 0\}) - 1} \sum_{d_r<0} d_r^2, \tag{17}$$

$$\hat{\sigma}_r^2 = \frac{1}{N(\{d_r : d_r > 0\}) - 1} \sum_{d_r>0} d_r^2, \tag{18}$$

and are sufficient for estimating σ^2:

$$\hat{\sigma}^2 \approx \hat{\sigma}_l^2 \approx \hat{\sigma}_r^2. \tag{19}$$

The idea behind the SPOA watermarking method is to use the symmetry (19) of the distribution of d_r and alter only one side of the distribution. In other words, the embedding procedure affects only one of the two variance estimators, (17), (20), and the other one is used in the watermark detection procedure.

3.1 Watermark Generation

In this scheme, the watermark generation aims at separating the interval $[0, \pi]$ in L intervals $\mathbf{\Theta}_j$, $j = 1, .., L$. At each interval $\mathbf{\Theta}_j$, a label $w_j = l(\mathbf{\Theta}_j) \in \{-1, 0, 1\}$ is assigned indicating how this interval will be altered by the embedding procedure. The value, $l(\mathbf{\Theta}_j)$, for each interval is determined by the owners' digital key using a pseudorandom number generator. The intervals $\mathbf{\Theta}_j$ for which $l(\mathbf{\Theta}_j) \in \{-1, 1\}$ have fixed length of t rad. The length t is determined by the tolerance that the algorithm should have in case of principal object axis alterations.

3.2 Watermark Embedding

The watermark embedding procedure is an iterative procedure applied to each interval $\mathbf{\Theta}_j, j = 1, \ldots, L$ and ends when the entire interval $[0, \pi]$ is covered. In the first step, a number $\theta(1) \in [0, \pi]$ is picked, using a pseudorandom number generator fed with the owner's private key. Afterward, at each step m of the procedure, two uniform distributed pseudorandom number generators are used for producing a number $w_m \in \{-1, 1\}$ and an angle $\theta_1(m) \in (0, \epsilon)$. The value w_m is used for labeling the set $\mathbf{I}([\theta(m), \theta(m) + t))$, while the set $\mathbf{I}([\theta(m) + t, \theta(m) + t + \theta_1(m)))$ remains unaltered (labeled with 0). The watermark embedding in these sets is described subsequently. Then, $\theta(m + 1)$ is set equal to $\theta(m) + t + \theta_1(m)$. The algorithm continues in the same way until the interval $[0, \pi]$ is covered.

The parameter ϵ is a constant that controls the length of the intervals that will remain unaltered during the embedding procedure. These intervals help the procedure to be owner's key-dependent.

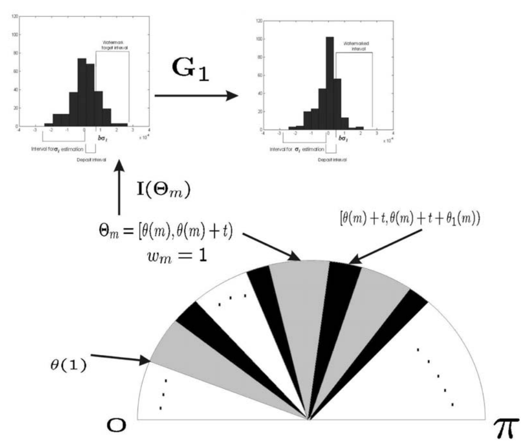
The watermark is embedded in the 3D mesh object after the application of the transforms described in Section 2.1 by altering the r component of the vertices of $\mathbf{I}(\mathbf{\Theta}_j)$ according to:

$$\mathbf{R}^w(\mathbf{I}^w(\mathbf{\Theta}_j)) = \begin{cases} \mathbf{R}(\mathbf{I}(\mathbf{\Theta}_j)) & \text{if } l(\mathbf{\Theta}_j) = 0 \\ \mathbf{G}_1(\mathbf{I}(\mathbf{\Theta}_j)) & \text{if } l(\mathbf{\Theta}_j) = 1 \\ \mathbf{G}_2(\mathbf{I}(\mathbf{\Theta}_j)) & \text{if } l(\mathbf{\Theta}_j) = -1, \end{cases} \tag{20}$$

where $\mathbf{R}$ denotes the vector of the r components of a set of vertices $\mathbf{I}$.

The embedding functions $\mathbf{G}_1$ and $\mathbf{G}_2$ cast a watermark sample w_j by assigning $l(\mathbf{\Theta}_j) = w_j$ when applied to a set $\mathbf{I}(\mathbf{\Theta}_j)$ by changing the distribution of the random variable d_r (16) of the vertices contained in $\mathbf{I}(\mathbf{\Theta}_j)$. $\mathbf{G}_1$ changes the distribution of d_r by inducing deformations in the r component of the vertices of a set $\mathbf{I}(\mathbf{\Theta}_j)$ without altering $\hat{\sigma}_l^2$. The application of $\mathbf{G}_1$ alters the r component of some of the vertices $\mathbf{u}_i^s \in \mathbf{I}(\mathbf{\Theta}_j)$ that have $d_r(\mathbf{u}_i^s) > b\hat{\sigma}_l$ so that it falls inside the interval $(0, b\hat{\sigma}_l)$. Constant b controls the watermark perceptibility.

