Cybersecurity

RESEARCH

Open Access

Deep 3D mesh watermarking with self-adaptive robustness

Feng Wang[1], Hang Zhou[2], Han Fang[3], Weiming Zhang[1] and Nenghai Yu[1*]

Abstract

Robust 3D mesh watermarking is a traditional research topic in computer graphics, which provides an efficient solution to the copyright protection for 3D meshes. Traditionally, researchers need *manually* design watermarking algorithms to achieve sufficient robustness for the actual application scenarios. In this paper, we propose the first deep learning-based 3D mesh watermarking network, which can provide a more general framework for this problem. In detail, we propose an end-to-end network, consisting of a watermark embedding sub-network, a watermark extracting sub-network and attack layers. We employ the topology-agnostic graph convolutional network (GCN) as the basic convolution operation, therefore our network is not limited by registered meshes (which share a fixed topology). For the specific application scenario, we can integrate the corresponding attack layers to guarantee adaptive robustness against possible attacks. To ensure the visual quality of watermarked 3D meshes, we design the curvature consistency loss function to constrain the local geometry smoothness of watermarked meshes. Experimental results show that the proposed method can achieve more universal robustness while guaranteeing comparable visual quality.

Keywords: 3D mesh watermarking, Graph convolution network, Attack layer

Introduction

With the advent of the new industrial revolution, the 3D industry has become an important industry in society. Therefore, 3D graphics model has became a popular data format in many fields such as arts, games and scientific research. As the dominant 3D shape representation of graphics models, 3D meshes have attracted many researchers in the past few years (Garland and Heckbert 1997; Chen et al. 2009). Since designing and producing 3D meshes is a time-consuming and labor-intensive process, protecting the copyright of 3D meshes has also become a popular task in the 3D mesh industry. Robust 3D mesh watermarking (Wang et al. 2008) is an efficient solution to this problem.

Figure 1 shows the general 3D mesh watermarking model. The watermark represents the message to be embedded. First, the embedding module can embed the watermark into a 3D mesh and generate a watermarked mesh. In actual scenarios, there are many complex geometric and topological operations for 3D data (Vasic and Vasic 2013), which can cause serious damage to the watermarked mesh. These operations can be regarded as the attack process. For the extracting process, we employ an extracting module to extract the watermark from the attacked mesh. Note that in this paper, we mainly discuss *blind* watermarking techniques, which means that we can extract the watermark without the reference of the original mesh.

There are the four requirements for 3D mesh watermarking task. *Robustness*: The watermark should be resilient and not easily removable by possible attacks during the transmission channel. *Imperceptibility*: The visual quality of the watermarked mesh should be guaranteed. *Efficiency*: The time for embedding and extracting the watermark should be as short as possible. *Capacity*: larger capacity indicate that we can embed more message into the mesh. Among all, the most important

*Correspondence: ynh@ustc.edu.cn

[1] University of Science and Technology of China, Hefei, China
Full list of author information is available at the end of the article

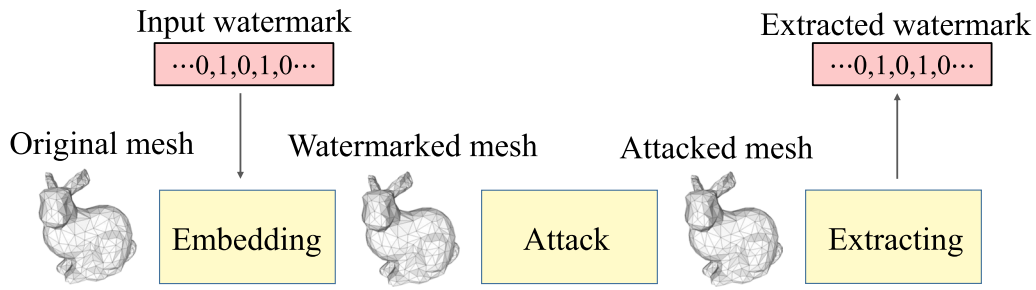

Fig. 1 3D mesh watermarking model. Actual watermarking procedure includes three processes: watermark embedding, attack and watermark extracting. Attack process represents the possible operations on the watermarked mesh

requirement is robustness, which directly influences the protection ability and transmission accuracy.

To achieve the above properties, we have to make sufficient efforts to cope with 3D meshes. A 3D mesh can be defined by its vertices and faces, where vertices define the 3D coordinates in the Euclidean space, and faces indicate the topological structure of the mesh. For the data in a 3D mesh file, various attacks may modify it in different ways. These attacks can be divided into three types (Wang et al. 2010): *vertices reordering attack*, *geometry attack* and *connection attack*. Vertices reordering attack can reorder the vertices in the 3D file but does not change the 3D coordinates or the topology. Thus it would not change the mesh shape. For the geometry attack, it modifies the vertex coordinates without changing the topological connection. Geometry attacks include similarity transformation, noise addition and smoothing, *etc*. Similarity transformation operations consist of three types of transformation: translation, rotation and uniform scaling. Noise addition simulates the artifacts generated during the mesh transmission (e.g. Gaussian noise addition). And smoothing is a common processing operation for 3D meshes, to remove the unevenness of the mesh surface. Contrary to the geometry attack, the connection attack, such as cropping, modifies the topological connection between vertices, causing intense damage to the geometric properties of the mesh.

Designing a watermarking algorithm robust against all attacks is impossible. Traditionally, in a specific scenario, to achieve better robustness, we must manually design the specific watermarking algorithm to resist the possible attacks. For example, real-time 3D model rendering needs intense mesh simplification and optimization, which may remove the watermark data (Vasic and Vasic 2013). As most algorithms are not robust enough against both attacks, we need to develop a specific watermarking algorithm to cope with such scenario. However, designing a watermarking algorithm for every specific scenario is labor-intensive. What's more, it's difficult to manually design algorithms robust against some attacks, such as the cropping attack.

To overcome these shortcomings and design a more general watermarking framework (Zhang et al. 2020, 2021a, b) for robust 3D mesh watermarking, we propose the first deep learning-based method, which can achieve more universal robustness than traditional methods. In detail, we propose an end-to-end network, consisting of an embedding sub-network, an extracting sub-network and attack layers. Both sub-networks are trained to achieve the watermark embedding and extracting. And attack layers simulate the actual attacks in the specific scenario. With the differential attack layers, we can jointly train the whole network to find the theoretically optimal solution in the current scenario. For different scenarios, we can adaptively adjust the attack layers to meet the various requirements.

There have existed some researches on deep learning-based methods for 2D watermarking (Zhu et al. 2018; Wengrowski and Dana 2019; Jia et al. 2021; Zhang et al. 2021c; Luo et al. 2020; Tancik et al. 2020; Wang et al. 2020), which design a convolutional neural network for image watermarking. However, compared with 2D data, the convolution on 3D mesh has more difficulties because of the irregularity and complexity. And 3D mesh watermarking suffers from more threats with the increased dimensional space. As graph convolutional network (GCN) (Kipf and Welling 2017) can be applied in deep learning on 3D mesh, it is difficult to converge the network to the optimal solution with the varied topologies of different 3D meshes. Therefore, we propose the topology-agnostic GCN to adapt the network to different topologies. And consequently, our network can be applied to non-template-based meshes (meshes do not have to share a fixed topology). The pre-trained model also has enough transferability on another remeshed dataset. To better measure the distance between the original mesh and the watermarked mesh, we propose the curvature consistency loss as a constraint for watermarked meshes.

In summary, our main contributions are three-fold:

- We are the first to introduce a deep learning-based method for robust 3D mesh watermarking task. We hope we can open up new research direction and inspire more works in this field.
- We propose a novel deep 3D mesh watermarking network to achieve the adaptive robustness to specific attacks. The curvature consistency loss is proposed to guarantee the visual quality of watermarked meshes.
- We quantitatively and qualitatively evaluate the proposed method with two datasets. Experimental results demonstrate that the proposed method can achieve more universal robustness and higher efficiency than baseline methods while guaranteeing comparable visual quality and the same capacity. Besides, our method can be applied to non-template-based meshes, which is very practical in the actual application scenarios.

Related work

Traditional robust 3D mesh watermarking

Robust 3D mesh watermarking methods can be divided into two categories: spatial domain-based methods (Cho et al. 2007; Bors and Luo 2012; Rolland-Neviere et al. 2014; Lee et al. 2021; Jang et al. 2018; Zhou et al. 2018) and transform domain-based methods (Cayre et al. 2003; Uccheddu et al. 2004; Wang et al. 2008; Hamidi et al. 2017; Liu et al. 2017).

