Reversible Watermarking of 3D Mesh Models by Prediction-error Expansion

Hao-tian Wu, Jean-Luc Dugelay

Multimedia Communications Department,
Eurecom Institute, 2229, Route des Crêtes,
06560 Valbonne Sophia Antipolis, France
E-mail: {Haotian.Wu, Jean-Luc.Dugelay}@eurecom.fr

***Abstract*—In this paper, a reversible watermarking algorithm is proposed for 3D mesh models based on prediction-error expansion. Firstly, we predict a vertex position by calculating the centroid of its traversed neighbors. Then the prediction error, i.e. the difference between the predicted and real positions, is expanded for data embedding. So only the vertex coordinates are modified to embed a watermark into the mesh content without changing the topology. We further reduce the distortion by adaptively choosing a threshold so that the prediction errors with too large magnitude will not be expanded. The chosen threshold value and critical location information should be saved in the watermarked mesh to guide the recovery process. The experiments show that the original mesh can be exactly recovered and consequently our algorithm can be used for symmetric or public key authentication of 3D mesh models.**

I. Introduction

Recently, quite a few reversible watermarking algorithms (e.g. [1]-[10]) have been proposed to make the data embedding process reversible. In particular, one should be able to recover the original object after extracting the embedded watermark. It is required that no specific knowledge of the original object is known in the recovery process except those can be learnt from the watermarked object. Moreover, the data format and file size of the original object should not be changed by the embedding process. Since the early reversible watermarking algorithms can only provide a limited capacity, the recently proposed approaches are aiming at embedding more data into a cover with less distortion.

To prevent the embedding from definitively modifying a host (i.e. the original object), a feasible way is to compress it by adopting some lossless techniques to make room for the embedded data, see in [3], [5]. But the efficiency of this method depends on the redundancy in the original object that can be exploited for lossless compression. Alternatively, another method is based on difference expansion to make the embedding process reversible, which is proposed by Tian in [7]. This technique takes advantage of the redundancy in data representation, i.e. two values within a predefined range can be represented in the same size. Further, it can be combined with lossless compression to achieve better performances.

In the original difference expansion algorithm as shown in [7], a pair of pixel values are simultaneously changed to expand their difference while the mean value is maintained. In the case that the modified values would exceed the allowed range (e.g. the integers within $[0, 255]$ for a grayscale image), the expansion can not be performed. So the location information called location map should be saved in the watermarked image to guide the exact recovery of the original picture. To take advantage of the inter-independence between pixels, two neighboring pixels are chosen to form a pair because of the high probability that their difference is small.

Alternatively, the inter-dependency between the neighboring elements can be exploited by prediction. In Thodi and Rodriguez's work [9], the unprocessed neighbors of a pixel are utilized for prediction and the difference between the predicted and real values, i.e. the prediction error, is expanded for data embedding. To recover the original image, the embedded data should be inversely extracted while the previously restored pixel values are used for prediction. Since one bit value is embedded by modifying one pixel value, high capacity is achievable with the prediction-error expansion method. Further, our study shows that it is unnecessary to perform the embedding and recovery processes in the inverse orders, especially when all the prediction errors can be expanded. Different from the algorithm in [9], we predict an element with the neighbors that have been traversed and their original values are used to fully exploit the inter-dependencies. In the recovery process, the elements are recovered in the same order as in the embedding process while the restored values are used to predict the next one.

The former work on reversible watermarking of 3D mesh models have been reported in [11]-[13]. In [11], invertible authentication of 3D meshes is first introduced by combining a public verifiable digital signature protocol with the embedding method in [14], which appends extra faces and vertices to the original mesh. After extracting the embedded signature, the appended faces and vertices can be removed on demand to reproduce the original mesh with a secret key. In [12], a fragile watermarking algorithm is proposed by saving the modulation information in the watermarked mesh model. Despite that the original mesh can only be approximately recovered, the algorithm is limited to symmetric authentication. In [13], a reversible data hiding algorithm is proposed in the predictive vector quantization (VQ) domain to embed data into a mesh stream. Although the compressed mesh can be recovered, it is slightly different from the original one because the VQ compression is lossy.

