International Journal of Innovative Computing, Information and Control
Volume **8**, Number **5(A)**, May **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 3299–3314

A THREE DIMENSIONAL MODEL WATERMARKING ALGORITHM IN FREQUENCY DOMAIN BASED ON THE NORMALIZATION OF HOST MODELS

Chen-Chung Liu[1], Jiun-You Chen[1], Pei-Chung Chung[2]
Shyr-Shen Yu[2] and Ta-Shan Tsui[2]

[1]Department of Electronic Engineering
National Chin-Yi University of Technology
No. 35, Lane 215, Sec. 1, Chung-Shan Rd., Taiping, Taichung 411, Taiwan
ccl@ncut.edu.tw; gn00139173@yahoo.com.tw

[2]Department of Computer Science and Engineering
National Chung-Hsing University
No. 250, Kuo kuang Rd., Taichung 402, Taiwan
iam.abaw@gmail.com; pyu@nchu.edu.tw; tstsui@amath.nchu.edu.tw

Received January 2011; revised May 2011

Abstract. *At present, even though there is a robust 3D models watermarking scheme, its robustness against cropping and simplifying attacks is still weak. The main goal of this paper is to configure a blind 3D models watermarking algorithm with high robustness against cropping and simplifying attacks. The high robust blind watermarking algorithm for 3D models is synthesized by combining the 3D model normalization scheme, vertices clustering scheme and discrete cosine transform (DCT). Experimental results show that the proposed algorithm is high performance and has the following advantages: (i) the watermarked 3D model possesses excellent imperceptibility without noticeable degradation; (ii) both the embedding capacity and the extracted watermark's validality are better than other relative schemes; (iii) the security is strong enough because the watermark is embedded in DCT domain; (iv) the robustness against affine transformations is 100%, and the robustness withstanding the cropping, simplifying, reordering, and smoothing attacks is also high; (v) the original 3D model and watermark are not needed during the watermark extraction process.*

Keywords: Watermarking, 3D model, Discrete cosine transform (DCT)

1. **Introduction.** Recently, due to the speedy development of internet and digital media information processing, digital information is easy to transmit and duplicate unauthorized reproduction to become a serious problem [1]. There is an urgent demand for techniques to protect the copyright of the original digital data and prevent unauthorized duplication or tampering. A lot of researches have been carried out to protect the copyright protection of image, video and audio. These researches show that digital watermarking, a technique designed to hide information in a certain type of digital data, is the end-step in information security and protects the copyright of owner [2].

Embedded watermarks can be used to enforce copyright, data authentication or add information to the data. In the last decades, most of the research on watermarking has concentrated on audio signals, images or video sequences [3]. The watermarking algorithms for 3D models are few because the watermarking technique for 3D model has many difficulties for the following reasons: (i) only a small number of data (vertices) are available for watermark embedding; (ii) there is neither unique representa tion nor implicit ordering for 3D models and (iii) no robust transformation field could be used

to embed watermark [4]. In recent years, 3D graphic models, such as VRML, MPEG-4 and 3D geometrical CAD, have become very popular leading the development of 3D watermarking algorithms to protect the copyright of 3D graphic models [5]. Ohbuchi [6] proposed several watermarking algorithms for 3D models: triangle similarity quadruple (TSQ) embedding algorithm, tetrahedral volume ratio (TVR) embedding algorithm, and mesh density pattern embedding algorithm. However, these algorithms are not sufficiently robust against attacks. Yin et al. [7] proposed a new mesh watermarking scheme for triangular meshes. In their scheme watermark information is embedded into a suitable coarser mesh which consists of the low-frequency components. The scheme is not robust against cropping attacks. Wang et al. [8] proposed a fragile watermarking algorithm for 3D models in spatial domain. In their algorithm, a reversible quantization index modulation embedding scheme was conducted to employ the principal component analysis and modulate the distance between vertex and the mass center of the host model to increase the data-hiding and a spectrum technique was employed to enhance the robustness. Although this algorithm is robust against the vertex reordering and affine transformations, it is not robust against cropping attacks.

The discrete cosine transform (DCT), introduced by Ahmed et al. in 1974, can decorrelate signal data to less redundancy and provide a robust approximation of the statistically ideal Karhunen-Loeve transform [3,9]. DCT is widely used for signal processing recently due to its lower computational complexity and can withstand image operations quite well. Moreover, invisibility constraints are easier to achieve while working in DCT domain [3,9]. In this paper, the security and invisibility of the proposed watermarking scheme are enhanced by embedding the watermark into the DCT coefficients of host 3D models vertices. In addition, the normalization of model can keep the model in a standard state to maintain the model data when the model is attacked [10]. In the proposed scheme, watermark is embedded in the normalized model and the normalization information is then embedded into the watermarked model. On the watermark extracting stage, the normalization information is first detected from the attacked model, then normalizes the attacked model by using the detected the normalization information, and finally extracts the watermark from the normalized model. With the normalization phase, the robustness and validality of the proposed watermarking algorithm are powered.

In this proposal, we utilize DCT based on the normalization of host model to develop a robust watermarking scheme for 3D models. To explore the utility and demonstrate the efficiency of the proposed scheme, simulations under various conditions are conducted. The experiment results show that our proposed scheme is a blind and robust watermarking scheme for 3D model. The remainder of this paper is organized as follows. In Section 2, the watermark embedding algorithm is presented. The watermark extraction process from the host 3D model is illustrated in Section 3. Experimental results are presented in Section 4. Finally, Section 5 concludes this paper.

2. **Watermarking Embedding.**

A robust watermarking scheme for 3D models must be extremely secure without reducing the visual quality of the cover 3D model and embedded watermarks must be robust to against attacks. In order to construct a superior watermarking scheme for 3D model, several schemes are used in this paper to achieve the goal. The overall watermark embedding process is shown in Figure 1.

The grayscale image watermark W is first arranged into a grayscale pixel sequence $\{S\}$. The watermark sequence is then hashed into three hashed watermark sequences by chaotic mechanism (CM) using three pseudo random sequences (PN) generated by three private keys to enhance the security [3]. On the other hand, the host 3D model X is normalized to a standard state by a series of translation, rotation and scaling operations.

