3D mesh object watermarking

Improving robustness of feature vertex localisation by centre-of-volume

Matthijs C. van Andel[1]

Supervisor(s): Zekeriya Erkin[1]**, Devriş İşler**[2,3]

[1]EEMCS, Delft University of Technology, The Netherlands
[2]IMDEA Networks Institute
[3]Universidad Carlos III de Madrid, Spain

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 22, 2024

Name of the student: Matthijs C. van Andel
Final project course: CSE3000 Research Project
Thesis committee: Zekeriya Erkin, Devris Isler, Asterios Katsifodimos

Abstract

Digital modelling is becoming more prevalent in many applications. The underlying 3D mesh objects are therefore getting increasingly valuable, such that methods for ownership protection are required. Watermarking is a solution to this problem, yet the research combining watermarking and 3D mesh objects is limited. This work aims to improve an existing watermarking method that embeds watermarking bits into a 3D model by selecting features, or the vertices with a larger mean curvature. We propose an improvement where a centre of mass calculation is changed to a centre of volume, aiming to achieve better robustness against mesh simplification attacks on 3D mesh objects by use of this volume centre instead of a mass centre. The proposed method of applying the watermark reaches an accuracy score that is 15.78 percentage points higher on average than the original method, when attacked by mesh simplification. Therefore, this work shows a more robust approach to the method build upon.

1 Introduction

With advancements in digital modelling and 3D graphics in many applications such as virtual reality, augmented reality or medical imaging, the need for safeguards against unauthorised use or manipulation is growing quickly [6]. Watermarking plays a big role in this field [10], [7], as the aim of watermarking is to embed data into any model such that the utility of the data of that model remains equal or as high as possible. The embedding should be such that it is robust against attacks or attempts to remove this watermark, without severely degrading the utility of the original data. Robustness is defined by how easily the watermark is removed, or how much of the utility of the data has to be sacrificed before the watermark is undetectable.

Additionally, there exists a partition of watermarking algorithms that are called fragile, watermarks that break as soon as the underlying data is changed. These fragile methods are mainly used to validate that the original data has not been tampered with. The watermarking of 3D mesh objects is particularly difficult, because there is no specific ordering of the data. Perceivability is another aspect that makes watermarking difficult, because changes in the data of a model quickly result in visible alterations to the shape of a 3D mesh.

Currently, all existing watermark embedding and detection methods could be split up into two different categories; blind and informed watermarking. Blind watermarking does not require the original mesh to be able to retrieve the watermark, where informed watermarking detects the watermark only in correspondence to the original data. Blind watermarking is often much preferred, because it could save a lot of time and resources for the owner of the original data. Another benefit of the blind detection approach is that a watermark could be publicly verified.

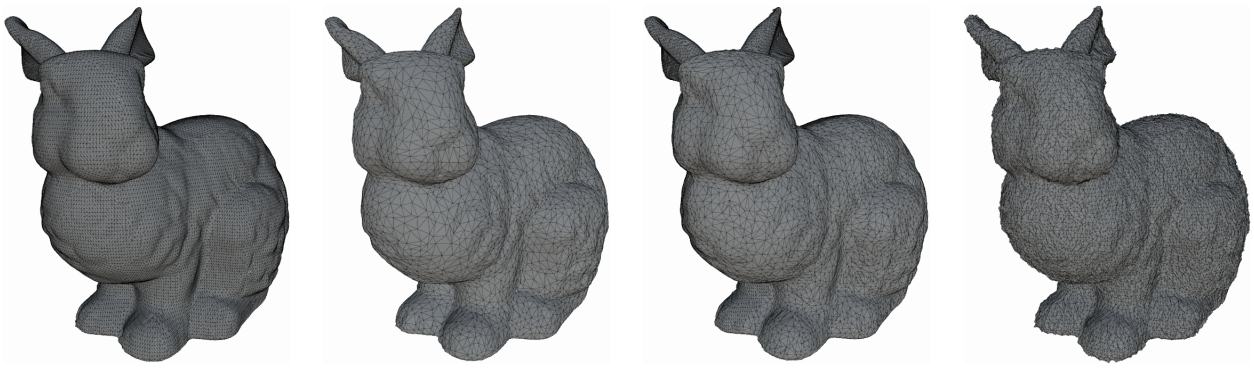
Sufficiently different watermarking techniques for 3D mesh objects exist [9], yet no standard has been achieved. In [4] an algorithm is proposed that is based on embedding the watermark in the features of a point cloud model. The idea is to transform vertices to a spherical coordinate system, to then embed the watermark by changing the azimuth angles of defined feature vertices. In this work, we propose a change to the algorithm of [4] to improve on robustness, particularly against mesh simplification attacks. Instead of weighting each vertex equally as proposed in [4] when calculating the origin of the new coordinate system, we propose to calculate the origin of this new coordinate system by computing the centre of volume of a mesh. The idea is to improve the robustness of the origin calculation not by relying on the distribution of vertices, but rather on the shape of the 3D mesh object. Other computations from [4] are not changed, however, because the weight of the mesh will now no longer rely on local details as heavily as before, mesh simplification attacks will have a different impact and the improved algorithm is noticeably more robust against mesh simplification attacks.