In the same manner, $\mathbf{G}_2$ deforms the distribution of d_r by altering the r component of some of the vertices $\mathbf{u}_i^s \in \mathbf{I}(\mathbf{\Theta}_j)$ that have $d_r(\mathbf{u}_i^s) < -b\hat{\sigma}_r$ so that $d_r(\mathbf{u}_i^s)$ falls inside the interval $(-b\hat{\sigma}_r, 0)$ without altering the parameters $\hat{\sigma}_r^2$ of d_r. That is, watermark embedding is performed by altering only some of the vertices $\mathbf{u}_i^s$ with $d_r(\mathbf{u}_i^s) > b\hat{\sigma}_l$, when embedding the watermark sample $w_j = 1$, $(l(\mathbf{\Theta}_j) = 1)$, whereas the remaining vertices $\mathbf{u}_i^s$ for which $d_r(\mathbf{u}_i^s) < 0$ remain unaltered since they are used in the detection procedure. If $l(\mathbf{\Theta}_j) = -1$, the vertices with $d_r(\mathbf{u}_i^s) > 0$ remain unaltered and those with $d_r(\mathbf{u}_i^s) < -b\hat{\sigma}_r$ are used for watermark embedding.



$\mathbf{G_1}$
$\mathbf{I}(\Theta_m)$
$\Theta_m = [\theta(m), \theta(m)+t)$
$w_m = 1$
$[\theta(m)+t, \theta(m)+t+\theta_1(m))$
$\theta(1)$
O
π

Fig. 4. The region $[0, \pi]$ is separated into black and gray intervals. The watermark samples are embedded in the gray intervals.

Fig. 5 shows how the watermark value $w_j = 1$ can be embedded in a set $\mathbf{I}(\mathbf{\Theta}_j)$. As can been seen in this figure, the distribution domain is separated in three intervals in order to embed $w_j = 1$. The first interval ($d_r < 0$) is used for evaluating $\hat{\sigma}_l^2$ and the second one ($0 < d_r < b\hat{\sigma}_l$) is modified by depositing the vertices from the watermark target interval ($d_r > b\hat{\sigma}_l$). Fig. 6 shows the local distribution modifications for embedding watermark value $w_j = -1$. One interval is used for evaluating $\hat{\sigma}_r^2$ ($d_r > 0$) and the second one ($-b\hat{\sigma}_r < d_r < 0$) is modified by depositing vertices from the watermark target interval ($dr < -b\hat{\sigma}_r$) to the deposit interval ($-b\hat{\sigma}_r, 0$). The functions $\mathbf{G}_1$ and $\mathbf{G}_2$ can be just summing rules. That is, they can just add a constant number to the r component of the vertices to be altered. The proposed algorithm is described in its general form and various functions for $\mathbf{G}_1$ and $\mathbf{G}_2$ could be used (e.g., multiplication rules).

Masking procedures can be also applied in order to prevent the creation of visual artifacts as described in the previous section. The embedding algorithm is described pictorially in Fig. 4. The gray intervals are the ones in which the watermark is to be embedded. The black intervals remain unaltered. The watermark sample 1 will be embedded in the interval $\mathbf{\Theta}_m = [\theta(m), \theta(m) + t)$. The pseudocode of the embedding algorithm is given in Appendix B.

For an unwatermarked 3D mesh object and for a set of vertices $\mathbf{I}(\mathbf{\Theta}_j)$ of this object, it is valid, under the assumption that d_r follows a Gaussian distribution, that:

$$\hat{Prob}(d_r > b\hat{\sigma}_l) = \frac{N(\mathbf{I}_r(\mathbf{\Theta}_j))}{N(\mathbf{I}(\mathbf{\Theta}_j))} \approx G(-b) \tag{21}$$

and

$$\hat{Prob}(d_r < -b\hat{\sigma}_r) = \frac{N(\mathbf{I}_l(\mathbf{\Theta}_j))}{N(\mathbf{I}(\mathbf{\Theta}_j))} \approx G(-b), \tag{22}$$

where $\mathbf{I}_r(\mathbf{\Theta}_j) = \{\mathbf{u}_i^s \in \mathbf{I}(\mathbf{\Theta}_j) : d_r(\mathbf{u}_i^s) > b\hat{\sigma}_l\}$, $\mathbf{I}_l(\mathbf{\Theta}_j) = \{\mathbf{u}_i^s \in \mathbf{I}(\mathbf{\Theta}_j) : d_r(\mathbf{u}_i^s) < -b\hat{\sigma}_r\}$ and $\hat{Prob}(X)$ is the probability estimate of the hypothesis X. The function G is given by:

$$G(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-y^2}{2}} dy. \tag{23}$$

Let $\mathbf{I}_r^w(\mathbf{\Theta}_j)$ and $\mathbf{I}_l^w(\mathbf{\Theta}_j)$ be the vertex sets $\mathbf{I}_r(\mathbf{\Theta}_j)$ and $\mathbf{I}_l(\mathbf{\Theta}_j)$ after watermarking. The following inequality holds for the probability estimates and the set $\mathbf{I}^w(\mathbf{\Theta}_j)$ that has been produced by $\mathbf{G}_1$ on the watermarked 3D mesh object:

$$\hat{Prob}^w(d_r > b\hat{\sigma}_l) = \frac{N(\mathbf{I}_r^w(\mathbf{\Theta}_j))}{N(\mathbf{I}^w(\mathbf{\Theta}_j))} < G(-b). \tag{24}$$