978-1-4244-2295-1/08/$25.00 © 2008 IEEE

In this paper, we apply a new prediction-error expansion algorithm to 3D meshes. Different from [11], our algorithm does not generate new face or vertex so that the topology is unchanged. By adopting a mesh traversal strategy, a vertex position is predicted by calculating the centroid of its traversed neighbors, i.e. the vertices directly connected and having been previously traversed. By comparing the predicted and real positions, the prediction error can be obtained and expanded to embed one bit value in each modified coordinate. To reduce the distortion, a threshold is adaptively chosen and the prediction errors with magnitude larger than it will not be expanded. We also compare the proposed prediction-error expansion algorithm with the one in [9] in terms of suitability for meshes. The experiments show that the original mesh model can be exactly recovered while the extracted watermark can be used for symmetric or public-key authentication.

The rest of this paper is organized as follows: In the following section, a new prediction-error expansion algorithm is presented and applied to 3D mesh models. We also apply the algorithm in [9] to meshes and compare it with the newly proposed one. The performance of the reversible watermarking algorithm for meshes is evaluated in Section III. Finally, a conclusion is drawn in Section IV.

II. Prediction-error Expansion for 3D Meshes

A. A New Prediction-error Expansion Algorithm

In [9], the prediction-error expansion is introduced as a complement to the difference expansion method in [7] when high information rate is preferred. A generalized prediction-error expansion algorithm can be given as follows: Given an element with the integer value x and a set of its neighbors S, we can use S to predict the value of x. Suppose the predicted integer value is $\bar{x}$, the prediction error is $d = x - \bar{x}$. To embed a bit value, the element value is modified from x to $\tilde{x}$ by

$$\tilde{x} = x + d + b, \tag{1}$$

where b is a bit value to be embedded and considered as an integer (1 or 0). The difference $d = x - \bar{x}$ is interpreted as expandable if the modified pixel value $\tilde{x}$ is still representable. Given that the same prediction can be made in the recovery process, the original element value x and the embedded bit value b can be obtained by

$$\begin{cases} b = \tilde{x} - \bar{x} - 2 \times \lfloor \frac{\tilde{x} - \bar{x}}{2} \rfloor \\ x = \tilde{x} - \lfloor \frac{\tilde{x} - \bar{x}}{2} \rfloor - b \end{cases}. \tag{2}$$

In the recovery process, the original values of S should be used for prediction to generate $\bar{x}$.

The way we adopt for prediction-error expansion is shown in Fig. 1 when taking the pixels in a digital image for instance. The one at the northwest corner is unchanged if the pixels are scanned in the raster order, i.e. by rows from top to bottom and within each row from left to right. Then each prediction is based on the neighbors that have been previously processed, i.e. the upper, left and upper-left ones. Since the pixel values will be modified to expand the prediction error, the original values are employed in the prediction to fully exploit the inter-dependency between the neighboring pixels. In the recovery process, the pixel values that have been modified for data embedding are recovered and the restored values are further used to predict the subsequent ones.

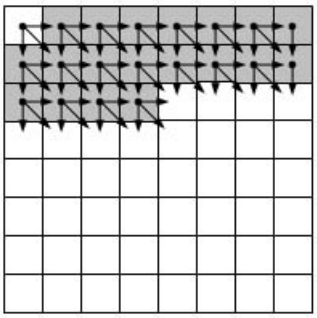


Fig. 1. The original values of the previously processed pixels are used to predict the subsequent neighbors.

For digital images, the range of pixel values is limited so that some prediction errors are not expandable. Usually, a threshold is chosen to expand the prediction errors with small magnitude only. Therefore, the location map representing the locations where the prediction errors are expanded should be obtained before the embedding process. In this paper, we will not detail the implementation of the proposed algorithm on digital images. Instead, we apply it to 3D mesh models by expanding the difference between the predicted and real positions of a vertex. Since the vertex coordinates are represented by floating or fixed point numbers, the representation range is not limited so that all the prediction errors can be expanded.

B. The Embedding Process

A polygonal mesh consists of a set of vertices, as denoted by $\mathbf{V} = \{\mathbf{v_1}, \cdots, \mathbf{v_N}\}$ where a vertex position $\mathbf{v_i}$ specifies the coordinates $\{v_{ix}, v_{iy}, v_{iz}\}$ in $\Re^3$. The mesh topology, i.e. the connectivity between vertices, specifies the j vertices $\{\mathbf{v}_k^1, \cdots, \mathbf{v}_k^j\}$ in the k-th polygon, as described by IndexedFaceSet in VRML format [15]. In the following, the indices of vertices and polygons are denoted by I and F, respectively.

