Received October 2, 2019, accepted October 15, 2019, date of publication November 6, 2019, date of current version November 18, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2952058

Adaptive Unified Data Embedding and Scrambling for Three-Dimensional Mesh Models

LIU-YAO HAO[1], BIN YAN[1], (Member, IEEE), JENG-SHYANG PAN[2], (Senior Member, IEEE), NA CHEN[1], HONG-MEI YANG[2], AND MOSES ARHINFUL ACQUAH[1]
[1]College of Electronic and Information Engineering, Shandong University of Science and Technology, Qingdao 266590, China
[2]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China

Corresponding author: Bin Yan (yanbinhit@hotmail.com)

This work was supported in part by the MOE (Ministry of Education in China) Project of Humanities and Social Sciences under Grant 18YJAZH110, in part by the National Natural Science Foundation of China (NSFC) under Grant 61272432, and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2014JL044.

ABSTRACT Traditionally, unified data embedding and scrambling techniques have been designed for grayscale images, which cannot be applied directly to a three-dimensional (3D) mesh. Recently, the universal use of 3D technology inspired us to innovate in this field. In this paper, an adaptive unified data embedding and scrambling technique for 3D mesh models (3D-AUES) is proposed, which can embed external data and scramble 3D mesh simultaneously. First, a vertex coordinate prediction method called cross prediction is adopted to accurately predict half of the vertices from the other half. The predicted vertices are used to embed external data. We further increase the embedding rate by bit replacement embedding. Then, to improve security, we propose an adaptive threshold to select vertices for embedding. To ensure lossless scrambling, the thresholds and prediction errors are embedded as side information with secret information into the vertices. By adopting an adaptive threshold and multilayer embedding, scalable scrambling quality can be achieved. On the decoder side, with the help of losslessly embedded side information, external data can be successfully extracted, and the original mesh can be restored to predetermined distortion levels, from lossless recovery to partial recovery. Experiments show that 3D-AUES has a high embedding rate, scalable scrambling quality and scalable recovery quality.

INDEX TERMS Adaptive threshold, bit replacement, reversible data hiding, three-dimensional mesh models, unified embedding-scrambling.

I. INTRODUCTION

With the rapid development of 3D printing and 3D modeling, the spread of 3D models has accelerated. Applications of 3D models are found in diverse fields, including medical 3D models [1], military geography 3D models, chemical molecular structure 3D models, 3D mechanic models for 3D printing, and 3D character scenes in movies [2]–[5].

Among these applications, 3D models for 3D printing and 3D manufacturing are valuable due to intensive efforts and investment in designing them. Currently, cloud manufacturing is expanding where designer and product companies are connected to the manufacturing factory through cloud storage and cloud computing [2]. This connection offers the designer and low user costs and fast prototyping, but it may introduce copyright violations to 3D models. A 3D model may be easily copied from cloud storage [3], [4]. To ensure secure cloud storage, 3D models are usually encrypted before uploading to the cloud. Furthermore, some authentication and copyright information also need to be inserted into the encrypted model, usually using reversible data hiding (RDH).

A typical application scenario in secure cloud storage is illustrated in Fig. 1. Before uploading, a 3D mesh model can be protected through scrambling in service provider 1 (such as a cloud storage client installed on the user's desktop computer), and then it is transmitted to cloud server 1, where the cloud administrator can manage the file of the scrambled mesh. Later, the user may need to download the file

The associate editor coordinating the review of this manuscript and approving it for publication was SK Hafizul Islam.

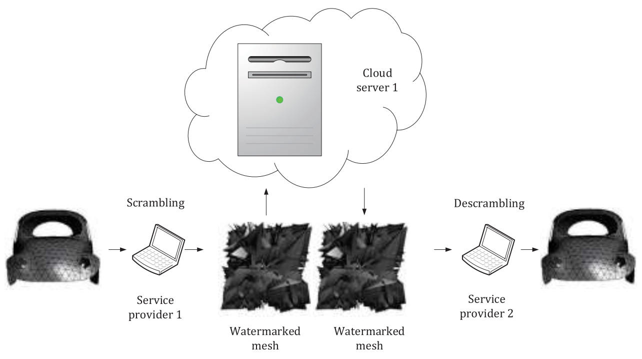


FIGURE 1. A typical application scenario of secure cloud storage.

from cloud server 1 to another service provider, i.e., service provider 2, such as another cloud storage client installed on his/her laptop. Then, service provider 2 needs to extract the external data and recover the original 3D mesh.

The application scenario of the unified algorithm is different from the common application scenario of RDH in the encrypted domain. For a typical application scenario, the user completely encrypts the mesh and then uploads it to the cloud. To facilitate the management of images, the cloud service managers embed additional data, such as a tag to mark the source of the file. Thus, additional data are embedded by cloud service managers, and mesh encryption and data encryption use different keys. At the receiving end, the user can extract the data and restore the mesh. For the unified approach, the mesh is first scrambled by the user using cloud storage client software before uploading to the cloud server. Thus, mesh scrambling and payload encryption and embedding are implemented in the cloud storage client. The payload is divided into two parts. The threshold is encrypted along with the original 4 LSBs of the nonembedded set, and the external secret information (such as user information) is encrypted. Cloud server 1, using the external secret information key, can visit the user information to facilitate managing the mesh model. At the receiving end, the user may download the mesh from the cloud, extract the data from the client, and finally restore the mesh.

The application scenario in Fig. 1 imposes the following two requirements on the secure cloud storage system: (1) different access levels for the service provider and cloud administrator, and (2) scalable quality for the scrambled 3D mesh model. The service provider has a higher access level so that it can both extract the external data and recover the original mesh model. In contrast, the cloud administrator has a lower access level. The administrator needs to have access to the external data (such as authentication and copyright information) for managing the model database without recovering the 3D mesh model. Imposing scalable quality for the scrambled model requires that the service provider be able to control the perceptual quality of the scrambled mesh. This feature is very valuable because the scrambled mesh can be presented to users with diverse perceptual quality. A lower quality scrambled model can be shown to potential buyers and can be later restored to high quality after purchase.

To address the above two requirements, reversible data hiding in the encryption domain (RDH-ED) provides a possible solution. Encryption and RDH were initially developed to solve different multimedia security problems. Encryption aims to scramble the contents [6]–[10], while RDH aims to nonperceptibly and reversibly embed secret information into a multimedia signal [11]–[13]. RDH-ED was developed from RDH for grayscale images [14]–[16] by Zhang [17]–[19]. However, it cannot be directly applied to mesh models because the mesh model features vertex and face instead of color. Although there are some available works on RDH for 3D models [20]–[23], it cannot be applied directly to the encrypted 3D mesh models. For encrypted 3D mesh models, the correlation between adjacent vertex coordinates is destroyed. Thus, one cannot predict coordinates from its neighbors. This problem forced researchers to develop RDH algorithms for an encrypted 3D mesh. Jiang *et al.* proposed RDH in encrypted 3D mesh models, which encrypted the mesh by a bit-stream encryption technique [24]. Then, they manipulated the LSBs to embed external data. However, in Jiang's algorithm, scrambling and data hiding are separated and not jointly designed. This affects the embedded capacity of the data hiding phase, although data embedding does not affect the recovery of the original mesh. The mesh after embedding the external data is completely incomprehensible, which fails to meet the second requirement for the application scenario in Fig. 1. Given these problems, we propose an adaptive unified data embedding and scrambling algorithm for the 3D mesh model (3D-AUES).

The proposed 3D-AUES is designed and partially inspired by Rad *et al.*'s work for grayscale images. More importantly, it incorporates two unique features of the mesh model: (1) different prediction domains than the digital image, and (2) different numerical representations than the digital image.

1) **Prediction domain:** Unlike the image signal, the 3D mesh model has no range but only domain, which is the set of vertices. The domain of the image signal is a regularly spaced grid in 2D, and its range is [0, 2^d-1] for a d-bits quantized image. In contrast, the 3D mesh model has an irregular domain and no range. The correlation between adjacent vertices is weaker than the correlation between two pixels in an image. As a consequence, The relative prediction error for vertex coordinates is usually larger than for pixel values.
2) **Numerical representation:** The vertex coordinates are represented by a floating-point number with positive and negative values. In contrast, digital images are usually represented by integer numbers. The floating number creates new issues in coordinate prediction and mesh recovery.

The algorithm Rad *et al.* proposed is only suitable for pixels of a grayscale image, which cannot address the two issues outlined above. In contrast, our proposed 3D-AUES is designed considering the two contrasting features of the 3D mesh model. First, a vertex coordinate prediction method called cross prediction is adopted to accurately predict the vertices. Through the selection of the threshold, we determine the coordinates of the embedded vertices. Then, the embedded vertices are used to embed external data by bit replacement. To ensure lossless scrambling, the threshold and prediction error are embedded as side information with secret information into the vertices. By adopting an adaptive threshold, multilayer embedding and scalable scrambling quality can be achieved. According to the help of losslessly embedded side information, the external data can be successfully extracted, and the original mesh can be restored to predetermined distortion levels, from lossless recovery to partial recovery. Experiments show that 3D-AUES has a high embedding rate, scalable scrambling quality, and scalable recovery quality.

The structure of the rest of this paper is organized as follows. Related works are reviewed in Section II, including RDH algorithms for 3D mesh models, RDH algorithms in the encryption domain, and the RDH algorithms for 3D encrypted meshes. The algorithm we propose is elaborated in Section III. Experimental results are presented and discussed in Section IV, which includes experimental setups, performance metrics, and various tests. Finally, we conclude our paper in Section V.

II. RELATED WORKS

In this section, we briefly review the necessary background for the proposed algorithm, including RDH for the 3D mesh model, RDH in the encryption domain and scrambling, and RDH in the 3D encryption domain. Next, we describe them separately in detail.

A. RDH FOR THE 3D MESH MODEL

First, we describe the structure and representation of a 3D mesh model. Then, existing 3D RDH in the literature is reviewed, along with a brief discussion of the difference between RDH and unified scrambling and RDH.

As discussed in Section I, a 3D mesh model has an entirely different structure than a digital image. The 3D mesh model is composed of two lists: a list of vertices and a list of faces that characterize the topology of the model. To hide external data in the 3D mesh model, one cannot modify the sample values because the mesh model has no signal defined on each vertex. Instead, the external data are embedded by modifying the coordinates of the vertices, the length of the line segment connecting two vertices, or the area of the polygon face.

Available RDH algorithms for 3D mesh models are very few compared to RDH for digital images. To increase the robustness and capacity of steganography for 3D models, various RDH algorithms are proposed from different perspectives [25]–[27]. The existing algorithms can be classified into three categories: 1) 3D RDH in the spatial domain, 2) 3D RDH in the transform domain, and 3) 3D RDH in the compressed domain.

1) **3D RDH in the spatial domain:** The spatial domain RDH algorithm has the advantage of ease of design and low complexity. This makes it the current mainstream method. Wu first proposed an algorithm by modifying the distance from the face centroid to the mesh centroid [28]. Nonetheless, the original mesh can only be approximately reconstructed. A second algorithm proposed by Wu was based on the prediction-error expansion (PEE) of the vertex coordinate, which is completely reversible [29]. Zhu *et al.* also proposed an algorithm based on prediction-error expansion [30]. Half the distances from the vertices to the centroid are used for error expansion, and the distortion is lower than Wu's algorithm. Molaei *et al.* proposed a blind fragile 3D RDH algorithm [31], where the secret information is embedded into the median of the three sides of the marked triangle instead of embedding the information into the vertices. While robust 3D RDH is used to protect digital content, fragile 3D RDH is used to authenticate digital content. An RDH algorithm based on histogram shifting (HS) was proposed in [32]. The histogram construction was based on the normalized distance difference of adjacent vertices. This method can resist similar transformation attacks. Zhang *et al.* proposed a new algorithm using a hybrid prediction and a multilayer strategy [33]. He also proposed an improved method based on PEE and sorting [34]. The algorithm uses a ring pattern prediction to improve prediction accuracy and sorting. Furthermore, a two-layer strategy is employed to increase capacity and reduce distortion. The feature of the spatial domain algorithm is that it can hide a large quantity of data by making an imperceptible change to the original mesh. The computational complexity is lower. However, the algorithms are incredibly fragile. If the watermarked mesh undergoes a slight change, the hidden information may be lost entirely.
2) **3D RDH in the transform domain:** To address the robustness problem, some researchers have proposed different RDH algorithms in the transform domain. In 2006, Luo *et al.* proposed an RDH algorithm in the transform domain [35], which recorded 8 adjacent vertices into a cluster, and performed integer DCT (discrete cosine transform) on these 8 vertices. The external data were embedded by modifying the high-frequency coefficients. A wavelet transform of hierarchical mesh was proposed in [36]. Through wavelet decomposition, three categories of secret information were inserted into the mesh at different resolution levels. Feng *et al.* proposed a double RDH algorithm, which is based on feature segmentation with DCT transform and redundancy information [37]. These two RDH algorithms do not interfere with each other, which can improve robustness and resilience to intentional attacks. The feature of the transform domain algorithm is that the secret information was embedded by changing some of the transform domain coefficients. Compared with the spatial domain algorithm, the transform domain

algorithm is more robust, and the secret information is less detectable. However, the capacity is relatively smaller than that of spatial domain algorithms, and the complexity of the algorithm is usually higher.

3) **3D RDH in the compressed domain:** For compressed domain RDH, some redundancies are removed by the compression algorithm. Thus, RDH in the compressed domain is more challenging. Moreover, RDH can be combined with the compression process. Sun *et al.* proposed an algorithm in compressed domains [38], where the external data are embedded into a PVQ (predictive vector quantization) compressed stream by modifying the prediction mechanism. In 2007, Lu also proposed an algorithm in the compressed domain [21], where the external data were embedded, and the hash of the original mesh was hidden during PVQ compression. It should be noted that 3D RDH in the compressed domain is designed for a given type of compression algorithm or file format, so the hidden data may be completely destroyed if the file is converted to other compression formats.

Finally, we note that the available RDH algorithms cannot be directly applied to scrambling mesh models because the requirements for RDH and scrambling are quite different. The RDH requires that less distortion is introduced to the mesh after embedding. In contrast, scrambling requires the mesh to be severely distorted after embedding the external data. For applications requiring RDH and scrambling, joint RDH and scrambling are more appropriate (or RDH in encryption domain).

B. RDH IN THE ENCRYPTION DOMAIN

RDH in the encryption domain (RDH-ED) can be classified into two classes: separable RDH and scrambling and unified scrambling and RDH.

For separable RDH and scrambling, the scrambling and RDH are separately designed and implemented by different parties [39], [40]. For example, in the cloud storage scenario, scrambling is implemented by the client-side software (i.e., service provider 1), and the cloud administrator implements RDH. This creates an apparent disadvantage in RDH because the redundancy required by RDH is partially or wholly destroyed by scrambling. In other words, the scrambling step is not aware of the later RDH step. As a result, the embedding capacity of RDH is significantly lower than RDH for a nonencrypted image. For example, Zhang's algorithm provides an effective payload of approximately 10^{-4} bpp (bits per pixel), and Fujiyoshi's algorithm provides a useful payload of approximately 1.9×10^{-2}. Furthermore, the quality of the scrambled image is not scalable. Both Zhang's algorithm and Fujiyoshi's algorithm fully scramble the original image.

For unified scrambling and RDH, such as [41], [42], the scrambling step and RDH step are jointly designed. This creates two advantages: higher embedding capacity and scalable quality of the scrambled image. The RDH step can explore the redundancy of the original image and thus can obtain higher embedding capacity. During unified scrambling and embedding, one may choose an embedding parameter to control the degree of scrambling. For example, in [42], the controlling parameter was a threshold applied on prediction error, which determined the number of pixels to scramble. Scalability is a desired characteristic when the owner of the image wants to draw the attention of the potential customer by showing him/her the partially scrambled image.

In [41], Ong *et al.* proposed a unified scrambling and RDH algorithm, which is reversible and scalable. However, its reversibility is not controllable. In contrast, the UES (unified data embedding and scrambling) algorithm proposed in [42] can provide scalability and controllable reversibility. The UES algorithm uses checkerboard-based prediction (CBP) to predict the pixel value [42]. The prediction error is encoded, and the predicted pixels are classified according to the threshold. Those pixels whose prediction error is below the threshold are selected to embed the external data.