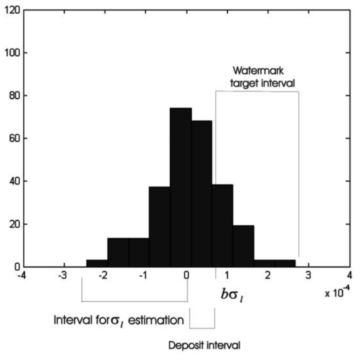
Similarly, if $\mathbf{I}^w(\mathbf{\Theta}_j)$ was created by $\mathbf{G}_2$, the corresponding inequality is valid:

$$\hat{Prob}^w(d_r < -b\hat{\sigma}_l) = \frac{N(\mathbf{I}_l^w(\mathbf{\Theta}_j))}{N(\mathbf{I}^w(\mathbf{\Theta}_j))} < G(-b). \tag{25}$$

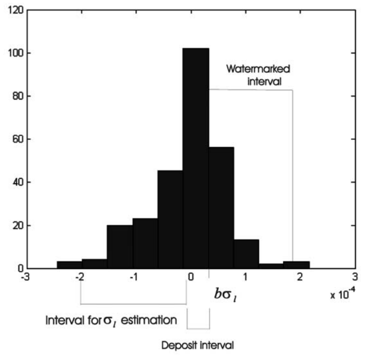
Equations (21), (22), (24), and (25) are used for calculating the detection ratio, which is used for deciding whether a 3D mesh object is watermarked or not.

3.3 Watermark Detection

In order to cope with object transposition in the principal object axis, the detection is held twice, one for each transposition. Prior to watermark detection, the 3D mesh object under investigation is geometrically transformed, as described in Section 2.1. Afterward, the watermark sequence is generated according to the owner's digital key, forming the intervals $\mathbf{\Theta}_j$ and the labels $w_j = l(\mathbf{\Theta}_j)$. For



(a)



(b)

Fig. 5. (a) Original distribution of d_r in a set $I(\Theta_i)$. (b) Distribution of d_r after the application of $\mathbf{G}_1$.

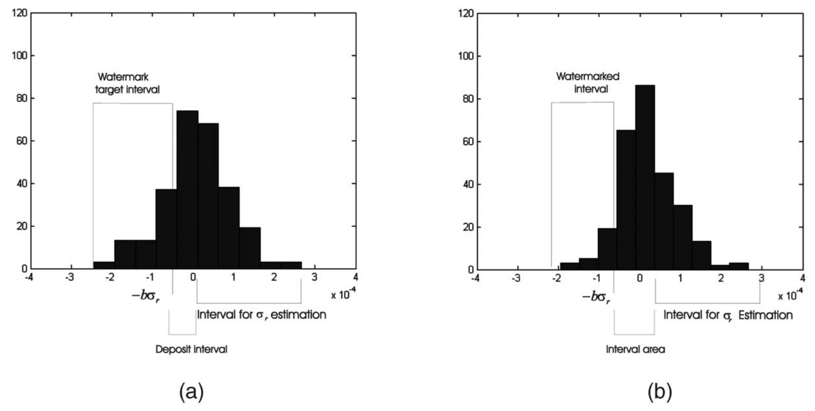


Fig. 6. (a) Original distribution of d_r in a set $I(\Theta_i)$. (b) Distribution of d_r after the application of $\mathbf{G}_1$.

the sets $\mathbf{I}^w(\Theta_j)$ with $l(\Theta_j) \in \{-1, 1\}$, the detection ratio is formed as:

$$d(\Theta_j) \triangleq \begin{cases} \frac{N(\mathbf{I}_r^w(\Theta_j))}{N(\mathbf{I}^w(\Theta_j))} & \text{if } l(\Theta_j) = 1 \\ \frac{N(\mathbf{I}_l^w(\Theta_j))}{N(\mathbf{I}^w(\Theta_j))} & \text{if } l(\Theta_j) = -1. \end{cases} \tag{26}$$

The average detection ratio:

$$D_w \triangleq \frac{1}{N(\mathbf{M})} \sum_{j \in \mathbf{M}} d(\Theta_j) \tag{27}$$

($\mathbf{M} = \{k : l(\Theta_k) \in \{-1, 1\}\}$) is used for watermark detection. The decision about the ownership of the 3D mesh object is taken by comparing the watermark detection ratio given by (27) to a predefined threshold T. For an unwatermarked 3D mesh object:

$$D_w \approx G(-b), \tag{28}$$

whereas, for a watermarked 3D mesh object:

$$D_w < G(-b). \tag{29}$$

The detection ratio D_w is invariant to uniform scaling attack due to the property described in (6). A proof of this statement can be found in Appendix C.

4 Performance Evaluation

To evaluate the performance of the proposed algorithm, the ROC (Receiver Operating Characteristic) curves have been derived and the SNR has been used in order to measure watermark perceptibility. A more detailed description follows.

