Separable Reversible Data Hiding in Encrypted 3D Mesh Models Based on Spatial Clustering and Multi-MSB Prediction

Yangsu Mao[1], Yanyan Xu[1]

[1]State Key Laboratory of Information Engineering in Surveying, Mapping and Remote Sensing, Wuhan University, Wuhan 430072, China - (Email: {maoyangsu, xuyy} @whu.edu.cn).

Keywords: Reversible data hiding, Encrypted domain, Three-dimensional model, Spatial clustering, Multi-MSB prediction.

Abstract

Reversible data hiding in the encrypted domain (RDH-ED) involves performing data encryption to protect privacy and hiding data for covert communication or access control. Current research mainly focuses on exploring spatial correlation of adjacent vertices and local spatial correlation to reserve embedding space, while ignoring global spatial correlation, limiting embedding capacity. To address this, a method combining spatial clustering with multi-MSB (multiple most significant bit) prediction is proposed to enhance embedding rate and capacity. Input model vertices are partitioned into exclusive clusters through spatial clustering, ensuring close proximity within each cluster. An optimal reference vertex search algorithm is then used to identify the best reference vertex in each cluster to maximize embeddable capacity, and multi-MSB prediction determines the embedding length for each embeddable vertex. The basic embedding length of the reference vertex also serves as additional embedding space, further boosting capacity. Finally, data are embedded in the reserved space through bit substitution. Experimental results demonstrate that the proposed method achieves higher embedding capacity and low computational overhead while supporting reversibility and separability. The proposed scheme provides a robust, high-capacity, and scalable solution for secure data embedding in encrypted 3D graphics, which is increasingly relevant in real-world 3D, geological model, and virtual reality applications.

1. Introduction

With the development of computer graphics and increased computational power in the big data era, the digital representation of the physical world has shifted to three dimensions. Three-dimensional (3D) data are extensively used in various domains (e.g., medicine, art, and geographic information systems). Due to the complex structure and large volume of 3D data, outsourcing its storage to Cloud servers has become common practice. While Cloud services enhance the convenience of storing and accessing 3D data, they also pose security risks. Additionally, Cloud data managers often need to embed labels and annotations into the data to effectively manage and protect the vast amounts of information on the Cloud (Hou et al., 2024), but this process can cause irreversible distortion, which is unacceptable in applications requiring precise content, such as medicine, geography, and art. Reversible data hiding in the encrypted domain (RDH-ED) allows for the embedding and extraction of secret messages without exposing and compromising the carrier data. RDH-ED is a growing research focus in information hiding. Nevertheless, compared to reversible data hiding in plaintext, encryption eliminates redundancy, leading to lower payload capacity in encrypted domains. Moreover, unlike images, 3D models have complex structures, uneven vertex distributions, and irregular geometric shapes. The complex spatial relationships and structural information of 3D models make it difficult to reserve spatial redundancy for embedding data in the encrypted domain. Therefore, achieving high-capacity data embedding while ensuring reversibility remains a challenge in reversible data-hiding for encrypted 3D data.

Current research on RDH-ED has primarily focused on image data (Qiu et al., 2022, Yin et al., 2022), with limited studies on RDH-EDs for 3D data. Existing RDH-ED methods for 3D models are simplistic extensions of image-based methods and overlook the global spatial correlations of 3D model. These methods are categorized into vacating rooms after encryption (VRAE) and reserving rooms before encryption (RRBE). VRAE-based methods (Jiang et al., 2018, Tsai, 2021, Shah et al., 2018, Van Rensburg et al., 2021) utilize spatial correlations of encrypted carriers to embed data but may suffer high error rates in data extraction or model recovery, making them not fully reversible. Additionally, the reduced spatial correlation in encrypted carriers limits the embedding capacity. In contrast, RRBE-based RDH-ED methods utilize the spatial correlation of the original content to reserve embedding space, encrypt the model, and embed data into the reserved space, achieving reversibility and higher embedding capacity.

The embedding capacity improves when reference vertices are closer to the embedded vertices (Tsai and Liu, 2023). As neighboring vertices tend to be spatially close, hence some RRBE schemes (Yin et al., 2021, Xu et al., 2022, Lyu et al., 2022, Tang et al., 2023, Tsai and Liu, 2023) exploit spatial correlations of adjacent vertices by partitioning neighboring spaces to increase the number of embeddable vertices or reduce the distance between embedded and reference vertices, thus improving the embedding capacity. These methods partition vertices within limited neighboring spaces, failing to ensure global proximity between embedded and reference vertices, and have fewer embeddable vertices. Therefore, relying solely on adjacent vertex correlations results in relatively low embedding capacity and rate.

Unlike the above approaches, the method in (Hou et al., 2024) leveraged local spatial correlations to reserve embedding space, significantly improving embedding capacity. This approach divided the original model into subblocks utilizing an octree and applied multi-MSB prediction to each subblock to determine its embedding length. However, the octree's equal division leads to loss of global spatial correlation, causing subblock vertices to not necessarily be the closest globally, resulting in suboptimal embedding capacity. Furthermore, some methods in (Yin et al., 2021, Xu et al., 2022, Lyu et al., 2022, Tang et al., 2023, Tsai and Liu, 2023, Hou et al., 2024) employ reference vertices to recover coordinates of embedded vertices rather than embedding data, reducing the embeddability rate and overall embedding capacity.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-G-2025-567-2025

In summary, the challenges in RDH-ED stem from inadequate exploration of global spatial correlations for reserving embedding space and underuse of reference vertices for data embedding, resulting in limited capacity. We propose a reversible data-hiding method that integrates spatial clustering with multi-MSB prediction to enhance embedding capacity. First, we introduce a spatial subdivision method based on spatial clustering, leveraging global spatial correlations to group nearest neighbor vertices into the same subspace, enhancing vertex correlations, and increasing capacity. Second, we propose an optimal reference vertex search algorithm that maximizes the embedding capacity by leveraging local spatial correlations to adaptively identify the best reference vertex within each subspace. Additionally, a cluster-index-based grouping strategy is employed to remesh the 3D model, converting the cluster index information to reduce the overhead. Multi-MSB prediction is used to calculate the embeddable length of embeddable vertices, and an embedding strategy using the basic embedding length of the reference vertex also as the reserved embedding space is proposed to boost the embedding capacity. A label map is compressed using Huffman coding, and secret messages are embedded in the multi-MSB of the vertex coordinate values.

