Hindawi
Security and Communication Networks
Volume 2017, Article ID 1096463, 11 pages
https://doi.org/10.1155/2017/1096463

*Research Article*

# A Low-Complexity Region-Based Authentication Algorithm for 3D Polygonal Models

**Yuan-Yu Tsai,[1,2] Tsung-Chieh Cheng,[3] and Yao-Hsien Huang[4]**

[1]*Department of M-Commerce and Multimedia Applications, Asia University, Taichung, Taiwan*
[2]*Department of Medical Research, China Medical University Hospital, China Medical University, Taichung, Taiwan*
[3]*Institute of Nuclear Energy Research, Taoyuan, Taiwan*
[4]*Department of Information Technology and Management, Shih Chien University, Taipei, Taiwan*

Correspondence should be addressed to Yao-Hsien Huang; daboo.huang@gmail.com

Received 27 July 2016; Revised 20 October 2016; Accepted 9 November 2016; Published 12 January 2017

Academic Editor: Ángel Martín Del Rey



This study proposes a low-complexity region-based authentication algorithm for three-dimensional (3D) polygonal models, based on local geometrical property evaluation. A vertex traversal scheme with a secret key is adopted to classify each vertex into one of two categories: embeddable vertices and reference vertices. An embeddable vertex is one with an authentication code embedded. The algorithm then uses reference vertices to calculate local geometrical properties for the corresponding embeddable vertices. For each embeddable vertex, we feed the number of reference vertices and local properties into a hash function to generate the authentication code. The embeddable vertex is then embedded with the authentication code, which is based on a simple message-digit substitution scheme. The proposed algorithm is of low complexity and distortion-controllable and possesses a higher and more adaptive embedding capacity and a higher embedding rate than most existing region-based authentication algorithms for 3D polygonal models. The experimental results demonstrate the feasibility of the proposed algorithm.

## 1. Introduction

With the rapid development of information technology, traditional data is now stored in digital form on computers. Users can make digital data available worldwide via the Internet; however, such convenient and advanced technology also engenders new problems relating to information security, such as multimedia content being copied and distributed illegally or being tampered with. Therefore, the development of powerful techniques to protect both intellectual property rights and the integrity of multimedia content is a noteworthy field of research.

Petitcolas et al. [1] classified information hiding techniques into several branches, copyright marking being just one of them. According to different application levels, copyright marking techniques can be categorized as either robust watermarking or fragile watermarking. Robust watermarking algorithms can protect the intellectual property rights of multimedia content. Even after malicious attacks, the relative copyright information, commonly called watermark, is still detectable. The goal of fragile watermarking algorithms is to verify the integrity of multimedia content. Even after minimal unauthorized modification, tampered regions can still be located using the embedded authentication code; some techniques can even perform self-recovery operations on tampered regions.

Fragile watermarking is a popular research field, and its application to digital images has been extensively studied [2–5]; Abhayaratne and Bhowmik [6] even proposed a real-time authentication approach for JPEG images. However, fragile watermarking algorithms on 3D polygonal models still need a breakthrough. Several approaches [7–14] have been proposed, but most of them provide a low embedding capacity and low robustness against similarity transformation (ST) attacks, including rotation, translation, and scaling operations. Some algorithms cannot effectively detect modifications in the topological relationship between vertices; moreover, some algorithms require a high level of computational

performance during the authentication code embedding procedure. Such requirements may originate from iterative perturbation or complicated neighboring structures for embeddable vertices. As a result, a more powerful fragile watermarking algorithm for 3D polygonal models is urgently needed. Regarding the development of various 3D interactive applications, the present study proposes an algorithm with low computational complexity. Relative to existing methods, the proposed method should be more suitable and more efficient for real-time implementation.

With a view to verifying the integrity of 3D polygonal models, this paper proposes a low-complexity, high-capacity, distortion-controllable, region-based fragile watermarking algorithm with a high embedding rate.

This approach has five characteristics. First, an iterative vertex traversal mechanism is used to classify each vertex into one of two categories to enhance the security of the authentication code. Second, the authentication code generation is based on local geometrical properties concerning the topological relationship in a one-ring neighboring structure. Thus, the modifications of the topological relationship can be considered. Third, the proposed algorithm supports adaptive embedding capacity controlled by an embedded threshold. The embedding capacity is the highest among existing region-based tamper detection algorithms. Fourth, the embedding rate is higher than most existing region-based authentication algorithms. Fifth and finally, relative to existing methods, the proposed method is more suitable and more efficient for real-time implementation. The above characteristics demonstrate the feasibility of the proposed technique in fragile watermarking.

The remainder of this paper is organized as follows: Section 2 provides a review of related studies; Section 3 details the proposed technique; Section 4 presents a discussion on the experimental results; and a brief conclusion and directions for future research are provided in Section 5.

## 2. Related Studies

Depending on the scale of tampered region localization, fragile watermarking algorithms for 3D polygonal models can be classified as methods for either vertex-based or region-based tamper detection. A vertex-based tamper detection algorithm can accurately locate the altered vertex but cannot consider the modifications of the topological relationship. A region-based tamper detection algorithm can only locate a suspicious region with some unaltered vertices inside that region; however, even with a region-based method, the topological relationship can be considered.

For region-based tamper detection algorithms, Yeo and Yeung [14] proposed the first fragile watermarking algorithm for authenticating 3D polygonal models: Two indices, named the location index and the value index, are generated from two predefined hash functions for each vertex. Their algorithm modifies the vertex position iteratively to allow its status to meet the two aforementioned calculated indices. Tamper detection is performed by examining the consistency of the two indices in the decoding and reconstruction phases. Although their scheme is blind, the distortion is not controllable by users because of the iterative vertex perturbing process. Adopting the concept of distance invariants, Chou and Tseng [8] employed a blind fragile watermarking algorithm that is robust against ST attacks. The watermark is embedded using a multifunction-vertex embedding method and an adjusting-vertex method; however, the embedding capacity is low, and false alarms concerning the unprocessed vertices can occur during the watermark extraction process. Furthermore, the applied neighboring structures increase not only computational performance, but also the scale of tampered region localization. Molaei et al. [10] modified the triangle traversal strategy proposed by Chou and Tseng [8] to effectively determine the mark triangles. For each mark triangle, the watermark, the hash result, and an identification mark are embedded into the middles of triangle sides in the spherical coordinate system by quantization index modulation (QIM) [15] technique. Finally, the new mark triangle is reconstructed by crossing three lines derived from the data-embedded middles. The authors claim that the recognizing accuracy of the proposed algorithm can be increased with the help of more marked vertices.

