Extended Geometry based Watermarking of 3D Meshes

Improving robustness of geometrical 3D mesh watermarking by introducing mesh regularization

Jaden Nierop[1]

Supervisor(s): Zekeriya Erkin[1]**, Devris Isler**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 29, 2023

Name of the student: Jaden Nierop
Final project course: CSE3000 Research Project
Thesis committee: Zekeriya Erkin, Devris Isler, Petr Kellnhofer

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

Digital watermarking has been used extensively in media in recent years. Yet, there are still relatively few techniques for watermarking 3D meshes. In this paper we implement a watermarking algorithm proposed by O. Benedens that encodes a bit string in the distribution of the normals of the faces of the mesh and investigate a method to improve the robustness of 3D mesh watermarking algorithms against connectivity attacks by introducing a mesh regularization step at the watermark insertion and extraction phases. Furthermore, O. Benedens' watermarking technique is augmented to embed bit strings of any length and we suggest a method to measure the similarity of two bit strings in terms of probabilities.

1 Introduction

As gathering data becomes an increasingly popular investment, naturally, so does the interest in protecting this data against piracy and unlawful distribution. Protecting against these threats to intellectual property requires the ability to prove ownership of the data. Dataset watermarking techniques enable proving ownership by embedding a secret into the data in such a way that this secret is hard to remove and hard to obtain from the data. The owner can then prove ownership by showing that only they know the secret that was embedded into the dataset during the watermarking process which implies that only they could have watermarked it. This embedded secret is called a watermark.

During the watermarking process small errors are inserted into the data. These errors must remain small enough not to lower the utility of the dataset and be made such that an external party cannot remove them without sacrificing the utility of the data.

There are already many existing digital watermarking techniques out there each with their own strengths, weaknesses and practical applications.[1] Many of these techniques are already being used in media to watermark data such as audio, video and images. However, there are still relatively few watermarking techniques available to watermark 3D objects such as 3D meshes used in computer graphics and computer aided design (CAD). This is a result of the challenges that arise when handling the arbitrary geometry of 3D meshes and the complexity of the possible attacks on the watermarks [2].

In this paper we investigate a method proposed in [2] to improve the robustness of existing 3D mesh watermarking techniques to remeshing attacks by introducing a mesh regularization step both before embedding and extracting the watermark. This method is applied to the existing non-blind 3D mesh watermarking technique proposed by O. Benedens in [3] to test it's effectiveness when used in conjunction with this technique.

Furthermore, we extend the capabilities of the watermarking technique proposed by O. Benedens to be able to embed bit strings of any length.

We also give a method to calculate the probability that a given bit string was used to embed the watermark.

Next the rest of the structure of the paper is given. In section 2 we lay out the algorithms used in this paper and the experimental setup. In section 3 we present the results of these experiments and explain how these should be interpreted. In section 4 we discuss the implication of the experimental results and some topics of further research. Finally, in section 5 we discuss how the values of responsible research were expressed in this research.

2 Methodology

In this section we cover the implementation of the watermarking algorithm and experimental setup. The watermarking algorithm described in [3] had to be re-implemented for this research because an existing implementation could not be found. The new implementation was written in python and adapted to be able to embed a bit string of any length. However, this does not mean that the capacity of the algorithm is infinite but simply that the algorithm can take an arbitrary bit string as input. Attempting to embed a bit string with more bits than the mesh has faces is guaranteed to generate some empty bins that are not able to be used to encode bits.

2.1 3D mesh watermarking algorithm

The watermarking technique proposed by O. Benedens in [3] embeds a secret bit string by first creating an orientation histogram of all the normals of the faces of the mesh with one bin per bit in the bit string. Each bin is defined by its center, a vector in 3D space, and a radius. If the angle between a normal and the bin's center is less than the radius then the normal is placed in that bin. A "0" bit is encoded into a bin by moving the center of mass of the normals in that bin in a particular direction and a "1" bit is encoded by moving the center of mass in the opposite direction. The original center of mass of each bin needs is used to retrieve the secret bit string. This is the reason that this algorithm is not blind.