The main contributions of this paper are as follows:

- A spatial clustering-based subdivision strategy is first proposed to fully exploit global spatial correlations of the input model for reserving embedding space. This strategy iteratively calculates distances between vertices and centroids to partition vertices into mutually exclusive clusters, enhancing prediction efficiency by grouping closely related vertices.
- An optimal reference vertex search algorithm is proposed to maximize embeddable capacity within each cluster by exploring local spatial correlations. A cluster-index-based grouping strategy and an embedding strategy using the basic embedding length of reference vertex as reserved space are proposed to achieve higher embedding rates and overall capacity.
- Experimental results demonstrate that the proposed algorithm supports lossless model recovery, error-free data extraction, and separability. Compared to state-of-the-art methods, our algorithm increased the embedding rate to 100% and achieved an average 32% gain in the embedding capacity.

The remainder of this paper is organized as follows: Section 2 details the proposed algorithm, Section 3 presents performance evaluations, and Section 4 provides the conclusions.

2. Proposed Method

This section details the proposed high-capacity separable reversible data-hiding algorithm for encrypted 3D models. The framework of the proposed method is illustrated in Figure 1. and comprises three main parts: (1) The model owner starts by applying spatial clustering to divide the vertices of the input model into mutually exclusive clusters. Prediction-error detection is then conducted to determine the embedding length for each embeddable vertex. The embedding lengths and number of vertices per cluster are expressed in a label map of the input model. The model owner then encrypts the model using an encryption key and embeds the label map into the reserved embedding space; (2) The data hider extracts the label map to identify the reserved embedding space and uses a data-hiding key to embed additional data, resulting in a marked encrypted model; (3) The receiver extracts the additional data and recovers the original model using the provided secret keys.

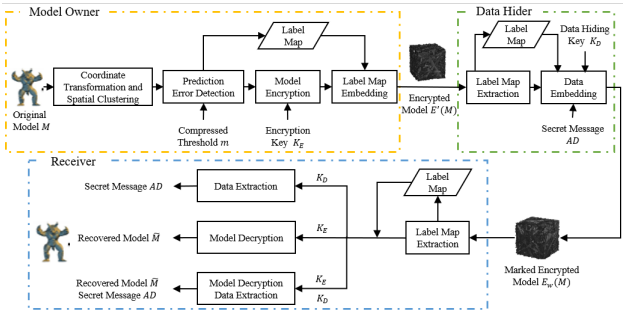


Figure 1. Framework of the proposed method.

2.1 Coordinate Transformation

In the coordinate transformation process, a 3D model is represented as $M = \{V, F\}$, where the set of vertices $V = \{v_1, v_2, \cdots, v_N\}$ and the set of surfaces $F = \{f(v_i, v_j, v_k) | v_i, v_j, , v_k \in V\}$. The smallest boundary vertex is determined using Eq. (1).

$$B_{min} = \begin{bmatrix} x_m \\ y_m \\ z_m \end{bmatrix} = \begin{bmatrix} \min\limits_{i\in\{1,2,\cdots N\}} x_i \\ \min\limits_{i\in\{1,2,\cdots N\}} y_i \\ \min\limits_{i\in\{1,2,\cdots N\}} z_i \end{bmatrix} \quad (1)$$

A coordinate transformation procedure based on Eq. (2) is applied to each vertex, where R_i is a decimal value between 0 and 1, and k is the longest integer digit of all the shifted vertex coordinate values (Tsai and Liu, 2023):

$$R_i = \begin{bmatrix} \frac{x_i - min(x_m, y_m, z_m)}{10^k} \\ \frac{y_i - min(x_m, y_m, z_m)}{10^k} \\ \frac{z_i - min(x_m, y_m, z_m)}{10^k} \end{bmatrix} \quad (2)$$

Therefore, each vertex of the input model can be converted into a tuple of decimal values R_i. The model owner can then use a compression threshold m to reserve partial digits of each vertex coordinate, transforming them into integer values C_i as per Eq. (3), and recover the decimal value R_i' using Eq. (4), where $\lfloor . \rfloor$ and m represent the floor function and compression threshold, respectively. A larger compression threshold m results in the recovered model being closer to the original model:

$$C_i = \lfloor R_i \times 10^m \rfloor \quad (3)$$

$$R_i' = \frac{C_i}{10^m} \quad (4)$$

The integer coordinate values C_i of the compressed model are converted into fixed-length binary representations for storage, as determined by Eq. (5). The binary representation length L is based on the compression threshold m, as shown in Eq. (6). Table 1 presents the relationships between the binary representation length L of the vertex coordinates, the basic embedding length EL_B, and the compression threshold m.

$$C_i = \sum_{p=0}^{L-1} c_{i,p} \times 2^p \quad (5)$$

$$L = \begin{cases} 8, & 1 \le m \le 2 \\ 16, & 3 \le m \le 4 \\ 32, & 5 \le m \le 9 \\ 64, & 10 \le m \le 33 \end{cases} \quad (6)$$

m	L	EL_B	m	L	EL_B
1	8	4	10	64	30
2	8	1	11	64	27
3	16	6	12	64	24
4	16	2	13	64	20
5	32	15	14	64	17
6	32	12	15	64	14
7	32	8	16	64	10
8	32	5	17	64	7
9	32	2	18	64	4

Table 1. Binary representation length L and basic embedding length EL_B under the compression thresholds m

2.2 Spatial Clustering and Multi-MSB Prediction

The purpose of this process is to divide the vertices of the input model into non-overlapping, compact subspaces and perform multi-MSB prediction to reserve embedding space. First, the model owner performs spatial clustering to divide the vertices into mutually exclusive clusters, ensuring that vertices within each cluster are as closely grouped as possible. Following this, the optimal reference vertex search algorithm is employed to select the best reference vertex in each cluster, maximizing the embedding length. The model is then remeshed using a cluster index-based grouping method. Finally, a prediction-error detection process is applied to determine the embedding length for each vertex coordinate value. The detailed steps of this process are as follows.

