Using Radial Basis Function Networks For Watermark Determination In 3D Models

Rakhi Motwani, Mukesh Motwani, and Frederick Harris, Jr.
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, NV 89557

Abstract—**In this paper, we investigate the ability of a radial basis function network to determine the values of the watermark to be inserted in a 3D triangulated mesh model. The challenge in a watermarking algorithm is to achieve high watermark embedding capacity without causing perceptual distortion to the model. The proposed technique overcomes this challenge. The principal, mean and Gaussian curvature values computed at each vertex of the 3D model collectively represent the local geometry of the vertex and its neighborhood. These curvature values are used as input feature vectors to train the neural network. The amount of watermark to be inserted at a vertex is non-linearly proportional to these feature vectors. The neural network is trained with watermark values to learn this non-linear relationship. A wide range of 3D models are used to train the neural network such that the large variations in the geometry of the vertices is incorporated by the training phase. The algorithm adopts a non-blind extraction process to retrieve the watermark. Experimental results prove that the watermarking algorithm achieves imperceptibility, high capacity and robustness to noise and cropping attacks up to 20% level.**

I. Introduction

3D models have become increasingly popular over the past several years due to their diverse applications in computer aided design, virtual environments, robotics simulations, and bio-informatics only to name a few. Copyright issues are crucial in media arts to enforce ownership rights since digital artwork can easily be duplicated, altered, and plagiarized. Watermarking techniques have been used for copyright protection of digital media. Watermarking encodes copyright information into multimedia with no perceivable difference.

The shape of the 3D object is represented as a mesh consisting of a set of triangles. The most common schemes for 3D watermarking encode information into a mesh model by slightly changing the position of vertices that form the triangles, or by altering the connectivity of the vertices to form a different set of triangles that represent the mesh. The key piece of the watermarking algorithm is the selection of appropriate vertices for embedding the watermark, such that no perceivable distortions are caused to the model by the watermarking process. The most intuitive approach to selection of appropriate vertices is based on features that characterize the local geometry of the 3D mesh model. The curvature of a surface is a commonly used feature to determine a surface suitable for watermarking [1]–[3]. In [4], the choice of local surfaces from a 3D model selected to embed watermarks, is based on the analysis of the mean and Gaussian curvatures and the fluctuations in these curvatures that correspond to the variation in the geometry of the local surface. The algorithm in [5] uses the normal vector distribution and extended Gaussian image(EGI) [6] for watermarking 3D meshes. The 3D model is divided into patches and the normal vectors of each mesh in the patch are mapped to bins on the EGI sphere. The length of each bin is the sum of area for all the meshes that are mapped into it. Bins with large length are selected as locations for watermark embedding. The authors in [7] use geodesic Delaunay triangulation and umbilical points to remesh the 3D model and then embed a watermark in the spectral domain.

This paper contributes a novel technique towards determining the amount of information, i.e. watermark, that can be embedded in the 3D model by using a Radial Basis Function (RBF) neural network. The remainder of this paper is structured as follows: Section II describes the 3D local geometric feature extraction process, the training phase of the RBF network and outlines the watermark embedding and retrieval phase of the proposed algorithm. Experimental results are presented in Section III. Conclusions are summed up in Section IV.

II. Proposed Algorithm

The algorithm involves four steps: Local Geometric Feature Extraction, RBF Neural Network Training, Watermark Insertion and Watermark Extraction. The first step extracts geometric features from the 3D model which are used as input vectors for training the neural network. These geometric features represent the shape of the 1-ring surface patches of the 3D model. The neural network learns the relationship between these shapes of the model and the amount of watermark that can be accommodated by each shape. The following subsections present the details for each step of the algorithm.

A. Local Geometric Feature Extraction

A 3D model is a triangulated surface S in a three dimensional Euclidean space R^3, and comprises of a vertex set V and an edge set E. The feature extraction process analyzes the 1-ring patch (see Fig. 1) of each vertex v in the set V to compute the vertex's curvature values. The curvature value is computed by the rate of the surface bending along a tangent direction at the vertex. The tangent direction at a vertex is the derivative of the unit normal vector n_v at a vertex v. We use the method proposed by [8], [9] and implemented by [10] to compute the curvature values. The two *principal curvatures* K_{max} and K_{min} represent the rate of maximum and minimum bending of the surface along a tangent direction at the vertex. The *mean curvature* H and the *Gaussian curvature* G at the vertex are the average and product of these two *principal curvatures*, respectively. These four curvature values represent the local geometry(i.e. shape) of the 3D surface and are used as input vectors for the RBF network training.