The remainder of this paper is divided into the sections (2) Related work, where a brief overview of other works is given, followed by section (3) Attacks on watermarked 3D mesh objects, where techniques used in this work are introduced. After that, section (4) Watermarking by feature localisation which gives a detailed explanation of the algorithm from [4] and the proposed changes. Next, section (5) Experimental setup and results lays out the practical experiments and shows their results. Afterwards, section (6) Discussion goes over the ethical aspects as well as comparing results with previous work. Finally, we draw conclusions and suggest further work in section (7) Conclusions and future work.

2 Related work

As previously mentioned, algorithms for watermarking 3D mesh objects already exist. The main body of this work considers the algorithm proposed in [4], where a mesh is split into feature and reference vertices. Based on these sets, the watermark is embedded in the feature vertices, as explained in chapter 4. The idea of using certain feature vertices is not new, as [8] already used detection of features in a mesh to apply the watermark. However, the approach in [8] is to watermark the selected feature vertices in the spectral domain, where [4] uses a new spherical coordinate system.

Within the field of watermarking 3D mesh objects, many algorithms exist [9], though a standard has not yet been



(a) The Stanford Bunny [2] without attacks.

(b) Bunny attacked by mesh simplification.

(c) Bunny attacked by non-uniform mesh simplification.

(d) Bunny attacked by additive noise.

Figure 1: Original mesh and attacked variants.

set. Next to different approaches, these algorithms fall into different categories. Modern approaches to 3D mesh watermarking attempts to retrieve the watermark in a blind fashion, meaning that the original data is not required at the extraction phase [1]-[8], [10].
Additionally, watermarking algorithms that break the watermark when the original data has been changed are also proposed [6], with the purpose of detecting unauthorised manipulation, instead of ownership detection. In [6] the idea of the watermark is to be reversible, such that the high-detail original mesh could be used. The purpose of the robustness of the algorithms proposed in [3], [4], [8] and [10] are rather to detect original ownership of the underlying model. Others apply these watermarking algorithms to 3D-printing [1] and scanning, to be able to detect ownership of the digital model that was 3D-printed.

3 Attacks on watermarked 3D mesh objects

This chapter introduces various attacks possible on 3D mesh watermarks, such as mesh simplification. Examples of these attacks are shown in Figure 1.

3.1 Mesh simplification

Simplification of a mesh is achieved by creating a new mesh that retains the same shape as its original, whilst having a percentage of its vertices removed. An example of a simplified mesh is seen in Figure 1b. The edges to these vertices are replaced, such that the resulting mesh has no gaps or other errors. Important to note here, when mesh simplification is applied with a high attack strength, the usability of the mesh decreases rapidly. Therefore, the aim when attempting to design a watermarking algorithm is to design the algorithm such that it is robust against up to a decided point of strength.

Another approach to this attack is to perform the attack non-uniformly. That is, a mesh is cut into two or more parts, each of them attacked by the simplification algorithm with varying strengths. The results are then added back together. This approach is shown in 1c, where half of the bunny has been simplified, whilst the other half remains untouched.

3.2 Additive noise

Another method of disturbing the original data structure of a mesh is by adding random noise to the vertex positions. This attack is performed by applying formula 1:

$$V_i = V_i + d_r \cdot \vec{r} \tag{1}$$

where V_i represents vertex i of the set V, containing all the vertices of a mesh. The vector $\vec{r}$ represents a random vector with length 1 and the value d_r represents a random value in the range of zero to d, where d represents the maximum length a vertex could be translated, therefore representing the strength of the noise attack.

Additive noise is a very destructive attack because of the fluctuation in the position in neighbouring vertices, as seen in 1d. Consequentially, the attack is easily perceived by the human eye. Meshes attacked by the additive noise attack are disturbed on every vertex of the mesh, creating much rougher surface areas. The severity of the attack should therefore be limited when comparing the robustness of a watermark, as the data model would quickly lose utility. Data without utility is not regarded important to protect by watermarking, since its value has been drastically decreased.

4 Watermarking by feature localisation

This chapter introduces the details of the algorithm proposed in [4], next to introducing the proposed changes to improve on this algorithm.

4.1 The watermarking algorithm

This paper is based on an algorithm to watermark a 3D point cloud [4]. The algorithm in [4] is based on feature vertex localisation, meaning that the most important vertices of a mesh are the ones possibly carrying the watermarking information. Each vertex is either a feature vertex or a reference vertex. The watermarking algorithm consists of the following steps: feature vertex localisation, creating a new coordinate system and embedding the watermark data.

Feature vertex localisation

The main idea of [4] is to change the azimuth angle of feature vertices, after they have been transformed to a spherical domain, which is calculated using the reference vertices. The azimuth angle of a position is the rotation on the horizon with respect to the origin, where the horizon is often the y-axis.

