Rethinking Mesh Watermark: Towards Highly Robust and Adaptable Deep 3D Mesh Watermarking

Xingyu Zhu[1,2,3], **Guanhui Ye**[2], **Xiapu Luo**[3], **Xuetao Wei**[2,1*]
[1]Research Institute of Trustworthy Autonomous Systems,
Southern University of Science and Technology, Shenzhen 518055, China
[2]Department of Computer Science and Engineering, Southern University of Science and Technology, China
[3]Department of Computing, Hong Kong Polytechnic University, Hong Kong
12150086@mail.sustech.edu.cn, 12132370@mail.sustech.edu.cn, csxluo@comp.polyu.edu.hk, weixt@sustech.edu.cn

Abstract

The goal of 3D mesh watermarking is to embed the message in 3D meshes that can withstand various attacks imperceptibly and reconstruct the message accurately from watermarked meshes. The watermarking algorithm is supposed to withstand multiple attacks, and the complexity should not grow significantly with the mesh size. Unfortunately, previous methods are less robust against attacks and lack of adaptability. In this paper, we propose a robust and adaptable deep 3D mesh watermarking DEEP3DMARK that leverages attention-based convolutions in watermarking tasks to embed binary messages in vertex distributions without texture assistance. Furthermore, our DEEP3DMARK exploits the property that simplified meshes inherit similar relations from the original ones, where the relation is the offset vector directed from one vertex to its neighbor. By doing so, our method can be trained on simplified meshes but remains effective on large size meshes (size adaptable) and unseen categories of meshes (geometry adaptable). Extensive experiments demonstrate our method remains efficient and effective even if the mesh size is 190× increased. Under mesh attacks, DEEP3DMARK achieves 10%∼50% higher accuracy than traditional methods, and 2× higher SNR and 8% higher accuracy than previous DNN-based methods.

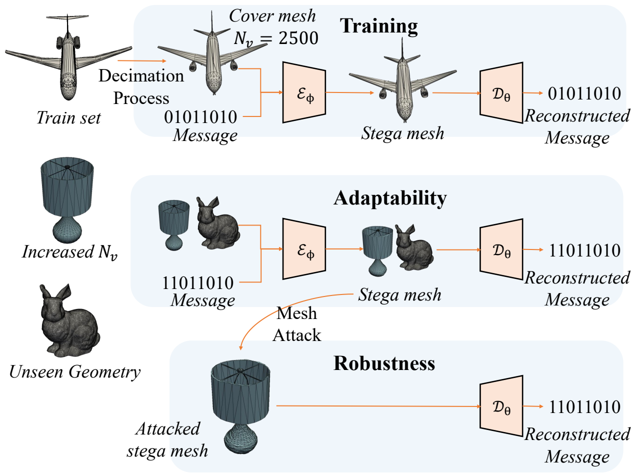


Figure 1: We train our DEEP3DMARK on simplified meshes (top) and test on varied mesh sizes and unseen geometry (middle) to show adaptation. We further test DEEP3DMARK on multiple mesh attacks (bottom).

Introduction

Digital watermarking is a technology used in copyright protection of multimedia, such as images, videos, point clouds, and meshes. The goal of digital watermarking is to obtain watermarked media by embedding messages in the media in the embedding phase and reconstructing the message from the watermarked media in the reconstruction phase. However, previous 3D mesh watermarking methods pursue high capacity while ignoring robustness. The watermark should be *imperceptible* and *robust* so that it can withstand attacks and be *adaptable* so that it can be applied to arbitrary mesh sizes and geometries.

Previous 3D mesh watermarking methods can be classified into DNN-based and traditional methods. Traditional methods focus on improving the capacity of watermarking (*i.e.*, the number of embedded bits per vertices) while ignoring the robustness of their watermark. For example, some methods (Peng, Long, and Long 2021; Tsai 2020) embed secret messages in the least significant bits (LSBs) of vertex coordinates, but they are vulnerable to Gaussian noises. Others (Tsai and Liu 2022; Hou et al. 2023) embed secret messages in the most significant bits (MSBs) of vertex coordinates, which makes them robust against noises. However, they still cannot withstand rotation and scaling. Recent DNN-based methods show the possibility of embedding watermarks in either vertex domain (Wang et al. 2022) or texture domain (Yoo et al. 2022). However, these methods are either only able to extract messages with textured meshes(Yoo et al. 2022) or low in watermarked mesh quality (Wang et al. 2022). Moreover, watermark quality and embedding overhead should be less impacted by variations in mesh sizes and geometries for practical application consideration. Previous work has yet to explore a robust and adaptable watermarking method.

In this paper, we propose a highly robust and adaptable deep 3D watermarking DEEP3DMARK. Compared to traditional methods, DEEP3DMARK is more robust against multiple mesh attacks even without prior knowledge of the

*Corresponding Author.