To measure the SNR of a watermarked 3D mesh object, the following formula is used:

$$SNR = 10 \log_{10} \left(\frac{\sum_{i=0}^{N-1} (x_i^2 + y_i^2 + z_i^2)}{\sum_{i=0}^{N-1} ((\tilde{x}_i - x_i)^2 + (\tilde{y}_i - y_i)^2 + (\tilde{z}_i - z_i)^2)} \right), \tag{30}$$

where x_i, y_i, z_i and $\tilde{x}_i$, $\tilde{y}_i$, $\tilde{z}_i$ are the coordinates of vertex $\mathbf{u}_i^c$ before and after the watermark embedding, respectively.

The decision on whether a 3D mesh object is watermarked is taken by comparing the detection ratio D_w to a threshold T. For a given threshold, the performance of the system can be expressed as a function of the false alarm probability $P_{fa}(T)$ (i.e., the probability of detecting a watermark in a nonwatermarked object or in an object that is watermarked with another key) and the false rejection probability $P_{fr}(T)$ (i.e., the probability of not detecting a watermark in a watermarked object using the correct key):

$$P_{fa}(T) = Prob(D_w > T | H_0) = \int_T^{\infty} f_{D_w|H_0}(t) dt, \tag{31}$$

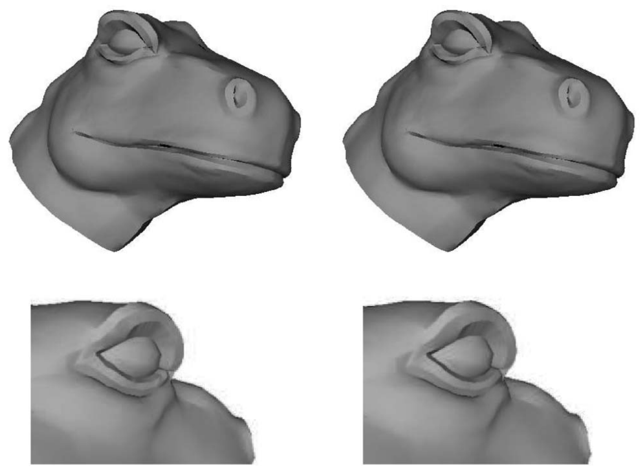
$$P_{fr}(T) = Prob(D_w < T | H_1) = \int_{-\infty}^{T} f_{D_w|H_1}(t) dt, \tag{32}$$

where H_1 is the hypothesis that the watermark exists in the object and H_0 is the hypothesis that the watermark under investigation does not exist in the object and $f_{D_w|H_1}(t)$ and $f_{D_w|H_0}(t)$ are the probability distribution functions of the variable D_w given by (10) or (27). Ideally, P_{fa} and P_{fr} should be zero.

The ROC is the curve defined by $P_{fa}(T)$, $P_{fr}(T)$ for various T. The operating point where $P_{fa} = P_{fr}$ is called equal error rate (EER) and can be used as a quantitative estimation of the watermark detection performance. If a Gaussian distribution is assumed for both $f_{D_w|H_0}$ and $f_{D_w|H_1}$, having means $\mu_{D_w|H_0}$, $\mu_{D_w|H_1}$ and variances $\sigma^2_{D_w|H_0}$, $\sigma^2_{D_w|H_1}$, the following formula can used to evaluate the ROC curve:

$$P_{fa} = \frac{1}{2} \left[1 - erf \left(\frac{\sqrt{2}\sigma_{D_w|H_0} erf^{-1}(2P_{fr} - 1) + \mu_{D_w|H_0} - \mu_{D_w|H_1}}{\sqrt{2}\sigma_{D_w|H_1}} \right) \right]. \tag{33}$$

A set of experiments using several 3D mesh objects has been conducted to illustrate the robustness of the proposed techniques against several geometric attacks and 3D mesh simplification. A panel of viewers has also been used for verifying the visual imperceptibility of the watermark. The geometrical attacks that were tested are 3D translation, rotation, and uniform scaling. Due to the invariance properties of the transform that is applied to the 3D mesh object prior to watermark embedding and detection, the results for these attacks were identical to the ones obtained



(a) (b)

Fig. 7. (a) Dino model. (b) Dino model with 600 watermarked vertices.

when no attack was performed and, thus, they will not be presented separately.

For the POA watermarking algorithm, described in Section 2, the watermark embedding power is related to the constants a_1 and a_2. In practice, the values of a_1 and a_2 are iteratively increased until a specified SNR value is achieved. The algorithm was tested for many watermark lengths and it was found that the use of at least 300 watermarked vertices gives good results even for small 3D mesh objects. Of course, the performance is improved if the watermark length is increased.

The experiments were realized on a Pentium IV 3.4 GHz processor machine, where the execution of either the embedding or the detection procedure lasted between 0.05 and 6 seconds, depending on the watermark length and the size of the 3D mesh object. Thus, the watermarking method is very fast. The 3D mesh object used for demonstrating the performance of the first watermarking scheme is the "Dino" 3D mesh object with 5,497 vertices and 10,778 triangles. The original "Dino" 3D mesh object and the watermarked object are depicted in Fig. 7. There are no visible differences between the two objects. The ROC curves for this object for varying watermark vertex number are depicted in Fig. 8a.