Spatial domain-based methods usually embed the watermark by modifying the spatial parts of a 3D mesh, thus relatively weak to connectivity attack and noise addition attack. And the original structures of 3D meshes can be destroyed by the watermark embedding process, which affects the subsequent mesh synchronization (causality problem). Cho et al. (2007) proposed a classic watermarking algorithm based on the distribution of distances between vertices and the mesh gravity center. Before embedding, the vertices are grouped into bins and each bin is assigned with one watermark bit. Based on Cho et al. (2007), some optimization algorithms are proposed in Bors and Luo (2012) and Rolland-Neviere et al. (2014). The visual quality can be improved but more time is costed during the optimization. Zhou et al. (2018) proposed to design a distortion function based on vertex normals and embeds bit information into bit planes of vertex coordinates. Jang et al. (2018) proposed to use the shape diameter function (SDF) to divide a 3D mesh into several segments. Then the watermark is embedded into all the segmented regions. Recently, Lee et al. (2021) proposed a novel watermarking technique based on spherical coordinate and skewness measurement. The vertices are also grouped into bins, but the watermark bit is embedded according to the skewness value. Therefore, the robustness can be highly enhanced.

For transform domain-based methods, the common operation is applying the spectral analysis to the original mesh. Then the watermark is embedded by modifying the spectral coefficients of medium frequency parts so that the modification spreads to the spatial components of a mesh. Unfortunately, existing spectral analysis tools have their limitations on the robustness performance against some attacks (Wang et al. 2008). In Cayre et al. (2003), first proposed to employ Laplacian matrix as the spectral analysis tool in 3D mesh watermarking task. Uccheddu et al. (2004) proposed a wavelet-based watermarking algorithm but the capacity is limited in one bit. Wang et al. (2008) proposed the hierarchical watermarking algorithm based on wavelet transform. This method can allow for higher capacity, but with weaker robustness. Based on this algorithm, Hamidi et al. (2017) proposed quantize the wavelet coefficient vectors and embed the watermark bit into the ratio relationship between the quantized wavelet coefficient vectors. Liu et al. (2017) proposed a multi-resolution adaptive parameterisation-based 3D mesh watermarking method. The vertices at the coarse level are used to establish an invariant space and the vertices at the fine level are selected as feature vertices for watermark embedding.

Deep learning-based methods for 3D mesh representations

Different from convolution operation on images, convolution operation on 3D meshes is difficult due to their irregularity and complexity. To lift this limitation, some researches (Feng et al. 2019; Hanocka et al. 2019; Hu et al. 2021; Milano et al. 2020; Verma et al. 2021) have been proposed to effectively learn 3D mesh representation. Yet they can only be applied in discriminative tasks such as classification and semantic segmentation. For generative tasks such as 3D reconstruction, most mesh-based methods use graph convolutional network (GCN) (Kipf and Welling 2017) as the basic convolution operation, where vertices and edges are regarded as nodes and connections in a graph.

$$f_i^{l+1} = \phi\left(w_i f_i^l + \sum_{j \in \mathcal{N}(i)} w_j f_j^l\right), \tag{1}$$

where $\mathcal{N}(i)$ defines the neighboring vertices of the vertex i, f_i^l is the l-layer feature of the vertex i, and ϕ is the activation function. Usually, they predict the 3D mesh shape as a deformation from a template (Wang et al. 2018;

Hanocka et al. 2020). Besides, a series of efforts (Gao et al. 2021; Gong et al. 2019; Zhou et al. 2020) have been proposed to train deep neural auto-encoders to learn latent representations for 3D meshes. These methods usually employ anisotropic filters (each weight w_j variable for every neighboring vertex) to represent the 3D mesh. However, these filters are usually defined based on the fixed vertex order or fixed edge order.

Deep learning-based methods for digital images watermarking

There have been some deep learning-based researched on 2D watermarking (Zhu et al. 2018; Wengrowski and Dana 2019; Luo et al. 2020; Tancik et al. 2020; Wang et al. 2020; Jia et al. 2021). Zhu et al. (2018) proposed HiDDeN, an end-to-end deep image watermarking framework. This framework includes an Encoder module, an Decoder module and noise layers. The Encoder and Decoder are responsible for watermark embedding and extracting respectively. And noise layers simulate the possible attacks during the image transmission process, such as dropping, Gaussian noise and cropping, *etc*. Based on HiDDeN, subsequent researchers mainly concentrated on designing new noise layers and extend the application scenarios. Luo et al. (2020) proposed to replace fixed image attacks with the adversarial network. Wang et al. (2020) proposed a two-stage training strategy to train the network on non-differentiable noise layers such as JPEG comppression (Liu et al. 2021). Jia et al. (2021) proposed an novel training strategy with the mini-batch of simulated JPEG compression, real JPEG compression, and noiseless training to enhance the robustness against JPEG compression. Zhang et al. (2021c) proposed to decouple the forward process of noise layers and achieve the joint training of Encoder and Decoder with any non-differentiable noise layers. Tancik et al. (2020) proposed StegaStamp, which models the print-photography process and demonstrated the robustness against real world attacks. Wengrowski and Dana (2019) proposed Light Field Messaging (LFM), which constructed over one million original image-screenshot pairs and trained a network that simulates the distorting effects of camera-display transfer.

Proposed approach

Topology-agnostic GCN

Due to the possible attacks, watermarked meshes cannot simply be treated as template-based meshes. Even original meshes can also be non-template-based in the actual scenario. To represent these meshes, we employ isotropic filters to compose our convolution operation, with a fixed w_j in Eq. 1 for each neighboring vertex:

$$f_i^{l+1} = \phi\left(w_0 f_i^l + \sum_{j\in\mathcal{N}(i)} w_1 f_j^l\right). \tag{2}$$

During training, we find our network converges slowly. We analyze this phenomenon for two reasons: randomly generated watermark bits in each iteration step and different connectivity for each vertex. To speed up training and ensure the convergence, we apply the degree normalization in GCN and design the GraphConv+BatchNorm+ReLU block as the main component of our network. We first define our GraphConv operation:

$$f_i^{l+1} = w_0 f_i^l + w_1 \sum_{j\in\mathcal{N}(i)} \frac{1}{|\mathcal{N}(i)|} f_j^l, \tag{3}$$

where $|\cdot|$ denotes the cardinal number, indicating the vertex degree. Different from previous GCNs in generative tasks, the topology for each 3D mesh is agnostic. For each mesh with its own topology, topology-agnostic GCN needs to search the neighboring vertices for every vertex. For every mini-batch data, we employ the batch normalization operation to normalize the feature from the output of GraphConv. Then we define the graph residual block consisting of two GraphConv+BatchNorm+ReLU blocks with a short connection (He et al. 2016), as shown in Fig. 2. For the initial block of the embedding sub-network and extracting sub-network, the input feature is the 3D coordinates of vertices and outputs 64-dim feature. For other blocks, the output feature has the same shape as the input feature with 64 dimensions.

As shown in Fig. 3, our network includes a watermark embedding sub-network, attack layers and a watermark extracting sub-network. In the network, we define a 3D mesh as $\mathcal{M} = (\mathcal{V}, \mathcal{F})$, where $\mathcal{V}$ denotes vertices and $\mathcal{F}$ denotes faces. And we use N_{in} to denote the number of input vertices. For each vertex $i \in \mathcal{V}$, we use $\mathbf{v}_i = [x_i, y_i, z_i]^{\mathrm{T}} \in \mathbb{R}^3$ to denote the 3D coordinates in the Euclidean space. And we define watermark length as C bits.

Watermark embedding sub-network

In this sub-network, we take original mesh $\mathcal{M}_{in} = (\mathcal{V}_{in}, \mathcal{F}_{in})$ and watermark $\mathbf{w}_{in}$ as the input. We employ five cascaded graph residual blocks to form the feature learning module $\mathbf{F}$. We first employ this module to learn the feature map F_{in} from input vertices $\mathcal{V}_{in}$. The watermark encoder $\mathbf{E}$ is responsible for encoding

