Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 305960, 12 pages
http://dx.doi.org/10.1155/2014/305960

Research Article

Applying 3D Polygonal Mesh Watermarking for Transmission Security Protection through Sensor Networks

Roland Hu,[1] Li Xie,[1] Huimin Yu,[1] and Baocang Ding[2]

[1] *Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*
[2] *School of Electronic and Information Engineering, Xi'an Jiaotong University, Xi'an 710049, China*

Correspondence should be addressed to Li Xie; xiehan@zju.edu.cn

Received 24 September 2013; Accepted 31 October 2013; Published 9 January 2014

Academic Editor: Ge Guo

Although many research works have been carried out in the area of transmission 3D data through sensor networks, the security issue of transmission remains to be unsolved. It is important to develop systems for copyright protection and digital right management (DRM). In this paper, a blind watermarking algorithm is proposed to protect the transmission security of 3D polygonal meshes through sensor networks. Our method is based on selecting prominent feature vertices (prongs) on the mesh and then embedding the same watermark into their neighborhood regions. The embedding algorithm is based on modifying the distribution of vertex norms by using quadratic programming (QP). Decoding results are obtained by a majority voting scheme over neighborhood regions of these prongs. Assuming that cropping cannot remove all prongs, we can achieve robustness against the cropping attack both theoretically and experimentally. Experiments indicate that the proposed method is also robust against noise, smoothing, and mesh simplification. The proposed method has provided a solution for 3D polygonal watermarking which is potential to withstand a variety of attacks.

1. Introduction

Nowadays, the processing, transmission, and visualization of 3D objects are a part of possible and realistic functionalities over sensor networks [1]. Confirmed 3D processing techniques exist and a large scientific community works hard on open problems and new challenges, including progressive transmission, fast access to huge 3D databases, or content security management. Although many research works have been carried out in the area of transmission 3D data through sensor networks, the security issue of transmission remains to be unsolved. 3D objects can be duplicated, modified, transformed, and shared easily during the transmission process. In this context, it is important to develop systems for copyright protection and digital right management (DRM).

Watermarking is a promising area for reinforcing the security of 3D object transmission, which has received much attention in the past years, as summarized by [2, 3]. 3D objects can be represented by polygonal meshes [1], nonuniform rational B-splines (NURBS) [4], point-sampled surfaces, [5] and voxel representation [6]. Among these structures, polygonal mesh is the most popular one due to its simplicity and easiness to be converted to other representations. A 3D polygonal mesh is represented by a set of vertices and connections. In consequence, watermark can be embedded by modifying positions or connections of these vertices.

Watermarking algorithms can be classified into blind and nonblind ones. In the blind watermarking, only the watermarked objects are needed for the decoding process, while in the nonblind watermarking, both the original and watermarked objects are needed. Blind watermarking has wider applications, but it is generally more difficult to be designed, and not as robust as nonblind one in resisting attacks. Algorithms for 3D mesh watermarking can be classified into the spacial domain methods [7–13] and transformed domain ones [14–19]. For the spatial domain methods, the vertices, the normals, and the geometrical invariants are modified for embedding, while for the transformed domain methods, 3D watermarking is treated as a common signal processing problem. The regular signal processing conceptions, such

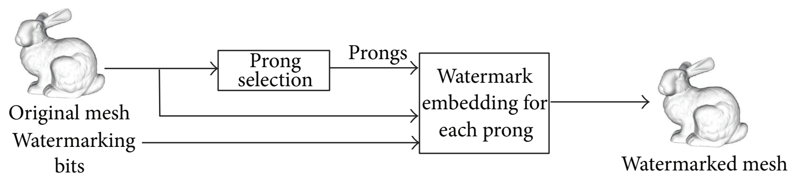


Figure 1: The block diagram of watermark embedding.

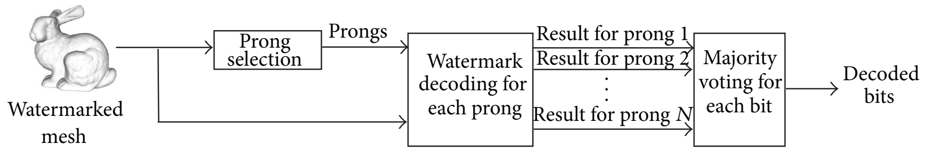


Figure 2: The block diagram of watermark decoding.

as frequency analysis and wavelet decomposition, are implemented for the watermarking approach.

A watermarked mesh is likely to be attacked by a malicious user with the aim of eliminating the embedded watermark. In addition, channel noise can also degrade the watermark during the transmission process. Thus, robustness against attacks and transmission channels is one of the major concerns in designing a watermarking system. Common attacks include affine transforms (rotation, scaling, and translation), noise, smoothing, connectivity attacks (modification of vertex connectivity), vertex reordering (reordering the sequence of vertices), simplification (removing vertices and faces but keeping the 3D shape unchanged), remeshing (resampling the 3D objects to obtain new meshes), and cropping (part of the 3D mesh being cropped from the original mesh) [1–3]. Each algorithm has its own preferences in resisting attacks. For example, compared with the spatial domain approaches, the transformed domain ones are more robust against noise. However, most transformed domain methods are not robust against connectivity attacks because they use connectivity information for watermarking.

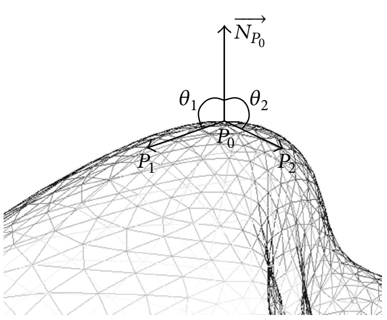
Cropping is a special attack which aims at removing part of the mesh. It is possible to design a watermarking system which is robust against cropping by repetitively embedding the same watermark into different patches of the mesh. Suppose that several patches remained after the cropping attack; it is possible to recover watermark from these unaffected patches. The work of Ohbuchi et al. [20] has been one of the first to propose a nonblind approach based on repetitive embedding to resist the cropping attack. Other nonblind approaches are proposed in [21, 22]. For nonblind watermarking, the original mesh serves as a reference to indicate embedding regions. By using synchronization and registration techniques, it is not difficult to extract the embedded watermark. However, for blind watermarking, the situation is relatively more difficult because of the lack of a reference to indicate the place where watermarking occurs. The basic idea is to segment the mesh into patches, which have special geometrical and topological properties, as references for watermark embedding, with the expectation that the same patches can be extracted during the decoding process [10, 12, 23]. Such an approach has aroused the causality problem, the watermarking algorithm should produce the same patches before and after embedding, which is a difficult problem because embedding may change mesh properties which are important to the segmentation algorithm. From our knowledge, this problem is not fully solved by the previous research works.

In this paper, we propose a blind watermarking scheme which is robust to various routine attacks during transmission of 3D polygonal meshes through sensor networks, for example, noising, smoothing, simplification, and cropping. The basic idea follows our previous work [36] and the work of Rondao-Alface et al. [25] Firstly, protrusive feature vertices (referred as “prongs” in this paper) of the mesh are selected as references for segmentation. Secondly, watermark is repetitively embedded into neighborhood regions of these prongs. Because the selection procedure is local, prongs are evenly distributed on the mesh. If an attacker crops all the prongs, most probably he/she will obtain a meaningless mesh, so it is likely that there will be several prongs remained after the cropping attack. For decoding, prongs are retrieved and then their neighborhood regions obtained. Watermark can be decoded from the neighborhood region associated with each prong. Then a majority voting scheme is used to obtain the final decoding results.

