Embedding Data in 3D Models

Ryutarou Ohbuchi, Hiroshi Masuda, Masaki Aono
ohbuchi@acm.org, masuda@trl.ibm.co.jp, aono@acm.org
IBM Japan Tokyo Research Laboratory
1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa, 242, Japan

Abstract. The recent popularity of digital media such as CD-ROM and the Internet has prompted exploration of techniques for embedding data, either visibly or invisibly, into text, image, and audio objects. Applications of such data embedding include copyright identification, theft deterrence, and inventory.

This paper discusses techniques for embedding data into 3D models. Given objects consisting of points, lines, (connected) polygons, or curved surfaces, the algorithms described in this paper produce polygonal models with data embedded into either their vertex coordinates, their vertex topology (connectivity), or both.

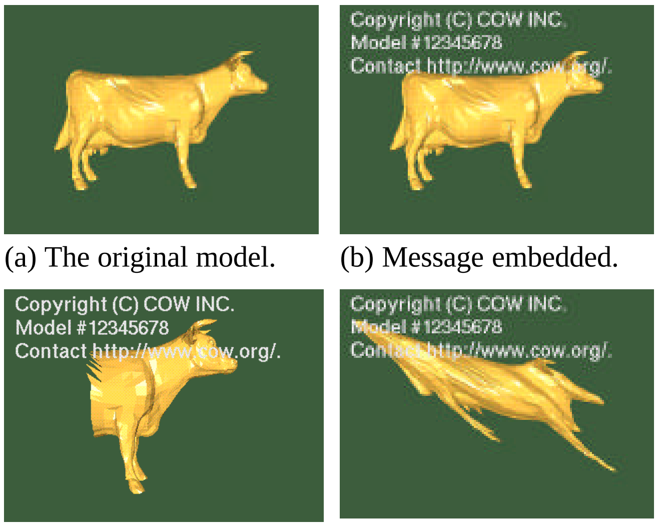
A description of the background and requirements is followed by a discussion of where, and by what fundamental methods, data can be embedded into 3D polygonal models. The paper then presents several data-embedding algorithms, with examples, based on these fundamental methods.

Additional Keywords: 3D graphics, data hiding, digital watermarking, steganography, geometric modeling, copyright protection.

1. Introduction

The advent of digital media, such as CD-ROMs and the Internet, has made possible rapid dissemination of various kinds of data objects, including, images, text, movies, audio data, and recently 3D models. The primary advantage of digital data lie in the ease with which they can be duplicated, distributed, and modified. These advantages, however, have prompted unauthorized duplication and distribution of data.

One way of addressing this problem is to add invisible structures, or *(digital) watermark*, to the data objects. The structures convey information, such as owner identifications or copyright notices. Watermarks can be used, for example, to deter theft, to notify users of how to contact the copyright owner for payment of licensing fees, to discourage unauthorized copying, or to take inventory. The technology associated with adding watermarks is called *steganography*, *(digital) watermarking*, *data embedding*, or *fingerprinting*. In this paper, the act of adding watermark is called *embedding*, and retrieving the information encoded in the watermark for perusal is called *extraction*. Following [Pfitzmann96], the information to be embedded is called *embedded<datatype>*, the object in which the information is embedded is called *cover<datatype>*, and the object with watermark is called *stego<datatype>*. Suffix "<datatype>" varies with data types such as image or text. For example, embedding text data into a still image will be termed as, "embedded-text is embedded into cover-image, producing stego-image".



(a) The original model. (b) Message embedded.

(c) Cut in half. (d) Affine-transformed.

Figure 1. (a) The original model of a cow (5804 triangles). (b) A message is embedded. The message, which is displayed on demand, survives (c) resection or (d) affine transformation.

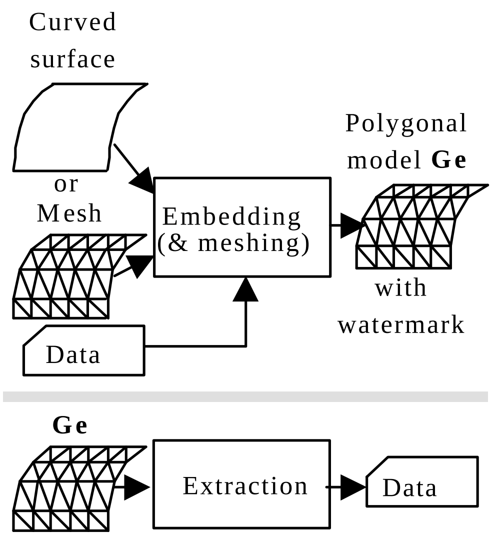


Figure 2. Data are embedded into 3D polygonal models. Embedded data travels with the polygonal model.