The retrieval algorithm is very similar. First the orientation histogram of the watermarked mesh with the same number of bins is created. By comparing the center of mass of a bin of the watermarked mesh with the original center of mass it is possible to determine if a "0" or a "1" bit is encoded in the bin.

This algorithm relies on the fact that the distribution of the normals of a mesh should be robust against remeshing, randomization of points and simplification attacks. A detailed explanation of the algorithm follows in the next two subsections.

Embedding algorithm

To create the orientation histogram of the mesh we first calculate the normals of each face in the mesh so that we can assign them to their appropriate bins. 3D meshes are often defined by a set of faces each of which is defined by 3 vertices. The normal of a face can be calculated by the cross product of two of the edges:

$$face_i = (p_1, p_2, p_3) \tag{1}$$

$$\vec{u} = p_2 - p_1 \tag{2}$$

$$\vec{v} = p_3 - p_1 \tag{3}$$

$$\vec{n_i} = \frac{cross(\vec{u}, \vec{v})}{|cross(\vec{u}, \vec{v})|} \tag{4}$$

Here $face_i$ is the i-th face of the mesh and p_1, p_2 and p_3 are the vertices that define it. $\vec{n_i}$ denotes the normal of $face_i$ and is a unit vector.

Next we generate the set of bins of the orientation histogram. Each bin is defined by its center, unit vector in 3D space, and its radius. We want the set of bins to cover as much of the unit circle as possible so that the histogram captures as many normals as possible. This task of generating good bin centers then translates to the problem of generating evenly spaced points on a unit sphere which is a notoriously hard problem in mathematics and computer science. Fortunately, we do not need the points to be perfectly evenly spaced. For this implementation we generate a fibonacci sphere which is a very simple way to produce an arbitrary amount of points that are approximately evenly distributed on the unit sphere [4]. Each point defines the center of one of the bins. Consequently, this allows the algorithm to embed a bit string of an arbitrary length. To guarantee that a normal cannot fall into two different bins we must ensure that no bins overlap. This is achieved by calculating the smallest angle between all the bin centers and cutting it in half. The result is the radius of all the bins. Lets denote this radius with r.

Lets use bin_j to denote the bin that will encode the bit in position j and let $center_j$ denote the center of bin_j. Inserting the normals into the correct bin is achieved by assigning normal $\vec{n_i}$ to bin_j if and only if the angle between $\vec{n_i}$ and $center_j$ is less than r. Because the bins are circular and do not overlap that there are points on the surface of the unit sphere that do not belong to any bin. This means that that some normals may not fall in to any bins.

When the normals have been assigned to their corresponding bins, the orientation histogram is created. To determine how to move the center of mass of each bin we map the normals of each bin from a section of the unit sphere in R^3 to a unit circle in R^2 where the center of the bin becomes the origin of the unit circle. Working in R^2 makes the following calculations simpler because it both normalizes bins to a unit circle and eliminates a degree of freedom being the distance to the origin in R^3 which is irrelevant since all the normals are unit vectors. To calculate the 2D coordinates (x_{ij}, y_{ij}) of $\vec{n_i}$ which is in bin_j we use transformation T_j. T_j represents an initial rotation around the x-axis followed by a second rotation around the y-axis such that $center_j$ under this transformation maps to the z-axis. In other words:

$$T_j * center_j = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5}$$

If $\vec{n_i}$ is assigned to bin_j then its 2D coordinates (x_{ij}, y_{ij}) are calculated as follows:

$$T_j * \vec{n_i} = \begin{pmatrix} xp \\ yp \\ h \end{pmatrix} \tag{6}$$

$$l_1 = arccos(h) \tag{7}$$

$$l_2 = |(xp, yp)| \tag{8}$$

$$x_{ij} = \begin{cases} xp * \frac{l_1}{l_2} & \text{if } l_2 > 0.01 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$y_{ij} = \begin{cases} yp * \frac{l_1}{l_2} & \text{if } l_2 > 0.01 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

It is important to calculate the center of mass in 2D of each bin here because this information will be needed to detect the watermark. The center of mass of bin_j is the sum of 2D coordinates of all the normals in bin_j divided by the number of normals in bin_j.

