Multimedia and Security Workshop at ACM Multimedia '98, Bristol, U.K., September 12-13, 1998.

# Watermarking Multiple Object Types in Three-Dimensional Models

Ryutarou Ohbuchi
IBM Tokyo Research Laboratory
1623-14 Shimo-tsuruma, Yamato-shi,
Kanagawa, 242-8502, Japan
ohbuchi@acm.org

Hiroshi Masuda
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku
Tokyo, 113-8656, Japan
masuda@nakl.t.u-tokyo.ac.jp

Masaki Aono
IBM Tokyo Research Laboratory
1623-14 Shimo-tsuruma, Yamato-shi,
Kanagawa, 242-8502, Japan
aono@acm.org

## 1. ABSTRACT

**Three-dimensional (3D) graphical model is about to become a full-fledged multimedia data type, prompted by increasing popularity of Virtual Reality Modeling Language (VRML) [7] and imminent standardization of MPEG4 [8].**

**Following an introduction on data embedding, this paper presents a discussion on potential targets of data embedding that exist in both VRML and MPEG4 formats. We then present several algorithms that embed data in shape (i.e., geometry and topology of the shapes) and shape attributes associated with shape (e.g., per-vertex texture coordinates).**

### 1.1 Keywords

Three-dimensional computer graphics, geometrical modeling, information security, digital watermark.

## 2. INTRODUCTION

The advantages of digital media, such as the Internet and CD-ROMs lies in the fact that the duplication, distribution, and modification of contents are much easier than the older media, such as printed media. For example, duplication of a digital content can be performed without any loss of its quality. These advantages, however, are double-edged swords. Digital media made unauthorized duplication, distribution, and modification of their valuable contents easier.

*Data embedding*, or *(digital) watermarking* put structures called *watermarks* into digital contents (e.g., images) in such a way that the structures do not interfere with intended use (e.g., viewing) of the contents. The watermarks carry information that can be used to manage the contents, for example, to add annotations, to detect tampering, or to authenticate rightful purchasers. While data can be embedded in an analog media, digital media provided an opportunity for a robust data embedding with significant data capacity.

In the past, a multimedia content typically meant a content that includes text, image, video, and audio data types. As a result, data embedding techniques for these "traditional" digital content data types has been studied by many [18, 19, 23, 1, 2, 11, 17, 3, 5, 10, 21, 22]. As 3D model gains status as an important member of multimedia data types, prompted by increasing popularity *of Virtual Reality Modeling Language* [6] and imminent standardization of *MPEG-4* [7], we added 3D polygonal model of geometry to the list of data embedding targets [12, 13, 14, 15].

In this paper, we will first introduce data embedding in general, followed by a discussion on embedding targets that exists in 3D models that follows VRML and MPEG4. We will then present three embedding algorithms, each of which is based on vertex coordinate modification, vertex topology modification, and texture coordinate modification, respectively.

### 2.1 Data Embedding Classifications

In this paper, following recommendation in [16], the act of adding watermark is called (data) *embedding or watermarking*, and retrieving the information encoded in the watermark for perusal is called *extraction*. The object in which the information is embedded is called *cover-<datatype>*, the object with watermark is called *stego-<datatype>*, and the information embedded is called

*embedded-<datatype>*. The suffix "<datatype>" varies with data types, such as image, text, or 3D model. For example, an embedded-text is embedded in a cover-polygonal mesh to produce a stego-polygonal mesh with embedded-text.

A watermarks can be classified by its (1) *visibility* (or, more generally, *perceptibility*) and (2) *robustness*, as suggested by Mintzer, et al. [10]. A *visible watermark* is made intentionally visible to serve their purposes, for example, to deter a third party from unauthorized sales of contents. On the other hand, an *invisible watermark* is imperceptible without processing by mechanical means. A *robust watermark* should resists both intentional and unintentional modifications of the watermarked content. A *fragile watermark*, on the other hand, must be altered by intentional (and some unintentional) modifications so that it could detect tampering of or damage to the content. Here, *unintentional modifications* are the kind a content should expect during a course of its intended use, while *intentional modifications* are the kind that are applied with an intention of destroying or altering the watermark**.**

A watermark can be classified further by its use of cover data for extraction. If an extraction algorithm requires original cover data as well as the (possibly corrupted) stego-data, the scheme is called *private watermarking*. Otherwise, the scheme is called *public watermarking*. An embedding scheme by Cox et al [3] is an example of private watermarking.

A watermarking scheme may employ a random sequence generator to make an embedded message secure from being read by a third party. For example, in an image watermarking, positions of pixels to be modified for watermarks can be scrambled by a pseudo-random sequence generated from a stego-key (or stego-keys) by using a public-key cryptographic method [9]. The scrambling can also be used to erase (reduce) statistical signature in order to make watermarking less detectable. Both public-key cryptography and shared-(private-)key cryptographic method can be used for this purpose.