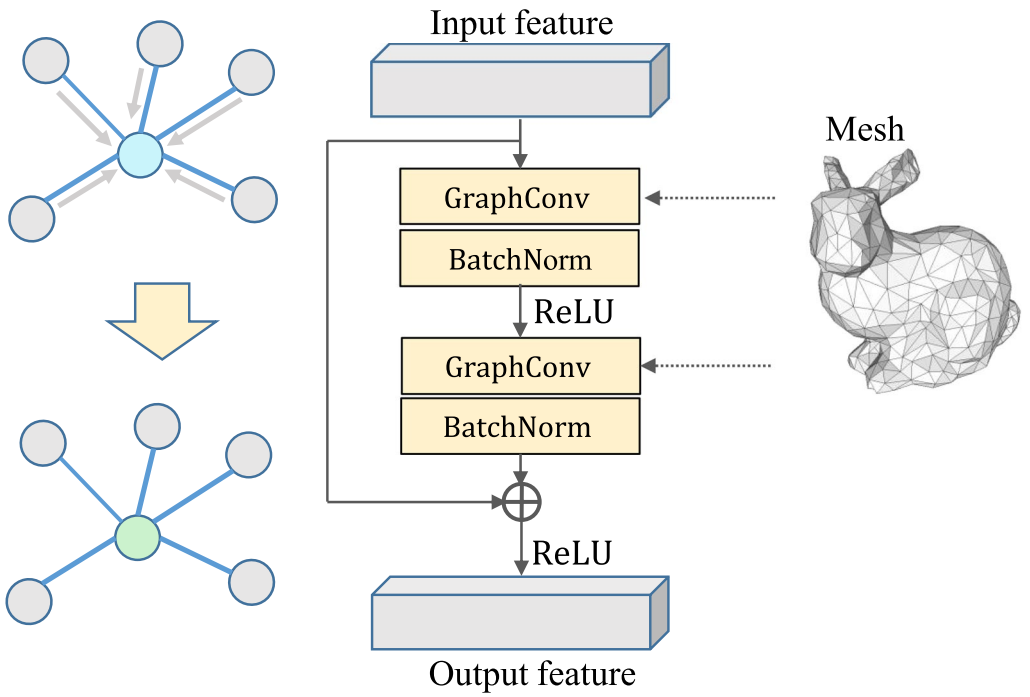

Fig. 2 Graph residual block. The dashed line represents that the mesh is utilized for querying the adjacent vertices in GraphConv operation

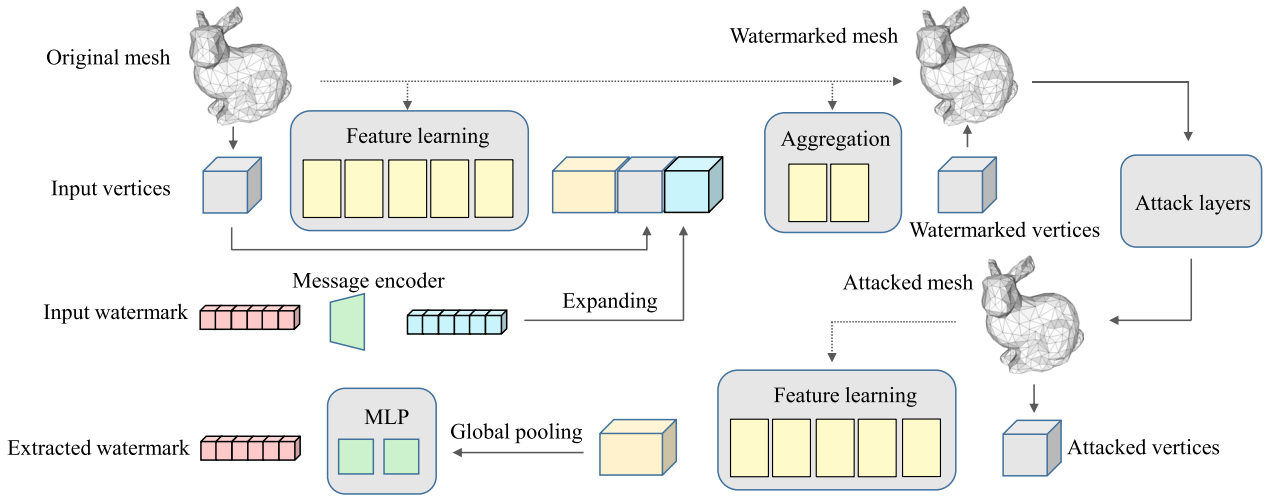

Fig. 3 Proposed watermark network architecture. The dashed line represents the reference information that guides the convolution operation and mesh reconstruction

the input watermark into a latent code $\mathbf{z}_w$ by a fully connected layer. Then the latent code $\mathbf{z}_w$ is expanded along the number of vertices to align the vertices. After expanding, the latent code is concatenated with input vertices $\mathcal{V}_{in}$ and the mesh feature F_{in}, and then fed into the aggregation module $\mathbf{A}$. In the last block of $\mathbf{A}$, there is a branch that applies an extra GraphConv layer and outputs the 3D coordinates of watermarked vertices $\mathcal{V}_{wm}$. The aggregation module $\mathbf{A}$ includes two graph residual blocks and outputs the 3D coordinates of mesh vertices. According to the original mesh $\mathcal{M}_{in}$ and watermarked vertices $\mathcal{V}_{wm}$, the watermarked 3D mesh $\mathcal{M}_{wm}$ can be

constructed. Note that the symmetric function *Expanding* is used to align the vertices and the watermark feature, making the embedding process invariant to the reordering of input vertices, which may be very practical in the actual scenario.

Attack layers

To guarantee the adaptive robustness to specific attacks, we train our network with attacked meshes. In this paper, we mainly consider representative attacks (including cropping, Gaussian noise, rotation and smoothing) and integrate them into attack layers. Note that we can integrate different attacks as the attack layers, according to the actual requirements.

Rotation

We rotate the 3D mesh in three dimensions with the rotation angle randomly sampled in every dimension. We use θ to denote the rotation scope and the rotation angle in each dimension is randomly sampled: $\theta_x, \theta_y, \theta_z \sim U[-\theta, \theta]$. Then we rotate $\mathcal{V}_{wm}$ with the corresponding angle for every dimension in the Euclidean space.

Gaussian noise

We employ a zero-mean Gaussian noise model, sampling the standard deviation $\sigma_g \sim U[0, \sigma]$ to generate random noise to 3D meshes. We generate $noise \sim \mathcal{N}(0, {\sigma_g}^2)$ and attach it on the 3D coordinates of watermarked vertices.

Smoothing

Laplacian smoothing model (Taubin 2000) is employed to simulate the possible smoothing operation. For the watermarked mesh $\mathcal{M}_{wm} = (\mathcal{V}_{wm}, \mathcal{F}_{wm})$, we first calculate the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N_{in} \times N_{in}}$, and use $\alpha_s \sim U[0, \alpha]$ to control the level of Laplacian smoothing. For the coordinate matrix $\mathbf{V}_{wm} \in \mathbb{R}^{N \times 3}$ of watermarked vertices $\mathcal{V}_{wm}$, we calculate the the coordinate matrix $\mathbf{V}_{att}$ of attacked vertices $\mathcal{V}_{att}$ as :

$$\mathbf{V}_{att} = \mathbf{V}_{wm} - \alpha_s \mathbf{L} \mathbf{V}_{wm}. \tag{4}$$

Cropping

We simulate this attack by cutting off a part of the mesh. We first normalize the vertices in a unit square and search for the two farthest points in the negative quadrant and the positive quadrant respectively. Then We connect two points and simulate using a knife cutting perpendicular to the line. So that we can cut off the part of the mesh, with β to control the minimum ratio of the reservation. $\beta_c \sim U[\beta, 1]$ is used to denote the actual ratio of the reservation at each cropping operation.

During training, we set the hyperparameters as follows: $\theta = 15°, \sigma = 0.03, \alpha = 0.2, \beta = 0.8$. Besides four attacks, we also integrate one identity layer which does not have any attack, to ensure the performance when no attack is suffered. During training, we randomly select one attack as the attack layer in each mini-batch. Then we can generate the attacked mesh $\mathcal{M}_{att} = (\mathcal{V}_{att}, \mathcal{F}_{att})$ after the watermarked mesh $\mathcal{M}_{wm} = (\mathcal{V}_{wm}, \mathcal{F}_{wm})$ passes through the attack layer. Figure 4 shows the original and attacked meshes under different attacks. With the differentiable attack layers, we can jointly train our embedding sub-network and extracting sub-network, and update the parameters simultaneously.

Watermark extracting sub-network

We design a straightforward structure to extract the watermark. For the attacked vertices $\mathcal{V}_{att}$, we first employ the same feature learning module $\mathbf{F}$ to acquire the feature map F_{no}. Followed by the global average pooling layer and a two-layer fully connected layer (MLP), the extracted watermark $\mathbf{w}_{ext}$ is obtained. The symmetric function *Global pooling* aggregates information from all vertices, which can also guarantee the variance under the vertices reordering attack.