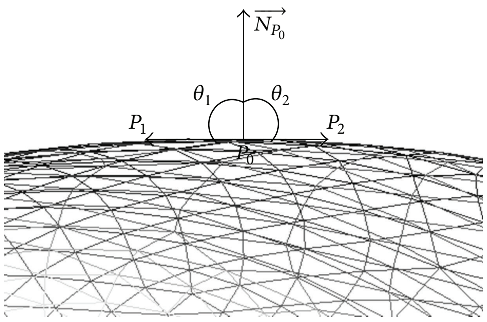
The rest of this paper is organized as follows. The proposed watermarking scheme is described in Section 2, which includes prong selection, watermark embedding, and decoding. Robustness of the proposed method against cropping attack is shown in Section 3. Section 4 theoretically proves that the performance of the watermarking scheme is a function of the number of correct prongs and the total number of prongs. Section 5 shows simulation results of the proposed method against noise, smoothing, and simplification attacks. Finally, Section 6 concludes this paper.

2. The Proposed Watermarking Method

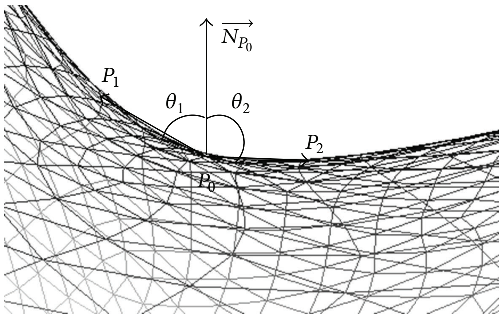
Figures 1 and 2 illustrate the watermark embedding and decoding processes, which are described in the following subsections.



(a) Prongness < 0



(b) Prongness = 0



(c) Prongness > 0

FIGURE 3: The prongness value is negative for protrusive vertices and positive for concave vertices.

2.1. Selecting Prongs. The first step of watermark embedding is to select prominent feature vertices, or "prongs", from 3D meshes. Feature vertices are common descriptors of 3D surfaces, which have been used for mesh segmentation [26, 27] and object recognition [28]. In this application, we require feature vertices to be protrusive because protrusive regions contain most information of the shape [29]. If an attacker removes all protrusive regions, most probably he/she would obtain a meaningless shape. We associate a prongness value to each vertex. In this paper, the prongness value is an indicator of how protrusive a vertex could be. Prongness value is calculated by adding up the dot products between the normal direction of this vertex and the vector from this vertex to its nearest neighbors:

$$\text{Prongness}\left(P_0\right) = \sum_{P\in V_1(P_0)} \frac{\overrightarrow{PP_0}\cdot\overrightarrow{N_{P_0}}}{\left|\overrightarrow{PP_0}\right|\left|\overrightarrow{N_{P_0}}\right|}, \tag{1}$$

where $\overrightarrow{N_{P_0}}$ represents the normal direction of vertex P_0 and $V_1(P_0)$ is the set of vertices which are close to vertex P_0 in geodesic distance. Various methods have been proposed to calculate normal directions in 3D meshes [30, 31]. In this paper, we use a simple and common one. The first step is to calculate the surface normal $\overrightarrow{N_j}$ at each polygon from the neighborhood of P_0. The surface normal of a polygon is calculated as the vector product of the orientations of two of its edges divided by the vector length. The normal direction at vertex P_0 is taken as the average of surface normal directions corresponding to its adjacent polygons; that is,

$$\overrightarrow{N_{P_0}} = \frac{\sum_{V_j\in V(P_0)}\overrightarrow{N_j}}{N}, \tag{2}$$

where $V(P_0)$ is the set of vertices which are connected with P_0, and N is the total number of these vertices.

The algorithm to calculate geodesic distances was presented by Dijkstra [32]. In 1998, Kimmel and Sethian proposed the fast marching algorithm running in complexity $O(n\log n)$ [33]. The fast marching algorithm was modified by our previous work to increase its speed [34, 35]. In this paper, we use the method of our previous work to calculate geodesic distances. The prongness calculation is to add up the cosines between $\overrightarrow{N_{P_0}}$ and $\overrightarrow{PP_0}$ for all P's which are geodesically close to P_0, as shown in Figure 3.

Prongs are selected as local minimums of prongness values. In this paper, a vertex is selected as a prong if it takes the lowest prongness value compared with its neighborhood vertices. We use $V_2(P_0)$ to represent the set of P_0's neighborhood vertices which need to be compared with P_0. We choose $V_2(P_0)$ as the set of the first N_2 vertices which are close to P_0 in geodesic distance. In other words, a vertex is selected as a prong if it has the lowest prongness value among its closest N_2 vertices in geodesic distance.

Figure 4 shows the selected prongs of the bunny, head, and hand models. Each prong is represented as a red point on the mesh. There are also prongs at the back of each model, which are not shown here. We choose $N_1 = 200$, which means that we take the closest 200 vertices to calculate the prongness

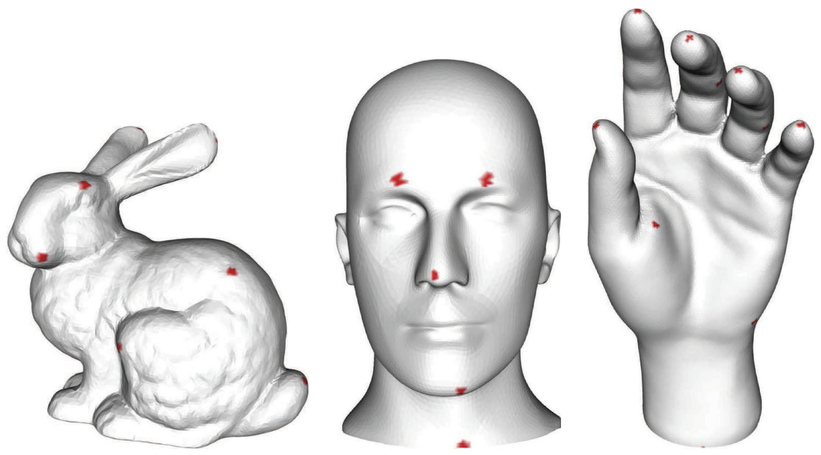

(a) Bunny model (b) Head model (c) Hand model

Figure 4: The prongs of three models.

value. For the head and bunny models, which have 11,703 and 14,007 vertices, respectively, we choose $N_2 = 1000$. In other words, a prong is selected by comparing its prongness value with its 1000 neighborhood vertices in geodesic distance. For the hand model, which has 38,219 vertices, we set $N_2 = 1600$, a little bit greater than the bunny and hand models to limit the number of selected prongs. We can see that prongs are scattered in protrusive regions and evenly distributed on the whole mesh.

2.2. Segmentation Based on Prongs. The next step is to segment patches for watermarking based on the obtained prongs. These patches are geodesic circles centered on the prongs. For each prong, we segment the patch by taking the first N_3 vertices which are geodesically close to it. Here N_3 is a predefined value, which is decided by considering the number of prongs, the watermarking algorithm, and the number of watermarking bits to be embedded. For example, we can choose $N_3 = 600$ for the bunny model if we need to embed 32 bits by using the histogram-based approaches, because the histogram-based algorithm can obtain good results if approximately 20 vertices are used to embed each bit [13].