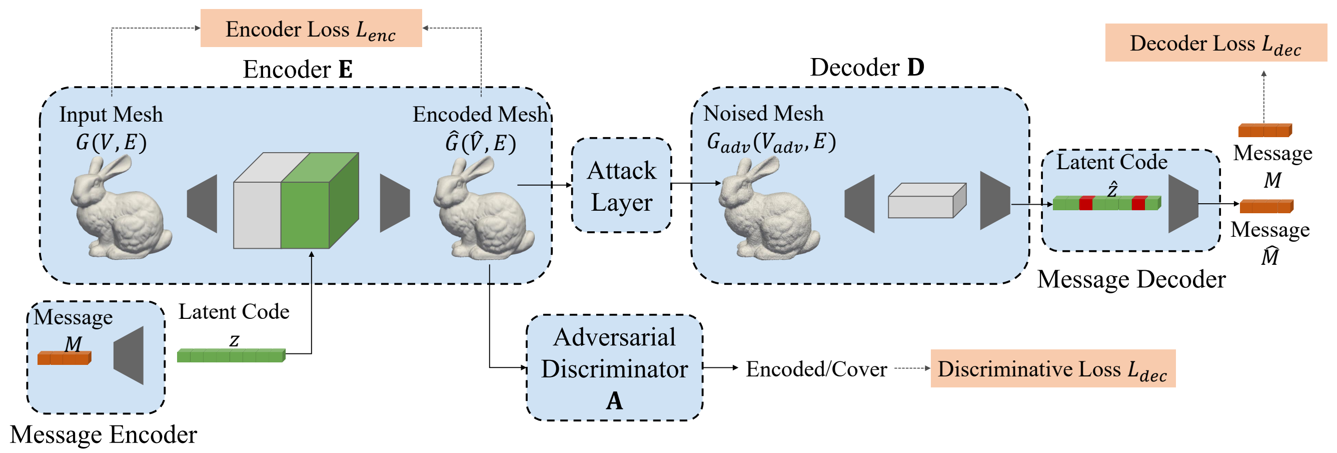


Figure 2: Deep3DMark overview. Message encoder first maps message M into latent code z, which is further fed to the watermark encoder along with the input mesh G to generate the encoded mesh $\hat{G}$. The attack layer generates a noised mesh G_{adv}. Given the noised mesh, the watermark decoder produces the decoded latent code $\hat{z}$ followed by the message decoder, which decodes latent code into decoded message $\hat{M}$. The adversarial discriminator encourages minimizing the difference between G and $\hat{G}$.

type of attacks. Compared to previous DNN-based methods, Deep3DMark generates higher-quality watermarked meshes and can be applied to different mesh sizes and unseen geometries. We watermark the vertex coordinates of meshes by adopting graph-attention network (GAT) (Veličković et al. 2017). To achieve robustness, we apply adversarial training with GAT-generated perturbations. To achieve adaptability, we provide an insight that different sizes of meshes under the same categories share similar features. By exploiting such similarities, we can train our Deep3DMark with simplified meshes and keep effective on meshes with increased sizes.

Specifically, our adopted GAT is the backbone of our Deep3DMark, which can generate watermarked meshes given the original meshes and binary messages and reconstruct the binary messages from the watermarked meshes. Our Deep3DMark consists of 1) an encoder, 2) a decoder, 3) a message autoencoder, 4) an attack layer, and 5) a discriminator, as shown in Fig. 2. We train our Deep3DMark using simplified data from the train set under all scenarios to better evaluate the adaptability. To prove effectiveness, our Deep3DMark is tested on complete test data. To prove robustness, we test Deep3DMark on multiple mesh attacks. To prove adaptability, we test Deep3DMark on multiple datasets and different sizes of meshes. In summary, our contributions are the following:

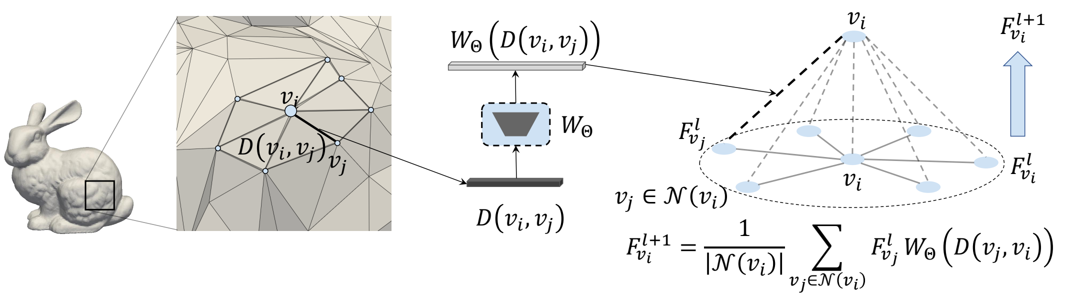
- We investigate mainstream watermarking methods and observe their low robustness. To tackle this problem, we propose a highly robust deep 3D watermarking Deep3DMark, which embeds binary messages in vertex distributions by incorporating the graph attention network (GAT) and achieves robustness against unknown mesh attacks by incorporating adversarial training.
- We achieve adaptability by exploiting the property that simplified meshes inherit similar relations from the original meshes, where the relation is an offset vector directed from a vertex to its neighbor. Our Deep3DMark can be trained on simplified meshes but remains effective on large-sized meshes and unseen categories of meshes.
- We conduct extensive experiments on various datasets to prove Deep3DMark's effectiveness, robustness, and adaptability. Our Deep3DMark achieves 10%∼50% higher accuracy when facing attacks compared to traditional methods and achieves 50% lower distortions and 8% higher accuracy compared to previous DNN-based methods. Our experiment shows that our method can also be robust against multiple unknown attacks.

Related Work

Mesh Watermarking

Early 3D mesh watermarking methods (Son et al. 2017; Al-Khafaji and Abhayaratne 2019) used Fourier and wavelet analysis to transfer meshes into the frequency domain and embed watermark bits into Fourier/wavelet coefficients. However, the time complexity of these methods grows cubically with the number of vertices. (Zhou et al. 2018; Jiang et al. 2017; Tsai and Liu 2022; Hou et al. 2023) proposed to embed watermarks into the least significant bits and the most significant bits of vertex coordinates. (Hou, Kim, and Lee 2017) leveraged the layering artifacts of 3D printed meshes to apply watermark embedding and reconstruction.