Although UES can provide scalability and controllable reversibility, owing to the structural differences between the 3D mesh and the grayscale image, it cannot be applied directly to 3D mesh models.

C. RDH IN 3D ENCRYPTION DOMAIN

The only available RDH for 3D encrypted mesh is a recent work published by Jiang *et al.* [24], which is separable and nonscalable. Because this algorithm is strictly relevant to our work, we review it. This algorithm consists of two parts: encoding (preprocessing, encryption, data embedding) and decoding (decryption, data extraction and recovery).

To start the encoding, the vertex coordinates are preprocessed into integers. The original coordinates are represented by floating-point numbers that are difficult to handle. To convert the coordinates to integers, the floating-point number is first truncated to the n-the position after the decimal point. Then, it is amplified by 10^n to obtain integer numbers.

Next, the coordinates are encrypted by a stream cipher with key K_e. Let the coordinate of a vertex be represented by $b_{i,j,0}, b_{i,j,1}, b_{i,j,2}, b_{i,j,3}, \ldots b_{i,j,m}$, where $b_{i,j,q}$ is the q-th bit in representing the j-th dimension of the i-th vertex, where $j \in \{x, y, z\}$. Then, the encrypted coordinate is:

$$d_{i,j,q} = b_{i,j,q} \oplus r_{i,j,q}, \quad q = 0, \ldots, m \tag{1}$$

where $r_{i,j,q}$ is generated by a pseudorandom number generator (PRNG).

To embed data, the vertices are classified into two sets: embedded set and referenced set, denoted as $\mathcal{P}_e$ and $\mathcal{P}_r$, respectively. The data to be embedded are first encrypted by key K_h and then embedded into a vertex in $\mathcal{P}_e$:

$$d'_{i,j,q} = d_{i,j,q} \oplus b, \ j = x, y, z, \quad q = 0, \ldots, m, \ i \in P_e \tag{2}$$

where $d'_{i,j,q}$ denotes the embedded mesh. No modification is applied to each vertex in $\mathcal{P}_r$.

The decoding part consists of decryption, data extraction and mesh recovery. First, the encrypted and embedded mesh is decrypted using the same stream cipher and key as in (1). Then, data extraction and mesh recovery can be accomplished using a spatial correlation of the 3D model itself.

In summary, the proposed 3D-AUES is different from Jiang's algorithm in the following aspects:

1) Jiang's algorithm is separable, while the 3D-AUES algorithm is unified.
2) Jiang's algorithm is reversible, while the 3D-AUES algorithm has controllable reversibility.
3) The data embedding part of the 3D-AUES algorithm is not affected by encryption, which may reduce the redundancy needed for embedding, so the embedding rate is higher than Jiang's algorithm.

We conclude this section by remarking that Jiang's algorithm is an extension of separable scrambling and RDH for grayscale images. Hence, it inherits the disadvantages of this type of algorithm, such as low embedding capacity and nonscalability. Given this problem, we propose an adaptive UES algorithm for a 3D mesh model, which is unified, scalable and has controllable reversibility.

III. PROPOSED SCHEME

To address the above requirements in Section II, we designed a 3D-AUES algorithm, which is unified, scalable and has controllable reversibility. The 'unified' feature is implemented so that the mesh is scrambled, whereas the external data are embedded. To achieve scalability, the algorithm is designed so that only partial vertex coordinates can be disturbed by adaptively choosing a threshold parameter μ. The reversibility is controlled implicitly by the step size of the quantizer for prediction error, which is implemented explicitly by setting the parameter μ and the number of bits that are used to represent the prediction error. To implement the above features, our 3D-AUES encoder and decoder include the following functional units: preprocessing, vertex coordinates prediction, 3D-AUES, data extraction and mesh reconstruction, which are described in detail in the subsections below.

The overall block diagram of the proposed scheme is illustrated in Fig. 2, where S is external secret information and μ is a threshold. The external secret information S and threshold μ are called *external data*, which is embedded into the mesh. As shown in Fig. 2, three parties are involved: the service provider, the cloud administrator and the user.

Before uploading to the cloud, first, service provider 1 (such as a cloud storage client installed on the user's computer) preprocesses the original mesh **O** to **M**. Next, the scrambled mesh **W** is obtained by data embedding and scrambling. Then, **W** is sent to the cloud administrator. The cloud administrator can manage **W** by accessing the external data embedded in **W** (such as authentication information) even without knowing the content of **O**. The mesh **W** is transmitted to service provider 2. This can be done by downloading from the cloud using a cloud storage client.

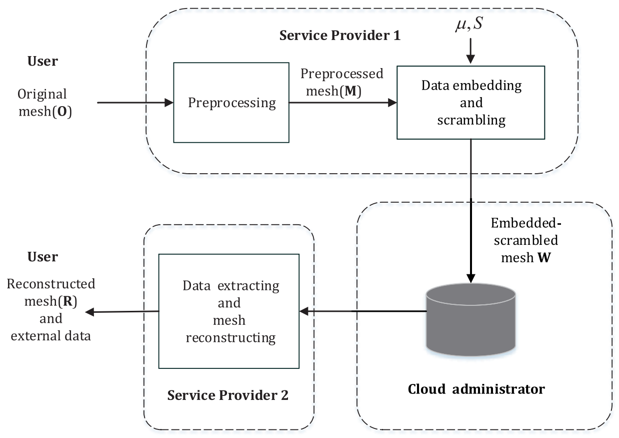

User
Original mesh(**O**)
Service Provider 1
Preprocessing
Preprocessed mesh(**M**)
μ, S
Data embedding and scrambling
Embedded-scrambled mesh **W**
Cloud administrator
User
Reconstructed mesh(**R**) and external data
Data extracting and mesh reconstructing
Service Provider 2

FIGURE 2. Block diagram of the proposed scheme.

Service provider 2 can then extract the external data and reconstruct the original mesh. We describe this process in detail below.

A. PREPROCESSING

The purpose of preprocessing is to convert the mesh vertex coordinates to integers to facilitate embedding and scrambling in later steps. The vertex coordinates are usually represented by floating-point numbers and may have different ranges due to the diverse source of capturing devices. This imposes difficulty in coordinate prediction, mesh scrambling, and mesh recovery because one may need to determine appropriate parameters for different input ranges. Due to the composition characteristics of 3D mesh, to facilitate subsequent operations, the coordinate values should be truncated and normalized.

Before elaborating on the preprocessing method, we first describe the composition of a 3D mesh model. The 3D mesh model is mainly composed of spatial vertices and triangle faces. The triangles are composed of vertices, and all triangles form a complete 3D mesh model. The set of vertices is represented as $\mathcal{V} = \{\boldsymbol{V}_1, \boldsymbol{V}_2, \cdots \boldsymbol{V}_M\}$, where each vertex coordinate contains three components, i.e., $\boldsymbol{V}_i = \{x_i, y_i, z_i\}$. For the convenience of future processing, one can organize all the vertex coordinates as a matrix with dimensions $3 \times M$, where M is the number of columns. In the 3D mesh model, all the connections between vertices represent the topology. The two vertices connected by an edge are called neighbors. One vertex may have multiple neighbors. By traversing all the faces containing a given vertex, all the neighbors of this vertex can be obtained.

There are many storage formats for 3D mesh, such as OFF, OBJ, and PLY. In these files, vertices and faces have an implicit order. For example, the file formats of the vertices and faces are shown in Table 1 and Table 2, respectively. Each vertex has an implicit order in the vertex list (column 1 in Table 1). After the mesh is scrambled, the implicit order of vertex remains the same. Only the coordinate values are changed.

TABLE 1. File format for vertices in a 3D mesh.

Index of vertex (i)	x_i	y_i	z_i
1	x_1	y_1	z_1
2	x_2	y_2	z_2
3	x_3	y_3	z_3
4	x_4	y_4	z_4
5	x_5	y_5	z_5
6	x_6	y_6	z_6
...	...	...	...
...	...	...	...
M	x_M	y_M	z_M

TABLE 2. File format for faces in a 3D mesh.

Index of face (q)	f_{q1}	f_{q2}	f_{q3}
F_1	f_{11}	f_{12}	f_{13}
F_2	f_{21}	f_{22}	f_{23}
F_3	f_{31}	f_{32}	f_{33}
F_4	f_{41}	f_{42}	f_{43}
F_5	f_{51}	f_{52}	f_{53}
F_6	f_{61}	f_{62}	f_{63}
...	...	...	...
...	...	...	...
F_N	x_{N1}	y_{N2}	z_{N3}

For preprocessing, generally, we assume that each vertex coordinate of the uncompressed mesh model is usually represented as the precision level of 10^{-m}. Deering [43] noted that some scenarios do not require very high precision so that a 32- or 64-bit float point representation can be reduced to 16 bits. Furthermore, the 3D printer has much lower precision than can be represented by the floating-point number. Thus, the vertex coordinates can be truncated to a lower precision.

Let one of the uncompressed vertex coordinates be $c \times 10^{-m}$, then the truncated result is $\hat{c} \times 10^{-n}$, where $n \leq m$ and $\hat{c}$ retain up to n significant digits of c behind the decimal point. Then, it is normalized by dividing the quantization step 10^{-n}

$$\frac{\hat{c} \times 10^{-n}}{10^{-n}}.$$

For each vertex $\boldsymbol{V}_i$, the above truncation and normalization steps can be combined into

$$\boldsymbol{V}_i' = \lfloor \boldsymbol{V}_i \times 10^n \rfloor, \quad i = 1, \ldots, M \tag{3}$$

where $\boldsymbol{V}_i' = \{x_i', y_i', z_i', \}$. Let each normalized vertex coordinate be represented as a d-bits binary string, corresponding to integers between 0 and $2^d - 1$. As suggested in Jiang's work, we take $d \in [1, 33]$ [24]. The data embedding/extraction and mesh scrambling/reconstruction are all performed using the normalized coordinates.

After embedding and scrambling, we inversely transform the coordinates of the processed vertex $\tilde{\boldsymbol{V}}_i$ to $\tilde{\boldsymbol{V}}_i{}'$ with

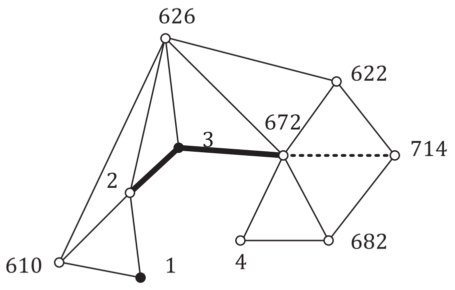


FIGURE 3. Illustration of vertex 672, where white dots represent set $\mathcal{P}$ and black dots represent set $\mathcal{R}$.

precision 10^{-n}:

$$\tilde{\boldsymbol{V}}_i{}' = \tilde{\boldsymbol{V}}_i \times 10^{-n}, \quad i = 1, \ldots, M \tag{4}$$

which can then be written to a mesh file and shared to the cloud.

B. VERTEX COORDINATES PREDICTION

As with other RDH algorithms, redundancy should be explored to embed external data. For 3D mesh models, the available redundancy is the correlation of coordinates between neighboring vertices, which can be explored by vertex prediction. To improve the restoration quality of reconstructed meshes and to control the quality of scrambling, we adopt a cross prediction scheme, which is motivated by a cross prediction for RDH [44] because using cross prediction and adaptive thresholds, one can achieve multilayer scrambling with different degrees of scrambling effects. Cross prediction can accurately predict half of the vertices from the other half, which not only provides considerable redundancy but also guarantees the accuracy of prediction during the recovery process.

The coordinates are traveled in ascending order. According to the index of the vertex, vertex coordinates are divided into two categories: (1) the set of vertices with even indices is denoted as *predicted set* $\mathcal{P} \triangleq \{s \in \{1, \ldots M\}, s.t. \bmod (s, 2) = 0\}$. The vertices in the predicted set $\mathcal{P}$ are used to embed external data. (2) The set of vertices with odd indices is marked as the *referenced set*, denoted as $\mathcal{R} \triangleq \{t \in \{1, \ldots M\}, s.t. \bmod (t, 2) = 1\}$. The coordinates of the vertices in the referenced set $\mathcal{R}$ remain unchanged during embedding.

The process of cross prediction is described as follows. First, we define $D\left(\boldsymbol{V}_i', \boldsymbol{V}_j'\right), i, j \in \{1, \ldots, M\}$ as the number of hops from $\boldsymbol{V}_i'$ to $\boldsymbol{V}_j'$. We find all the faces containing the vertex $\boldsymbol{V}_i'$; that is, when $D\left(\boldsymbol{V}_i', \boldsymbol{V}_j'\right) = 1, i, j \in \{1, \ldots, M\}$, and $\boldsymbol{V}_j'$ is a neighbor of the vertex $\boldsymbol{V}_i'$. The set of neighbors of $\boldsymbol{V}_i'$ is denoted as

$$\mathfrak{N}_i = \left\{i \in \mathcal{P}, j \in \mathcal{R}, \text{ s.t. } D\left(\boldsymbol{V}_i', \boldsymbol{V}_j'\right) = 1\right\}.$$

We take vertex 672 as an example and obtain its neighbors $\boldsymbol{V}_j'$. As shown in Fig. 3, vertices 3, 4, 622, 626, 682, 714 are neighbors of vertex 672. Black dots represent the referenced set, whereas white dots represent the predicted set. One may notice that $D\left(\boldsymbol{V}'_{672}, \boldsymbol{V}'_{2}\right) = 2$ (the thickest line); hence, vertex 2 is not a neighbor of vertex 672. Similarly, $D\left(\boldsymbol{V}'_{672}, \boldsymbol{V}'_{714}\right) = 1$ (the dotted line), so vertex 714 is a neighbor of vertex 672.

Next, for every vertex in $\mathcal{P}$, its coordinates are predicted using the mean value of its neighbors, i.e., the prediction $\hat{\boldsymbol{V}}_i = \left\{\hat{x}_i, \hat{y}_i, \hat{z}_i\right\}$ is calculated as in (5)

$$\hat{x}_i = \left\lfloor \frac{\sum_{k\in\mathfrak{N}_i} x'_k}{|\mathfrak{N}_i|} \right\rfloor,\quad \hat{y}_i = \left\lfloor \frac{\sum_{k\in\mathfrak{N}_i} y'_k}{|\mathfrak{N}_i|} \right\rfloor,\quad \hat{z}_i = \left\lfloor \frac{\sum_{k\in\mathfrak{N}_i} z'_k}{|\mathfrak{N}_i|} \right\rfloor. \tag{5}$$

The prediction error $\boldsymbol{e}_i = \left\{e_{xi}, e_{yi}, e_{zi}\right\}$ is calculated as in (6).

$$\begin{aligned} e_{xi} &= x_i{}' - \hat{x}_i \\ e_{yi} &= y_i{}' - \hat{y}_i \\ e_{zi} &= z_i{}' - \hat{z}_i, \end{aligned} \tag{6}$$

where $x_i{}', y_i{}', z_i{}'$ are the coordinates of the preprocessed vertices, and $|\mathfrak{N}_i|$ denotes the cardinality of set $\mathfrak{N}_i$, i.e., the number of elements in it if the set is finite.

An example of the above calculation can be shown as in Fig. 4. Vertex 2 is the first vertex when traversing the predicted set $\mathcal{P}$. There are 3 faces containing vertex 2, which are composed of vertices 1, 2, 3, 610, and 626. Thus, vertices 1, 3, 610, and 626 are the neighbors of vertex 2. In the referenced set $\mathcal{R}$, the neighbors of vertex 2 are vertices 1 and 3. The predicted coordinates of vertex 2 and the predicted errors are calculated as in (7) and (8).

$$\begin{cases} \hat{x}_2 = \lfloor \frac{x_1{}'+x_3{}'}{2} \rfloor \\ \hat{y}_2 = \lfloor \frac{y_1{}'+y_3{}'}{2} \rfloor \\ \hat{z}_2 = \lfloor \frac{z_1{}'+z_3{}'}{2} \rfloor \end{cases} \tag{7}$$

$$\begin{cases} e_{x2} = x_2{}' - \hat{x}_2 \\ e_{y2} = y_2{}' - \hat{y}_2 \\ e_{z2} = z_2{}' - \hat{z}_2 \end{cases} \tag{8}$$

The vertex whose neighboring vertices are all with even indices is not predicted. Let such vertex be referred to as *isolated vertex* $\boldsymbol{V}_I$. Since $\boldsymbol{V}_I$ cannot be predicted, it is treated the same way as vertices in non-embedded set $\mathcal{N}$. According to Algorithm 1 (step 4 to step 8), for the vertex $\boldsymbol{V}_I$ we just replace the last 4 bits by '0000', and the remaining bits are kept unchanged. At the receiving end, when restoring the mesh, it is still necessary to predict the coordinates of the predicted set by the referenced set. During this prediction process, the receiver can also find the isolated vertices by checking their indices. Correspondingly, according to the recovery method of $\mathcal{N}$ in Section III-D, the vertex $\boldsymbol{V}_I$ can be recovered.