Figure 5 shows the patches of the bunny model by taking $N_1 = 1000$, $N_2 = 200$, and $N_3 = 600$, where each patch is represented by a unique color. Because there are overlapping regions belonging to more than one patch, we use the blue color to represent them.

The next step is to embed the same watermark into each patch. The embedding algorithm is an extension of the histogram-based approach proposed by [13], which is based on modifying the distribution of vertex norms. We have extended their work by minimizing distortions of vertex norms using quadratic programming (QP) and thus further improved its robustness [36]. In this paper, we use the proposed QP method for watermark embedding.

Another advantage of the QP method is the simplicity to deal with overlapping vertices (vertices with the blue color in Figure 5). Because overlapping vertices belong to more than one patch, they will be assigned different displacements during the watermarking process. Thus, embedding into one patch may destroy watermarks of other patches. By using the QP method, the overlapping vertices can be constrained to their original positions to ensure correct embedding.

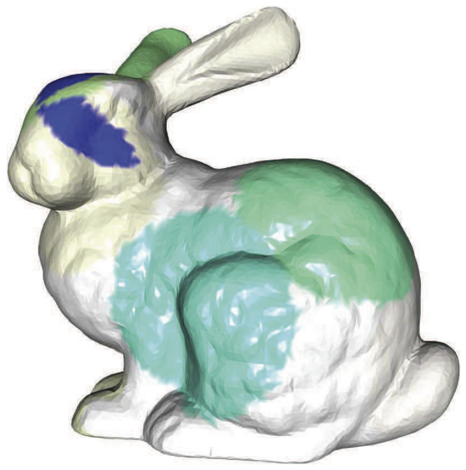


Figure 5: The patches of the bunny model by choosing $N_1 = 1000$, $N_2 = 200$, and $N_3 = 600$. Patches are shown by different colors, and overlapping regions are shown by blue.

2.3. The Embedding Algorithm. In this section, we describe our embedding algorithm. Part of this section has been published in [36]. The main difference is that we have added a scheme to deal with overlapping vertices belonging to more than one patch.

For embedding watermark into each patch, firstly, the Cartesian coordinates of vertices in that patch are converted into spherical coordinates $(\rho_i, \theta_i, \phi_i)$:

$$\rho_i = \sqrt{\left(x_i - x_g\right)^2 + \left(y_i - y_g\right)^2 + \left(z_i - z_g\right)^2},$$

$$\theta_i = \arctan \frac{y_i - y_g}{x_i - x_g},$$
$$\phi_i = \arccos \frac{z_i - z_g}{\rho_i}, \quad i \in \{0, 1, \ldots, L-1\}, \tag{3}$$

where L is the number of vertices in the patch, ρ_i is the ith vertex norm, and (x_g, y_g, z_g) is the patch's center of gravity, which can be calculated as

$$x_g = \frac{1}{L}\sum_{i=0}^{L-1} x_i, \qquad y_g = \frac{1}{L}\sum_{i=0}^{L-1} y_i, \qquad z_g = \frac{1}{L}\sum_{i=0}^{L-1} z_i. \tag{4}$$

Secondly, vertex norms are divided into N distinct bins according to their magnitude. Each bin is used to hide one bit of watermark. In this paper, we use ω_n to represent the watermarking bit to be embedded into the nth bin.

Here $\rho_{\min}$ and $\rho_{\max}$ represent the minimum and maximum values of all vertex norms. The nth bin B_n is defined as follows ($n \in \{0, 1, \ldots, N-1\}$):

$$\rho_{n,\min} = \rho_{\min} + \frac{n(\rho_{\max} - \rho_{\min})}{N},$$
$$\rho_{n,\max} = \rho_{\min} + \frac{(n+1)(\rho_{\max} - \rho_{\min})}{N}, \tag{5}$$
$$B_n = \left\{\rho_{n,j} \mid \rho_{n,\min} \le \rho_{n,j} \le \rho_{n,\max}\right\}.$$

Here $\rho_{n,\min}$ and $\rho_{n,\max}$ are lower and upper boundaries of the nth bin and $\rho_{n,j}$ is the jth vertex norm in the nth bin. In this paper, we also use $\theta_{n,j}$ and $\phi_{n,j}$ to represent the spherical angles of the jth vertex norm in the nth bin. In addition, we use M_n to represent the number of vertex norms belonging to the nth bin.

The third step is to map the vertex norms belonging to the nth bin to the normalized range $[0, 1]$:

$$\widetilde{\rho_{n,j}} = \frac{\rho_{n,j} - \rho_{n,\min}}{\rho_{n,\max} - \rho_{n,\min}}, \tag{6}$$

where $\widetilde{\rho_{n,j}}$ is the normalized jth vertex norm in the nth bin. The aim of the watermarking process is to slightly modify $\widetilde{\rho_{n,j}}$, so that the mean of the vertex norms is moved into a specific range according to the watermarking bit to be embedded. We introduce the normalized distortion for the jth vertex in the nth bin, which is represented by $\Delta\widetilde{\rho_{n,j}}$. Our aim is to calculate each $\Delta\widetilde{\rho_{n,j}}$. After $\Delta\widetilde{\rho_{n,j}}$ is obtained, we can calculate the new vertex norm $\widetilde{\rho_{n,j}}'$ by adding the previous one with its distortion:

$$\widetilde{\rho_{n,j}}' = \widetilde{\rho_{n,j}} + \Delta\widetilde{\rho_{n,j}}. \tag{7}$$

Then we need to transform the vertex norms to the original ones by (8), which is an inverse transformation of (6):

$$\rho'_{n,j} = \widetilde{\rho_{n,j}}'(\rho_{n,\max} - \rho_{n,\min}) + \rho_{n,\min}. \tag{8}$$

The watermark embedding process is completed by converting the spherical coordinates to Cartesian coordinates. Let ρ'_i be the ith vertex norm. A watermarked mesh consisting of vertices (x'_i, y'_i, z'_i) is obtained by

$$x'_i = \rho'_i \cos\theta_i \sin\phi_i + x_g,$$
$$y'_i = \rho'_i \sin\theta_i \sin\phi_i + y_g, \tag{9}$$
$$z'_i = \rho'_i \cos\phi_i + z_g.$$

Our aim is to minimize the sum of squares of $\Delta\widetilde{\rho_{n,j}}$:

$$\text{Minimize:} \quad \sum_{n=0}^{N-1}\sum_{j=0}^{M_n-1} \Delta\widetilde{\rho_{n,j}}^2. \tag{10}$$

Three constraints are applied to ensure that the embedded watermarking bits can be correctly decoded later. The first constraint is to limit the distortion of each vertex into a reasonable range. If a vertex belongs to more than one patch (the vertex with the blue color in Figure 5), then its displacement is set to zero; for the nonoverlapping vertices, we limit the transformed vertex norm $\widetilde{\rho_{n,j}}'$ into the range of $[\Delta G, 1 - \Delta G]$. Here ΔG is a parameter to control the distance gap between adjacent bins. The constraint is given as follows.

