Abstract:

This project tries to study the watermarking scheme proposed in “A new watermarking scheme for 3D triangular mesh models”- Ai, Lui, Zhou, Yang, Xie. A slight variation of 2D watermarking scheme, to that proposed in “Reversible watermarking of 2-D Vector Data” – Coigt, Yang, Busch, is used by us. Experimental results show that the proposed watermarking scheme results in an increased capacity at the cost of slightly higher level of distortion, measured in terms of PSNR.

Introduction:

Most current watermarking technologies focuses on 1D ( audio) or 2D (image or video). Watermarking for 3D data is still at its infancy. However, with development of 3D models used in high quality cinema or complex 3D mesh models of structural parts of a machine developed as CAD models, there is a need to watermark them for ownership issues. This is in itself a challenge because of the following issues:

1) A 3D model has limited number of points, compared to an image which has around 100 thousand pixels.
2) A 3D model is typically characterized by far apart discrete set of points unlike audio or images where these points occur close together.
3) No implicit ordering of data exists.
4) A variety of 3D transformation can easily destroy the embedded watermark while still preserving the model shape intact.

A novel approach is presented in [1] that tackles all these issues to some extent. We follow the approach while making slight variations to it. The steps involved in the algorithm are mentioned in the next section.

Algorithm:

The algorithm can be broken down into the following main steps. Each step is explained with the necessary mathematical background and details.
1) **Feature Point Selection:**

*Definition:* An $\alpha$-ring neighbourhood of a vertex $v$ are the set of points whose geodesic* distance from $v$ is equal to $\alpha$. Informally, these are the set of points that are at one-hop distance from the vertex $v$.

A feature point corresponds to a vertex with maximum information. Informally, these are points that have sharp geometric characteristics, e.g.: a horn, a hollow area. These points make good feature points because any modification to these results in a distortion easily perceptible to the naked eye. Feature points detected from the abrupt changing regions of a 3D mesh model possess its global geometry characteristics and therefore can survive after common attack operations.

To extract feature points from a mesh model, proceed as follows:

**Step 1:** The construction of neighborhood for vertexes

If a vertex of the 3D triangular mesh model (denoted by $M$) is represented by $v_i \in M (i = 1; 2; ...; N)$, its coordinates can be defined by the vector $\vec{v}_{i} (i = 1; 2; ...; N)$. Then the one-ring neighborhood vertexes of vertex $v_i$ can be defined as

$$N(v_i) = \{v_j | v_i v_j > 0, i = 1, 2, \ldots, N; j = 1, 2, \ldots, N\}$$

where $|v_i v_j| > 0$ demonstrates the connection relationship between vertex $v_i$ and vertex $v_j$, $N$ is the total vertex number of a 3D mesh model. The $\alpha$-ring ($\alpha\neq 1$) neighborhood vertexes of $v_i$ can be defined in the similar way.

**Step 2:** Calculate a normal for each vertex

The most appropriate vertexes that can be chosen as feature points of a 3D mesh model are located at regions with abrupt changing. In order to find out these points, the normal direction of each vertex must be determined.

The normal direction of vertex $v_i$ can be approximately determined by the weighted average of its one-ring neighborhood vertexes, which can be calculated using the following equation:

$$\vec{n}_{v_i} = \frac{\sum_{j=1}^{N_i} v_j \in N(v_i) (\vec{v}_j - \vec{v}_{m})}{N_i}$$

where $N_i$ is the total number of the neighborhood vertexes of $v_i$, $\vec{v}_{m}$ is the average vector of all the vertexes to the center of a 3D mesh model, vector $\vec{n}_{v_i}$ denotes the normal.
Step 3: After determining the normal direction of each vertex, the regions of the 3D mesh model can be estimated according to the following formula:

\[ D(v_i) = \sum_{j=1}^{N_i} \cos^{-1}\left( \frac{n_{v_i}}{|n_{v_i}|}, \frac{n_{v_j}}{|n_{v_j}|} \right) \]

\[ v_j \in N(v_i) \]

This is a measure of a change for a vertex \( v_i \). Larger the value of \( D(V_i) \), greater is the change associated with the vertex.

