An Intelligent Learning Approach For Information Hiding In 3D Multimedia

Rakhi Motwani, Mukesh Motwani, and Frederick Harris, Jr.,
Department of Computer Science and Engineering
University of Nevada, Reno USA
{*mukesh, rakhi, fredh*}*@cse.unr.edu*

***Abstract*—This paper presents a new watermarking algorithm for 3D triangular mesh models that is based on surface curvature estimation and supervised learning. A feedforward backpropagation neural network is adopted for selecting vertices for watermark insertion. A variety of 3D models with varying degrees of surface curvature are used to train and simulate the neural network. An array of neural networks is used for vertices with different valences to achieve higher watermark embedding capacity. A gray scale bitmap image is used as the watermark. The watermark extraction process is informed and needs the original watermark and 3D model. Experimental results evaluate the embedding capacity, imperceptibility and robustness of the proposed algorithm and simulate various attacks including noise addition, smoothing and cropping.**

***Keywords*-watermarking; 3D models; surface curvature; supervised learning;**

I. Introduction

The medical industry makes extensive use of 3D models for surgeries [1] and computational modeling [2]. The high sensitivity for the integrity of medical data being modeled demands tamper-proofing mechanisms. Information hiding techniques are used as a means to protect digital data against tampering by embedding copyright information into the digital content. The embedded hidden information, referred to as the watermark, serves as proof of original ownership and authenticates the digital content. However, the watermark insertion process must maintain visual integrity of the digital data and should not introduce any perceivable artifacts that distort or interfere with the original content. This requires identifying those regions of a 3D model for watermark insertion, where the human visual system is the least sensitive to distortions.

Research on 3D model watermarking has employed various approaches [3] to achieve imperceptibility. Some of these techniques involve changing the order of the 3D data in the model's file format to encode the watermark, re-triangulating parts of the 3D triangular mesh, changing positions of vertices according to their local moments, using masking functions to add the watermark, hiding information in the heights of the triangles of the mesh, altering the distance between vertices of the model and the center of gravity of a given reference triangle for each selected interval, modifying the length ratio between edges of a triangle, and algorithms that are based on the local mean and Gaussian curvatures to identify appropriate regions for watermark insertion. The approach in this paper is an extension of our previous work [4]. The proposed technique computes the local curvature of a 3D surface and trains an artificial neural network to classify regions with varying levels of watermark embedding strengths.

The remainder of this paper is structured as follows: In Section 2, we briefly outline the structure and functionality of an artificial neural network. Section 3 describes the training process for artificial neural network, the watermark insertion and the extraction algorithm. Section 4 provides the experimental results. Final remarks and conclusions are drawn in Section 5.

II. Artificial Neural Networks

Artificial neural networks (ANN) [5] are an algorithmic modeling of biological neural systems. An ANN is a layered network of neurons (Fig. 2). The neuron, which is the basic component of an ANN, upon activation fires an output signal corresponding to a set of input signals. A neuron receives input signals p_i and aggregates these signals into a net input signal n by multiplying each input signal with corresponding numerical weights w_i and summing up all of these computed numerical values along with a bias b, as shown in Fig. 1.

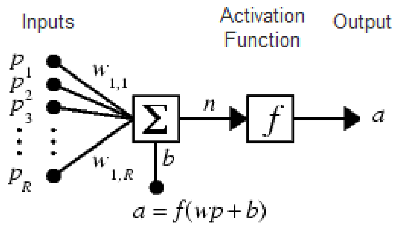

Figure 1. Neuron [6] - The Building Block of a ANN

The output signal a is computed by an activation function f that takes n as the input. An activation function can be linear or non-linear in nature. The most commonly used activation functions, such as *sigmoid* and *hyperbolic tangent*, map n to a in a non-linear way. The neuron transmits an output signal only when it is activated i.e. the net input signal falls within the working range of the activation function. The bias is used to change the threshold at which a neuron activates and is adjusted by the learning phase of the ANN. The weights w, that control the strengths of the input signals, are very critical in defining the behavior of the ANN and evolve during the learning phase as well. The goal of the learning phase of ANN is to determine the best values for w and b from a given set of data and adjust these values until a certain criterion is satisfied.