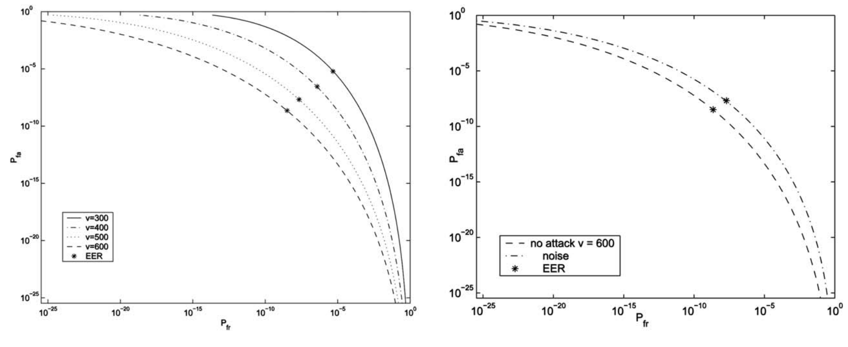
TABLE 1
Watermark Detection Results for the 3D Mesh Objects Dino

Object Used	Watermarked Vertices	EER	SNR (dB)
Dino	300	5.1×10^{-6}	116.4
	400	4.2×10^{-7}	113.28
	500	3.1×10^{-8}	110.6
	600	2×10^{-9}	109.2

Detection has been performed using 1,000 correct and 1,000 wrong keys. The EER and SNR values for the 3D mesh object Dino for various values of the embedding power are summarized in Table 1. The EER is very small, thus guaranteeing excellent detection performance. The large SNR values ensure the imperceptibility of the watermark. An attack considered for the POA algorithm is the noise addition with SNR equal to the one of the watermark. The ROC curves for this attack are depicted in Fig. 8b. However, the performance of the algorithm reduces as the noise level is increased. Such an attack may cause severe alterations to the shape of the 3D mesh object.

The watermark detection capability of SPOA algorithm, described in Section 3, and its robustness against mesh simplification have been verified in several experiments using mesh objects much larger than the ones used for the first algorithm. The models used for the experiments are comprised of 25,000 to 500,000 vertices. The experiments were realized on the same computer and both the embedding and detection execution time have been measured between 5 and 10 seconds for neighborhood operators given by (11) and (14), respectively.

It was found from the experiments that the principal component axis of the simplified object differs from the original object's principal component axis up to about 0.7 degrees for mesh simplification factors up to 40 percent (the simplification factor is the percentage of vertices removed from the 3D mesh object by the mesh simplification



(a) (b)

Fig. 8. (a) ROC curves for Dino model for a number of watermarked vertices varying from 300 to 600 and for embedding power 0.001. (b) ROC curves of Dino model for random noise addition.

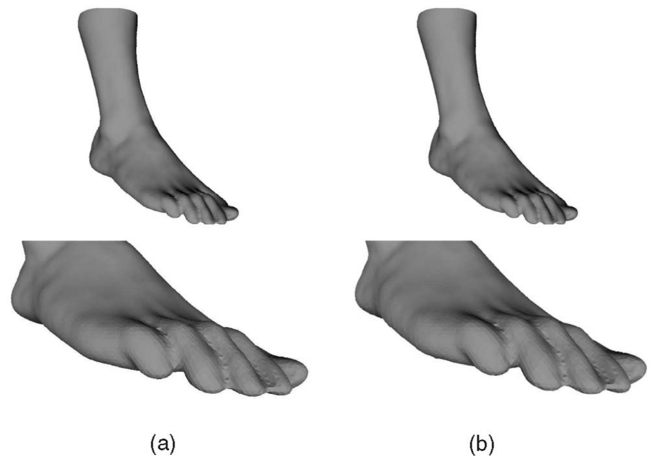


Fig. 9. (a) Foot model. (b) Watermarked Foot model.

TABLE 2
Watermark Detection EER for Foot Model for Various Mesh Simplification Rates

Object Used	Mesh Simplification	EER H(14)
Foot	0.0	1×10^{-9}
	0.2	1.21×10^{-5}
	0.3	4×10^{-4}
	0.4	7.2×10^{-4}
	0.5	1.3×10^{-2}

procedure). Thus, the parameter t that controls the length of the interval $\boldsymbol{\Theta}_j$ was chosen to be 1.4 degrees for achieving robustness against mesh simplification. The parameter b that controls the watermark perceptibility was set to 0.6. The 3D mesh object is "Foot" [29] with 25,845 vertices and 51,690 triangles. The objects have been watermarked using 1,000 random keys and then simplified with various mesh simplification factors. Detection has been performed using the 1,000 correct and 1,000 wrong keys. The original "Foot" can be seen in Fig. 9a, whereas the watermarked object with $t = 1.4$, $b = 0.6$ and using (14) is depicted in Fig. 9b. The SNR measure for the watermarked model was 126.8 dB. The corresponding ROC curves are depicted in Fig. 10a. It can be seen from Fig. 10a that the watermarking method resists simplification attacks fairly well up to 40 percent of the simplification factor. Of course, the level of the simplification that a 3D mesh object resists is also based on the nature of the object. Noise addition at the level of watermark SNR has been also considered. The algorithm resists this attack fairly well. The corresponding ROC curves can be seen in Fig. 10b.