Recently, (Yoo et al. 2022) and (Wang et al. 2022) explored the feasibility of DNN in watermarking work. (Wang et al. 2022) stacked graph residual blocks to embed and extract the watermark. (Yoo et al. 2022) embedded secret messages in textures of meshes and then extracted the message from the rendered 2D image, but cannot reconstruct an accurate message without the help of a texture encoder. In this



$W_\Theta\left(D(v_i, v_j)\right)$
W_Θ
$D(v_i, v_j)$
v_i
v_j
$F_{v_i}^{l+1}$
$F_{v_j}^{l}$
$F_{v_i}^{l}$
$v_j \in \mathcal{N}(v_i)$
$F_{v_i}^{l+1} = \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} F_{v_j}^{l} W_\Theta\left(D(v_j, v_i)\right)$

Figure 3: The process of generating the new feature $F_{v_i}^{l+1}$ of v_i. Left: the local region centered at v_i. Middle: within neighborhood $v_j \in \mathcal{N}(v_i)$, an MLP generates weights between v_i and v_j given their relation $D(v_i, v_j)$. Right: given vertex feature $\mathbf{F}^l = \{F_{v_0}^l, F_{v_2}^l, ..., F_{v_N}^l\}$ and generated weights of neighborhood $v_j \in \mathcal{N}(v_i)$, compute the new feature vector $F_{v_i}^{l+1}$ as the weighted sum of neighbor features.

case, replacing the texture image can completely remove the watermarks. Moreover, recent research (Dong, Kumar, and Liu 2022) also shows how to detect DNN-generated images. Hence, watermarking in mesh geometries is more secure than watermarking in textures.

Neural Networks for 3D Meshes

Existing methods for 3D data built features from faces (Xu, Dong, and Zhong 2017; Feng et al. 2019; Lian et al. 2019; Hertz et al. 2020; Hu et al. 2022; Kim and Chae 2022), edges (Simonovsky and Komodakis 2017; Veličković et al. 2017; Wang et al. 2019) and vertices (Qi et al. 2017a,b; Wu, Qi, and Fuxin 2019; Liu et al. 2019; Xu et al. 2018; Hermosilla et al. 2018; Groh, Wieschollek, and Lensch 2018). The built features were applied to downstream tasks such as classification and segmentation. (Veličković et al. 2017; Simonovsky and Komodakis 2017) introduced an attention-based mechanism into graph convolution, where the weights of each neighbor were adjusted based on the edge information. Such graph-based convolution can be further extended to 3D meshes.

Definition

Triangle meshes can be viewed as undirected graphs $G(V, E)$. Vertices $V \in \mathbb{R}^{N_v \times C_v}$ contains N_v vertices, and each vertex has C_v vertex elements such as coordinates and normals. Edges E can be transformed from faces set of triangle meshes, where each face is a triangle formed by three vertex indices. Since changes of E produce unexpected artifacts, we embed a binary message $M \in \{0,1\}^{N_m}$ into the vertex distribution V, i.e., we embed binary messages in vertex distributions V. Let $V, \hat{V}, M, \hat{M}$ denote the original vertex, watermarked vertex, binary messages, and reconstructed messages, respectively. We model the problem by the following equations:

$$\begin{aligned}\hat{V} &= \mathcal{E}_\phi(V, M)\\ \hat{M} &= \mathcal{D}_\theta(\hat{V})\end{aligned} \tag{1}$$

In Eq 1, a parameterized encoding function $\mathcal{E}_\phi$ generates watermarked vertices $\hat{V}$ given original vertices V and a binary message M. A parameterized decoding function $\mathcal{D}_\theta$ reconstructs $\hat{M}$ from $\hat{V}$. The encoding function should minimize the perturbation between V and $\hat{V}$ by minimizing the following loss to achieve imperceptible embedding:

$$L_{enc}(\phi, \theta) = \mathbb{E}_{V,M}[\|\hat{V} - V\|_2^2] \tag{2}$$

To achieve precise reconstruction, we try to minimize the following loss:

$$L_{dec}(\phi, \theta) = \mathbb{E}_{V,M}[\|\hat{M} - M\|_2^2] \tag{3}$$

Finally, we have combined the optimization problem:

$$\phi^*, \theta^* = \arg\min_{\phi,\theta}(L_{enc}(\phi, \theta) + L_{dec}(\phi, \theta)) \tag{4}$$

Method

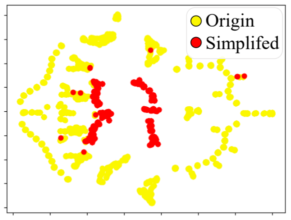
We propose DEEP3DMARK, an end-to-end imperceptible watermarking method that can be robust to arbitrary attacks and be adaptable to different mesh sizes and geometries. To watermark a graph signal $G(V, E)$, we utilize local features in the spatial domain using graph attention network (GAT), which is the backbone of our DEEP3DMARK. We first introduce GAT on mesh. Then we introduce all DEEP3DMARK modules, followed by a detailed introduction to our training details.

Graph Attention Network on Mesh

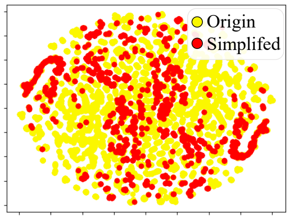
Graph attention network (GAT) is a convolution operator defined on graphs. For l-th layer of GAT, the input is a set of vertex features $\mathbf{F}^l = \{F_{v_0}^l, F_{v_2}^l, ..., F_{v_N}^l\}$, where N is the number of vertices. This layer produces a new set of vertex features $\mathbf{F}^{l+1} = \{F_{v_0}^{l+1}, F_{v_2}^{l+1}, ..., F_{v_N}^{l+1}\}$. For each vertex v_i, its new feature $F_{v_i}^{l+1}$ is computed as the averaged weighted sum of its neighbor features $F_{v_j}^l$ for all $v_j \in \mathcal{N}(v_i)$. To increase expressive power, weights for each neighbor v_j are obtained from learnable linear transform W_Θ.

