Fragile Watermarking of 3D Models Using Genetic Algorithms

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

***Abstract*—This paper describes a novel algorithm for fragile watermarking of 3D models. Fragile watermarking requires detection of even minute intentional changes to the 3D model along with the location of the change. This poses a challenge since inserting random amount of watermark in all the vertices of the model would generally introduce perceptible distortion. The proposed algorithm overcomes this challenge by using genetic algorithm to modify every vertex location in the model so that there is no perceptible distortion. Various experimental results are used to justify the choice of the genetic algorithm design parameters. Experimental results also indicate that the proposed algorithm can accurately detect location of any mesh modification.**

***Index Terms*—3D mesh models, fragile watermarking, genetic algorithms, SNR.**

doi: 10.3969/j.issn.1674-862X.2010.03.008

1. Introduction

3D digital content has grown exponentially in the past few years. The designers exhibit, share, and sell this valuable artwork as 3D models or meshes. However, the duplication and unauthorized modification of these meshes have made watermarking an essential part of copyright protection while selling these models. Watermarking can be broadly classified into two types. Robust Watermarking is used to prove ownership. Thus, in such a type of watermarking scheme, the watermark should be preserved even after attacks. On the other hand, fragile watermarking is used to authenticate integrity of the model. It is required that fragile watermarking algorithm also locates the regions where the mesh has been modified intentionally. This can be achieved if the position of every vertex in the model is modified or watermarked. Thus, fragile watermarking is a commonly used technique for tamper detection. However, watermarking all the vertices in a 3D model poses a challenge since perceptible distortion can be caused. The proposed algorithm successfully solves that challenge by watermarking all the vertices in a model without causing perceptible distortion.

A comprehensive survey of 3D model watermarking techniques has been done by [1]. There has been a significant amount of research to embed watermarks in images, video sequences, and audio sequences using genetic algorithms (GA). Reference [2] used GA to find the optimum embedding density to modify the DCT coefficients of the host image for image watermarking. GAs have also been used in frequency domain, DCT and DWT domain for image watermarking[3]-[6]. The fitness function is peak signal to noise ratio (PSNR) for imperceptibility of the watermark and normalized cross correlation (NCC) for the robustness of the watermark. GA has also been used in a dual image watermarking scheme[7]. The above scheme employs insertion of a robust watermark into a secondary image which acts as the fragile watermark for the host image. Embedding of fragile watermark uses a watermark embedding factor which is obtained using GA. Thus, the objective of the GA is to maximize PSNR. Reference [8] applied GA for fragile watermarking of images. In this algorithm, the block edge characteristics of an image are used and the edge information is used as the fitness function parameter. In audio watermarking[9], a pseudo noise (PN) sequence is embedded in the audio sequence. During extraction, GA is used with the fitness function being the cross correlation between estimated PN sequence and the spread spectrum. GAs have never been used for fragile watermarking of 3D models. The novelty of this paper lies in the use of GAs to generate a fragile watermark, which is inserted in all the vertices of the 3D model.

Genetic algorithms are a branch of evolutionary algorithms that use evolution and Darwin's theory of survival of the fittest as a source of inspiration to solve optimization problems[10]. In genetic algorithm, all the possible solutions to solve the given optimization problem are called chromosomes. Each generation has a specific number of chromosomes, which constitute the population of that generation. The fundamental block of GA is the fitness function. The fitness function defines the parameter that has to be optimized. Each chromosome is evaluated using the fitness function and returns a value known as the

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.

M. C. Motwani, R. C. Motwani, and F. C. Harris, Jr. are with University of Nevada, Reno, 13644 NV 89507, USA (e-mail: mcmotwani@gmail.com, rcmotwani@gmail.com, fredh@cse.unr.edu).

Color versions of one or more of the figures in this paper are available online at http://www.intl-jest.com.

fitness value of that chromosome. According to the best fitness values, some chromosomes are selected to reproduce the next generation. The most common operators used for reproduction are selection, mutation and crossover operators.