2.2.1 Spatial Clustering for 3D Model: The model owner performs spatial clustering (Xie and Jiang, 2010, Park and Jun, 2009, Rodriguez and Laio, 2014, Ng et al., 2001) to partition the vertices of the input model M into mutually exclusive clusters, ensuring that vertices within each cluster are as close to each other as possible. This spatial subdivision method, based on spatial clustering. effectively creates compact subspaces and maximizes the embedding length. The subdivision results are represented as:

$$C = \{c_1, c_2, \cdots, c_i, \cdots c_k\} \quad (7)$$

$$idx = \{idx_1, idx_2, \cdots, idx_i, \cdots idx_N | idx_i \in [1, k]\} \quad (8)$$

where k is the number of clusters, c_i and idx_i are the i-th cluster and the index of the cluster to which the i-th vertex, respectively.

2.2.2 Search for Best Reference Vertex: Each cluster c_i is treated as an independent unit. Within each cluster, every vertex is sequentially considered as a reference vertex, and the embedding lengths of the other embeddable vertices in the cluster are measured. The goal is to select an optimal reference vertex that maximizes the sum of the embedding lengths of the other embeddable vertices in the cluster. The search results are obtained using Eq. (9) and (10):

$$arg \max_r \sum_{r,e \in c_i} I_{r,e} - L_{c_i} \approx arg \max_r \textstyle\sum_{r,e \in c_i} I_{r,e} - \omega, \forall r \in c_i;\ c_i \in C \quad (9)$$

$$I_{r,e} = arg \max_t (v_r^{tMSB} = v_e^{tMSB}), t = 1,2, \cdots, L; r, e \in c_i \quad (10)$$

where r and e are the reference vertex and embeddable vertex within cluster c_i, $I_{r,e}$ represents the embedding length of each embeddable vertex, and L_{c_i} is the size of the label map for cluster c_i, L_{c_i} is approximately constant when the number of vertices in the cluster does not change. v_r^{tMSB} and v_e^{tMSB} are the t-MSB (MSBs of the coordinate) values of the reference vertex r and the embedding vertex e, respectively.

2.2.3 Remesh 3D Model: After obtaining the cluster index and the best reference vertex for each cluster, the vertices of the input model are grouped according to their cluster indices. The best reference vertex of each cluster is positioned first position within its group, and the surface indices are updated accordingly as the vertex indices are reordered. This approach ensures that vertices within the same cluster are arranged sequentially, allowing for clear differentiation of clusters by using information on the number of vertices in each cluster.

2.2.4 Perform Multi-MSB Prediction: In cluster c_i, the best reference vertex, obtained through the optimal reference vertex search algorithm, is used to restore the coordinate values of the other embeddable vertices. The compression threshold m serves as prior knowledge of the basic embedding length of the reference vertex. Specifically, when the compression threshold is known, the binary digits within the basic embedding length of the reference vertex are consistently zero. By recording the compression threshold in the label map, the basic embedding length of the reference vertex can be reserved as embedding space for data hiding.

For embeddable vertices, the coordinate values are transformed into a series of binary digits of length L. The prediction-error detection process is applied to both the best reference vertex and the embeddable vertex. This involves comparing binary digits of the reference vertex and the embeddable vertex, starting from the MSB down to the LSB, until a difference between the two digits is found. The comparative length is the embedding length I of the processing vertex. The embedding length I comprises the basic embedding length EL_B and the prediction embedding length EL_P. The multi-MSB prediction is given in Eq. (11) and Eq. (12), where $v_{r,1,j}^{tMSB}$ and $v_{e,i,j}^{tMSB}$ represent the t-MSB MSBs of the coordinate values of the reference vertex r and the embedding vertex e in the direction of $x-, y-, and\ z-$axes, respectively. N_c is the number of vertices in the cluster c_i.

$$I_j = arg \max_t (v_{e,i,j}^{tMSB} = v_{r,1,j}^{tMSB}), t = 1,2, \cdots, L; \quad i = 2,3, \cdots, N_c; j = x, y, z \quad (11)$$

$$I = min(I_x, I_y, I_z) \quad (12)$$

The prediction-error detection process is illustrated in Figure 2. For example, with $m = 4$, the vertex coordinates in cluster c_i are transformed into corresponding 16-bit binaries. As described in Section 3.2, vertex v_1, first in its group, is the best reference vertex for cluster c_i, the basic embedding length EL_B (highlighted in orange in Figure 2) of vertex v_1 is used as reserved embedding space. For each embeddable vertex, the corresponding 16-bit binaries of the embeddable vertex and the best reference vertex are compared sequentially from the MSB to the LSB until a difference appears. If the embedding vertex v_4 differs from v_1 at the 9th bit plane, the embedding position on the $x-$axis for v_4 is first eight bits. The embedding length is divided into a basic embedding length (shown in orange in Figure 2) and a predicted embedding length (shown in blue in Figure 2). The embedding lengths for $x-, y-, and\ z-$axes of each embeddable vertex are calculated sequentially to obtain the final embedding length $3 \times min(I_x, I_y, I_z)$. That is, the embeddable vertex can provide $3 \times min(I_x, I_y, I_z)$ bits of spatial redundancy for embedding data. This process is repeated for each cluster to compute the total embedding capacity.

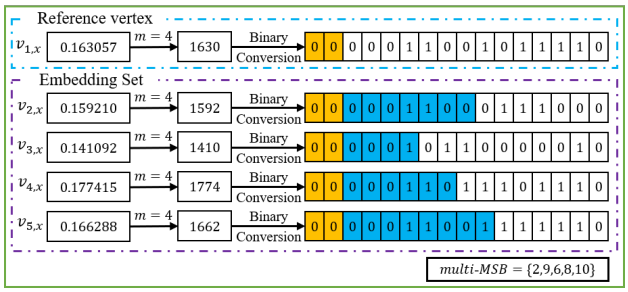


Figure 1. Example of prediction-error detection process for cluster in x-axis.