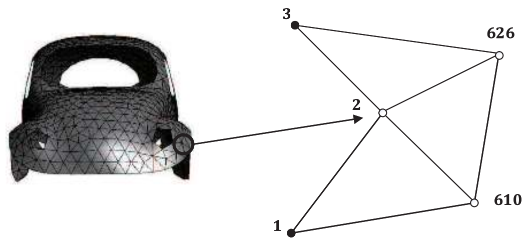


FIGURE 4. An example of a mesh (beetle).

TABLE 3. The number of isolated vertices for 22 mesh models.

Mesh	Number of isolated vertices $\boldsymbol{V}_I$	Total number of vertices (M)
VENUS	0	711
COW	1	726
HORSE	0	712
BEETLE	0	988
MUSHROOM	0	226
BIGNEFERTITI	0	4439
CAT1	10	4538
LEOPARD	6	5169
SKATEBOARD	9	6101
BIRD	13	6634
BIGCAMEL	21	8286
BIGCOW	14	9105
ENGINE	21	10576
DOG	30	10978
LARGEDOG	57	12515
PEPPER	28	14980
BIGMANNEQUIN	43	17662
HEAD	46	19052
FISH	34	21915
DAVID	61	23889
WATCH	2	30546
BUNNY	58	34835

Obviously, if a large number of isolated vertices exist in the mesh model, the embedding capacity could be significantly reduced. Fortunately, we found that for the models we tested, the number of isolated vertices are negligible compared to the total number of vertices M. For the 22 test models used in our experiments, the number of isolated vertices are listed in Table 3, along with the total number of vertices.

C. 3D ADAPTIVE UNIFIED EMBEDDING AND SCRAMBLING (3D-AUES)

To embed more external data, vertices with more considerable redundancy should be chosen. With the guidance of this principle, we should choose vertices whose prediction errors are smaller in magnitude. Therefore, the proposed 3D-AUES first classifies all vertices in $\mathcal{P}$ by their prediction error and an adaptive threshold μ. Then, only those chosen vertices are scrambled and embedded with the payload (including the secret information, the threshold and the prediction errors).

1) VERTEX COORDINATE CLASSIFICATION

The purpose of vertex coordinate classification is to select vertex coordinates with larger redundancy to scramble and embed data. According to the cross prediction scheme presented in Section III-B, we divide all vertex coordinates into two parts: predicted set $\mathcal{P}$ and referenced set $\mathcal{R}$. Based on the magnitude of the prediction error and a threshold, we can classify the vertex coordinates more precisely.

Let $e_{qi} \in \{e_{xi}, e_{yi}, e_{zi}\}$ be the prediction error of coordinate q for vertex $\mathbf{V}'_i$, where $q \in \{x, y, z\}$. Comparing the threshold μ with the magnitude of the prediction error e_{qi}, each coordinate q'_i, where $q \in \{x, y, z\}$, is evaluated to see if it is suitable for external data embedding. In particular, if the prediction error e_{qi} falls into a given interval

$$-\mu \le e_{qi} \le \mu, \tag{9}$$

where $\mu \in \mathbb{R}$, it is used for external data embedding and scrambling. Otherwise, it remains unchanged.

Using the selection rule in (9), the predicted set $\mathcal{P}$ is further divided into two subsets. (1) A set of embedded vertex coordinates $\mathcal{E}$, whose prediction error satisfies inequality (9). All vertex coordinates in this set are used for embedding external data. Thus, $\mathcal{E}$ can be formally defined as

$$\mathcal{E} \triangleq \{q'_i, \text{ where } q \in \{x, y, z\}\}, \\ i \in \{1, \ldots M\}, s.t. \bmod(i, 2) = 0, \text{ and } |e_{q,i}| \le \mu. \tag{10}$$

(2) A set of vertex coordinates $\mathcal{N}$, whose prediction error fails to satisfy inequality (9). This set is referred to as the nonembedded set $\mathcal{N}$:

$$\mathcal{N} \triangleq \{q'_i, \text{ where } q \in \{x, y, z\}\}, \\ i \in \{1, \ldots M\}, s.t. \bmod(i, 2) = 0, \text{ and } |e_{q,i}| > \mu. \tag{11}$$

The vertex coordinates of referenced set $\mathcal{R}$ are not used for embedding and scrambling. Hence, they retain their original values.

After vertex coordinate classification, we obtain a set of vertex coordinates $\mathcal{E}$, which is used for embedding and scrambling.

2) UNIFIED DATA EMBEDDING AND SCRAMBLING IN 3D MESH

In this section, we describe the unified data embedding and scrambling method in the 3D mesh. For simultaneously embedding data and scrambling coordinates in $\mathcal{E}$, we replace the d-bits representation of each coordinate as follows. For every $q'_i \in \mathcal{E}$:

1) The least α LSBs are replaced by the source coding result of the prediction error e_{qi}. To encode the prediction error, one must quantize the range $[-\mu, \mu]$. Here, we use a simple uniform quantizer and a simple fixed-length coding (α bits). If more bits are used such that $2^\alpha > 2\mu + 1$, then the prediction error can be losslessly compressed and recovered. Otherwise, the compression is lossy, and the prediction can only be approximately recovered [1]. Obviously, α is one of the influencing factors of mesh restoration and determines the coding accuracy of prediction error. Determining α is discussed in the subsection of rate-distortion-reversibility tradeoff.
2) The two MSBs are unchanged because the prediction precision of vertex coordinates is usually worse than the prediction of color as in digital images. This large prediction error can only be approximately quantized by the source coder and hence hinders the recovery of the original mesh model. To better recover the mesh, we fix the two MSBs during embedding and scrambling. Namely, the coordinates are only allowed to be scrambled within a given quadrant specified by the two MSBs. This may weaken the scrambling effect. However, because we are scrambling coordinates instead of color, the scrambling effect is not severely weakened by restricting to a quadrant.
3) The remaining $d - \alpha - 2$ bits are replaced by one segment from the payload.

[1]The ‘compression’ here controls the reversibility, not scalability. By compressing the prediction error using different quantization steps (or an equivalently different number of bits α used to encode the quantized error), various levels of reversibility (of the mesh) can be achieved.

All the coordinates in referenced set $\mathcal{R}$ are not modified. The decoder can recover this set by checking the parity of the vertex index. Similarly, all the coordinates in the nonembedded set $\mathcal{N}$ remain unchanged. However, the decoder cannot distinguish between the embedded set $\mathcal{E}$ and the nonembedded set $\mathcal{N}$. To help the decoder recognize $\mathcal{N}$, the last α LSBs are replaced by a flag pattern of all zeros. The original α bits are gathered into the payload and embedded into the mesh. This flag pattern, hence, is not used to encode prediction error in the last α bits of coordinates in $\mathcal{E}$.

To better illustrate the embedding and scrambling process, let us consider one specific example, as shown in Fig. 5. Each vertex coordinate is represented as a d-bit binary string. To increase embedding capacity and manage the restoration quality of the mesh, we take $\alpha = 4$ to encode the prediction error into $2^\alpha = 16$ levels, as shown in Table 6. The flag pattern ‘0000’ marks the α least significant bits of the nonembedded vertex coordinates $\mathcal{N}$ by bit replacement. The remaining $d - 4$ bits of the $\mathcal{N}$ remain unchanged. The code ‘0001’, ‘0010’, $\cdots$, ‘1111’ mark the α LSBs of the embedded vertex coordinates $\mathcal{E}$ by bit replacement. The 2 most significant bits store the main contents of the vertex coordinates. To facilitate the restoration of the original mesh, the 2 most significant bits of $\mathcal{E}$ remain unchanged. The remaining $d - 6$ bits of the $\mathcal{E}$ are used to embed the payload $\mathbf{p}$.

The payload $\mathbf{p}$ is constructed sequentially from three sources: (1) threshold μ; (2) the original α LSBs of the coordinates in $\mathcal{N}$; (3) the external secret information S. That is

$$\mathbf{p} = \left[\left[(\mu)_B \,||\mathrm{LSB}_\alpha(i), \forall i \in \mathcal{N}\right] \oplus \mathbf{r_e}\right] || \left[(S)_B \oplus \mathbf{r_s}\right], \tag{12}$$

where $(\mu)_B$ is the binary representation of μ, $\mathrm{LSB}_\alpha(i)$ is α LSBs of the coordinates i in $\mathcal{N}$, $(S)_B$ is binary representation of S. The operator $||$ represents bit sequence concatenation. The threshold μ along with the original 4 LSBs of the nonembedded set $\mathcal{N}$ of the payload $\mathbf{p}$ are encrypted by a pseudorandom sequence $\mathbf{r_e}$. The external secret information S

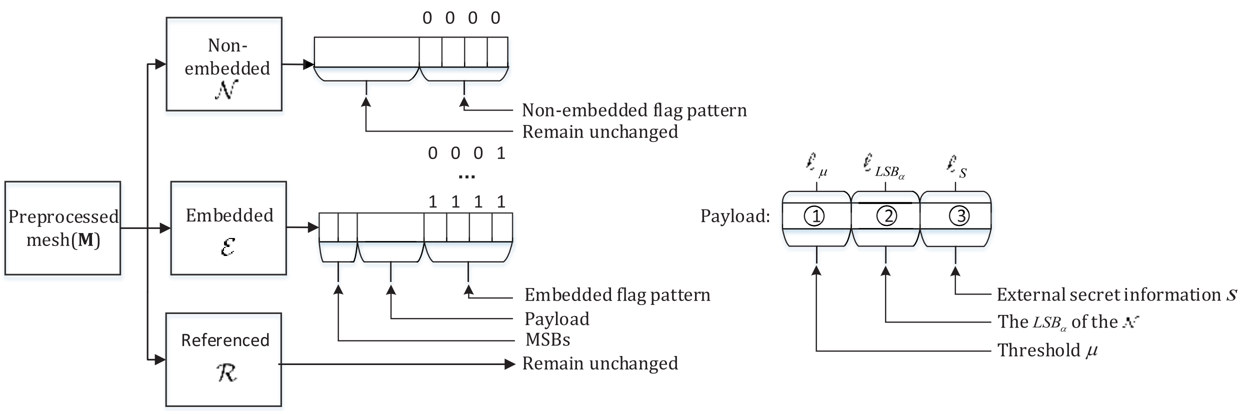


FIGURE 5. Block diagram of embedding and scrambling by bit replacement in 3D-AUES ($\alpha = 4$).

TABLE 4. Correspondence between 3D mesh vertices and indices for BEETLE.

Mesh before embedding				Mesh after embedding		
Index of vertex	x coordinate	y coordinate	z coordinate	x coordinate	y coordinate	z coordinate
1	-867893521	-317659621	1242027948	-867893521	-317659621	1242027948
2	-868710277	-407180807	1151518487	0	0	0
3	-867278804	-512080219	1079424725	-867278804	-512080219	1079424725
4	-862786647	-626426058	1023700472	0	0	0
5	-831216878	-711158045	934136729	-831216878	-711158045	934136729
6	-750921166	-668428810	845123221	0	0	0
...	...	...	...	...	...	...
29	-861617396	-404332908	-1597879940	-861617396	-404332908	-1597879940
30	-846344059	-495379708	-1691652126	0	0	0
...	...	...	...	...	...	...
51	872669392	-280595664	-1196139159	872669392	-280595664	-1196139159
52	872261014	-327146887	-1073006744	0	0	0
...	...	...	...	...	...	...
100	-865593708	-247242358	1604512857	0	0	0
...	...	...	...	...	...	...

of the payload $\mathbf{p}$ is encrypted by another pseudorandom sequence $\mathbf{r_s}$. The pseudorandom sequences $\mathbf{r_e}$ and $\mathbf{r_s}$ are generated by RC4 or the more secure eSTREAM algorithm [45]. As shown in Fig. 5, ℓ_μ, ℓ_{LSB_α} and ℓ_S are lengths of the binary bit with μ, α LSBs of the $\mathcal{N}$ and S, respectively. The detailed procedures of 3D-AUES are summarized in Algorithm 1.

In Algorithm 1, from step 4 to step 8, part of the coordinates in the nonembedded set $\mathcal{N}$ are scrambled. The 4 LSBs are replaced by '0000'. From step 10 to step 14, all the coordinates in the embedded set $\mathcal{E}$ are scrambled. The 4 LSBs are replaced by '0001, 0010, 0011, ..., 1111'. The remaining $d - 6$ LSBs are replaced by payload $\mathbf{p}$. The *replace* operation in Algorithm 1 achieves two goals: scrambling the original mesh coordinates and embedding the payload.

We distinguish between embedded vertices and referenced vertices based on index values. During the mesh scrambling, there is no change to the index values. Only the value of the vertex coordinate is changed. Table 4 shows the changes in the vertex coordinates and indices. All x, y, and z coordinate components whose index values are even are embedded with 0. After embedding, we traverse the three coordinate components whose index values are even. It can be found that these coordinate values are still 0. Therefore, when decoding, the 3D-AUES algorithm can ensure the same processing order as encoding.

Because the positions of the referenced set and the predicted set are fixed, we can achieve a second layer prediction on the mesh. The role of the referenced set $\mathcal{R}$ and the predicted set $\mathcal{P}$ are switched in the second layer prediction and embedding. However, we should note that because all coordinates in the set $\mathcal{P}$ are scrambled during first layer embedding, the prediction error is expected to be large when predicting the coordinates in the referenced set $\mathcal{R}$. Therefore, it is almost impossible to embed additional data during second layer embedding. However, it is still useful to further scramble the mesh. To increase the scrambling effect, the above operations can be repeated to complete the embedding of the third layer and more. The more layers we embed, the more blurry the mesh becomes.

3) DISTORTION MEASURE

To quantify and control the distortion of the scrambled mesh, an appropriate distortion measure should be adopted. For mesh processing, the quality of the processed mesh is usually measured by the 3D signal-to-noise ratio (3D SNR) [29], which is a measure of 3D geometric distortion. 3D SNR is the ratio between the signal strength and the noise strength, where the signal energy measures the signal strength, and the noise energy measures the noise strength.

For data embedding in the 3D mesh model, the coordinates are modified to encode the data. The SNR should reflect the distortion of shape due to this coordinate modification by using the relative change in coordinates to the original range of the coordinates. The change in coordinates is measured by the MSE between the modified/watermarked coordinates and the original coordinates, i.e., $\text{MSE}\left(\mathcal{V}, \mathcal{V}_w\right)$. To measure the range of the original coordinates, we use a quantity that reflects the size of the bounding box of the mesh model. For a mesh model with a larger size, most vertices should deviate far from the centroid of that model. Thus, we use the MSE between original vertices and the centroid of the model as a measure of the size of the original mesh model, i.e., $\text{MSE}\left(\mathcal{V}, \bar{\mathcal{V}}\right)$.