2. Proposed Approach

2.1 3D Model Representation

A 3D mesh consists of a vertex list, which gives the co-ordinates in 3D space of every vertex in the model, and a face list, which describes how the vertices are connected to each other. Each vertex in the model is represented by (*X*, *Y*, *Z*) coordinates in the Cartesian axis and also represents a chromosome. The 1-ring neighborhood of a vertex V is defined as the surfaces formed by that vertex V with its neighbors as shown in Fig. 1.

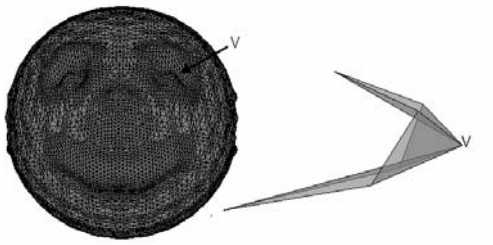


Fig 1. Smiley triangular mesh model and one ring neighborhood of vertex V.

2.2 Watermark Generation and Insertion

Normalization of 3D models as a pre-processing step before insertion of watermark makes the watermark resilient to modifications in the 3D model due to affine and scaling transformations. The centre of mass of the 3D model is shifted to the origin and the model is scaled to fit in a unit cube. For each vertex in the 3D model, the 1-ring neighbourhood for the vertex is extracted. Modifying the coordinate position of centre vertex of the 1-ring neighborhood can be considered as adding random noise with reference to the original location of the vertex. Thus, the watermark added to a vertex is equivalent to noise being added to the vertex. The amount of distortion in the 1-ring can be measured by computing the signal to noise ratio (SNR)[11] as follows:

$$\mathrm{SNR} = \frac{\sum_{i=1}^{N}\left(X_i^2 + Y_i^2 + Z_i^2\right)}{\sum_{i-1}^{N}\left[\left(X_i - X_i'\right)^2 + \left(Y_i - Y_i'\right)^2 + \left(Z_i - Z_i'\right)^2\right]} \qquad (1)$$

where N is the number of vertices in the 1-ring neighborhood of the centre vertex including the centre vertex, X_i, Y_i, and Z_i are the Cartesian coordinates of the vertices in the 1-ring neighborhood including the centre vertex, X_i', Y', and Z_i' are the modified coordinate of the centre vertex.

The objective of the genetic algorithm is to maximize signal to noise ratio (SNR) for the 1-ring neighborhood. Fitness function chosen is

$$\text{Fitness function} = \frac{1}{\mathrm{SNR}}. \qquad (2)$$

Genetic algorithm is used to compute the near optimal value of the amount of watermark to be added to the centre vertex of the 1-ring neighbourhood without causing any perceptible distortion. All the vertices in the 3D model are watermarked using GA. The model is then scaled back to the original size and shifted to its original position as well.

The choice of fitness function determines what parameter GA is going to optimize. Chromosome with the best fitness value at the end of pre-determined number of generations is considered as the optimized output of the algorithm. The best fitness value from a pool of chromosomes corresponds to the chromosome with the least SNR.

2.3 Watermark Extraction

In non-blind approach, original model is subtracted from the watermarked model and thus the watermark is extracted. However, this approach is not restricted to non-blind watermarking. If a key is used to store the vertex index and the amount of watermark added by the above proposed algorithm using GA, the original model is not required.

2.4 Genetic Algorithm Parameters

Each chromosome is represented by (*X*, *Y*, *Z*) cartesian coordinates and a population of 100 such chromosomes is initially created near the centre vertex of the 1-ring neighborhood. The uniform creation function creates a normally distributed random population within the initial range. The initial range defines the upper bound and the lower bound of the *X*, *Y*, and *Z* co-ordinates while creating the initial population. These limits have been set to the maximum and the minimum value of the respective co-ordinates of the 1-ring. This ensures that the optimal chromosome does not move outside the 1-ring, thus avoiding perceptible distortion. The selection operator used is stochastic uniform. Stochastic uniform is used because it considers a certain value below which it does not allow to reproduce, i.e., it follows the basic rule of GA that ability to reproduce is directly proportional to its fitness value. In this type of selection, a threshold value is taken and this threshold value is equal to 1/no. of parents. This threshold value is also called the step size and if the fitness value of any chromosome lies below this step size, it is considered to be weak and hence not allowed to reproduce. Similarly, if the fitness value of a chromosome is greater than or equal to *M* times the threshold value, then that chromosome is selected to reproduce *M* times. *M* is determined by