2.3 Encryption

To effectively enhance the privacy of the input model, the model owner uses the encryption key K_E to generate a series of binary random sequences b, and converts the coordinate values of the input model into binary representations $c_{i,p}$. For each vertex coordinate, the binary random sequence b is continuously sampled for L bits. The exclusive-OR (XOR) operation is then performed between the binary representation of the vertex coordinate values by using Eq. (13). Where $c_{i,p}$ is the binary representation of the coordinate values of the input model, $b_{i,p}$ is the sampled binary random sequence, and L is the binary representation length based on compression threshold m. Finally, the encrypted model can be obtained.

$$c'_{i,p} = c_{i,p} \oplus b_{i,p}, where\ p = 0,1,\cdots L-1 \tag{13}$$

2.4 Label Map

In this process, the model owner uses Huffman coding to compress the label map and embed it into the encrypted model. The embedding lengths, compression threshold, and vertex counts for each cluster are compiled into a label map. The compression threshold is encoded at a fixed length. The Huffman coding tree structure is generated using a tree-structured coding process according to the frequency of embedding length symbols for each embeddable vertex and the frequency of the number symbols corresponding to the vertices in each cluster. By traversing the constructed Huffman coding binary tree, the encoding results for different embedding lengths and vertex counts can be obtained.

The final label map is represented as a binary sequence of length HL, which is determined from encoding the compression threshold, the Huffman coding tree structure, and the encoding results. The label map is then embedded into the basic embedding length EL_B of each vertex (including reference vertices) using bit substitution. If all basic embedding lengths are used, the predicted embedding length EL_P are replaced. This results in an encrypted model with an embedded label map.

As shown in Figure 3, the label map is first embedded into the basic embedding lengths of the encrypted vertices. If all basic embedding lengths are used, the predicted embedding length EL_P can be replaced to store the encoding results. In Figure 3, the yellow and green areas represent the embedding regions for the Huffman coding tree structure and the encoding results, respectively. The encrypted model with an embedded label map is obtained by embedding the label map into the reserved embedding space through the bit substitution strategy.

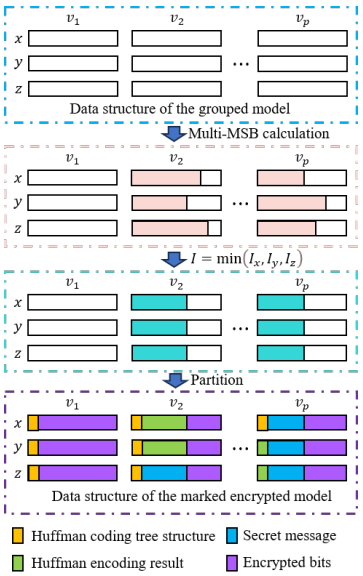


Figure 3. Example of label map embedding and data embedding processes.

2.5 Data Hiding

After receiving the encrypted model with the embedded label map, the data embedding process is initiated by the data hider. To enhance the security of the additional data, a series of binary random sequences s are generated using the data-hiding key K_D. The XOR operation is performed between these sequences and the additional data AD, as described by Eq. (14), where ED_k is k −th bit encrypted additional data.

$$ED_k = s_k \oplus AD_k \tag{14}$$

Next, the data hider extracts the Huffman coding tree structure from the label map to reconstruct the Huffman tree and retrieve the encoding codes for each symbol. This allows the data hider to determine the reserved embedding space. Finally, the encrypted additional data are embedded into the available embedding room through bit substitution, as indicated by the blue areas in Figure 3 to produce the marked encrypted model. The embedding process is defined by Eq. (15), where $E_w(M)_{i,j}$ is i −th vertex coordinate of the marked encrypted model, and j represent the directions of $x-, y-, and\ z$ −axes, respectively.

$$E_w(M)_{i,j} = ED_1 \times 2^{L-1} + ED_2 \times 2^{L-2} + \cdots + ED_{I_i} \times 2^{L-I_i} + E'_i(M) mod 2^{L-I_i}, i = [1, N]; j = x, y, z \tag{15}$$

2.6 Data Extraction and Model Recovery

After receiving the marked encrypted model, the receiver can perform model decryption or data extraction, depending on the keys they possess. There are three cases as follows:

1） The receiver has a decryption key K_E. The receiver extracts the label map to obtain the basic embedding length and embedding length of each vertex coordinate. An XOR operation was performed with a random binary sequence generated according to the decryption key to obtain the initially decrypted model. The binary digits in the basic embedding length of all the reference vertices were replaced with zero to recover the coordinate values of the reference vertices. Finally, the modified MSBs of each embeddable vertex coordinate are restored from their

corresponding reference vertices using the prediction-error detection process.

2） The receiver has a data-hiding key K_D. The receiver first extracts a label map to obtain the reserved embedding room. The encrypted additional data are extracted from the reserved embedding room and an XOR operation is performed with a binary random sequence generated according to the data-hiding key to obtain the decrypted additional data.

3） The receiver has both the decryption key K_E and the data-hiding key K_D. The receiver can recover the original model and extract additional data without any errors. Data extraction and model recovery can be performed in an arbitrary order, which indicates separability.

3. Experimental Results and Analysis

This section outlines the experimental methodology, including the experimental settings, evaluation metrics, and results. The 3D mesh models used are depicted in Figure 4(a), with their parameters listed in Table 2. For performance evaluation, we used the Princeton segmentation benchmark (PSB) (Chen et al., 2009), and the additional data consisted of randomly generated binary sequences. The proposed algorithm was implemented in MATLAB R2022a on a personal computer with an Intel Core i5-10210U 1.60 GHz processor and 16 GB of memory.