We adopt the following mesh traversal strategy to apply the prediction-error expansion algorithm. At the beginning, the polygon first indexed by F is examined. Among the vertices within it, the one first indexed by I is traversed. Then those vertices directly connected to the traversed one, i.e. the neighboring vertices, are searched out and the one first indexed by I is subsequently traversed. In the rest of traversal process, we always update the neighbors of the traversed vertices and further traverse the one first indexed by I among them. The updating and traversing operations are repeated to the end of the traversal process.

An illustration of the mesh traversal order is given in Fig. 2, where the labeled numbers are vertex indices. Suppose that the colored triangle containing vertices 1, 4, and 5 is first indexed by F. Then vertex 1 is first traversed and its neighbors are searched out, i.e. those with the indices of 4, 5, 6, and

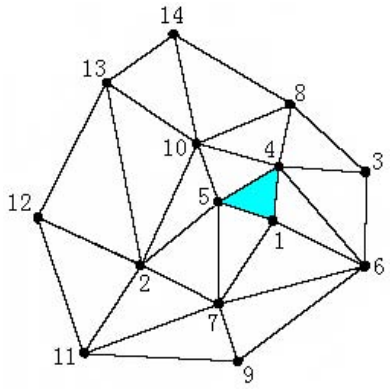


Fig. 2. An illustration of the mesh traversal order, where the labeled numbers are vertex indices.

7. Among them, vertex 4 is secondly traversed and then the neighbors of the traversed vertices include vertices 3, 5, 6, 7, 8, 10. So vertex 3 is subsequently traversed. After that, the neighbors of the traversed vertices are those 5, 6, 7, 8, 10. The process goes on until all the vertices are traversed in the order of 1, 4, 3, 5, 2, 6, 7, 8, 9, 10, 11, 12, 13, 14.

For the i-th traversed vertex as denoted by $\mathbf{v_i}$, suppose that there are N_i vertices directly connected to it and having been previously traversed as denoted by $(\mathbf{n}_i^j)_{j=1}^{N_i}$. Obviously, there is no traversed neighbor for the first traversed vertex. As for the case in Fig. 2, the traversed neighbor of vertex 4 is vertex 1 and the traversed neighbor of vertex 3 is vertex 4. For vertex 5, the traversed neighbors are vertices 1 and 4. We use the traversed neighbors of a vertex to predict its position $\mathbf{v_i}$ by calculating the centroid via

$$\mathbf{c_i} = \frac{1}{N_i}\sum_{j=1}^{N_i}\mathbf{n}_i^j. \tag{3}$$

Then the difference $\mathbf{d_i}$ between $\mathbf{v_i}$ and $\mathbf{c_i}$, i.e. the prediction error, is obtained by

$$\mathbf{d_i} = \mathbf{v_i} - \mathbf{c_i} = \{v_{ix} - c_{ix}, v_{iy} - c_{iy}, v_{iz} - c_{iz}\}, \tag{4}$$

where $\{v_{ix}, v_{iy}, v_{iz}\}$ and $\{c_{ix}, c_{iy}, c_{iz}\}$ are the coordinates of $\mathbf{v_i}$ and $\mathbf{c_i}$ in $\Re^3$, respectively. Three bit values $\mathbf{b_i} = \{b_{i1}, b_{i2}, b_{i3}\}$ can be embedded by expanding the error vector $\mathbf{d_i}$. Without loss of generality, we suppose that the coordinates in the original mesh are at the precision level of 10^{-n}. Then the error vector $\mathbf{d_i}$ and the bit values $\mathbf{b_i}$ are used to adjust the original vertex position $\mathbf{v_i}$ by

$$\tilde{\mathbf{v}}_\mathbf{i} = \mathbf{v_i} + T_m(\mathbf{d_i}) + \mathbf{b_i} \times 10^{-m}, \tag{5}$$

where $m \leq n$ and $T_m(\cdot)$ is the truncation function that truncates the part under the precision level of 10^{-m}. After the truncation, the precision of the error vector $\mathbf{d_i}$ is no higher than 10^{-n} for $m \leq n$. On the other side, m should not be lower than n to prevent the adding of 10^{-m} from aggravating the distortion. At each time, the position of the newly traversed vertex are adjusted by (5) to embed $\mathbf{b_i}$ in the expanded error vector. All the prediction errors can be expanded because they are no more precise than the coordinates in the watermarked meshes after truncation. Since a vertex consists of three coordinates, the amount of embedded bits are three times of the adjusted vertices. The position of the first traversed vertex $\mathbf{v}_1$ remains the same because it has no traversed neighbor. Given N vertices in the original mesh, there will be $3(N-1)$ bit values embedded in total. The watermarked mesh is generated after the position of the last vertex is finally adjusted.