The first step in achieving the watermark is locating these feature vertices. To calculate this, a bumpy performance S is introduced. This measures the roughness of an area by computing a score using the normal vector of a vertex V_i, and the normal vectors of all the vertices with an edge to V_i. The normal vector $\vec{n_i}$ for vertex V_i is calculated by taking the mean of all the directions to its neighbours. This can be seen in formula 2:

$$\vec{n_i} = \frac{\sum_{j=1}^{N_i}(\vec{V_i} - \vec{V_j})}{N_i} \tag{2}$$

After that, the bumpy performance S is calculated using formula 3:

$$S(V_i) = \sum_{j=1}^{N_i}(\vec{n_i} \cdot \vec{n_j}) = \sum_{j=1}^{N_i}(x_i x_j + y_i y_j + z_i z_j) \tag{3}$$

where the sum of all dot products between normal vectors of neighbouring vertices is taken. The reference of (x_i, y_i, z_i) represent the x, y and z components of the normal vector $\vec{n_i}$. The idea of formula 3 is to project normal vectors on their neighbours to measure the bumpy performance, where a lower value means higher change in angle. Using a threshold T, all vertices are split into two sets; all vertices with a bumpy performance smaller or equal to T are labeled feature vertices stored in S_f. The other vertices form the reference vertices, S_r. The set of feature vertices S_f will carry the watermark information, where the reference vertices S_r will be used to create a new coordinate system. Given that the watermark capacity is M, the vertices will be divided such that there are M bins. The capacity of a watermark denotes how many bits of data could be embedded into the model. The threshold T is then chosen such that there are no shallow bins, that is, each bin contains at least two feature vertices.

Creating a spherical coordinate system

For 3D mesh objects, the order of vertices is not consistent through models. Therefore, [4] proposes a new spherical coordinate system in order for data to have the same order between the process of watermark embedding and extraction. In the last step of the algorithm, this new coordinate system is used to embed watermarking bits.

The first step into creating this new coordinate system is to establish the new origin. This is done through the calculation of the centre of mass of the reference set S_r, using:

$$V_o = \frac{1}{N_r}\sum_{i}^{N_r} V_i \tag{4}$$

where V_o represents the new origin coordinates, N_r denotes the length of the reference set S_r, and V_i with $(i = 1, 2, ..., N_r)$ are vertices in S_r.

With the new origin established, we can now create the new axes by creating a covariance matrix Cov and computing the eigenvalue decomposition of matrix Cov:

$$Cov = \begin{pmatrix} \sum_{i=0}^{N_r} x_i^2 & \sum_{i=0}^{N_r} x_i y_i & \sum_{i=0}^{N_r} x_i z_i \\ \sum_{i=0}^{N_r} x_i y_i & \sum_{i=0}^{N_r} y_i^2 & \sum_{i=0}^{N_r} y_i z_i \\ \sum_{i=0}^{N_r} x_i z_i & \sum_{i=0}^{N_r} y_i z_i & \sum_{i=0}^{N_r} z_i^2 \end{pmatrix} \tag{5}$$

with the eigenvalue decomposition of Cov given as:

$$Cov = UHU^T = U(diag(h1, h2, h3))U^T \tag{6}$$

where H denotes the diagonal matrix that consists of eigenvalues sorted in a descending order, or more formally: $h1 > h2 > h3$. The matrix U represents the accompanying orthonormal matrix, such that the i^{th} column of U denotes the i^{th} eigenvector, and corresponds to eigenvalue h_i of matrix H. The three columns of U are the directions of u, v and n for the first, second and third column respectively. The directions u, v and n represent the new axes of the spherical coordinate system, and hold a rotation with respect to the x, y and z axes of the original Cartesian coordinate system. To compute the new coordinates of a vertex in the new coordinate system, matrix $M_{c\to s}$ is created, representing a rotation and translation matrix that yield a transformation from Cartesian coordinates to the new axes of u, v and n:

$$M_{c\to s} = \begin{pmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ n_x & n_y & n_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -V_{ox} \\ 0 & 1 & 0 & -V_{oy} \\ 0 & 0 & 1 & -V_{oz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{7}$$

where V_o represents the established origin. After applying formula 7 the vertices are converted to spherical coordinates. In [4] the spherical coordinates are

represented as $(\rho_i, \phi_i, \theta_i)$, denoting a conversion from Cartesian components (V_{iu}, V_{iv}, V_{in}) of vertex V_i, transformed using formula 7. The spherical coordinates are calculated with:

$$\begin{aligned} V_s &= (\rho_i, \phi_i, \theta_i) \\ &= (\sqrt{V_{iu}^2 + V_{iv}^2 + V_{in}^2}, \arctan(\frac{V_{iv}}{V_{iu}}), \\ &\quad \arccos(V_{in}/\rho_i)) \end{aligned} \tag{8}$$

where V_s represents the resulting spherical coordinates of transformed vector V_i.