Data embedding has many potential applications. Obviously, requirements for data embedding scheme vary depending on its intended application(s). Some of the potential applications are listed below.

- **Theft deterrence:** A robust, visible yet unobtrusive watermark in an image could deter unauthorized sales of the image by lowering commercial value of the image.
- **Copyright notification:** A copyright could be embedded as a robust invisible watermark into an image. Such notification could direct users of the model to the web site of the model's copyright owner.
- **Tamper detection:** Images taken by a digital still camera can be marked in the camera with a fragile invisible watermark so that modification made to the image afterward can be detected.
- **Content integrity check:** Since MPEG4 contents are editable, content creators might fear that a part of her/his creation is extracted and played without context, or a part of the content might is substituted. Watermarks in polygonal models and other 3D model contents could be used to detect such tampering.
- **Fingerprinting:** If an image is "fingerprinted" with the identities (e.g., digital signatures) of its purchaser and seller by using a robust watermarking technique, circulation of unauthorized copies of the image could be traced to the purchaser.
- **Play or duplication control:** Robust invisible watermark could control hardware devices to stop delivery of pornographic or violent digital-video contents (a la v-chip for broadcast TV in USA), or to prevent unauthorized duplication.

## 3. EMBEDDING TARGET OBJECTS IN 3D MODELS

3D models in VRML [6] and MEPG4 [7] formats contain many types of objects. Among them, we consider objects in the following list to be important targets for data embedding. These objects are important since they have relatively large quantity of redundancy that can be exploited for data embedding.

**1. Shape**

- Polygonal Mesh Topology and Geometry
- Regular Mesh Geometry
- Elevation Grid Height Field Values

**2. Shape attributes**

- Vertex color (opacity), vertex texture coordinate, vertex normal vector, etc.
- Line color, etc.
- Face color (opacity), face normal vector, index of refraction, etc.
- Volume color (opacity), etc.

**3. Animation parameters**

- Interpolators
  - Point/vertex coordinate and orientation..
  - Colors and normal vector.
  - Camera position and orientation.
- Face and Body Animation Parameters
  - Parameterized position of eyes, tongue, etc.
  - Angle of joints, etc.
- Animated Mesh
  - Vertex coordinates displacements.

**4. Others**

- 2D still texture image, movie texture.
- Sampled sound.

- Text string, text position and orientation, text color, etc.

Text-to-speech phoneme strings and synthetic sound symbol sequences (e.g., a MIDI command sequence) contain little redundancy to be used as good embedding targets.

### *3.1.1 Shapes*

*Shapes*, or *geometrical components*, of 3D objects are arguably the most important class of target, for without shape, 3D model means little. While point set and poly-lines are viable candidate for data embedding targets, *polygonal mesh* is probably the most important target for data embedding in 3D models.

Two components, *vertex coordinates* and *vertex topology*, define shape of a 3D polygonal mesh. Vertex coordinate combined with vertex topology defines more complex *geometrical primitives*, that are, lines, polygons, and polyhedrons. These geometrical primitives have their own quantities such as length of a line segment and volume of a polyhedron that are called *geometrical quantities* in this paper. The geometrical primitives have topology of their own, which are, for example, connectivity of triangles and tetrahedrons.

Data can be embedded in a 3D shape by modifying either geometry or topology of its geometrical primitives. A unit of such modification is called *embedding primitive*. It is also important to arrange these embedding primitives in an order so that the arranged set of embedding primitive as a whole carry a significant amount of information. Arrangement can be created either by topology or by quantity of geometrical primitives.

Details of fundamental methods to embed data into shapes can be found in papers [14] and [15]. Section 4 of this paper presents two data embedding algorithms, one that targets geometry and the other targets topology. (These algorithms have previously appeared in [14] and [15], but included in here for completeness. )

### *3.1.2 Shape Attributes*

Shape-*attributes*, such as vertex color, per-vertex texture coordinates, per-face color and per-volume refractive index, are essentially sets of numerical values that can be modified to embed data. While less important than a shape itself, a shape attribute still is an important class of target for embedding.

The approach to embedding using shape attributes is similar to those used for algorithms that employ coordinate embedding primitives; Values of the attributes are modified and the modifications are ordered to embed a significant amount of information.

Details of a data embedding algorithm that targets texture coordinate will be presented in Section 4. (This algorithm has previously appeared in [15].)

### *3.1.3 Animation Parameters*

Animation parameters have potentials to become very important targets for data embedding. For example, if polygon-based counterparts of music videos are made, moves of a popular musician captured to animate his/her figure could carry a very high value.

VRML provides various *interpolators* for animation. An interpolator is a sequence of multiple sets of values that are linearly interpolated to produce continuously varying values. These varying values can be used to translate, rotate, or deform objects.

Figure 1 shows an example of VRML interpolator data generated by a 3D modeling and animation software. It is a plot of trajectory of the torso of a skateborder model as it performs a manuever called "540". In the torso alone, this animation sequence contained 53 coodinate points, each of which is a 3D coordinate. Combined with the other parts, such as head, upper-arm, lower-arm, camera, etc., there are siginificant amount of data that can be exploited for embedding in the animated 3D model.