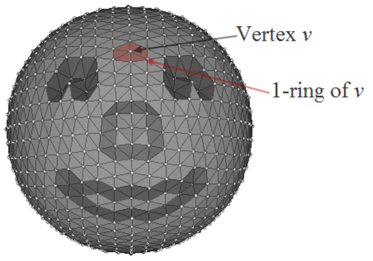

Fig. 1. 3D Tirangulated Mesh Surface - White dots represent the vertices, Red highlighted patch represents the 1-ring of a vertex

To extract the principal curvatures, the curvature tensor field at a vertex v on the mesh of S is calculated by:

$$T(v) = \frac{1}{|B|}\sum_{e \in E} |e \cap B| \beta(e) \bar{e}\bar{e}^t \tag{1}$$

where, B is the surface area around v over which the tensor is estimated (i.e. 1-ring patch), $\beta(e)$ is the angle between the normals n_v to the triangles of the 1-ring patch incident on edge e, $|e \cap B|$ is the length of $e \cap B$, and $\bar{e}$ is a unit vector in the direction of $\bar{e}$. The normal at each vertex is estimated by the eigenvector of $T(v)$ associated with the minimum eigenvalue. The two remaining eigenvalues K_{min} and K_{max} are estimates of the *principal curvatures* at v. The discrete *Gaussian curvature* measure of S is given by:

$$G = K_{min}.K_{max} \tag{2}$$

The discrete *mean curvature measure* is given by:

$$H = \frac{K_{min} + K_{max}}{2} \tag{3}$$

The relative signs of the principal curvatures determine the shape [11] of S near v. If K_{min} and K_{max} are both positive or both negative, v is an *elliptic* point of S, if K_{min} and K_{max} are of opposite sign then v is a *hyperbolic* point of S, if either one of K_{min} and K_{max} is zero and the other one is non-zero then v is a *parabolic* point of S, if both K_{min} and K_{max} are zero then v is a *planar* point of S. The sign of the *Gaussian curvature* indicates the geometry of the surface, as demonstrated by Fig. 2. If G is positive, S is bending away from its tangent plane in all tangent directions at v (e.g. corners of the cube); if G is negative, S is saddle-shaped near point v; if G is zero, S is either a plane($K_{min}, K_{max} = 0$) or cylindrical(K_{min} or $K_{max} = 0$) in nature near v. The *mean curvature* is zero for *minimal* surfaces [12] and constant for spherical surfaces.

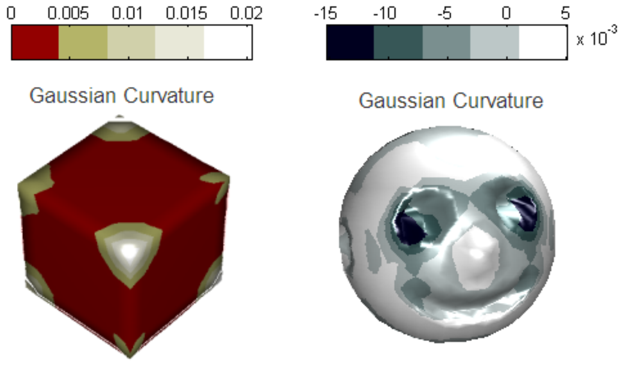

Fig. 2. Gaussian Curvatures of *Cube* and *Smiley* Models

B. RBF Neural Network Training

The training phase of the RBF network requires a large set of 1-ring surfaces of 3D models that have different geometries. These 1-ring surfaces are used to determine the principal, mean and Gaussian curvature values for each vertex. These curvature values serve as input vectors for training the RBF network. The watermark values that can be embedded in the vertices of the 1-ring surfaces serve as output vectors for the training phase. During the training phase, the RBF network learns the relation between the curvature values and the watermark value to determine the extent of watermark insertion for a new vertex.

The RBF network [13] is characterized by a set of inputs, a hidden layer for processing inputs, and an output layer, as illustrated by Fig. 3.