The last step before embedding the watermark data into these feature vertices is to apply a normalisation to the ρ component of the feature vertices, making the effect of the watermark less perceivable. Additionally, normalisation of the ρ component allows for robustness against scaling attacks. The ρ_i component of feature vertex i is normalised according to formula 9, whilst retaining the other components:

$$\rho_{ni} = \frac{\rho_i - \rho_{min}}{\rho_{max} - \rho_{min}} \tag{9}$$

where ρ_{ni} represents the normalised ρ component, and ρ_{min} and ρ_{max} represent the minimum and the maximum ρ components of all feature vertices before normalising.

Embedding the watermark

With the spherical coordinates computed, the model is divided into M bins with equal width. These bins are filled based on the θ_i angle of V_{si}. To compute the feature vertices of a single bin, we use the watermark capacity to retrieve the boundaries of a bin:

$$R_m = \{\theta_m | (m-1) \cdot \frac{360}{M} \leq \theta \leq m \cdot \frac{360}{M}\} \quad \text{with } m \leq M \tag{10}$$

where R_m represents the vertices contained in bin m, θ denotes the azimuth angle of a vertex. It is these azimuth angles that will be changed when embedding the watermark bits, one per bin, while leaving the other components unchanged.

Each bin represents one bit of the watermark data. The main idea of this step, proposed in [4] is to change the angle of the feature vertices by half a bin, as long as they stay in the same bin. This forms bins where most feature vertices are either in the upper or lower half of the bin. Whether the upper or lower half is selected, depends on the bit to embed, as seen in formula 11:

$$\theta_{mi} = \begin{cases} \theta_{mi} + \Delta, \text{ if } w_m = 1 \\ \theta_{mi} - \Delta, \text{ if } w_m = 0 \end{cases} \tag{11}$$

where δ represents half of the width of a bin, or $\Delta = 360/2M$. The azimuth angle of the i^{th} feature vertex in bin m is represented by $\theta_m i$, the bit to embed is denoted by w_m. Vertices that are carrying watermark information should not have their azimuth angle changed such that it no longer is contained in the same bin. Therefore, the azimuth angle is only changed if a vertex meets the requirements shown in formula 12:

$$\begin{aligned} &\text{if } w_m = 1, \quad (m-1) \cdot \alpha \leq \theta_i < \left(m - \frac{1}{2}\right) \cdot \alpha \\ &\text{if } w_m = 0, \quad \left(m - \frac{1}{2}\right) \cdot \alpha \leq \theta_i < m \cdot \alpha \end{aligned} \tag{12}$$

where α represents the width of a bin. When formula 12 and formula 11 are combined, the resulting bin will have either the upper or the lower region filled with vertices, because of the updated azimuth angles. The transformations are now applied in an inverse manner, to retrieve the watermarked mesh.

Retrieving the watermark

The retrieving of the algorithm of [4] is blind. This means that the original data is not necessary to be able to prove ownership. The first steps of the extraction are very similar to watermarking a 3D mesh, where the bumpy performance S is calculated and vertices are split into reference and feature sets S_r and S_f respectively. Then, as in the embedding phase, a new coordinate system is introduced, including the calculation of a new origin. Afterwards, all the feature vertices are transformed, and separated into bins. The difference between embedding and retrieving the watermark lies in the step after creation of the bins. Within each bin, the amount of vertices in the lower and upper half are summed. Then, formula 13 is used to extract the watermarking bit embedded in a bin:

$$w'_m = \begin{cases} 0, \text{ if } sum0 > sum1 \\ 1, \text{ if } sum0 \leq sum1 \end{cases} \tag{13}$$

where $sum0$ and $sum1$ represent the amount of feature vertices in the lower and upper half of the bin, respectively.

4.2 Centre of volume

In the algorithm explained in the previous section, the new coordinate system is created around an origin point, calculated with the centre of mass. This however may be vulnerable to manipulation, for example by duplicating a vertex. Since the origin point in [4] is regarded as the average point of all the vertices, addition of points such that the shape of the model does not change will influence the origin point, which quickly degrades the quality of the watermark embedded into the model. Therefore, we propose an alternative method for calculating this origin point, using centre of volume. The idea is to calculate a new origin that is at the center of the shape of the 2D representation of the model, regardless of the density

(a) Bunny. (b) Armadillo. (c) Dragon.

Figure 2: 3D mesh objects experimented on, shown as wireframes.

of vertices in a certain area. This approach will disregard duplicated points, and will only take into account the added or decreased volume when vertices are manipulated. Since the spherical coordinate system is established using reference set S_r in [4], the volume centre is computed with the same set in this work.

Table 1: Model Information.