The EER values for the 3D mesh object used in the experiments and for various mesh simplification percentages using the simplification algorithm reported in [24] are summarized in Table 2 using the neighboring operator in (14). The main difference between the two different neighboring operators, (11), (14), is related to the watermark perceptibility. Operators like (11) give a crude approximation of the $r(\mathbf{u}^s)$ component of a vertex $\mathbf{u}^s$ using its neighborhood vertices, without taking into consideration the way these vertices are distributed in the 3D space. The approximation of the $r(\mathbf{u}^s)$ by operators like (14) and (15) is more elegant due to the fact that they take into consideration the way the vertices are distributed locally in the 3D space. These remarks are confirmed by the SNR values of 126.8 dB and 107 dB for neighborhood operators (14), (11), respectively.

5 CONCLUSIONS

Two novel blind 3D mesh object watermarking methods have been proposed in this paper. The POA and SPOA watermarking algorithms are robust against 3D translation, rotation, and uniform scaling (similarity transformations). Furthermore SPOA is robust against mesh simplifications. Both algorithms are based on principal component analysis. Thus, in the case of spherical objects, the algorithms will

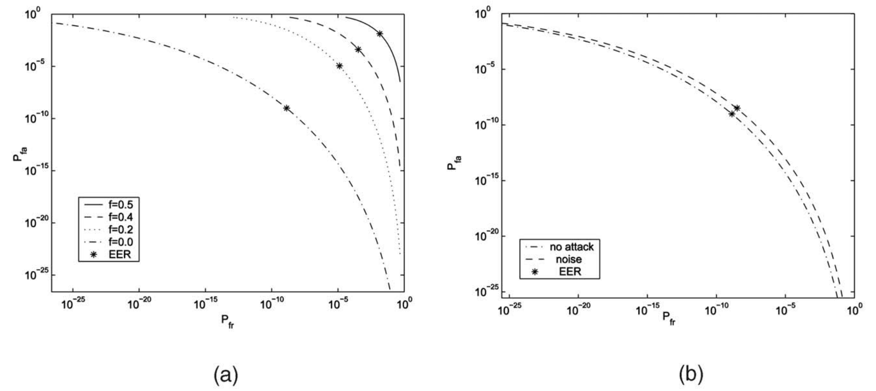


Fig. 10. (a) ROC curve of the Foot model for mesh simplification rates f from 0 percent to 50 percent. (b) ROC curves of Foot model for random noise addition.

become more sensitive to attacks that affect the principal component.

Both algorithms fail against cropping due to the fact that such an attack can cause severe alteration to both the principal object axis and mass center. The cropping attack can be successfully handled if the center of the mass and the principal object axis of the original 3D mesh object are available during watermark detection. Another attack that can cause watermark detection errors is the general affine transformations. Such an attack can cause alterations to the principal object axis and the mass center and, additionally, disturb the entire object geometry.

If the mesh simplification is applied in a nonuniform manner, it may also affect the calculation of the principal axis. That is, if different regions of the mesh are simplified on purpose at different rates, it is quite likely that the principal axis will change significantly and cause loss of synchronization. One way of compensating for this attack is to use nonblind detection and calculate the principal axis in the original object. Another way is to calculate the principal axis of the solid test 3D mesh object instead of using only the object's vertices. The solid 3D mesh object can be constructed by sampling the object's interior space.

Appendix A
Pseudocode of the POA Algorithm

```
label all vertices with 0
while $i < N_w$ or all vertices have not been labeled with -1, 1, or 2
    select an angle $\theta_i$ and a sample $w_i \in \{-1, 1\}$ using the owner's key
    find the vertex $\mathbf{u}_i^s$ such that $\theta(\mathbf{u}_i^s) \approx \theta_i$
    if $l(\mathbf{u}_i^s) = 0$
        if $w_1 = 1$ embed watermark sample in $\mathbf{u}_i^s$ using $g_1$
        else embed watermark sample in $\mathbf{u}_i^s$ using $g_2$
        end if
        label all the neighboring vertices of $\mathbf{u}_i^s$ with 2
        $i \leftarrow i + 1$
    endif
end while
```

Appendix B
Pseudocode of the SPOA Algorithm

```
select a number $\theta(1) \in [0, \pi]$ using the owner's key
$\mathbf{\Theta}(1) \leftarrow [\theta(1), \theta(1) + t)$, $m \leftarrow 1$
while $[0, \pi]$ is not covered do
    select a number $w_m \in \{-1, 1\}$ using the owner's key
    if $w_m = 1$ embed watermark sample in $\mathbf{I}(\mathbf{\Theta}(m))$ using $\mathbf{G}_1$
    else embed watermark sample in $\mathbf{I}(\mathbf{\Theta}(m))$ using $\mathbf{G}_2$
    end if
    select a number $\theta_1(m) \in (0, \epsilon)$ using the owner's key
    $\theta(m+1) \leftarrow \theta(m) + t + \theta_1(m)$, $m \leftarrow m + 1$
    $\mathbf{\Theta}(m) \leftarrow [\theta(m), \theta(m) + t]$
end while
```

