An Eigen-Normal Approach for 3D Mesh Watermarking Using Support Vector Machines

Rakhi Motwani, Mukesh Motwani, Frederick Harris, Jr., and Sergiu Dascalu

***Abstract*—The use of support vector machines (SVM) for watermarking of 3D mesh models is investigated. SVMs have been widely explored for images, audio, and video watermarking but to date the potential of SVMs has not been explored in the 3D watermarking domain. The proposed approach utilizes SVM as a binary classifier for the selection of vertices for watermark embedding. The SVM is trained with feature vectors derived from the angular difference between the eigen normal and surface normals of a 1-ring neighborhood of vertices taken from normalized 3D mesh models. The SVM learns to classify vertices as appropriate or inappropriate candidates for modification in order to accommodate the watermark. Experimental results verify that the proposed algorithm is imperceptible and robust against attacks such as mesh smoothing, cropping and noise addition.**

***Index Terms*—3D mesh models, support vector machine, watermarking.**

doi: 10.3969/j.issn.1674-862X.2010.03.007

1. Introduction

The market for 3D models has evolved tremendously over the past few years owing to the widespread applications of 3D models in virtual reality, e-commerce, computer aided design, medical science and entertainment industry, only to name a few application domains. Such widespread use of 3D multimedia has lately attracted the research community to explore watermarking techniques for the copyright protection of 3D models. Watermarking research during the past decade has mostly focused on text, images, audio and video. Machine learning techniques for embedding and extracting watermarks have been widely investigated for images[1]-[3], audio[4],[5], and video[6],[7] domain. Such techniques have yet to be explored for the 3D graphics domain. Towards this end, the presented work aims at exploring the potential of support vector machines (SVM) in 3D watermarking.

Organization of the paper is as follows. A brief theory of SVM and related work are presented in Section 2 and 3. The proposed approach is outlined in Section 4 and experimental results are given in Section 5. Finally, directions for future work are provided in Section 6.

2. Support Vector Machines

SVM classifiers[8] are based on a class of hyperplanes, i.e., a decision surface which characterizes the boundary between the classes of the data. A set of training data or feature vectors is used to create the SVM classifier which predicts the class label of future data sets that is not provided in the training set. The SVM tries to find the optimal hyperplane which gives the largest margin of separation between the classes. The optimal hyperplane is a weighted combination of a subset of the elements of the training data set known as support vectors. A linear decision surface is expressed by the equation:

$$w\mathbf{x}+b=0 \tag{1}$$

where $\mathbf{x}$ is the input vector, b is the bias and w is computed by the following equation:

$$w=\sum_{i=1}^{N}\alpha_i \mathbf{s}_i y_i \tag{2}$$

where α is the weight, $\mathbf{s}_i$ is the support vector, and y is the class label.

A nonlinear decision surface makes use of a kernel function $K(\cdot,\cdot)$ that transforms the data to a higher dimensional space where the data set is separable. The kernel function must satisfy the Mercer's condition[9] and be expressed as a dot product in some space. The nonlinear decision surface is expressed by the equation:

$$\sum_{i=1}^{N}\alpha_i y_i K(\mathbf{s}_i,\mathbf{x})+b=0\,. \tag{3}$$

The equations for the three basic kernels are:

Linear: $K(x_i, x_j) = x_j$

Polynomial: $K(x_i, x_j)= (x_i x_j+c)^d$

Radial basis: $K(x_i,x_j)=\exp\left|-x_i-x_j\right|^2$.

Manuscript presented at 2010 International Conference on Signal Acquisition and Processing (ICSAP 2010), Bangalore, India, February 9-10, 2010; recommended by TPC of ICSAP 2010, revised July 15, 2010.

R. Motwani, M. Motwani, F. Harris, Jr., and S. Dascalu are with the Computer Science & Engineering Department, the University of Nevada, Reno 13644 NV 89507, USA (email: {rakhi, mukesh, fredh, dascalus}@cse.unr.edu).