The MPEG4 proposal [7] contains other types of data objects for animation. It contains animated (deforming) regular mesh whose vertices move over time given a continuously transmitted list of incremental displacements. The displacements may be coded in two ways, either by using simple difference or by using discrete cosine transformation of a short sequence of displacement values.

The MPEG4 proposal also contains human face and body models that can be animated by transmitted animation parameters. For example, a facial model is controlled by a set of about sixty integer values, each of which specify location of eyebrows, eyes, a tounge, ears, etc. Most of

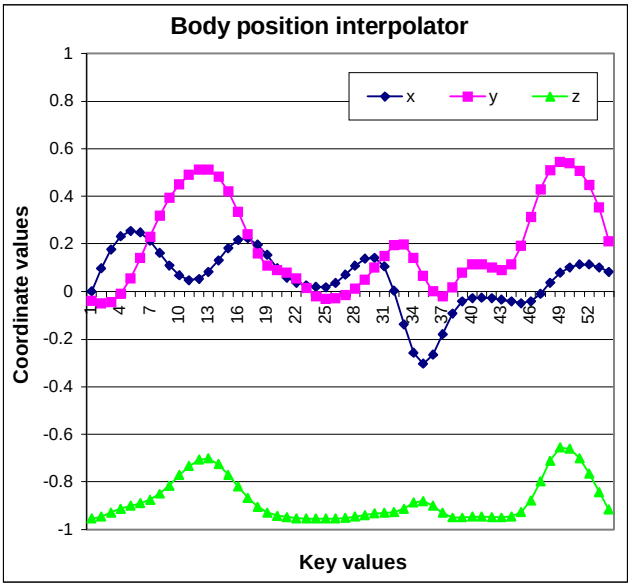



**Figure 1. Plot of key values of a VRML coordinate interpolator in an animation sequence.**

these parameters use small quantization levels (from 2 to 5), except for a few. The body model is controlled by over 170 parameters, each of which has quantization levels of 256. While face animation parameters with its small quantization levels may lack redundancy for data embedding, body animation parameters will have enough redundancy to be exploited for data embedding.

It must be noted, however, that the MPEG4 proposal contains extensive list of both lossless and lossy compression algorithms for shapes, shape attributes, and for animation parameters. Data compression algorithms in general try to find and remove redunduncies that are necessary for data embedding algorithms. Embedding algorithms must take these data compression algorithms into account.

### *3.1.4 Others*

Sampled sound, 2D still texture and 2D movie texture are obvious targets of embedding by using data embedding algorithms developed previously for respective data types. However, care must be taken in using these objects for embedding since these objects in a 3D model can be removed effortlessly.

## 4. EMBEDDING ALGORITHMS FOR 3D POLYGONAL MESHES

In this section, we will present algorithms that target shape and a shape attribute of polygonal meshes for data embedding. All the algorithms in this section are implemented by using a kernel for a non-manifold modeler [9]. The system employs *radial edge* structure [20] to represent the topological relationship among vertices, edges, faces, and regions.

### 4.1 An Algorithm Based on Geometrical Quantity Modification

A pair of dimensionless quantities, for example, $\{e_{14}/e_{24}, h_4/e_{12}\}$ in Figure 2, defines a set of similar triangles. The algorithm described in this section, Triangle Similarity Quadruple (TSQ) algorithm, uses such dimensionless quantity pair as the geometrical embedding primitive to watermark triangular meshes.

The TSQ algorithm can be classified as a public watermarking scheme. Watermarks produced by the TSQ algorithm withstand translation, rotation, and uniform-scaling transformations of the stego-polygonal-meshes. An embedded message is resistant to resection and local deformation if it is repeatedly embedded over a mesh. The watermarks are destroyed, among other disturbances, by a randomization of coordinates, by a more general class of geometrical transformation, or by a topological modification such as re-meshing.

In order to realize subscript ordering, the algorithm uses a quadruple of adjacent triangles in the configuration depicted in Figure 2 as a Macro-Embedding-Primitive (MEP). Each MEP stores a quadruple of symbols *{Marker, Subscript, Data1, Data2}*. In Figure 2, the triangle marked *M* stores a marker, *S* stores a subscript, and $D_1$ and $D_2$ stores data values. A marker is a pair of values that identifies MEPs. As mentioned above, this public watermarking scheme does not require cover-polygonal-mesh for extraction. However, the marker value pair is necessary for extraction. A watermarked mesh would contain multiple MEPs to embed a significant amount of data as shown in the example of Figure 3. While each MEP is formed by topology, a set of multiple MEPs is arranged by quantity of the subscript.