To encode a "0" in bin_j we move the center of mass of the bin in 2D in the positive x direction, to encode a "1" bit we move it in the negative x direction.

However, we cannot move the normals directly because they are a product of the positions of the vertices that make up the face but it is possible to indirectly move the normals by repositioning the vertices that define it's face. This presents yet another challenge: vertices usually have multiple adjacent faces that are all simultaneously affected by a vertex displacement. This means that when optimizing the position of a vertex we need to take it's effect on all the adjacent faces into account.

The embedding process loops over every vertex in the mesh and attempts to find it a new position that minimizes the cost of that vertex. The optimization method used in this paper and by O. Benedens in [3] is the Nelder-Mead method [5]. The inner workings of the Nelder-Mead method is not discussed in this paper.

After every vertex has been optimized the mesh can be considered watermarked. It is possible verify that the mesh has indeed been watermarked by attempting to retrieve the secret from the mesh. The retrieval algorithm is described in the next section.

Retrieval algorithm

The retrieval algorithm consists largely of the same first few steps of the embedding process. First calculate the normal of each face in the watermarked mesh. Then generate the fibonacci sphere with the same number of bins as as there are bits in the secret bit string. The orientation histogram is then created and the center of mass of each bin is calculated using the 2D coordinates of each of the normals with respect to their bins. With the knowledge of the original centers of mass prior to watermarking it is possible to derive the embedded secret by using the following simple rule: If the x coordinate of the original center of mass is less than the x coordinate of the new center of mass then a "0" bit is encoded, if not then a "1" bit is encoded.

Scoring the extracted bit string

The watermarking scheme defined previously have a caveat however. The retrieved bit string is not guaranteed to match the original bit string in every bit position. This is a consequence of the cost function. The cost function guarantees that normals inside a bin are not allowed to leave their bin. However, normals that originally do not belong to any bin could have drifted in such way that they now fall within the radius of a bin. This drifting normal is contributing to the center of mass of this new bin during the watermark retrieval process while not being accounted for during the embedding process which can occasionally cause the encoded bit in these bins to flip.

It is also possible to attempt to retrieve a watermark from a mesh that has not been watermarked. In this case a bit string with equal length to the provided centers of mass list will be retrieved. Since this mesh was not watermarked, bit strings retrieved this way behave as if sampled from a uniform distribution. This means that when failing to successfully embed a watermark there is a small but non-zero probability that the retrieved bit string completely matches the bit string that was not successfully embedded.

For this reason and to capture the fact that this algorithm allows the embedding of bit strings of any length while tolerating a small number of bit flips we propose a method to score the similarity between the retrieved bit string and another that may or may not have been used to embed it. This method is described next.

To decide whether to accept that the retrieved bit string is likely the result to another bit string that may or may not have been used in it's embedding process we use a strategy inspired from the hypothesis testing. Let the null hypothesis be that the two bit string are independent. Let there also be a p-value which will indicate whether to reject or accept the null hypothesis. Assuming the null hypothesis it can be argued that the retrieved bit string is sampled from a uniform distribution. Denote the number of matching bit positions between the retrieved bit string and the embedding one with k and denote the length of the secrets with n. If the probability of having k or more matching bit positions is smaller than the p-value the null hypothesis can safely be rejected meaning that we believe that the retrieved bit string was likely produced by the embedding bit string. The probability of having at k or more matching bit positions between two bit strings of length n is calculated with the following equation:

$$P(k \leq X \leq n) = \sum_{i=k}^{n} \frac{\binom{n}{i}}{2^n} \tag{11}$$

Here X is the number of matching bit positions between two bit strings of length n.