The proposed approach makes use of SVM as a classifier to identify vertices in the 3D model that are optimal locations for embedding the watermark imperceptibly.

3. Related Work

SVM has not been used so far for watermarking of 3D objects. A review of literatures reveal that SVM has been used for various other applications in 3D such as object recognition, face detection, face recognition, texture classification, head pose estimation, content-based database search, and shape processing. This section reviews such literatures along with related works that utilize SVM for digital media watermarking. The purpose of the review is to determine what feature vectors are used for training the SVM, which kernel function is employed and what classification is achieved by the SVM.

3.1 Use of SVM in Multimedia Watermarking

The authors in [10] described a texture based image watermarking algorithm. A two-class SVM is used to distinguish the image's texture. The proposed method divides the image into blocks, calculates the texture value for each block and uses these texture values for training the SVM. The blocks with class label=1 have a strong texture, and are used to embed the watermark with strong intensity while the blocks with class label= −1 have a weak texture, and are used to embed the watermark with weak intensity. In [11], an SVM was employed by the watermark detector to extract the watermark from digital images. The watermark consists of reference data and a digital signature. The reference data and the watermarked image are used for the training of the SVM. The SVM learns the relationship between the embedded watermark and the watermarked image in order to correctly recover the embedded information bits from a given image.

The authors in [12] used a linear SVM for the extraction of watermark from digital images. The feature vectors used for training the SVM are derived from the watermark embedded positions and consist of a reference watermark and the deviation of the watermarked pixel from the mean of the value of its surrounding pixels. The trained SVM is then used to determine the value of the pixel under consideration during the watermark extraction phase.

In [13], a two-class SVM was used to locate the optimal embedding positions for the watermark in an audio signal. The feature vectors for training are the means of the absolute value of the coarse signal and the local maximal peaks of the detail signal in each wavelet subband of an audio segment. For higher average and peak values, the feature vector belongs to class 1 that represents optimal location for watermark insertion. For lower average and peak values, the feature vector belongs to class 0 that is not suitable for watermark insertion. A radial basis kernel function is used for the SVM.

In [14], a video watermarking approach was proposed by using SVM for watermark embedding and extraction. The watermark comprises of a training sequence and a digital signature. For training the embedding SVM, feature vectors are derived based on the difference of blue channels among a watermarked video frame and its neighbors. The trained SVM then embeds the watermark in the rest of frames. The detector takes frames and the training sequence from the watermark to train the extracting SVM and uses it to extract watermarks from the remaining frames. The final watermark is the average of these extracted watermarks.

3.2 Use of SVM in 3D Graphics

Researchers in [15] classified 3D aerial LiDAR data into four classes (buildings, trees, roads, and grass) using the SVM algorithm. They used five features - normalized height, height variation, normal variation, LiDAR return intensity, and image intensity. Linear, Gaussian, sigmoid, polynomial (degree 2-6) kernels were used for experiments. The authors in [7] used a multi-class SVM for classification of 3D textures using a histogram model. Invariant features are extracted from images of textured surfaces which are transformed by 3D translation and rotation, by means of a non-linear histogram model. These invariant features are used as input feature vectors of the SVM in order to do classification in high-dimensional spaces.

In [16], the authors used SVMs to estimate the 3D head pose in video sequences. The feature vector is a displacement vector derived from a 3D physics-based deformable model created from each pixel's location and intensity from every frame of the given video sequence. This feature vector is used to train a multi-class SVM to classify one of the three head pose angles (i.e. pan, tilt and roll angles of the head). A polynomial function kernel is used for the SVM classifier.

The authors in [17] used a linear SVM for 3D object recognition and proposed an appearance based approach that learns from images of 3D objects in the COIL-100 database. Images are regarded as points of a space of high dimensionality. Each image is converted to gray-scale with the spatial resolution reduced and the image is represented as a vector. An SVM is trained for each pair of 3D objects and a tennis game tournament strategy is used to classify a given test object using the various trained SVMs.