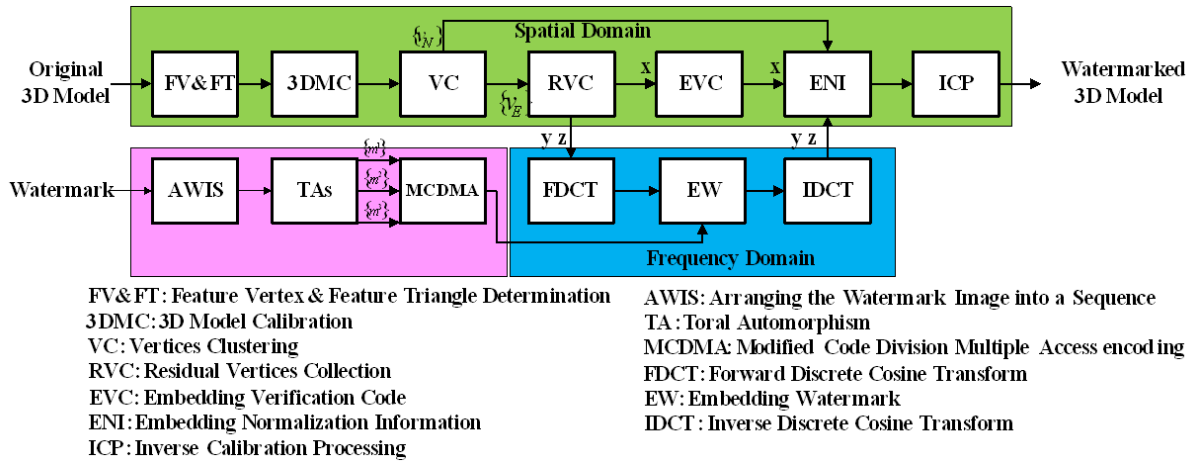

FIGURE 1. The flow chart of the proposed watermark embedding scheme

The vertices of normalized model are classified into normalizing vertices set $\{v^N\}$ and watermarking set $\{v^W\}$; the normalizing vertices are used to embed normalization information and the watermarking vertices are used to embed watermark. The x-component of a watermarking vertex is embedded the verification code for indicating whether the vertex is a watermarked vertex and which pixel of watermark is embedded at this vertex. The y-component and z-component of watermarking vertices are clustered and are transformed into frequency domain by a DCT for watermark embedding, and are transformed return to the original spatial domain by the inverse discrete cosine transform (IDCT) when they have been embedded watermark. Finally, the normalization information of the watermarked model is embedded into each vertex of the normalizing vertices set, respectively. The details of each block of our watermark embedding algorithm are described in the following subsections.

2.1. **Original 3D model calibrating [10].** The proposed watermarking algorithm for 3D model needs to embed the host model's normalization information into the host model itself. There must have a selecting rule to select suitable triangles for embedding the host model's normalization information, otherwise the embedded normalization information should not be extracted correctly and the attacked model should not be normalized. For achieving the goal, the feature vertex, feature edge and the feature triangle of the host 3D model have to be determined before host 3D model calibrating. The proposed algorithm then translates, rotates and scales the original 3D model to calibrate the original 3D model's pose.

First, the presented algorithm assigns a vertex that has maximum degree and maximum total length of connecting edges as the feature vertex, A. Edge $\overline{AB}$ is called the feature edge when it is the longest edge of the connecting edges of feature vertex A, and vertex B is called the principal reference vertex. Triangle ΔABC is defined as the feature triangle if the triangle ΔABC has the maximum area among these triangular meshes that use $\overline{AB}$ as their common side, and vertex C is called as the reference vertex.

In order to get the actual normalization information, the original 3D model has to be calibrated to a standard state by translating, rotating and scaling operations. In standard state, the feature vertex is coinciding with the origin of the Cartesian coordinate system, the feature edge vector $\overrightarrow{AB}$ is coinciding with the positive x-axis, the feature triangle ΔABC is completely lying on the xy-plane, and the coordinates of each vertex are neither larger than one nor less than minus one. Figure 2 shows the calibrating for 3D model Bunny.

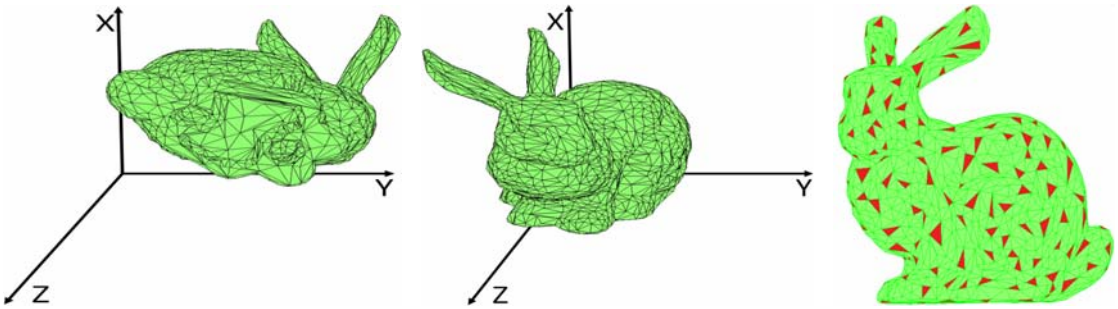

FIGURE 2. The visions of 3D model Bunny; (a) before normalization, (b) after normalization and (c) the feature triangular (red triangular) meshes

2.2. **Vertices clustering.** For enhancing the robustness and the validation of the watermark, the proposed scheme not only embeds the watermark into the model but also embeds the host model's normalization information into the host model itself. There must have a selecting rule to select some vertices for embedding the host model's normalization information, otherwise the embedded normalization information should not be extracted correctly and the attacked model should not be normalized to accurately extract the embedded watermark. For achieving the goal, ten percentages of the vertices are selected as the set of the normalization information embedding vertices (NIEV), $\{v^N\}$, that are used for normalization information embedding. One triangular mesh is selected from the neighbor triangular meshes of a normalization information embedding vertex as the feature triangle of the normalization information embedding vertex, and the angles of the feature triangle are used to embed the normalization information. The feature triangular meshes of model Bunny is shown in Figure 2(c). The other vertices are then clustered as the set of the watermark embedding vertices (WEV), $\{v^W\}$, that are used for watermark embedding.

2.3. **Watermark embedding.** To enhance the security and the robust of embedded watermarks, the proposed algorithm uses the arranging watermark scheme to convert the watermark (gray-level image of size $M * N$) into a sequence, $\{s\}_{1\times MN}$, where each element s is a pixel value ($0 \sim 255$) represented with 8 binary bits. The watermark sequence of pixels is hashed into three hashed watermark sequences, $\{m^1\}_{1\times MN}$, $\{m^2\}_{1\times MN}$ and $\{m^3\}_{1\times MN}$ by 3 chaotic mechanisms (CMs), respectively [3]. Figure 3 shows an example of a grayscale watermark with size 6×6 pixels, corresponding sequence $\{s\}_{1\times 36}$, and three hashed watermark sequences $\{m^1\}_{1\times 36}$, $\{m^2\}_{1\times 36}$ and $\{m^3\}_{1\times 36}$.

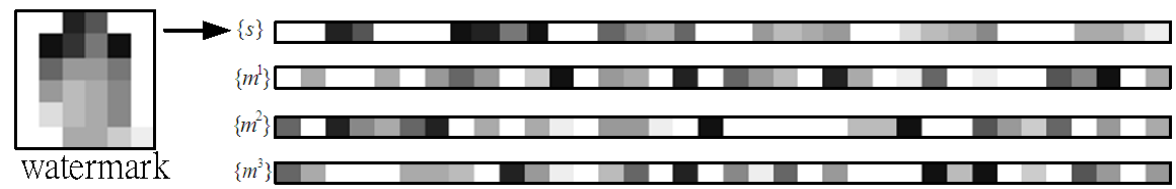

FIGURE 3. An example of a grayscale watermark and its hashed watermark sequences

In this paper, a fast pseudorandom number traversing method that the relation between the pixel-sequences before and after permutation is presented in the following formulas:

$$m^i(k) = s(k'),\ i = 1, 2, 3,\ 1 \le k, k' \le M \times N \tag{1}$$

$$k = permutation(k') \tag{2}$$

The three hashed watermark sequences are encoded into 4-digit decimal watermark units, $w_Y(k)$, $w_Z(k)$, $1 \le k \le M \times N$ by the following equations:

$$m^i = \prod_{j=1}^{8} m_j^i (Radix\ 2) = m_8^i m_7^i m_6^i m_5^i m_4^i m_3^i m_2^i m_1^i\ radix\ 2,\ i = 1, 2, 3 \text{ and } m_j^i \in \{0, 1\} \tag{3}$$

$$w_C(k) = \sum\nolimits_{i=1}^{3} \sum\nolimits_{j=1}^{8} m_j^i(k) \times 2^{8i+j-9} \tag{4}$$

$$w_Y(k) = fix(w_C(k) \times 10^{-4}) = a_7 a_6 a_5 a_4(k)\ radix\ 10,\ 1 \le k \le M \times N \tag{5}$$

$$w_Z(k) = \mathrm{mod}(w_C(k),\ 10^4) = a_3 a_2 a_1 a_0(k)\ radix\ 10,\ 1 \le k \le M \times N. \tag{6}$$

The overall watermark embedding is completed with the following steps:

Step 1: Randomly select a vertex from the set of the watermark embedding vertices, $\{v^W\}$, and delete the selected vertex from $\{v^W\}$.

Step 2: Select $n^2 - 1$ nearest vertices of the vertex selected in Step 1 from $\{v^W\}$, delete these $n^2 - 1$ selected vertices from $\{v^W\}$. These n^2 vertices are arranged into the gth embedding-vertex sequence (EVS), $\{v^W(g)\}$, and each vertex is given an embedding number EN, respectively. Here, n^2 is the length of each embedding-vertex sequence with $n = 4, 8, 16, 32$, g is the embedding-vertex sequence number with $g = 1, 2, \ldots, fix\,(length\,(\{v^W\})/n^2)$ and $EN = 1, 2, \ldots, length\,(\{v^W\})$.

Step 3: Embed the corresponding embedding number of each vertex into the x-coordinate of each vertex.

Step 4: Transfer the y-coordinate sequence of $\{v^W(g)\}$ into DCT domain by forward discrete cosine transformation (FDCT), and embed the 4-digit decimal watermark units $w_Y(k)$ into each corresponding DCT element with the following equations:

$$Y_{WDj}^g = \begin{cases} Y_{Dj}^g - \mathrm{mod}(fix(abs(Y_{Dj}^g) \times 10^9) \times 10^{-4}, 1) \times 10^{-5} + w_Y(k) \times 10^{-9}, \\ \qquad\qquad Y_{Dj}^g \ge 0, j = 1, 2, \ldots, 16 \\ Y_{Dj}^g + \mathrm{mod}(fix(abs(Y_{Dj}^g) \times 10^9) \times 10^{-4}, 1) \times 10^{-5} - w_Y(k) \times 10^{-9}, \\ \qquad\qquad Y_{Dj}^g < 0, j = 1, 2, \ldots, 16 \end{cases} \tag{7}$$

and transfer the water marked DCT sequence $\{Y_{WD}^g\}$ into space domain by inverse discrete cosine transformation (IDCT).

Step 5: Transfer the z-coordinate sequence of $\{v^W(g)\}$ into DCT domain by FDCT, and embed the 4-digit decimal watermark units $w_Z(k)$ into each corresponding DCT element by the following equations:

$$Z_{WDj}^g = \begin{cases} Z_{Dj}^g - \mathrm{mod}(fix(abs(Z_{Dj}^g) \times 10^9) \times 10^{-4}, 1) \times 10^{-5} + w_Z(k) \times 10^{-9}, \\ \qquad\qquad Z_{Dj}^g \ge 0, j = 1, 2, \ldots, 16 \\ Z_{Dj}^g + \mathrm{mod}(fix(abs(Z_{Dj}^g) \times 10^9) \times 10^{-4}, 1) \times 10^{-5} - w_Z(k) \times 10^{-9}, \\ \qquad\qquad Z_{Dj}^g < 0, j = 1, 2, \ldots, 16 \end{cases} \tag{8}$$

and transfer the water marked DCT sequence $\{Z_{WD}^g\}$ into space domain by IDCT.

Step 6: Repeat previous steps until the length of $\{v^W\}$ is less than n^2.

2.4. **Normalization information embedding [10].** The normalization information is used to normalize a 3D model that may be attacked, and the normalization information embedding is processed by the following steps:

Step 1: For a normalization vertex A, determine the normalization candidate triangle ΔABC, which has minimum area among those triangular meshes that share vertex A as their common vertex, and $\overline{AB} > \overline{AC}$.

Step 2: Take the normalization candidate triangle ΔABC as an embedding triangle, and then transfer the x-coordinate, y-coordinate and z-coordinate of the vertex $B(x_B, y_B, z_B)$ into digit strings consisting of 6 decimal digits, respectively, with the following formulas:

$$x_{TB} = fix\left(x_B \times 10^6\right) = \sum\nolimits_{i=0}^{5} \left(x_i \times 10^i\right) \tag{9}$$

$$y_{TB} = fix\left(y_B \times 10^6\right) = \sum\nolimits_{i=0}^{5} \left(y_i \times 10^i\right) \tag{10}$$

$$z_{TB} = fix\left(z_B \times 10^6\right) = \sum\nolimits_{i=0}^{5} \left(z_i \times 10^i\right). \tag{11}$$

where x_B, y_B and z_B are the x-coordinate, y-coordinate and z-coordinate of vertex B, respectively. x_{TB}, y_{TB} and z_{TB} are the transferred x-coordinate, y-coordinate and z-coordinate of vertex B, respectively.

Step 3: Embed x_{TB} and a one-digit decimal characteristic verification code (CVC) in $\angle B$ by translating the normalization vertex A along vector $\overrightarrow{AC}$. Here CVC is used to record the characteristic of the coordinates of vertex B and is defined as follows:

$$c_\eta = \begin{cases} 1, & \eta_B = \pm 1 \\ 0, & \text{otherwise} \end{cases}, \quad \eta \in \{x, y, z\} \tag{12}$$

$$CVC = 4 \times c_z + 2 \times c_y + c_x \tag{13}$$

Step 4: Embed y_{TB} and a one-digit decimal sign verification code (SVC) in $\angle C$ by translating the normalization vertex A along vector $\overrightarrow{AB}$. Here SVC is used to record the sign of the coordinates of vertex B and is determined by the following formulas:

$$s_\eta = \begin{cases} 1, & \eta_B \geq 0 \\ 0, & \text{otherwise} \end{cases}, \quad \eta \in \{x, y, z\} \tag{14}$$

$$SVC = 4 \times s_z + 2 \times s_y + s_x \tag{15}$$

Step 5: Determine the geometric center G of all the vertices connected to vertex A.