For vertex-based tamper detection algorithms, Chen et al. [7] proposed two fragile watermarking algorithms, named adaptive authentication and vertex digest, for 3D polygonal models. The adaptive authentication algorithm feeds the number of neighboring vertices, the watermark, and the $x$, $y$ coordinates for the processing vertex into the hash function. The hash result is then embedded into the $z$ coordinate on the basis of a modulus operation. The vertex digest algorithm applies a secret key to replace the watermark in the former algorithm, thereby achieving blind tamper detection; however, the proposed algorithm cannot protect against ST attacks because the vertex coordinates are modified, resulting in a different hash result. Wang et al. [11] proposed a reversible fragile watermarking scheme in the spatial domain. The researchers employed reversible QIM to modify the distance between the embedding vertex and the center of gravity. This algorithm integrates principal component analysis and spectrum techniques to significantly enhance robustness against ST and vertex reordering attacks. In addition, the embedding capacity can be up to 48 bits per vertex under IEEE 754 single precision for vertex coordinate representation with a two-level embedding strategy. In a later study, Wang et al. [12] proposed a different fragile watermarking algorithm for 3D polygonal models based on a Hamming code. The proposed algorithm normalizes each vertex coordinate into a floating value between 0 and 1. The fourth to seventh least significant bits in the binary form of the normalized value are extracted to be the data bits. Three parity check bits are generated based on the (7, 4) Hamming code and subsequently embedded into the three least significant bits of the normalized value. Huang et al. [9] proposed a spherical coordinate-based fragile watermarking scheme for 3D polygonal models, in which the Cartesian coordinates $(x, y, z)$ of each vertex of the cover model are transformed into spherical coordinates $(r, \theta, \varphi)$. The researchers adopted the QIM technique to embed the watermark into the $r$ coordinates; however, for this method

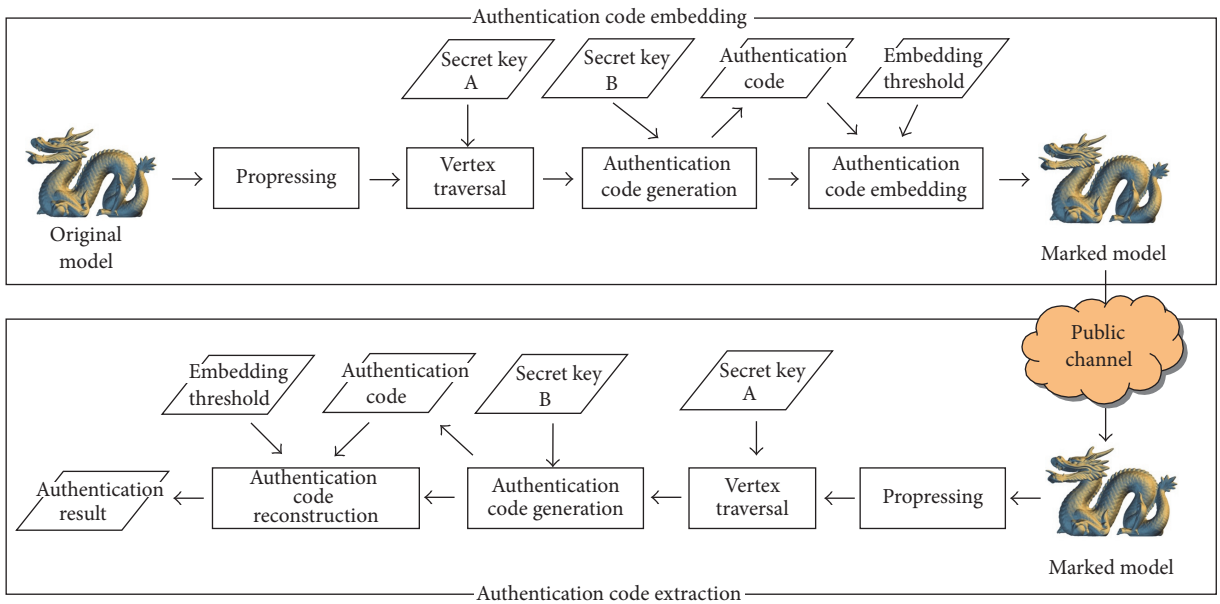



Figure 1: Flowchart of proposed algorithm, including authentication code embedding procedure and authentication code extraction procedure.

the original model's center of gravity must be recorded; otherwise the authentication procedure fails.

## 3. Proposed Algorithm

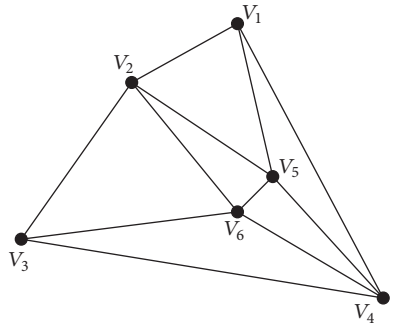
This section details the proposed authentication mechanism for 3D polygonal models. This method first employs a secret key and a vertex traversal method to visit each vertex. Each vertex is then categorized as either an embeddable vertex or a reference vertex. Local geometrical property calculation is performed on the embeddable vertex by considering its geometrical relationship with the neighboring reference vertices. Subsequently, the aforementioned information is entered into the hash function to generate the authentication code; and finally, each embeddable vertex is embedded with the authentication code, based on a simple message-digit substitution scheme. Figure 1 presents a flowchart of the proposed algorithm. Sections 3.1 and 3.2 discuss each process of the proposed algorithm in detail, and Section 3.3 illustrates the robustness of the proposed algorithm in preventing ST attacks.