Constraint 1. For every $n \in \{0, 1, \ldots, N-1\}$ and $j \in \{0, 1, \ldots, M_n - 1\}$, if vertex $P_{n,j}$ is an overlapping vertex which belongs to more than one patch, then

$$\Delta\widetilde{\rho_{n,j}} = 0, \tag{11}$$

else

$$\Delta G - \widetilde{\rho_{n,j}} \le \Delta\widetilde{\rho_{n,j}} \le 1 - \Delta G - \widetilde{\rho_{n,j}}. \tag{12}$$

As discussed in Section 2.2, the overlapping vertices have to be constrained into their original positions to ensure correct embedding, and nonoverlapping vertices are modified according to this constraint.

We can see from (7) and (12) that after watermarking, $\widetilde{\rho_{n,j}}'$ will be in the range of $[\Delta G, 1 - \Delta G]$ if the above constraint is satisfied, so Constraint 1 ensures that vertices belonging to the nth bin still belong to that bin after the watermarking process, which is also implied in [13].

The second constraint is directly derived from [13], which ensures that the mean of the transformed vertex norms in the nth bin is greater (or smaller) than a reference value when the embedded watermarking bit $\omega_n = +1$ (or $\omega_n = -1$). This constraint must be satisfied to ensure that the embedded watermarking bits could be correctly extracted later. Our aim is to make the mean of the vertex norms in the nth bin:

$$\widetilde{\mu_n}' = \frac{1}{M}\sum_{j=0}^{M_n-1} \widetilde{\rho_{n,j}}' \tag{13}$$

greater than $1/2 + \alpha$ (or smaller than $1/2 - \alpha$) when $\omega_n = +1$ (or $\omega_n = -1$). Here α is a strength factor to control the watermarking effect. The second constraint is given as follows.

Constraint 2. For every $n \in \{0, 1, \ldots, N-1\}$,

(1) if $\omega_n = +1$, then

$$\sum_{j=0}^{M_n-1} \Delta\widetilde{\rho_{n,j}} > M_n\left(\frac{1}{2}+\alpha\right) - \sum_{j=0}^{M_n-1} \widetilde{\rho_{n,j}}; \tag{14}$$

(2) if $\omega_n = -1$, then

$$\sum_{j=0}^{M_n-1} \Delta\widetilde{\rho_{n,j}} < M_n\left(\frac{1}{2}-\alpha\right) - \sum_{j=0}^{M_n-1} \widetilde{\rho_{n,j}}. \tag{15}$$

It can be deduced from (7) and (13) that when Constraint 2 is satisfied, $\widetilde{\mu_n}'$ is greater than $1/2+\alpha$ (or smaller that $(1/2)-\alpha$) when $\omega_n = +1$ (or $\omega_n = -1$).

We proposed another constraint to guarantee that the center of gravity of the watermarked patch is the same as the original one. If the center of gravity (x_g, y_g, z_g) has been changed, by (3), the vertex norms ρ_i will also be changed. Thus, it is possible that the decoding process fails to extract the embedded bits. Such a problem is not addressed in [13]. Here we propose the following constraint to solve it.

Constraint 3.

$$\sum_{n=0}^{N-1}\sum_{j=0}^{M_n-1} \Delta\widetilde{\rho_{n,j}} \cos\theta_{n,j} \sin\phi_{n,j} = 0,$$
$$\sum_{n=0}^{N-1}\sum_{j=0}^{M_n-1} \Delta\widetilde{\rho_{n,j}} \sin\theta_{n,j} \sin\phi_{n,j} = 0, \tag{16}$$
$$\sum_{n=0}^{N-1}\sum_{j=0}^{M_n-1} \Delta\widetilde{\rho_{n,j}} \cos\phi_{n,j} = 0.$$

Thus, we have changed the problem of assigning distortions to an optimization problem, with a quadratic objective function and three linear constraints. This is exactly a quadratic programming problem and can be solved efficiently [37, Chapter 4].

2.4. Solving the Causality Problem. Because watermark embedding modifies positions of vertices which are close to each prong, the local minimum of the prongness value is also likely to be changed. If this situation occurs, the prongs cannot be retrieved during the decoding process. Such a problem, referred as the causality problem in [3], is illustrated in Figure 6(a).

Suppose after the watermarking process, P_0, P_1, and P_2 have been changed to P_0', P_1', and P_2'. It can be seen that the prongness value of P_0 increases because the angles between the norm $\overrightarrow{N_{P_0}}$ and $\overrightarrow{P_0P_1}$, $\overrightarrow{P_0P_2}$ become smaller. Thus, after watermarking, other vertices may substitute P_0 as the new local minimum in prongness value.

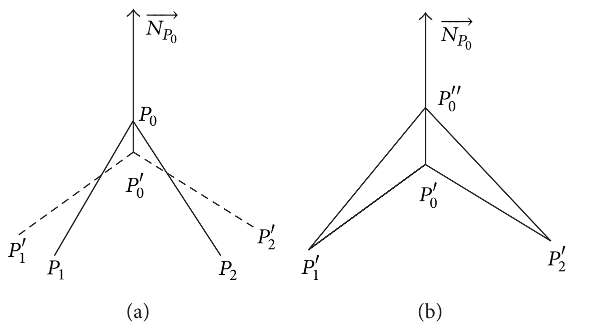


Figure 6: An example of solving the causality problem. (a) After the watermarking process, P_0, P_1, and P_2 have been modified to P_0', P_1', and P_2'. In this situation, the prongness value of P_0 increases, so other vertices may substitute P_0 to be a new local minimum. (b) P_0' can be slightly moved along its norm direction to P_0'', so that its prongness value can be decreased and local minimum again obtained.

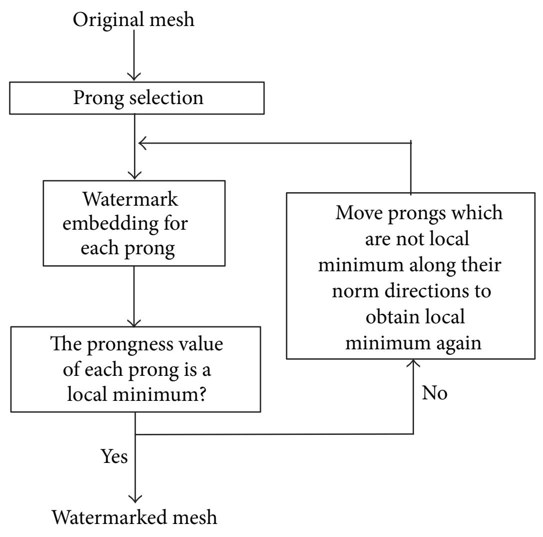

Original mesh
Prong selection
Watermark embedding for each prong
The prongness value of each prong is a local minimum?
Move prongs which are not local minimum along their norm directions to obtain local minimum again
No
Yes
Watermarked mesh

Figure 7: The block diagram of iterative embedding.

We propose an iterative approach to solve the causality problem—ensuring that the same prongs can be retrieved after the watermarking process. Firstly, we embed the watermark into the neighborhood region of each prong. After embedding, we check whether the prongness value of each prong is the local minimum among its neighborhood vertices. If so, the embedding is successful and the iteration finishes; else we slightly move the prong along its normal direction to decrease its prongness value and obtain local minimum again, as shown in Figure 6(b).

The iteration continues until watermark is embedded and simultaneously all prongs take local minimum prongness values. We also set up a maximum iteration number $M_{\text{iteration}} = 20$, which means that if this process does not converge after 20 rounds, watermarking is stopped and embedding is failed on this prong. Figure 7 illustrates the procedure of the iterative process. In experiments, around 1 out of 10 prongs fails to be embedded. Considering that there are more than one prong in a mesh, this ratio does not significantly influence the performance of the whole scheme.