Model Name	Vertex count	Triangle count
Bunny	34.900	208.353
Armadillo	173.006	1.037.832
Dragon	438.135	2.614.242

Creating the 2D hull

Computing the 2D hull of a mesh is done through the Quickhull algorithm [5]. The main idea behind this method is to select the furthest points from the current hull, labeling them as hull points, after which all vertices within the hull are disregarded. This process is recursively repeated until no vertex is left. The result is a set of points P_t which describe the smallest 2D hull such that all vertices of the mesh model are included. Within this algorithm, the z-value of the vertices are not taken into account, since we calculate a 2D hull to compute the new origin.

The first step in computing the 2D hull is selecting the minimum p_{min} and maximum p_{max} points, where the coordinate has the minimum or maximum value on the x axis:

$$\begin{aligned} min &= \min_{(x_i,y_i,z_i)\in S_r} x_i \\ max &= \max_{(x_i,y_i,z_i)\in S_r} x_i \end{aligned} \tag{14}$$

and labelling them as hull points by adding them to P. A line L_p is constructed such that the starting point is p_{min} and the endpoint is p_{max}. All remaining vertices are now split into two partition sets P_l and P_r, which represent all the vertices to the left or right of the line respectively. Afterwards, in both partitions the point with the highest distance to the line on the x, y plane is selected and added to the hull, p_{Lmax} and p_{Rmax} for P_l and P_r respectively. The computation continues recursively, as a new line is constructed between the latest two points added to P and the newly found maximum distance point. When there are no more vertices outside of the shape constructed from P, the 2D hull is complete.

Computing the centroid

The centroid of the 2D hull is used to compute the origin point for the coordinate system. The centroid C is computed by calculating the signed area, a weighted mass centre method. We first calculate a partial signed area A_{si} for every vertex i:

$$A_{si} = x_i \cdot y_{i+1} - x_{i+1} \cdot y_i \tag{15}$$

where the vertices from P are represented as $\{P_0(x_0, y_0), P_1(x_1, y_1), \ldots, P_n(x_n, y_n)\}$, taking P_n as the last vertex in the set. Next, the total signed area A_i is calculated by taking the sum of all the partial signed area values:

$$A_i = \frac{1}{2} \cdot \sum_{i=0}^{n} A_{si} \tag{16}$$

where A_{si} denotes the partial signed area per vertex. Finally, the coordinates C_x and C_y are computed with formula 17:

$$\begin{aligned} C_x &= \frac{1}{6 \cdot A_i} \cdot \sum_{i=0}^{n} (x_i + x_{i+1}) \cdot A_{si} \\ C_y &= \frac{1}{6 \cdot A_i} \cdot \sum_{i=0}^{n} (y_i + y_{i+1}) \cdot A_{si} \end{aligned} \tag{17}$$

where the weighted average is taken by using the partial signed area computed in formula 15. Note that indices are cyclic, meaning that P_{n+1} means the same as P_0.

Table 2: Watermark extraction accuracy a.

	Bunny		Armadillo		Dragon		Average	
	Mass	Volume	Mass	Volume	Mass	Volume	Mass	Volume
Baseline	90.63	97.27	100.00	100.00	99.61	100.00	**96.75**	**99.09**
Additive noise								
d = 0.001	87.62	97.62	100.00	100.00	99.42	99.96	**95.68**	**99.19**
d = 0.01	64.67	92.77	99.9	99.82	81.78	97.79	**82.12**	**96.79**
d = 0.05	53.40	75.74	90.63	89.88	75.57	81.43	**73.20**	**82.35**
d = 0.1	50.37	67.73	73.24	72.40	67.05	69.10	**63.55**	**69.74**
d = 0.2	50.43	56.60	56.35	54.53	56.76	58.71	**54.51**	**56.61**
Mesh simplification								
5%	62.11	78.91	92.97	97.27	75.00	99.61	**76.69**	**91.93**
10%	49.22	66.80	96.09	94.53	69.14	98.44	**71.48**	**86.59**
15%	49.61	72.27	91.8	92.58	68.75	99.22	**70.05**	**88.02**
20%	56.64	58.98	89.45	89.45	66.02	97.27	**70.70**	**81.90**
30%	57.42	69.14	74.22	84.77	58.2	94.14	**63.28**	**82.68**

Because of the use of weighted average, the risk of manipulation by the attacked model is minimised, as this method is not dependent on the amount of vertices in the original model. The average complexity of this method is $\theta(nlog(n))$, which applies if most partitions are balanced, meaning that the set of vertices considered is split roughly into two equal length sets. The worst case complexity is $O(n^2)$, which occurs when all partitions are highly unbalanced, meaning that a recurrence equation of $T(n) = T(n-1)+O(n) = T(n-1)+cn$ represents the balancing of vertices, such that only one vertex is assigned to one partition, and all other vertices are assigned into the second partition.

5 Experimental Setup and Results

To measure the effectiveness of the proposed change, multiple tests will be executed. The results will then be displayed, comparing the algorithm of [4] with the centre of mass approach with our centre of volume technique. The models used in these experiments are from the Stanford Computer Graphics Laboratory [2], as seen in Figure 2, from left to right: *Bunny*, *Armadillo*, *Dragon*. The *Bunny* model has the least vertices and the *Dragon* model has the most, as seen in Table 1. For comparability with [4], a watermark with a capacity of 256 bits was embedded with each experiment.