*3.1. Authentication Code Embedding Procedure.* The preprocessing process analyzes the topological information of the input model to generate a vertex neighboring table, $\mathrm{VNT}_{V_i}$, that records information about the neighbors of each vertex $V_i$. Model information is also derived during this process. The vertex traversal process employs a secret key to determine the start vertex, and an iterative vertex traversal mechanism is initialized. The authentication code generation process feeds the number of reference vertices and the local geometrical properties of the embeddable vertex into the hash function to obtain the authentication code. The authentication code is finally embedded into the embeddable vertex by a simple message-digit substitution scheme. For the sake of simplicity, this study illustrates the proposed technique by using two-dimensional (2D) vertices; however, these could easily be transferred to a three-dimensional (3D) space.

*3.1.1. Preprocessing Process.* The preprocessing process prepares four temporary stores for efficient authentication code embedding and extraction. The first store, $\mathrm{VNT}_{V_i}$, is a vertex neighboring table that records the indices of the actual neighbors of each vertex $V_i$. The second store, Ex, records the value of each vertex $V_i$ in the 3D polygonal model, including explored ("T") and unexplored ("F"). The initial value of each vertex is set as "F." The third store, Sort, records the processing order of each vertex, and the fourth store, Ref, records the indices of the reference vertices of each embeddable vertex for local geometrical property calculation. All values in these four stores are based on the topological relationship and input order of each vertex in the model file. After preprocessing, only the store $\mathrm{VNT}_{V_i}$ can be determined. The values of the other three stores are obtained following the vertex traversal process. Figure 2(a) shows a model consisting of six vertices. As previously mentioned, only the set in the store $\mathrm{VNT}_{V_i}$, which records the indices of the actual neighbors of the corresponding vertex $V_i$, is determined after the preprocessing process.

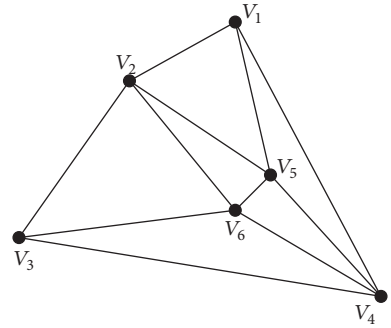
*3.1.2. Vertex Traversal Process.* According to the input order of each vertex in the input model, this process employs a secret key (secret key A in Figure 1) to be a seed for performing a random permutation of the input vertices and therefore enhances the security of the authentication code. The permutation result is exactly the processing order for each vertex. During vertex traversal, each vertex is categorized as either an embeddable vertex or a reference

| $V_i$ | $VNT_{V_i}$ | Sort | Ex | Ref |
|---|---|---|---|---|
| $V_1$ | 2, 4 ,5 | | F | |
| $V_2$ | 1, 3, 5, 6 | | F | |
| $V_3$ | 2, 4, 6 | | F | |
| $V_4$ | 1, 3, 5, 6 | | F | |
| $V_5$ | 1, 2, 4, 6 | | F | |
| $V_6$ | 2, 3, 4, 5 | | F | |



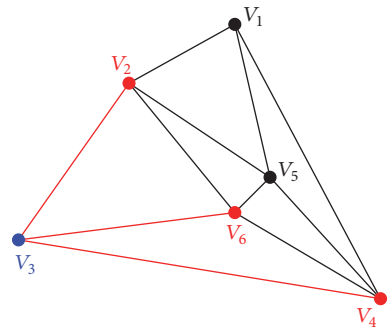

(a)

| $V_i$ | $VNT_{V_i}$ | Sort | Ex | Ref |
|---|---|---|---|---|
| $V_1$ | 2, 4 ,5 | 6 | F | |
| $V_2$ | 1, 3, 5, 6 | 5 | F | |
| $V_3$ | 2, 4, 6 | 1 | F | |
| $V_4$ | 1, 3, 5, 6 | 3 | F | |
| $V_5$ | 1, 2, 4, 6 | 2 | F | |
| $V_6$ | 2, 3, 4, 5 | 4 | F | |



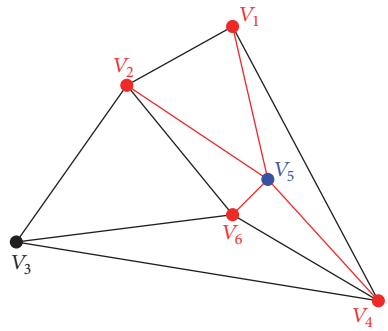

(b)

| $V_i$ | $VNT_{V_i}$ | Sort | Ex | Ref |
|---|---|---|---|---|
| $V_1$ | 2, 4 ,5 | 6 | F | |
| $V_2$ | 1, 3, 5, 6 | 5 | T | $x$ |
| $V_3$ | 2, 4, 6 | 1 | T | 2, 4, 6 |
| $V_4$ | 1, 3, 5, 6 | 3 | T | $x$ |
| $V_5$ | 1, 2, 4, 6 | 2 | F | |
| $V_6$ | 2, 3, 4, 5 | 4 | T | $x$ |




(c)

| $V_i$ | $VNT_{V_i}$ | Sort | Ex | Ref |
|---|---|---|---|---|
| $V_1$ | 2, 4 ,5 | 6 | T | $x$ |
| $V_2$ | 1, 3, 5, 6 | 5 | T | $x$ |
| $V_3$ | 2, 4, 6 | 1 | T | 2, 4, 6 |
| $V_4$ | 1, 3, 5, 6 | 3 | T | $x$ |
| $V_5$ | 1, 2, 4, 6 | 2 | T | 1, 2, 4, 6 |
| $V_6$ | 2, 3, 4, 5 | 4 | T | $x$ |




(d)

Figure 2: Example of vertex traversal process.

vertex. Each embeddable vertex has an authentication code embedded into it; each reference vertex is used to perform local geometrical property calculation for the corresponding embeddable vertex. The vertex traversal scheme performs the subsequent steps iteratively over the vertices in the 3D polygonal model, including selecting the embeddable vertex, modifying the status of neighboring vertices, and determining the reference vertices. This process is terminated until no vertices remain with the Ex value "F."

First, we select the first vertex $V_i$ in the sorted list, and its value, $Ex_{V_i}$, changes to "T." All the indices $i$ appear in the store $VNT_{V_i}$ and are subsequently added into the store $Ref_{V_i}$, these being the reference vertices for local geometrical property calculation. The value $Ex_{V_j}$ for each index $j$ appearing in store $Ref_{V_i}$ is also modified to "T." Thereafter, we detect and select the next vertex $V_k$ with $Ex_{V_k}$ value "F" in the sorted list and adopt its neighboring vertices as reference vertices. The aforementioned action is repeated iteratively until there are no remaining vertices to be selected.