The authors in [18] used SVM to iteratively refine results of a content-based search algorithm for a database of 3D objects. The search result is given as several nearest neighbor objects to the query object. The weighted Euclidean distance measure between 3D object features is used to measure the similarity between objects. The features used are normalized moments estimated based on uniformly distributed random generation of points on the 3D objects' surface. SVM is trained with labeled 3D objects which are returned as results in search of the query object with binary labels representing similarity or dissimilarity from the query object. A parameterized Euclidean distance based linear kernel function is used to adapt the weights of the distance measure causing similar objects to become nearer, and the dissimilar objects to become distant from the query object. Subsequent searches yield an improved set of results returning higher number of similar objects in response to the search query.

Researchers in [19] discussed 3D object classification with SVMs. COIL image library of 3D objects is used for experiments and three different pixel level representations of the images are used as feature vectors. Linear and polynomial kernels are used for the SVM. A multi-class pattern recognition system is obtained by combining binary SVMs. Each SVM is trained as a classifier for one class against another class. In order to classify test data, pair-wise competition between all the machines is performed and in analogy with a tennis tournament, each winner competes against another winner until a single winner remains. This final winner determines the class of the test data.

In [20], an SVM based method was used for shape processing by approximating implicit surfaces from a point cloud data of a 3D object. The point cloud data of a 3D object surface and off-surface points area are used as training data and the SVM regression algorithm uses a radial basis kernel. The authors in [21] introduced an SVM based technique for 3D face recognition. The 3D face shape is normalized and a two-dimensional principal component analysis (PCA) is applied to get principal images which are used as the feature vectors. Classification is carried out by calculating the similarity score between the feature vectors. The SVM classifier is used in choosing the closest match for the face recognition. A radial basis kernel is used.

4. Approach

The proposed approach employs SVM for the watermark embedding phase. Fig. 1 gives an overview of the watermarking system. Since SVM is a supervised learning algorithm, it requires a training stage. The trained SVM is then used in the watermark embedder. Once the SVM is trained, feature vectors extracted from any 3D mesh model can be fed to the SVM classifier to decide which vertices are appropriate for watermark insertion. The watermark extractor retrieves the imperceptible watermark inserted during the insertion stage. Robustness of the watermark is evaluated by comparing the correlation between original watermark and attacked watermark using correlation.

4.1 Feature Vectors

A 3D mesh model, shown in Fig. 2, is represented by a set of vertices and a collection of triangular faces formed by the vertices. The feature vectors used to train the SVM represent the curvature of the patch areas of the 3D mesh

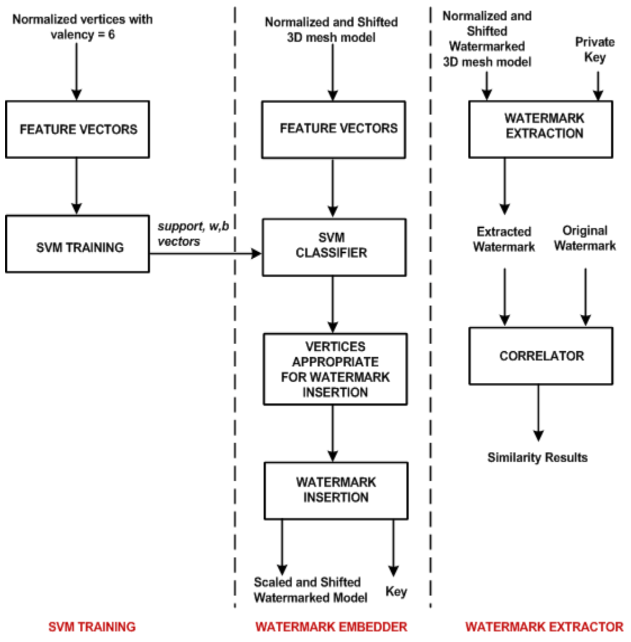

Fig. 1. System block diagram.