5.1 Hardware setup

All experiments shown in this work have been executed on the same machine. This computer contains an AMD Ryzen 7 5800H CPU at 3.2GHz, with 16 cores and 16GB of RAM. A video card is installed; NVIDIA GeForce RTX 3060 Laptop GPU, but all code execution was done on the CPU. The programming language used for all experiments is C#, and the operating system of this machine is Windows 11. All models experimented on come from the Stanford 3D Scanning Repository [2].

5.2 Robustness comparison

To measure the robustness of each method objectively, the extracted watermark bits are compared to the embedded watermark bits using an accuracy score a. This score is denoted by the percentage of bits that are correctly extracted from the watermarked mesh. This allows for meaningful and comparable results. These experiments were limited in strength, because attacks even stronger than currently experimented with would degrade the models' utility beyond worthwhile protection.

Table 3: Spherical origin on Bunny model.

Origin method (and simplification percentage)	Origin point (Cartesian)
Mass embedding	$(1.45, 6.08 - 0.48)$
Mass extraction (0%)	$(1.46, 6.05, -0.48)$
Mass extraction (30%)	$(1.56, 5.93, -0.56)$
Volume embedding	$(1.24, 6.72, 0.00)$
Volume extraction (0%)	$(1.23, 6.72, 0.00)$
Volume extraction (30%)	$(1.23, 6.72, 0.00)$

5.3 Results

When executing the experiments, for all models the approach detailed in Section 4 is applied. In summary, normal vectors were calculated by comparing the directions of the vertices in the neighbourhood of each vertex. Next, using these normal vectors, bumpy performance values were calculated. Based on these performance values and a threshold value, the vertices were split into

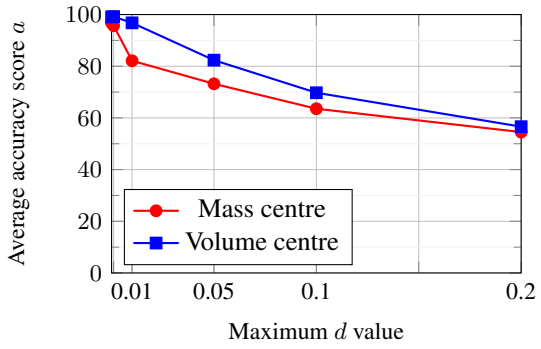


Figure 3: Accuracy against additive noise attacks.

a feature set and a reference set, selecting the features of a model. A new coordinate system was established based on the reference set, computing a new origin to then combine the new origin with a rotation matrix, consisting of the new axes of the established coordinate system. All feature vertices were then transformed by the previously computed matrices, and divided into bins. These bins were created by dividing a sphere into bins, such that there is a bin for each watermark bit to be embedded. Afterwards, the feature vertices in each bin had their azimuth angles slightly changed to embed the watermark data, before transforming all feature vertices back to the original Cartesian coordinate system.
The result is a watermarked 3D mesh object. This process was applied to each model twice, once with the origin point of the new coordinate system calculated by centre of mass as proposed in [4], once with a volume centre computation.

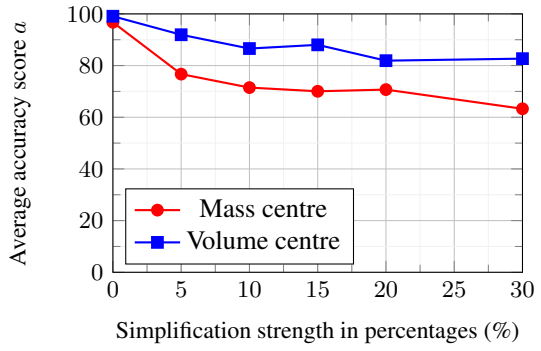


Figure 4: Accuracy against mesh simplification attacks.

Some of the watermarked mesh objects were then subjected to attempts to disturb or destroy the watermark, before attempting a blind extraction of the watermark. As a baseline, some watermarked mesh objects were not attacked in any way before attempting the blind extraction. The results of all experiments are seen in Table 2, as well as a reference to the method used for calculating the origin point of the spherical coordinate system. Because the additive noise attack utilises random values for the distance d, these experiments have been executed 20 times, with their average displayed in Table 2. An overview of the average scores for additive noise and mesh simplification are given in Figure 3 and Figure 4 respectively. The values in the Table represent the accuracy of the extracted watermark compared to the embedded data, in percentages.
Analysing the results of Table 2, it can be seen that the proposed method does not necessarily yield a better accuracy score than its counterpart for every experiment. However, when comparing the average scores, it can be noted that all average scores but one are higher when embedding and extracting the watermark with a volume centre, instead of a mass centre. Our analysis shows that the robustness against additive noise is 80.94% average using a volume centre, over 73.81% when using a mass centre.