In supervised learning, the neuron is provided with a training data set consisting of input vectors and a target (desired output) associated with each input vector. A supervised learning ANN uses the target vector to determine how well it has learned, and to guide adjustments to weight values to minimize the overall error between the real output of the neuron and the target output.

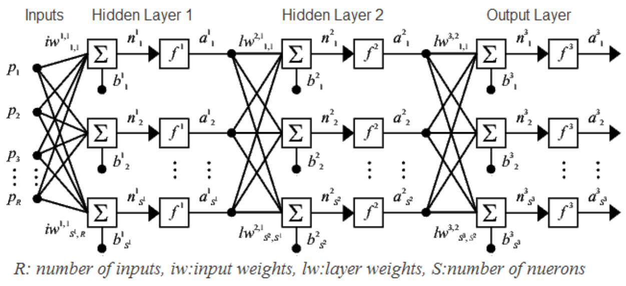
Inputs
Hidden Layer 1
Hidden Layer 2
Output Layer
R: number of inputs, iw:input weights, lw:layer weights, S:number of nuerons

Figure 2. Multi-Layer ANN [6]

A feedforward ANN propagates the signals through all the layers to obtain the result, which is the output of the last layer in the network. Backpropagation is one of the various approaches to train the ANN such that the output of the network is an accurate approximation of the target values. During the learning iterations, the output value of the ANN for each training pattern is computed and the error signal is propagated back from the output layer toward the input layer so that the weights are revised appropriately.

For the proposed approach, described in the following section, the neural network is used as a classifier(to predict the class of an input vector). The objective of the employed feedforward backpropogation ANN in this paper is to learn the watermark embedding capacity of a vertex from a given set of training data.

III. Approach

3D triangular mesh models are represented by a set of vertices and a list of triangular faces formed by the vertices. A vertex v_i is a neighbor of another vertex v_j if an edge exists that connects v_i and v_j . The set of all the neighbors of a vertex v_i is called 1-ring of the vertex. The set of all neighbors of the 1-ring neighbors of a vertex v_i along with the set of 1-ring neighbors is called 2-ring of the vertex, as shown in Fig. 3. The number of neighbors of v_i in it's 1-ring neighborhood is the valence of the vertex v_i. Fig. 3 shows a vertex of valence 5 (1-ring) and vertices of valence 5 and 6 (2-ring).

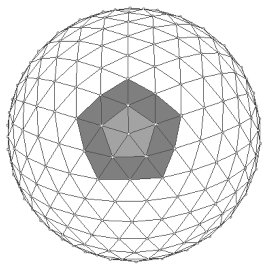

Figure 3. Neighborhood of a Vertex - 1-ring demonstrated by light gray patch, 2-ring demonstrated by light and dark gray patches

The input vectors that are used to train the neural network are derived from the 1-ring and 2-ring neighborhoods of each vertex. The 1-ring and 2-ring neighborhoods take into account the local geometry of the vertex. Based on analysis of the local curvature which is estimated by the angles between surface normals of a neighborhood, a neural network is trained such that it can appropriately choose regions from any 3D model to embed the watermark. Fig. 4 demonstrates the surface normals for 3D models with varying levels of curvature. For flat surfaces, the angles between surface normals are small in magnitude since these normals are almost parallel to each other. For regions representing edges, the angle between the neighboring normals is much larger in magnitude. While smoother regions like the *Mushroom* top have relatively smaller variations in orientation of the surface normals. Thus, angular difference between neighboring surface normals represents the curvature or local shape of a region and is an appropriate input vector for training and simulation of the artificial neural network. The neighborhood size is restricted to 2-ring such that local details of surfaces are not lost.