Figure 8 shows the watermarking effects of the bunny, head, and hand models. In each model, 32 bits are repetitively embedded into each patch.

2.5. Watermark Decoding. The watermark decoding process is as follows. We assume that the decoder knows the values of N_1, N_2, and N_3. These values can be transmitted with the watermarked mesh, or predefined. Based on N_1, N_2, and N_3, we can obtain the prongs and patches by using the same method as the embedding process. Then we can extract watermark for each patch. Firstly, the center of gravity of each patch is calculated by (4); then the Cartesian coordinates are converted to spherical coordinates by (3). After obtaining the maximum and minimum, the vertex norms are classified into N bins and mapped onto the range of $[0, 1]$ by (5) and (6). Then, the mean of the nth bin $\widetilde{\mu_n}$ is calculated by (13), and compared with the reference value $1/2$. The watermark hidden in the nth bin, represented by ω_n, is extracted by

$$\omega_n = \begin{cases} +1 & \text{if } \widetilde{\mu_n} > \dfrac{1}{2}, \\ -1 & \text{if } \widetilde{\mu_n} < \dfrac{1}{2}. \end{cases} \tag{17}$$

Each patch produces a series of decoded bits. The final result is obtained by a majority voting scheme. Each bit is decoded by counting the 0's and 1's of all the series at that bit. If there are more 0's than 1's, the bit is decoded as 0; otherwise it is decoded as 1.

3. Robustness Test against the Cropping Attack

Because watermark embedding is a local process, several prongs remain unchanged after the cropping attack. It is also possible that cropping introduces new prongs that no watermark is embedded around it. However, the majority voting scheme ensures that if the number of incorrect prongs is less than half, watermark can be extracted without errors. In Section 4, we will show that low bit error rates can also be achieved even if the number of incorrect prongs is more than half. Figure 9 shows the bunny, head, and hand models where 50% vertices of each model have been cropped (i.e., cropping ratio = 50%). Compared with the original meshes in Figure 8, it can be seen that many prongs remain on the cropped meshes, but cropping also introduces new prongs.

We embed 32 bits into each mesh. The settings of N_1, N_2, and N_3 for each model are the same as in Section 2. The final decoding results are shown in Table 1. We have listed the bit error rates (BER) of each model with three cropping ratios. The BER is calculated as the ratio of incorrectly decoded bits to all embedded ones. We also listed the total number of prongs in the cropped mesh, and the number of prongs in which we have embedded the watermark (denoted by "correct prongs").

From Table 1, we can see that the watermark can be correctly decoded even if 70% of the mesh has been cropped. The robustness is decided by the number of correct prongs and the number of all detected prongs. If the number of correct prongs is more than half of the number of all detected prongs, the embedded bits can be decoded without errors. Otherwise there will be decoding errors, such as the first line of the bunny model, where 2 out of 6 prongs are correct prongs and the BER is 12.5% in this situation.

4. Theoretical Analysis of the Decoding Scheme

In this section, we will theoretically prove that the performance of the watermarking scheme is a function of the correct prongs and the total number of prongs. Suppose the cropped mesh totally produces N prongs, in which there are M correct prongs and $N - M$ incorrect prongs. We further assume that each of the M prongs decodes the watermarking bits without any error, and the N incorrect prongs randomly guess the watermarking bits (i.e., half correct and half wrong). Then the probability that a bit can be correctly decoded in this situation, denoted by $P(N, M)$, can be obtained as follows:

if N is odd, then

$$P(N, M) = \left(\frac{1}{2}\right)^{(N-M)} \sum_{k=((N+1)/2)-M}^{N-M} C_{N-M}^{k}; \tag{18}$$

if N is even, then

$$P(N, M) = P(N + 1, M). \tag{19}$$

The deduction of the above equation is as follows. When N is odd, by the rule of majority voting, a watermarking bit can be correctly decoded if at least $(N + 1)/2$ prongs correctly decode that bit. Because from M of these N prongs we can always obtain the correct bit, for the remaining $N-M$ prongs, at least $(((N + 1)/2) - M)$ prongs should produce the correct bit. We further assume that the probability of correct decoding for each of the remaining $(N - M)$ prongs is $1/2$ (i.e., half correct and half wrong), so the total probability that one bit can be correctly decoded is $(1/2)^{(N-M)}$ times the sum of C_{N-M}^{k}, with k varying from $(((N + 1)/2) - M)$ to $(N - M)$, as indicated in (18).

When N is even, the probability that the watermarked bit can be correctly decoded is considered under two situations.

Situation 1. Correct decoding can be obtained if there are at least $((N/2)+1)$ prongs correctly decoding the bit. By similar deductions, the probability relating to this situation is

$$P_1 = \left(\frac{1}{2}\right)^{(N-M)} \sum_{k=(N/2)+1-M}^{N-M} C_{N-M}^{k}. \tag{20}$$

Situation 2. When exactly $N/2$ prongs correctly decode the bit, the decoder will randomly guess the watermarking bit. We further assume that the probability of correct guess is $1/2$. Then the probability related to this situation is

$$P_2 = \left(\frac{1}{2}\right)^{(N-M+1)} C_{N-M}^{(N/2)-M}. \tag{21}$$

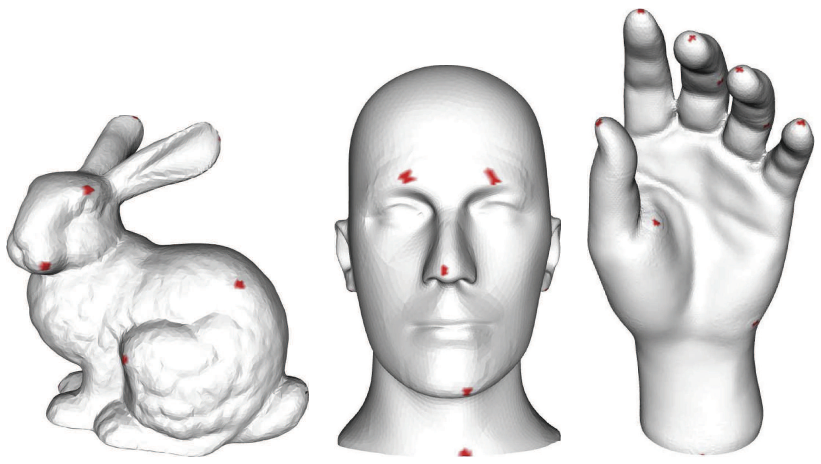


FIGURE 8: The watermarked meshes of the bunny, head, and hand models. For each model, 32 bits are repetitively embedded into each patch. (a) Bunny, $N_1 = 200$, $N_2 = 1000$, and $N_3 = 600$; (b) head, $N_1 = 200$, $N_2 = 1000$, and $N_3 = 600$; (c) hand, $N_1 = 200$, $N_2 = 1600$, and $N_3 = 1000$.