The greatest difference in robustness can be found when a mesh is attacked by mesh simplification. In this experiment, the deterioration of the watermark grows faster when using a mass centre, compared to using a volume centre. The gap in performance is noticeable, with an average score of 86.23% when applying the watermark from 3D mesh objects that had been attacked by this simplification using a volume centre against an average score of 70.44% when applying the watermarking process with a mass centre. The trend of the accuracy score is compared in Figure 4, where the difference of the two approaches is highlighted again.
Additionally, watermarked meshes have been attacked by the non-uniform mesh simplification attack twice. Once, only the front of the model, or the vertices in the positive z-axis, were taken into account when applying the simplification. Afterwards, the attack was performed on the back of the model, or the vertices in the negative z-axis. The non-uniform mesh simplification attack proves to degrade the accuracy of the watermark rapidly, regardless of the method by which the origin of the spherical coordinate system was calculated. However, it must be noted that the non-uniform version of the mesh simplification attack weighs much heavier on the utility of the data, since only half of the model was simplified, leading to a visible distinction between the sections of the model that were affected and those that were not. This in turn means that the attack is highly perceivable. However, results of this experiments yield no conclusion. The experiment is added as an appendix for transparency, in Appendix A.

6 Discussion

After comparing experimental results, it is clear that there is potential for this approach. Especially notable is the dragon model, for which the proposed method of

applying a centre of volume approach kept the obfuscation of a watermark to a minimum when attacked with mesh simplification, a mesh simplification attack of 30% still yields a 94.14% accuracy score when using a volume centre. Attempting an equal embedding with a centre of mass massively reduces the robustness and the ability to extract the watermark, yielding a score of only 58.20%. This is resembled in the average scores as well, where mesh simplification with the volume centre approach sees an average increase of 15.78 percent points in the average accuracy score over the approach from [4]. This is an increase in robustness of over 22%. These differences are likely because of the influence removing vertices has on the mass centre, as seen in 3. Since mesh simplification will change the distribution of vertices, the mass centre is more likely to move with it. Centre of volume however is impacted less, since the distribution of vertices is less influential with this approach.
Of note is the non-uniform mesh simplification attack, in which the watermark is quickly deteriorated. Because of the seemingly inconclusive results, these results are not taken into consideration for the analysis of the performance of the proposed approach. However, the armadillo models yields reasonable accuracy scores when watermarked with the volume centre approach. On average, the volume centre approach yields higher accuracy than its counterpart. Yet it must also be noted that the centre of mass approach yields some low accuracy scores, making the non-uniform attack a potential weakness for this approach. For transparency, the results of these experiments have been added in appendix A.

Ethical considerations

This work intends to improve on existing work, where the goal is to detect ownership of 3D data. As mentioned in section 1, watermarking has a great many applications to further improve in this field. The conclusions of this work are made purely upon the experiments documented and all experiments conducted have been documented. The results of the experiments against non-uniform mesh simplification are seemingly inconclusive. For full transparency, they have been added as an appendix; appendix A. These results are not taken into account for the full conclusion of this work, but they are taken into account when laying out further research opportunities. The code with which all experiments have been executed, as well as the code to embed and extract watermarks is found in appendix B. The models experimented with are all from Stanford [2], and open source.

7 Conclusions and Future Work

This work shows an improvement of a known algorithm to watermark 3D mesh objects [4]. The embedding of this algorithm is achieved by locating feature vertices, those vertices that make up the important areas of the shape of the model, constructing a new spherical coordinate system in which these feature vertices have the watermarking embedded and to divide the vertices in bins. Of importance is the construction of the spherical coordinate system, where the origin is calculated using mass centre as proposed in [4]. In this work, it is shown that using a volume centre calculation instead will improve the robustness of the algorithm proposed in [4]. The volume centre is calculated by computing a 2D-convex hull, using the quickhull algorithm. The centre of volume is the minimally sized outline of a set of points, such that all the points of the set are contained within the hull. The weighted centre of this hull is computed and used as the new origin for the spherical coordinate in the watermarking algorithm.
This approach proved to be more robust than the originally proposed mass centre, mainly because the volume centre approach is not as much influenced by changes of the vertex distribution of a 3D mesh object.

Future work could further investigate the use of multi-dimensional volume centre calculations, a more complex approach to possibly improve on the 2D convex hull proposed in this work. Specifically against non-uniform mesh simplification attacks in-depth work is required in an attempt to create a method that is robust against these attacks, and to further investigate the full effect of the non-uniform mesh simplification attack against this watermarking approach. Additionally, the robustness of the algorithm proposed in [4] could be improved by continued work, to further improve on the robustness of 3D mesh object watermarking.

References

[1] Arnaud Delmotte et al. “Blind 3D-Printing Watermarking Using Moment Alignment and Surface Norm Distribution”. In: *IEEE Trans. Multim.* 23 (2021), pp. 3467–3482. DOI: 10.1109/TMM.2020.3025660. URL: https://doi.org/10.1109/TMM.2020.3025660.