Step 4: The vertices with largest \( D(V_i) \) values are the feature points. Sort the vertices in descending order of their \( D \) values. The first \( N_f \) points are the \( N_f \) feature points. \( N_f \) is the number of feature points(constant) supplied by us.

2) Partition of the model into \( N_f \) voronoi patches:

After the feature points of a 3D mesh model are selected, the model can be divided into a number of Voronoi patches, and the range image of each patch can be generated as well.

Each feature point is employed as the centroid of a Voronoi patch, and then the geodesic* distance between each vertex and every feature point is measured; a vertex will belong to a Voronoi patch if it has the minimum geodesic* distance with the Voronoi centroid (feature point). The geodesic* distance between each vertex and every feature point is calculated as

\[ P(v_i)_n = \min[d_g(v_i, V_n), n = 1, 2, \ldots, N_f], \]

\[ i = 1, 2, \ldots, N \]

where \( d_g(v_i, V_n) \) is the measurement function to calculate the geodesic* distance between vertex \( v_i \) and feature point \( V_n \).

Since \( N_f \) feature points have been selected, a 3D mesh model can be divided into \( N_f \) Voronoi patches.

*Author has used geodesic distance in his implementation of this algorithm, we have used Euclidean distance
3) Watermark Embedding process

The key idea in the algorithm is that the watermark is embedded in each of the Voronoi patches. All the processing described henceforth is applicable to a single voronoi patch. The same procedure is applied to each of the Voronoi patch. The steps are as follows:

**Step 1:** The normal direction of a Voronoi patch is defined as the normal direction of the centroid (the feature point that resulted in the clustering of this voronoi patch). All the points in the voronoi patch are projected along the normal direction onto a plane perpendicular to the normal direction of the centroid.

Let \((x, y, z)\) be the projection of a point \((p_x, p_y, p_z)\) along the direction \((n_x, n_y, n_z)\). The values of \(x, y, z\) can be obtained as follows:

\[
\begin{align*}
x & = p_x - k \times n_x \\
y & = p_y - k \times n_y \\
z & = p_z - k \times n_z
\end{align*}
\]

where

\[
k = \frac{(p_x + p_y + p_z)}{(n_x^2 + n_y^2 + n_z^2)}
\]
Here we can see two voronoi pathes projected along the normal direction onto a plane. This plane is well known as “Range image”. All the processing will be done on the range images of a voronoi patch.

**Step 2:** Once we have a range image, which are 2D points on a plane, we need to define the x and y directions for indexing these points.

Suppose V1 is the centroid of the voronoi patch. Find the vertex V2 amongst the feature points which has maximum distance (Euclidean, Geodesic, etc) from this vertex. The projection of the line joining these points onto the plane defines the x-axis for the projected points. For example, in the above figure, the direction of the arrow is the x-direction and the one perpendicular to it in the plane(not shown) can be called y-direction.

**Step 3:** Now we will need to” sweep” the points(projected points in the plane) in the x-direction thus defined. The sweep mechanism is defined as follows:

Imagine a plane perpendicular to the sweep direction. If we start from the left most point (-\infty in the x-direction thus defined) and move towards the other end, the point that passes through this plane
first is the first to be swept. Points are ordered in the order of their crossing through this imaginary
sweeping plane.

**Step 4:** Let the order for the points thus found be \([p_1, p_2, \ldots p_n]\). Calculate the distance between these
points and their original points in the surface. We want the projected depths of these points in the very
same order as defined by the sweep. Let the height array be \([h_1, h_2, \ldots h_n]\).