Loss function

To train the network, we define some loss functions. Mean square error (MSE) loss is first employed for constraining the watermark and mesh vertices:

$$l_w(\mathbf{w}_{in}, \mathbf{w}_{ext}) = \frac{1}{C} ||\mathbf{w}_{in} - \mathbf{w}_{ext}||_2^2, \tag{5}$$

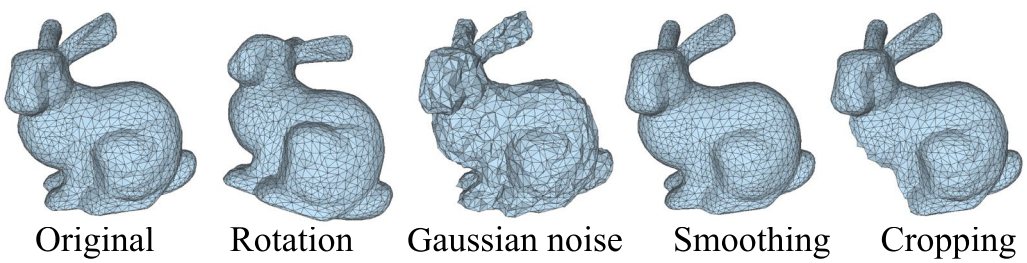

Fig. 4 Stanford Bunny model and its attacked meshes

$$l_m(\mathcal{M}_{in}, \mathcal{M}_{wm}) = \frac{1}{N_{in}} \sum_{i \in \mathcal{V}_{in}} ||\mathbf{v}_i - \mathbf{v}_{i'}||_2^2, \tag{6}$$

where i' denotes the paired vertex of vertex i in the watermarked mesh $\mathcal{M}_{wm}$.

l_m can constrain the spatial modification on mesh vertices as a whole. Yet the local geometry smoothness is also supposed to be guaranteed, as it greatly affects the visual perception of human eyes (Mariani et al. 2020). The local curvature can reflect the surface smoothness property (Torkhani et al. 2012). For 3D meshes, the local curvature should be defined based on the connection relations. As shown in Fig. 5, we use $\theta_{ij} \in [0°, 180°]$ to represent the angle between the normalized normal vector $\mathbf{n}_i$ for vertex i and the direction of neighboring vertex j. We can find that the vertex's neighboring angles represent the local geometry. For each vertex i in the mesh $\mathcal{M}$, we define the vertex curvature as:

$$cur(i, \mathcal{M}) = \sum_{j \in \mathcal{N}_i} \cos(\theta_{ij}), \tag{7}$$

where

$$\cos(\theta_{ij}) = \frac{(\mathbf{v}_j - \mathbf{v}_i)^{\mathrm{T}} \mathbf{n}_i}{||\mathbf{v}_j - \mathbf{v}_i||_2}. \tag{8}$$

To guarantee the local curvature consistency between original 3D mesh $\mathcal{M}_{in}$ and watermarked 3D mesh $\mathcal{M}_{wm}$, we define the curvature consistency loss function:

$$l_{cur}(\mathcal{M}_{in}, \mathcal{M}_{wm}) = \frac{1}{N_{in}} \sum_{i \in \mathcal{V}_{in}} ||(cur(i, \mathcal{M}_{in}) - cur(i', \mathcal{M}_{wm}))||_2^2. \tag{9}$$

The combined objective is employed in the network: $\mathcal{L} = \lambda_1 l_w + \lambda_2 l_{cur} + \lambda_3 l_m$. By default, $\lambda_1 = \lambda_2 = 1$, and $\lambda_3 = 5$.

Experiments

Implementation details

Our network is implemented by PyTorch and trained on two NVIDIA GeForce RTX 2080Ti GPUs. Kingma and Ba (2015) is applied as the gradient descent

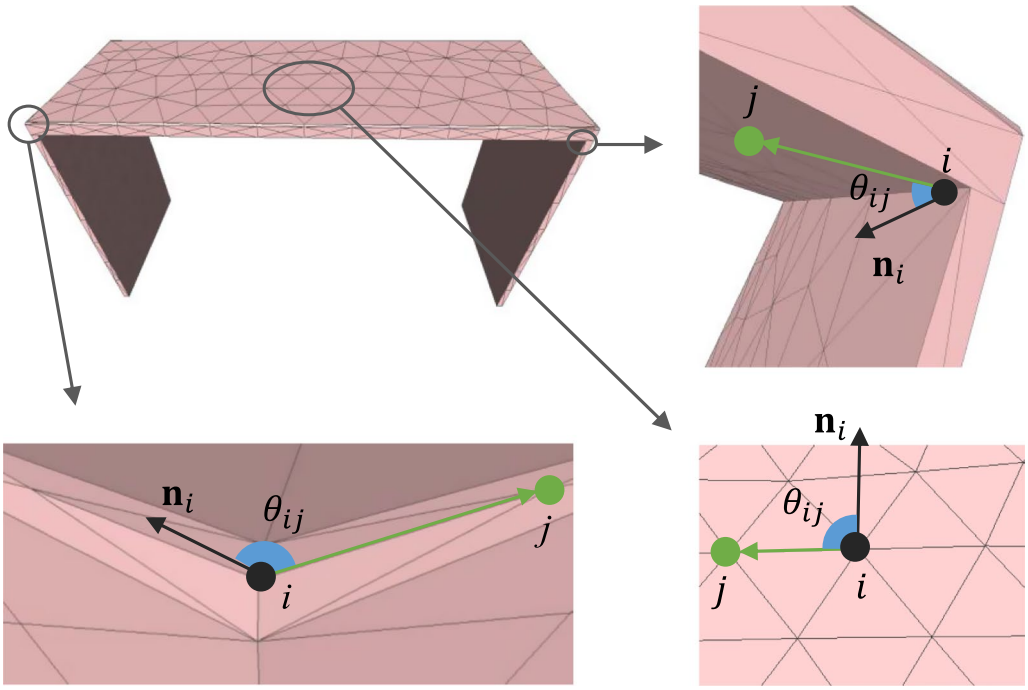

Fig. 5 This is a 3D model of a desk. In the bottom right figure, the desktop is flat and the normal vector $\mathbf{n}_i$ is perpendicular to the local area. For each $j \in \mathcal{N}_i, \theta_{ij} = 90°$. The bottom left figure and the top right figure are the convexity and concavity of the the desk respectively, with $\theta_{ij} > 90°$ and $\theta_{ij} < 90°$

algorithm with the learning rate of 0.0001. We use two scanned datasets: 2D-manifold Hand dataset (triangle meshes with 778 vertices and 1538 faces) (Romero et al. 2017) and 3D-manifold Asiadragon dataset (tet meshes with 959 vertices and 10364 faces) (Stanford 2021). For Hand dataset, they are divided into 1554 models for train and 50 models for test, and the batch size is 600. For Asiadragon dataset, we use models provided by Zhou et al. (2020), with 7503 models for train and 500 models for test, and the batch size is 400. The network is trained with about one week on both datasets respectively. Before feeding meshes into the network, we normalize vertices to a unit cube. In the experiment, we set the watermark length of all methods as $C = 64$.

Evaluation metrics

We employ Hausdorff distance (HD), maximum root mean square (MRMS) and the curvature consistency loss l_{cur} to measure the distances between watermarked meshes and original meshes. To evaluate the robustness, we compare the input watermark bits and extracted watermark bits, and calculate the bit accuracy. Besides, we test algorithms on Intel Xeon Gold 5218 CPU (2.30 GHz) and record the mean time consumption for one 3D mesh to compare the efficiency.

Comparisons with baseline methods

We select five methods as our baseline methods: two classic watermarking methods: Cho et al. (2007) and Cayre et al. (2003), one optimization-based method L-M (Bors and Luo 2012), and two latest methods: MAPS (Liu et al. 2017) and SCKM (Lee et al. 2021). As there are few open source codes for these methods, we have tried our best to reproduce them.

Table 1 shows the quantitative comparisons with baseline methods. The proposed method outperforms all other methods in terms of accuracy. On Hand and Asiadragon dataset, we can get the accuracy of 92.06% and 95.22% respectively. There is maximum difference of nearly 20% between ours and baseline methods. That demonstrate the clear advantage in terms of robustness for our method. In terms of visual assessment indicators, the proposed method perform worse than Bin, L-M, MAPS and SCKM. That's because they only make minor modifications to the grouped vertices, yet the proposed method need to learn the neural representation for 3D meshes, which is currently difficult to achieve the competitive quality as the former. However, as shown in Fig. 6, the proposed method can still keep comparable visual quality and make the watermarked perturbations imperceptible. Compared with HD and MRMS, the vertex curvature *cur* can better reflect the local geometry smoothness. With the curvature consistency loss l_{cur} employed during training, the proposed method causes little surface curvature distortion on the watermarked mesh. For Asiadragon dataset, the proposed method get 0.001 of l_{cur}, but Laplacian gets 30× of l_{cur}. Besides, we can find that the proposed method can acquire better visual quality than Laplacian. In Fig. 6, Laplacian causes more distortions on the surface smoothness, making artifacts of watermarked meshes clearly visible. For the efficiency comparison, the proposed method also has comparable performance. And Bin, L-M and Laplacian cost at least 10× of our time for the watermark embedding.