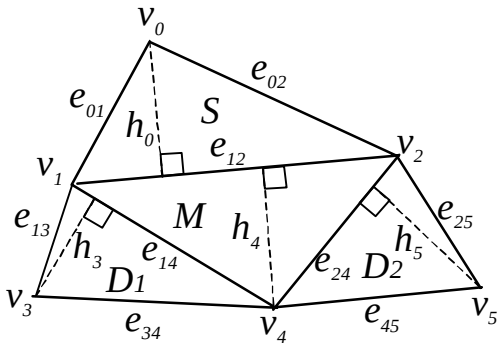



**Figure 2. A macro-embedding-primitive. In the figure, $v_i$ are vertices, $e_{ij}$ are lengths of the edges, and $h_i$ are heights of the triangles.**

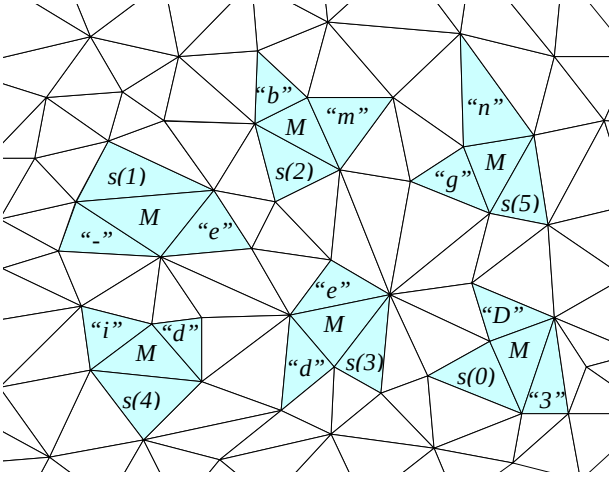



**Figure 3. In this example of TSQ watermarking, six macro embedding primitives on a mesh embed a string "3D-embedding". Subscripts (denoted *s(i)* for a subscript *i*) arranges the MEPs.**

The TSQ algorithm embeds a message according to the following steps. (For the detailed explanation and execution examples, please refer [14].)

(1) Traverse the input triangular mesh to find a set of four triangles to be used as a MEP. MEPs must not share edges or vertices to avoid interference.

(2) Embed the marker value by changing a

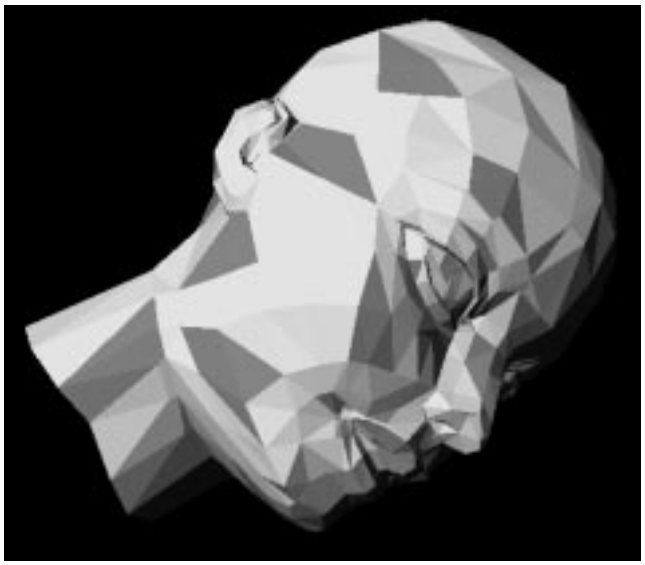


**Figure 4. Macro embedding primitives, each of which consists of four adjacent triangles, are shown in dark gray.**

dimensionless quantity pair in the center triangle of the MEP. In Figure 2, it is $\{e_{14}/e_{24}, h_4/e_{12}\}$. This modifies positions of vertices $v_1$, $v_2$, and $v_4$.

(3) Embed a subscript and two data symbols in a similar manner by displacing vertices $v_0$, $v_3$, and $v_5$. Subscript is embedded in the pair $\{e_{02}/e_{01}, h_0/e_{12}\}$, and two data symbols are embedded in the pairs $\{e_{13}/e_{34}, h_3/e_{14}\}$ and $\{e_{45}/e_{25}, h_5/e_{24}\}$.

(4) Repeat (1) to (3) above until all the data symbols of the message are embedded.

For each triangle, the algorithm first modifies the ratio $h_i/e_{ij}$ by changing $h_i$ only. Then the algorithm modifies the ratio $e_{ij}/e_{kl}$ while keeping the height $h_i$ constant. In order to embed the message repetitively, steps (1) to (4) are repeated many times.

Figure 4 shows triangles that formed MEPs in darker gray. Due to the mutual exclusion rule described in the step (1) above, MEPs do not share vertices.

Given a watermarked mesh and two numbers that identify marker triangles, extraction proceeds according to the following steps.

(1) Traverse a given triangular mesh and find a triangle with the marker, thereby locating a MEP.

(2) Extract a subscript and two data symbols from the triangles in the MEP.

(3) Repeat (1) to (2) above for all the marker triangles on a given triangular mesh.