$$M = \text{floor (fitness value/step size)}. \qquad (3)$$

Table 1: Genetic algorithm parameters

Population Size	100	
No. of generations	20 generations	
Initial range	Upper bound	Max value of the coordinates of 1-ring
	Lower bound	Min value of the coordinates of 1-ring
Creation function	Uniform	
Selection operator	Stochastic uniform	
Crossover operator	Scattered crossover	
Crossover rate	0.8	
Mutation operator	Uniform mutation	
Mutation rate	0.001	
Elite count	2	
Fitness threshold value	10^{-9}	

Scattered crossover is used. The mutation operator used is uniform mutation operator. In this type of mutation operator, the *X*, *Y*, and *Z* co-ordinates of the chromosomes are randomly replaced by a value within the range. Stopping criteria is met when the algorithm encounters one of the following 2 conditions.

1) Generation limit, i.e., the number of specified generations is exceeded, or

2) Change in the fitness values between consecutive generations is less than the fitness threshold value.

The values of the above described parameters used in the algorithm have been tabularized in Table 1.

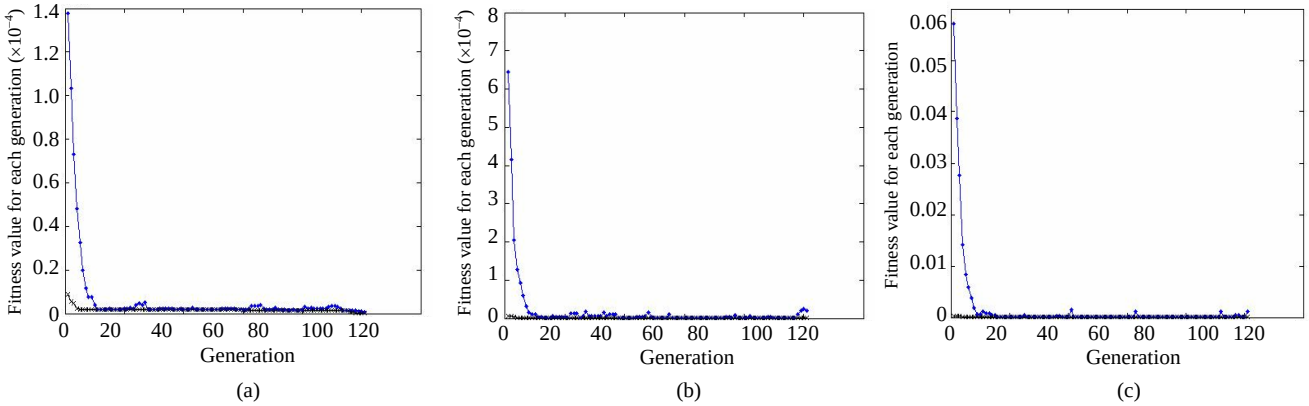
3. Experimental Results

3.1 Population Size and Number of Generations

The algorithm is tested with many different shapes of 1-ring neighbourhoods for deciding the parameters for the GA. The advantage of using 1-ring representation of the vertex is that it is easy to visualize the change in location of the vertex and the computation of distortion is local to the 1-ring vertex. Different sets of experiments were conducted to determine the population size and the number of generations. Black markers in the figures correspond to the best fitness value from the pool of chromosomes for each generation whereas the mean fitness value is denoted by blue markers. 4 sets of experiments are performed on 3 models to determine the optimum combination of population size and the number of generations.