Table 1 Quantitative comparisons with baseline methods on Hand and Asiadragon dataset

	Method	HD	MRMS	l_{cur}	Accuracy (%)	Emb time (s)	Ext time (s)
Asiadragon	Bin (Cho et al. 2007)	0.0027	**0**	**0**	74.47	0.406	0.015
	L-M (Bors and Luo 2012)	0.0027	**0**	**0**	76.62	2547	0.015
	Laplacian (Cayre et al. 2003)	0.0319	0.012	0.030	94.34	0.725	0.566
	MAPS (Liu et al. 2017)	0.0002	**0**	**0**	75.41	**0.004**	**0.003**
	SCKM (Lee et al. 2021)	**0**	**0**	**0**	85.39	0.012	0.012
	Proposed	0.0498	0.014	0.001	**95.22**	0.032	0.016
Hand	Bin (Cho et al. 2007)	0.0032	**0**	**0**	71.84	0.327	0.013
	L-M (Bors and Luo 2012)	0.0063	**0**	**0**	70.09	1040	0.013
	Laplacian (Cayre et al. 2003)	0.0208	0.006	0.028	91.22	0.523	0.347
	MAPS (Liu et al. 2017)	0.0001	**0**	**0**	74.25	**0.010**	**0.007**
	SCKM (Lee et al. 2021)	**0**	**0**	**0**	86.34	0.011	0.011
	Proposed	0.1131	0.018	0.003	**92.06**	0.022	0.012

Distance between original meshes and watermarked meshes (second to fourth column), bit accuracy under attacks from the attack layers (%, fifth column) and running time (sixth to seventh column). Running time consists of the watermark embedding time and watermark extracting time for one 3D mesh. For all indicators the lower the better except accuracy

The bold represents the ones with best preformance

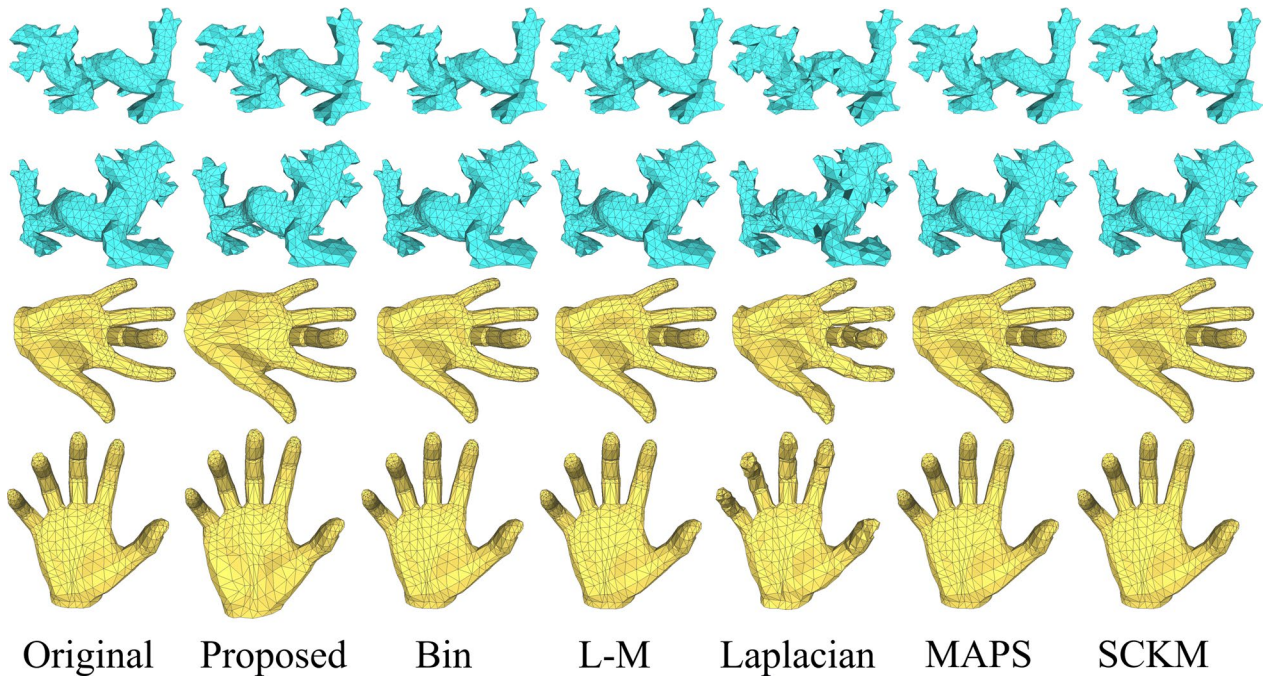

Fig. 6 Qualitative comparison on visual quality with baseline methods on Asiadragon (top two rows) and Hand (bottom two rows) dataset

As shown in Fig. 7, we test the bit accuracy under each attack with different intensities. L-M, Bin and MAPS are robust against the rotation attack, but perform badly under other attacks, even with near 50% of accuracy rate under Gaussian noise attack. Laplacian can keep relatively high accuracy under low-intensity attacks, but its accuracy decreases rapidly with the attack intensity increasing. And SCKM also performs badly with high-intensity attack. Compared with baseline methods, the proposed method can achieve more universal robustness under all attacks. Although the proposed method cannot guarantee to outperform baseline methods under all conditions, we can still keep the sufficient accuracy under intense attacks, which guarantees the practicality in the actual scenario. For example, we can still obtain the accuracy rate of about 90% on Hand dataset under smoothing attack with $\alpha = 0.8$. And under cropping attack with $\beta = 0.3$, we have more than 80% accuracy rate on Asiadragon dataset.

The importance of the attack layers

As described above, to enhance the robustness against specific attacks, we employ the attack layers during training. To demonstrate the necessity, we also train our network without the attack layers (labelled with †). As shown in Fig. 7, we can find that the accuracy decreases a lot under all attacks when training without the attack layers. Under the rotation attack with $\theta = 30°$, the model training without the attack layers is about 30% of accuracy rate lower than the default model. Under the smoothing attack with $\alpha = 0.8$, the accuracy rate is only about 70% in Asiadragon dataset. When training with the attack layers, the accuracy rate can surpass 90%.

The importance of the curvature consistency loss

Besides MSE loss constraining the spatial range of vertices, curvature consistency loss can guarantee the

Table 2 Quantitative results on remeshed Hand and remeshed Asiadragon dataset

	Method	HD	MRMS	l_{cur}	Accuracy (%)
Asiadragon	Model	0.059	0.014	**0.002**	**93.25**
	Model†	**0.042**	**0.010**	**0.002**	84.63
	Pre-Model	0.069	0.016	0.006	83.63
	Pre-Model†	0.058	0.012	0.004	76.51
Hand	Model	0.112	0.018	0.005	**91.78**
	Model†	0.101	**0.012**	**0.003**	85.09
	Pre-Model	0.113	0.019	0.004	83.00
	Pre-Model†	**0.099**	**0.012**	**0.003**	77.34

Distance between original meshes and watermarked meshes (second to fourth column), bit accuracy under attacks from the attack layers (%, last column). † denotes training without the attack layers, Model represents the model trained on the remeshed dataset, and Pre-Model represents the model trained on the original dataset

The bold represents the ones with best preformance

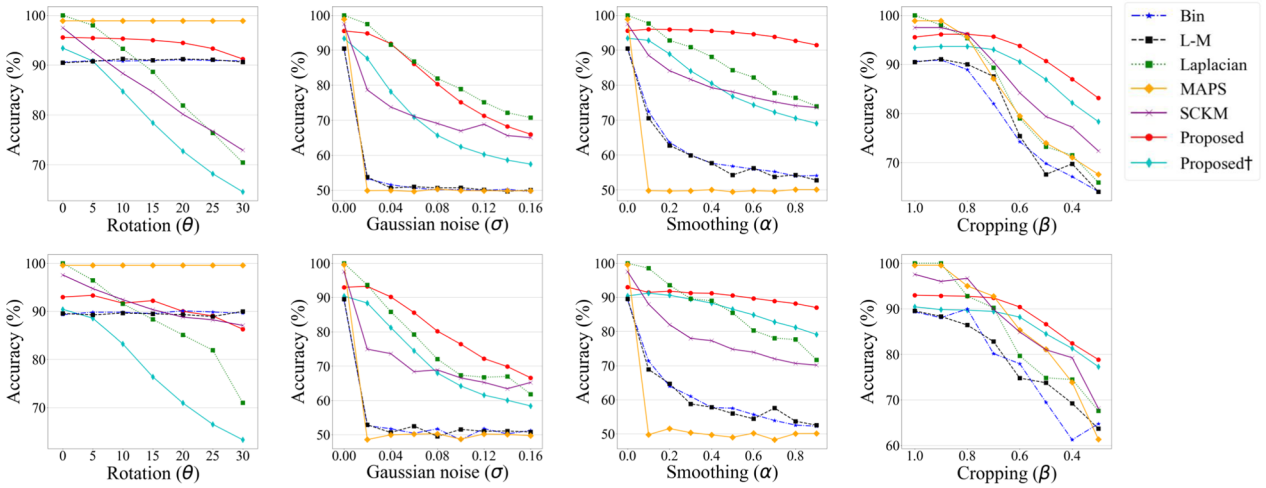

Fig. 7 Bit accuracy rate (%) under different attacks and intensities on Asiadragon (top) and Hand (bottom) dataset. Proposed (red), Proposed† (cyan, † denotes training without the attack layers), Bin Cho et al. (2007) (blue), L-M Bors and Luo (2012) (black), Laplacian Cayre et al. (2003) (green), MAPS (Liu et al. 2017) (orange) and SCKM (Lee et al. 2021) (purple) are showed