C. The Recovery Process

The mesh traversal in the recovery process is performed in the same order as in the embedding process. Since the first traversed vertex $\mathbf{v}_1$ has no traversed neighbor, its position has not been modified in the embedding process. So it is just the prediction of the next traversed vertex, i.e. $\mathbf{c}_2$. With the vertex position $\tilde{\mathbf{v}}_\mathbf{2}$ in the watermarked mesh, we can obtain its original position $\mathbf{v}_2$ and the embedded data $\mathbf{b}_2$ by the inverse transform of (5), i.e.

$$\begin{cases} \mathbf{b_i} = T_m(\tilde{\mathbf{v}}_\mathbf{i} - \mathbf{c_i} - 2 \times T_m(\frac{\tilde{\mathbf{v}}_\mathbf{i} - \mathbf{c_i}}{2})) \times 10^m \\ \mathbf{v_i} = \tilde{\mathbf{v}}_\mathbf{i} - T_m(\frac{\tilde{\mathbf{v}}_\mathbf{i} - \mathbf{c_i}}{2}) - \mathbf{b_i} \times 10^{-m} \end{cases}. \tag{6}$$

The restored position $\mathbf{v_i}$ is used to predict the subsequently traversed vertices so that the prediction $\mathbf{c_i}$ can be calculated with (3) as in the embedding process. Therefore, the embedded data $\{\mathbf{b}_2, \cdots, \mathbf{b}_N\}$ can be extracted from the watermarked mesh and the vertex positions $\{\mathbf{v}_2, \cdots, \mathbf{v}_N\}$ in the original mesh can be restored with (6).

D. Adaptive Embedding with a Threshold

To reduce the distortion caused by adjusting the vertex positions, we can avoid to expand the prediction errors with too large magnitude by adaptively choosing a threshold in the following way:

Assuming that there are $L + 32$ prediction errors that have been calculated. We reserve the first 32 ones to embed a 32-bit floating point number that represents the threshold value. The next L prediction errors are sorted by their magnitude. Given a watermark consisting of P bits ($P < L$), a threshold μ can be adaptively chosen so that a prediction error with magnitude larger than μ will not be expanded. After expansion, the magnitudes originally larger than $\frac{\mu}{2}$ will exceed μ. So the flag bits should be saved in the watermarked mesh to indicate whether the expansion happens or not. Suppose there are L_1 prediction errors with magnitude less than $\frac{\mu}{2}$, L_2 ones between $\frac{\mu}{2}$ and μ, and L_3 ones larger than μ, respectively. Then a string consisting of $L_2 + L_3$ bits will be used to represent the location information, where 1 represents the expansion and 0 means no expansion. Besides the first 32 bits representing the threshold value, there are $L_1 + L_2$ bits embedded. It is required that $L_1 + L_2 \geq P + L_2 + L_3$, i.e.

$$L_1 - P \geq L_3. \tag{7}$$

For a watermark consisting of P bits (including a string of flag bits, an end of code (EOC) marker to separate them from the data to be embedded), the value of μ is adaptively chosen to satisfy (7) and represented by a 32-bit floating point number,

which is embedded in the first 32 prediction errors. In the recovery process, the threshold value is firstly extracted from the watermarked mesh, followed by the flag bits. Guided by the extracted information, the embedded data can be correctly extracted and the original mesh can be recovered.

E. Applying the Algorithm in [9] to Meshes

To apply the algorithm in [9] to 3D meshes, the untraversed neighbors of a vertex should be used to predict its position. The prediction error, i.e. the difference between the predicted and real positions of each vertex, can be expanded by modifying its coordinates in the same way as in (5) for data embedding. Since the last traversed vertex has no untraversed neighbor, its coordinates will not be modified. If we adopt the *inverse* mesh traversal order as in Section II. B to apply the algorithm in [9], the prediction for each vertex will be exactly the same as in Section II. B. The reason is very simple: the untraversed neighbors of a vertex will become its traversed neighbors if the traversal order is reversed. Therefore, the distortion caused by data embedding and the number of embedded bits are the same for the two prediction-error expansion algorithms.