Step 6: Embed z_{TB} and a one-digit decimal feature verification code (FVC) in the angle between the triangle ΔABC and the triangle ΔABG by rotating the vertex A about the line segment $\overline{BC}$, where FVC is used to record whether the triangle ΔABC is embedded or not, and FVC is determined by the following formulas:

$$M = \mathrm{mod}\left(\left(SVC + CVC + \sum_{i=0}^{5} (x_i + y_i + z_i)\right), 10\right) \tag{16}$$

$$FVC = \begin{cases} 10 - M, & M \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

3. **Watermark Extraction.** The proposed watermarking algorithm is blind, so in the watermark detection stage, it requires only the watermarked model for recovering the embedded watermark. In the detection stage, each embedded watermark pixel is retrieved from the watermarked model. The watermark extraction is the inverse process of

a watermark embedding process; most of the retrieval steps are identical to the embedding process. The proposed watermark recovering consists of model normalization and watermark extracting phases, which are illustrated in the following two subsections.

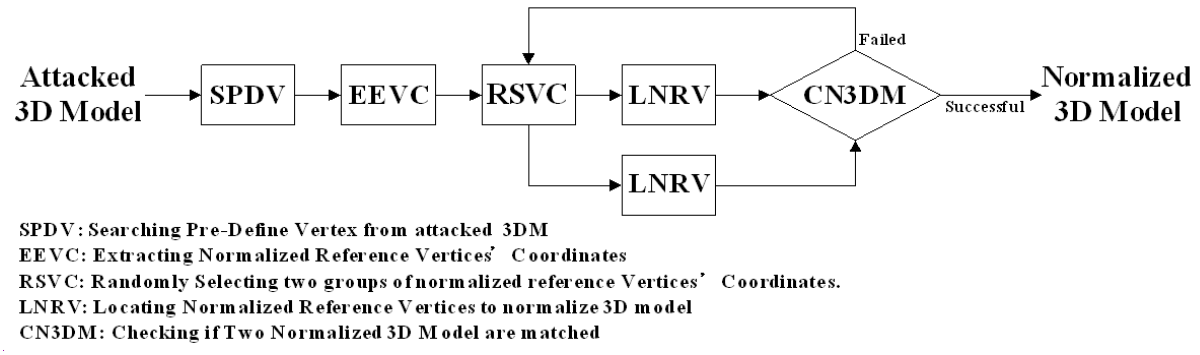

FIGURE 4. The flow chart of the normalization of an attacked model

3.1. **Normalization of attacked 3D model [10].** The procedure of an attacked 3D model normalizing is used to correctly extract the real normalization information embedded in the survived parts of an attacked 3D model by verifying the extracted verification codes, and using them to normalize the attacked 3D model so that all the survived vertices of the attacked 3D model are placed in their original positions. The overall normalization process for attacked 3D models is shown in Figure 4, and is completed with the following steps:

Step 1: Let a vertex of an attacked 3D model as a normalization vertex A to determine the corresponding embedding triangle ΔABC of the normalization vertex A.

Step 2: Verify the verification codes taken from angles $\angle A$, $\angle B$, and the angle between the triangle ΔABC and the triangle ΔABD, respectively, for each candidate triangle. If all the three verification codes CVC, SVC and FVC are correct, then the current coordinate (x_B, y_B, z_B) and the transferred coordinate (x_{TB}, y_{TB}, z_{TB}) of embedding vertex B are extracted, respectively.

Step 3: Repeat previous two steps three times to extract the current coordinates and the transferred coordinates of the distinct three embedded vertices E, F and G.

Step 4: Use the current coordinates extracted in Step 3 to construct the current triangle $\Delta E_c F_c G_c$, and use the corresponding transferred coordinates to construct the transferred triangle $\Delta E_t F_t G_t$, respectively.

Step 5: Check if the equations $\overline{E_c F_c}/\overline{E_t F_t} = \overline{F_c G_c}/\overline{F_t G_t} = \overline{G_c E_c}/\overline{G_t E_t} = S$ are true or false. If they are false, the proposed algorithm will repeat previous steps until three distinct real embedded vertices are found.

Step 6: Find calibrating parameter vector $(\overrightarrow{T_t}, \alpha_t, \beta_t, \gamma_t)$ for the transferred triangle $\Delta E_t F_t G_t$. Then, calibrate the transferred triangle with the calibrating parameter vector so that vertex E_t coincides with the original point of the global coordinate system, vector $\overrightarrow{E_t F_t}$ coincides with the positive x-axis, and $\Delta E_t F_t G_t$ lays on the first quadrant of the xy-plane ($x \geq 0$, $y \geq 0$ and $z = 0$).

Step 7: Find calibrating parameter vector $(\overrightarrow{T_c}, \alpha_c, \beta_c, \gamma_c)$ for the current triangle $\Delta E_c F_c G_c$. Then, calibrate the current triangle with the calibrating parameter vector so that vertex E_c coincides with the original point of the coordinate system, vector $\overrightarrow{E_c F_c}$ coincides with the positive x-axis, and $\Delta E_c F_c G_c$ lays on the first quadrant of the xy-plane. Finally, scale it with scale parameter s found in Step 5 by dividing each coordinates of each vertex of $\Delta E_c F_c G_c$ with s.

Step 8: Scale and calibrate the attacked 3D model with the same scaling parameter and calibrating parameter vector used in Step 7 to obtain a preliminary calibrated 3D model. Finally, the preliminary calibrated 3D model is calibrated with the calibrating parameter vector $(-\overrightarrow{T_t}, -\alpha_t, -\beta_t, -\gamma_t)$ to obtain the corresponding normalization model of the attacked 3D model.

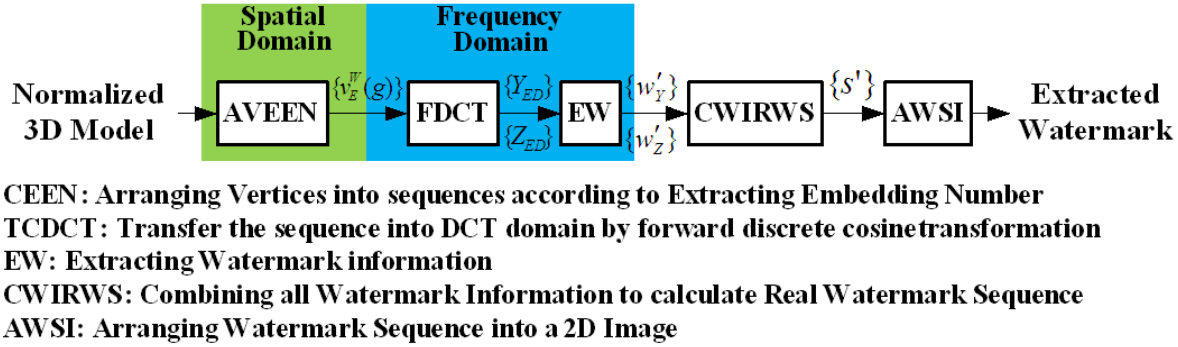

FIGURE 5. The flow chart of watermark extracting

3.2. **Watermark extraction.** The overall watermark extraction process from a normalized model is shown in Figure 5, and is completed with the following steps:

Step 1: Traverse each vertex of the input normalized model to extract the embedding number of each watermark-embedded vertex, and assign each embedded vertex to its own embedding-vertex sequence $\{v_E^W(g)\}$ according its embedding number.

Step 2: Transfer the y-coordinate sequence of each $\{v_E^W(g)\}$ into DCT domain y-coordinate sequence $\{Y_{ED}(g)\}$ by forward-DCT (FDCT), and extract the watermark units from $\{Y_{ED}(g)\}$ to form extracted-watermark-unit sequence $\{w'_Y\}$.