We use this probability to measure the results of the experiments in this paper because it enables comparing results when the bit string length varies between runs and it indicates a level of confidence even in the presence of bit flips. Note that the smaller this probability the more confident we are that the retrieved bit string is a result of embedding the other bit string.

Randomization of points attack

A randomization of points attack is a very simple attack that can be performed on a 3D mesh to attempt to destroy an embedded watermark. This attack consists of displacing every vertex in a mesh a distance d in a direction chosen at randomly per vertex. We use this method because it is possible to carefully control the intensity of the noise this attack introduces by controlling the displacement distance d. A displacement of 0.1 units corresponds to moving every vertex a 0.1 units from its original it's original position and a displacement of 0 corresponds to not attacking the mesh at all.

2.2 Experimental setup

This section describes the experiments that we conducted. The implementation of the algorithm proposed in [3] used in this paper is written in python along with the implementation of the randomization of points attack. The regularization algorithm is provided by the open-source 3D modeling software Blender.

Furthermore, all watermarking experiments will use a 3D mesh of a bishop (chess piece) to the mesh that will be watermarked. See figure 1. This was chosen because it had both small and large geometric features allowing it to serve as a good benchmark for how these watermarking algorithms could affect the quality of the mesh.

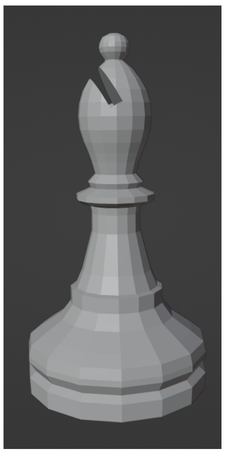

Figure 1: 3D rendering of the bishop mesh used for watermarking experiments. This mesh consists of 612 vertices and 1220 faces.

Regularization Experiment

[2] suggests a method to make 3D mesh watermarking techniques more robust against remeshing attacks. It is suggested that remeshing the 3D mesh such that it is more regular before embedding the watermark and performing a similar remeshing step again before extracting the watermark could make the strategy more robust against connectivity attacks such as randomization of points. Here a 'regular' mesh is defined to consist largely of faces of equal size. The reasoning behind this suggestion is that the regularization step before extracting the watermark is likely to reduce the (irregular) noise introduced by connectivity attacks. This can potentially undo some of the effects of the attack increasing the likelihood of the watermark being preserved.

We measure the effectiveness of this suggestion by first measuring the robustness of the watermarking scheme without the regularization step against randomization of points attacks of varying intensity. We then run this experiment again but this time we introduce the mesh regularization step before the watermark is embedded and again before the extraction.

The regularization scheme used in the experiment is provided by Blender. Blender is a very powerful open-source software for 3D design and animation. The feature we use is called the 'remesh modifier' with a voxel size of 0.1 meters which generates a new mesh with the same volume as the input mesh but with highly regular faces with lengths of roughly 0.1 units. It was not possible to find an implementation of this remeshing algorithm in python. As a result the regularization steps are executed manually by first importing the 3D mesh into Blender for regularization and saving the output to a file which can be read by our python implementation of the watermarking algorithm. The watermarked mesh is then attacked by our randomization of points implementation and the result is saved to a file. To extract the watermark the mesh in this file is made regular again by importing it into Blender and applying the remeshing modifier to the mesh with the same settings and exported to a new file so that the watermark can be extracted using the python implementation of the extraction algorithm.

By running this experiment with varying vertex displacement distances in the randomization of points attack and plotting the score against these distances we can visualize the robustness of this regularization step different degrees of noise.

By repeating this experiment while omitting the regularization steps it becomes possible to measure the effectiveness of the regularization. If the introduction of these regularization steps indeed improves the robustness against randomization attacks we expect the runs that include these steps to score better than the runs that do not.

Additionally we repeat this experiment using O. Benedens algorithm, however, now regularizing the mesh only before embedding. Simply having a mesh with more regular faces may already improve robustness.