Let $\mathcal{V} = \{\mathbf{V}_1, \cdots, \mathbf{V}_M\}$ and $\hat{\mathcal{V}} = \left\{\hat{\mathbf{V}}_1, \cdots, \hat{\mathbf{V}}_M\right\}$ be the original and processed meshes, respectively. Then, the MSE

Algorithm 1 An Adaptive Unified Data Embedding and Scrambling Technique in 3D Mesh: 3D-AUES ($\alpha = 4$)

```
Input:
    The preprocessed 3D mesh $\mathbf{M}$; threshold $\mu$; external secret information $S$.
Output:
    Embedded and scrambled 3D mesh $\mathbf{W}$.
 1: The mesh $\mathbf{M}$ is divided into two categories according to the threshold parameter $\mu$: $\mathcal{R}$ and $\mathcal{P}$, where $\mathcal{P}$ is further divided into sets $\mathcal{E}$ and $\mathcal{N}$ according to inequality (9).
 2: Construct the payload $\mathbf{p}$ as follows.
 3: Convert the threshold $\mu$ to a $d$-bit binary string as the first part of the payload $\mathbf{p}$.
 4: for each coordinate in $\mathcal{N}$ do
 5:     Extract the last $\alpha$ LSBs as the second part of the payload $\mathbf{p}$.
 6:     Replace the last 4 bits by '0000'.
 7:     The remaining $d - 4$ bits remain unchanged.
 8: end for
 9: Append to $\mathbf{p}$ the external secret information $S$.
10: for each coordinate in $\mathcal{E}$ do
11:     Replace the last 4 bits by '0001', '0010', '0011' $\cdots$ '1111' according to Table 6.
12:     The two MSBs remain unchanged.
13:     Replace the remaining $d - 6$ bits by one segment of the payload $\mathbf{p}$.
14: end for
```

(mean squared error) between $\mathcal{V}$ and $\hat{\mathcal{V}}$ is defined as the total energy of error between them:

$$\text{MSE}\left(\mathcal{V}, \hat{\mathcal{V}}\right) = \sum_{i=1}^{M} \left[\left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2 + \left(z_i - \hat{z}_i\right)^2 \right], \quad (13)$$

where $\mathbf{V}_i = (x_i, y_i, z_i)$ and $\hat{\mathbf{V}}_i = \left(\hat{x}_i, \hat{y}_i, \hat{z}_i\right)$.

Given the definition of MSE, the signal strength of a mesh $\mathcal{V}$ is the MSE between this mesh and its centroid $\bar{\mathbf{V}}$, where $\bar{\mathbf{V}} = 1/M \sum_{i=1}^{M} \mathbf{V}_i$. Let $\bar{\mathcal{V}} = \left\{\bar{\mathbf{V}}, \cdots, \bar{\mathbf{V}}\right\}$ be a collapsed mesh at the centroid with M vertices; then the signal strength is measured by $\text{MSE}\left(\mathcal{V}, \bar{\mathcal{V}}\right)$. The noise strength is measured by the MSE between the original mesh $\mathcal{V}$ and the scrambled and embedded mesh $\mathcal{V}_w = \{\mathbf{V}_{w1}, \cdots, \mathbf{V}_{wM}\}$: $\text{MSE}(\mathcal{V}, \mathcal{V}_w)$. Finally, the 3D SNR can be calculated as the ratio between these two MSEs:

$$\text{SNR}(\mathcal{V}, \mathcal{V}_w) = 10 \log_{10} \frac{\text{MSE}\left(\mathcal{V}, \bar{\mathcal{V}}\right)}{\text{MSE}(\mathcal{V}, \mathcal{V}_w)}. \quad (14)$$

According to (14), the numerator $\text{MSE}\left(\mathcal{V}, \bar{\mathcal{V}}\right)$ is a fixed value for a given mesh. The value of the SNR is larger if the value of the denominator $\text{MSE}(\mathcal{V}, \mathcal{V}_w)$ is smaller. It can be seen from (13) that if $\mathcal{V} = \mathcal{V}_w$, the denominator $\text{MSE}(\mathcal{V}, \mathcal{V}_w)$ degenerates to zero. Thus, the maximum value of SNR can be infinity if no modification is made to the mesh model.

4) ADAPTIVE THRESHOLD

To achieve the scalability of the mesh distortion, we propose an adaptive threshold selection algorithm. The purpose of this algorithm is to select vertices for embedding and scrambling so that the desired SNR can be achieved for a given amount of secret information ℓ_S. In particular, the threshold parameter μ should be determined adaptively from a given SNR and ℓ_S. First, we identify a lower bound on μ imposed by the reversibility condition, thus reducing the searching space of μ. Then, we derive an explicit relationship between μ and MSE. Using this relationship, we can determine μ from a given MSE requirement or SNR requirement.

The lower bound of the threshold μ can be determined from the reversibility requirement. To ensure reversibility, the encoded prediction error, the code for μ and the original α LSBs for each vertex in $\mathcal{N}$ must be fully embedded into all vertices in $\mathcal{E}$. A lower bound of μ can be found by assuming $\ell_S = 0$, i.e., no external secret information, which is $|\mathcal{N}| = (3|\mathcal{P}| - |\mathcal{E}|)$. Thus, the payload consists of the code for μ and the α LSBs for all vertices in $\mathcal{N}$. This payload requires a total of $\ell_\mu + \alpha|\mathcal{N}|$ bits [2], or equivalently $\ell_\mu + \alpha(3|\mathcal{P}| - |\mathcal{E}|)$. The embedding space provided by all vertices in $\mathcal{E}$ is $(d-6)|\mathcal{E}|$, since each vertex can only provide $d - 6$ bits space after excluding its two MSBs and α LSBs. To ensure reversibility, the available embedding space must be larger than the size of the payload: $(d-6)|\mathcal{E}| \geq \ell_\mu + \alpha(3|\mathcal{P}| - |\mathcal{E}|)$. Thus, the size of $\mathcal{E}$ must satisfy:

$$|\mathcal{E}| \geq \frac{\ell_\mu + 3\alpha|\mathcal{P}|}{d + \alpha - 6}. \quad (15)$$

Note that $\ell_\mu = 32$ is fixed, and only $|\mathcal{E}|$ is controlled by μ: the larger the μ, the larger the $|\mathcal{E}|$. To control the distortion level of the mesh, we adaptively select the threshold μ to meet the inequality (15). First, all the vertices in $\mathcal{P}$ are predicted, and the prediction errors are analyzed to obtain an unnormalized error histogram $h(e)$, where e is prediction error. Then, according to (9), we can find the size of $\mathcal{E}$ as $|\mathcal{E}| = \sum_{e=-\mu}^{\mu} h(e)$. Now, the requirement on μ can be explicitly expressed as

$$\sum_{e=-\mu}^{\mu} h(e) \geq \frac{\ell_\mu + 3\alpha|\mathcal{P}|}{d + \alpha - 6}. \quad (16)$$

From (16), one can easily solve for μ, given $|\mathcal{P}|, \ell_\mu, d$, and $h(e)$. Recall that this μ is the lower bound imposed by the reversibility requirement; hence, it is denoted as $\mu_{\min}$. This lower bound can be used as a starting point when determining μ for a given distortion requirement.

Furthermore, if the user specifies the amount of secret information to embed, i.e., ℓ_S, then we may obtain the following inequality by following a similar analysis procedure as outlined above:

$$\sum_{e=-\mu}^{\mu} h(e) \geq \frac{\ell_\mu + 3\alpha|\mathcal{P}| + \ell_S}{d + \alpha - 6}. \quad (17)$$

[2]Given a finite set $\mathcal{N}$, $|\mathcal{N}|$ is the number of elements in $\mathcal{N}$.

Using (17), our algorithm can adaptively determine an appropriate threshold μ to meet the user's embedding capacity requirement ℓ_S.

To adaptively determine μ from a given distortion requirement, we derive an explicit relationship between μ and $\mathrm{MSE}(\mathcal{V}, \mathcal{V}_w)$, as can be found in the appendix. Based on the different assumptions for the original coordinates, we obtain two MSE predictors: $\mathcal{M}_{\mathrm{H}}(\mu)$ in (31) and $\mathcal{M}_{\mathrm{U}}(\mu)$ in (36). For example, the predictor $\mathcal{M}_{\mathrm{U}}(\mu)$ is given by

$$\mathrm{MSE}(\mathcal{V}, \mathcal{V}_w) \approx \frac{1}{18M}\left(q_{\max}^2 - 1\right)\sum_{e=-\mu}^{\mu} h(e), \tag{18}$$

where $q_{\max} = 2^{d-2} - 1$. Given a specified 3D SNR from the user, we first calculate the signal energy $\mathrm{MSE}(\mathcal{V}, \bar{\mathcal{V}})$, then the requirement MSE can be found as:

$$\mathrm{MSE}(\mathcal{V}, \mathcal{V}_w) = \frac{\mathrm{MSE}(\mathcal{V}, \bar{\mathcal{V}})}{10^{SNR/10}}. \tag{19}$$

Adding this required SNR to (18), we obtain an explicit expression for the desired SNR as a function of the threshold μ. Given SNR, the threshold μ can be found by resorting a simple bisection algorithm [46]. Equivalently, one needs to solve for μ from the specified $\mathrm{MSE}(\mathcal{V}, \mathcal{V}_w)$ (by (19)). For example, the MSE predictor $\mathcal{M}_{\mathrm{U}}(\mu)$ given by (36), one needs to solve a nonlinear equation $f(\mu) = 0$, where

$$f(\mu) \triangleq \mathrm{MSE}(\mathcal{V}, \mathcal{V}_w) - \frac{1}{18M}\left(q_{\max}^2 - 1\right)\sum_{e=-\mu}^{\mu} h(e).$$

The initial interval for bisection searching can be set as $[\mu_{\min}, e_{\max}]$, where $\mu_{\min}$ can be determined from (16) and $e_{\max}$ is the maximum value of the prediction error.

In summary, our algorithm can provide scalable embedding capacity and scrambling effect by adaptively choosing the threshold parameter μ from the given capacity requirement or the 3D SNR requirement.

5) RATE-DISTORTION-REVERSIBILITY TRADEOFF

In this section, we discuss the tradeoff between three key performance indices: rate, distortion, and reversibility. For a given mesh, the unnormalized rate can be characterized by the size of the secret information ℓ_S, whereas distortion is measured by the SNR between the original mesh and embedded and scrambled mesh $\mathrm{SNR}(\mathcal{V}, \mathcal{V}_w)$. In addition, because the reversibility can be controlled in 3D-AUES, we use the MSE between the scrambled mesh $\mathcal{V}$ and the recovered mesh $\mathcal{C}$, i.e., $\mathrm{MSE}(\mathcal{V}, \mathcal{C})$ as a measure of reversibility. An illustration of tradeoffs between these three performance indices is summarized in Fig. 6, where the 'turning knobs' are the two parameters: threshold μ and length α of the code for prediction error (also referred to as flag bits).

The length of flag bit α determines the coding accuracy of prediction error. The larger this length α is, the higher the coding accuracy of prediction error will be. The encoding precision of the prediction error determines the distortion of the recovery mesh. The higher the encoding precision,

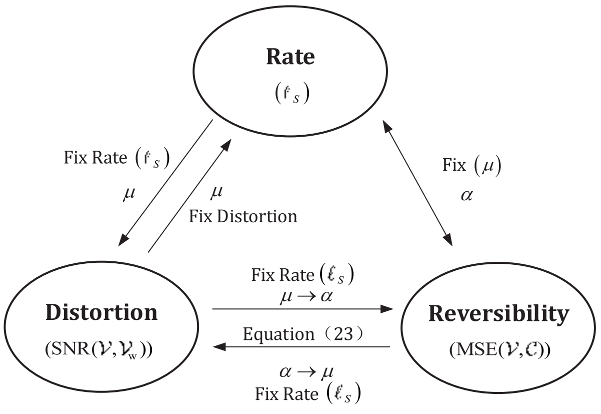


FIGURE 6. Illustration of rate-distortion-reversibility tradeoff.

the closer the mesh recovery is to the original mesh. In contrast, a larger α may affect redundancy that should be used to embed external data.

Threshold μ determines the number of coordinates in the embedded set $\mathcal{E}$. If μ increases, the number of embedded coordinates $\mathcal{E}$ also increases. Meanwhile, the distortion of the recovered mesh increases accordingly.

Reversibility can be controlled, from total reversibility to partial reversibility. Fixing μ, one may use α to control reversibility. To ensure total reversibility, recall that to encode all the integers in the range $[-\mu, \mu]$, we must guarantee that $2^\alpha - 1 \geq 2\mu + 1$. Thus, the length of flag bits must satisfy:

$$\alpha \geq \left\lceil \log_2(2\mu + 2) \right\rceil. \tag{20}$$

If α fails to meet (20), the prediction error is quantized using a step size larger than 1. Thus, one can control the reversibility of the mesh by setting the appropriate α.

For a given rate ℓ_S, one may find tradeoff between distortion $\mathrm{SNR}(\mathcal{V}, \mathcal{V}_w)$ and reversibility $\mathrm{MSE}(\mathcal{V}, \mathcal{C})$. Because ℓ_S is fixed, if μ increases, we can use a larger α. Thus, a small $\mathrm{SNR}(\mathcal{V}, \mathcal{V}_w)$ and a smaller $\mathrm{MSE}(\mathcal{V}, \mathcal{C})$ can be obtained. A smaller μ can lead to larger $\mathrm{SNR}(\mathcal{V}, \mathcal{V}_w)$ and larger $\mathrm{MSE}(\mathcal{V}, \mathcal{C})$.

The notion of 'scalability' in multimedia compression is to provide different qualities of the compressed media according to the transmission or storage requirement. In this paper, reference to Fig. 6, 'scalability' refers to providing different qualities of the scrambled mesh according to the requirement from the user. Therefore, in this paper, the mesh is not compressed but is only partially scrambled. This 'scalability' can be controlled by a threshold μ. Only those vertices whose prediction errors fall below the threshold μ are scrambled.

D. DATA EXTRACTION AND MESH RECONSTRUCTION

Data extraction and mesh reconstruction start with the same preprocessing as in (3). After the same vertex partition and prediction as presented in Section III-B, we obtain the predicted coordinates $\hat{V}_i''$ in the predicted set $\mathcal{P}$.

Next, we need to identify the embedded set $\mathcal{E}$ to extract the embedding parameters μ, the secret information S and the encoded prediction error. This can be achieved by utilizing different flag patterns of the α LSBs in the predicted set $\mathcal{P}$. As shown in Table 6, the coordinates in $\mathcal{N}$ and $\mathcal{E}$ can be distinguished according to the different flag patterns. For example, for $\alpha = 4$, when we find the α LSBs of the coordinate to be one of '0001', '0010', '0011', $\cdots$, '1111', then this coordinate belongs to the set $\mathcal{E}$. Thus, we can recover the two sets $\mathcal{E}$ and $\mathcal{N}$. Then, we extract the bits between the 5th LSB and the $(d-2)$-th LSB, which forms the payload $\mathbf{p}$. From $\mathbf{p}$, the secret information S, and the code for prediction error can be extracted.

For data extraction, recall that the storage of the payload $\mathbf{p}$ is sequential, as shown in Fig. 5. Therefore, the secret information can be extracted from the $\left(\ell_\mu + \ell_{LSB_\alpha} + 1\right)$-th bits to the last bits.

For mesh recovery, we need to recover all coordinates in $\mathcal{N}$ and all coordinates in $\mathcal{E}$.

1) To recover coordinates in $\mathcal{N}$, recall that, in $\mathbf{p}$, the $(d+1)$-th to $((d+1)+\alpha\,|\mathcal{N}|)$-th bits are the last α bits of the original coordinates in $\mathcal{N}$. Therefore, we replace each flag pattern '0000' by one segment of these bits.
2) To recover coordinates in $\mathcal{E}$, we need the predicted vertex $\hat{\boldsymbol{V}}_i''$ and the recovered prediction error. The code for prediction error can be extracted from the α LSBs of coordinates in $\mathcal{E}$. However, to recover the exact numerical value of the prediction error, we also need to recover the quantization table (or equivalently, the quantization step). First, we extract the code for μ from the first d bits of the extracted payload $\mathbf{p}$. With the knowledge of μ and α, the receiver can reconstruct the quantization table, such as the one shown in Table 6. The median of all integers in each interval is utilized to represent the prediction error $\tilde{\boldsymbol{e}}_i = \left\{\tilde{e}_{xi}, \tilde{e}_{yi}, \tilde{e}_{zi}\right\}$. By choosing the median value, the reconstruction error can be minimized given the quantization interval by assuming a uniform distribution over each interval. For each $\mathcal{E}$ coordinate, we restore it by (21).

$$\tilde{\boldsymbol{V}}_i' = \tilde{\boldsymbol{e}}_i + \hat{\boldsymbol{V}}_i'', \tag{21}$$

where $\tilde{\boldsymbol{V}}_i'$, $\tilde{\boldsymbol{e}}_i$ and $\hat{\boldsymbol{V}}_i''$ are the reconstruction coordinates, prediction error and the predicted coordinates, respectively, during the reconstruction process.