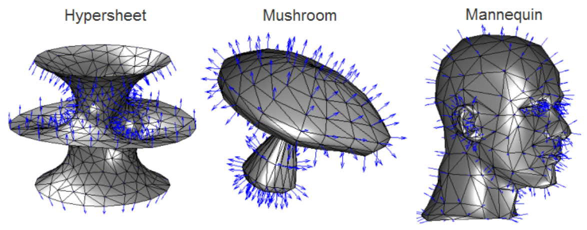

Figure 4. Curvature Estimation From Normal Vector Distribution

Fig. 5 outlines the system block diagram and the following subsections describe the three blocks of the system - Neural Network Training, Watermark Insertion, and Watermark Extraction.

A. Neural Network Training

For training the neural network, 3D models with varying degrees of surface curvature are chosen. Input vectors are computed for vertices of valence 4, 5 ,6 and 7 from each 3D model chosen for training. Input vectors are the values of

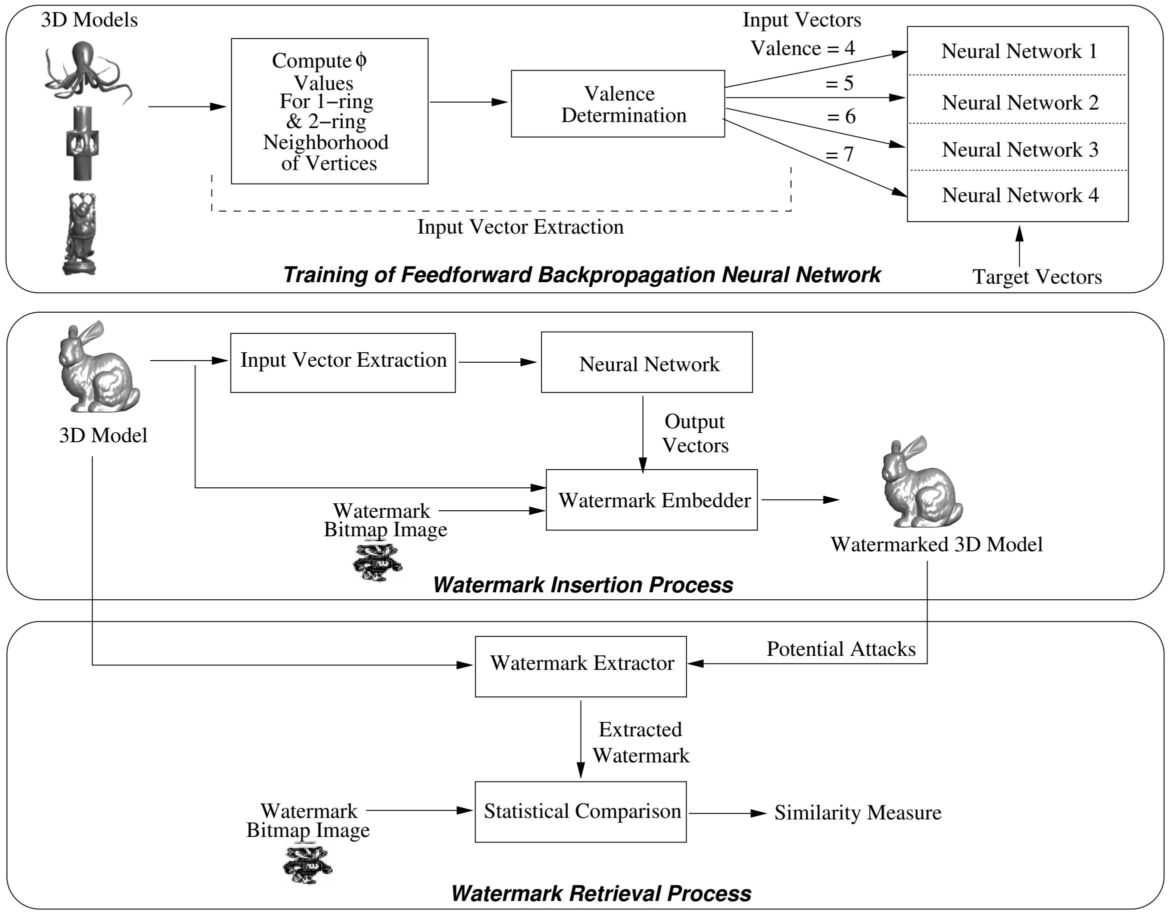

Figure 5. System Block Diagram

angles ϕ between the face normals and the average normal for a 1-ring and 2-ring neighborhood of a vertex. These ϕ values are computed using the technique specified in our previous work [4]. The limitation in the previous work was that it considered vertices of valence 6 only and the output vector specified whether a vertex was appropriate for watermark insertion or not. Here we use multiple neural networks, each trained with input vectors derived from vertices of a different valence. We use 4 neural networks, one each for vertices of valence 4, 5, 6 and 7. The output of the neural network is one of four levels(1,2,3,4) used to represent the watermark embedding strength of a vertex. The watermarking algorithm selects vertices with levels 2, 3 & 4 to embed a watermark of different strength in each level. This approach increases the embedding capacity of the algorithm while maintaining imperceptibility of the watermark. To achieve rotational invariance, all permutations of the input vectors for each vertex are added to the training set.

Target vectors, for a set of input vectors, are derived from original and watermarked models using the algorithm in [7] with the parameter for embedding capacity set at 100%(this ensures that all the vertices are modified). Subtracting vertices of the original model from the watermarked model gives the value of the watermark embedded into each vertex. The minimum and maximum value of the strength of this watermark is determined and the difference between the two values is divided into four intervals. The watermark values lying in these four ranges are assigned levels of 1,2,3,or 4. This level serves as the target vector for a set of input vectors corresponding to a vertex.