Model Name	Number of vertices	Number of surfaces
Bunny	35947	69451
Dragon	437645	871414
Airplane	8679	17354
Bird	5054	10104
Bearing	14994	29984
Happy Buddha	543652	1087716

Table 2. Model information of experiment.

The performance of the proposed scheme was evaluated in terms of model visualization, reversibility, and embedding capacity. Visualization of the model at different stages was demonstrated. Reversibility was assessed through the Hausdorff distance and signal-to-noise ratio (SNR). The Hausdorff distance measures the similarity between two sets of points, with smaller values indicating higher similarity. The Hausdorff distance between two sets of points, $A = (a_1, a_2, \cdots a_p)$ and $B = (b_1, b_2, \cdots b_q)$, is defined as follows:

$$HD(A,B) = max\big(h(A,B), h(B,A)\big) \tag{16}$$

where $h(A,B)$ and $h(B,A)$ are

$$h(A,B) = \max_{a\in A}\min_{b\in B}\|a-b\| \tag{17}$$

$$h(B,A) = \max_{b\in B}\min_{a\in A}\|b-a\| \tag{18}$$

and $\|\cdot\|$ calculates the distance between the two sets of points.

The geometric disparity between the reconstructed and original models can be measured by SNR, with a larger SNR indicating smaller geometric deformation. The SNR is defined as follows:

$$SNR = 10 \times log_{10}\frac{\sum_{i=1}^{N}\left[\left(v_{i,x}-\overline{v_x}\right)^2+\left(v_{i,y}-\overline{v_y}\right)^2+\left(v_{i,z}-\overline{v_z}\right)^2\right]}{\sum_{i=1}^{N}\left[\left(v'_{i,x}-v_{i,x}\right)^2+\left(v'_{i,y}-v_{i,y}\right)^2+\left(v'_{i,z}-v_{i,z}\right)^2\right]} \tag{19}$$

where N is the number of vertices in the model; $v_{i,x}$, $v_{i,y}$, $v_{i,z}$ are the coordinates of the vertices in the original model; $v'_{i,x}$, $v'_{i,y}$, $v'_{i,z}$ are the coordinates of the vertices in the reconstructed model; and $\overline{v_x}$, $\overline{v_y}$, $\overline{v_z}$ are the average coordinates of the original model. Embedding capacity was measured using pure embedding capacity (PEC) and embedding rate. PEC was determined by subtracting the capacity of the embedded label map from the total capacity. A higher PEC value indicates a greater ability to embed additional data. The definitions of total capacity tec and pure embedding capacity pec are as follows:

$$tec = \frac{3\times EL_B\times N_r + 3\times\sum_{i=1}^{N_e}\left(EL_B + EL_{P_i}\right)}{N} = 3\times EL_B + 3\times\frac{\sum_{i=1}^{N_e} EL_{P_i}}{N} \tag{20}$$

$$pec = tec - \frac{size(labelmap)}{N} \tag{21}$$

where N, N_r, and N_e represent the number of vertices in the input model, reference vertices, and embeddable vertices, respectively. EL_B is the basic embedding length of a vertex, EL_{P_i} is the predicted embedding length of the i-th vertex, and $size(\cdot)$ calculates the capacity of the label map.

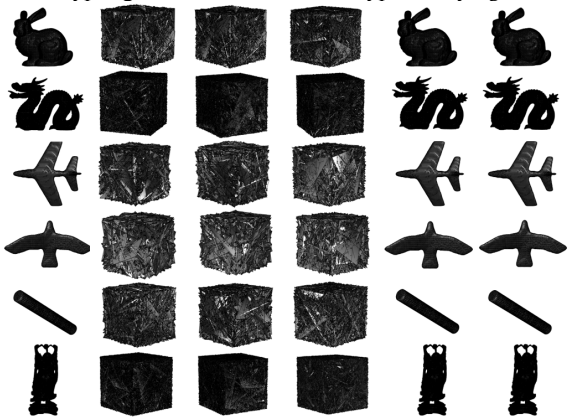
An ablation experiment was conducted to assess the impact of clustering algorithms and the use of the basic embedding length of reference vertices as part of the embedding room on embedding capacity. These evaluations confirmed the effectiveness and feasibility of each component. Finally, comparisons with existing methods in Section 4.5 verified the feasibility of our proposed algorithm.

3.1 Model Visualization

Figure 4 demonstrates the 3D model visualization at different stages of the proposed algorithm. Initially, the original models (Figure 4(a)) are encrypted, resulting in encrypted models (Figure 4(b)). Label maps are then embedded into these encrypted models, producing encrypted models with embedded label maps (Figure 4(c)). The data hider extracts the label maps from these models to identify the reserved embedding room and embeds additional data, resulting in the models shown in Figure 4(d). For recipients, the process of model decryption and data extraction depends on the keys they possess:

1） Models in Figure 4(e) are obtained by decrypting with the encryption key K_D.

2） Models in Figure 4(f) are produced by first extracting additional data using the data-hiding key K_E, followed by decrypting the model with the encryption key K_D.



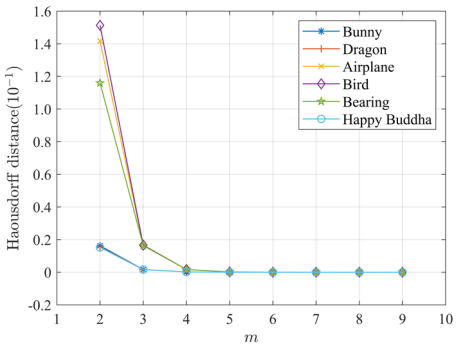
(a) (b) (c) (d) (e) (f)

Figure 4. Result of model visualization. (a) Original models. (b) Encrypted models. (c) Encrypted models with label map. (d) Marked encrypted models. (e) Direct decrypted models. (f) Recovered models with data extraction.