**Step 5:** Apply standard 1D DCT on to chunks of 8 taken from the height array. Embed watermark in the
time domain with an embedding strength \(\beta\) as follows:

- Let \(A\) be a chunk of 8.
- Apply DCT on \(A_i\) to obtain \(A_i'\)
- Select the highest frequency component of \(A_i'\)
  \[ A_i'[8] = A_i'[8] + \beta \times \text{Watermark} \]
- \(\beta = \text{Embedding strength}\)
- Watermark comes from the Watermark sequence \((w_1, w_2, w_3 \ldots)\) each extracted from \(N(0,1)\)
- Apply Inverse DCT now to \(A_i'\)

This gives us new values of heights for each points in the range image.

**Step 6:** Given the new height values, the watermarked surface can be reconstructed using basic 3D co-
ordinate geometry.

Let \((d_x, d_y, d_z)\) be the new point that corresponds to watermarked distance for a point \((x, y, z)\).
Let original points be \((p_x, p_y, p_z)\). The new points can be obtained as follows:

\[
\begin{align*}
    d_x &= k(p_x - x) + x \\
    d_y &= k(p_y - y) + y \\
    d_z &= k(p_z - z) + z \\
\end{align*}
\]

where \(k = D'/D\)
\(D' = \text{Watermarked Distance, } D = \text{Original Distance}\)

**Step 7:** Repeat steps 1-6 for each Voronoi patch to get the watermarked mesh model. The whole
procedure can be summarized as follows:
Watermark Extraction process

The algorithm is such that we require storing the original range images for each patch. This is done so that we can moderate embedding strength as per our requirements. Thus the whole procedure to extract the range image is performed on a watermarked mesh model to get individual range images. Once again the sorting is done to get the watermarked height array: \([h'_1, h'_2, \ldots h'_n]\). This can be compared with the original height array to get the embedded watermark as follows:

- Let original chunk be \(A\) and watermarked chunk be \(A_w\)
- Apply DCT to both to get \(A'\) and \(A'_w\)
- Consider highest frequency component in both. Watermark is \(w_i = (A'_w[8] - A[8]) / \beta\)
- Extract the watermark sequence \((w_1, w_2, w_3, \ldots)\) from a range image.

We can get the watermark from each patch. Get the averaged watermark \(W\) as the average of all the watermark extracted.

- Calculate correlation coefficient between the embedded and the extracted watermark \(W^*\) as follows
  \[
  c(W, W^*) = \frac{\sum_{i=1}^{L}(w_i - \bar{w})(w_i^* - \bar{w}^*)}{\sqrt{\sum_{i=1}^{L}(w_i - \bar{w})^2} \sqrt{\sum_{i=1}^{L}(w_i^* - \bar{w}^*)^2}}
  \]
  - If \(c > T\), where \(T\) is a threshold, watermark is detected.

For estimating the quality of watermarked mesh model we use PSNR (Peak Signal to Noise Ratio), defined between original and watermarked mesh model as:
\[ PSNR = 10 \log_{10} \frac{N \cdot \| \bar{v} \|_{\text{max}}^2}{\sum_{i=1}^{N} \| \bar{v}_i - v_i \|^2} \]

where \( v \) = vertex farthest from center, \( V_i^\star - V_i \) = perturbation, \( N \) = Total vertex number of the model. Larger the PSNR better is the watermarking.

**Results:**

The following table shows the results we obtained when the algorithm was run for various values of beta on the mesh model venus:

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>PSNR</th>
<th>Correlation Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>67.7027</td>
<td>0.8302</td>
</tr>
<tr>
<td>0.1</td>
<td>51.0723</td>
<td>0.7848</td>
</tr>
<tr>
<td>1</td>
<td>30.6755</td>
<td>0.2524</td>
</tr>
</tbody>
</table>

More results are provided in the original paper as researched by the author.

**References**

1) A new Digital Watermarking Scheme for 3D triangular mesh models” – Ai, Liu, Zhou, Yang, Xie, 2009
2) Reversible watermarking of 2-D Vector Data” – Coigt, Yang, Busch.
3) [http://www.ceremade.dauphine.fr/~peyre/teaching/manifold/tp2.html](http://www.ceremade.dauphine.fr/~peyre/teaching/manifold/tp2.html)