(4) Sort the extracted symbols according to their subscripts.

The TSQ algorithm embedded 210 bytes of data, that is, 0.15 byte/triangle, in the model of Figure 4, which consisted of 1406 triangles. Experiments using seven polygonal mesh models showed that the TSQ algorithm was able to embed 0.15-0.18 byte/triangle.

## 4.2 An Algorithm Based on Topological Modification

The *Triangle Strip Peeling Symbol sequence (TSPS)* embedding algorithm that will be presented in this section is a public watermarking scheme based on a topological embedding primitive. It employs, as its embedding primitive, an adjacency of a pair of triangles in a triangle strip, each of which encodes a binary bit of information. One-dimensional arrangement of embedding primitives is induced by the adjacency of triangles on the triangle strip. To recognize the triangle strip with watermark, the strip is peeled off from the original mesh.

Since both embedding primitive and arrangement are topological, watermarks produced by the algorithm are immune to geometrical transformation. Repetitive embedding makes the watermarks resistant to resection. The watermarks can be destroyed by topological manipulations, for example, by polygon simplification algorithms. A disadvantage of this algorithm is its low space efficiency compared to many algorithms based on geometrical primitives.

Inputs to this embedding algorithm are an orientable triangular mesh and a message bit string. The TSPS embedding algorithm embeds data according to the following steps. (See Figure 5.)

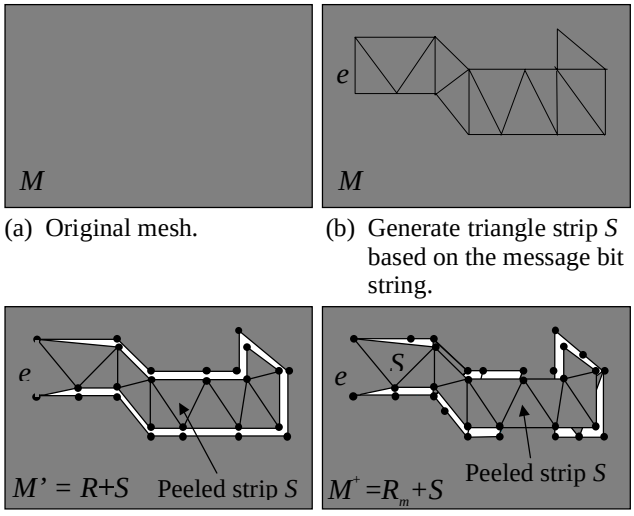
(1) Starting from an edge $e$ selected from the input mesh $M$, grow a triangle strip $S$ on $M$ by using the message bit-string to determine the direction of growth of the strip. Observe that a triangle at the end of (current) strip has two "free" edges, i.e., edges that are not adjacent to triangles of the current triangle strip. Since $M$ is orientable, these two edges can be ordered on the triangle by traversing the edges in a fixed order (either counterclockwise or clockwise). Depending on the data bit, choose one of the two free edges as the edge to be shared with the next triangle of the strip. (See Figure 6.)

(2) "Peel off" the triangle strip $S$ from $M$ by splitting all the edges and vertices on the boundary of $S$ except the initial edge $e$. The strip $S$ is connected to the rest of the mesh only by the edge $e$.

The edge $e$ serves as the initial condition for finding the triangle strip. Arrangement of embedding primitives is induced naturally by the connectivity of triangles on the triangle strip. Since the peeled strip caps the hole completely, proper colors and vertex normal vectors make the watermark invisible.

Figure 6 shows an example of a triangle strip. The strip drawn with solid lines, which start at edge *e*, embeds a bit string "10101101011" in a sequence of 12 triangles. Each bit of the bit string steers the direction of growth of the triangle strip. If the last bit of the string is "0" instead of "1", the last triangle will become the one that is drawn with broken lines.

Steering by message bit strings produces strips whose shape may not fit in a given mesh, depending on a given bit string. In the example of Figure 6, a message bit string with all "1" would keep steering the strip to the left. If the message string is sufficiently long, the strip will either hit the boundary of the mesh or circle back to itself. To avoid this problem, shapes, locations and orientations of the strips must be controlled carefully. We manipulate the shape of the triangle by using *steering symbols*. A steering symbol is a bit that does not carry information but simply steer direction of growth of a triangle strip. Steering symbols are interleaved with data symbols, that are, symbols that encode embedded data, in order to control shape of triangle strips. Obviously, steering symbol halves the embedding data capacity. Our current implementation determines initial locations, directions of growth, and shapes of triangle strip manually.




(a) Original mesh.

(b) Generate triangle strip *S* based on the message bit string.

(c) Cut out the strip *S* from the mesh *M*, except at the edge *e*.

(d) Scramble mesh topology of the stenciled mesh (*M*-*S*) by adding vertices, edges, and polygons, if necessary.