During the embedding and scramble, the referenced set $\mathcal{R}$ is not changed; hence, we do not need to recover it. Finally, we note that the mesh reconstruction may not be perfect, as discussed in the last section (Rate-Distortion-Reversibility Tradeoff).

In summary, the decoder can losslessly recover the secret information and reconstruct the mesh to a given distortion level.

E. SECURITY ANALYSIS

The security level of the proposed 3D-AUES algorithm relies on the purpose, knowledge, and tools available to the attacker. For the purpose, we assume that the attacker intends to (1) extract and read the secret information, and (2) recover the original mesh. For the knowledge of algorithms available to the attacker, we consider two scenarios that were also addressed in [42]. (1) The only knowledge available to the attacker about the AUES algorithm is the number of vertices in the predicted set. (2) The attacker has partial knowledge of the AUES algorithm so that the attacker knows which vertex belongs to the predicted set $\mathcal{P}$.

To prevent the illegal reading of the secret message, a stream cipher RC4 is used to encrypt the secret information before embedding. Thus, the security of the secret information relies on the keyspace and security of available stream ciphers.

For the first attacking scenario, because the attacker has no other knowledge of the AUES algorithm, the only tool available to the attacker is brute-force attack. The attacker needs to try a large number of permutations to recover the original mesh. Given a scrambled mesh having $3M$ vertices and that there are w vertices in the predicted set, the attacker needs to try every combination of w vertices from M vertices. For each such combination, the attacker further needs to identify the embedded set, which involves labeling each vertex with a binary label. The number of combinations is 2^{3M} because each vertex can be labeled either as 1 (for the embedded set) or 0 (for the nonembedded set). After choosing the embedded set, the attacker needs to exhaust all possible values that a coordinate can take, which needs 2^d trials. In summary, the total number of trial permutations is:

$$\binom{3M}{w} \times 2^{3M} \times 2^d, \tag{22}$$

where $\binom{3M}{w}$ calculates the number of combinations when choosing w items from $3M$ items. From the Stirling approximation to factorials [47], we obtain the lower bound for the binomial coefficient:

$$\binom{3M}{w} \geq \left(\frac{3M}{w}\right)^w = 10^{w\log_{10}\frac{3M}{w}}.$$

Combining this lower bound with a rough approximation $2^{10} \gtrapprox 10^3$, the total number of permutations in (22) is lower bounded by

$$\binom{3M}{w} \times 2^{3M} \times 2^d \geq 10^{w\log_{10}\frac{3M}{w} + \lfloor\frac{9M+3d}{10}\rfloor}.$$

For example, take a small mesh model, with parameters $M = 988$, $d = 32$ and $w = 1500$. The number of combinations that an attacker should consider is

$$\binom{3\times 988}{1500} \times 2^{(3\times 988)} \times 2^{32} \gtrapprox 10^{1341}. \tag{23}$$

This number of combinations is vast because the total number of atoms in the universe is estimated to be approximately 10^{80}.

For the second scenario, the attacker is assumed to know which coordinates belong to the predicted set $\mathcal{P}$. Therefore, the attacker needs to identify the embedded set, which involves 2^{3M} trials. After that, the attack needs to exhaust all possible values that a coordinate can take, which involves 2^d trials. Therefore, the total number of trials is

$$2^{3M/2} \times 2^d \geq 10^{\lfloor \frac{9M/2+3d}{10} \rfloor} \approx 10^{454}, \tag{24}$$

for a small-sized mesh with parameters $M = 988, d = 32$ and $w = 1500$. Even for this scenario, the number of trials that an attacker should try exceeds the ability of current computers.

For the above two attack scenarios, our algorithm is very useful. However, we also have to consider the scenario of unintentional attacks. Considering the current application scenarios of 3D mesh models, at least the following data processing should be regarded as unintentional attacks: 1) similarity transformation including translation, rotation and uniform scaling of the vertices, and 2) vertex coordinate quantization. These two operations are commonly used when processing a mesh model. Therefore, they should be considered unintentional attacks. The two operations have a different impact on visual quality. After the similarity transformation, the visual quality of the model remains the same as before transformation. In contrast, because quantization reduces the precision of coordinates, the perceived roughness of the surface may increase after quantization.

Recall that in the 3D-AUES algorithm, we modified most of the bits of the vertex coordinates to embed external data and model recovery information. Both the similarity transform and the coordinate quantization change the numerical representation of the coordinate. Thus, the embedded data are destroyed after these two operations. As a result, 3D-AUES is not robust to these unintentional attacks. However, for an image, neither the RDH nor the UES is robust [48]. Unlike ordinary data hiding or watermarking, fragility to unintentional attacks is a common feature of reversible data hiding. Furthermore, we note that in 3D-AUES, only the scrambled mesh is exposed to these attacks. Thus, the perceptual quality of the attacked scrambled mesh is not as important as that of ordinary data hiding.

To verify the above analysis, two experiments were conducted to test the mesh recovery quality after similarity transformation. For scaling operations, the scrambled mesh was scaled by 10 times. Then, the scaled mesh was restored according to the recovery stage of the 3D-AUES algorithm. As shown in Fig. 7, the restored mesh after scaling is even worse than the scrambled mesh because the binary representation of the coordinate has changed, and mesh recovery information cannot be correctly restored. For translation operations, the scrambled mesh was translated 0.1 units to the right and then restored according to the 3D-AUES algorithm. As shown in Fig. 8, the restored mesh after translation is completely different from the original mesh.

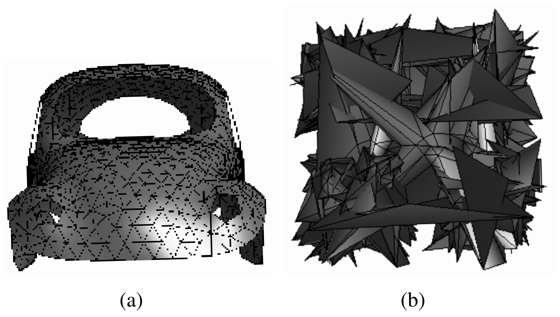


FIGURE 7. Reconstructed mesh: (a). Reconstructed scrambled mesh without unintentional attack. (b). Reconstructed scaled mesh.

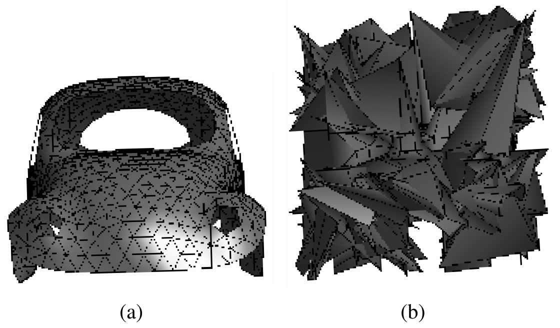


FIGURE 8. Reconstructed mesh: (a). Reconstructed scrambled mesh without unintentional attack. (b). Reconstructed translation mesh.

IV. EXPERIMENTAL RESULTS

In this section, we show our experimental results. First, the setup for the experiment is described. Next, the performance metric is introduced in Section IV-B, including the embedding rate and SNR. Third, the various tests are analyzed using a set of typical 3D mesh models. Finally, we compare the proposed 3D-AUES with RDH in the encrypted 3D mesh.

A. SETUPS

The 3D-AUES algorithm is implemented in the MATLAB R2010b platform using a Windows 7 operating system. The CPU is an AMD Athlon X4 855 with a quad-core processor running at 3.50 GHz. The physical memory is 4 GB of RAM.

Regarding the storage of vertex coordinates, standard file formats for the 3D model are .PLY, .OBJ, and .OFF. We use the .OFF format in our paper. If the precision of the coordinate is preserved when transforming from one format to another, the scrambled mesh can still be recovered. However, if the precision of the coordinate is reduced after transformation, then the scrambled mesh cannot be recovered because the recovery information is lost.

For mesh processing, we use a 3D tool package, including *toolbox_graph*, *toolbox_signal*, and *toolbox_general* from [49], and *Model directory 0-3* from [50], [51].

Based on the above experimental environment, we select 22 mesh models from two popular mesh datasets, including the Princeton Shape Benchmark and the Laser Design dataset [50], [51]. The number of vertices for each model

TABLE 5. The experimental meshes and the execution efficiency with minimum payload.

3D mesh models	Sizes (length, width, height)	Number of faces	Number of vertices M	Execution efficiency(s)
VENUS	(2.5750, 5.25, 1.87)	1396	711	3
COW	(0.5283, 0.9875, 1.7157)	1448	726	4
HORSE	(0.4579, 0.8255, 1.0048)	1420	712	4
BEETLE	(0.4057, 0.3385, 0.9991)	1763	988	8
MUSHROOM	(1.7411, 1.2397, 1.9157)	448	226	2
BIGNEFERTITI	(3.9022, 4.8618, 2.4044)	8740	4439	51
CAT1	(1.0441, 1.5203, 2)	8971	4538	58
LEOPARD	(0.2723, 0.5375, 0.9750)	10061	5169	77
SKATEBOARD	(0.9947, 0.1192, 0.3085)	12166	6101	109
BIRD	(0.6958, 0.1493, 0.9980)	13106	6634	119
BIGCAMEL	(0.3081, 0.9805, 1.0027)	16568	8286	174
BIGCOW	(2, 1.2265, 0.6581)	18208	9105	211
ENGINE	(1.207, 2, 1.207)	20732	10576	370
DOG	(0.2459, 0.618, 0.975)	21940	10978	393
LARGEDOG	(0.3661, 0.8324, 0.975)	25030	12515	412
PEPPER	(60.3922, 135.9362, 51.1958)	29956	14980	634
BIGMANNEQUIN	(1.8839, 2.4605, 2.9099)	35205	17662	828
HEAD	(0.6448, 0.7635, 0.9368)	37548	19052	988
FISH	(1.6368, 2.0010, 1.4874)	43696	21915	1629
DAVID	(1.7767, 1.7829, 2.2348)	47280	23889	1521
WATCH	(0.4919, 0.7667, 0.975)	59640	30546	2471
BUNNY	(0.1557, 0.2203, 0.1207)	69473	34835	3288

ranges from 10^2 to 10^5. Our experimental results show that the proposed 3D-AUES applies to a large quantity of mesh data. The execution efficiency is listed in Table 5. The unit of elapsed time is second. We can see that each mesh has a different size in Table 5. Five original meshes selected from these 22 meshes are shown in Fig. 9.

The parameter in the processing step is set as $n = 9$. Therefore, to represent each coordinate, the word length is set as $d = 32$. The number of embedding layers $L \in \{1, 2\}$, and the code length for prediction error is set as $\alpha = 4$. Instead of using varying α to test reversibility, we use a fixed α, but the reversibility can still be controlled by varying the threshold parameter μ, as discussed in Section III-C5 (See (20)).

The encoding of prediction error is based on a uniform quantizer, where the range of the quantizer is $[-\mu, \mu]$, and the reconstruction point is the middle of each quantization interval. For example, the quantization interval for the mesh model BEETLE is illustrated in Table 6, where the parameters are set as $\alpha = 4$ and $\mu \in \left[2.850 \times 10^6, 1.350 \times 10^8\right]$.

TABLE 6. Allocation of intervals for encoding the prediction error of the model BEETLE, using parameter $\alpha = 4$, $-1.350 \times 10^8 \le e \le 1.350 \times 10^8$ and $\mu \in \left[2.850 \times 10^6, 1.350 \times 10^8\right]$.

μ	0000	0001	0010	0011
$2.850 \times 10^6 \le \mu \le 2.888 \times 10^6$	$\mathcal{N}$	$-\mu \le e \le -2.470 \times 10^6$	$-2.470 \times 10^6 < e \le -2.090 \times 10^6$	$-2.090 \times 10^6 < e \le -1.710 \times 10^6$
$2.888 \times 10^6 < \mu \le 8.940 \times 10^6$	$\mathcal{N}$	$-\mu \le e \le -7.748 \times 10^6$	$-7.748 \times 10^6 < e \le -6.556 \times 10^6$	$-6.556 \times 10^6 < e \le -5.364 \times 10^6$
$8.940 \times 10^6 < \mu \le 2.015 \times 10^7$	$\mathcal{N}$	$-\mu \le e \le -1.746 \times 10^7$	$-1.746 \times 10^7 < e \le -1.478 \times 10^7$	$-1.478 \times 10^7 < e \le -1.209 \times 10^7$
$2.015 \times 10^7 < \mu \le 1.350 \times 10^8$	$\mathcal{N}$	$-\mu \le e \le -1.170 \times 10^8$	$-1.170 \times 10^8 < e \le -9.900 \times 10^7$	$-9.900 \times 10^7 < e \le -8.100 \times 10^7$

0100	0101	0110	0111
$-1.710 \times 10^6 < e \le -1.330 \times 10^6$	$-1.330 \times 10^6 < e \le -9.500 \times 10^5$	$-9.500 \times 10^5 < e \le -5.700 \times 10^5$	$-5.700 \times 10^5 < e \le -1.900 \times 10^5$
$-5.364 \times 10^6 < e \le -4.172 \times 10^6$	$-4.172 \times 10^6 < e \le -2.980 \times 10^6$	$-2.980 \times 10^6 < e \le -1.788 \times 10^6$	$-1.788 \times 10^6 < e \le -5.960 \times 10^5$
$-1.209 \times 10^7 < e \le -9.403 \times 10^6$	$-9.403 \times 10^6 < e \le -6.717 \times 10^6$	$-6.717 \times 10^6 < e \le -4.030 \times 10^6$	$-4.030 \times 10^6 < e \le -1.343 \times 10^6$
$-8.100 \times 10^7 < e \le -6.300 \times 10^7$	$-6.300 \times 10^7 < e \le -4.500 \times 10^7$	$-4.500 \times 10^7 < e \le -2.700 \times 10^7$	$-2.700 \times 10^7 < e \le -9.000 \times 10^6$

1000	1001	1010	1011
$-1.900 \times 10^5 < e \le 1.900 \times 10^5$	$1.900 \times 10^5 < e \le 5.700 \times 10^5$	$5.700 \times 10^5 < e \le 9.500 \times 10^5$	$9.500 \times 10^5 < e \le 1.330 \times 10^6$
$-5.960 \times 10^5 < e \le 5.960 \times 10^5$	$5.960 \times 10^5 < e \le 1.788 \times 10^6$	$1.788 \times 10^6 < e \le 2.980 \times 10^6$	$2.980 \times 10^6 < e \le 4.172 \times 10^6$
$-1.343 \times 10^6 < e \le 1.343 \times 10^6$	$1.343 \times 10^6 < e \le 4.030 \times 10^6$	$4.030 \times 10^6 < e \le 6.717 \times 10^6$	$6.717 \times 10^6 < e \le 9.403 \times 10^6$
$-9.000 \times 10^6 < e \le 9.00 \times 10^6$	$9.00 \times 10^6 < e \le 2.700 \times 10^7$	$2.700 \times 10^7 < e \le 4.500 \times 10^7$	$4.500 \times 10^7 < e \le 6.300 \times 10^7$

1100	1101	1110	1111
$1.330 \times 10^6 < e \le 1.710 \times 10^6$	$1.710 \times 10^6 < e \le 2.090 \times 10^6$	$2.090 \times 10^6 < e \le 2.470 \times 10^6$	$2.470 \times 10^6 < e \le \mu$
$4.172 \times 10^6 < e \le 5.364 \times 10^6$	$5.364 \times 10^6 < e \le 6.556 \times 10^6$	$6.556 \times 10^6 < e \le 7.748 \times 10^6$	$7.748 \times 10^6 < e \le \mu$
$9.403 \times 10^6 < e \le 1.210 \times 10^7$	$1.210 \times 10^7 < e \le 1.478 \times 10^7$	$1.478 \times 10^7 < e \le 1.746 \times 10^7$	$1.746 \times 10^7 < e \le \mu$
$6.300 \times 10^7 < e \le 8.100 \times 10^7$	$8.100 \times 10^7 < e \le 9.900 \times 10^7$	$9.900 \times 10^7 < e \le 1.170 \times 10^8$	$1.170 \times 10^8 < e \le \mu$

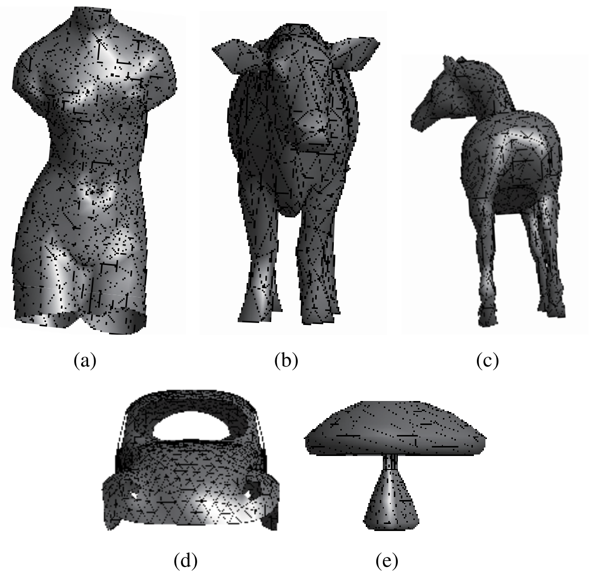


FIGURE 9. Original image *O*: (a). VENUS (b). COW (c). HORSE (d). BEETLE (e). MUSHROOM.