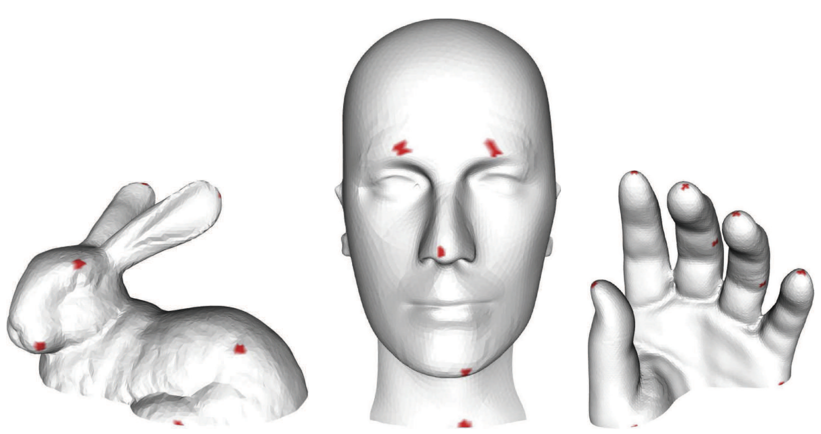


FIGURE 9: The cropped meshes of the bunny, head, and hand models (cropping ratio = 50%).

If we sum up probabilities of these two situations, we can obtain the probability of correct decoding when N is even:

$$\begin{aligned} P(N,M) &= P_1 + P_2 \\ &= \left(\frac{1}{2}\right)^{(N-M+1)} \sum_{k=(N/2)+1-M}^{N-M+1} C_{N-M+1}^{k} \qquad (22) \\ &= P(N+1,M). \end{aligned}$$

Figure 10 plots $P(N, M)$ with $M = 1, 2, 3, 4, 5$, respectively. It can be observed that the theoretical analysis coincides with experimental results. For example, it can be seen from the first line of Table 1 that the BER is 12.5% when the number of correct prongs is 2 and the number of all prongs is 6, while the theoretical curve in Figure 10 indicates that $P(6, 2) = 0.8125$. Thus, the theoretical BER is $1 - 0.8125 = 18.75\%$, which is consistent with the experimental results.

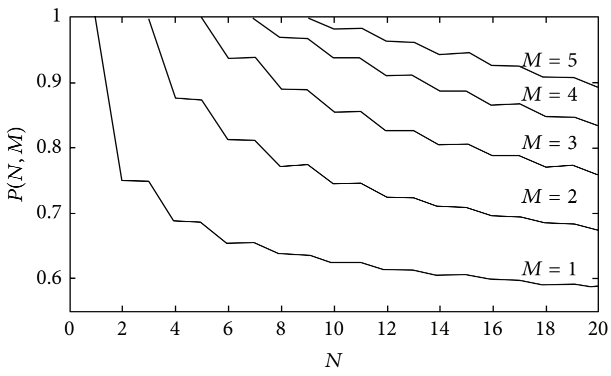


FIGURE 10: The graph of $P(N, M)$ with $M = 1, 2, 3, 4, 5$, respectively.

From the above analysis, we can see that the performance of watermark decoding can be improved if N takes smaller value and M takes greater value. The number of correct

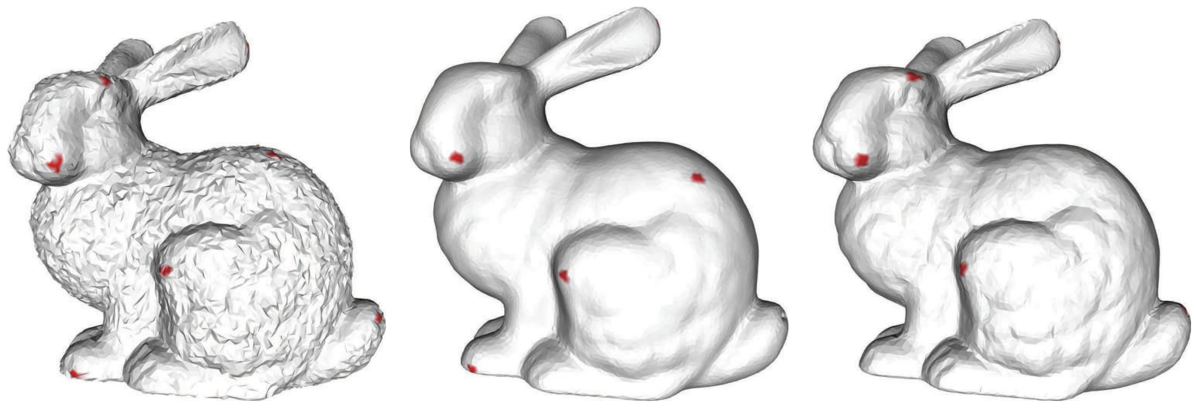

(a) Noise attack (b) Smoothing attack (c) Simplification attack

Figure 11: The prongs of the bunny model under the noise, smoothing, and simplification attacks.

Table 1: Evaluation of robustness against the cropping attack.

Model	Cropping ratio	Number of prongs	Number of correct prongs	BER
Bunny	70%	6	2	12.5%
	50%	8	4	0.00%
	20%	10	6	0.00%
Head	70%	4	3	0.00%
	50%	7	4	0.00%
	20%	11	7	0.00%
Hand	70%	7	5	0.00%
	50%	11	8	0.00%
	20%	15	13	0.00%

Table 2: Evaluation of robustness against the noise attack.

Model	Noise level	Hausdorff distance	BER
Bunny	0.1%	0.0116	12.50%
	0.3%	0.0123	31.87%
	0.5%	0.0125	38.12%
Head	0.1%	0.0088	11.25%
	0.3%	0.0219	23.75%
	0.5%	0.0433	42.50%
Hand	0.1%	0.4564	11.25%
	0.3%	1.4374	27.50%
	0.5%	2.4380	33.75%

Table 3: Evaluation of robustness against the smoothing attack.

Model	Smoothing iterations	Hausdorff distance	BER
Bunny	5	0.0043	18.75%
	10	0.0060	31.25%
	15	0.0086	43.75%
Head	5	0.0128	9.38%
	10	0.0229	28.13%
	15	0.0310	43.75%
Hand	5	0.2435	3.13%
	10	0.4042	18.75%
	15	0.5449	34.38%

prongs M increases if we decrease the value of N_1. However, if N_1 is smaller, the number of vertices which can be used for embedding decreases, so the watermarking capacity also decreases. In practice, we need to make compromise between robustness and watermarking capacity. We can also decrease N by discarding incorrect prongs. It can be seen from Figure 9 that the number of incorrect prongs are distributed near the cropping boundary. If the cropping boundary is known beforehand or can be estimated, we can discard prongs which are close to the cropping boundary in order to improve robustness. Normally, if the cropping boundary is simple, we can estimate it from experiences. For example, we can easily distinguish the cropping planes in Figure 9 even if we do not see the original models, because we have the experience that a rabbit should have legs and so forth. We can directly discard prongs close to the cropping boundary to improve performance. However, it is difficult to build algorithms that can automatically estimate cropping boundaries due to the complexity of the human visual systems (HVS).

5. Simulations on Other Attacks

Because of the good property of histogram-based approaches, our system is invariant to rotation, scaling, and translation (RST) attacks and vertex reordering. We then test the robustness of the system under four distortion attacks—noise, smoothing, simplification, and mixed (50% cropping + noise). The same as in previous sections, we use the bunny, head, and hand models for this test. We embed 32 bits into each model. The settings of N_1, N_2, and N_3 are the same as those in Section 2. The similarity between the original mesh and attacked mesh is measured by the Hausdorff distance, which is calculated by Metro [38].

Examples of the noise, smoothing, and simplification attacks are shown in Figure 11. Prongs are also shown on these attacked models. For noise and smoothing attacks, N_1, N_2, and N_3 are the same as in Section 2; for simplification attacks,

Table 4: Evaluation of robustness against the simplification attack.