If the algorithm in [9] is applied, the mesh traversal should be inversely performed in the recovery process to make the same prediction as in the embedding process. For the sake of clarity, we still use the traversal order in the embedding process to denote the vertices. For the last traversed vertex, it has no untraversed neighbor and its position has not been modified in the embedding process. So it is just the prediction of the second last traversed vertex, whose position will be first restored in the recovery process. After a vertex position has been restored as shown in (6), the restored position are used to predict its previously traversed neighbors and the whole mesh is recovered when the position of the first traversed vertex is restored. Meanwhile, the embedded bits can be extracted from the expanded prediction errors using (6), in the inverse of embedding order. In terms of complexity, our prediction-error expansion algorithm in Section II. A is more suitable for 3D meshes for its relative simplicity.

III. Performance Analysis

We implemented the reversible watermarking algorithm on the mesh models listed in Table I. The parameter m used in (5) and (6) was set to 2, which was used in both embedding and recovery processes. By encrypting the hash value of a mesh model, a signature of 320 bits was generated with the DSA algorithm [16]. The proposed algorithm is applicable to 3D polygonal meshes with arbitrary connectivity. Given N vertices in the cover mesh, the maximum capacity is $3(N-1)$ bits, tending to 3 bits/vertex. If a mesh model consists of J separate parts such as those in Table I, the maximum capacity will be $3(N-J)$ bits for the first traversed vertex within each separate part cannot be used for embedding.

In the case of adaptive embedding with a threshold, a 32-bit floating point number was used to represented the threshold value μ and firstly embedded in the watermarked mesh. It should be noted that the first 32 prediction errors are expanded without regard to the threshold. Then a string of flag bits was embedded to distinguish the prediction errors with magnitude between $\frac{\mu}{2}$ and μ from those larger than μ. The embedded watermark also included a two-byte EOC marker to separate the flag bits from the signature. It should be noted that the "payload" in Table I and Fig. 3 refers to the bits that can be embedded besides the 32-bit threshold value, the flag bits, and the EOC marker.

TABLE I
THE MESH MODELS USED IN THE EXPERIMENTS

3D mesh models	Number of parts	Number of vertices	Maximum embedding		Adaptive embedding with a threshold	
			Capacity (bits)	3D SNR (dB)	Payload	3D SNR
fish	1	742	2223	18.75	380	31.34
teapot	5	1631	4878	19.86	463	29.25
dog	48	7616	22704	28.65	4254	40.37
wolf	90	8176	24258	31.35	5723	42.10
horse	31	10316	30855	31.72	13972	40.11
lion	313	20315	60006	29.08	41398	33.09

The geometrical distortion of a mesh model was measured by using the 3D signal-to-noise ratio (SNR) defined as follows: Given N vertices in a mesh model as represented by $\mathbf{V} = \{\mathbf{v_1}, \ldots, \mathbf{v_N}\}$, a vertex position $\mathbf{v_i}$ specifies the coordinates $\{v_{ix}, v_{iy}, v_{iz}\}$ in R^3 for $i \in \{1, 2, \ldots, N\}$. After adding some noise to the mesh content, the vertex positions are changed to $\mathbf{G} = \{\mathbf{g_1}, \ldots, \mathbf{g_N}\}$. The 3D SNR of the mesh with the noise is defined by

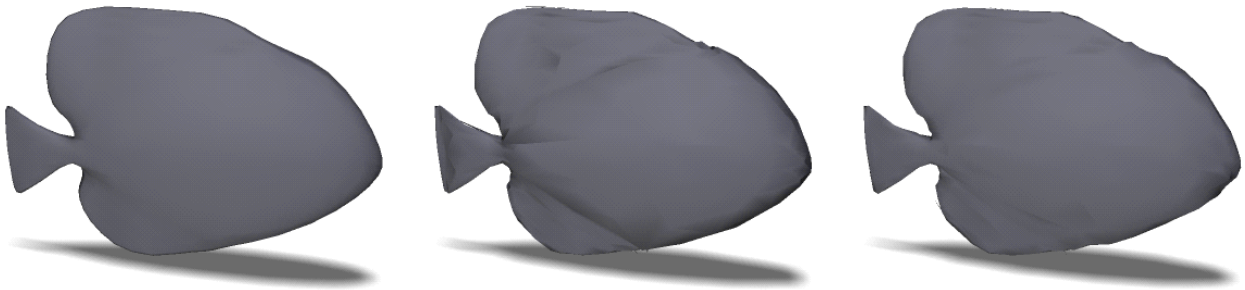
$$SNR = 10 \log_{10} \frac{MS(\mathbf{V} - \overline{\mathbf{v}})}{MS(\mathbf{G} - \mathbf{V})}, \tag{8}$$