Step 3: Transfer the z-coordinate sequence of each $\{v_E^W(g)\}$ into DCT domain z-coordinate sequence $\{Z_{ED}(g)\}$ by FDCT, and extract the watermark units from $\{Z_{ED}(g)\}$ to form extracted- watermark-unit sequence $\{w'_Z\}$.

Step 4: Combine all extracted-watermark-unit sequences $\{w'_Y\}$ and $\{w'_Z\}$ to form the chaotic extracted watermark sequences $\{m_e^1\}$, $\{m_e^2\}$ and $\{m_e^3\}$.

Step 5: Use three inverse chaotic mechanisms to return the three chaotic extracted watermark sequences into the real extracted watermark sequence s'.

Step 6: Arrange the real extracted watermark sequence s' into the extracted two dimensional watermark.

4. **Experimental Results.** In this section, some experimental results under various conditions are shown to explore the utility and efficiency of the proposed scheme. To confirm the experimental results, each experiment is repeated 5 times and the results are reported as an average of these 5 evaluations. In experiments, the vertex coordinates of the original model are represented with IEEE 754 single precision decimal 32 format. For example, the x-coordinate of a vertex occupies 32 bits with 23-bit significant precision (seven decimal digits). In this paper, the grayscale portrait image of an author of this paper with size $64 * 64$ pixels is used as the watermark to avoid to be copied by others. And, the host 3D models used in experiments are Angelfish with 38620 meshes and 19724 vertices, Teapot with 57600 meshes and 28922 vertices, Lion with 32098 meshes and 17352 vertices, and Dolphin with 61836 meshes and 32828 vertices. Moreover, although there is no unified criterion to decide which scheme is the best, robust watermarking schemes are usually characterized by Security, Invisibility, Validation and Robustness [11].

4.1. **Security.** An embedded watermark should be secret and undetectable by an unauthorized user in general; however, it must be correctly accessible to authorized users. This requirement is regarded as the security of a watermarking system and it is generally achieved by utilizing cryptographic keys. The proposed algorithm adopts three strategies to enhance the security of the watermarking system: (i) Using an author's portrait grayscale image as the watermark; (ii) Before embedding, the watermark is randomly permuted using 3 pseudorandom number sequences generated from 3 stego-keys to generate 3 hushed watermark sequences for embedding; (iii) The 3 hushed watermark sequences are encoded into watermark units; (iv) The main watermarking vertices are randomly selected using a pseudorandom number sequence generated from a stego-key, therefore, watermark pixels spread in all regions of the host model chaotically.

4.2. **Invisibility.** In order to demonstrate the fact that the watermark embedded 3D model possesses excellent invisibility without noticeable degradation when comparing to the original 3D models, they are shown in Figure 6. Column 2 of Figure 6 shows the visualizations of original 3D models before the watermark embedding, the visual effects and the corresponding signal-to-noise-ratio (*SNR*) of watermarked models with varying EVS lengths are shown from column 3 to column 6, respectively. The comparison illustrates that the visual differences between the original 3D models and their corresponding watermark embedded 3D models are almost imperceptible.

EVS length	0×0	4×4	8×8	16×16	32×32
Angelfish		*SNR*=120.67	*SNR* =120.67	*SNR*=120.70	*SNR*=120.70
Teapot		*SNR*=124.50	*SNR*=124.54	*SNR*=124.62	*SNR*=124.61
Lion		*SNR*=120.59	*SNR*=120.60	*SNR*=120.64	*SNR*=120.84
Dolphin		*SNR*=126.40	*SNR*=126.41	*SNR*=126.43	*SNR*=126.50
Here, the SNR in blue is the highest and SNR in red is the lowest.					

FIGURE 6. Invisibitis of 3D models after watermark embedding with various EVS lengths

In quantified measurement, the signal-to-noise-ratio (*SNR*), the correlation coefficient (*CC*), and the modified Hausdorff distance (*MHD*) [11] are employed to measure the degree of invisibility of embedded models. *SNR* and *MHD* are determined as follows:

(i) To evaluate the *SNR* of a watermarked 3D model, the following formula is used:

$$SNR = 10\log_{10}\left(\sum_{i=0}^{N-1}\left(x_i^2+y_i^2+z_i^2\right)\Big/\sum_{i=0}^{N-1}\left((x_i'-x_i)^2+(y_i'-y_i)^2+(z_i'-z_i)^2\right)\right), \tag{18}$$

where (x_i, y_i, z_i) and (x_i', y_i', z_i') are the coordinates of vertex v_i before and after the watermark embedding, respectively. The larger SNR indicates more invisible positional changes during the watermark embedding.

(ii) The definition of correlation coefficient between the original and the watermarked 3D models is described by the following equation:

$$CC = \frac{\sum_{i=0}^{N-1} \left((x_i - x_c)(x_i' - x_c') + (y_i - y_c)(y_i' - y_c') + (z_i - z_c)(z_i' - z_c') \right)}{\sqrt{\sum_{i=0}^{N-1} \left((x_i - x_c)^2 + (y_i - y_c)^2 + (z_i - z_c)^2 \right)} \sqrt{\sum_{i=0}^{N-1} \left((x_i' - x_c')^2 + (y_i' - y_c')^2 + (z_i' - z_c')^2 \right)}}. \tag{19}$$

where (x_i, y_i, z_i) and (x_i', y_i', z_i') are the coordinates of vertex ν_i before and after the watermark embedding, respectively. Moreover, (x_c, y_c, z_c) and (x_c', y_c', z_c') are the coordinates of the geometric center of the model before and after watermark embedding, respectively. Larger CC indicates that the embedded watermark is more invisible.

(iii) The modified Hausdorff distance (MHD) is utilized to assess the shape similarity of the watermarked model X^w to the original model X, which is defined as:

$$MHD(X^w, X) = \sum\nolimits_{v \in X} d(v, X^w)/|X|, \tag{20}$$

where $|X|$ is the number of vertices of the original model, $d(v, X^w)$ is the distance from vertex v of the original model to the watermarked model which is defined as:

$$d(v, X^w) = \min\{\text{distance}(v, v^w) | v^w \in X^w\}, \tag{21}$$

where distance (v, v^w) is the Euclidean distance between vertices v and v^w.

The evaluated results for embedding a 64×64 pixels grayscale image into models under various EVS lengths are arranged in Figure 6 and Table 1 to show the models' embedding capacities and the watermarks' invisibilities.