We view 3D meshes as graphs $G(V, E)$. We first define our GAT on meshes as:

$$F_{v_i}^{l+1} = \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} F_{v_j}^l W_\Theta(D(v_j, v_i)) \tag{5}$$



(a) Coordinates Distribution



(b) Relation Distribution

Figure 4: t-SNE (Van der Maaten and Hinton 2008) visualization of the distribution between original and simplified meshes. (a) shows the existence of distribution shifts between the original and decimated coordinates. However, (b) shows that the distributions of decimated relations $D(v_i, v_j)$ are included in the distributions of the original ones.

where $F^{l+1}(v_i)$ is the feature vector of v_i at $(l+1)$-th layer. The neighborhood $\mathcal{N}(v_i) = \{v_j|(v_j, v_i) \in E\} \cup \{v_i\}$ is defined as all points adjacent to the point v_i and itself. We use a multilayer perceptron (MLP) to model the learnable linear transform W_Θ. The input of the MLP is the relation $D(v_i, v_j)$ between the target vertex v_i and its neighbor v_j. Figure 3 gives a visualization process of our GAT.

We show that it is beneficial to learn from relation $D(v_i, v_j)$. We define relation $D(v_i, v_j) = \vec{v_i} - \vec{v_j}$ as the coordinate offset from v_i to v_j. Such the relation can survive the mesh simplification algorithm (Lindstrom et al. 1998, 1999). First, we simplify meshes to reduce the vertex number to $1/5$ of the original, *i.e.* $N'_v = 1/5N_v$. Then we visualize the coordinates and relation distributions for both original and simplified meshes in Figure 4. Figure 4 shows coordinate distribution differences between original and simplified meshes, while relation distributions are still included in the original distributions. Based on this insight, our method is trained on simplified meshes and shows adaptability to increased-size meshes.

Deep3DMark

Our architecture (Figure 2) consists of five parameterized learnable components: 1) a message autoencoder that can map a binary message M to a latent code z and decodes z back to M, 2) an encoder $\mathbf{E}$ models function $\mathcal{E}_\phi$ that generates a watermarked vertices $\hat{V}$ given V and z, 3) an attack layer applies perturbation over $\hat{V}$ to increase the robustness in the way of data augmentation, 4) a decoder $\mathbf{D}$ models $\mathcal{D}_\theta$ that reconstructs binary message from $\hat{v}$, and 5) a discriminator $\mathbf{A}$ encourages $\hat{V}$ indistinguishable from V.

The **encoder E** first applies convolutions to input V to form some intermediate representation. We aim to incorporate message latent code z in the way that the encoder learns to embed parts of it at any spatial location of V. To achieve this, we replicate the latent code and concatenate it to the intermediate representations. We apply more convolutions to transform the concatenated feature to watermarked vertices $\hat{V}$.

The **attack layer** applies perturbations to generated $\hat{V}$. The perturbations consider several mesh attacks, including 1) Gaussian noise with mean μ and deviation σ, 2) random rotation attacks with rotate center (x, y, z) and degree α, 3) translation attack and 4) scaling attack with a scaling ratio s. Our ablation study shows that the attack layer effectively increases the robustness against multiple attacks.

The **decoder D** first applies several convolutions to generate the intermediate representation of $\hat{V}$. It finally uses a global average pooling followed by an MLP layer to generate a vector of the same size as the latent code z. The global average pooling layer ensures that our method aggregates information from all vertices.

The **adversarial discriminator A** shares a similar structure as the decoder except that its final MLP layer transforms the aggregated vector into a binary classification, which indicates whether the given $\hat{V}$ is generated by the encoder $\mathbf{E}$.

According to Shannon's capacity theory (Shannon 1948), redundancy is necessary to achieve robustness. The **message autoencoder** increases the robustness of our system by injecting redundancy into the system. Given a binary message M of length N_m, the message encoder maps it into a latent code z of length $N_z > N_m$, which can be used to recover M through a message decoder. We train the autoencoder in a way that the decoder can recover M from the noised latent code $\hat{z}$. We choose NECST (Choi et al. 2019), a learnable channel coding method, as our message autoencoder. Our message autoencoder is trained independently from the entire watermarking model.

Training and Losses

We achieve the objective in Eq 4 using three losses: encoding loss L_{enc}, reconstruction loss L_{dec}, and discriminative loss L_{dis}. Formally:

$$\phi^*, \theta^* = \arg\min_{\phi,\theta}(\lambda_{enc}L_{enc}(\phi,\theta) + \lambda_{dec}L_{dec}(\phi,\theta) + \lambda_{dis}L_{dis}(\phi,\theta)) \tag{6}$$

where $\lambda_{enc}, \lambda_{dec}, \lambda_{dis}$ are weight factors. Both L_{enc}, L_{dis} encourage generated $\hat{V}$ indistinguishable from V. For L_{enc}, we use both the L2 norm and infinite norm of geometry difference to penalize the distortion:

$$L_{enc} = \frac{1}{N_v}\sum_i^{N_v}(V[i] - \hat{V}[i])^2 + \max_i\{V[i] - \hat{V}[i]\} \tag{7}$$

For L_{dis}, we use part of sigmoid cross entropy loss:

$$L_{dis} = \log(1 - \sigma(\mathbf{A}(\hat{V}))) \tag{8}$$

We apply standard sigmoid cross entropy loss to encourage precise message reconstruction:

$$L_{dec} = \frac{\sum_i^{N_m}(M[i] \cdot \log\sigma(\hat{M}[i]) + (1 - M[i]) \cdot \log(1 - \sigma(\hat{M}[i])))}{N_m} \tag{9}$$