where $MS(\cdot)$ is the mean square function while $\overline{\mathbf{v}} = \{\overline{v}_x, \overline{v}_y, \overline{v}_z\}$ is the mean of $\mathbf{V}$. In the definition, $\mathbf{V} - \overline{\mathbf{v}}$ is used to represent the signal, whereas $\mathbf{G} - \mathbf{V} = \mathbf{G} - \overline{\mathbf{v}} - (\mathbf{V} - \overline{\mathbf{v}})$ is the noise. In particular,

$$\begin{cases} MS(\mathbf{V} - \overline{\mathbf{v}}) = \frac{\sum_{i=1}^{N} (v_{ix} - \overline{v}_x)^2 + (v_{iy} - \overline{v}_y)^2 + (v_{iz} - \overline{v}_z)^2}{N} \\ MS(\mathbf{G} - \mathbf{V}) = \frac{\sum_{i=1}^{N} (g_{ix} - v_{ix})^2 + (g_{iy} - v_{iy})^2 + (g_{iz} - v_{iz})^2}{N} \end{cases}. \tag{9}$$

The 3D SNR of the watermarked models were calculated and listed in Table I. It can be seen that the distortion has been reduced without expanding the prediction errors with magnitude larger than μ. The pictures rendered from the original and watermarked mesh models of "fish" and "horse", and "lion" in the same illumination environment are shown in Fig. 3. We can see that the geometrical distortion has been reduced by adaptive embedding with a threshold.

In the experiments, the embedded watermarks could be correctly extracted from the watermarked models and the original meshes were exactly recovered. One application of our algorithm is for authentication of 3D mesh models by embedding a signature generated from the original mesh model. In the recovery process, the signature is blindly extracted while the original mesh model is recovered. a new hash value can be produced from the recovered model and compared with the one decrypted from the extracted signature. Compared



(a) The original mesh model of "fish"

(b) The watermarked mesh model with 2223 bits embedded

(c) The watermarked model after adaptively embedding 380 bits of payload

(d) The original mesh model of "horse"

(e) The watermarked mesh model with 30855 bits embedded

(f) The watermarked model after adaptively embedding 13972 bits of payload

(g) The original mesh model of "lion"

(h) The watermarked mesh model with 60006 bits embedded

(i) The watermarked model after adaptively embedding 41398 bits of payload

Fig. 3. The original mesh models of "teapot", "horse" and "lion" listed in Table I, the watermarked mesh models generated after the maximum embedding (without a threshold) and adaptive embedding (with a threshold), respectively.

with the fragile watermarking algorithms with a symmetric key (e.g. [17]-[20]), the reversible algorithm is not restricted to symmetric authentication. For public key authentication, a signature can be generated with a private key and decrypted with the corresponding public key. To achieve the property of tamper localization, the proposed algorithm can be applied to a mesh in a patch-wise manner. For instance, each patch should contain at least $\frac{320}{3} + 1 = 108$ vertices to embed a DSA signature with 320 bits.

IV. Conclusion

In this paper, we have applied the prediction-error expansion technique to 3D meshes. By using the traversed neighbors of a vertex for prediction, a reversible watermarking algorithm has been generated with a mesh traversal order. The prediction errors are expanded by modifying the vertex coordinates without changing the mesh topology. The distortion caused by the reversible data embedding can be reduced by choosing an adaptive threshold for the magnitudes of prediction errors. The embedded data can be blindly extracted from the watermarked mesh while the original mesh can be exactly recovered without any extra information. In addition to symmetric or public key authentication, the reversible algorithm can also be used for content annotation and archiving of 3D meshes.

References

[1] C. W. Honsinger, P. Jones, M. Rabbani, and J. C. Stoffel, "Lossless Recovery of an Original Image Containing Embedded Data," *U.S. Patent*, No. 6278791, 2001.