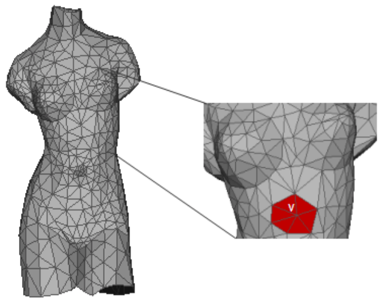

Fig. 2. 3D mesh model and the 1-ring neighborhood (6 shaded triangles around vertex *v*).

model. This curvature measure determines the eligibility of the patch area for watermark addition. The patch area is restricted to an 1-ring neighborhood for each vertex.

The feature vector is a set of angles derived by computing the orientation of the surface normals to the eigen-normal of the triangular faces that form an 1-ring neighborhood for a vertex, as shown in Fig. 4 and 5. The length of the feature vector is equal to the valence of the vertex, which is the count of how many other vertices the vertex is connected to in the 3D model. Since feature vectors used for training of SVM have to be of fixed length, vertices with valence 6 are selected for feature extraction since most of the vertices in a 3D model have valence 6.

The steps for extracting feature vectors are given below:

Step 1: consider a vertex *v* with valence equal to 6 from the 3D mesh model. Let *M* be the number of its adjacent faces which is equal to 6. Find normal's n_i to each face which is formed by *v* and its neighboring vertices, as shown in Fig. 3, by taking the cross-product of any two edges that form the face.

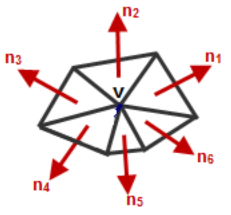

Fig. 3. Surface normals (n_i) for an 1-ring vertex neighborhood.

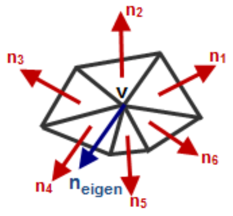

Fig. 4. Surface normals (n_i) and eigen-normal (n_{eigen}) or an 1-ring vertex neighborhood.

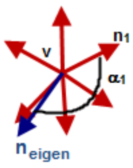

Fig. 5. Angular variation between surface normals and eigen normal.

Step 2: find the eigen normal n_{eigen} of all the above normal's passing through v by computing the eigen-vectors from the eigen-decomposition of the covariance matrix of all these surface normals. The eigen-vector corresponding to the maximun eigen-value constitutes the eigen-normal.

Step 3: now compute angles α_i between each pair of n_i and n_{eigen}.

$$\alpha_j = \cos^{-1}\left(\left(n_i n_{\text{eigen}}\right) / \left(|n_i||n_{\text{eigen}}|\right)\right). \tag{4}$$

Feature vector $\mathbf{F}=[\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6]$ represents the curvature of the 1-ring vertex neighborhood.

4.2 SVM Training

During the training stage, feature vectors are fed to the SVM as input along with a corresponding class label of either 1 (vertex appropriate for watermark addition) or –1 (vertex inappropriate for watermark addition). The class label for each feature vector is determined manually by a human operator. Random amount of information is added to a vertex and if the information added causes perceptible distortion, the vertex with valence 6 is labeled as –1. If the information added is imperceptible the vertex is labeled as 1. 100 sets of vertex rings with different geometrical structures (see Fig. 6) are extracted from 7 normalized 3D objects and labeled appropriately deciding whether to insert the watermark or not. The labelings of output vectors is a manual process thereby transferring human intelligence to the classifier. The output of the training stage is the support vectors, the weight vector **w** and the bias b defining the optimal hyper plane for separation of the feature vectors in to class labels of 1 and –1.