Figure 2(a) assumes that the input order for each vertex is $V_1$, $V_2$, $V_3$, $V_4$, $V_5$, and $V_6$, and the Ex value for each vertex is initialized as "F." A secret key is employed to generate a sorted list $V_3$, $V_5$, $V_4$, $V_6$, $V_2$, and $V_1$ [Figure 2(b)]. Vertex $V_3$ in Figure 2(c) is selected first because it is the first vertex in the sorted list. The $Ex_{V_3}$ value is then modified to "T." The values of $Ex_{V_2}$, $Ex_{V_4}$, and $Ex_{V_6}$ for vertices $V_2$, $V_4$, and $V_6$ are also modified to "T" because indices 2, 4, and 6 all appear in the store $VNT_{V_3}$. Indices 2, 4, and 6 are then added sequentially into the store $Ref_{V_3}$. Subsequently, vertex $V_5$ in Figure 2(d) is selected and the $Ex_{V_5}$ value is modified to "T." Evidently, only the $Ex_{V_1}$ value is modified to "T." Indices 1, 2, 4, and 6 are then added into the store $Ref_{V_5}$. This process is terminated because at this point each vertex has been processed. Vertices $V_3$ and $V_5$ are categorized as embeddable, whereas $V_2$, $V_4$, and $V_6$ are

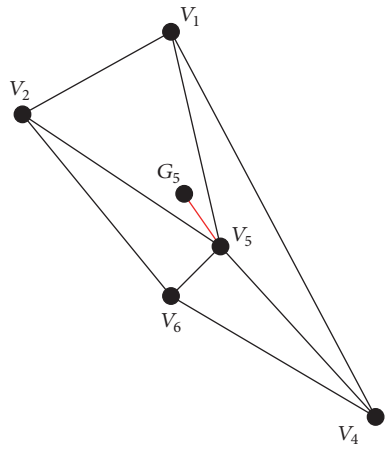



Figure 3: Example of local geometrical property calculation.

the reference vertices of $V_3$, with vertices $V_1$, $V_2$, $V_3$, and $V_6$ being the reference vertices of $V_5$.

*3.1.3. Authentication Code Generation Process.* We obtain the reference vertices for each embeddable vertex and perform the authentication code generation process. In the algorithm proposed by Chen et al. [7], only the number of connected neighbors is used to generate the authentication code. The tamper detection process fails, and the topological relationship is changed but still has the same number of connected neighbors. Therefore, in this step, we feed the number of reference vertices along with their geometrical properties into the hash function to generate the authentication code. Here, to resist ST attacks, we used a simple geometrical property, the ratio between the length of each side $L_{V_i}$ between the reference vertices of the embeddable vertex $V_i$ and the summation $S_{V_i}$ of all sides. The hash results must be different for different input data; therefore, we feed each ratio, starting from the lowest, into the hash function, thereby obtaining some required bits from the hash result based on another secret key. Each embeddable vertex only contains several bits from the hash result, denoted as authentication code, and embedded because our proposed algorithm does not allow each vertex to convey such a large message.

Following the example in Figure 2, vertex $V_5$ in Figure 3 is an embeddable vertex with four reference vertices: $V_1$, $V_2$, $V_4$, and $V_6$. $G_5$ is the center of gravity of all the reference vertices of vertex $V_5$, and the values for the length of each side from low to high can be expressed as $\overline{V_1V_2}$, $\overline{V_2V_6}$, $\overline{V_6V_4}$, and $\overline{V_4V_1}$. Therefore, in this case, we feed the value $\mathrm{Val}_{V_5}$ shown in (1) into the hash function, where "$\|$" is the concatenation operator, and $S_{V_5}$ equals the sum of $\overline{V_1V_2}$, $\overline{V_2V_6}$, $\overline{V_6V_4}$, and $\overline{V_4V_1}$.

$$\mathrm{Val}_{V_5} = 4 \parallel \frac{\overline{V_1V_2}}{S_{V_5}} \parallel \frac{\overline{V_2V_6}}{S_{V_5}} \parallel \frac{\overline{V_6V_4}}{S_{V_5}} \parallel \frac{\overline{V_4V_1}}{S_{V_5}}. \tag{1}$$

*3.1.4. Authentication Code Embedding Process.* After generating the authentication code, we then calculate the ratio $P_{V_i}$ with a value lower than 1, between the distance $\|V_iG_i\|$ and the perimeter of reference vertices $S_{V_i}$ (see (2)). An embedding threshold, $T$, is used to determine the start position for data embedding, which is the $T$th digit after the first nonzero digit. To avoid extraction errors resulting from the floating-point truncation/rounding calculation, the end position for data embedding is set as the fifth digit after the decimal point. If the start position is already larger than the end position, the authentication code is directly embedded into the fifth digit. For each digit between the start and end positions, we sequentially obtain three to four bits from the authentication code and modify the value in the corresponding position based on a simple message-digit substitution scheme shown in Table 1. Finally, the data-embedded vertex can be obtained following the direction of $\overrightarrow{V_iG_i}$, using (3) with a data-embedded ratio $P'_{V_i}$.

Table 1: Message-digit substitution.

| Decimal digit | Binary authentication code |
|---|---|
| 0 | 000 |
| 1 | 001 |
| 2 | 010 |
| 3 | 011 |
| 4 | 100 |
| 5 | 101 |
| 6 | 1100 |
| 7 | 1101 |
| 8 | 1110 |
| 9 | 1111 |

$$P_{V_i} = \begin{cases} \dfrac{\|V_iG_i\|}{S_{V_i}} & \text{if } \|V_iG_i\| \le S_{V_i} \\ \dfrac{S_{V_i}}{\|V_iG_i\|} & \text{if } \|V_iG_i\| > S_{V_i}, \end{cases} \tag{2}$$

$$V_i' = \begin{cases} \left(P'_{V_i} \times S_{V_i}\right) \times \dfrac{\overrightarrow{V_iG_i}}{\|V_iG_i\|} + G_i & \text{if } \|V_iG_i\| \le S_{V_i} \\ \left(\dfrac{S_{V_i}}{P'_{V_i}}\right) \times \dfrac{\overrightarrow{V_iG_i}}{\|V_iG_i\|} + G_i & \text{if } \|V_iG_i\| > S_{V_i} \end{cases} \tag{3}$$