Fig. 2 to Fig. 5 and Table 2 and Table 3 indicate that a population size of 100 run for 20 generations is the most optimal value. When a GA is run for 20 generations with a population size of 20, the perceptible distortions is high. This is because the initial pool of chromosomes is scattered randomly within the range. Since the population size is just 20, the chances of these chromosomes being closer to the centre vertex of the 1-ring are very low. When the algorithm with these parameters is run for only 20 generations, there is not enough time for the GA to converge at a value close to the centre vertex. When the GA is run for 100 generations with population size of 100, the algorithm is computationally expensive. As seen in the Fig. 6, the perceptible distortion with these parameters is very low. But the same degree of low perceptible distortion is achieved with a population size of 100 run for just 20 generations and this also decreases the computational costs. When the population size is 20 and is run for 100 generations, the initial pool of chromosomes is once again scattered within the range and the chances of them being close to the centre vertex are very low. Thus, over a period of 100 generations, much of the newer chromosomes are produced as a result of mutation, i.e., mutation will take place at least 100 times, once for each generation. Mutation is replacement of a chromosome with a random value. This random value may or may not have a good fitness value. Thus, as the results show, the performance of this combination is not as good as the performance of the combination of population size 100 run for 20 generations.

The graphs in Fig. 2 to Fig. 5 show that there is no considerable change in the "*best fitness value*" after 20 generations. Thus, the parameters chosen for embedding the watermark were population size=100 and number of generations=20.



(a) (b) (c)

Fig. 2. In running for 100 generations with a population size of 100, the change in best fitness value and the mean fitness value: (a) for vertex index 10 of the Smiley, (b) for vertex index 10 of the Mannequin, and (c) for vertex index 20 of the Mechanical.

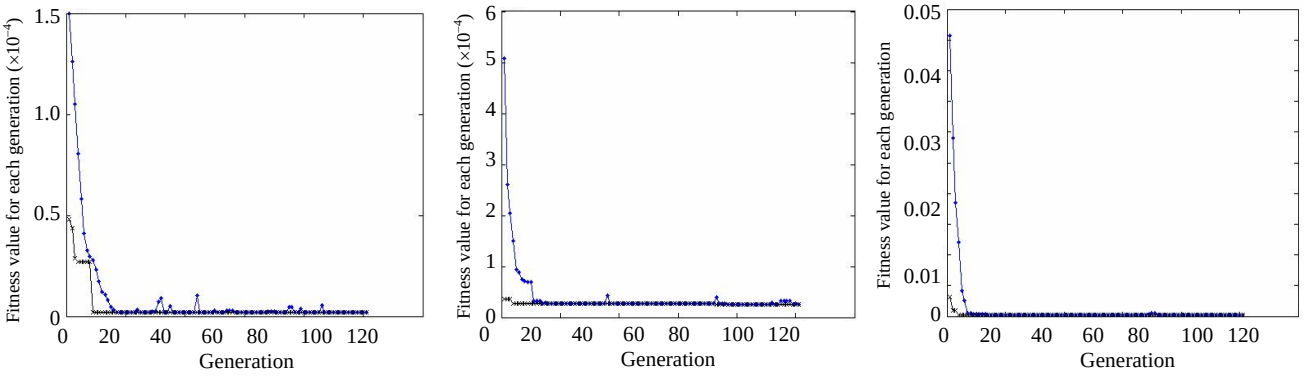


Fig. 3. In running for 100 generations with a population size of 20, the change in best fitness value and the mean fitness value: (a) for vertex index 10 of the Smiley, (b) for vertex index 10 of the Mannequin, and (c) for vertex index 20 of the Mechanical.