**Figure 5. Triangle strip *S* encoding a message bit string is peeled off from the cover-polygonal mesh *M*. (The cracks around *S* in the figure is for illustration purpose only.)**

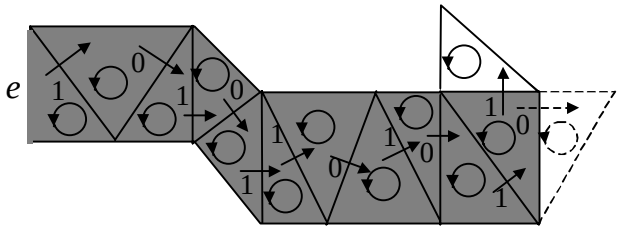



**Figure 6. Connectivity of 12 triangles (drawn with solid lines) in a triangle strip encodes the bit string "10101101011" (11 bits). If the bit string is "10101101010" (change in the last bit), the last triangle will be the one drawn with broken lines.**

Extraction of a message is carried out according to the following steps.

(1) Traverse the watermarked mesh and find an edge with topological features that starts a triangle strip of known length that is attached to the stencil mesh by an edge.

(2) Starting from the initial edge, traverse the triangle strip to the open end as embedded bits are extracted.

Figure 7 shows a simple example of TSPS embedding, in which a triangle strip of length 27 is peeled off from a mesh that consisted of 214 triangles. The triangle strip encodes 13 data bits and 13 steering bits. Selection of steering bits in this case was done manually. As another example, a model of triceratops (499 triangles) in Figure 8 is marked with a triangle strip of length 19 triangles, which encodes 9 bits. (The colors of the strips in these examples are intentionally changed to show their location.)

Watermarks produced by the TSPS embedding algorithm can be erased if a geometrical "mending" program, for example the one similar to [4], which would stitch the triangle strip back to the stencil mesh. Such mending can be prevented to some extent by modifying topology of stego-polygonal-mesh in order to confuse mending algorithms. For example, vertices, edges, and polygons can be added into the stencil mesh *R* so that finding correspondence of edges and vertices to be stitched

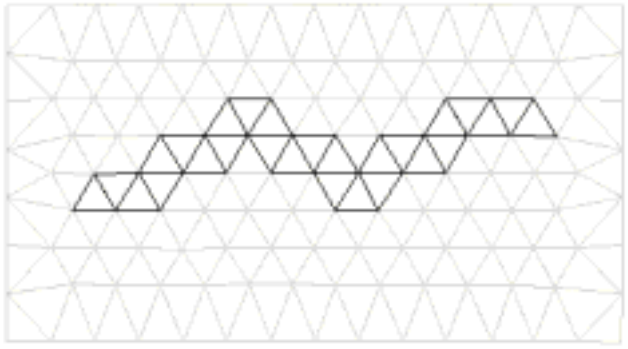


**Figure 7. A triangle strip consisting of 27 triangles was cut out from a flat triangular mesh (214 triangles). The triangle strip, displayed in darker gray, encodes 13 data bits interleaved with 13 steering bits.**

together is difficult (Figure 5(d)).

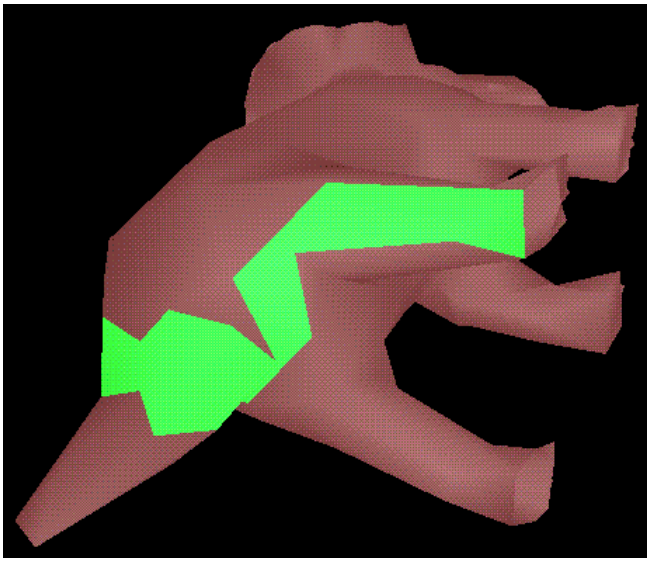


**Figure 8. A triangle strip, 19 triangles long and shown in a light gray, is generated and peeled off from a model of a triceratops (499 triangles). (A part of the strip is not visible from this viewpoint.)**

## 4.3 An Algorithm Based on Shape Attribute Modification

An algorithm explained below embeds data in texture coordinates of polygonal mesh. A similar algorithm can be used to embed data in other per-vertex attributes, such as vertex colors. Data embedding into per-face attributes of a polygonal mesh surface is also possible; Modify per-face attributes and then arrange these modified attributes.

A set of texture coordinates associated with vertices of a polygonal model is a good target for data embedding. This is because a set of roper texture coordinate is crucial to properly render texture mapped objects, and a set of texture coordinates is difficult to regenerate once it is lost.