Input vectors and the corresponding target vectors are used to train the 4 neural networks such that the aggregate neural network can classify new input vectors by associating an appropriate output vector for each vertex. Fig. 6 illustrates the architecture of the feedforward back propagation neural networks used for the implementation.

The artificial neural network has 1 hidden layer with R=4, 5, 6 or 7 inputs and 1 output layer with S^1 inputs. The hidden layer uses a hyperbolic tangent sigmoid transfer function(f^1) while the output layer uses a linear transfer function(f^2). The hidden layer has $S^1 = 20$ neurons and the size $S^2 = 1$

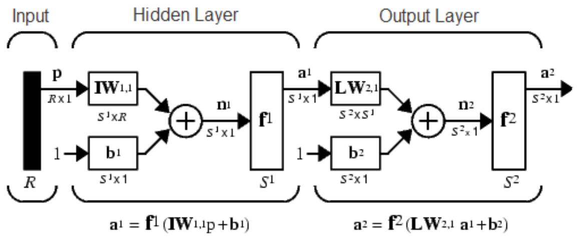

Figure 6. Neural Network Architecture [6]

of the output layer is determined by the target vector. LW denotes the layer weight matrices, IW represents the input weight matrices. 3D models different from those used in the training session are used for network simulation. Mean square error is used as the performance function to compute the error between the network outputs and the target vectors.

B. Watermark Insertion Process

For a given 3D model that needs to be watermarked, the input vectors are computed for vertices of different valences and used to simulate the artificial neural network. The output of the artificial neural network specifies the watermark embedding capacity level of the vertex corresponding to the input vector. The watermark embedder selects vertices with level 2, 3 and 4 for watermark insertion. The algorithm perturbs a vertex by using a scaling factor for the watermark to be embedded. A scaling factor of $K_1 = 0.25 * 10^{-4}$ is used for vertex of level 2, $K_2 = 0.5 * 10^{-4}$ is used for vertex of level 3 and $K_3 = 10^{-4}$ is used for vertex of level 4. A gray scale bitmap image is used as a watermark. For each co-ordinate of a vertex selected to be modified, the modification to the vertex is determined by the following equation:

$$v_{x,y,z} = v'_{x',y',z'} + KW \qquad (1)$$

where, v = Original Vertex,
v' = Watermarked Vertex,
K = Scaling Factor,
W = Watermark Data (image pixel value between 0-255).