The final message bits are computed from the following:

$$M_{final} = clamp(sign(\hat{M} - 0.5), 0, 1) \tag{10}$$

Algorithm	Geometry Difference			Accuracy & Robustness (%)						Time (s)
	L_1d	Hausdorff	SNR	w/o attack	Gauss	Trans	Rot	Scale	Crop	
JIANG2017	0.1051	1.3950	39.50	80.59	76.98	48.2	80.58	49.97	49.68	81.24
SPECTRAL	0.0090	0.0125	44.99	87.48	54.31	50.41	50.40	54.29	56.01	70.350
PENG2021	0.0182	0.0158	46.70	97.98	50.24	58.30	50.55	50.25	50.30	30.470
PENG2022	0.0073	0.0375	36.94	80.89	79.28	80.89	80.89	80.89	54.12	3.240
TSAI2020	0.0101	0.0145	40.90	96.41	50.03	49.99	49.75	50.02	50.10	5.989
TSAI2022	**0**	**0**	**inf**	84.49	84.48	84.48	64.32	61.76	50.51	3.079
HOU2023	0.0194	0.0268	45.10	90.84	90.84	69.22	87.92	49.55	50.21	2.090
WANG2022	0.1337	0.1856	15.04	97.50	90.84	90.22	90.92	90.55	70.21	0.010
DEEP3DMARK **(Ours)**	0.0338	0.0617	30.84	**98.17**	**91.06**	**98.17**	**96.84**	**98.17**	**78.90**	**0.0089**

Table 1: Comparison of watermarked mesh quality (geometry difference), reconstruction accuracy without attack (w/o attack), robustness under Gaussian noise (Gauss), translation (Trans), rotation (Rot), scaling (Scale), and cropping attack (Crop).

Experiment

Experiment Setup

Training Settings. Our experiment is conducted on Ubuntu 18.04, with 503GB RAM and five Nvidia RTX 3090. Our DEEP3DMARK uses GATs to build **E**, **D** and **A**, where channel size are all 64. At the first layer, we take coordinates (x, y, z) as the feature of points, i.e., $C_v = 3$. Our experiment for both DEEP3DMARK and other baselines are evaluated under the message length $N_m = 8$. During training, we set $\lambda_{enc} = 2, \lambda_{dec} = 1, \lambda_{dis} = 0.001$ under the settings of 8-bit message lengths, and we set $\mu = 0$, $\sigma = 0.001$, $\alpha \in [0, \pi)$, $s \in [0.1, 1)$.

Baselines. We adopt nine state-of-the-art watermarking algorithms as our baselines. JIANG2017 (Jiang et al. 2017), PENG2022 (Peng, Liao, and Long 2022), TSAI2020 (Tsai 2020), TSAI2022 (Tsai and Liu 2022) and HOU2023 (Hou et al. 2023) are encrypted domain watermarking algorithms, whose geometry difference can only be evaluated after model decryption. We evaluate the geometry difference between the original and decrypted watermarked mesh. PENG2021 (Peng, Long, and Long 2021), SPECTRAL (Al-Khafaji and Abhayaratne 2019) and WANG2022 (Wang et al. 2022) are plain-text domain watermarking algorithms whose geometry difference can be directly evaluated between the original and watermarked mesh. We also compare with YOO2022 (Yoo et al. 2022) in geometry difference and accuracy.

Metrics. To evaluate the extracted message accuracy, we use average bit accuracy. To evaluate geometry differences, we use Hausdorff distance, the L1 norm of vertex difference (L_1d), and signal-to-noise ratio (SNR).

Dataset

Our DEEP3DMARK is trained on a simplified train set from ModelNet40 (Wu et al. 2015) and then tested on the entire test of ModelNet40 and other datasets such as ShapeNet (Chang et al. 2015), GraspNet (Fang et al. 2020), ScanNet (Dai et al. 2017) and Hands (Romero, Tzionas, and Black 2022). For all datasets, we normalize the vertex coordinates (x, y, z) to $[-1, 1]$ before meshes are fed into the network unless explicitly mentioned.

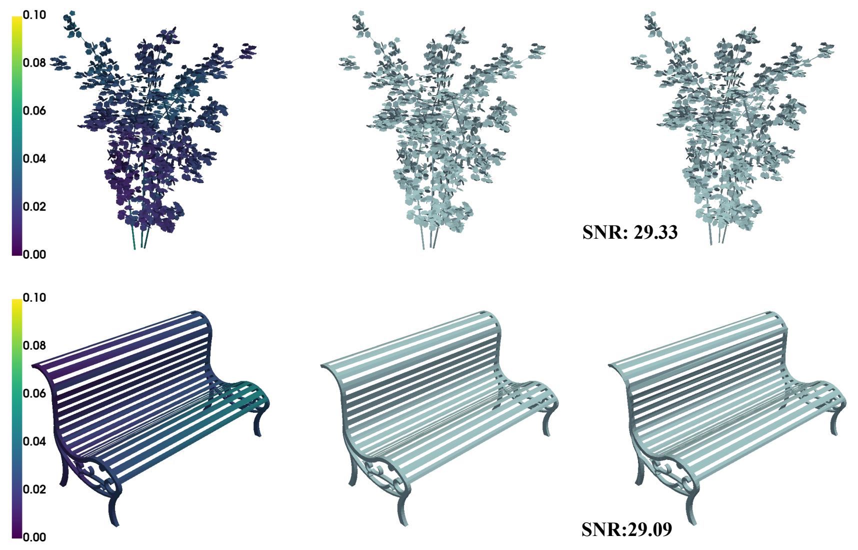
We acquire simplified data through a simplification using CGAL (The CGAL Project 2022), which performs edge-collapse or half-edge-collapse algorithms to reduce the number of triangles by merging vertices. We generated two train sets *m500* and *m2500*. The number of vertices in *m500* and *m2500* are $N_v = 500$ and $N_v = 2500$, respectively. For *m2500*, we manually filter out meshes whose N_v is originally less than 2500 and those with low quality after simplifications. We also perform the same process for *m500*. As a result, we get 3508 train meshes and 879 test meshes for *m2500*, and 1147 train meshes and 337 test meshes for *m500*. The original ModelNet has 9843 and 2468 meshes for training and testing. We train two replicas of DEEP3DMARK on *m500* and *m2500*, respectively. Both are further tested on the test set of ModelNet to evaluate the size adaptability.