TABLE 1. Models' embedding capacities, extracted watermarks' invisibities and validations (bold indicates the best and italic indicates the worst)

Model	**Watermark EVS length**	**Embedding rate (%)**	**Invisibility**			**Validation**			
			SNR	***CC***	***MHD***	***NC***	***SCC***	***PER***	***BER***
Angelfish	4×4	99.9645	120.6720	1	7.3225 e-07	1	1	0	0
	8×8	99.8833	120.6745	1	7.2672 e-07	1	1	0	0
	16×16	99.5589	120.7045	1	7.2129 e-07	1	1	0	0
	32×32	98.2610	120.7076	1	7.2083 e-07	1	1	0	0
Teapot	4×4	99.9550	124.5037	1	7.1130 e-07	1	1	0	0
	8×8	99.8444	124.5450	1	7.1146 e-07	1	1	0	0
	16×16	99.4018	124.6226	1	6.9798 e-07	1	1	0	0
	32×32	98.5167	124.6142	1	6.8869 e-07	1	1	0	0
Lion	4×4	**100.000**	*120.5926*	1	*7.4666 e-07*	1	1	0	0
	8×8	**100.000**	120.6052	1	7.4347 e-07	1	1	0	0
	16×16	**100.000**	120.6405	1	7.3500 e-07	1	1	0	0
	32×32	98.5246	120.8428	1	7.0147 e-07	1	1	0	0
Dolphin	4×4	99.9725	126.4020	1	7.0698 e-07	1	1	0	0
	8×8	99.8751	126.4148	1	7.0328 e-07	1	1	0	0
	16×1	99.6801	126.4327	1	6.9545 e-07	1	1	0	0
	32×32	*97.3406*	**126.5038**	1	**6.7450 e-07**	1	1	0	0

In model's embedding rate, the average is 99.42%, the standard deviation is 0.8112%, the worst is 97.34% by using EVS length 32×32 for model dolphin, and the best is 100% by using EVS length no more than 16×16 for model lion; the 100% embedding capacity means that each vertex of model is embedded with watermark or normalization information. Furthermore, a watermark unit consists of 24-bit watermark, 12-bit feature verification code (FVC), and 14-bit index, altogether 50 bits. So, the lowest embedding capacity of the proposed algorithm is $50 \times 97.3406\% = 48.67$ (bits/vertex). In signal-to-noise-ratio (*SNR*), the average is 123.1 db, the standard deviation is 2.584 db, the best is 126.5 db by using EVS length 32×32 for model dolphin, and the worst is 120.6 db by using EVS length 4×4 for model lion. In general, *SNR* is high when embedding capacity is low and vice versa. In modified Hausdorff distance (*MHD*), the average is 7.136×10^{-7}, the standard deviation is 0.2017×10^{-7}, the best is 7.467×10^{-7} by using EVS length 32×32 for model dolphin, and the worst is 6.745×10^{-7} by using EVS length 4×4 for model lion. Furthermore, all the correlation coefficients in each case are perfect value 1; this means that the embedded watermark is completely invisible. In other words, the distortions in the embedded models compared to the original models are too small to be found. This means that the perceptual quality of the watermarked models is quite good even at very high embedding capacity. Figure 6 and Table 1 also show that the *SNR* of each watermarked 3D model is slightly increased when the EVS length of the watermark is increased.

4.3. **Validation.** The similarity measurement of a watermark depends on the knowledge of experts, experimental conditions, etc. Therefore a quantitative measurement is necessary to provide a fair judgment of the extracted fidelity. In this paper, several measure metrics are conducted to measure the performance of the proposed algorithm. They are the normalized correlation (*NC*), the standard correlation coefficient (*SCC*), the pixel error rate (*PER*), and the bit error rate (*BER*). These performance measures are based on the original watermark W and the extracted watermark W'. The *NC*, *SCC* and *PER* value of extracted watermark are defined respectively by the following formulas [11]:

$$NC = \sum_i \sum_j W(i,j)W'(i,j) \Big/ \sum_i \sum_j (W(i,j))^2 \tag{22}$$

$$SCC = \sum_i \sum_j \left(W(i,j) - \overline{W}\right)\left(W'(i,j) - \overline{W'}\right) \Big/ \left(\left(\sum_i \sum_j \left(W(i,j) - \overline{W}\right)^2\right)^{1/2} \left(\sum_i \sum_j \left(W(i,j) - \overline{W}\right)^2\right)^{1/2}\right) \tag{23}$$

$$PER = n_{EP}/(W_x \times W_y) \tag{24}$$

$$BER = n_{EB}/(W_x \times W_y). \tag{25}$$

where $\overline{W}$ is the mean of W, $\overline{W}'$ is the mean of W', n_{EP} is the number of error pixels and n_{EB} is the number of error bits in the extracted watermark, and $W_x \times W_y$ is the size of the watermark.

The evaluated results in extracted watermarks' validation for each model under various lengths of embedding-vertex sequence are also arranged in Table 1, where the embedded capacity is the ratio of the number of embedded vertices to the total number of vertices of the watermarked model. In each case, *NC* and*SCC* are all one, *PER* and *BER* are all zero; this means that the extracted watermarks are completely correct. As shown in Table 1, the proposed algorithm achieves high embedding capacity with low distortion and high validation for extracted watermarks.

4.4. **Robustness.** To evaluate the robustness of the proposed algorithm against several affine transformation attacks, the watermarked models are attacked by affine transformations, vertex reordering, random noise addition, mesh simplification, cropping and multiple attacks. Moreover, the normalized correlation (NC), the standard correlation coefficient (SCC), the pixel error rate (PER) and the bit error rate (BER) are evaluated, respectively. These measurements will be evaluated many times for the same testing model and the resultant measurements will be the mean value. Furthermore, with a view to enhancing the robustness against attacks, the proposed algorithm uses the average value of the lost pixel's neighbor pixels to replace the lost pixel's value to repair the watermarks extracted from attacked models.

4.4.1. *Affine transformation attacks and reordering attacks.* To evaluate the robustness of the proposed algorithm against distortionless attacks, affine transformations and vertex reordering are carried out on watermarked models. Affine transforms were carried out with many combinations of translation, rotation and uniform scaling. Vertex reordering attacks are repeated 10 times, and the seed of random number generator is changed in each time. The change of the vertex interconnectivity can affect the embedded watermark. But the proposed algorithm does not change the vertex interconnectivity (or the mesh topology) in the watermarking procedure, so the proposed watermarking algorithm is absolutely robust against distortionless attacks.

4.4.2. *Smoothing, simplifying and cropping attacks.* Smoothing attacks add extra vertices and edges to a model. On the other hand, simplifying and cropping attacks remove certain vertices and edges from a model. These three attacks significantly change the mesh connectivity to affect the watermarked vertex selection and ordering.

The robustness against smoothing is tested as follows: A smoothed version of the model appears when people randomly increase vertices and then link the new vertices to their first ring neighbor vertices with new edges of the original model. The higher degree of smoothing means that more vertices and meshes are added into the model. Figure 7 shows the visualizations of watermarks extracted from smoothing attacked model angelfish; row 1 shows the smoothing model angelfishes that are respectively embedded a 64×64 grayscale image watermark with using EVS length 8×8 pixels, row 2 shows the number of vertices (N_V) and the number of meshes (N_M) of smoothed models, and row 3 shows the corresponding watermarks extracted from smoothed models. Experimental results show that the proposed scheme can completely withstand any smoothing attack because all the watermark embedded vertices are completely preserved.