Model	Noise level	Hausdorff distance	BER
Bunny	5%	0.0117	15.63%
	10%	0.0117	31.25%
	15%	0.0117	46.88%
Head	5%	0.0084	21.88%
	10%	0.0084	28.13%
	15%	0.0084	53.13%
Hand	5%	0.1923	12.50%
	10%	0.1923	18.75%
	15%	0.1923	37.50%

Table 5: Evaluation of robustness against the mixture of cropping and noise attack.

Model	Noise level	Hausdorff distance	BER
Bunny (50% cropped)	0.1%	0.0710	29.38%
Head (50% cropped)	0.1%	1.4359	31.87%
Hand (50% cropped)	0.1%	94.0903	24.37%

they need to be adjusted according to the reduction ratio. For example, if 5 percent of the vertices are vanished after simplification, the values of N_1, N_2, and N_3 should be reduced 5 percent accordingly. Compared with prongs in Figure 4(a), we can see that most prongs are preserved, while some prongs are missing and new prongs appear after these attacks.

The experimental results are shown in Tables 2, 3, 4, and 5. Generally, decoding errors of the system are caused by two aspects. The first is that attacks may change the positions of prongs, delete prongs, and introduce new prongs, so that we cannot obtain the same patches as not attacked. The second aspect is that vertex positions in patches can also be modified by attacks, which also produces decoding errors. Because of the interaction of these two factors, the proposed method is not as robust as the one which does not take prongs into consideration (such as the watermarking system proposed in [36]). This fact indicates that if robustness to the cropping attack is to be improved, we need to sacrifice robustness against other attacks.

The robustness against the noise attack is shown in Table 2. Gaussian noise is added to each of the vertices in the watermarked mesh. The mean of the Gaussian noise is zero, and its variance is proportional to the maximum vertex norm in the mesh. We define the noise level as the ratio of the noise variance to the maximum vertex norm in the mesh. In order to filter out the randomness, we repeat the noise-adding process five times and obtain the bit error rates in different noise levels. Three noise levels have been tested in experiments. It can be seen that the BERs increase when noise level increases.

Table 3 shows the performance of the watermarking scheme after the smoothing attack [39]. The relaxation parameter is set to 0.03 and three different pairs of iteration are applied. Because the head and hand models are smoother than the bunny model, they are more robust against smoothing attacks.

For simplification attacks, watermarked models are simplified by three reduction ratios, 5%, 10%, and 15%. The reduction ratio is defined as the percentages of vanished vertices to the total number of vertices. In order to obtain similar patches as the original mesh, N_1, N_2, and N_3 are adjusted according to the reduction ratio. We use R to represent the reduction ratio. Then the adjusted values of N_1, N_2, and N_3 can be obtained by

$$\begin{aligned} N_1' &= N_1(1-R), \\ N_2' &= N_2(1-R), \\ N_3' &= N_3(1-R). \end{aligned} \tag{23}$$

Here N_1', N_2', and N_3' are used for watermark decoding, and the results are obtained in Table 4.

The proposed method is not very robust against simplification because the modification of N_1, N_2, and N_3 is based on the assumption that vertices are evenly distributed on the mesh so that after simplification, approximately the same number of vertices vanishes in the neighborhood region of each prong. However, most meshes have relatively dense and sparse regions, which will introduce big errors for patch estimation after simplification. Another observation is that the Hausdorff distance between the simplified mesh and the original mesh does not change with different reduction ratios. This is because the Metro software interpolates vertices into the simplified mesh before calculation and thus obtains relatively similar Hausdorff distances.

Finally, robustness against a mixture of cropping and noise attacks is shown in Table 5. Here 50% vertices of the watermarked models are cropped; then 0.1% noise is added to the cropped models. We repeat the noise-adding process five times to obtain the BERs. It can be seen that the BERs for the mixed attacks are around 30% for these three models.

In summary, our method is robust against the cropping attack, as well as other attacks such as RST attacks, connectivity attacks, noise, smoothing, simplification, and mixtures of these attacks. Although robustness level against some attacks still needs to be improved, it is one of the first blind watermarking schemes which can withstand such a variety of attacks.

6. Conclusions

Although many research works have been carried out in the area of transmission 3D data through sensor networks, the security issue of transmission remains to be unsolved. In this context, it is important to develop systems for copyright protection and digital right management (DRM). In this paper, a blind watermarking algorithm is proposed to protect the transmission security of 3D polygonal meshes through sensor networks. Our method is based on selecting prominent feature vertices (prongs) on the mesh and then embedding the same watermark into their neighborhood regions. The embedding algorithm is based on modifying the distribution of vertex norms by using quadratic programming (QP). Decoding results are obtained by a majority voting scheme over neighborhood regions of these prongs. Assuming that cropping cannot remove all prongs, we can achieve robustness against the cropping attack both theoretically and experimentally. Experiments indicate that the proposed method is also robust against noise, smoothing, and mesh simplification. The proposed method has provided a solution for 3D polygonal watermarking which is potential to withstand a variety of attacks.

In this paper, watermark is retrieved by a majority voting scheme under the assumption that most prongs remain after the cropping attack. We also tested our method on other attacks such as noise, smoothing, simplification, and a mixture of these attacks. Experiments indicate that our method is robust against these attacks. Although robustness level against some attacks still needs to be improved, the simulation results demonstrate a blind watermarking scheme for 3D polygonal meshes which can resist a wide spectrum of attacks.

Conflict of Interests

The authors declare that they have no competing interests regarding the publication of this paper.

Acknowledgments

This work is supported by the Nature Science Foundation of China (Grant no. 61202400), the National Research Foundation for the Doctoral Program of Higher Education of China (Grant no. 20110101120053), the Fundamental Research Funds for the Central Universities (Grant no. 2011FZA5003), and the Nature Science Foundation of Zhejiang Province (Grant no. LQ12F02014). Part of this work has been done when Roland Hu was a postdoctoral researcher in Université catholique de Louvain, Belgium. Roland Hu is indebted to Benoit Macq, Patrice Rondao-Alface and Joachim Giard for their ideas and suggestions to finish this paper.

References

[1] J. L. Dugelay, A. Baskurt, and M. Daoudi, *3D Object Processing: Compression, Indexing and Watermarking*, Wiley, West Sussex, UK, 2008.

[2] P. Rondao-Alface and B. Macq, "From 3D mesh data hiding to blind and robust 3D shape watermarking: a survey," in *Transactions on Data Hiding and Multimedia Security II*, vol. 4499 of *Lecture Notes in Computer Science Transactions on Data Hiding and Multimedia Security*, pp. 91–115, Springer, Berlin, Germany, 2007.

[3] K. Wang, G. Lavoue, F. Denis, and A. Baskurt, "A comprehensive survey on three-dimensional mesh watermarking," *IEEE Transactions on Multimedia*, vol. 10, no. 8, pp. 1513–1527, 2008.

[4] L. Piegl and W. Tiller, *The NURBS Book*, Springer, Berlin, Germany, 1997.

[5] B. Adams, R. Keiser, M. Pauly, L. J. Guibas, M. Gross, and P. dutré, "Efficient raytracing of deforming point-sampled surfaces," in *Proceedings of the 26th Annual Conference of the European Association for Computer Graphics (EUROGRAPHICS '05)*, vol. 24, pp. 677–684, Dublin, Ireland, September 2005.