To evaluate the effectiveness on geometry variations, two replicas of DEEP3DMARK, which is trained on *m500* and *m2500*, are tested on ShapeNet, GraspNet, ScanNet, and Hands. ShapeNet has different categories of meshes from ModelNet, such as birdhouse, camera, clock, etc. Scannet is a dataset of scanned and reconstructed real-world scenes. Hands contain meshes of human hands.

Experiment on Robustness

Settings. Our first experiment is to (1) prove that it is hard to achieve full robustness even under the settings of 8-bit message length for traditional methods and (2) DEEP3DMARK is robust against Gaussian, rotation, scaling, translation, and cropping attack while maintaining relatively high quality. DEEP3DMARK is trained on the train set of *m2500*. DEEP3DMARK and baselines are further evaluated on the test set of *m2500*. For a fair comparison, the watermarked meshes are rescaled back to their original coordinates before we apply any attacks. We choose Gaussian noise ($\sigma = 0.1$), translation (random translation vector in $[0, 1000]^3$), rotation (origin point as the rotation center and $\alpha \in [0, \frac{\pi}{2}]$,), scaling ($s \in [0.1, 1)$) and cropping attacks (cropping ratio $c = 0.1$).

Results. Table 1 shows that although traditional methods have relatively high watermarked mesh quality, they are vulnerable to multiple attacks. PENG2021 embeds secret messages in the least significant bits of vertex coordinates, thus vulnerable to Gaussian attacks. TSAI2022 embeds secret messages in the most significant bits of vertex coordinates. However, it still cannot withstand rotation and scaling attacks. Compared to traditional methods, DEEP3DMARK



(a) Visualization $L_1 d$ (b) Original Mesh (c) Watermarked Mesh

Figure 5: Results to show the imperceptible watermarking: (a) L1 Norm of vertex difference ($L_1 d$), (b) the original mesh, and (c) the watermarked mesh with DEEP3DMARK (ours).

is robust against arbitrary attacks and achieved 10%∼50% higher accuracy when facing arbitrary attacks. Compared to DNN-based methods, DEEP3DMARK achieved 1%∼8% higher accuracy and 2× SNR value. Table 2 shows that we achieve similar geometry difference and accuracy to Yoo2022. We provide a visual quality example in Fig. 5 for the concern of the SNR drop compared with traditional methods.

Algorithm	L1 Vertex Normal	Bit Accuracy (%)
Yoo2022	**0.1041**	0.9362
DEEP3DMARK	0.1143	**0.9403**

Table 2: Comparison with Yoo2022.

Experiment on Unknown Distortions

Settings. Our second experiment is to prove DEEP3DMARK is robust against unknown distortions because a practical watermarking experiment must be robust against a wide range of distortions. We choose Gaussian noise, cropping, reordering, ARAP (Sorkine and Alexa 2007), implicit laplacian smoothing (Smooth) (Desbrun et al. 2023) and Draco (Google 2017) compression. For the ARAP attack, we randomly select $[0, 10]$ handle points and move all the handle points along a vector with length 0.1. For implicit laplacian smoothing, we increase the distortion strength by increasing the parameter λ_I. For Draco compression, we increase the distortion strength by decreasing the quantization bits N_q.

Results. Table 3 shows the bit accuracy of our model on these additional distortions. Overall, our model shows full robustness on these unknown distortions.

Distortions Made By	Bit Accuracy (%)
No Distortion	98.17
Gaussian Noise ($\sigma = 0.005$)	98.06
Gaussian Noise ($\sigma = 0.01$)	97.11
Gaussian Noise ($\sigma = 0.02$)	90.47
Cropping ($c = 0.1$)	78.41
Cropping ($c = 0.5$)	75.04
Cropping ($c = 0.9$)	65.81
Implicit Laplacian Smooth ($\lambda_I = 1.0$)	93.57
Implicit Laplacian Smooth ($\lambda_I = 5.0$)	89.25
Implicit Laplacian Smooth ($\lambda_I = 10$)	80.29
Draco Compression($N_q = 15$)	97.32
Draco Compression($N_q = 10$)	96.98
Draco Compression($N_q = 5$)	88.60
Reorder	98.17
ARAP	96.42

Table 3: The robustness under unknown attacks, where Gaussian noise is tested with out-of-domain parameters.

Experiment on Adaptability

Settings. Our third experiment is conducted to prove that our method can be generalized to different mesh sizes and unseen geometry. We train DEEP3DMARK on *m500* and *m2500* to get two replicas. Both are further evaluated on the original test set of ModelNet40, ShapeNet40, ScanNet, GraspNet, and Hands. The data distributions in ShapeNet40, ScanNet, GraspNet, and Hands are unseen to both DEEP3DMARK replicas during training.