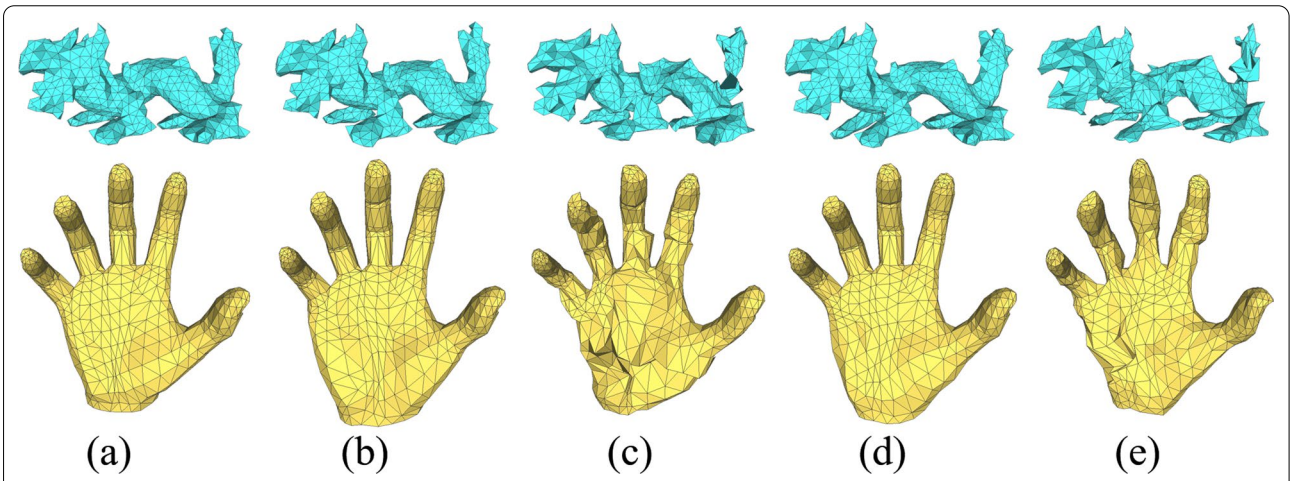

Fig. 8 The importance discussion of curvature consistency loss on Asiadragon (top) and Hand (bottom) dataset. **a** Original; **b** Proposed (w/ l_{cur}); **c** Proposed (w/o l_{cur}); **d** Proposed† (w/ l_{cur}); **e** Proposed† (w/o l_{cur})

surface smoothness of watermarked meshes. To validate its importance for the visual quality of watermarked meshes, we retrain our models without the curvature consistency loss. As shown in Fig. 8, we can find that there are many visual artifacts on the watermarked meshed when training without the curvature consistency loss.

Performances on non-template-based datasets and the transferability discussion

In the above sections, we mainly discuss the performance of the proposed method on template-based 3D meshes. In the actual scenario, we may need to embed the watermark into non-template-based meshes. To evaluate the proposed method on non-template-based datasets, we independently remesh each shape of Hand and Asiadragon dataset to 1024 vertices by Trimesh library (https://trimsh.org/). So that each dataset is made up of non-template-based meshes (They do not share a fixed topology). Then we retrain our network using each remeshed dataset. Meanwhile, using new dataset, we also retrain the network without the attack layers (labelled with †). Besides, to test the transferability of our method, we also test our pre-trained model on the remeshed dataset, which is trained with original Hand dataset and Asiadragon dataset. For the sake of distinction, we use

Model to denote the model trained on the remeshed dataset, and use Pre-Model to denote the model trained on the original dataset.

Table 2 shows the quantitative results. As our GCN is topology-agnostic, when trained with the remeshed dataset, we can still get the similar performance on imperceptibility and robustness. We can get 93.25% of accuracy rate on remeshed Asiadragon dataset and 91.78% of accuracy rate on remeshed Hand dataset. And the visual quality can also be guaranteed as shown in Fig. 9. For Pre-Model, we can still guarantee comparable performance when tested on remeshed datasets. There is still 83.63% of accuracy rate on remeshed Asiadragon dataset and 83.00% of accuracy rate on remeshed Hand dataset. And in terms of MRMS, HD and l_{cur}, the performance of Pre-Model is as similar as Model. In addition, we can find that the attack layers can help improve the accuracy both in Pre-Model and Model.

As shown in Fig. 10, we test the bit accuracy under each attack with different intensities on remeshed datasets. Under each attack, Pre-Model exhibits the similar curve property to Model, with about 10% of accuracy rate reduction. Pre-Model can keep more than 78% accuracy rate under rotation attack, and more than 75% accuracy rate under smoothing attack. That means the proposed method can also guarantee the transferability of the pre-trained model on the another remeshed dataset.

Discussion: How does our network embed the watermark into the 3D mesh?

Different from traditional methods, we do not know how the network modifies the vertices and embeds the watermark into 3D meshes. Therefore, we explore to analyze the modification based on spatial domain and transform domain. For watermarked vertices and original vertices, we calculate the distances between them in the Euclidean space. Then we color the original 3D mesh based on the l_2 distance. As shown in Fig. 11, we can find that our network prefers to modify vertices on flatting areas, such as the wrist, yet the fingers have fewer modifications. We speculate that there are undulating curvatures in the finger areas, resulting in larger loss from modifications. So the network is trained to prefer to embed the watermark bit in relatively flatting areas.

Meanwhile, we perform Laplace-Beltrami operator on Hand dataset and calculate the mean power spectrum of 3D meshes (Cayre et al. 2003). The residual power spectrum between watermarked meshes and original meshes is also calculated. In Fig. 12, low coefficients represent the principal components of the mesh, with higher power spectrum intensity. In the right figure, we find that low coefficients also have more residual power spectrum. That means the network prefers to modify the vertices on the principal components.

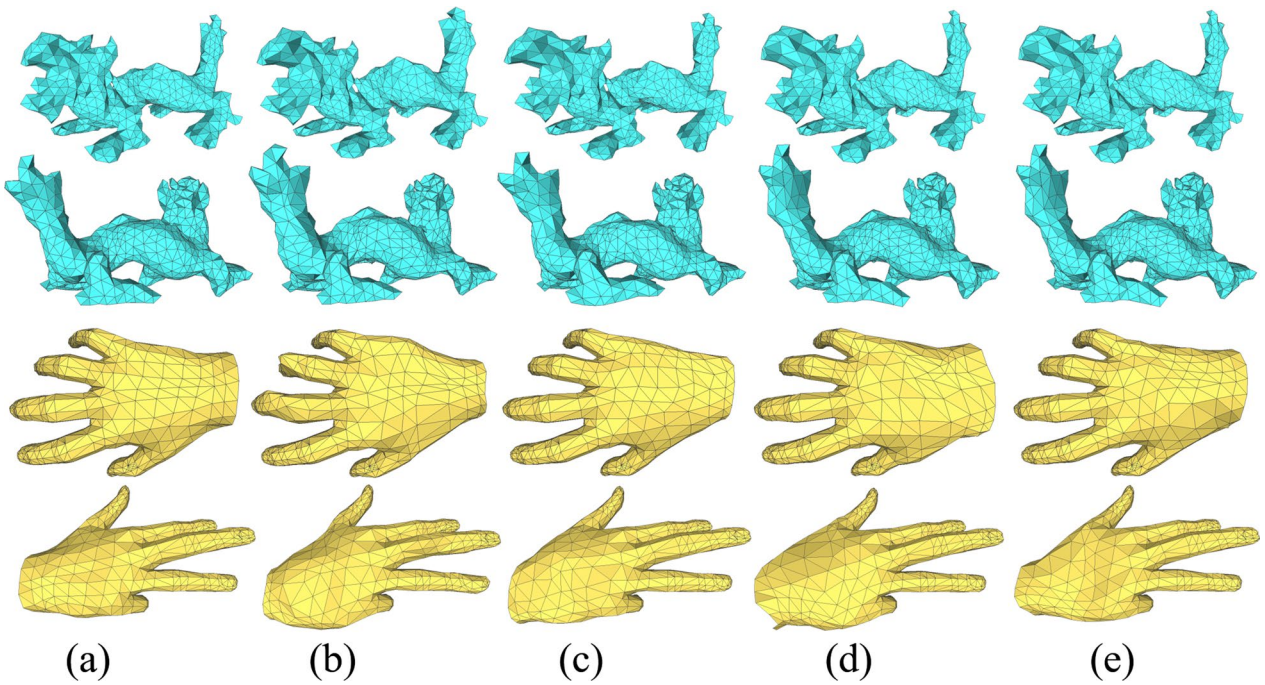

Fig. 9 Visual quality on remeshed Asiadragon (top two rows) and remeshed Hand (bottom two rows) dataset. **a** Original; **b** Model; **c** Model†; **d** Pre-Model; **e** Pre-Model†. † denotes training without the attack layers, Model represents the model trained on the remeshed dataset, and Pre-Model represents the model trained on the original dataset