Figure 3 assumes that the ratio $P_{V_5} = \|V_5G_5\|/S_{V_5} = 0.021589$, because $S_{V_5}$ is larger than $\|V_5G_5\|$. The permutation hash result is a string consisting of "10111011001101…" and the embedding threshold $T$ is set as 1. As mentioned previously, the first nonzero digit is 2, located at the second digit. Therefore, the positions for data embedding are from the third to the fifth digit. We then take part of the hash result three times for embedding, with values of "101," "1101," and "100" as the authentication code. The corresponding decimal digits equal 5, 7, and 4, respectively. Therefore, the ratio with authentication code embedding equals 0.025749. Finally, vertex $V_5$ is modified to $V_5'$ following the direction of $\overrightarrow{V_5G_5}$ (Figure 4).

Table 2: Model information of five 3D polygonal models.

| Model name | $N_V$ | $N_F$ | $\mathrm{DL_{BV}}$ | $N_{\mathrm{EV}}$ | $N_{\mathrm{EV}}/N_V$ |
|---|---|---|---|---|---|
| Cow | 46433 | 92864 | 30.49 | 13968 | 30.08% |
| Lucy | 262909 | 525814 | 1918.29 | 68797 | 26.17% |
| Max Planck | 49132 | 98260 | 697.49 | 12777 | 26.01% |
| Elephant | 19753 | 39290 | 144.69 | 4816 | 24.38% |
| Rabbit | 67038 | 134074 | 33.91 | 18504 | 27.60% |

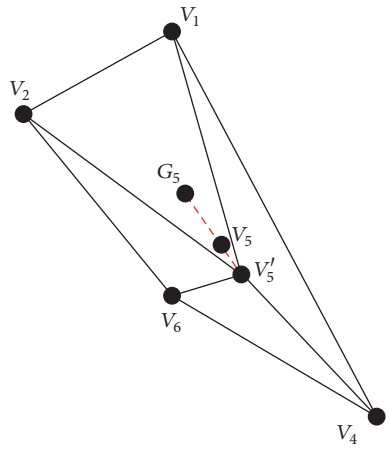



Figure 4: Example of authentication code embedding process.

*3.2. Authentication Code Extraction Procedure.* After obtaining the secret key and the marked model, the authentication code extraction procedure performs integrity verification. The preprocessing process first prepares four temporary stores for authentication code extraction. As mentioned, the construction of these stores is based on the topological properties and input order of each vertex; however, the authentication code embedding procedure never modifies the topological information, and these stores can be correctly reconstructed using the same processes as those in the authentication code embedding procedure. The vertex traversal process iteratively determines the embeddable vertex and its reference vertices and properly modifies the temporary stores with the help of the secret key. The authentication code generation process feeds the number of reference vertices accompanied by their local geometrical properties into the hash function to generate the hash result. The same secret key is used to take some bits from the hash result to generate the authentication code. In addition, we also obtain the ratio between $\|V_iG_i\|$ and $S_{V_i}$. With the help of the embedding threshold $T$, we can extract several bits of the authentication code from the ratio. Finally, by comparing the data derived from the two sides, integrity verification is achieved.

*3.3. Robustness Assessment.* The proposed algorithm is robust against rotating and translation attacks because the ratio between the distance $\|V_iG_i\|$ and the perimeter of the reference vertices $S_{V_i}$ remains invariant under such attacks. When the stego model comes under a scaling attack, both the distance $\|V_iG_i\|$ and the perimeter of the reference vertices $S_{V_i}$ are scaled proportionally. The ratio still remains invariant; however, when the marked model is subjected to a downscaling attack, the precision of the numbers for representing the vertex coordinates for each vertex should be sufficient for extracting the authentication code.

## 4. Experimental Results

This section presents the experimental results obtained from five 3D polygonal models shown in Figure 5, called "Cow," "Lucy," "Max Planck," "Elephant," and "Rabbit." Table 2 shows relevant model information, including the number of vertices $N_V$, the number of faces $N_F$, and model size (represented by the diagonal length $\mathrm{DL_{BV}}$ of the bounding volume) of each model. We also show that the number of embeddable vertices $N_{\mathrm{EV}}$ in our proposed vertex traversal method varies from 24% to 30% of all vertices for different 3D polygonal models. Microsoft Visual C++ programming language was used to implement the proposed algorithm on a personal computer with an Intel Core i7-6700K 4.00 GHz processor and 16 GB RAM. The distortion between the original and marked 3D polygonal models was measured by normalized Hausdorff distance (NHD) [16], which is derived from dividing the Hausdorff distance by $\mathrm{DL_{BV}}$.

First, this section presents the visual effects for five marked 3D polygonal models with different embedding thresholds and also the quantities of embedded authentication codes under different embedding thresholds. Second, we present the frequency of the embedding capacity for each embeddable vertex, to explain the distribution of the calculated local geometrical properties, and the experimental results for tamper detection in our proposed algorithm. Finally, a comparison between this algorithm and other existing algorithms demonstrates the feasibility of the proposed method.

Table 3 shows the embedding capacity (EC), model distortion (NHD), and execution time for different 3D polygonal models under different embedding thresholds. We also calculate the average embedding capacity bpv for each embeddable vertex. Evidently, as the embedding threshold increases, both the embedding capacity and model distortion decrease, mainly because with a larger embedding threshold, the start position is also larger, and the number of digits for data embedding is lower. Therefore, the embedding capacity decreases from 8.95 to 3.27 bits per embeddable vertex, and the model distortion also decreases, from 0.1957% to 0.00009% of the diagonal length of the bounding volume. Our proposed algorithm is efficient; except for Lucy model