[2] B. Macq and F. Deweyand, "Trusted Headers for Medical Images," *Proceeding of DFG VIII-DII Watermarking Workshop*, Erlangen, Germany, Oct. 1999.

[3] J. Fridrich, M. Goljan, and R. Du, "Invertible Authentication," *Proceeding of SPIE: Security and Watermarking of Multimedia Contents*, Vol. 4314, pp. 197-208, San Jose, CA, USA, Jan. 2001.

[4] M. Goljan, J. Fridrich, and R. Du, "Distortion-Free Data Embedding for Images," *Proceedings of the 4th Information Hiding Workshop*, Vol. 2137, pp. 27-41, New York, USA, Apr. 2001.

[5] M. Celik and G. Sharma, M. Tekalp and E. Saber, "Reversible Data Hiding," *Proceedings of the IEEE International Conference on Image Processing*, Vol. 2, pp. 157-160, Rochester, NY, USA, Sep. 2002.

[6] G. Xuan, J. Zhu, J. Chen, Y. Q. Shi, Z. Ni and W. Su, "Distortionless Data Hiding Based on Integer Wavelet Transform," *IEE Electronics Letters*, Vol. 38, No. 25, pp. 1646-1648, Dec. 2002.

[7] J. Tian, "Reversible Data Embedding Using a Difference Expansion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 8, pp. 890-896, Aug. 2003.

[8] C. De Vleeschouwer, J. E. Delaigle and B. Macq, "Circular Interpretation of Bijective Transformations in Lossless Watermarking for Media Asset Management," *IEEE Transactions on Multimedia*, Vol. 5, No. 1, pp. 97-105, Mar. 2003.

[9] D. M. Thodi and J. J. Rodriguez, "Prediction-error Based Reversible Watermarking," *Proceedings of the IEEE International Conference on Image Processing*, Vol. 3, pp. 1549-1552, Oct. 2004.

[10] Z. Ni, Y. Q. Shi, N. Ansari and W. Su, "Reversible Data Hiding," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 3, pp. 354-362, Mar. 2006.

[11] J. Dittmann and O. Benedens, "Invertible Authentication for 3D Meshes," *Proceedings of SPIE, Security and Watermarking of Multimedia Contents V*, Vol. 5020, pp. 653-664, Jun. 2003.

[12] H. T. Wu and Y. M. Cheung, "A Reversible Data Hiding Approach to Mesh Authentication," *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, pp. 774-777, Compiègne University of Technology, France, Sep. 2005.

[13] Z. M. Lu and Z. Li, "High Capacity Reversible Data Hiding for 3D Meshes in the PVQ domain," *Proceedings of the 6th International Workshop on Digital Watermarking*, pp. 593-596, Guangzhou, China, Dec. 2007.

[14] X. Mao, M. Shiba and A. Imamiya, "Watermarking 3D Geometric Models Through Triangle Subdivision," *Proceedings of SPIE, Security and Watermarking of Multimedia Contents III*, Vol. 4314, pp. 253-260, Jan. 2001.

[15] ISO/IEC DIS 14772-1, "The Virtual Reality Modeling Language." http://www.web3d.org/x3d/specifications/vrml/

[16] A. Menezes, P. van Oorchot, and S. Vanstone, "Handbook of Applied Cryptography," *Boca Raton, FL: CRC*, 1997.

[17] B. L. Yeo and M. M. Yeung, "Watermarking 3-D Objects for Verification," *IEEE Computer Graphics and Application*, Vol. 19, No. 1, pp. 36-45, Jan./Feb. 1999.

[18] F. Cayre, O. Devillers, F. Schmitt, and H. Maitre, "Watermarking 3D Triangle Meshes for Authentication and Integrity," *INRIA Research Report RR-5223*, Jun. 2004.

[19] H. Y. S. Lin, H. Y. M. Liao, C. S. Lu and J. C. Lin, "Fragile Watermarking for Authenticating 3-D Polygonal Meshes," *IEEE Transactions on Multimedia*, Vol. 7, No. 6, pp. 997-1006, Dec. 2005.

[20] Y. M. Cheung and H. T. Wu, "A Sequential Quantization Strategy for Data Embedding and Integrity Verification," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 17, No. 8, pp. 1007-1016, Aug. 2007.