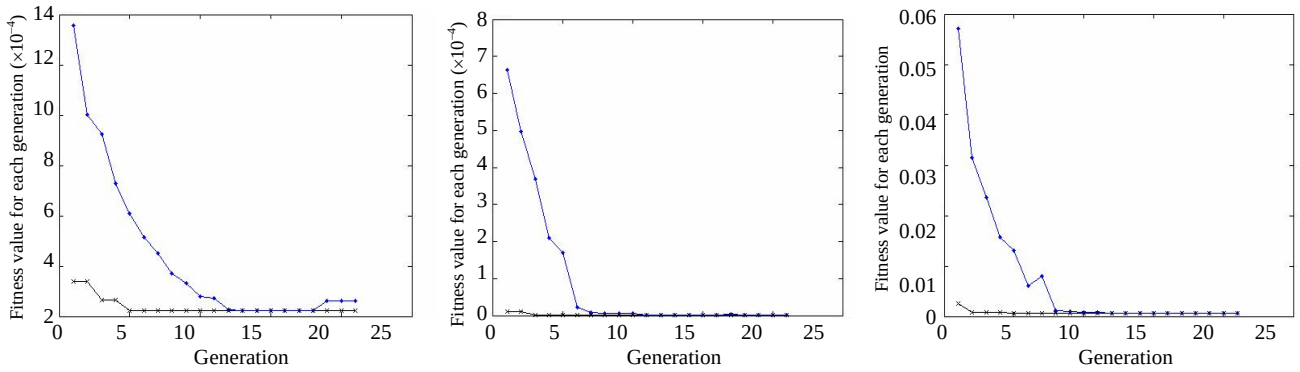


Fig. 4. In running for 20 generations with a population size of 20, the change in best fitness value and the mean fitness value: (a) for vertex index 10 of the Smiley, (b) for vertex index 10 of the Mannequin, and (c) for vertex index 20 of the Mechanical.

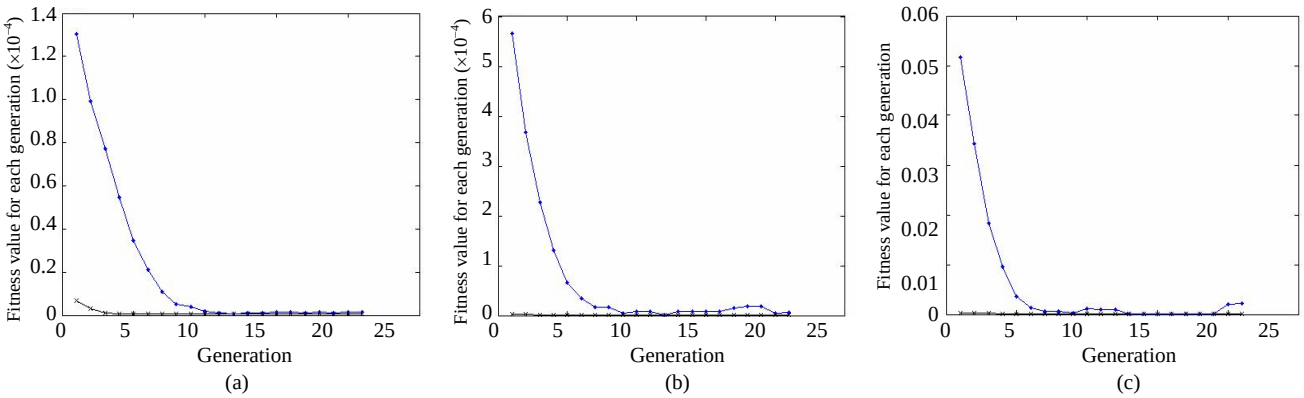


Fig. 5. In running for 20 generations with a population size of 100, the change in best fitness value and the mean fitness value: (a) for vertex index 10 of the Smiley, (b) for vertex index 10 of the Mannequin, and (c) for vertex index 20 of the Mechanical.