B. PERFORMANCE METRIC

The performance of 3D-AUES can be evaluated with rate, distortion, and reversibility.

1) EMBEDDING RATE

The embedding rate of 3D-AUES is defined as the number of secret bits embedded in each coordinate, i.e.,

$$R \triangleq \frac{\ell_S}{3M}, \tag{25}$$

where ℓ_S is the number of secret bits, and $3M$ is the total number of coordinates. Hence, the rate unit is bits per coordinate (bpc).

According to (17) in Section III-C4, we obtain

$$R = \frac{\ell_S}{3M} = \frac{|\mathcal{E}|\,(d + \alpha - 6) - \ell_\mu - 3\alpha\,|\mathcal{P}|}{3M}. \tag{26}$$

Specifically, given α, the maximum rate can be achieved if the whole predicted set is used to embed the payload, namely, $\mathcal{P} = \mathcal{E}$. Therefore,

$$R_{\max} = \frac{|\mathcal{P}|\,(d - 6 - 2\alpha) - \ell_\mu}{3M}. \tag{27}$$

2) DISTORTION

The distortion between the scrambled mesh and the original mesh is measured by the 3D SNR introduced in Section III-C3 [29], which is a measurement of the 3D mesh geometric distortion. The 3D SNR between the preprocessed mesh $\mathcal{V}'$ and the scrambled mesh $\mathcal{V}_w$ can be written as:

$$\mathrm{SNR}\left(\mathcal{V}', \mathcal{V}_w\right) = 10\log_{10}\frac{\mathrm{MSE}\left(\mathcal{V}', \overline{\mathcal{V}}\right)}{\mathrm{MSE}\left(\mathcal{V}', \mathcal{V}_w\right)}, \tag{28}$$

where $\mathcal{V}' = \left\{\boldsymbol{V}'_1, \boldsymbol{V}'_2, \cdots, \boldsymbol{V}'_M\right\}$ is the preprocessed mesh, and $\mathcal{V}_w = \{\boldsymbol{V}_{w1}, \boldsymbol{V}_{w2}, \cdots, \boldsymbol{V}_{wM}\}$ is the scrambled mesh. The mesh $\overline{\mathcal{V}} = \left\{\bar{\boldsymbol{V}}, \cdots, \bar{\boldsymbol{V}}\right\}$ is a hypothetical mesh with all its M vertices collapsed to the centroid $\bar{\boldsymbol{V}}$ of mesh $\mathcal{V}'$.

3) REVERSIBILITY

Because our scheme can control the reversibility of the scrambled mesh, an appropriate index should be employed to quantify the degree of reversibility. For this purpose, we use the 3D SNR between the original mesh $\mathcal{V}$ and the recovered mesh $\mathcal{C}$: SNR($\mathcal{V}, \mathcal{C}$). If a mesh can be completely reversed, we have SNR($\mathcal{V}, \mathcal{C}$) $= \infty$.

C. VARIOUS TESTS FOR 3D-AUES

In this section, we present testing results for rate, distortion, and reversibility, including (1) scalability controlled by threshold μ, (2) the rate-distortion tradeoff, and (3) controllable reversibility by threshold μ.

First, we demonstrate the perceptual quality of scrambled mesh using different μ on BEETLE. The experimental results are shown in Fig. 10. When $\mu = 2.850 \times 10^6$, The SNR is relatively high (i.e., max=17.01 dB for BEETLE), and the overall profile of the mesh is clearly visible. With the increase in μ, more distortion is introduced. When $\mu = 1.350 \times 10^8$, the SNRs reach their lowest values, which is caused by the increased number of coordinates in the embedded set $\mathcal{E}$. Using such a μ, the profile of the mesh is entirely unrecognizable. This experiment shows that our 3D-AUES can achieve scalable perceptual quality. Furthermore, we see that the SNR values correlate well with the perceptual quality.

In Table 7, we present the scalability result for five meshes using SNR as a distortion measure. We observe that when the threshold μ increases from minimum to maximum, the SNR decreases accordingly for all testing meshes. This demonstrates that the scalability can be controlled for all the testing meshes, even using a different set of thresholds for each mesh.

To test the performance of the two MSE predictors $\mathcal{M}_{\mathrm{H}}$ and $\mathcal{M}_{\mathrm{U}}$ (31)) and (36) in Appendix), we use two models, BEETLE and VENUS. The testing results are shown in Fig. 11. For both the models BEETLE and VENUS, we observe that both the MSE predictor $\mathcal{M}_{\mathrm{H}}$ and the simplified predictor $\mathcal{M}_{\mathrm{U}}$ can predict the SNR very well. The SNR prediction errors are summarized in Table 8. For example, for the model VENUS, the largest error is approximately 1 dB, and the average error is approximately 0.3 dB. Comparing $\mathcal{M}_{\mathrm{H}}$ and $\mathcal{M}_{\mathrm{U}}$, the complete MSE predictor $\mathcal{M}_{\mathrm{H}}$ is superior to the simplified predictor. Using $\mathcal{M}_{\mathrm{H}}$, the average prediction error for SNR is approximately 0.176 dB. Thus, for the two models, the actual SNR is within the 0.2 dB range of the specified SNR. This result shows that our algorithm can scramble the mesh to a target distortion level.

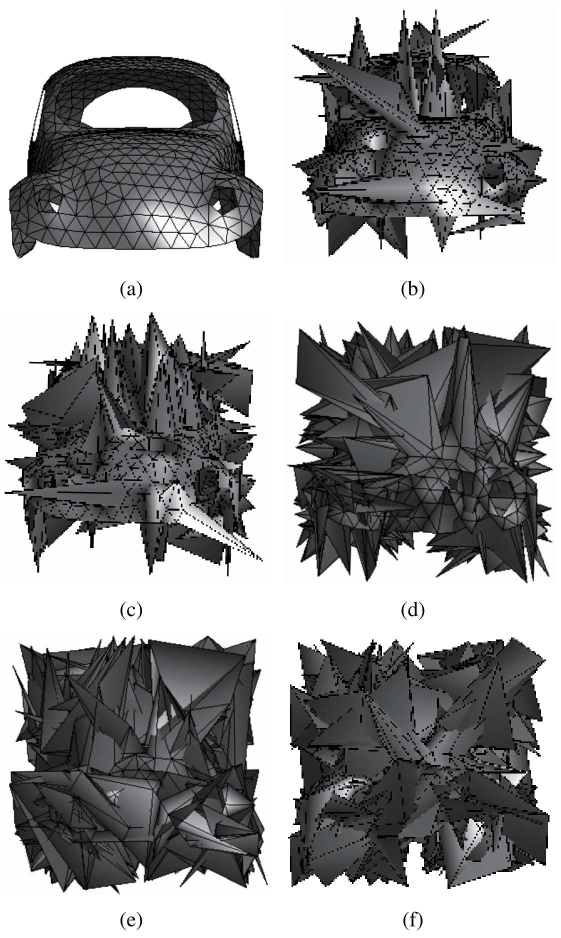
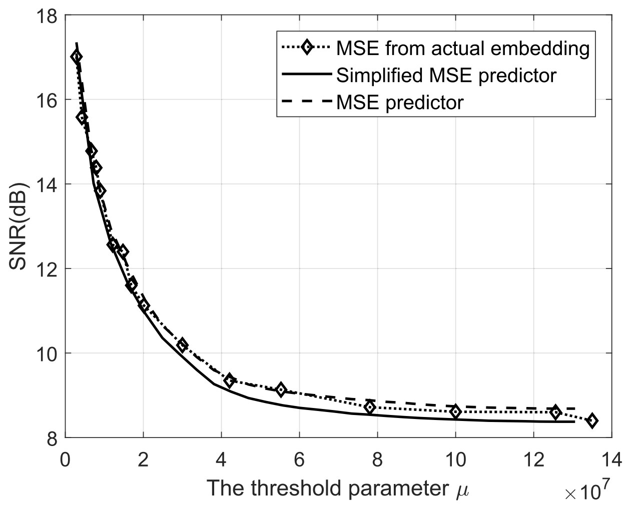


FIGURE 10. Scrambled mesh using the 3D-AUES method ($L = 1$): a. Original. b. Scrambled mesh with $\mu = 2.850 \times 10^6$, SNR($\mathcal{V}', \mathcal{V}_w$) = 17.01dB. c. Scrambled mesh with $\mu = 6.730 \times 10^6$, SNR($\mathcal{V}', \mathcal{V}_w$) = 15.58dB. d. Scrambled mesh with $\mu = 2.015 \times 10^7$, SNR($\mathcal{V}', \mathcal{V}_w$) = 11.12dB. e. Scrambled mesh with $\mu = 5.526 \times 10^7$, SNR($\mathcal{V}', \mathcal{V}_w$) = 9.133dB. f. Scrambled mesh with $\mu = 1.350 \times 10^8$, SNR($\mathcal{V}', \mathcal{V}_w$) = 8.399dB.

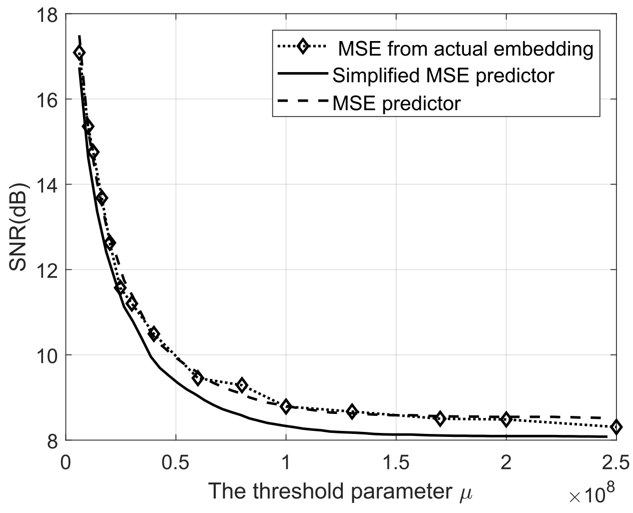
TABLE 7. The 3D SNR of the meshes.

$L = 1$	VENUS	COW	HORSE	BEETLE	MUSHROOM
$\mu_{\min}$	6.138×10^6	8.220×10^6	7.800×10^6	2.850×10^6	1.699×10^7
SNR($\mathcal{V}, \mathcal{V}_w$)	17.09	17.16	17.44	17.01	15.75
μ	2.473×10^7	5.238×10^7	1.700×10^7	7.920×10^6	5.864×10^7
SNR($\mathcal{V}, \mathcal{V}_w$)	11.57	9.937	14.14	14.38	12.73
μ	6.000×10^7	1.000×10^8	5.000×10^7	4.207×10^7	1.000×10^8
SNR($\mathcal{V}, \mathcal{V}_w$)	9.454	8.967	10.54	9.349	11.49
$\mu_{\max}$	2.500×10^8	3.300×10^8	2.200×10^8	1.350×10^8	3.350×10^8
SNR($\mathcal{V}, \mathcal{V}_w$)	8.308	8.656	9.087	8.399	8.636

In Section III-C2, we mentioned that multiple layer embedding could be employed to further increase the effect of scrambling. The results for three testing meshes are shown in Fig. 12, Fig. 13 and Fig. 14. For example, let μ_1 and μ_2 take their minimum values. Then, after two layers of scrambling,



(a) BEETLE



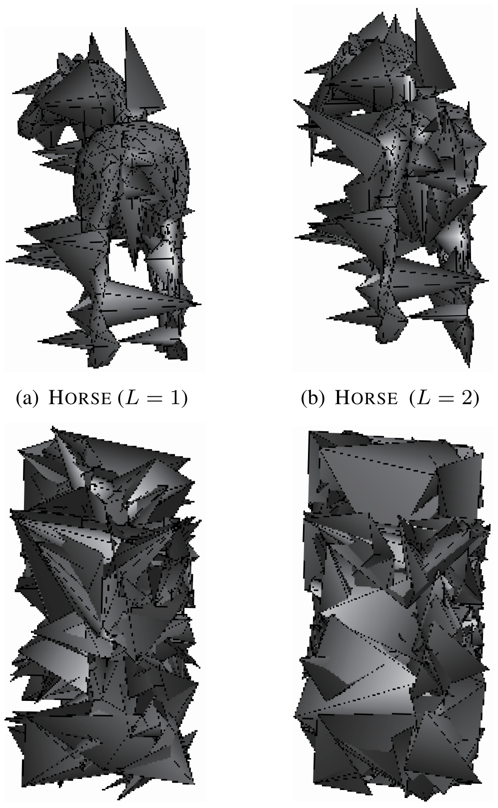
(b) VENUS

FIGURE 11. Experimental results for SNR prediction on two typical mesh models. The MSE predictor is $\mathcal{M}_H$, and the simplified MSE predictor is $\mathcal{M}_U$.

TABLE 8. SNR prediction error (dB).

Predictor	BEETLE		VENUS	
	Average	Max	Average	Max
$\mathcal{M}_U$	0.256	0.612	0.537	1.068
$\mathcal{M}_H$	0.165	0.339	0.176	0.405

the model VENUS looks more distorted compared with the one layer scrambling in Fig. 13(a). When μ_1 and μ_2 take their maximum values, as shown in Fig. 13(d), after two layers scrambling, the model VENUS looks more distorted compared with one layer scrambling (Fig. 13(c)). Because of



(a) HORSE ($L = 1$) (b) HORSE ($L = 2$) (c) HORSE($L = 1$) (d) HORSE($L = 2$)

FIGURE 12. The scrambled mesh models of HORSE. For $L = 1$ and $L = 2$, the corresponding thresholds are μ_1 and μ_2, respectively. (a). $\mu_{1,min} = 7.800 \times 10^6$. (b). $\mu_{1,min} = 7.800 \times 10^6$. $\mu_{2,min} = 1.07 \times 10^7$. (c). $\mu_{1,max} = 2.200 \times 10^8$. (d). $\mu_{1,max} = 2.200 \times 10^8$. $\mu_{2,max} = 1.000 \times 10^9$.

the cross prediction, the threshold of the first layer affects the choice of threshold for the second layer. The maximum value (minimum value) of the second layer in Figs. 12, 13 and 14 correspond to the maximum value (minimum value) of the first layer. For COW, when $\mu_{2,\max} = 3.300 \times 10^8$, its first layer is $\mu_{1,\max} = 1.200 \times 10^9$.

We can see that the scrambled mesh is too distorted because we use different thresholds for different levels of scrambling. For instance, in Section IV, Fig. 10(d)-(f), Fig. 12(c), Fig. 13(c), and Fig. 14(c) are that we specifically select a large threshold to disturb the mesh. For example, in Fig. 12(d), Fig. 13(d), and Fig. 14(d), we specifically use two-layer scrambling and a large threshold to disturb the mesh to achieve maximum visual distortion.

To verify the rate-distortion tradeoff discussed in Section III-C5, we plot the R vs. SNR$(\mathcal{V}', \mathcal{V}_w)$ curve by varying the threshold μ. This result is shown in Fig. 15, which shows a clear tradeoff between rate and distortion. Using this curve, one may determine an upper bound on rate R given a specified distortion measure SNR.