Fig. 7 shows the 3D models that are to be watermarked.

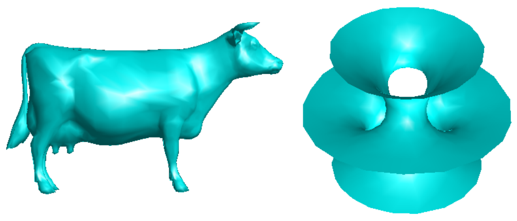

Figure 7. *Cow* 3D Model

The output vectors for vertices with varying degrees of watermark embedding capacity are shown in Fig. 8. Level 1 represents the least embedding strength and level 4 indicates that a vertex can accommodate a higher value for insertion. Toolbox Graph [8] in Matlab has been used to generate the figures.

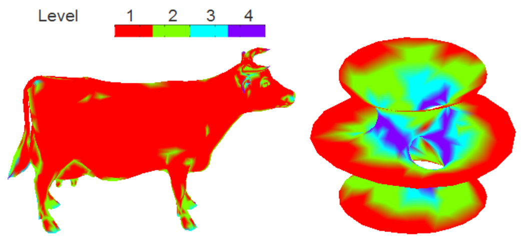

Figure 8. Watermark Embedding Strengths for Vertices of *Cow* 3D Model

The cyan regions in Fig. 9 denote the vertices selected for watermark insertion.

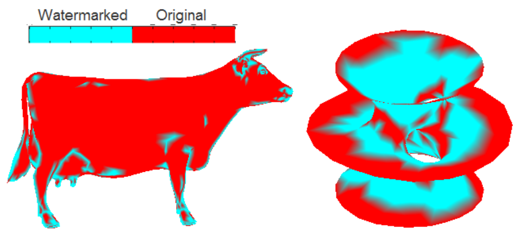

Figure 9. Regions Selected for Watermark Insertion indicated in *cyan*

The watermarked models are shown in Fig. 10

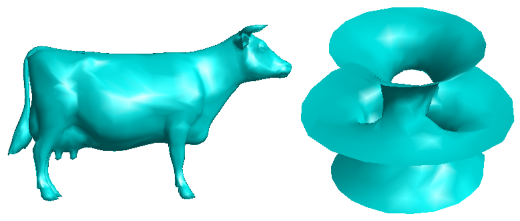

Figure 10. Watermarked *Cow* 3D Model

C. Watermark Retrieval Process

The watermark extractor is non-blind and requires the original 3D model along with the watermarked model to retrieve the watermark. The values of the extracted watermark are rescaled by factor K, which was used in Eq. 1, such that the pixel values of the watermark image are restored to their original values. The indices of vertices with level 2, 3 and 4 are saved during the insertion process to assist the rescaling of the retrieved watermark. The pixel values of this image are then compared against the originally embedded image to determine the measure of similarity between the embedded and retrieved watermark. If this measure is above 80% the 3D model is declared to be authentic. A similarity measure below 80% indicates that the model has been tampered with.

IV. Experiments

This section evaluates the capacity, visibility and robustness of the watermarking algorithm. The capacity is determined by the size of the payload that is hidden in the 3D model. A gray scale bitmap image of size 32x32 pixels is used as payload for our experiments. 3D models with embedding capacity higher than 32x32 use the same image, in whole or parts, repetitively until the entire embedding capacity is utilized. To evaluate the visibility of the watermarking process, the vertex signal-to-noise ratio(VSNR) measure determines whether the distortion caused to the watermarked model is perceptible or not. VSNR is computed by Eq. 3.

$$SNR = \frac{\sum_{i=1}^{N} x_i^2 + y_i^2 + z_i^2}{\sum_{i=1}^{N} (x_i^{'} - x_i)^2 + (y_i^{'} - y_i)^2 + (z_i^{'} - z_i)^2} \qquad (2)$$

where,
x_i, y_i, z_i are co-ordinates of vertex v_i in the original 3D model,
$x_i^{'}$, $y_i^{'}$, $z_i^{'}$ are co-ordinates of the same vertex in the watermarked 3D model, and
N is the total number of vertices in the 3D model.

$$VSNR = 20 * Log_{10}(SNR) \qquad (3)$$

Table I lists the VSNR values for different 3D models.