The algorithm we experimented modulates amplitude of texture coordinates based on message bit string. Let $s_i$ be $i$th bit of a bit string $S$. The embedding algorithm modifies a coordinate value $x_i$ (e.g., either $u$ or $v$) of a texture coordinates by the following steps, given a modulation amplitude $A$.

$r=x_i-x_i/A;$
**if** $s_i$='0' **then** $b=A/4$ **else if** $s_i$='1' **then** $b=A*3/4$;
$x_i= r+b$;

This is just an example of modulation method. Many other alternatives, including multi-valued modulations, are possible. Whatever the modulation method, we can make two such modulations per 2D-texture coordinates. Thus, if we embed one bit per floating point number, we can embed $2N$ bits into $N$ 2D-texture coordinates.

In modifying the texture coordinate, the modulation amplitude $A$ must be chosen so that the watermark is robust enough without degrading quality of texture mapped objects. We conducted experiments to see how amplitude affect appearances of texture-mapped 3D polygonal mesh objects. Some of the results are shown in Figure 9 and Figure 10.

Texture images are a synthetic red-and-white stripe image (256 x 256 pixels) and a photograph of a human face with a tree leaves in background (300 x 300 pixels image area, 1024 x 1024 pixels overall). These images were texture-mapped onto a model of a sphere tessellated into 1800 triangles, which contained 961 vertices (and thus 961 texture coordinate). We can embed a maximum of 961 bytes in the sphere if we modify four bits per single-precision floating-point number. In this experiment, we embedded a 358 byte long text.

In the figures, $Ar$ is the modulation amplitude relative to the range of texture coordinate variation on the model. In these examples, the texture coordinates varied in the range [0,1] in both $u$ and $v$ coordinates so that the maximum variation range of texture was 1.0. In another word $Ar$=0.1 % means amplitude of 0.001.

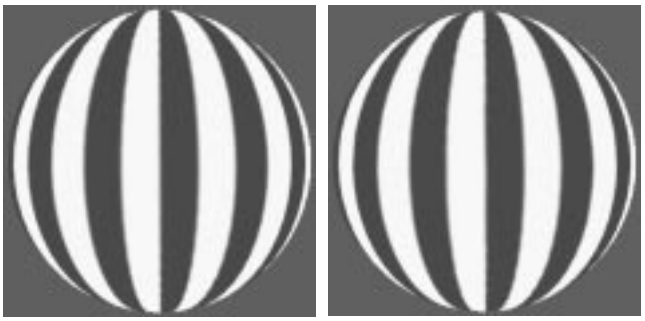
In Figure 9, in which the red-and-white stripe texture is used, distortion in the rendered image is perceptible in rendered images when $Ar$=0.5 % (Figure 9c) and $Ar$=1 % (Figure 9d). Complex, less geometrical, texture images reduced perceptibility of texture distortions. Distortions of the human face texture shown in Figure 10 were difficult to perceive. Even for the image with $Ar$=1 % (Figure 10c), a careful comparison with the original image (Figure 10a) was necessary to reveal distortions.

In our prototype implementation, we used the order of appearance of texture coordinates in the input file as the arrangement for embedding. This arrangement is destroyed easily by shuffling the positions of the texture coordinates in the file. If this is a problem, there are alternative methods to introduce ordering into a set of texture coordinates. Since each texture coordinates is associated with a vertex, ordering vertices implies ordering of texture coordinates. Several examples of methods to order vertices are described in [14]. It is also possible to arrange texture coordinate by using a non-geometrical quantity itself. In the example of texture coordinate, texture coordinate or quantity derived from it can be used to order vertices.

Note that watermark that modifies geometry and/or topology of a polygonal mesh do not interfere directly with non-geometrical attributes. It is possible to combine an attribute-modifying algorithm (e.g., the one described in this section that modifies texture coordinate) with an algorithm that modifies geometry or topology (e.g., the triangle strip peeling algorithm).

This experiment showed that, if modification amplitude is chosen appropriately, data embedding into texture

coordinates is possible without noticeable change in the models rendered appearance.



(a) *Ar=0* (No embedding) (b) *Ar*=0.1 %

(c) *Ar*=0.5 % (d) *Ar*=1 %

**Figure 9. A red-and-white stripe image is mapped onto a sphere model (1800 triangles). Texture coordinates are modulated with relative amplitudes *Ar*=0 % to Ar=1 %.**

(a) *Ar=0.* (No embedding) (b) *Ar*=0.1 %

(c) *Ar*=0.5 % (d) *Ar*=1 %

**Figure 10. A photograph of a human face is mapped onto a sphere model (1800 triangles). Texture coordinates are modulated with several relative amplitudes *Ar*=0.0% to 1%.**

## 5. SUMMARY AND FUTURE WORK

In this paper, we first presented introduction to data embedding technology. It is followed by a discussion on possible data embedding targets that exist in 3D models, that are, shape (both topological and geometrical components of shape), shape-attributes (e.g., texture coordinates and vertex color), and others, such as mesh animation parameters and face/body animation parameters. As examples, we presented three algorithms. Two of the algorithms embed data in shape, using both geometry and topology of 3D polygonal meshes. The other algorithm embeds data in a shape-attribute, that is, texture coordinates, of 3D polygonal mesh models.