3 Analysis

In this section we present the results of the experiments that were conducted and interpret these results.

3.1 Regularization Experiment

In this experiment we evaluate the score averaged over 20 runs for 3 variations of the watermarking algorithm. We plot these scores against the vertex displacement distance used in the randomization attack. See figure 2

The effect of the mesh through the regularization experiment can be visualized in figure 3. The figure shows the bishop mesh at several stages of the watermarking process. The first bishop in this figure has already been regularized.

When compared to the original mesh in figure 1 the new mesh appears more smooth. This is a consequence of the regularized mesh having many more faces (circa 10000) of roughly equal sizes. This mesh is subsequently watermarked. The distortion introduced into this regular mesh is much less noticeable to the human eye when compared to the watermarked mesh that has not been regularized, see figure 4.

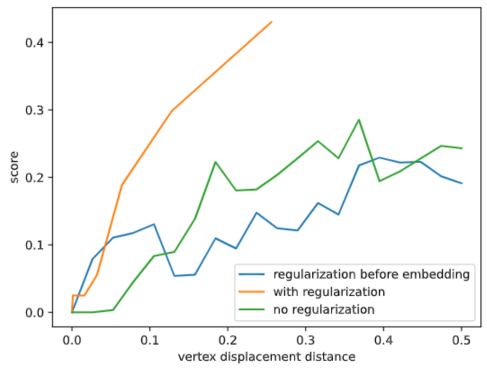

Figure 2: Scores of randomization attacks on O. Benedens algorithm. The lower the score the more bits of the watermark survived the attack. The green line shows the scores of the algorithm without any mesh regularization. The orange line shows the scores of the algorithm with regularization before both the embedding and retrieval phases while the blue line shows the score when only regularizing before the embedding phase.

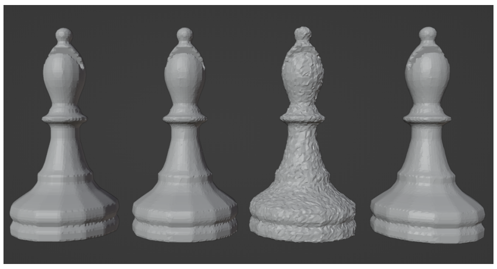

Figure 3: The regularized bishop mesh (left). The regularized bishop mesh with an embedded watermark (left-middle). The regularized bishop mesh with an embedded watermark that has been randomly attacked with a vertex displacement distance of 0.05 (right-middle). The watermarked and attacked mesh after it has been regularized again (right).

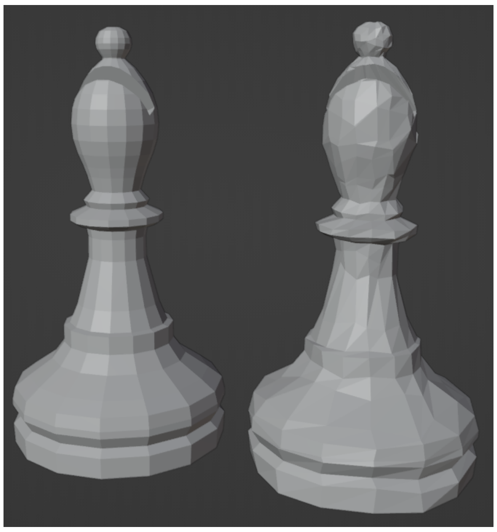

Figure 4: The original bishop before (left) and after (right) being watermarked using O. Benedens technique.

4 Conclusion

In this section we discuss the results of the experiments and discuss their compatibility with the hypothesis that regularization may increase robustness against randomization of points attacks.

The results of the experiments appear to contradict the idea that mesh regularization improves the robustness against randomization attacks. The results show a significant decrease in robustness with the introduction of mesh regularization steps with the worst performance when applying both regularization when embedding and extracting the secret bit string. The run with the best robustness against these randomization attacks does not include any regularization steps at all.