Model Name	Number of Vertices	Number of Modified Vertices	VSNR (dB)
Cow	2904	883	98.75
Hypersheet	487	287	119.23
Mushroom	226	98	104.57
Mannequin	428	224	61.25

Table I
Capacity and Visibility of the Watermarking Algorithm

To evaluate the robustness of the watermarking method, various attacks are simulated on the watermarked models. The proposed method is naturally resistant to rotational attacks since the neural network is trained with all permutations of the input vectors that represent rotated set of vertices. Amongst geometrical(additive noise, mesh smoothing) attacks, the algorithm is only robust against low levels of noise addition. A Gaussian noise is added to x, y, z coordinates of the watermarked model with 10-100% of the vertices selected randomly from the whole set of vertices in the model. Depending on the location of the cropping plane, experiments verified that the watermark can be retrieved despite of cropping attacks that remove parts of a 3D model. Table II lists similarity measure values for the various attacks simulated on the watermarked 3D models.

3D Model Name	Similarity Measure		
	Noise (% level)	HC Smoothing (# of steps)	Cropping (# of cropped vertices)
Cow	87.66% (10% level)	40.07% (2 steps)	67.57% (758 vertices)
Hypersheet	77.13% (10% level)	30.38% (2 steps)	62.41% (86 vertices)
Mushroom	85.06% (10% level)	42.42% (2 steps)	33.09% (105 vertices)
Mannequin	89.82% (10% level)	56.50% (2 steps)	92.91% (42 vertices)

Table II
Similarity Measure Results For Embedded and Extracted Watermarks after Various Attacks

V. Conclusion

This paper explores the use of artificial neural networks for the watermark embedding process. An extension of our previous work [4] on watermarking 3D triangular meshes using artificial neural networks has been presented. Multiple neural networks have been adopted to accommodate vertices of different valences. The outputs of a high embedding capacity algorithm [7] are utilized for training the neural networks. A gray scale bitmap images is utilized as a watermark. Watermark invisibility is achieved through embedding the watermark with different scaling factors in vertices with higher embedding strengths. Experimental results show that the presented watermarking algorithm is of higher capacity than the formerly devised algorithm [4].

References

[1] E. Takahashi and T. Kaneko, “Facial surgery simulation using 3D bubble mesh,” *Systems and Computers in Japan*, vol. 32, 2001.

[2] J. Sienz, I. Szarvasy, E. Hinton, and M. Andrade, “Computational modelling of 3D objects by using fitting techniques and subsequent mesh generation,” *Computers Structures*, vol. 78, no. 1-3, pp. 397 – 413, 2000.

[3] J.-L. Dugelay, A. Baskurt, and M. Daoudi, *3D Object Processing: Compression, Indexing and Watermarking*. Wiley Publishing, 2008.

[4] M. Motwani, S. Dascalu, B. Bryant, and F. Harris, “3D multimedia protection using artificial neural network,” in *IEEE International Workshop on Digital Rights Management*, Las Vegas, USA, 2010.

[5] A. Engelbrecht, *Computational Intelligence - An introduction*. 2nd edition, Wiley Publishing, 2007.

[6] H. Demuth, M. Beale, and M. Hagan, *MATLAB Neural Network Toolbox 6: User’s Guide*. The Mathworks, 2009.

[7] M. Motwani, B. Sridharan, R. Motwani, and F. Harris, “Tamper proofing 3D models,” in *IEEE International Conference on Signal Acquisition and Processing*, Bangalore, India, 2010.

[8] G. Peyre, “Toolbox graph - a toolbox to perform computations on graph,” Released Jun 2004(Updated Jul 2009).