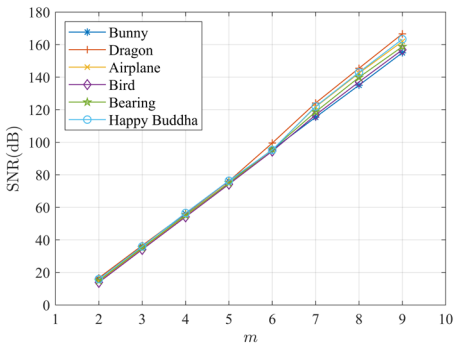
From the model visualization results, it is evident that the proposed algorithm effectively encrypts the original models, securely embeds additional data, and allows for accurate reconstruction of the models. This confirms the algorithm's feasibility and practical applicability.

3.2 Reversibility Analysis

The Hausdorff distance (HD) and SNR were used to evaluate the quality of the reconstructed models. Figure 5 presents the HD and SNR values between the original and recovered models for various compression threshold values m. The results indicate that the HD gradually decreases as the compression threshold increases, while the SNR increases linearly with the compression threshold. This indicates that the quality of the recovered model improves as the compression threshold rises. When the compression threshold of $m = 5$, the HD $< 10^{-4}$ and the SNR >70 dB, indicating that the quality of the recovered model very high. These experimental results confirm that the proposed scheme is reversible.



(a)



(b)

Figure 5. Results of reversibility under different values of compression threshold m . (a) Hausdorff distance. (b) SNR.

3.3 Embedding Capacity Analysis

PEC and embedding rate were used to evaluate the embedding capacity of the proposed scheme. Figure 6 presents the PEC for different values of the compression threshold m for each model. When $3 \leq m \leq 4$, the binary representation length of the vertices is 16 bits. At $m = 4$, the basic embedding length was 2 bits, which is significantly lower than the EL_B of 6 bits at $m = 3$. This indicates that a higher compression threshold m reduces the correlation among vertices, resulting in a significant decrease in embedding capacity.

For $5 \leq m \leq 9$, the average pure embedding capacities are 64.56, 54.73, 44.47, 34.71, and 24.88 bits per vertex, respectively. The embedding capacity increases substantially as the binary representation length of the vertices increases from 16 bits (for $3 \leq m \leq 4$) to 32 bits. However, as the compression threshold m continues to increase, the basic embedding length of the vertices decreases, reducing the correlation between vertices and, consequently, lowering the embedding capacity. These results indicate that spatial clustering significantly enhances embedding capacity. Additionally, when $2 \leq m \leq 9$, the embedding rate for each input model reaches 100%, indicating that incorporating the basic embedding length of reference vertices into the embedding room effectively improves both the embedding rate and capacity.

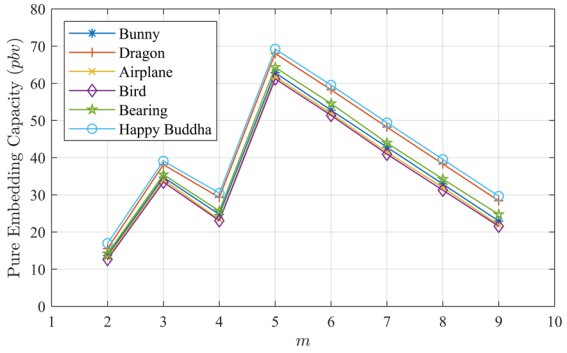


Figure 6. Pure embedding capacity under different values of compression threshold m.

3.4 Ablation Experiment

This section verifies the impact of two key modules on the performance of the embedding capacity: the clustering algorithm, and use of reference vertices for embedding data. We tested the two modules using the rabbit model to illustrate the impact of the choice of clustering algorithm and the use of reference points for embedded data on the performance, respectively.

3.4.1 Clustering Algorithms: Figure 7 illustrates the impact of different clustering algorithms on embedding capacity under different compression thresholds m. The algorithms compared include k-means, k-medoids (Park and Jun, 2009), density peak-based clustering (FSFDP) (Rodriguez and Laio, 2014), and spectral clustering (Ng et al., 2001). The k-means and k-medoids algorithms iteratively calculate distances between vertices and centroids (medoids) to partition the vertices of the input model into k mutually exclusive subsets or clusters, aiming to minimize the sum of distances between each vertex and its cluster center. Both methods achieved the highest PEC due to their effectiveness in creating tightly knit subspaces. The FSFDP and Spectral clustering algorithm ensure that vertices within each subspace are relatively compact, which enhances embedding capacity. However, their performance was consistently lower than that of k-means and k-medoids. The superior performance of k-means and k-medoids is attributed to their ability to effectively minimize the sum of distances between centroids and member vertices within each cluster, aligning well with the goal of creating compact subspaces that maximize embedding capacity.

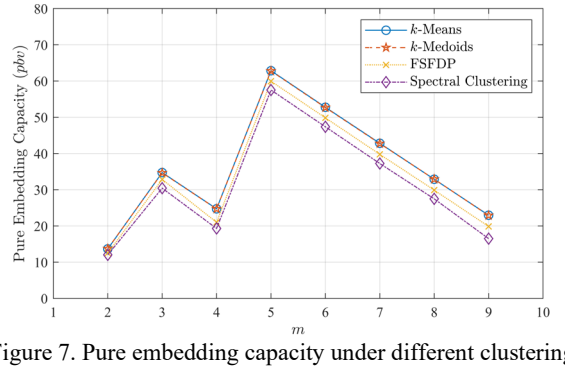


Figure 7. Pure embedding capacity under different clustering algorithms.

3.4.2 Basic Embedding Length of Reference vertices as Embedding Room: Figure 8 shows the impact of including the basic embedding length of reference vertices as part of the embedding room on embedding capacity under different compression thresholds m. For $5 \leq m \leq 9$, with a constant number of clusters (and consequently a constant number of reference vertices), the basic embedding length of vertice decreases as the compression threshold m increases. This reduction in the embedding space of reference vertices leads to a decrease in overall embedding capacity when the basic embedding length of reference vertices is used as part of the embedding room. At $m = 5$, incorporating the basic embedding length of the reference vertices into the embedding room increases the embedding capacity by up to 3 bits per vertex compared to not embedding messages in reference vertices, representing a 5% improvement in performance.