Table 2: Average time per vertex

Model	Mannequin (s)	Smiley (s)	Mechanical (s)
Population size 100 run for 100 generations	1.340	1.330	1.090
Population size 20 run for 100 generations	0.420	0.480	0.325
Population size 20 run for 20 generations	0.240	0.200	0.110
Population size 100 run for 20 generations	0.480	0.320	0.285

Table 3: Vertex signal to noise ratio

Model	Mannequin (dB)	Smiley (dB)	Mechanical (dB)
Population size 100 run for 100 generations	144.30	146.00	126.73
Population size 20 run for 100 generations	117.00	119.00	101.34
Population size 20 run for 20 generations	103.00	97.88	82.15
Population size 100 run for 20 generations	126.60	121.25	102.81

3.2 Analysis of Flat and Cylindrical Surfaces

The algorithm should insert a watermark in 3D models that have flat surfaces as well. This is difficult because in that condition the chromosome has to move only on a flat surface on a 2D plane. Since the fitness function used is the SNR, every time the chromosome moves out of the plane, the SNR decreases. Thus, the objective of the GA is to maximize this SNR, due to which the vertex, V stays in the 2D plane, as illustrated in Fig. 6.

Cylindrical surfaces are formed by the intersection of 2 planes. Thus while watermarking the vertex of the cylinder, it is required that the chromosome should move along the line of intersection of these two planes. Mechanical Fig. 7 (e) is a combination of both cylindrical as well as flat surfaces. Fig. 7 (e) shows genetic algorithm watermarks the mechanical model without any perceptible distortion. Thus, the GA works for flat as well as cylindrical surfaces.

3.3 Watermarking the Models

The algorithm was run with a population size of 100 for 20 generations. The algorithm was tested for 5 different models: bunny, mannequin, smiley, horse, and mechanical. The results in Table 4 and Fig. 8 indicate there is no perceptible distortion between watermarked and original models.

3.4 Attacks on the Model

The attacks for fragile watermarking of 3D model differ from the attacks for robust watermarking. The attacks on the fragile watermarked 3D model deal only with unauthorized modification of any region in the model.

Due to normalization of the model as a preprocessing step, the watermark is not affected by attacks such as translation, rotation and scaling as shown in Fig. 9 to Fig. 11. Correlation of 100% is obtained for affine transformations. Note that the model has not been tampered with since there are no colored (red) patches in Fig. 9 (c).

The other type of attacks are deformation or cropping attacks, where the vertices are deformed as shown in Fig. 10 (b). In such a case, the algorithm is able to find out the location at which the deformation or cropping has taken place as shown in Fig. 10 (c).

Apart from these attacks, simultaneous modifications at more than one location can be detected due to the fragile watermarking as shown in Fig. 12.

Table 4: The algorithm test results for 5 different models

Model	Bunny	Mannequin	Smiley	Horse	Mechanical
No. of vertices	17446	6743	8194	19851	175
No. of vertices watermarked	17446	6743	8194	19851	175
SNR (dB)	137.75	126.64	121.5	132.5	102.81
Average time per vertex (s)	0.43	0.48	0.32	0.37	0.285

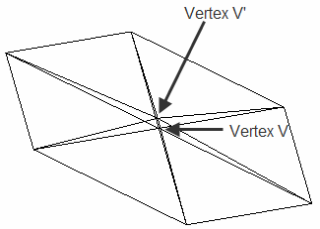


Fig. 6. Vertex 10 of the cube: V moves to V′ on the surface of the plane, preventing perceptible distortion.