To verify the controlled reversibility, we test the SNR$(\mathcal{V}, \mathcal{C})$ for a different choice of μ. Fig. 16 shows the reconstructed mesh BEETLE. As can be observed, the SNR$(\mathcal{V}, \mathcal{C})$ decreases

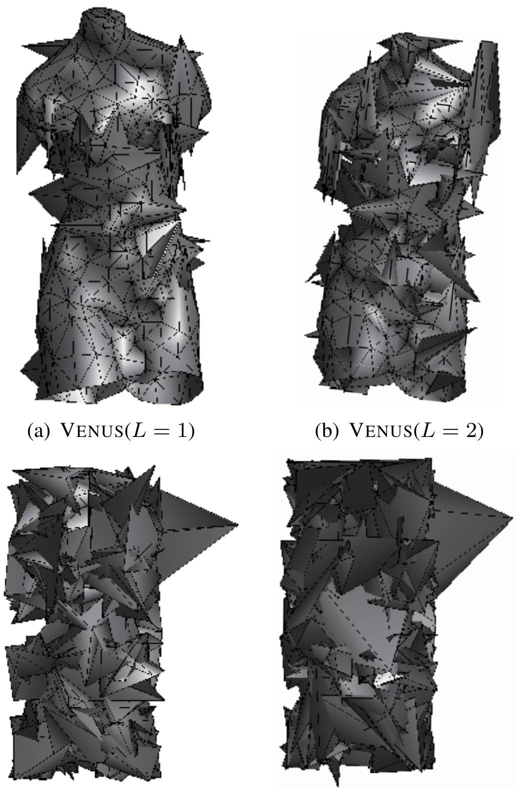

(a) VENUS($L = 1$) (b) VENUS($L = 2$) (c) VENUS($L = 1$) (d) VENUS($L = 2$)

FIGURE 13. The scrambled mesh models of VENUS. For $L = 1$ and $L = 2$, the corresponding thresholds are μ_1 and μ_2, respectively. (a). $\mu_{1,min} = 6.138 \times 10^6$. (b). $\mu_{1,min} = 6.138 \times 10^6$. $\mu_{2,min} = 1.050 \times 10^7$. (c). $\mu_{1,max} = 2.500 \times 10^8$ (d). $\mu_{1,max} = 2.500 \times 10^8$. $\mu_{2,max} = 1.050 \times 10^9$.

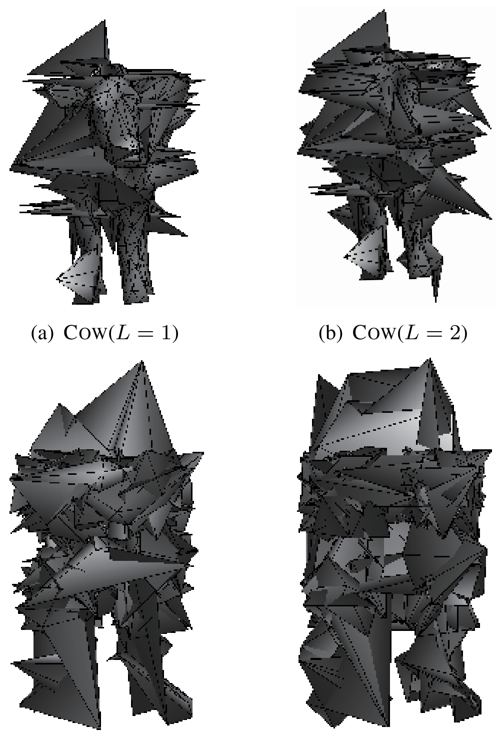

(a) COW($L = 1$) (b) COW($L = 2$) (c) COW($L = 1$) (d) COW($L = 2$)

FIGURE 14. The scrambled mesh models of COW. For $L = 1$ and $L = 2$, the corresponding thresholds are μ_1 and μ_2, respectively. (a). $\mu_{1,min} = 8.220 \times 10^6$. (b). $\mu_{1,min} = 8.220 \times 10^6$. $\mu_{2,min} = 1.267 \times 10^7$. (c). $\mu_{1,max} = 3.300 \times 10^8$ (d). $\mu_{1,max} = 3.300 \times 10^8$. $\mu_{2,max} = 1.200 \times 10^9$.

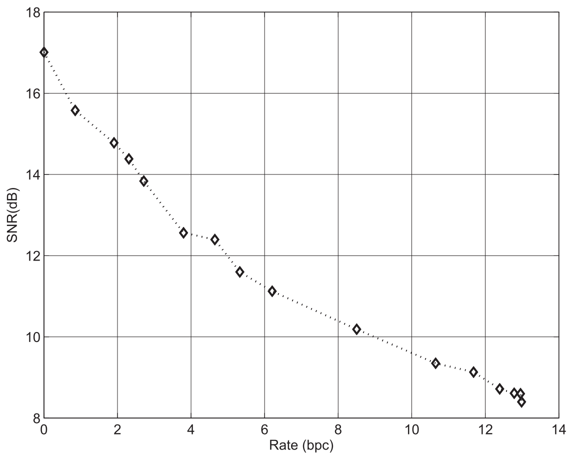


FIGURE 15. $L = 1$, SNR($\mathcal{V}'$, $\mathcal{V}_w$) values under different embedding rates R(bpc) for BEETLE.

as μ increases. For example, when $\mu = 2.850 \times 10^6$, the recovered mesh model has the highest fidelity with the original model. Accordingly, SNR($\mathcal{V}$, $\mathcal{C}$) is also the highest. In contrast, when choosing $\mu = 1.350 \times 10^8$, the reconstructed mesh shows visual distortion (for example, the part below the right headlamp). This observation is consistent with our analysis of rate-distortion-reversibility tradeoff in Section III-C5. The recovery quality of these meshes may not be bad because, as we noted, reversibility is controllable, from fully reversible to partially reversible. Complete reversibility is the range in which the total prediction error is encoded, and the length of the coded step is 1. The experimental result of Fig. 16(f) is in the case where the length of the coded step is not 1. This situation is partially reversible, so the recovered mesh will be distorted.

D. COMPARISON WITH RDH IN ENCRYPTED 3D MESH

We compare 3D-AUES with RDH in the encrypted 3D mesh [24], i.e., Jiang's algorithm, because it is closely relevant to 3D-AUES. First, Jiang's approach encrypts the mesh before embedding the data and does not have a mesh prediction stage. However, for our method, embedding the data and scrambling the mesh are simultaneous and have a mesh prediction stage. Having a mesh prediction stage is crucial to exploring the redundancy in the mesh to increase the embedding rate. Second, the vertex classification in 3D-AUES is from Jiang's approach. We distinguish vertex coordinates by the parity of the vertex index values. For Jiang's algorithm, the classification is more complicated. It first initializes the referenced set and embedded set as empty. Then, the vertices in the face are traversed one by one. If the vertices being traversed are not in the current embedded

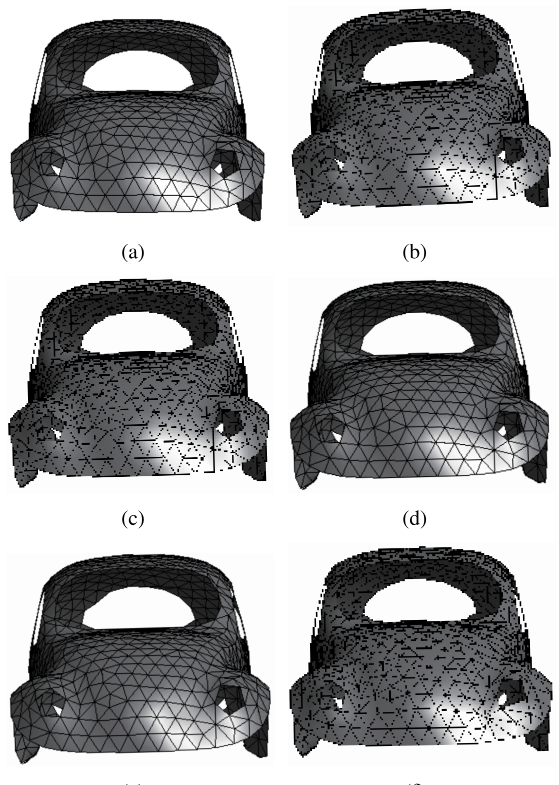

(a) (b) (c) (d) (e) (f)

FIGURE 16. Recovery mesh using the 3D-AUES method ($L = 1$): a. Original. b. Recovered mesh with $\mu = 2.850 \times 10^6$, SNR($\mathcal{V}, \mathcal{C}$) = 86.85dB. c. Recovered mesh with $\mu = 6.730 \times 10^6$, SNR($\mathcal{V}, \mathcal{C}$) = 74.81dB. d. Recovered mesh with $\mu = 2.015 \times 10^7$, SNR($\mathcal{V}, \mathcal{C}$) = 39.79dB. e. Recovered mesh with $\mu = 5.526 \times 10^7$, SNR($\mathcal{V}, \mathcal{C}$) = 35.61dB. f. Recovered mesh with $\mu = 1.350 \times 10^8$, SNR($\mathcal{V}, \mathcal{C}$) = 32.67dB.

set and referenced set, the first vertex in the face is placed in the embedded set. The rest are placed in the referenced set. If at least one vertex of the traversing face is not included in the embedded set but is included in the referenced set, the first vertex is deleted from the face being traversed, and the rest are placed in the embedded set. Our experimental results show that using the proposed simple classification provides outstanding performance for medium- and large-sized mesh. Third, our method embeds the payload by replacing the LSBs of the vertex coordinates, while Jiang's method XOR the LSBs to embed the payload. Finally, 3D-AUES can provide scalable scrambling quality, controllable reversibility, and a high embedding rate, which are lacking in the Jiang algorithm. These lead to different application scenarios. Hence, it is difficult to compare the 3D-AUES algorithm with Jiang's algorithm. It is difficult to compare other properties, except embedding rate, embedding-scrambling method, and reversibility. For a fair comparison, in Jiang's method, we use the same method to distinguish between referenced set $\mathcal{R}$ and predicted set $\mathcal{P}$. In these two methods, we take BEETLE as an example and choose the best embedded bits in an individual approach to obtain the embedding rate R (bpc). The result of the comparison is shown in Table 9.

The embedding rate of 3D-AUES is higher than Jiang's algorithm. Note that Jiang's algorithm completely scrambles

TABLE 9. Comparison of the 3D-AUES algorithm and Jiang's algorithm.

Method	Embedding rate(bpc)	Embedding-Scrambling	Reversibility
Jiang [24]	6	Separable	Reversible
3D-AUES	12.99	Unified	Controllable

the mesh, whereas 3D-AUES only partially scrambles the mesh. This high embedding rate is a result of joint embedding and scrambling, where the scrambling does not hinder the later embedding process. We also should note that Jiang's algorithm provides total reversibility, whereas 3D-AUES provides controlled reversibility, from total reversibility to partial reversibility of various degrees.

V. CONCLUSION

In this paper, an adaptive unified data embedding and scrambling algorithm for the 3D mesh model is proposed. We divide vertices into odd vertices, even vertices, and predict coordinates by cross prediction. The coordinates whose prediction error satisfies the threshold range are used to embed the external data. By embedding external data, we achieve the purpose of mesh scrambling. 3D-AUES can severely destroy the quality of the original mesh, thus improving the security of the mesh model. More importantly, the algorithm can be used to embed extensive external data. Theoretical analysis and experiments verified that 3D-AUES is unified, having a high embedding rate, scalability, and controllable reversibility. Therefore, it can be used in application scenarios where different requirements for the distortion of the mesh model are imposed. The embedded data can be extracted from the scrambled mesh, and the original mesh can be recovered to the desired fidelity.

One possible extension of this work is to combine Laplacian mesh processing with unified embedding and scrambling. The essence of Laplacian mesh deformation is the process of encoding and decoding the local detail features of the mesh model [52], [53]. The encoding process refers to the transformation of the Euclidean space coordinates of the mesh vertices to the Laplacian coordinates. The Laplacian coordinates contain the local detail features of the mesh, so the Laplacian mesh deformation algorithm can better preserve the local details of the mesh model. This can be exploited to utilize the redundancy in a mesh model in RDH.

APPENDIX
DERIVATION OF MSE PREDICTORS

In this appendix, we derive the MSE as a function of the threshold μ, which is a foundation for an adaptive choice of μ based on specified SNR.

Assume that the nonnormalized histogram of the prediction error is $h(e)$, where $e_{\min} < e < e_{\max}$ and the width of each bin is 1. Given μ, one can determine $|\mathcal{E}| = \sum_{e=-\mu}^{\mu} h(e)$ and $|\mathcal{N}| = 3\,|\mathcal{P}| - |\mathcal{E}|$.

Referring to Fig. 5, after embedding and scrambling, no distortion is introduced by vertices in the referenced set $\mathcal{R}$ because these coordinates are not modified.

Recall that the predicted set is split into two sets: the embedded set $\mathcal{E}$ and nonembedded set $\mathcal{N}$. To calculate the MSE introduced by coordinates in $\mathcal{N}$, we only need to focus on the α LSBs. We assume that at each position of these α LSBs, the probability of 1 and probability of 0 are equal. The decimal value of the α LSBs is in the range $q \in [0, 2^\alpha - 1]$, with probability $\Pr(q) = 1/2^\alpha$. Thus, the MSE introduced by replacing the α LSBs to zeros can be calculated as

$$\begin{aligned}\mathrm{MSE}_{\mathcal{N}} &= \sum_{q=0}^{2^\alpha-1} (q-0)^2 \Pr(q) = \sum_{q=0}^{2^\alpha-1} q^2/2^\alpha \\ &= \frac{1}{6}\left(2^{2\alpha+1} - 3\times 2^\alpha + 1\right). \end{aligned} \tag{29}$$

To calculate the MSE introduced by the $d-2$ LSBs in each coordinate of $\mathcal{E}$, we focus on the probability model only for the $d-2$ LSBs. Let p and q be the integers corresponding to the $d-2$ LSBs before and after embedding, respectively. Let $\Pr(p)$ and $\Pr(q)$ be the probability distributions of p and q, respectively. Then, the MSE can be calculated as:

$$\mathrm{MSE}_{\mathcal{E}} = \sum_{p=0}^{p_{\max}} \sum_{q=0}^{q_{\max}} (p-q)^2 \Pr(p)\Pr(q). \tag{30}$$

The distribution of q can be well modeled by uniform distribution because the embedded payload is encrypted so that at each bit position, 1 and 0 are equally likely (see (12)). Thus, we may assume that $\Pr(q) = 1/q_{\max}$, for $0 \le q \le q_{\max}$. The distribution $\Pr(p)$ of the original coordinates can be estimated from the histogram. To reduce the computational complexity of (30), we quantize the range of $[0, q_{\max}]$ to n_{B} bins, and the corresponding histograms are denoted as $g_{\mathrm{H}}(p)$ and $g_{\mathrm{U}}(q)$. Their corresponding bin centers are $[p_1, \cdots, p_{n_{\mathrm{B}}}]$. Using a coarsely quantized histogram, the $\mathrm{MSE}_{\mathcal{E}}$ can be estimated as:

$$\mathrm{MSE}_{\mathcal{E}} \simeq \sum_{i=1}^{n_{\mathrm{B}}} \sum_{j=1}^{n_{\mathrm{B}}} \left(p_i - p_j\right)^2 g_{\mathrm{H}}(p_i)\, g_{\mathrm{U}}(p_j), \tag{31}$$

where $g_{\mathrm{U}}(p_j) = \sum_{q=p_j-\Delta/2}^{p_j+\Delta/2} 1/q_{\max} = 1/n_{\mathrm{B}}$, and $\Delta = q_{\max}/n_{\mathrm{B}}$ is bin width. Experiments show that using $n_{\mathrm{B}} = 50$ provides a very accurate prediction.