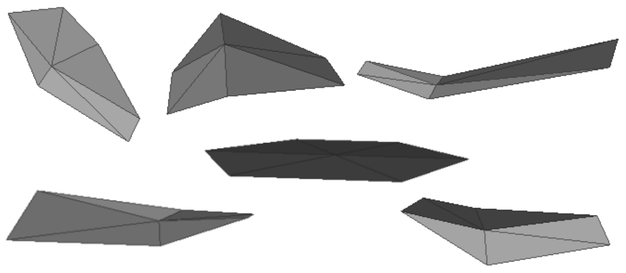

Fig. 6. 1-Rings of different geometries used to train the SVM.

4.3 Watermark Insertion

Initially, the 3D objects are normalized by scaling the size of the mesh to fit in a bounded cube with the vertices of the diagonal of the cube ranging from (–1, –1, –1) to (1, 1, 1) and shifting the centre of mass of the vertices to the origin. The extracted feature vectors are then fed to the SVM classifier which was trained in the earlier stage. The SVM gives an output of –1 (corresponding to the decision that the vertex should not be selected for watermarking) or output of 1 (corresponding to the decision that the vertex should be selected for watermarking). The feature vectors extracted from the 3D model may not have been used in the training set. A random number sequence W (the watermark data) is added to the vertices which have SVM output of 1 according to the following equation:

$$v'(x,y,z) = v(x,y,z) + KW \tag{5}$$

where v' is the watermarked vertex, and K is the scaling factor.

The key stores the vertex indices for which the watermark has been added and also the original values of those vertices. The 3D model is then scaled back to its original size and the center shifted from origin to the original center. The watermark inserted in the 3D mesh models are randomly distributed throughout the model as shown in Fig. 7.

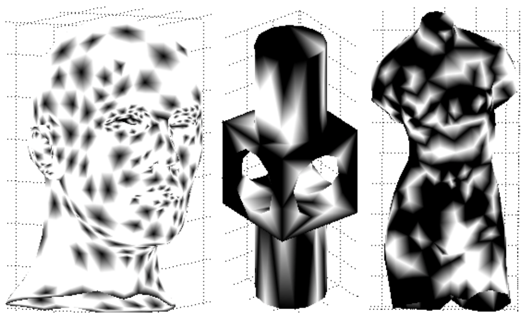

Fig. 7. Watermark locations indicated by white regions (black regions denote vertices that are not modified by the watermarking algorithm).

4.4 Watermark Extraction

To extract the watermark from a given watermarked or attacked 3D model, the model is normalized and scaled to the origin. Using the key generated by the watermark embedding phase, the difference in vertex coordinates determines the watermark. A correlation measure, as defined in (6) and (7), is used to get the degree of similarity between the extracted watermark and the watermark added during the embedding stage.

$$\text{Amount of correlation}=\frac{w+w'\times \text{correlation}}{W} \tag{6}$$

$$\text{Correlation} = \sum_{i=1}^{N}\frac{A_i A_i'}{|A_i||A_i'|} \tag{7}$$

$$A_i = x_i + y_i + z_i\,,\quad A_i' = x_i' + y_i' + z_i' \tag{8}$$

where w is count of vertices not attacked in watermarked model, w' is count of vertices attacked in watermarked model, W is total number of vertices in the watermarked model, x_i, y_i, z_i are co-ordinates of the *i*th vertex in the attacked model, and x_i', y_i', z_i' are co-ordinates of the corresponding *i*th vertex in the watermarked model.

5. Experiments

The algorithm has been implemented in Matlab using the Least Squares SVM Toolbox. This section presents experiments to evaluate the robustness and imperceptibility of the embedded watermark. Fig. 8 shows the original and watermarked 3D models along with various attacks performed on the watermarked 3D models. The Hausdorff distance and vertex signal-to-noise ratio (VSNR) quantify the visual differences between the original and watermarked objects and are very low as shown in Table 1, indicating very good imperceptibility of the watermark.