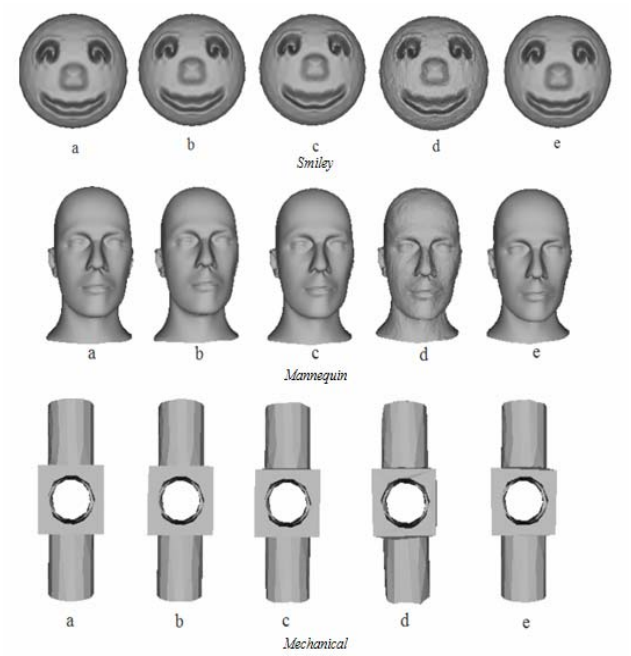


Fig. 7. Watermarked 3D models: (a) original model, (b) watermarked model with a population size of 100, run for 100 generations, (c) watermarked model with a population size of 20, run for 100 generations, (d) watermarked model with a population size of 20, run for 20 generations, and (e) watermarked model with a population size of 100, run for 20 generations.

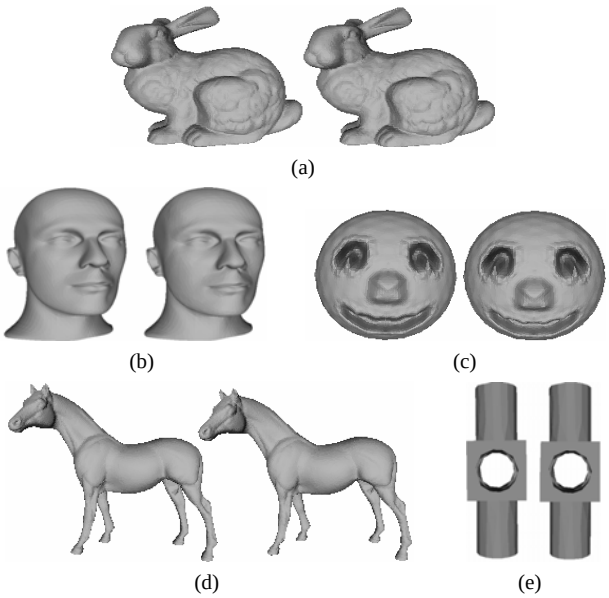
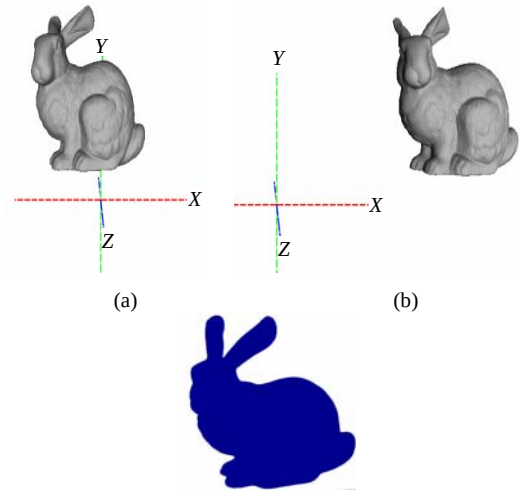


Fig. 8. Comparison between watermarked and original models: (a) bunny, (b) mannequin, (c) smiley, (d) horse, and (e) mechanical.



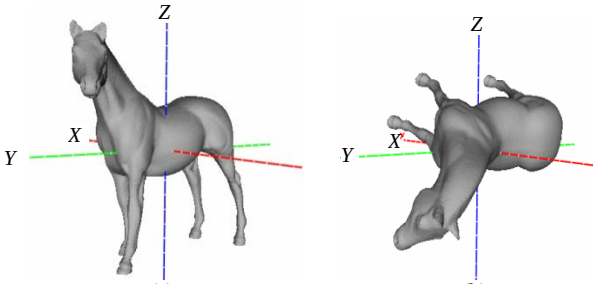
(a) (b)

(c)

Fig. 9. Attacks on bunny: a) watermarked bunny, (b) translated model, and (c) tamper region detection.

(a) (b) (c)

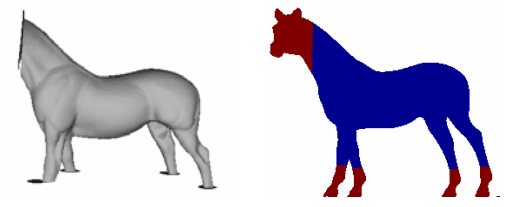
Fig. 10. Attacks on bunny: (a) original bunny, (b) deformed bunny, and (c) modified region shown by colored (red) patch.