A simpler expression of $\mathrm{MSE}_{\mathcal{E}}$ can be obtained by assuming that the original values are equally likely and that the values of the replaced bits are also equally likely. The replacement bits are independent of the original bits. The values that can be taken are limited to the range $q \in [0, q_{\max}]$, where $q_{\max} = 2^{d-2} - 1$, with equal probability $\Pr(q) = 1/q_{\max}$. The expected total MSE introduced by each coordinate is

$$\mathrm{MSE}_{\mathcal{E}} = \sum_{p=0}^{q_{\max}} \sum_{q=0}^{q_{\max}} (p-q)^2 \Pr(p)\Pr(q) \tag{32}$$

$$= \frac{1}{q_{\max}^2}\left[\sum_{p=0}^{q_{\max}} \sum_{q=0}^{q_{\max}} p^2 + \sum_{p=0}^{q_{\max}} \sum_{q=0}^{q_{\max}} q^2\right] \tag{33}$$

$$\begin{aligned} &- \frac{1}{q_{\max}^2}\left[2\sum_{p=0}^{q_{\max}} \sum_{q=0}^{q_{\max}} pq\right] \\ &= \frac{1}{6}\left(q_{\max}^2 - 1\right), \end{aligned} \tag{34}$$

where the last equality is obtained by resorting to the well-known summation of series $\sum_{k=1}^{n} k^2 = n(n+1)(2n+1)/6$ and $\sum_{k=1}^{n} = n(n+1)/2$.

Finally, the total MSE can be calculated as

$$\mathrm{MSE}\left(\mathcal{V}, \mathcal{V}_w\right) = \frac{|\mathcal{N}| \times \mathrm{MSE}_{\mathcal{N}} + |\mathcal{E}| \times \mathrm{MSE}_{\mathcal{E}}}{3M}. \tag{35}$$

Combining the simplification result in (32) and the result in (29), we may further obtain

$$\begin{aligned}\mathrm{MSE}\left(\mathcal{V}, \mathcal{V}_w\right) &= \frac{1}{18M}\left(2^{2\alpha+1} - 3\cdot 2^\alpha + 1\right) \\ &\quad\times \left(3\,|\mathcal{P}| - \sum_{e=-\mu}^{\mu} h(e)\right) \\ &\quad+ \frac{1}{18M}\left(q_{\max}^2 - 1\right)\sum_{e=-\mu}^{\mu} h(e) \\ &\approx \frac{1}{18M}\left(q_{\max}^2 - 1\right)\sum_{e=-\mu}^{\mu} h(e). \end{aligned} \tag{36}$$

The last approximation is true by considering the fact that for $q_{\max} = 2^{d-2} - 1$ and typical values $d = 32$ and $\alpha = 4$, the term with $q_{\max}^2$ dominates. The equation in (36) shows an explicit relationship between threshold μ and MSE of the scrambled and embedded mesh. From the desired MSE, one can find an appropriate μ. Note that this μ can be determined before actual embedding and scrambling. The two MSE predictors (31) and (36) are denoted as $\mathcal{M}_{\mathrm{H}}(\mu)$ and $\mathcal{M}_{\mathrm{U}}(\mu)$, respectively. Their performances are tested in Section IV.

REFERENCES

[1] Y. L. Moon, K. Sugamoto, A. Paoluzzi, A. Di Carlo, J. Kwak, D. S. Shin, D. O. Kim, D. H. Lee, and J. Kim, "Standardizing 3D medical imaging," *Computer*, vol. 47, no. 4, pp. 76–79, Apr. 2014.

[2] L. Zhang, Y. Luo, F. Tao, B. H. Li, L. Ren, X. Zhang, H. Guo, Y. Cheng, A. Hu, and Y. Liu, "Cloud manufacturing: A new manufacturing paradigm," *Enterprise Inf. Syst.*, vol. 8, no. 2, pp. 167–187, 2014.

[3] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang, "Edge computing in IoT-based manufacturing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 103–109, Sep. 2018.

[4] A. Galletta, L. Carnevale, A. Celesti, M. Fazio, and M. Villari, "A cloud-based system for improving retention marketing loyalty programs in industry 4.0: A study on big data storage implications," *IEEE Access*, vol. 6, pp. 5485–5492, 2017.

[5] H. Luo, T.-S. Pan, J.-S. Pan, S.-C. Chu, and B. Yang, "Development of a three-dimensional multimode visual immersive system with applications in telepresence," *IEEE Syst. J.*, vol. 11, no. 4, pp. 2818–2828, Dec. 2017.

[6] L. Tawalbeh, M. Mowafi, and W. Aljoby, "Use of elliptic curve cryptography for multimedia encryption," *IET Inf. Secur.*, vol. 7, no. 2, pp. 67–74, 2013.

[7] J.-S. Pan, C.-Y. Lee, A. Sghaier, M. Zeghid, and J. Xie, "Novel systolization of subquadratic space complexity multipliers based on toeplitz matrix–Vector product approach," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 7, pp. 1614–1622, Jul. 2019.

[8] T.-Y. Wu, C.-M. Chen, K.-H. Wang, C. Meng, and E. K. Wang, "A provably secure certificateless public key encryption with keyword search," *J. Chin. Inst. Eng.*, vol. 42, pp. 1–9, Jan. 2019.

[9] C.-M. Chen, B. Xiang, Y. Liu, and K.-H. Wang, "A secure authentication protocol for Internet of vehicles," *IEEE Access*, vol. 7, pp. 12047–12057, 2019.
[10] B. Yan, Y. Xiang, and G. Hua, "Improving the visual quality of size-invariant visual cryptography for grayscale images: An analysis-by-synthesis (AbS) approach," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 896–911, Feb. 2019.
[11] Y.-Q. Shi, X. Li, X. Zhang, H.-T. Wu, and B. Ma, "Reversible data hiding: Advances in the past two decades," *IEEE Access*, vol. 4, pp. 3210–3237, 2016.
[12] S. Weng, Y. Shi, W. Hong, and Y. Yao, "Dynamic improved pixel value ordering reversible data hiding," *Inf. Sci.*, vol. 489, pp. 136–154, Jul. 2019.
[13] S. Weng, Y. Chen, B. Ou, C.-C. Chang, and C. Zhang, "Improved k-pass pixel value ordering based data hiding," *IEEE Access*, vol. 7, pp. 34570–34582, 2019.
[14] Z. Ni, Y. Shi, N. Ansari, and W. Su, "Reversible data hiding," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 2, May 2003, pp. 354–362.
[15] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 8, pp. 890–896, Aug. 2003.
[16] L. Luo, Z. Chen, M. Chen, X. Zeng, and Z. Xiong, "Reversible image watermarking using interpolation technique," *IEEE Trans. Inf. Forensics Security*, vol. 5, no. 1, pp. 187–193, Mar. 2010.
[17] X. Zhang, "Reversible data hiding in encrypted image," *IEEE Signal Process. Lett.*, vol. 18, no. 4, pp. 255–258, Apr. 2011.
[18] F. Huang, J. Huang, and Y.-Q. Shi, "New framework for reversible data hiding in encrypted domain," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2777–2789, Dec. 2016.
[19] Q. Li, B. Yan, H. Li, and N. Chen, "Separable reversible data hiding in encrypted images with improved security and capacity," *Multimedia Tools Appl.*, vol. 77, no. 23, pp. 30749–30768, 2018.
[20] X. Mao, M. Shiba, and A. Imamiya, "Watermarking 3D geometric models through triangle subdivision," *Proc. SPIE*, vol. 4314, pp. 253–261, Aug. 2001.
[21] Z.-M. Lu and Z. Li, "High capacity reversible data hiding for 3D meshes in the PVQ domain," in *Proc. Int. Workshop Digit. Watermarking*. Berlin, Germany: Springer, 2007, pp. 233–243.
[22] F. Cayre and B. Macq, "Data hiding on 3-D triangle meshes," *IEEE Trans. Signal Process.*, vol. 51, no. 4, pp. 939–949, Apr. 2003.
[23] J. Dittmann and O. Benedens, "Invertible authentication for 3D meshes," *Proc. SPIE*, vol. 5020, Jun. 2003, pp. 653–665.
[24] R. Jiang, H. Zhou, W. Zhang, and N. Yu, "Reversible data hiding in encrypted three-dimensional mesh models," *IEEE Trans. Multimedia*, vol. 20, no. 1, pp. 55–67, Jan. 2018.
[25] C. I. Tan, C.-K. Lin, W.-K. Tai, and C.-C. Chang, "Hiding data: A high-capacity distortionless approach," *Multimedia Syst.*, vol. 15, no. 6, p. 325, 2009.
[26] S.-C. Tu, W.-K. Tai, M. Isenburg, and C.-C. Chang, "An improved data hiding approach for polygon meshes," *Vis. Comput.*, vol. 26, no. 9, pp. 1177–1181, 2010.
[27] S. Tu, H. Hsu, and W. Tai, "Permutation steganography for polygonal meshes based on coding tree," *Int. J. Virtual Reality*, vol. 9, no. 4, pp. 55–60, 2010.
[28] H.-T. Wu and Y.-M. Cheung, "A reversible data hiding approach to mesh authentication," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell.*, Sep. 2005, pp. 774–777.
[29] H.-T. Wu and J.-L. Dugelay, "Reversible watermarking of 3D mesh models by prediction-error expansion," in *Proc. IEEE 10th Workshop Multimedia Signal Process.*, Oct. 2008, pp. 797–802.
[30] A. Zhu, C. Zhang, X. Yang, and X. Gao, "Reversible watermarking of 3D mesh models using prediction-error expansion," in *Proc. 3rd Int. Congr. Image Signal Process.*, Oct. 2010, pp. 1171–1175.
[31] A. M. Molaei, H. Ebrahimnezhad, and M. H. Sedaaghi, "A blind fragile watermarking method for 3D models based on geometric properties of triangles," *3D Res.*, vol. 4, no. 4, p. 4, 2013.
[32] Y.-H. Huang and Y.-Y. Tsai, "A reversible data hiding scheme for 3D polygonal models based on histogram shifting with high embedding capacity," *3D Res.*, vol. 6, no. 2, p. 20, 2015.
[33] Q. Zhang, X. Song, T. Wen, and C. Fu, "Reversible data hiding for 3D mesh models with hybrid prediction and multilayer strategy," *Multimedia Tools Appl.*, vol. 78, no. 21, pp. 29713–29729, 2018.
[34] Q. Zhang, X. Song, T. Wen, and C. Fu, "Reversibility improved data hiding in 3D mesh models using prediction-error expansion and sorting," *Measurement*, vol. 135, pp. 738–746, Mar. 2019.
[35] H. Luo, Z.-M. Lu, and J.-S. Pan, "A reversible data hiding scheme for 3D point cloud model," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, Aug. 2006, pp. 863–867.
[36] K. Wang, G. Lavoue, F. Denis, and A. Baskurt, "Hierarchical watermarking of semiregular meshes based on wavelet transform," *IEEE Trans. Inf. Forensics Security*, vol. 3, no. 4, pp. 620–634, Dec. 2008.
[37] X. Feng, W. Zhang, and Y. Liu, "Double watermarks of 3D mesh model based on feature segmentation and redundancy information," *Multimedia Tools Appl.*, vol. 68, no. 3, pp. 497–515, 2014.
[38] Z. Sun, Z.-M. Lu, and Z. Li, "Reversible data hiding for 3D meshes in the PVQ-compressed domain," in *Proc. Int. Conf. Intell. Inf. Hiding Multimedia*, Dec. 2006, pp. 593–596.
[39] X. Zhang, "Separable reversible data hiding in encrypted image," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 826–832, Apr. 2012.
[40] M. Fujiyoshi, "Separable reversible data hiding in encrypted images with histogram permutation," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops (ICMEW)*, Jul. 2013, pp. 1–4.
[41] S. Ong, K. S. Wong, and K. Tanaka, "A scalable reversible data embedding method with progressive quality degradation functionality," *Signal Process., Image Commun.*, vol. 29, no. 1, pp. 135–149, 2014.
[42] R. M. Rad, K. Wong, and J.-M. Guo, "A unified data embedding and scrambling method," *IEEE Trans. Image Process.*, vol. 23, no. 4, pp. 1463–1475, Apr. 2014.
[43] M. Deering, "Geometry compression," in *Proc. 22nd Annu. Conf. Comput. Graph. Interact. Techn.*, 1995, pp. 13–20.
[44] R. Jiang, W. Zhang, D. Hou, H. Wang, and N. Yu, "Reversible data hiding for 3D mesh models with three-dimensional prediction-error histogram modification," *Multimedia Tools Appl.*, vol. 77, no. 5, pp. 5263–5280, 2018.
[45] M. Robshaw and O. Billet, *New Stream Cipher Designs: The ESTREAM Finalists*, vol. 4986. Berlin, Germany: Springer, Jan. 2008, pp. 1–6.
[46] S. C. Chapra and R. P. Canale, *Numerical methods for engineers*. Boston, MA, USA: McGraw-Hill, 2010.
[47] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*. New York, NY, USA: Dover, 1965.
[48] Q.-Y. Zhang, Q.-Y. Dou, R.-H. Dong, and Y. Yan, "Robust reversible data hiding algorithm for color image based on 2D-DCT," *J. Inf. Hiding Multimedia Signal Process.*, vol. 8, no. 2, pp. 392–403, 2017.
[49] *Basics About 3D Meshes*. Accessed: Apr. 4, 2018. [Online]. Available: http://www.numerical-tours.com/matlab/meshproc_2_basics_3d/
[50] *Princeton Shape Benchmark*. Accessed: May 16, 2019. [Online]. Available: http://shape.cs.princeton.edu/benchmark/
[51] *Laserdesign*. Accessed: May 22, 2019. [Online]. Available: https://www.laserdesign.com/sample-files/type/obj/
[52] O. Sorkine, "Laplacian mesh processing," in *Proc. Eurographics (STARs)*, 2005, pp. 53–70.
[53] K. Zhou, J. Huang, J. Snyder, X. Liu, H. Bao, B. Guo, and H.-Y. Shum, "Large mesh deformation using the volumetric graph Laplacian," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 496–503, 2005.

LIU-YAO HAO received the B.E. degree in electronic information engineering from the College of Information Science and Engineering, Wanfang College of Science and Technology, Henan Polytechnic University, China, in 2017. She is currently pursuing the master's degree with the College of Electronic and Information Engineering, Shandong University of Science and Technology. Her research interests include 3D model processing and reversible data hiding in the encryption domain.

BIN YAN received the B.S. degree in applied physics from Qingdao University, China, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from the Harbin Institute of Technology, China, in 2002 and 2007, respectively. From 1996 to 1999, he was an Engineer with Goma Company Group. From 2007 to 2012, he was a Lecturer with the Shandong University of Science and Technology. From 2015 to 2016, he was a Visiting Scholar with Deakin University, Australia. Since 2013, he has been an Associate Professor with the Department of Communication Engineering, Shandong University of Science and Technology. He published two monographs in multimedia security, including *Improving Image Quality in Visual Cryptography* and *Digital Audio Watermarking – Fundamentals, Techniques and Challenges*. His research interests include statistical signal processing and multimedia signal processing and security.

JENG-SHYANG PAN received the B.S. degree in electronic engineering from the National Taiwan University of Science and Technology, in 1986, the M.S. degree in communication engineering from National Chiao Tung University, Taiwan, in 1988, and the Ph.D. degree in electrical engineering from the University of Edinburgh, U.K., in 1996. He was the Director of the Fujian Provincial Key Lab of Big Data Mining and Applications and an Assistant President with the Fujian University of Technology. He is currently with the College of Computer Science and Engineering, Shandong University of Science and Technology. He is an IET Fellow, U.K., and has been the Vice-Chair of the IEEE Tainan Section. He was offered the Thousand Talent Program in China, in 2010.

NA CHEN received the Ph.D. degree from Xidian University, China, in 2016. She is currently with the Department of Communication Engineering, College of Electronic and Information Engineering, Shandong University of Science and Technology. Her research interests include cryptography and quantum information processing.

HONG-MEI YANG was born in Shandong, China, in 1969. She received the Ph.D. degree from the Shandong University of Science and Technology, in 2009. She is currently with the College of Computer Science and Engineering, Shandong University of Science and Technology. Her research interests include digital watermarking and image quality evaluation.

MOSES ARHINFUL ACQUAH received the B.E. degree in communication engineering from the Shandong University of Science and Technology, China, in 2017. He is currently pursuing the master's degree with the College of Electronic and Information Engineering, Shandong University of Science and Technology. His research interests include multimedia security and blockchains.