Appendix C
Scale Invariance of Detection Ratio

Let O be a 3D mesh object and O_γ be its uniform scaled version by a scaling factor $\gamma > 0$. Let $\mathbf{V}_O^c$ and $\mathbf{V}_{O_\gamma}^c$ be the set of vertices of O and O_γ, respectively, with $\mathbf{u}_i^c \in \mathbf{V}_O^c$ and $\mathbf{v}_j^c \in \mathbf{V}_{O_\gamma}^c$. Thus, $\mathbf{v}_j^c = \gamma \mathbf{u}_i^c$. Let an interval $\mathbf{\Theta}_j \subset [0, \pi]$ of θ angles and the set of vertices $\mathbf{K} = \mathbf{I}(\mathbf{\Theta}_j)$. For this set, $\mathbf{K}$, the random variables $d_r(\mathbf{u}_i^c)$ and $d_r(\mathbf{v}_j^c)$ are formed as in (16). Then:

$$\begin{aligned} d_r(\mathbf{v}_j^s) &= r(\mathbf{v}_j^s) - H(\mathbf{v}_j^s) \\ &= \gamma r(\mathbf{u}_i^s) - \gamma H(\mathbf{u}_i^s) \\ &= \gamma d_r(\mathbf{u}_i^s). \end{aligned} \tag{34}$$

Thus, $\hat{\sigma}_l^\gamma = \gamma \hat{\sigma}_l$ and $\hat{\sigma}_r^\gamma = \gamma \hat{\sigma}_r^s$, where $\hat{\sigma}_l^\gamma$ and $\hat{\sigma}_r^\gamma$ are the corresponding left and right standard deviations estimators of the O_γ in the interval $\mathbf{K}$. Finally,

$$\begin{aligned} Prob(d_r(\mathbf{v}_j^s) > b\hat{\sigma}_l^\gamma) &= Prob(\gamma d_r(\mathbf{u}_i^s) > \gamma b \hat{\sigma}_l) \\ &= Prob(d_r(\mathbf{u}_i^s) > b\hat{\sigma}_l) \\ Prob(d_r(\mathbf{v}_j^s) < -b\hat{\sigma}_r^\gamma) &= Prob(\gamma d_r(\mathbf{u}_i^s) < -\gamma b \hat{\sigma}_r) \\ &= Prob(d_r(\mathbf{u}_i^s) < -b\hat{\sigma}_r). \end{aligned} \tag{35}$$

Thus, the detection ratios D_w and D_w^γ of the objects O and O_γ defined by (27) are equal.

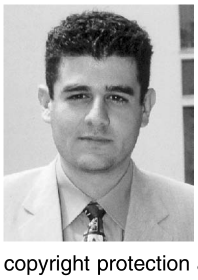
Acknowledgments

The work presented was developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org), funded under the European Commission IST FP6 program.

References

[1] R. Ohbuchi, H. Masuda, and M. Aono, "A Shape-Preserving Data Embedding Algorithm for Nurbs Curves and Surfaces," *Proc. Int'l Conf. Computer Graphics,* pp. 180-187, June 1999.
[2] O. Benedens, "Affine Invariant Watermarks for 3D Polygonal and Nurbs Based Models," *Proc. Third Int'l Workshop Information Security (ISW 2000),* pp. 329-340, Dec. 2000.
[3] J. Lee, N. Cho, and J. Nam, "Watermarking for 3D Nurbs Graphic Data," *Proc. IEEE Workshop Multimedia Signal Processing 2002,* pp. 304-307, Dec. 2002.
[4] F. Garcia and J. Dugelay, "Texture-Based Watermarking of 3-D Video Objects," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 13, no. 8, pp. 853-866, Aug. 2003.
[5] Y. Boon-Lock and M. Minerva, "Watermarking 3D Objects For Verification," *IEEE Computer Graphics and Applications,* special issue on image security, vol. 19, no. 1, pp. 36-45, Jan./Feb. 1999.
[6] O. Benedens, "Geometry-Based Watermarking of 3-D Polygonal Models," *IEEE Computer Graphics and Applications,* special issue on image security, vol. 19, no. 1, pp. 46-45, Jan./Feb. 1999.
[7] R. Ohbuchi, H. Masuda, and M. Aono, "Embedding Data in 3D Models," *Proc. Int'l Workshop Interactive Distributed Multimedia Systems and Telecomm. Services (IDMS '97),* pp. 1-11, Sept. 1997.
[8] T. Harte and A.G. Bors, "Watermarking 3-D Models," *Proc. IEEE Intl Conf. Image Processing,* vol. III, pp. 661-664, Sept. 2002.
[9] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking Three-Dimensional Polygonal Models," *Proc. Fifth ACM Int'l Conf. Multimedia '97,* pp. 261-272, Nov. 1997.
[10] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking Three-Dimensional Polygonal Models through Geometric and Topological Modifications," *IEEE J. Selected Areas in Comm.,* vol. 16, no. 4, pp. 551-560, May 1998.