In the future, we would like to experiment with algorithms that embed data in animation parameters that exists in MPEG4 and VRML formats. We need to evaluate effects of data compression algorithms used in these formats to compress shape, shape attributes, and animation parameters. We also would like to develop and test realistic scenarios employing data embedding algorithms for 3D models.

## 6. REFERENCES

[1] W. Bender, D. Gruhl, and N. Morimoto, Techniques for Data Embedding, *IBM Systems Journal*, Vol. 35, Nos. 3 & 4, 1996.

[2] G. Braudway, K. Magerlein, and F. Mintzer, Protecting Publicly-Available Images with a Visible Image Watermark, *IBM Research Report*, TC-20336 (89918), January 15, 1996.

[3] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, *IEEE Trans. on Image Processing*, Vol. 6, No. 12, pp1673-1678, 1997.

[4] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn, Cutting and Stitching: Efficient Conversion of a Non-Manifold Polygonal Surface to a Manifold, *IBM Research Report* RC-20935 (92693), July, 1997.

[5] F. Hartung and B. Girod, Copyright Protection in Video Delivery Networks by Watermarking of Pre-Compressed Video, *Lecture Notes in Computer Science*, Vol. 1242, pp.423-436, Springer, 1997.

[6] ISO/IEC 14772-1 Virtual Reality Model Language (VRML).

[7] ISO/IEC JTC1/SC29/WG11 *MPEG-4 Visual* and *MPEG 4 SNHC*.

[8] H. Masuda, Topological Operations for Non-Manifold Geometric Modeling and Their Applications, *Ph. D dissertation*, Department of Precision Machinery Engineering, University of Tokyo, 1996 (in Japanese).

[9] A. J. Menezes, P. C. van Oorshot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[10] F. Mintzer, G. W. Braudway, and M. M. Yeung, Effective and Ineffective Digital Watermarks, Proceedings of the *IEEE International Conference on Image Processing (ICIP) '97*, Vol. 3, pp. 9-12, 1997.

[11] J. J. K. O'Ruanaidh, W. J. Dowling and F. M. Boland, Watermarking Digital Images for Copyright Protection, *IEE Proc.-Vis. Image Signal Process.*, Vol. 143, No. 4, pp. 250-256, August 1996.

[12] R. Ohbuchi, H. Masuda, and M. Aono, Embedding Data in 3D Models, in Steinmetz, et al. eds, *Lecture Notes in Computer Science* No. 1309, pp.1-11 (Proceedings of the *IDMS '97*, Darmstadt, Germany, September) 1997.

[13] R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models, *Proceedings of the ACM Multimedia '97*, Seattle, Washington, USA, November 1997, pp. 261-272.

[14] R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications, pp. 551-560, *IEEE Journal on Selected Areas in Communications*, May 1998.

[15] R. Ohbuchi, H. Masuda, and M. Aono, Geometrical and Non-Geometrical Targets for Data Embedding in Three-Dimensional Polygonal Models, to appear in August 1998 issue of the *Computer Communications*, Elsevier.

[16] B. Pfitzmann, Information Hiding Terminology, in R. Anderson, Ed., *Lecture Notes in Computer Science* No.1174, pp. 347-350, Springer-Verlag, 1996.

[17] J. R. Smith and B. O. Comiskey, Modulation and Information Hiding in Images, in R. Anderson, Ed., *Lecture Notes in Computer Science* No.1174, pp. 207-296, Springer, 1996.

[18] K. Tanaka, Y. Nakamura, and K. Matsui, Embedding Secret Information into a Dithered Multilevel Image, *Proc. 1990 IEEE Military Communications Conference*, pp. 216-220, 1990.

[19] S. Walton, Image Authentication for a Slippery New Age, *Dr. Dobb's Journal*, pp. 18-26, April 1995.

[20] K. Weiler, The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Modeling, *Geometric Modeling for CAD Applications*, North Holland, pp. 3-36, May 1986.

[21] M. M. Yeung, F. C. Mintzer, G. Braudway, and A. R. Rao, Digital Watermarking For High-Quality Imaging, Proceedings of the *First IEEE Workshop on Multimedia Signal Processing*, Princeton, NJ, USA, June, 1997, pp. 357-362.

[22] M. M. Yeung and F. Mintzer, An Invisible Watermarking Techniques for Image Verification, Proceedings of the *IEEE ICIP '97*, Vol. 2, pp. 680-683, 1997.

[23] J. Zhao and E. Koch, Embedding Robust Labels into Images for Copyright Protection, Proc. of *the Int'l. Congress on Intellectual Property Rights for Specialized Information, Knowledge, and New Technologies*, Vienna, August 1995.