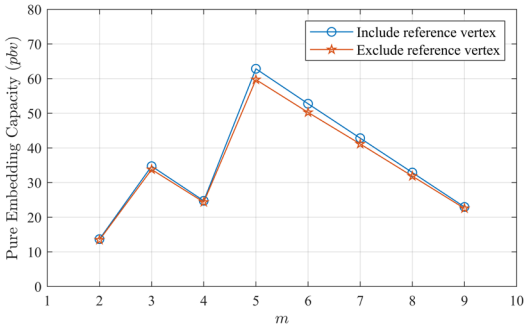


Figure 8. Pure embedding capability obtained when embedding space contains reference vertices or not.

3.5 Performance Comparison

The performance of the proposed method was compared with seven state-of-the-art methods (Hou et al., 2024, Tsai, 2021, Van Rensburg et al., 2021, Yin et al., 2021, Lyu et al., 2022, Tang et al., 2023, Tsai and Liu, 2023) in terms of embedding capacity and overall performance.

3.5.1 Embedding Capacity: Figure 9 shows the maximum average embedding capacity on the PSB dataset for the proposed method compared to existing algorithms. The work in (Tsai, 2021) integrated spatial subdivision and space encoding to embed additional data into encrypted vertices, achieving a PEC of only 7.68 bits per vertex. A two-layer RDH-ED method based on homomorphic encryption is proposed in (Van Rensburg et al., 2021), which resulted in an embedding capacity of 13.5 bits per vertex. This algorithm reached an average embedding capacity of 15.58 bits per vertex in (Yin et al., 2021). The work in (Lyu et al., 2022) utilized multi-MSB prediction and the odd-even property of vertex indices, this method achieved an embedding capacity of 26.11 bits per vertex. The work in (Tang et al., 2023) extended previous work by leveraging the model's topology to increase the number of embeddable vertices, raising the embedding capacity to 35.95 bits per vertex. This method reached an embedding capacity of the algorithm proposed in (Tsai and Liu, 2023) was 39.12 bits per vertex. The work in (Hou et al., 2024) divided the 3D model into non-overlapping subblocks and used octree spatial subdivision to enhance prediction length, achieving an embedding capacity of 51.96 bits per vertex. The algorithm proposed in this study employs spatial clustering to partition vertices into mutually exclusive clusters, ensuring that vertices within each cluster are as close as possible to one another, thereby maximizing prediction length. The proposed method achieved an average embedding capacity of 61.66 bits per vertex, representing an improvement of approximately 19% over (Hou et al., 2024).

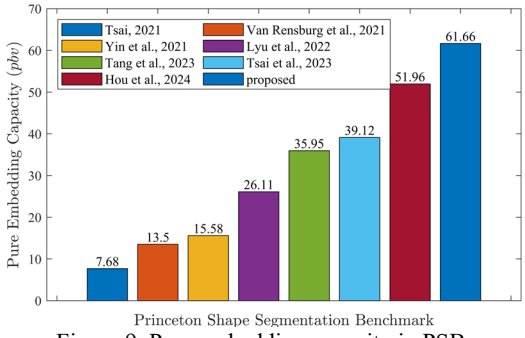


Figure 9. Pure embedding capacity in PSB.

3.5.2 Feature Comparison: Table 3 presents a comparison between the proposed algorithm and existing algorithms. The work in (Tsai, 2021) implemented an RDH-ED algorithm based on the VRAE framework by integrating spatial subdivision and encoding to embed additional data into encrypted vertices, excluding boundary vertices. While the embedding rate is approximately 100%, the embedding capacity is limited to approximately 7.68 bits per vertex. Improper threshold settings can lead to numerous message-extraction errors. A two-layer RDH-ED method based on homomorphic encryption is proposed in (Van Rensburg et al., 2021). However, the use of homomorphic encryption results in high computational overhead, and the method cannot losslessly recover the original model. The work in (Yin et al., 2021) proposed an RHD-ED algorithm based on BBRE framework that pre-calculates the multi-MSB prediction result of embeddable vertices using their neighboring vertices. However, because all neighboring vertices are used as references, the embedding rate is low, which restricts the method's efficiency in terms of embedding capacity. The work in (Lyu et al., 2022) divide all vertices into embeddable and reference vertices using the odd-even property of vertex indices to achieve a higher embedding rate. However, only half of the vertices can be used to embed messages, resulting in relatively low embedding capacity. The work in (Tang et al., 2023) extended the algorithm in (Lyu et al., 2022) by leveraging topological relationships to further exploit the spatial relationships between adjacent vertices. The work in (Tsai and Liu, 2023) performed random sampling to reduce the number of reference vertices and minimize the distance between the reference and its corresponding embedded vertices, leading to substantial increases in both total embedding capacity and embedding rate.

The aforementioned schemes (Yin et al., 2021, Lyu et al., 2022, Tang et al., 2023, Tsai and Liu, 2023) overlook both global and local spatial correlations, thereby limiting embedding capacity. The work in (Hou et al., 2024) utilized an octree spatial adaptive subdivision strategy to partition a 3D model into non-overlapping subblocks. While this method significantly improves embedding rate and capacity by leveraging local spatial correlations, it does not fully utilize global spatial correlation or use the basic embedding length of reference vertices as embedding room, resulting in a loss of potential embedding capacity and rate.

The proposed algorithm addresses these limitations by performing spatial clustering based on global spatial correlation to partition vertices into clusters, thereby enhancing the embedding capacity significantly. By using the basic embedding length of reference vertices into the embedding room, the proposed method further increases the model’s embedding capacity, achieving an embedding rate of 100%. The proposed method is both reversible and separable. These results confirm the feasibility and effectiveness of the proposed algorithm.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-G-2025-567-2025 | © Author(s) 2025. CC BY 4.0 License.