[6] S. Oomes, P. Snoeren, and T. Dijkstra, "3D shape representation: transforming polygons into voxels," in *Proceedings of the 1st International Conference on Scale-Space Theory in Computer Vision*, pp. 349–352, Utrecht, The Netherlands, 1997.

[7] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models through geometric and topological modifications," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 551–559, 1998.

[8] B.-L. Yeo and M. M. Yeung, "Watermarking 3D objects for verification," *IEEE Computer Graphics and Applications*, vol. 19, no. 1, pp. 36–45, 1999.

[9] O. Benedens and C. Busch, "Towards blind detection of robust watermarks in polygonal models," *Computer Graphics Forum*, vol. 19, no. 3, pp. 199–208, 2000.

[10] F. Cayre and B. Macq, "Data hiding on 3-D triangle meshes," *IEEE Transactions on Signal Processing*, vol. 51, no. 4, pp. 939–949, 2003.

[11] S. Zafeiriou, A. Tefas, and I. Pitas, "Blind robust watermarking schemes for copyright protection of 3D mesh objects," *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 5, pp. 596–607, 2005.

[12] A. G. Bors, "Watermarking mesh-based representations of 3-D objects using local moments," *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 687–701, 2006.

[13] J.-W. Cho, R. Prost, and H.-Y. Jung, "An oblivious watermarking for 3-D polygonal meshes using distribution of vertex norms," *IEEE Transactions on Signal Processing*, vol. 55, no. 1, pp. 142–155, 2007.

[14] S. Kanai, H. Date, and T. Kishinami, "Digital watermarking for 3D polygons using multiresolution wavelet decomposition," in *Proceedings of the 6th IFIP WG 5.2 International Workshop on Geometric Modeling: Fundamentals and Applications (GEO)-6*, pp. 296–307, Tokyo, Japan, 1998.

[15] E. Praun, H. Hoppe, and A. Finkelstein, "Robust mesh watermarking," in *Proceedings of the 26th International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pp. 69–76, Los Angles, Calif, USA, August 1999.

[16] K. Yin, Z. Pan, J. Shi, and D. Zhang, "Robust mesh watermarking based on multiresolution processing," *Computers & Graphics*, vol. 25, no. 3, pp. 409–420, 2001.

[17] R. Ohbuchi, A. Mukaiyama, and S. Takeaways, "A frequency-domain approach to watermarking 3D shapes," in *Proceedings of Annual Conference of the European Association for Computer Graphics (EUROGRAPHICS '02)*, vol. 21, pp. 373–382, Saarbrücken, Germany, 2002.

[18] F. Cayre, P. Rondao-Alface, F. Schmitt, B. Macq, and H. Maître, "Application of spectral decomposition to compression and watermarking of 3D triangle mesh geometry," *Signal Processing*, vol. 18, no. 4, pp. 309–319, 2003.

[19] J. Wu and L. Kobbelt, "Efficient spectral watermarking of large meshes with orthogonal basis functions," *The Visual Computer*, vol. 21, no. 8–10, pp. 848–857, 2005.

[20] R. Ohbuchi, H. Masuda, and M. Aono, "Watermarking three-dimensional polygonal models," in *Proceedings of the 5th ACM International Conference on Multimedia*, pp. 261–272, Seattle, Wash, USA, November 1997.

[21] Z. Yu, H. H. S. Ip, and L. F. Kwok, "A robust watermarking scheme for 3D triangular mesh models," *Pattern Recognition*, vol. 36, no. 11, pp. 2603–2614, 2003.

[22] Q. Liu and L. Yang, "Watermarking of 3D polygonal meshes based on feature points," in *Proceedings of the 2nd IEEE Conference on Industrial Electronics and Applications (ICIEA '07)*, pp. 1837–1841, Harbin, China, May 2007.

[23] C. Roudet, F. Dupont, and A. Baskurt, "Multiresolution mesh segmentation based on surface roughness and wavelet analysis," in *Visual Communications and Image Processing (VCIP '07)*, vol. 6508 of *Proceedings of SPIE*, pp. 65082E.1–65082E.12, San Jose, Calif, USA, February 2007.

[24] P. Rondao-Alface, M. D. Craene, and B. Macq, "Three-dimensional image quality measurement for the benchmarking of 3D watermarking schemes," in *The 17th Annual Symposium on Electronic Imaging—Security, Steganography, and Watermarking of Multimedia Contents VII*, vol. 5681 of *Proceedings of SPIE*, pp. 230–240, San Jose, Calif, USA, January 2005.

[25] P. Rondao-Alface, B. Macq, and F. Cayre, "Blind and robust watermarking of 3D models: how to withstand the cropping attack?" in *Proceedings of the 14th IEEE International Conference on Image Processing (ICIP '07)*, pp. V465–V468, San Antonio, Tex, USA, September 2007.

[26] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *The Visual Computer*, vol. 21, no. 8–10, pp. 649–658, 2005.

[27] S. Valette, I. Kompatsiaris, and M. Strintzis, "A polygonal mesh partitioning algorithm based on protrusion conquest for perceptual 3D shape description," in *Proceedings of the Workshop Towards Semantic Virtual Environments (SVE '05)*, pp. 1–9, Villars, Switzerland, 2005.

[28] C. Cristina, L. J. Rodríguez-Aragón, and E. Cabello, "Automatic 3D face feature plints extraction with spin images," in *Proceedings of the 3rd International Conference on Image Analysis and Recognition (ICIAR '06)*, pp. 317–328, Povoa de Varzim, Portugal, 2006.

[29] D. L. Page, A. F. Koschan, S. R. Sukumar, B. Roui-Abidi, and M. A. Abidi, "Shape analysis algorithm based on information theory," in *Proceedings of the International Conference on Image Processing (ICIP '03)*, pp. 229–232, Barcelona, Spain, September 2003.

[30] R. Yagel, D. Cohen, and A. Kaufman, "Normal estimation in 3 D discrete space," *The Visual Computer*, vol. 8, no. 5-6, pp. 278–291, 1992.

[31] P. Tellier and I. Debled-rennesson, "3D discrete normal vectors," in *Proceedings of th 8th International Conference on Discrete Geometry for Computer Imagery (DGCI '99)*, pp. 447–457, Marne-la-Vallée, France, 1999.

[32] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[33] R. Kimmel and J. A. Sethian, "Computing geodesic paths on manifolds," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 95, no. 15, pp. 8431–8435, 1998.

[34] J. Giard and B. Macq, "From mesh parameterization to geodesic distance estimation," in *Proceedings of the 25th European Workshop on Computational Geometry (EuroCG '09)*, pp. 97–100, Brussels, Belgium, 2009.

[35] J. Giard and B. Macq, "Fast geodesic distance approximation using mesh decimation and front propagation," in *Image Processing: Algorithms and Systems VII*, vol. 7245 of *Proceedings of SPIE*, p. 72450B, San Jose, Calif, USA, January 2009.

[36] R. Hu, P. Rondao-Alface, and B. Macq, "Constrained optimisation of 3D polygonal mesh watermarking by quadratic programming," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '09)*, pp. 1501–1504, Taipei, Taiwan, April 2009.

[37] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[38] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.

[39] D. A. Field, "Laplacian smoothing and Delaunay triangulations," *Communications in Applied Numerical Methods*, vol. 4, no. 6, pp. 709–712, 1988.