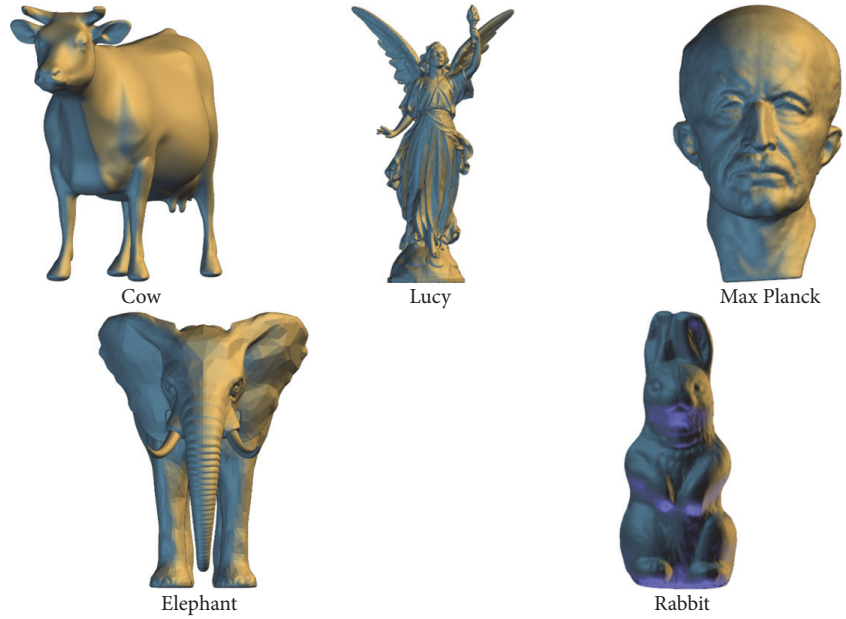



Figure 5: Visual effects of five 3D polygonal models.

Table 3: Comparisons of embedding capacity, model distortion, and time execution under different embedding thresholds.

| Model name | $T = 1$ | | | | $T = 2$ | | | | $T = 3$ | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | EC | bpv | NHD | Time | EC | bpv | NHD | Time | EC | bpv | NHD | Time |
| Cow | 108200 | 7.75 | 0.0380% | 0.209 | 62873 | 4.50 | 0.0035% | 0.209 | 45415 | 3.25 | 0.0005% | 0.203 |
| Lucy | 652499 | 9.48 | 0.0239% | 1.062 | 433616 | 6.30 | 0.0041% | 1.060 | 224752 | 3.27 | 0.0004% | 1.060 |
| Max Planck | 117011 | 9.16 | 0.1451% | 0.197 | 75540 | 5.91 | 0.0108% | 0.197 | 41556 | 3.25 | 0.0009% | 0.195 |
| Elephant | 46107 | 9.57 | 0.6942% | 0.077 | 30401 | 6.31 | 0.0170% | 0.077 | 15883 | 3.30 | 0.0019% | 0.077 |
| Rabbit | 162576 | 8.79 | 0.0774% | 0.279 | 102799 | 5.56 | 0.0074% | 0.278 | 60551 | 3.27 | 0.0008% | 0.275 |
| Average | 217279 | 8.95 | 0.1957% | 0.365 | 141046 | 5.72 | 0.0086% | 0.364 | 77631 | 3.27 | 0.0009% | 0.362 |

with larger number of vertices, the execution time from vertex traversal to authentication code embedding processes only needs less than 0.3 seconds. Figure 6 shows the embedding capacity distribution for each vertex under different thresholds. By analyzing this data, we can understand the distribution of local geometrical properties calculated by (2). With the exception of the cow model, the highest embedding capacity for each embeddable vertex in each 3D polygonal model is between 9 and 11. Three digits of the calculated ratio are used for data embedding, each with three or four bits. Because the end position for data embedding is always the fifth digit, we know that the start position is the third digit in most ratios; therefore, the first nonzero digit in most ratios is the second digit after the decimal point. Figure 7 shows the visual effects of data-embedded 3D polygonal models under various embedding thresholds. As shown in Figures 5 and 7, distinguishing the visual difference between the original models and the data-embedded models is difficult.

Figure 8 shows the visual effects of the detection results for different 3D polygonal models after some vertices have been perturbed. Following each process illustrated in Section 3, the tampered vertex (red) can easily be detected. In our experiments, we randomly modified the coordinate values of 50 vertices to evaluate the performance of our proposed algorithm. The experimental results show that all tampered vertices were detected. Note that both authentication code generation and the reconstruction process were only performed on the embeddable vertex. If the embeddable vertex is a modified one, the tamper detection region includes one embeddable vertex and its reference vertices. If a reference vertex is modified, the tamper detection region includes several embeddable vertices and their corresponding reference vertices. Evidently, the tamper detection region is larger if the modified vertex is a reference vertex. In our experimental results, the tamper detection region for each 3D polygonal model included 511, 406, 531, 468, and 575 vertices for Cow, Lucy, Elephant, Max Planck, and Rabbit models, respectively, because of the complex topological relationship between vertices. Future research should be devoted to more precise detection of tampered regions.

Finally, this section compares the proposed algorithm with other existing algorithms in terms of embedding rate, embedding capacity, adaptability, distortion control, robustness, and embedding scheme (Table 4). All fragile watermarking algorithms can be classified as either vertex-based or region-based algorithms, according to the scale