Results. Figure 6 shows the result using the DEEP3DMARK trained on *m2500*. The top row shows the cover meshes where (a-d) are simplified from the original mesh (e). The bottom row shows the watermarked meshes. Table 5 shows

	Metrics	Avg	birdhouse	camera	clock	spigot	knife	loudspeaker	mug	pistol	printer
m500	Hausdorff	0.0758	0.0778	0.0723	0.0754	0.0762	0.0747	0.0690	0.0882	0.0809	0.0794
	$L_1 d$	0.0308	0.0349	0.0339	0.0476	0.0364	0.0355	0.0421	0.0313	0.0380	0.0333
	SNR	26.54	27.91	27.99	26.08	25.79	25.22	26.58	26.66	27.67	26.18
	Acc	0.8863	0.8698	0.9347	0.9220	0.7943	0.7160	0.9221	0.8615	0.8969	0.9390
m2500	Hausdorff	0.0549	0.0680	0.0568	0.0570	0.0474	0.0493	0.0620	0.0527	0.0562	0.0547
	$L_1 d$	0.0248	0.0245	0.0302	0.0278	0.0281	0.0267	0.0293	0.0266	0.0254	0.0296
	SNR	31.01	29.92	27.55	31.11	27.14	30.30	27.02	30.68	30.76	31.19
	Acc	0.9348	0.9554	0.9800	0.9489	0.9358	0.8773	0.9682	0.9526	0.9324	0.9623

Table 4: Geometry adaptability on ShapeNet dataset. *m500* and *m2500* are trained on simplified ModelNet dataset with $N_v = 500$ and $N_v = 2500$, respectively. Here, we list the results of nine categories.

	Metrics	$(0, 20000)$	$[20000, 40000)$	$[40000, 60000)$	$[60000, 80000)$	$[80000, 100000)$
m500	Hausdorff	0.0721	0.0609	0.0600	0.0626	0.0666
	$L_1 d$	0.0408	0.0344	0.0348	0.0355	0.0304
	SNR	26.65	27.43	26.61	26.552	26.91
	Acc	0.9183	0.9052	0.8570	0.8717	0.8181
m2500	Hausdorff	0.0550	0.0473	0.0466	0.0502	0.0488
	$L_1 d$	0.0281	0.0225	0.0232	0.0232	0.0222
	SNR	29.83	31.21	30.18	30.33	30.02
	Acc	0.9462	0.9034	0.9183	0.9123	0.8708

Table 5: Size adaptability on ModelNet dataset with the varied number of vertices $N_v \in (0, 100000]$. *m500* and *m2500* are trained on simplified ModelNet dataset with $N_v = 500$ and $N_v = 2500$, respectively.

	Metrics	GraspNet	Hands	ScanNet
m500	Hausdorff	0.0817	0.0753	0.0970
	$L_1 d$	0.0365	0.0337	0.0378
	SNR	28.80	28.91	27.07
	Acc	0.9588	0.9288	0.8593
m2500	Hausdorff	0.0519	0.0515	0.0527
	$L_1 d$	0.0254	0.0277	0.0237
	SNR	30.62	30.73	30.93
	Acc	0.9673	0.9584	0.9929

Table 6: Geometry adaptability on other datasets.

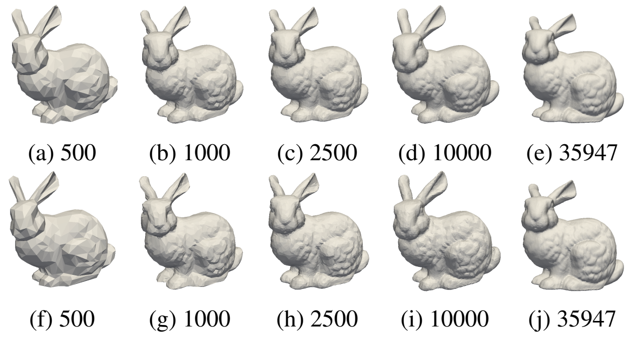


Figure 6: Results of the DEEP3DMARK trained on *m2500* under different mesh sizes. Top(a-e): original mesh G with varying N_v from 500 to 35947. Bottom(f-j): the corresponding watermarked mesh $\hat{G}$ using our DEEP3DMARK. (a-d) are the simplified version of the original mesh (e).

statistical results under size variations. We evaluate our method on meshes with $N_v \leq 100000$. The DEEP3DMARK trained on *m2500* is still effective when mesh size is 40× increased. Compared to the DEEP3DMARK trained on *m2500*, the one trained on *m500* achieves lower accuracy and introduces more distortions. However, it still achieves 81.81% accuracy when the mesh size is 190× increased. Table 4 shows results under geometry variations on ShapeNet. On ShapeNet, the DEEP3DMARK trained on *m2500* achieves an average 93.48% bit accuracy while only introducing 0.0248 L1 norm of vertex difference and 0.0549 Hausdorff difference. The results demonstrate that simplified meshes inherit the relation $D(v_i, v_j)$ distribution from the original meshes. However, as the size of training meshes decreases, the adaptability of GAT decreases as well.

Conclusion

3D watermarking is a key step toward copyright protection. Our paper has introduced DEEP3DMARK, which utilizes graph attention networks to embed binary messages in vertex distributions without texture assistance. Our approach has taken advantage of the property that simplified meshes inherit similar relations from the original ones, specifically the offset vector between adjacent vertices. This approach has enabled the training on simplified meshes but remains effective on larger and previously unseen categories of meshes (adaptability), resulting in fewer distortions and 10%∼50% higher bit accuracy than previous methods when facing attacks. Moreover, extensive experiments have shown that our DEEP3DMARK is robust against unknown mesh attacks, such as smoothing, ARAP, and compression.

Acknowledgments

This work was supported in part by National Key R&D Program of China under Grant 2021YFF0900300, in part by Key Talent Programs of Guangdong Province under Grant 2021QN02X166, and in part by Research Institute of Trustworthy Autonomous Systems under Grant C211153201. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

Al-Khafaji, H.; and Abhayaratne, C. 2019. Graph spectral domain blind watermarking. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2492–2496. IEEE.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; Xiao, J.; Yi, L.; and Yu, F. 2015. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago.

Choi, K.; Tatwawadi, K.; Grover, A.; Weissman, T.; and Ermon, S. 2019. Neural joint source-channel coding. In *International Conference on Machine Learning*, 1182–1192. PMLR.

Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; and Nießner, M. 2017. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE.*

Desbrun, M.; Meyer, M.; Schroder, P.; and Barr, A. H. 2023. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, 149–156.

Dong, C.; Kumar, A.; and Liu, E. 2022. Think Twice Before Detecting GAN-generated Fake Images from their Spectral Domain Imprints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7865–7874.

Fang, H.-S.; Wang, C.; Gou, M.; and Lu, C. 2020. Graspnet-1billion: A large-scale benchmark for general object grasping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11444–11453.

Feng, Y.; Feng, Y.; You, H.; Zhao, X.; and Gao, Y. 2019. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 8279–8286.

Google. 2017. Draco 3D Data Compression. https://google.github.io/draco/. Accessed: 2023-12-25.

Groh, F.; Wieschollek, P.; and Lensch, H. 2018. Flex-convolution. In *Asian Conference on Computer Vision*, 105–122. Springer.

Hermosilla, P.; Ritschel, T.; Vázquez, P.-P.; Vinacua, À.; and Ropinski, T. 2018. Monte carlo convolution for learning on non-uniformly sampled point clouds. *ACM Transactions on Graphics (TOG)*, 37(6): 1–12.

Hertz, A.; Hanocka, R.; Giryes, R.; and Cohen-Or, D. 2020. Deep geometric texture synthesis. *arXiv preprint arXiv:2007.00074.*

Hou, G.; Ou, B.; Long, M.; and Peng, F. 2023. Separable Reversible Data Hiding for Encrypted 3D Mesh Models Based on Octree Subdivision and Multi-MSB Prediction. *IEEE Transactions on Multimedia.*

Hou, J.-U.; Kim, D.-G.; and Lee, H.-K. 2017. Blind 3D mesh watermarking for 3D printed model by analyzing layering artifact. *IEEE Transactions on Information Forensics and Security*, 12(11): 2712–2725.

Hu, S.-M.; Liu, Z.-N.; Guo, M.-H.; Cai, J.-X.; Huang, J.; Mu, T.-J.; and Martin, R. R. 2022. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3): 1–16.

Jiang, R.; Zhou, H.; Zhang, W.; and Yu, N. 2017. Reversible data hiding in encrypted three-dimensional mesh models. *IEEE Transactions on Multimedia*, 20(1): 55–67.

Kim, S.; and Chae, D.-K. 2022. ExMeshCNN: An Explainable Convolutional Neural Network Architecture for 3D Shape Analysis. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 795–803.

Lian, C.; Wang, L.; Wu, T.-H.; Liu, M.; Durán, F.; Ko, C.-C.; and Shen, D. 2019. Meshsnet: Deep multi-scale mesh feature learning for end-to-end tooth labeling on 3d dental surfaces. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 837–845. Springer.

Lindstrom, P.; et al. 1998. Fast and memory efficient polygonal simplification. In *IEEE Vis*, 279–286. IEEE.

Lindstrom, P.; et al. 1999. Evaluation of memoryless simplification. *IEEE Transactions on Visualization and Computer Graphics*, 5(2): 98–115.

Liu, Y.; Fan, B.; Xiang, S.; and Pan, C. 2019. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8895–8904.

Peng, F.; Liao, T.; and Long, M. 2022. A semi-fragile reversible watermarking for authenticating 3d models in dual domains based on variable direction double modulation. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(12): 8394–8408.

Peng, F.; Long, B.; and Long, M. 2021. A general region nesting-based semi-fragile reversible watermarking for authenticating 3D mesh models. *IEEE transactions on circuits and systems for video technology*, 31(11): 4538–4553.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Romero, J.; Tzionas, D.; and Black, M. J. 2022. Embodied hands: Modeling and capturing hands and bodies together. *arXiv preprint arXiv:2201.02610*.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Simonovsky, M.; and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3693–3702.

Son, J.; Kim, D.; Choi, H.-Y.; Jang, H.-U.; and Choi, S. 2017. Perceptual 3D watermarking using mesh saliency. In *International Conference on Information Science and Applications*, 315–322. Springer.

Sorkine, O.; and Alexa, M. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, 109–116. Citeseer.

The CGAL Project. 2022. *CGAL User and Reference Manual*. CGAL Editorial Board, 5.5.1 edition.

Tsai, Y.-Y. 2020. Separable reversible data hiding for encrypted three-dimensional models based on spatial subdivision and space encoding. *IEEE transactions on multimedia*, 23: 2286–2296.

Tsai, Y.-Y.; and Liu, H.-L. 2022. Integrating Coordinate Transformation and Random Sampling Into High-Capacity Reversible Data Hiding in Encrypted Polygonal Models. *IEEE Transactions on Dependable and Secure Computing*.

Van der Maaten, L.; and Hinton, G. 2008. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11).

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, F.; Zhou, H.; Fang, H.; Zhang, W.; and Yu, N. 2022. Deep 3D mesh watermarking with self-adaptive robustness. *Cybersecurity*, 5(1): 1–14.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5): 1–12.

Wu, W.; Qi, Z.; and Fuxin, L. 2019. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9621–9630.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

Xu, H.; Dong, M.; and Zhong, Z. 2017. Directionally convolutional networks for 3D shape segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, 2698–2707.

Xu, Y.; Fan, T.; Xu, M.; Zeng, L.; and Qiao, Y. 2018. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 87–102.

Yoo, I.; Chang, H.; Luo, X.; Stava, O.; Liu, C.; Milanfar, P.; and Yang, F. 2022. Deep 3D-to-2D Watermarking: Embedding Messages in 3D Meshes and Extracting Them from 2D Renderings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10031–10040.

Zhou, H.; Chen, K.; Zhang, W.; Yao, Y.; and Yu, N. 2018. Distortion design for secure adaptive 3-d mesh steganography. *IEEE Transactions on Multimedia*, 21(6): 1384–1398.