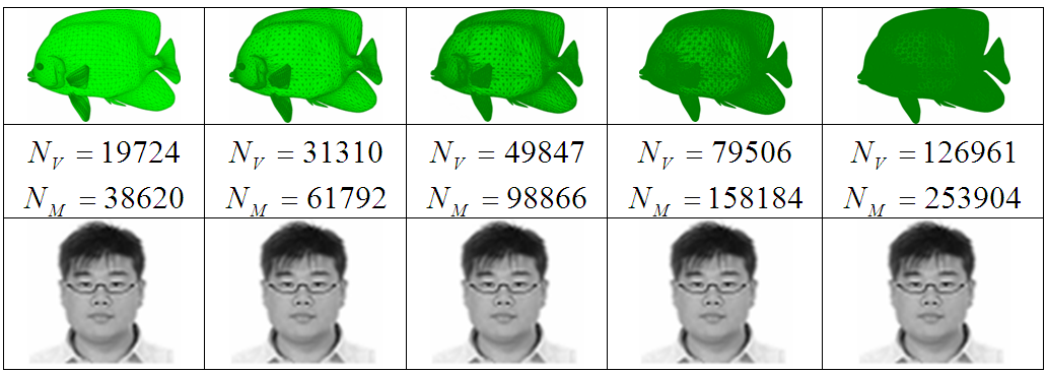

FIGURE 7. The visualizations of watermarks extracted from smoothing attacked angelfish

Some vertices are deleted in simplifying and cropping attacks in such a way that some pixels of the embedded watermark should be lost. The robustness against vertices deleting is tested as follows: (i) A simplified version of the model appears when people randomly delete some vertices and the meshes that are located in the first ring zone of a deleting vertex of the original watermarked model, and the first ring vertices are rearranged to form new triangular meshes. (ii) A cropped version of the model appears when people randomly delete some vertices and the meshes that are located around a deleting vertex. There are three types of cropping: type 1 deletes the randomly selected vertex and its 3 neighborhood vertices at the same time, type 2 deletes the randomly selected vertex and its 7 neighborhood vertices at the same time and type 3 deletes the randomly selected vertex and its 15 neighborhood vertices at the same time.

TABLE 2. The qualities of watermarks extracted from simplified models

Model	Reduced vertices	Reduced meshes	Before repairing				After repairing			
			NC	***SCC***	***PER***	***BER***	***NC***	***SCC***	***PER***	***BER***
Angelfish	386	772	1	1	0	0	1	1	0	0
	1158	2316	0.9853	0.9882	0.0037	0.0022	0.9968	0.9882	0.0029	9.16e-4
	1931	3862	0.8920	0.8683	0.0413	0.0288	0.9599	0.8722	0.0254	0.0097
	2703	5406	0.7814	0.6009	0.1685	0.1184	0.8331	0.6318	0.1057	0.0388
	3475	6950	0.4139	0.1981	0.5852	0.4170	0.4678	0.2883	0.4966	0.1536
	4248	8496	0.2673	0.1173	0.7324	0.5206	0.2987	0.2117	0.6541	0.2063
Teapot	576	1152	1	1	0	0	1	1	0	0
	1728	3456	0.9856	0.9485	0.0144	0.0100	0.9999	0.9491	0.0071	0.0029
	2880	5760	0.4671	0.2354	0.5332	0.3781	0.4910	0.3235	0.4338	0.1400
	4032	8064	0.0618	0.0251	0.9385	0.6643	0.0791	0.0959	0.8616	0.3229
Lion	320	640	0.9969	0.9884	0.0024	0.0020	1	0.9885	7.34e-4	3.05e-4
	962	1924	0.8439	0.7380	0.0930	0.0659	0.9077	0.7519	0.0564	0.0210
	1604	3208	0.3915	0.2001	0.6006	0.4226	0.4038	0.2940	0.5120	0.1689
	2246	4492	0.3728	0.1975	0.6018	0.4240	0.3983	0.2910	0.5122	0.1682
Dolphin	618	1236	0.9995	0.9980	4.88e-4	4.27e-4	1	0.9980	4.88e-4	1.83e-4
	1855	3710	0.9985	0.9961	0.0012	9.77e-4	0.9992	0.9961	7.32e-4	3.36e-4
	3598	7196	0.8915	0.9830	0.0049	0.0036	0.9537	0.9831	0.0034	0.0014
	4328	8656	0.7476	0.6518	0.1375	0.0969	0.8035	0.6757	0.0837	0.0308

Similarly, Table 2 shows the qualities of watermarks extracted from simplified models. Figure 8 shows the qualities of watermarks extracted from cropped models; it shows that the qualities of watermarks are lower when more vertices are deleted. Figure 9 shows the visualizations of watermarks extracted from model dolphin under type 3 cropping attacks; row 1 shows the cropped model dolphin, row 2 shows the extracted original watermarks which have many black points caused by vertices which were cropped, row 3 shows the corresponding repaired watermarks of row 2. Figure 9 also shows that the repaired watermark extracted from the model dolphin with 28% vertices have been cropped still can be identified. These experimental results show the proposed scheme can resist simplifying and cropping attacks.

Although smoothing, simplifying, and cropping attacks shall destroy the connectivity topology of vertices, the proposed algorithm is an indexed localization scheme and the watermarked model is normalized before watermark extraction. So, the proposed algorithm can resist these distortion attacks.

4.5. **Performance comparisons.** The performance comparisons among the relative schemes; Wang et al. algorithm [8], Wang and Hu's scheme [11], Liu and Chung's method [12]