The following attacks are simulated on the watermarked 3D models. The correlation coefficient between the attacked watermark and the original one is shown in Table 2. A threshold value of 0.7 is used to determine survival of the watermark.

Table 1: Comparison of original model with watermarked model

Model name	Number of vertices	Number of modified vertices	VSNR (dB)	Hausdorff distance
Hand	26000	4026	122.86	0.001729
Baby	5075	1212	108.65	0.004079
Mechanical	175	79	106.13	0.003492
Venus	711	241	124.63	0.001574
Smiley	1026	378	111.73	0.002146
Mannequin	1681	1106	113.57	0.003029

1) Translation, rotation, and scaling. The algorithm is completely resistant to uniform scaling and affine attacks since the model is normalized prior to watermark insertion. Any uniform rotation, translation, and scaling operations on the watermarked models gives correlation coefficient of 1.0.

2) Noise on geometry. Addition of Gaussian noise consists of adding random noise with mean 0 and variance 0.0035 to vertex coordinates (see Fig. 8). A noise level of 100%, i.e. all vertices modified by additive noise, destroys the watermark yielding lower than 0.7 correlation values. However, the watermark survives noise levels of 20% or less.

3) Mesh smoothing. Laplacian smoothing when applied to the watermarked model smooths the sharp edges in the model by applying a low pass gradient filter to the vertices. The watermark does not survive mesh smoothing operations.

4) Cropping. Various experiments that crop sections of the 3D model verified that the watermarked can be recovered even when a part of the mesh is removed, as shown in Fig. 8.

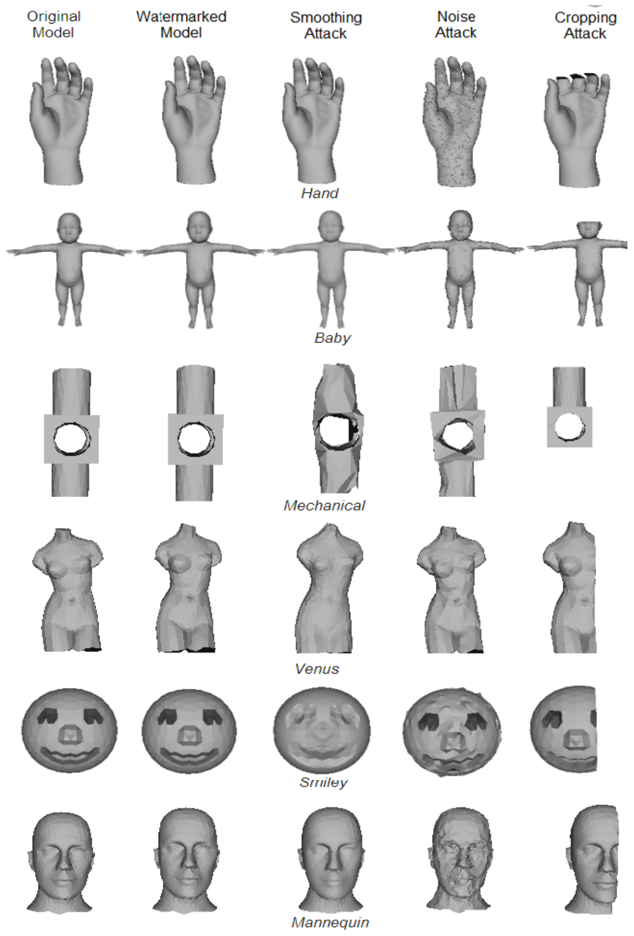

Fig. 8. Attacks on watermarked 3D models.