(a) (b)

(c)

Fig. 11. Attacks on horse: (a) original horse, (b) rotate by 250˚, and (c) no red patches indicating the model has not been modified.



(a) (b)

Fig. 12. Modification detection: (a) horse deformed and (b) deformed regions.

4. Conclusions

In this paper, a fragile watermarking technique using genetic algorithms has been proposed. The algorithm generates and embeds a watermark in each and every vertex of the model. The use of signal to noise ratio as the fitness function prevents any perceptible distortion in the model. Also, genetic algorithms are known to be slow, but the method of premature convergence used in this paper makes the proposed algorithm computationally inexpensive. The premature convergence was achieved by running the algorithm with a population size of 100 for 20 generations and was found to provide satisfactory results. The analysis of the working of this algorithm on flat and cylindrical surfaces has also been covered. The algorithm also detects any unauthorized structural modifications in the model.

References

[1] J. L. Dugelay, A. Baskurt, and M. Daoudi, *3D Object Processing: Compression, Indexing and Watermarking*, Chichester, UK: Wiley, 2008, ch. 4.

[2] Y. Chen and L. Peng, "Optimal image watermark using genetic algorithm and synergetic neural network," in *Proc. of the Second International Conference on Intelligent Computation Technology and Automation*, Changsha, 2009, vol. 3, pp. 209-212.

[3] F. Y. Shih and Y.-T. Wu, "A novel fragile watermarking technique," in *Proc. of IEEE International Conference on Multimedia and Expo*, Taipei, 2004, pp. 875-878.

[4] Z. Wei, J. Dai, and J. Li, "Genetic watermarking based on DCT domain techniques," in *Proc. of Canadian Conference on Electrical and Computer Engineering*, Ottawa, 2006, pp. 2365-2368.

[5] T. E. Areef, H. S. Heniedy, and O. M. O. Mansour, "Optimal transform domain watermark embedding via genetic algorithms," in *Proc. of the 3rd International Conference on Information and Communications Technology*, Cairo, 2005, pp. 607-617.

[6] T. Kumaran and P. Thangavel, "Genetic algorithm based watermarking in double-density dual-tree DWT," in *Proc. of International Conference on Wavelet Analysis and Pattern Recognition*, Hong Kong, 2008, vol. 2, pp. 585-590.

[7] D. R. ElShafie, N. Kharma, and R. Ward, "Parameter optimization of an embedded watermark using a genetic algorithm," in *Proc. of the 3rd International Symposium on Communications, Control and Signal Processing*, St Julians 2008, pp. 1263-1267.

[8] S.-K. Lee and Y.-S. Ho, "Fragile watermarking scheme using a simple genetic algorithm," in *Proc. of International Conference on Consumer Electronics*, Las Vegas, 2002, pp. 190-191.

[9] S. Sedghi, H. R. Mashhadi, and M. Khademi, "Detecting hidden information from a spread spectrum watermarked

signal by genetic algorithm," in *Proc. of IEEE Congress on Evolutionary Computation*, Vancouver, 2006, pp.173-178.

[10] M. Melanie, *An Introduction to Genetic Algorithms*, Cambridge, USA: The MIT Press, 1999, ch. 1.

[11] J. Bennour and J.-L. Dugelay, "Toward a 3D watermarking benchmark," in *Proc. of IEEE 9th Workshop on Multimedia Signal Processing*, Crete, 2007, pp. 369-372.

Mukesh Motwani received his B.E. degree in electronics engineering from the University of Pune, India, in 1999, and 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 include service oriented architecture, digital rights management systems, watermarking, and applied computational intelligence.

Rakhi Motwani received her B.E. degree from the University of Pune, India, in 2000, and M.S. degree from the University of Nevada, Reno (UNR), 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.

Fredrick Harris, Jr. is currently a professor in 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 parallel computation, graphics and virtual reality, and bioinformatics.