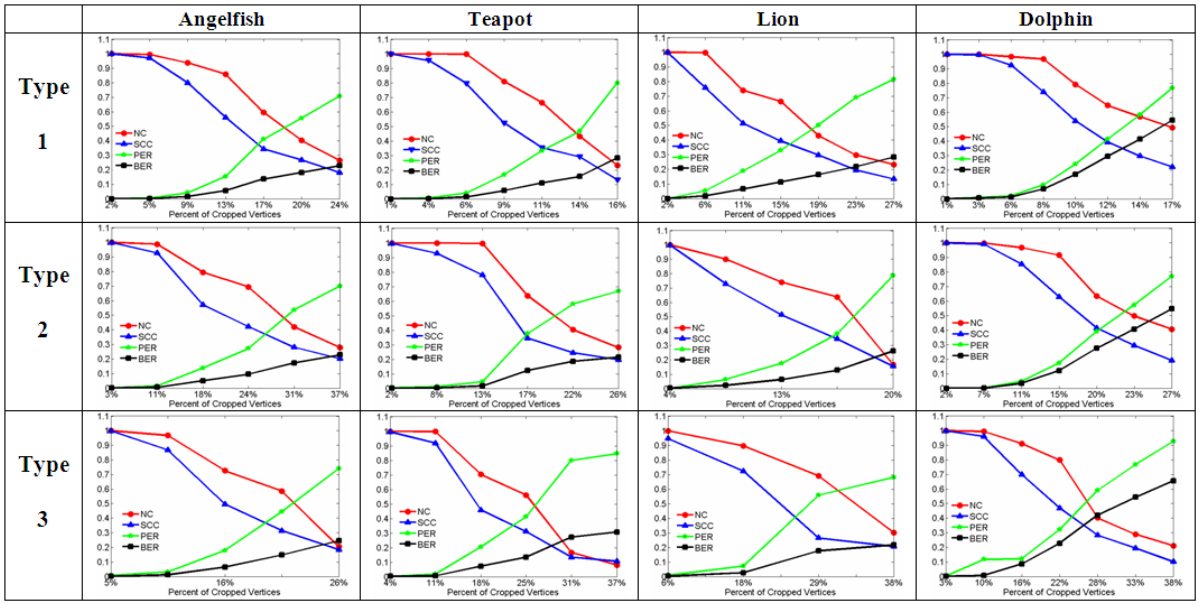

FIGURE 8. The qualities of watermarks extracted from cropped models

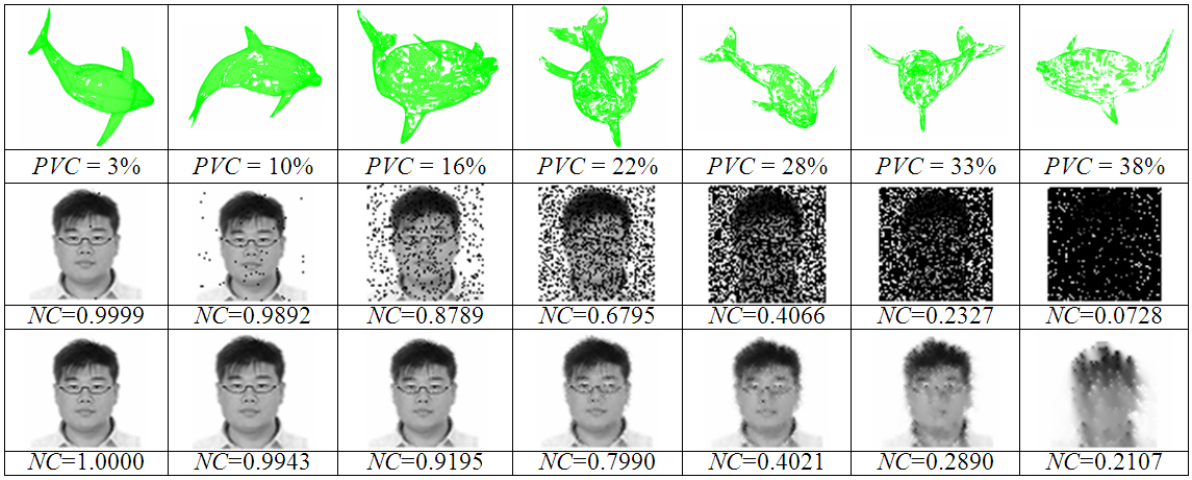

FIGURE 9. The visualizations of watermarks extracted from model dolphin under type 3 cropping attacks

and Agarwal and Prabhakaran's algorithm [13] is given in Table 3. As shown in Table 3, the proposed scheme has three advantages: (i) providing much higher embedding capacity (bits/vertex) with lower distortion than that of the other schemes, (ii) having the same performances as other schemes do in security, invisibility, validation, affine attacks and reordering attacks, (iii) possessing higher robust than the other methods do in smoothing, simplifying and cropping attacks.

5. **Conclusion.** Watermarking has been proved to be a good technology in the intellectual property and copyright protection for digital multimedia. In this paper, the 3D model normalization scheme, vertices clustering scheme, and discrete cosine transform (DCT) are combined to constitute a superior watermarking algorithm for 3D models. Experimental results show that the proposed algorithm is high performance 3D model watermarking algorithm, and has the following features: (i) the watermarked 3D model possesses excellent imperceptibility without noticeable degradation; (ii) both the embedding capacity

TABLE 3. Comparisons with relative schemes (◎: perfect, ◯: good, □: bad, ▼: worse)

Items	Capacity	Secu -rety	Invis -ibility	Valid -ation	Affine attack	Reordering attack	Smoothing attack	Simplifying attack	Cropping attack
Wang et al. [8]	$\leqq 48$	◎	◎	◎	◎	◎	▼	▼	▼
Wang and Hu [11]	$0.14 \sim 0.16$	◎	◎	◎	◎	◎	▼	▼	▼
Liu and Chung [12]	$0.3 \sim 0.4$	◎	◎	◎	◎	◎	▼	◯	◯
Agarwal et al. [13]	$0.4 \sim 0.5$	◎	◎	◎	◎	◎	▼	□	□
Proposed	$48.67 \leqq$	◎	◎	◎	◎	◎	◎	◯	◯

and the extracted watermark's validality are better than other relative schemes; (iii) the security is strong enough, due to the watermark is embedded in DCT domain; (iv) the robustness against affine transformations is 100 %, and the robustness withstanding the cropping, simplifying, reordering and smoothing attacks is also high; (v) the original 3D model and watermark are not needed during the watermark extraction process. In the future, we are going to combine other theories and new watermark repairing scheme to enhance the 3D model watermarking system.

Acknowledgment. This work is partially supported by National Science Council of Taiwan under Grants NSC 100-2221-E-167-028.

REFERENCES

[1] W. Y. Chen and C. C. Liu, Multiple-watermarking scheme of the European article number barcode using adaptive phase shift keying technique, *Optical Engineering*, vol.46, no.6, pp.1-12, 2007.

[2] D. Xiao and F. Y. Shih, A reversible image authentication scheme based on chaotic fragile watermark, *International Journal of Innovative Computing, Information and Control*, vol.6, no.10, pp.4731-4742, 2010.

[3] C. C. Liu and W. Y. Chen, Multiple-watermarking scheme for still images using the DCT and modified code division multiple-access techniques, *Optical Engineering*, vol.45, no.7, pp.1-12, 2006.

[4] D. Chou, C.-Y. Jhou and S.-C. Chu, Reversible watermark for 3D vertices based on data hiding in mesh formation, *International Journal of Innovative Computing, Information and Control*, vol.5, no.7, pp.1893-1901, 2009.

[5] J.-T. Wang, P.-C. Wang and S.-S. Yu, A PCA and perturb based fragile watermarking scheme for 3D models, *International Journal of Innovative Computing, Information and Control*, vol.6, no.7, pp.3145-3156, 2010.

[6] S. Yang, C. Li, S. Sun and Y. Xu, A robust 3D model watermarking scheme based on feature recognition, *Proc. of the 8th ACIS Int. Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, vol.3, pp.989-993, 2007.

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

[8] J.-T. Wang, Y.-W. Yang, Y.-T. Chang and S.-S. Yu, A high verification capacity reversible fragile watermarking scheme for 3D models, *International Journal of Innovative Computing, Information and Control*, vol.7, no.1, pp.365-378, 2011.

[9] C.-C. Chen and H.-A. Ke, Image authentication under DCT domain with attack recovery, *International Journal of Innovative Computing, Information and Control*, vol.6, no.3(A), pp.885-896, 2010.

[10] C. C. Liu and P. C. Chung, A robust normalization algorithm for three dimensional models based on clustering and star topology, *International Journal of Innovative Computing, Information and Control*, vol.7, no.11, 2011 (in press).

[11] Y. P. Wang and S. M. Hu, Optimization approach for 3D model watermarking by linear binary programming, *Computer Aided Geometric Design*, vol.27, pp.395-404, 2010.

[12] C. C. Liu and P. C. Chung, A robust three-dimensional model watermarking algorithm based on connected vertices clustering and star topology, *International Journal of Advanced Information Technologies*, vol.4, no.2, pp.104-121, 2010.

[13] P. Agarwal and B. Prabhakaran, Robust blind watermarking of point-sampled geometry, *IEEE Transactions on Information Forensics and Security*, vol.4, pp.36-48, 2009.