Table 2: Correlation results for various attacks

3D Model Name	Correlation value		
	Smoothing	Cropping	Noise
Hand	0.4813 (Laplacian smoothing 1 time)	0.8840 (3607 vertices cropped)	0.6542 (Gaussian noise added to 3900 vertices)
Baby	0.6407 (Laplacian smoothing 1 time)	0.8261 (811 vertices cropped)	0.7017 (Gaussian noise added to 761 vertices)
Mechanical	0.6714 (Laplacian smoothing 1 time)	0.7012 (48 vertices cropped)	0.7283 (Gaussian noise added to 175 vertices)
Venus	0.5234 (Laplacian smoothing 1 time)	0.8946 (193 vertices cropped)	0.6719 (Gaussian noise added to 711 vertices)
Smiley	0.4325 (Laplacian smoothing 1 time)	0.9217 (301 vertices cropped)	0.5758 (Gaussian noise added to 153 vertices)
Mannequin	0.5493 (Laplacian smoothing 1 time)	0.9083 (627 vertices cropped)	0.5349 (Gaussian noise added to 1681 vertices)

6. Summary and Future Work

This paper has investigated and illustrated the potential of SVM for embedding a watermark in 3D mesh objects. The performance of the algorithm is heavily dependent on the quality of training feature vectors and the size of the training set. Experimental results show good performance in terms of imperceptibility and robustness for the watermarking algorithm. Future work will include experimenting the use of various kernel functions for the SVM and deriving feature vectors of different lengths for vertices with different valence. SVM will also be evaluated in future work for not only selecting the vertices to be watermark but also determining how much watermark to add to the selected vertex[22]. Using SVM in the watermark detection phase is an additional task to explore.

References

[1] F. Meng, H. Peng, Z. Pei, and J. Wang, "A novel blind image watermarking scheme based on support vector machine in DCT domain," in *Proc. of International Conference on Computational Intelligence and Security*, Genova, Italy, 2008, pp. 16-20.

[2] X.-Y. Wang, H.-Y. Yang, and C.-Y. Cui, "An SVM-based robust digital image watermarking against desynchronization attacks," *Signal Process*sing, vol. 88, no. 9, pp. 2193-2205, 2008.

[3] S.-H. Yen and C.-J. Wang, "SVM based watermarking technique," *Tamkang Journal of Science and Engineering*, vol. 9, no. 2, pp. 141-150, 2006.

[4] X.-Y. Wang, P.-P. Niu, and W. Qi, "A new adaptive digital audio watermarking based on support vector machine," *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 735-749, 2008.

[5] H. Najafi, "A neural network approach to audio data hiding based on perceptual masking model of the human auditory system," *Applied Intelligence*, vol. 27, no. 3, pp. 269-275, 2007.

[6] C.-J. Wang, S.-H. Yen, and P.-S. Wang, "A multimedia watermarking technique based on SVMs," *International Journal of Pattern Recognitionand Artificial Intelligence*, vol. 22, no. 8, pp. 1487-1511, 2008.

[7] M. Li, P. Fu, and S.-H. Sun, "3D texture classification using 3D texture histogram model and SVM," in *Proc. of International Conference on Electronic Measurement and Instruments*, Xian, China, 2007, pp. 2.940-2.943.

[8] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge: Cambridge University Press, 2000.

[9] C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121-167, 1998.

[10] F. Yang and L. Li, "An adaptive, SVM-based watermarking in frequencydomain," in *Proc. of International Conference on Wavelet Analysis and Pattern Recognition*, Hong Kong, China, 2008, pp. 465-469.

[11] Y.-G. Fu, R.-M. Shen, L.-P. Shen, and X.-S. Lei, "Reliable information hiding based on support vector machine," *Informatica*, vol. 16, no. 3, pp. 333-346, 2005.

[12] D. Lakshmi, R. Ganesh, S. Marni, R. Prakash, and P. Arulmozhivarman, "SVM based effective watermarking scheme for embedding binary logo and audio signals in images," in *Proc. of IEEE TENCON*, Hyderabad, India, 2008, pp. 1-5.

[13] X.-Y. Wang, P.-P. Niu, and W. Qi, "A new adaptive digital audio watermarking based on support vector machine," *Journal of Network and Computer Applications*, vol. 31, no. 4, pp. 735-749, 2008.