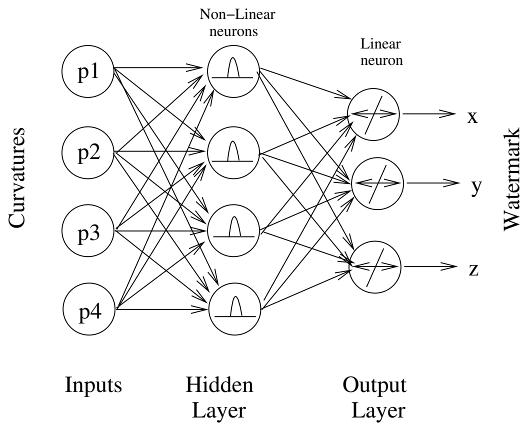

Fig. 3. Radial Basis Function Feedforward Neural Network Architecture

The activation function of the hidden layer is a Gaussian function, parametrized by a center(mean) and width(variance). For a given training set of input-output pairs, the nonlinear mapping relation between input vectors and output values is modeled by the RBF neural network by optimizing the network parameters i.e. center and width of the radial basis function, and the weights. A mean square error cost function evaluates the accuracy of the mapping function. The accuracy depends on the number of radial basis functions used, the center points and the width for each radial basis function. The values of each input variable are fed to each of the neurons in the hidden layer. The weights from the input to the hidden layer represent the center of the radial basis (Gaussian) function . At the input of each neuron in the hidden layer, the distance between the center of the radial basis function and the input vector is calculated. The output of the radial basis neuron (which serves as the weight from the hidden layer neuron to the output layer neuron,) is computed by applying the radial basis function to this distance. The neuron in the output layer implements a linear activation function and yields an output that is a linear combination of the radial basis functions in the hidden layer.

The optimal number of neurons in the hidden layer is determined by the training process and is set to the size of the input vector i.e. number of training samples. For the experiments, 1200 samples are fed to the network for training. The 4-dimensional input vectors $p = [p1, p2, p3, p4]$ are obtained from the 1-ring of every vertex from a set of 3D models that feature a wide span of curvature variations in their local geometry(1-ring), where $p = [K_{min}, K_{max}, G, H]$. The corresponding set of output values is the watermark value, for each x, y, z co-ordinate of the vertex.

C. Watermark Insertion

The process for embedding the watermark in the 3D model is outlined in Fig. 4. The 3D model is normalized before the feature extraction process to ensure that the curvature values are invariant of translation, scale and orientation operations on the 3D model. The curvature values for the normalized 3D model are computed by the feature extraction block using the method discussed in the previous section. The trained RBF neural network is simulated with these features vectors for each vertex in the 3D model, to predict the watermark value at each vertex. The watermarked model is computed by randomly selecting vertices in the 3D model and adding the corresponding watermark value to those vertices. Finally, the watermarked model is de-normalized.

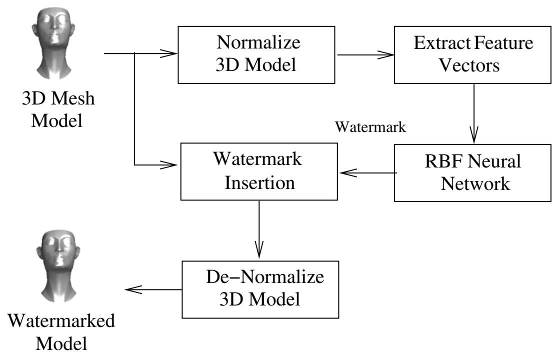

Fig. 4. Watermark Insertion Process

D. Watermark Extraction

The embedded watermark is retrieved from a test model by subtracting the vertices of the normalized original model from the vertices of the normalized version of the test model. This retrieved watermark is correlated with the originally inserted watermark to determine a measure of similarity between the two sets of watermark values. A correlation measure of 1.0 indicates a perfect match between the original and extracted watermark. If the correlation measure is above 0.75, the test model is declared to be authentic and tamper-proof. Fig. 5 sketches the block diagram for the non-blind retrieval process.