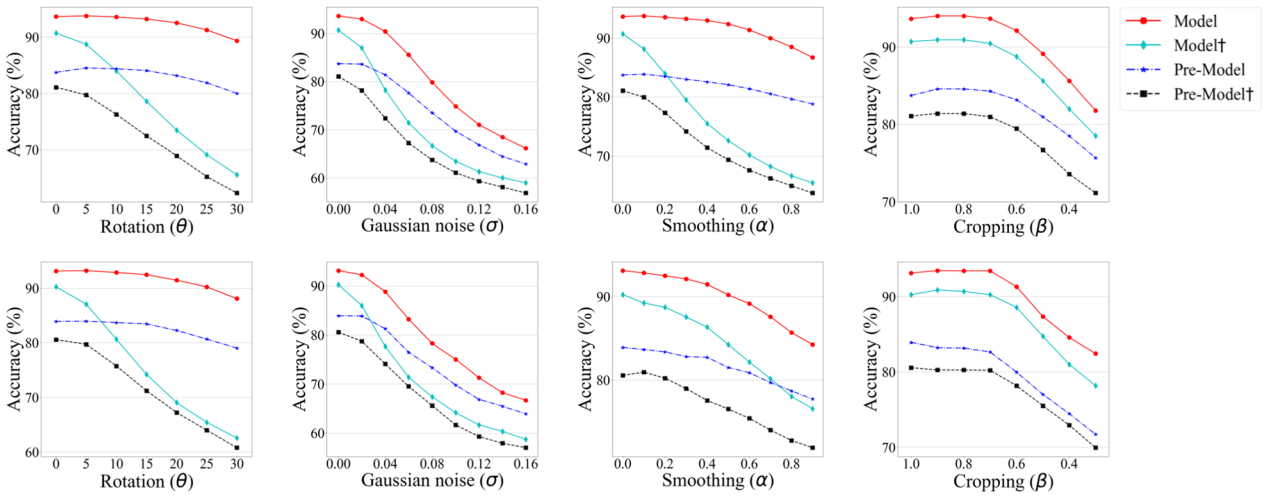

Fig. 10 Bit accuracy rate (%) under different attacks and intensities on remeshed Asiadragon (top) and remeshed Hand (bottom) dataset. Model (red), Model† (cyan), Pre-Model (blue) and Pre-Model† (black) are showed. † denotes training without the attack layers, Model represents the model trained on the remeshed dataset, and Pre-Model represents the model trained on the original dataset

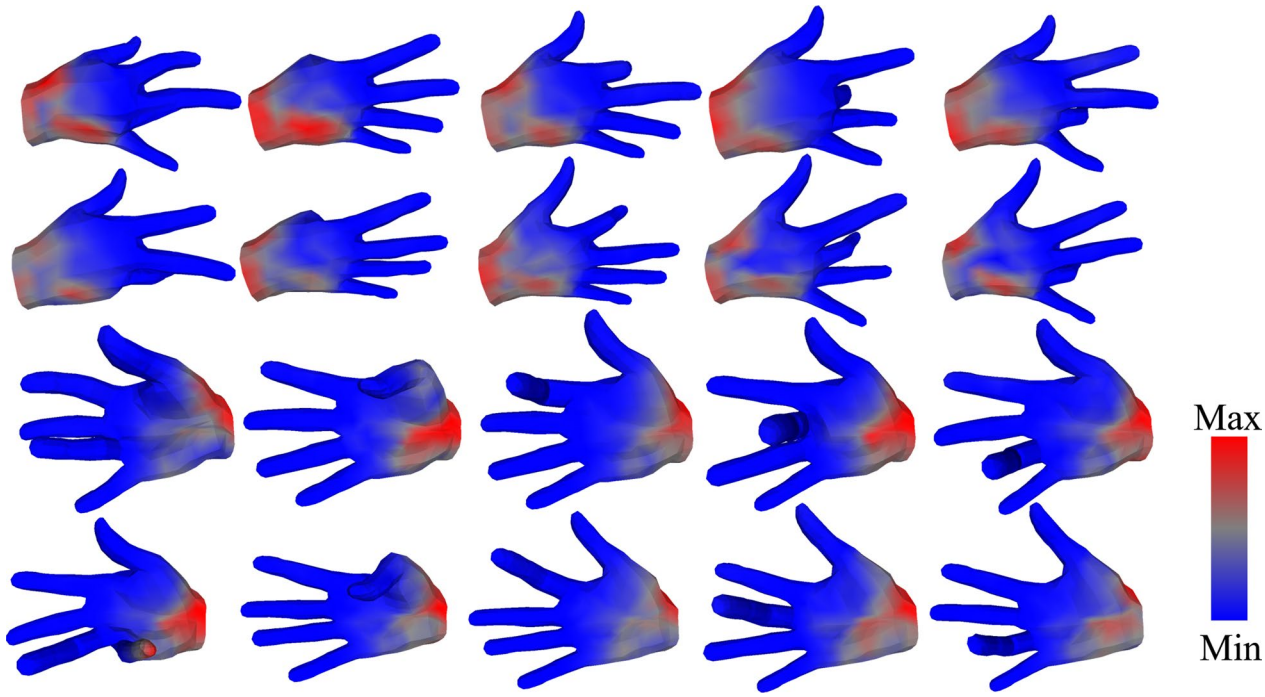

Fig. 11 Colormaps of per vertex Euclidean modication produced on Hand dataset

Limitations

Our experiments are limited in the digital domain and are conducted with several common attacks. To better evaluate the proposed method, we need conduct the experiments in real-world scenarios, such as 3D printing-scanning process (Hou et al. 2017) and 3D-to-2D process (Yoo et al. 2021). In the future, we will extend our research to these scenarios.

Conclusion

In this paper, we propose the first deep learning-based method for the robust 3D mesh watermarking task. We propose a novel end-to-end 3D mesh watermarking network, which can solve this task without manually designing algorithms. Attack layers can improve the robustness against corresponding attacks. In real applications, we can adaptively adjust our attack layers

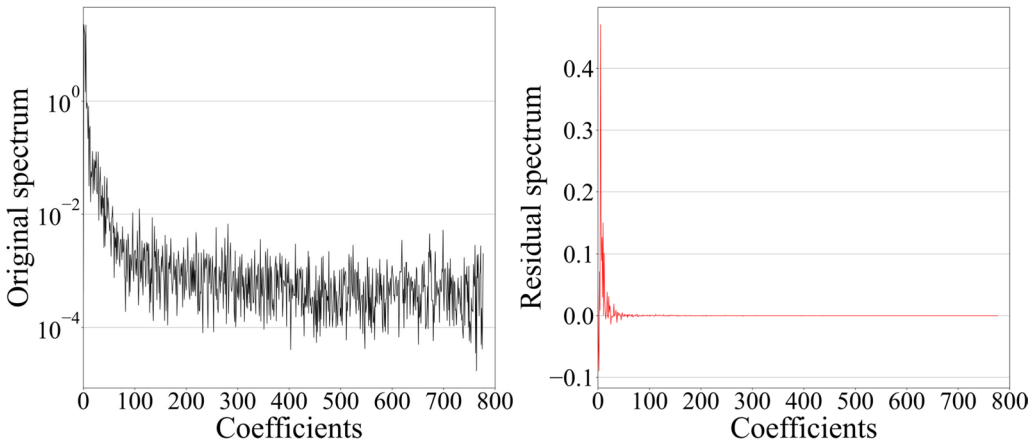

Fig. 12 The original power spectrum (left) and the residual power spectrum (right). Coefficients are ordered with respect to the corresponding eigenvalues of the Laplace-Beltrami operator

to meet the actual robustness requirement. For visual quality, we design the curvature consistency loss function to guarantee the surface smoothness. Extensive experiments demonstrate the effectiveness of our framework and the superior performance of the proposed method.

Acknowledgements
We would like to thank Xiaojuan Dong for her code for baseline methods, and Xi Yang for his suggestion about the design of loss function.

Author contributions
The design of the proposed method, the experiment deployment and the draft of the manuscript: FW and HZ. Revising the manuscript critically for important intellectual content: HF, WZ and NY. All authors read and approved the final manuscript.

Funding
This work was supported in part by the Natural Science Foundation of China under Grant 62072421, 62002334, 62102386, 62121002 and U20B2047, Anhui Science Foundation of China under Grant 2008085QF296, Exploration Fund Project of University of Science and Technology of China under Grant YD3480002001, and by Fundamental Research Funds for the Central Universities WK5290000001.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]University of Science and Technology of China, Hefei, China. [2]Simon Fraser University, Vancouver, Canada. [3]National University of Singapore, Kent Ridge, Singapore.