Algorithm	Tsai, 2021	Rensburg et al., 2021	Yin et al., 2021	Lyu et al., 2022	Tang et al., 2023	Tsai and Liu, 2023	Hou et al., 2024	Proposed
Type	VRAE	RRBE	RRBE	RRBE	RRBE	RRBE	RRBE	RRBE
Embedding Method	Spatial Subdivision Space Encoding	Paillier Additive Homomorphism	Multi-MSB Prediction Bit Substitution					
			Ring prediction	Odd-even property	Odd-even property/ Topology	Random sample	Octree subdivision	Spatial clustering
Embedding Rate	≈ 100%	100%	27.48%	50%	73%	27.48%~ 62.74%	95.49%	100%
Embedding Capacity	7.68	13.5	1.78-12.48	17.84-25.65	22.53-38.95	13.49-27.36	16.56-55.02	14.38-64.56
Separability	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
Extraction Errors	Yes	No	No	No	No	No	No	No
Model loss	Yes	Yes	No	No	No	No	No	No

Table 3. Comparison between proposed algorithm and other algorithms.

4. Conclusion

In this paper, a reversible data-hiding algorithm for encrypted 3D models was first proposed by integrating spatial clustering with multi-MSB prediction. The adaptive spatial subdivision method based on spatial clustering was proposed to divide the model's vertices into distinct and compact clusters. The optimal reference vertex search method explored local spatial correlations and measured the embedding lengths of the embeddable vertices within each cluster to maximize capacity. A cluster-index-based grouping strategy was proposed to remesh the 3D model, reducing transmission overhead. The basic embedding length of the reference vertex was employed to further enhance capacity. Multi-MSB prediction and Huffman coding complete the structure of the proposed algorithm. Experimental results demonstrated the algorithm's feasibility, achieved a high embedding rate and capacity, and supported reversibility and separability. Our proposed approach provides a reversible, high-capacity and scalable solution for secure data embedding in encrypted 3D graphics, making it highly efficient and suitable for practical applications. Future work will incorporate secret sharing to improve the security of the input model.

References

Chen, X., Golovinskiy, A., Funkhouser T., 2009. A benchmark for 3D mesh segmentation. *ACM Trans. Graph.*, 28(3), 1-12, doi: 10.1145/1531326.1531379

Hou, G., Ou, B., Long, M., Peng, F., 2024. Separable Reversible Data Hiding for Encrypted 3D Mesh Models Based on Octree Subdivision and Multi-MSB Prediction. *IEEE Transactions on Multimedia*, 26, 2395-2407. doi: 10.1109/TMM.2023.3295578.

Jiang, R., Zhou, H., Zhang W., Yu, N., 2018. Reversible data hiding in encrypted three-dimensional mesh models. *IEEE Trans. Multimedia*, 20(1), 55-67. doi: 10.1109/TMM.2017.2723244.

Lyu, W.-L., Cheng, L., Yin, Z., 2022. High-capacity reversible data hiding in encrypted 3D mesh models based on multi-MSB prediction. *Signal Process.*, 201. doi: 10.1016/j.sigpro.2022.108686.

Ng, A., 2001. On spectral clustering: Analysis and an algorithm. in *Proc. 14th Advances in Neural Information Processing Systems*, 849-856.

Park, H. S., Jun, C. H., 2009. A simple and fast algorithm for K-medoids clustering. *Expert Syst. Appl.*, 36(2), 3336-3341. doi: 10.1016/j.eswa.2008.01.039.

Qiu, Y., Ying, Q., Yang, Y., Zeng, H., Li, S., Qian, Z., 2022. High-capacity framework for reversible data hiding in encrypted image using pixel prediction and entropy encoding. *IEEE Trans. Circuits Syst. Video Technol.*, 32(9), 5874-5887. doi: 10.1109/TCSVT.2022.3163905.

Rensburg, B. J. v., Puteaux, P., Puech, W., Pedeboy, J. -P., 2021: Homomorphic two-tier reversible data hiding in encrypted 3D objects. *Proc. IEEE International Conference Image Process*, 3068-3072. doi: 10.1109/ICIP42928.2021.9506320.

Rodriguez, A., Laio, A., 2014. Machine learning. Clustering by fast search and find of density peaks. *Science*, 344(6191), 1492-1496. doi: 10.1126/science.1242072.

Shah, M., Zhang, W., Hu, H., Zhou, H., Mahmood, T., 2018. Homomorphic encryption-based reversible data hiding for 3D mesh models. *Arab. J. Sci. Eng.*, 43(12), 8145-8157. doi: 10.1007/s13369-018-3354-4.

Tang, Y., Cheng, L., Lyu, W., Yin, Z., 2023. High capacity reversible data hiding for encrypted 3D mesh models based on topology. *Proc. 21st Int. Workshop Digit. Forensics Watermarking*, 205-218. doi: 10.1007/978-3-031-25115-3_14.

Tsai, Y. -Y., 2021. Separable reversible data hiding for encrypted three-dimensional models based on spatial subdivision and space encoding. *IEEE Trans. Multimedia*, 23, 55-67. doi: 10.1109/TMM.2017.2723244.

Tsai, Y.-Y., Liu, H.-L., 2023. Integrating coordinate transformation and random sampling into high-capacity reversible data hiding in encrypted polygonal models. *IEEE Trans. Depend. Sec. Comput.*, 20(4), 3508-3519. doi: 10.1109/TDSC.2022.3204291.

Xu, N., Tang, J., Luo, B., Yin, Z., 2022. Separable reversible data hiding based on integer mapping and MSB prediction for encrypted 3D mesh models. *Cognit. Comput.*, 14(3), 1172-1181. doi: 10.1007/s12559-021-09919-5.

Yin, Z., Xiang Y., Zhang X., 2022. Reversible data hiding in encrypted images based on multi-MSB prediction and Huffman coding. *IEEE Trans. Multimedia*, 22(4), 874-884. doi: 10.1109/TMM.2019.2936314.

Yin, Z., Xu, N., Wang, F., Cheng L., Luo, B., 2021. Separable reversible data hiding based on integer mapping and multi-MSB prediction for encrypted 3D mesh models. *Proc. Chin. Conf. Pattern Recognit. Comput. Vis.*, 336-348. doi: 10.1007/978-3-030-88007-1_28.