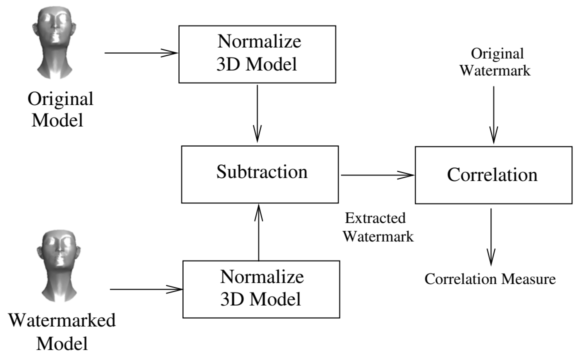

Fig. 5. Watermark Extraction Process

III. Experiments and Results

The experiments are performed using *MATLAB Neural Network Toolbox* [14] to evaluate the capacity, imperceptibility, and robustness of the proposed algorithm. The 3D models used for experimental evaluation are shown in Fig. 6. The capacity of the algorithm determines the maximum watermark that can be embedded within the 3D model without introducing any perceivable distortions in the triangulated surfaces of the model. To evaluate the imperceptibility of the presented approach, the mean square error (MSE) is measured for the original and watermarked model, and is given by:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} \left\| v_i - v_i^{'} \right\|^2 \tag{4}$$

where, v is the original vertex, v' is the watermarked vertex and N is the total number of vertices in the model. The proposed approach achieves high capacity and imperceptibility, as illustrated by Table I and Fig. 6.

3D Model	*Mushroom*	*Tre-Twist*	*Smiley*
Total Vertices	226	800	1026
Modified Vertices	146	510	937
Size of Watermark	6.7 KB	23.8 KB	43.1 KB
Mean Square Error (MSE)	3.5203 x 10^{-7}	8.5777 x 10^{-8}	2.4379 x 10^{-6}

TABLE I
Imperceptibility and Capacity of the watermarking algorithm

The correlation measure for various simulated attacks on the watermarked model such as additive Gaussian noise, HC smoothing operation and cropping is listed in Table II.

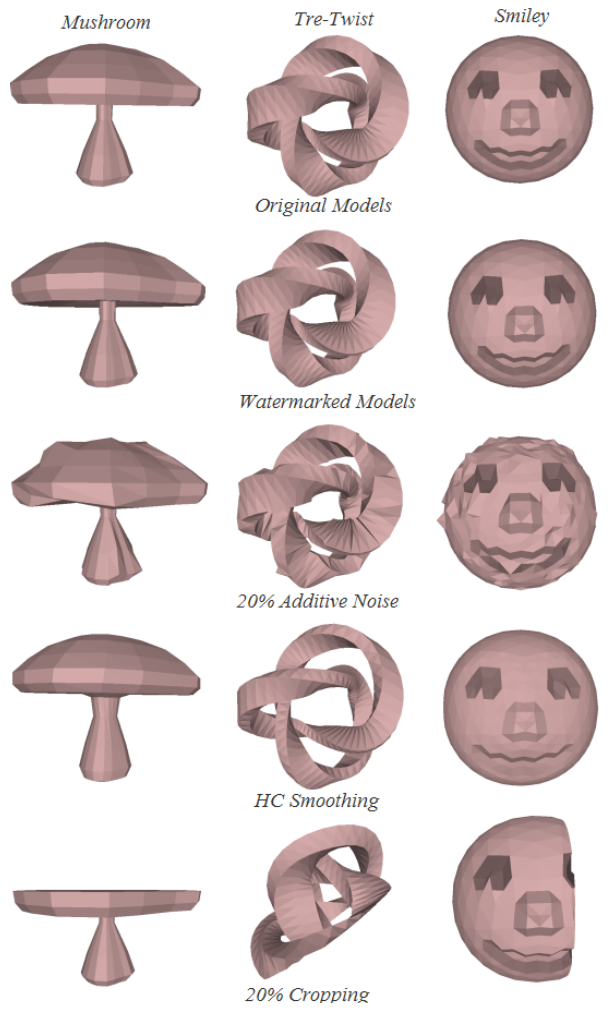

Fig. 6. Original and Watermarked 3D Models

3D Model	*Mushroom*	*Tre-Twist*	*Smiley*
Noise			
(10% level)	0.9324	0.9763	0.9807
(20% level)	0.8675	0.8142	0.8487
(30% level)	0.7248	0.7345	0.7721
HC Smoothing	0.3621	0.6110	0.6321
Cropping			
(10% level)	0.8976	0.9073	0.9210
(15% level)	0.8392	0.8761	0.8179
(20% level)	0.8016	0.7984	0.7835