Received: 31 January 2022 Accepted: 17 April 2022
Published online: 02 December 2022

References

Bors AG, Luo M (2012) Optimized 3d watermarking for minimal surface distortion. IEEE Trans Image Process 22(5):1822–1835
Cayre F, Rondao-Alface P, Schmitt F, Macq B, Maitre H (2003) Application of spectral decomposition to compression and watermarking of 3d triangle mesh geometry. Signal Process Image Commun 18(4):309–319
Chen X, Golovinskiy A, Funkhouser T (2009) A benchmark for 3d mesh segmentation. ACM Trans Graph 28(3):1–12
Cho J-W, Prost R, Jung H-Y (2007) An oblivious watermarking for 3-D polygonal meshes using distribution of vertex norms. IEEE Trans Signal Process 55(1):142–155
Dawson-Haggerty et al. Trimesh. https://trimsh.org/
Feng Y, Feng Y, You H, Zhao X, Gao Y (2019) Meshnet: mesh neural network for 3d shape representation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 33, pp 8279–8286
Gao Z, Yan J, Zhai G, Zhang J, Yang Y, Yang X (2021) Learning local neighboring structure for robust 3d shape representation. In: Proceedings of the AAAI conference on artificial intelligence, vol 35, pp 1397–1405
Garland M, Heckbert PS (1997) Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on computer graphics and interactive techniques, pp 209–216
Gong S, Chen L, Bronstein M, Zafeiriou S (2019) Spiralnet++: a fast and highly efficient mesh convolution operator. In: Proceedings of the IEEE/CVF international conference on computer vision workshops, pp 4141–4148
Hamidi M, El Haziti M, Cherifi H, Aboutajdine D (2017) A robust blind 3-d mesh watermarking based on wavelet transform for copyright protection. In: 2017 international conference on Advanced Technologies for Signal and Image Processing (ATSIP). IEEE, pp 1–6
Hanocka R, Hertz A, Fish N, Giryes R, Fleishman S, Cohen-Or D (2019) Meshcnn: a network with an edge. ACM Trans Graph 38(4):1–12
Hanocka R, Metzer G, Giryes R, Cohen-Or D (2020) Point2mesh: a self-prior for deformable meshes. ACM Trans Graph 39(4):126
He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
Hou J-U, Kim D-G, Lee H-K (2017) Blind 3d mesh watermarking for 3d printed model by analyzing layering artifact. IEEE Trans Inf Forensics Secur 12(11):2712–2725
Hu S-M, Liu Z-N, Guo M-H, Cai J-X, Huang J, Mu T-J, Martin RR (2021) Subdivision-based mesh convolution networks. arXiv:2106.02285
Jang H-U, Choi H-Y, Son J, Kim D, Hou J-U, Choi S, Lee H-K (2018) Cropping-resilient 3d mesh watermarking based on consistent segmentation and mesh steganalysis. Multimed Tools Appl 77(5):5685–5712
Jia Z, Fang H, Zhang W (2021) Mbrs: enhancing robustness of DNN-based watermarking by mini-batch of real and simulated jpeg compression. In: Proceedings of the 29th ACM international conference on multimedia, pp 41–49

Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: International conference on learning representations
Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: International conference on learning representations
Lee J-S, Liu C, Chen Y-C, Hung W-C, Li B (2021) Robust 3d mesh zero-watermarking based on spherical coordinate and skewness measurement. Multimed Tools Appl 80(17):25757–25772
Liu J, Wang Y, Li Y, Liu R, Chen J (2017) A robust and blind 3d watermarking algorithm using multiresolution adaptive parameterization of surface. Neurocomputing 237:304–315
Liu K, Chen D, Liao J, Zhang W, Zhou H, Zhang J, Zhou W, Yu N (2021) Jpeg robust invertible grayscale. IEEE Trans Vis Comput Graph. https://doi.org/10.1109/TVCG.2021.3088531
Luo X, Zhan R, Chang H, Yang F, Milanfar P (2020) Distortion agnostic deep watermarking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13548–13557
Mariani G, Cosmo L, Bronstein AM, Rodolà E (2020) Generating adversarial surfaces via band-limited perturbations. In: Computer Graphics Forum, vol 39. Wiley Online Library, pp 253–264
Milano F, Loquercio A, Rosinol A, Scaramuzza D, Carlone L (2020) Primal-dual mesh convolutional neural networks. Adv Neural Inf Process Syst 33:952–963
Rolland-Neviere X, Doerr G, Alliez P (2014) Triangle surface mesh watermarking based on a constrained optimization framework. IEEE Trans Inf Forensics Secur 9(9):1491–1501
Romero J, Tzionas D, Black MJ (2017) Embodied hands: modeling and capturing hands and bodies together. ACM Trans Graph 36(6):1–17
Stanford: The Stanford 3D Scanning Repository (2021). http://graphics.stanford.edu/data/. Accessed 6 Aug 2021
Tancik M, Mildenhall B, Ng R (2020) Stegastamp: invisible hyperlinks in physical photographs. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2117–2126
Taubin G (2000) Geometric signal processing on polygonal meshes. In: 21st annual conference of the European Association for Computer Graphics, Eurographics
Torkhani F, Wang K, Chassery J (2012) A curvature tensor distance for mesh visual quality assessment. In: Bolc L, Tadeusiewicz R, Chmielewski LJ, Wojciechowski KW (eds) Computer vision and graphics—international conference, vol 7594, pp 253–263
Uccheddu F, Corsini M, Barni M (2004) Wavelet-based blind watermarking of 3d models. In: Proceedings of the 6th workshop on multimedia & security, pp 143–154
Vasic B, Vasic B (2013) Simplification resilient LDPC-coded sparse-QIM watermarking for 3D-meshes. IEEE Trans Multimed 15(7):1532–1542
Verma N, Boukhayma A, Verbeek J, Boyer E (2021) Dualconv: dual mesh convolutional networks for shape correspondence. arXiv: 2103.12459
Wang K, Lavoué G, Denis F, Baskurt A (2008) Hierarchical watermarking of semiregular meshes based on wavelet transform. IEEE Trans Inf Forensics Secur 3(4):620–634
Wang K, Lavoué G, Denis F, Baskurt A (2008) A comprehensive survey on three-dimensional mesh watermarking. IEEE Trans Multimed 10(8):1513–1527
Wang R, Han S, Zhang P, Yue M, Cheng Z, Zhang Y (2020) A novel zero-watermarking scheme based on variable parameter chaotic mapping in NSPD-DCT domain. IEEE Access 8:182391–182411
Wang K, Lavoué G, Denis F, Baskurt A, He X (2010) A benchmark for 3d mesh watermarking. In: Shape Modeling International Conference. IEEE, pp 231–235
Wang N, Zhang Y, Li Z, Fu Y, Liu W, Jiang Y-G (2018) Pixel2mesh: generating 3d mesh models from single rgb images. In: Proceedings of the European Conference on Computer Vision, pp 52–67
Wengrowski E, Dana K (2019) Light field messaging with deep photographic steganography. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1515–1524
Yoo I, Chang H, Luo X, Stava O, Liu C, Milanfar P, Yang F (2021) Deep 3d-to-2d watermarking: embedding messages in 3d meshes and extracting them from 2d renderings. arxiv:2104.13450
Zhang J, Chen D, Liao J, Fang H, Ma Z, Zhang W, Hua G, Yu N (2021b) Exploring structure consistency for deep model watermarking. arXiv preprint arXiv:2108.02360
Zhang J, Chen D, Liao J, Fang H, Zhang W, Zhou W, Cui H, Yu N (2020) Model watermarking for image processing networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol 34, pp 12805–12812
Zhang J, Chen D, Liao J, Zhang W, Feng H, Hua G, Yu N (2021a) Deep model intellectual property protection via deep watermarking. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2021.3064850
Zhang C, Karjauv A, Benz P, Kweon IS (2021c) Towards robust deep hiding under non-differentiable distortions for practical blind watermarking. In: Proceedings of the 29th ACM international conference on multimedia, pp 5158–5166
Zhou H, Chen K, Zhang W, Yao Y, Yu N (2018) Distortion design for secure adaptive 3-d mesh steganography. IEEE Trans Multimed 21(6):1384–1398
Zhou Y, Wu C, Li Z, Cao C, Ye Y, Saragih J, Li H, Sheikh Y (2020) Fully convolutional mesh autoencoder using efficient spatially varying kernels. In: Advances in neural information processing systems
Zhu J, Kaplan R, Johnson J, Fei-Fei L (2018) Hidden: hiding data with deep networks. In: Proceedings of the European Conference on Computer Vision (ECCV), pp 657–672

Publisher's Note