[2] Stanford University Computer Graphics Laboratory. *The Stanford 3D Scanning Repository*. http://graphics.stanford.edu/data/3Dscanrep/. Accessed: 30-05-2024. 1996.

[3] Jing Liu, Yajie Yang, and Douli Ma. “A Blind 3D Point Cloud Watermarking Algorithm Based on Azimuth Angle Modulation”. In: *11th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics, CISP-BMEI 2018, Beijing, China, October 13-15, 2018*. Ed. by Wei Li, Qingli Li, and Lipo Wang. IEEE, 2018, pp. 1–7. DOI: 10.1109/CISP-BMEI.2018.8633259. URL: https://doi.org/10.1109/CISP-BMEI.2018.8633259.

[4] Jing Liu et al. “A Watermarking Method for 3D Models Based on Feature Vertex Localization”. In: *IEEE Access* 6 (2018), pp. 56122–56134. DOI: 10.1109/ACCESS.2018.2872783. URL: https://doi.org/10.1109/ACCESS.2018.2872783.

[5] Ernst P. Mücke. “Quickhull: Computing Convex Hulls Quickly”. In: *Comput. Sci. Eng.* 11.5 (2009), pp. 54–57. DOI: 10.1109/MCSE.2009.136. URL: https://doi.org/10.1109/MCSE.2009.136.

[6] Fei Peng et al. “Semi-Fragile Reversible Watermarking for 3D Models Using Spherical Crown Volume Division”. In: *IEEE Trans. Circuits Syst. Video Technol.* 33.11 (2023), pp. 6531–6543. DOI: 10.1109/TCSVT.2023.3272955. URL: https://doi.org/10.1109/TCSVT.2023.3272955.

[7] Xavier Rolland-Nevière, Gwenaël J. Doërr, and Pierre Alliez. “Triangle Surface Mesh Watermarking Based on a Constrained Optimization Framework”. In: *IEEE Trans. Inf. Forensics Secur.* 9.9 (2014), pp. 1491–1501. DOI: 10.1109/TIFS.2014.2336376. URL: https://doi.org/10.1109/TIFS.2014.2336376.

[8] Patrice Rondao-Alface and Benoıt Macq. “Blind watermarking of 3D meshes using robust feature points detection”. In: *Proceedings of the 2005 International Conference on Image Processing, ICIP 2005, Genoa, Italy, September 11-14, 2005*. IEEE, 2005, pp. 693–696. DOI: 10.1109/ICIP.2005.1529845. URL: https://doi.org/10.1109/ICIP.2005.1529845.

[9] Kai Wang et al. “A Comprehensive Survey on Three-Dimensional Mesh Watermarking”. In: *IEEE Trans. Multim.* 10.8 (2008), pp. 1513–1527. DOI: 10.1109/TMM.2008.2007350. URL: https://doi.org/10.1109/TMM.2008.2007350.

[10] Stefanos Zafeiriou, Anastasios Tefas, and Ioannis Pitas. “Blind Robust Watermarking Schemes for Copyright Protection of 3D Mesh Objects”. In: *IEEE Trans. Vis. Comput. Graph.* 11.5 (2005), pp. 596–607. DOI: 10.1109/TVCG.2005.71. URL: https://doi.org/10.1109/TVCG.2005.71.

A Non-uniform mesh simplification

The results of the watermark extraction accuracy after non-uniform mesh simplification attacks are found in Table 4.

B Source code

The source code of this work can be found at:
https://gitlab.ewi.tudelft.nl/cse3000/2023-2024-q4/Erkin/matthijsvanand-DatasetDatabase-Watermarking/

Table 4: Watermark extraction accuracy a against non-uniform mesh simplification.

	Bunny		Armadillo		Dragon		Average	
	Mass	Volume	Mass	Volume	Mass	Volume	Mass	Volume
Baseline	90.63	97.27	100.00	100.00	99.61	100.00	**96.75**	**99.09**
Non-uniform mesh simplification: Front only								
5%	52.73	51.95	48.44	92.19	63.28	20.31	**54.82**	**54.82**
10%	51.17	44.14	46.09	80.86	47.66	55.47	**48.31**	**60.16**
15%	48.44	46.48	44.92	54.69	42.97	47.27	**45.44**	**49.48**
20%	49.22	55.08	49.22	53.13	53.13	46.09	**50.52**	**51.43**
30%	49.22	47.66	45.70	53.52	55.86	51.17	**50.26**	**50.78**
Non-uniform mesh simplification: Back only								
5%	61.72	51.17	52.34	91.80	54.69	21.09	**56.25**	**54.69**
10%	50.00	53.52	47.27	83.59	49.22	54.3	**48.83**	**63.80**
15%	48.44	51.17	45.31	75.78	51.17	46.88	**48.31**	**57.94**
20%	51.95	51.56	44.53	71.09	48.83	46.88	**48.44**	**56.51**
30%	49.61	48.05	49.61	64.06	48.44	51.56	**49.22**	**54.56**