Upon considering possible causes for this discrepancy one particular explanation stands above the rest. The watermarking algorithm introduces noise into the mesh which encodes the watermark. However, the regularization step before retrieving the watermark tends to destroy small irregularities in the mesh, this includes the noise encoding the watermark, consequently destroying much of the watermark in the process.

Nevertheless, this does not explain why only regularizing the mesh before the embedding phase is also less robust than not regularizing at all. We believe there is another explanation for this. Namely, the regularization step produces a mesh with more regular faces that tend to be significantly smaller. In this case the regularization algorithm generates faces roughly 0.1 units long/wide. These faces are smaller than those present in the original irregular mesh and because they are smaller the orientation of the normals defined by these small faces are much more sensitive to vertex displacements. This theory is supported by the fact the experiment where we only regularize before embedding only performs significantly worse than not regularizing at all when the vertex displacements are small.

Although the suggestion to incorporate mesh regularization into the algorithm does not increase the robustness of the algorithm, it does seem to increase the capacity of the algorithm. When regularizing the mesh before watermarking it a new mesh is generated with roughly 10000 vertices and faces. The results show that this mesh is less robust against randomization of points attack, however, it is able to more reliably embed the watermark. Meaning that bit flips were less likely to occur during the embedding process. The noise introduced into the mesh by the watermark was in this case also more subtle and less visible to the human eye. As a result this technique may serve as good fragile watermarking technique but further research is needed.

We show that the regularization strategy does not increase the robustness of O. Benedens' watermarking technique. However, whether this strategy is more effective on other 3D mesh watermarking techniques such the frequency analyses based approaches presented in [6], [7] remains to be investigated.

We build upon the algorithm by using a fibonacci sphere to generate the bins of the orientation histogram which allows this algorithm to accept bit strings of any length. This feature may be used to more precisely further investigate the effects the geometric properties of meshes such as the number of faces and vertices have and the mesh's watermarking capacity.

Additionally, we propose a strategy to determine whether two bit strings should be considered matching by considering the probability that the two bit strings match in a certain amount of bit positions. This strategy also doubles as a measure for robustness for watermarking techniques that tolerate a small number of incorrect bits.

5 Responsible Research

In research it is very important to ensure that experiments are reproducible so that it can be tested and verified independently. For this reason we make sure to publish the code, implementations and meshes used in this research publicly. This also enables others to use these implementations and build upon this work in future research.

The ethical implication of this research are also important to consider. This research contributes our understanding of watermarking techniques which enables better protection of intellectual property rights. Therefore, we conclude that this study does not pose any ethical risks.

References

[1] S. Kumar, B. K. Singh, and M. Yadav, "A recent survey on multimedia and database watermarking," *Multimedia Tools and Applications*, vol. 79, no. 27, pp. 20 149–20 197, 2020.

[2] K. Wang, G. Lavoué, F. Denis, and A. Baskurt, "A comprehensive survey on three-dimensional mesh watermarking," *IEEE Transactions on Multimedia*, vol. 10, no. 8, pp. 1513–1527, 2008.

[3] O. Benedens, "Geometry-based watermarking of 3d models," *IEEE Computer Graphics and Applications*, vol. 19, no. 1, pp. 46–55, 1999.

[4] B. Keinert, M. Innmann, M. Sänger, and M. Stamminger, "Spherical fibonacci mapping," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–7, 2015.

[5] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, 1965.

[6] 1. R. Ohbuchi, 1. A. Mukaiyama, and 2. S. Takahashi, "A frequency-domain approach to watermarking 3d shapes," *Computer Graphics Forum*, vol. 21, no. 3, pp. 373–382, 2002.

[7] F. Cayre, P. Rondao-Alface, F. Schmitt, B. Macq, and H. Matre, "Application of spectral decomposition to compression and watermarking of 3d triangle mesh geometry," *Signal Processing: Image Communication*, vol. 18, no. 4, pp. 309–319, 2003.