TABLE II
CORRELATION MEASURE FOR EMBEDDED AND EXTRACTED WATERMARKS AFTER VARIOUS ATTACKS

A Gaussian noise of mean 0.005 and variance 0.035 is added to randomly permuted vertices of the watermarked models. The % level of noise indicates the amount of coverage of additive noise within the 3D model. A level of 10% injects noise into only 10% of the total number of vertices in the model. The HC Laplacian smoothing operation is performed using the algorithm in [15]. The cropping operation removes parts of the model along the $x, y,$ or z planes in order to damage the watermark. The level of cropping determines the extent to which the model has been truncated form its original size. The test models are subject to 10%-30% cropping levels to gauge the resistant of the watermark to cropping atacks. Experimental results indicate that the proposed algorithm can withstand upto 20% of additive Gaussian noise and cropping attacks using a threshold of 0.75 for the correlation measure.

IV. Conclusion

In this paper, the ability of a supervised learning RBF neural network is investigated for watermarking of 3D models. The proposed method is based on training the artificial neural network to learn the relationship between the geometry of the 1-ring neighborhood surface of vertices of the 3D model, and the extent of watermark that can be accommodated by the geometry of these vertices. The neural network is trained for various types of 3D surfaces (planar, cylindrical, minimal, curved, uneven) that characterize different geometries. The algorithm achieves minimal perceptual distortion and accomplishes high watermark embedding capacity. Experimental results prove that the algorithm is robust against a variety of attacks as well.

References

[1] B. Jabra and E. Zagrouba, “A new approach of 3D watermarking based on image segmentation,” in *IEEE Symposium on Computers and Communications (ISCC)*, 2008, pp. 994–999.
[2] M. Motwani, N. Beke, A. Bhoite, P. Apte, and F. C. Harris, “Adaptive fuzzy watermarking for 3D models,” *Computational Intelligence and Multimedia Applications, International Conference on*, vol. 4, pp. 49–53, 2007.
[3] G. Lavoue, “A roughness measure for 3D mesh visual masking,” in *Proceedings of the 4th symposium on Applied perception in graphics and visualization (APGV)*, 2007, pp. 57–60.
[4] D. Han, X. Yang, and C. Zhang, “A novel robust 3D mesh watermarking ensuring the human visual system,” in *Second International Workshop on Knowledge Discovery and Data Mining (WKDD)*, Jan. 2009, pp. 705–709.
[5] K.-R. Kwon, S.-G. Kwon, S.-H. Lee, T.-S. Kim, and K.-I. Lee, “Watermarking for 3D polygonal meshes using normal vector distributions of each patch,” in *Proceedings of The International Conference on Image Processing (ICIP)*, vol. 2, Sept. 2003, pp. II–499–502 vol.3.
[6] B. Horn, “Extended gaussian images,” *Proceedings of the IEEE*, vol. 72, no. 2, pp. 1671–1686, 1984.
[7] P. Alface and B. Macq, “Blind watermarking of 3D meshes using robust feature points detection,” in *IEEE International Conference on Image Processing (ICIP)*, vol. 1, Sept. 2005, pp. I–693–6.
[8] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Levy, and M. Desbrun, “Anisotropic polygonal remeshing,” in *SIGGRAPH 03: ACM SIGGRAPH*, 2003, pp. 485–493.
[9] C. David and M. Jean-Marie, “Restricted delaunay triangulations and normal cycle,” in *SCG 03: Proceedings of the nineteenth annual symposium on Computational geometry*, 2003, pp. 312–321.
[10] G. Peyre, “Toolbox graph - a toolbox to perform computations on graph,” Released Jun 2004(Updated Jul 2009).
[11] A. Pressley, *Elementary Differential Geometry*. Springer-Verlag London, 2001.
[12] B. ONeill, *Elementary Differential Geometry*. Academic Press, 1966.
[13] A. Engelbrecht, *Computational Intelligence - An introduction*. 2nd edition, Wiley Publishing, 2007.
[14] H. Demuth, M. Beale, and M. Hagan, *MATLAB Neural Network Toolbox 6: User’s Guide*. The Mathworks, 2009.
[15] J. Vollmer, R. Mencl, and H. Muller, “Improved laplacian smoothing of noisy surface meshes,” in *Computer Graphics Forum*, vol. 18, 1999, pp. 131–138.