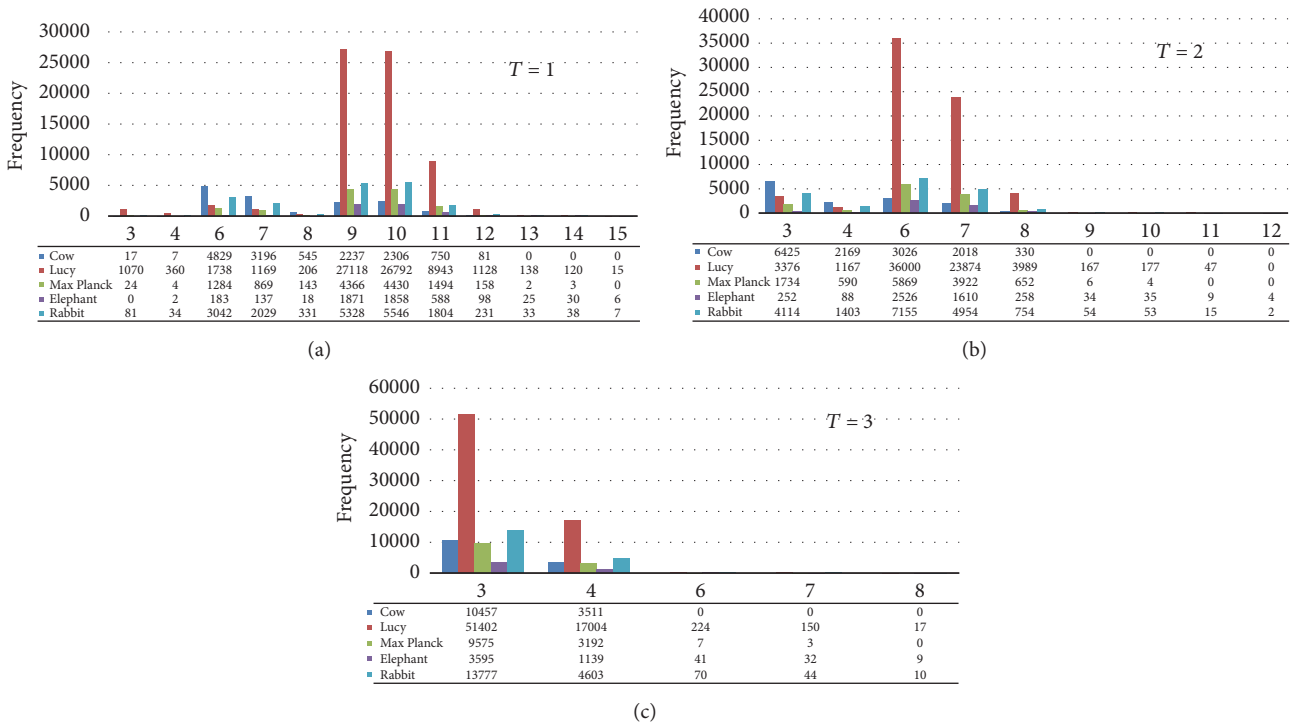



Figure 6: Embedding capacity distribution for each vertex under different thresholds.

Table 4: Comparison of previous and current methods.

| Type | Vertex-based | | | | Region-based | | |
|---|---|---|---|---|---|---|---|
| Algorithm | [9] | [11] | [12] | [8] | [10] | [14] | Our approach |
| Publicity | Semiblind | Blind | Blind | Blind | Blind | Semiblind | Blind |
| Embedding rate | 100% | 100% | 100% | 13.47% | 34.08% | N. A. | 26.85% |
| Embedding capacity | 1 | 48 | 3 | 1 | N. A. | <1 | 3.27–8.95 |
| Adaptability | No | No | No | No | No | No | Yes |
| Distortion control | Yes | Yes | No | Yes | Yes | No | Yes |
| Robustness | Vertex reorder | ST | Vertex reorder | ST | ST | Vertex reorder | ST |
| Method | QIM | QIM | Hamming code | QIM | QIM | Iterative perturbation | Message-digit substitution |

of tampered region localization. With respect to publicity, only the algorithms proposed by Huang et al. [9] and Yeo and Yeung [14] require additional overhead information for authentication, and others are blind. With respect to the embedding rate, each vertex in a vertex-based algorithm is embedded with an authentication code, and the embedding rate is equal to 100%; however, a region-based algorithm considers local complexity, and the embedding rate is clearly lower. Approximately 26.85% of all vertices in our proposed algorithm can be embedded with an authentication code. Because the embedding unit is a triangle in the algorithm proposed by Molaei et al., indicating that three vertices are involved in data embedding for each mark triangle, more vertices are marked than our proposed algorithm. Consequently, their scale of tampered region localization is also larger than ours. With respect to embedding capacity, the algorithm proposed by Wang et al. [13] provides the highest embedding capacity (up to 48 bits per vertex), owing to their use of IEEE 754 single-precision floating-point format for vertex coordinate representation and the two-level embedding strategy. Our proposed algorithm achieves the highest embedding capacity when applying the region-based tamper detection algorithm, and only our proposed algorithm achieves adaptive embedding capacity. With respect to robustness, each algorithm can resist either ST or vertex reordering attacks. Such characteristics render our proposed algorithm feasible in the field of fragile watermarking.

## 5. Conclusions and Future Work

The present study proposes an innovative region-based tamper detection algorithm for 3D polygonal models based on local geometrical properties. The proposed algorithm uses a vertex traversal method to categorize all vertices as either embeddable or reference vertices. The authentication code