[11] O. Benedens and C. Busch, "Towards Blind Detection of Robust Watermarks in Polygonal Models," *Proc. EUROGRAPHICS 2000,* pp. 199-209, Aug. 2000.
[12] O. Benedens, "Robust Watermarking and Affine Registration of 3D Meshes," *Information Hiding,* pp. 177-195, 2003.
[13] E. Praun, H. Hoppe, and A. Finkelstein, "Robust Mesh Watermarking," *SIGGRAPH '99 Proc.,* pp. 69-76, 1999.
[14] H. Date, S. Kanai, and T. Kishinami, "Digital Watermarking for 3D Polygons Model Using Multiresolutional Wavelet Decomposition," *Proc. Sixth IFIP WG 5.2 GEO-6,* pp. 296-307, Dec. 1998.
[15] R. Ohbuchi, A. Mukaiyama, and S. Takahashi, "A Frequency Domain Approach to Watermarking 3D Shapes," *Computer Graphics Forum,* vol. 21, pp. 373-382, 2002.
[16] R. Ohbuchi, S. Takahashi, T. Miyasawa, and A. Mukaiyama, "Watermarking 3-D Polygonal Meshes in the Mesh Spectral Domain," *Proc. Computer Graphics Interface,* pp. 9-17, June 2001.
[17] H. Song, N. Cho, and J. Kim, "Robust Watermarking of 3D Mesh Models," *Proc. IEEE Int'l Workshop Multimedia Signal Processing (MMSP 2002),* pp. 332-335, Dec. 2002.
[18] I.J. Cox, M.L. Miller, and J.A. Bloom, *Digital Watermarking.* Morgan Kaufman, 2001.
[19] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolutional Analysis of Arbitrary Meshes," *SIGGRAPH '95 Proc.,* pp. 173-182, Aug. 1995.
[20] Z. Karni and C. Gotsman, "Spectral Compression of Mesh Geometry," *Proc. SIGGRAPH 2000,* pp. 279-286, 2000.
[21] A. Kalivas, A. Tefas, and I. Pitas, "Watermarking of 3D Models Using Principal Component Analysis," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* vol. 5, pp. 676-679, Apr. 2003.
[22] E. Angel, *Interactive Computer Graphics: A Top-Down Approach with OpenGL,* third ed. Addison-Wesley, 2002.
[23] T. Nishita, T. Sederberg, and M. Kakimoto, "Ray Tracing Trimmed Rational Surface Patches," *Computer Graphics,* vol. 24, no. 4, pp. 337-345, 1990.
[24] W.J. Shroder, J.A. Zarge, and W.E. Lorensen, "Decimation of Triangle Meshes," *SIGGRAPH '92 Proc.,* pp. 65-70, 1992.
[25] M. Garland and P.S. Heckbert, "Surface Simplification Using Quadratic Error Metrics," *SIGGRAPH '97 Proc.,* pp. 189-198, 1997.
[26] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, "Simplification Envelopes," *SIGGRAPH '96 Proc.,* pp. 119-128, 1996.
[27] "Stanford Bunny," http://www.cc.gatech.edu/projects/large_models/bunny.html, 2003.
[28] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* third ed. New York: McGraw-Hill, 1991.
[29] "Fastscan," http://fastscan3d.com/download/samples/, 2003.



Stefanos Zafeiriou received the BS degree from the Department of Informatics of Aristotle University of Thessaloniki with honors in 2003. He is currently a researcher and teaching assistant and he is studying toward the PhD degree in the Department of Informatics at the University of Thessaloniki. His current research interests lie in the areas of signal and image processing, pattern recognition, and computer vision, as well as in the area of watermarking for copyright protection and authentication of digital media.

Anastasios Tefas received the BSc degree in informatics in 1997 and the PhD degree in informatics in 2002, both from the Aristotle University of Thessaloniki, Thessaloniki, Greece. From 1997 to 2002, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. From 2003 to 2004, he was a temporary lecturer in the Department of Informatics, University of Thessaloniki where he is currently, a senior researcher. He has coauthored more than 50 journal and conference papers. His current research interests include computational intelligence, pattern recognition, digital signal and image processing, detection and estimation theory, and computer vision. He is a member of the IEEE.

Ioannis Pitas received the diploma in electrical engineering in 1980 and the PhD degree in electrical engineering in 1985, both from the Aristotle University of Thessaloniki, Greece. Since 1994, he has been a professor in the Department of Informatics, Aristotle University of Thessaloniki. From 1980 to 1993, he served as a scientific assistant, lecturer, assistant professor, and associate professor in the Department of Electrical and Computer Engineering at the same university. He served as a visiting research associate at the University of Toronto, Canada, University of Erlangen-Nürnberg, Germany, Tampere University of Technology, Finland, as a visiting assistant professor at the University of Toronto, and as a visiting professor at the University of British Columbia, Vancouver, Canada. He was a lecturer in short courses for continuing education. He has published more than 140 journal papers, 350 conference papers, and contributed to 18 books in his areas of interest. He is the coauthor of the books *Nonlinear Digital Filters: Principles and Applications* (Kluwer, 1990), *3-D Image Processing Algorithms* (J. Wiley, 2000), *Nonlinear Model-Based Image/Video Processing and Analysis* (J. Wiley, 2001), and the author of *Digital Image Processing Algorithms and Applications* (J. Wiley, 2000). He is the editor of the book *Parallel Algorithms and Architectures for Digital Image Processing, Computer Vision and Neural Networks* (Wiley, 1993). He has also been an invited speaker and/or member of the program committee of several scientific conferences and workshops. In the past, he served as an associate editor of the *IEEE Transactions on Circuits and Systems*, *IEEE Transactions on Neural Networks*, *IEEE Transactions on Image Processing*, *EURASIP Journal on Applied Signal Processing*, and coeditor of *Multidimensional Systems and Signal Processing*. He was general chair of the 1995 IEEE Workshop on Nonlinear Signal and Image Processing (NSIP '95), technical chair of the 1998 European Signal Processing Conference, and general chair of IEEE ICIP 2001. His current interests are in the areas of digital image and video processing and analysis, multidimensional signal processing, watermarking, and computer vision. He is a senior member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.