[14] S.-H. Yen, H.-W. Chang, C.-J. Wang, P. Wang, and M.-C. Chang,"A scene-based video watermarking technique using SVMs," in *Proc. of International Conference on Pattern Recognition*, Tampa, USA, 2008, pp. 1-4.

[15] S. Lodha, E. Kreps, D. Helmbold, and D. Fitzpatrick, "Aerial lidardata classification using support vector machines (SVM)," in *Proc. of the Third International Symposium on 3D Data Processing, Visualization, and Transmission*, Chapel Hill, USA, 2006, pp. 567-574.

[16] M. Krinidis, N. Nikolaidis, and I. Pitas, "3D head pose estimation using support vector machines and physics-based deformable surfaces," in *Proc. of International Symposium on Signal Processing and Its Applications*, Sharjah, 2007, pp. 1-4.

[17] M. Pontil and A. Verri, “Support vector machines for 3D object recognition,” *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 6, pp. 637-646, 1998.

[18] M. Elad, A. Tal, and S. Ar, “Directed search in a 3D objects database using SVM,” *HP Labs*. [Online]. Available: www.hpl.hp.com/techreports/2000/HPL-2000-20R1.pdf.

[19] D. Roobaert and M. V. Hulle, “View-based 3D object recognition with support vector machines,” in *Proc. of IEEE International Workshop on Neural Networks for Signal Processing*, Madison, USA, 1999, pp. 77-84.

[20] F. Steinke, B. Scholkopf, and V. Blanz, “Support vector machines for 3D shape processing,” *Eurographics, Computer Graphics Forum*, vol. 24, no. 3, pp. 285-294, 2005.

[21] M. Mousavi, K. Faez, and A. Asghari, “Three dimensional face recognition using SVM classifier,” in *Proc. of IEEE/ACIS International Conference on Computer and Information Science*, Portland, 2008, pp. 208-213.

[22] R. Motwani, M. Motwani, and F. Harris Jr., “Using radial basis function networks for watermark determination in 3D models, in *Proc. of IEEE INDICON*, Gandhinagar, India, 2009, pp. 1-4.

Rakhi Motwani received her B.E. degree from the University of Pune, India, in 2000, and an M.S. degree from the University of Nevada, Reno (UNR), USA, in 2002, both in computer science. She received her Ph.D. degree in computer science and engineering from UNR, in 2010. She is currently a post-doctoral research associate with the Department of Computer Science and Engineering, UNR. She is an IEEE member. Her research interests lie in information hiding techniques and applied artificial intelligence.

Mukesh Motwani received his B.E. degree in electronics engineering from the University of Pune, India, in 1999, and an M.S. degree from UNR, in 2002, in computer science. He is presently pursuing his Ph.D. degree in computer science and engineering at UNR and works as a solutions architect to provide consulting services to the IT industry. His research interests lie in service oriented architecture, digital rights management systems, watermarking, and applied computational intelligence.

Frederick Harris Jr. is currently a professor with the Department of Computer Science and Engineering and the Director of the High Performance Computation and Visualization Lab at the University of Nevada, Reno, USA. He received his B.S. and M.S. in mathematics and educational administration from Bob Jones University in 1986 and 1988 respectively. He got his second M.S. and Ph.D. in computer science from Clemson University in 1991 and 1994, respectively. He is a member of ACM, IEEE, and ISCA. His research interests are in parallel computation, graphics and virtual reality, and bioinformatics.

Sergiu Dascalu is currently an associate professor with the Department of Computer Science and Engineering and the Director of the Software Engineering Laboratory at the University of Nevada, Reno. He received his Ph.D. degree in computer science from Dalhousie University, Canada in 2001 and M.S. degree in automatic control and computers from the Polytechnic of Bucharest, Romania in 1982. His research interests are in software engineering and human-computer interaction.