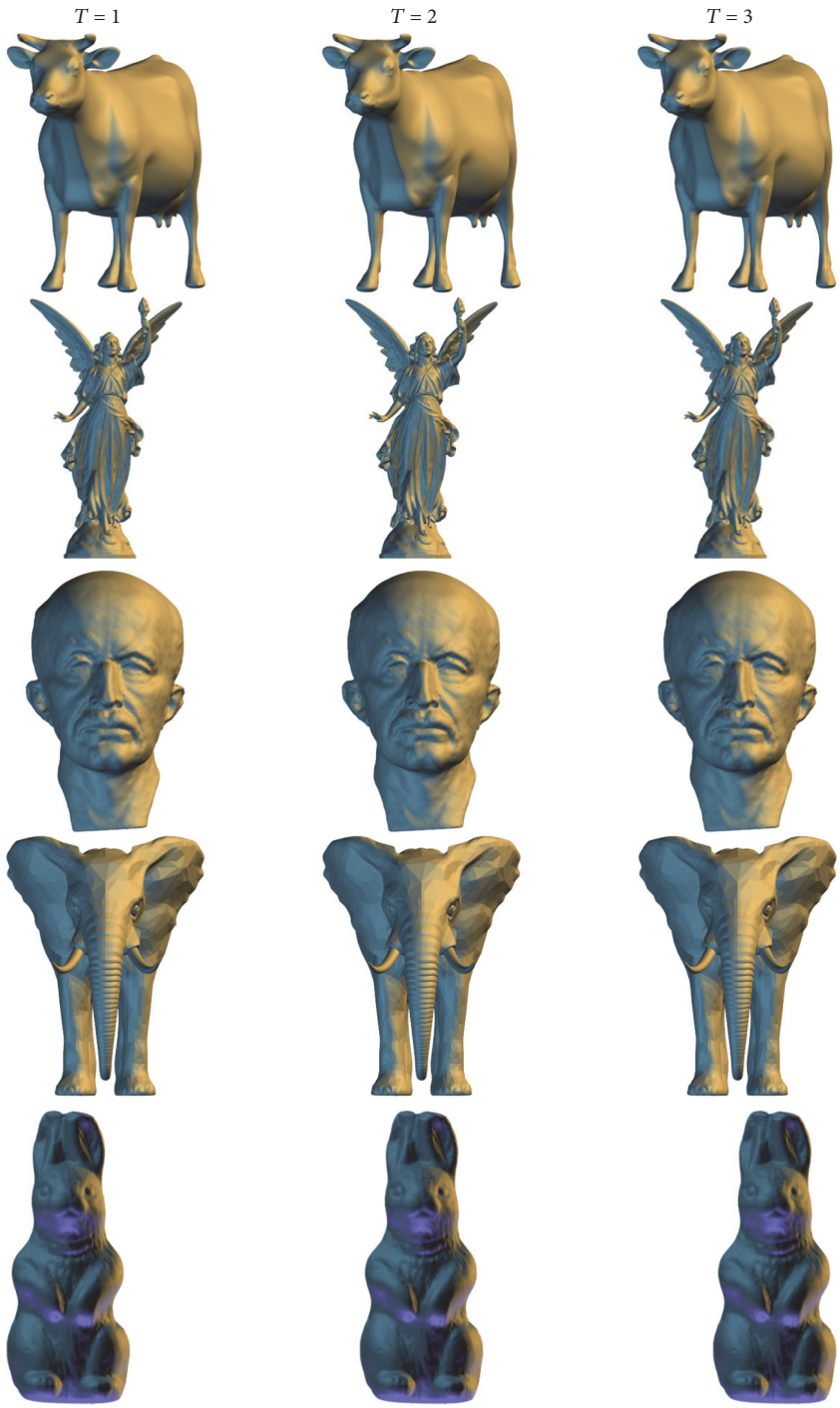



Figure 7: Visual effects of data-embedded models under different embedding thresholds.

is generated by feeding the number of neighboring vertices and local geometrical properties of the embeddable vertex into the hash function. The hash result is then embedded into the embeddable vertex by simple message-digit substitution. The proposed algorithm is simple and distortion-controllable and possesses a higher and more adaptive embedding capacity and a higher embedding rate than most existing region-based tamper detection algorithms. The experimental results demonstrate the feasibility of the proposed algorithm.

Although the proposed algorithm was proven to yield satisfactory results, additional improvements could be made. Currently, approximately only 27% of vertices can be used for authentication code embedding; our vertex decimation scheme [17] could be adopted to increase this percentage. Furthermore, the ratio used for authentication code embedding is too small, causing a lower embedding capacity; other ratios could be employed for achieving a higher embedding capacity. In addition, it would also be worth extending the

Figure 8: Visual effects of detection results for different 3D polygonal models.

proposed algorithm to support point geometries. Finally, real-time implementation for the proposed algorithm could be developed in the future.

## Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

This work was supported by Ministry of Science and Technology of Taiwan under Grant nos. MOST 104-2221-E-468-016 and MOST 105-2221-E-468-019.

## References

[1] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn, "Information hiding—a survey," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1062–1078, 1999.

[2] C.-W. Lee and W.-H. Tsai, "A data hiding method based on information sharing via PNG images for applications of color image authentication and metadata embedding," *Signal Processing*, vol. 93, no. 7, pp. 2010–2025, 2013.

[3] C.-C. Lo and Y.-C. Hu, "A novel reversible image authentication scheme for digital images," *Signal Processing*, vol. 98, pp. 174–185, 2014.

[4] T.-S. Nguyen, C.-C. Chang, and T.-F. Chung, "A tamper-detection scheme for BTC-compressed images with high-quality images," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 6, pp. 2005–2021, 2014.

[5] C.-N. Yang, J.-F. Ouyang, and L. Harn, "Steganography and authentication in image sharing without parity bits," *Optics Communications*, vol. 285, no. 7, pp. 1725–1735, 2012.

[6] C. Abhayaratne and D. Bhowmik, "Scalable watermark extraction for real-time authentication of JPEG 2000 images," *Journal of Real-Time Image Processing*, vol. 8, no. 3, pp. 307–325, 2013.

[7] T.-Y. Chen, M.-S. Hwang, and J.-K. Jan, "Adaptive authentication schemes for 3D mesh models," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12, pp. 4561–4572, 2009.

[8] C.-M. Chou and D.-C. Tseng, "Affine-transformation-invariant public fragile watermarking for 3D model authentication," *IEEE Computer Graphics and Applications*, vol. 29, no. 2, pp. 72–79, 2009.

[9] C.-C. Huang, Y.-W. Yang, C.-M. Fan, and J.-T. Wang, "A spherical coordinate based fragile watermarking scheme for 3D models," in *Recent Trends in Applied Artificial Intelligence: 26th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2013, Amsterdam,The Netherlands, June 17–21, 2013. Proceedings*, vol. 7906, pp. 566–571, Springer, Berlin, Germany, 2013.

[10] A. M. Molaei, H. Ebrahimnezhad, and M. H. Sedaaghi, "A blind fragile watermarking method for 3D models based on geometric properties of triangles," *3D Research*, vol. 4, no. 4, pp. 1–9, 2013.

[11] 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.

[12] J.-T. Wang, Y.-C. Chang, C.-Y. Yu, and S.-S. Yu, "Hamming code based watermarking scheme for 3D model verification," *Mathematical Problems in Engineering*, vol. 2014, Article ID 241093, 7 pages, 2014.

[13] W.-B. Wang, G.-Q. Zheng, J.-H. Yong, and H.-J. Gu, "A numerically stable fragile watermarking scheme for authenticating 3D models," *Computer Aided Design*, vol. 40, no. 5, pp. 634–645, 2008.

[14] 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.

[15] P.-C. Wang and C.-M. Wang, "Reversible data hiding for point-sampled geometry," *Journal of Information Science and Engineering*, vol. 23, no. 6, pp. 1889–1900, 2007.

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

[17] Y.-Y. Tsai, "An adaptive steganographic algorithm for 3D polygonal models using vertex decimation," *Multimedia Tools and Applications*, vol. 69, no. 3, pp. 859–876, 2014.