Data embedding has been studied for cover-data types of still image, movie image, audio and text data (for example, [Tanaka90, Cox95, Walton95, Berghel96, Braudway96, O'Ruanaidh96, Zhao96]). To the author's knowledge, however, no study has been published on data-embedding techniques for 3D geometrical objects, such as a model described by using Virtual Reality Modeling Language (VRML) [ISO96].

In the past, the comment and annotation capabilities of scene description formats have been the primary means for adding information to 3D models. However, these comments and annotations can be easily removed, either intentionally or unintentionally. For example, programs for converting between 3D model formats often remove comments and annotations. As a result, comments and annotations cannot meet most of the requirements of watermarks.

This paper discusses techniques for embedding data into 3D models, specifically, 3D polygonal models. Inputs of embedding may include curved surfaces, in addition to polygonal geometric models. Watermarks are added to polygonal models by altering their geometry (the coordinates of their vertices) and/or their topology (connectivity of vertices). Figure 1 shows an example of such embedding into a polygonal model of a cow using an algorithm described in Section 3.2. The embedded message, "Copyright (C) COW INC. <CR> Model #12345678 <CR> Contact http://www.cow.org/.", can be extracted and displayed by clicking a button while using a browser enhanced with the extraction capability. The watermark withstands such alterations as Affine transformation and resection.

Data-embedding techniques for 3D models have the following required characteristics, whose priorities vary according to the intended applications:

Unobtrusive: The embedding must not interfere with the intended use of a model, such as viewing. One published example of image watermarking [Braudway96] used the visibility of the watermarks as an advantage, but in most applications, the watermarks must be unnoticeable in terms of the model's intended use.

Robust: Robustness is crucial to the success of data embedding. Using conventional cryptography, data can be made unreadable to third parties [Schneier96]. Making watermarks indestructible, however, is not a trivial problem. With complete knowledge of how watermarks are embedded, any watermarks can theoretically bed removed. With partial knowledge (e.g., the knowledge of the basic algorithm), the removal must be difficult enough so that it either interferes with the intended use of the models or the effort of removal is greater than the value of the model.

What kind of modifications should a watermark in a 3D model resistant to? Data format conversion is a common practice. Floating-point-number representation errors are often introduced during file format conversions. Models are (geometrically) transformed. While transformations are sometimes limited to rotation, uniform-scaling, and translation, more general affine transformations are common. Local deformation is occasionally used to reshape a model. Topological alterations, such as resection of a desired part of a model, remeshing, and polygon simplification, may also be performed. Assuming that the degree of modification is limited so that the utility of the model is not compromised, watermarks in 3D models should *ideally* withstand these and other possible alterations, regardless of whether they are intentional or otherwise.

Space efficient: A data-embedding method should be able to embed a non-trivial amount of information into models whose complexity and value warrants the effort involved.

It is important to note that these three requirements are at odds. For example, in general, more robust watermarks are able to carry less amounts of information. Note also that, depending on applications of watermarks, there could be other requirements, such as reliable identification of source or intended recipient.

2. Fundamental methods of data embedding

2.1. Objects types in 3D models for embedding

A 3D scene model may contain many types of data objects, such as geometry, vertex color, images for texture mapping, vertex normal vectors and even Universal Resource Locator links. We argue that geometry is the best candidate as a target for data embedding since it is the least likely to be removed. While there are many possible representations of 3D geometry, we chose *polygonal models* as the output of embedding for the study reported in this paper (see Figure 2). A "polygonal model" in this paper may include one or more of the following geometrical primitives: points, lines, polygons, connected polygons (e.g., an indexed face set), polyhedrons, and connected polyhedrons. Some data embedding algorithms require topology (connectivity) among points, while others do not. Topology can be added - for example by Delaunay triangulation [O'rourke94] - to polygonal models.

As inputs, embedding algorithms in this paper may accept curved surfaces (e.g., NURBS patches) in addition to polygonal models. The curved surfaces are tessellated into polygonal meshes as they are being embedded with stego-messages.

While we do not discuss details in this paper, components of 3D scenes other than polygonal models of geometry can become targets of data embedding. For example, various other representations of geometry, such as control points of curved surfaces or voxels, can be used as targets of embedding. Depending on the intended use of the model, non-geometrical objects such as per-vertex texture coordinates and per-vertex normal vectors, and to a lesser degree, per-vertex colors or face colors, are viable candidates for data embedding. Image texture, movie texture, and audio data are natural targets for data embedding by using "conventional" data embedding techniques. Note, however, that many of these non-geometrical components of 3D scenes are in general less suitable for data embedding since they can be removed or altered more easily than geometry.

2.2. Embedding primitives

There are two kinds of attributes in a polygonal model that can be modified in order to add watermarks. One is *geometry* of geometrical primitives (e.g., points or triangles) and the other is *topology* of these primitives. Units of modification, either geometrical or topological, are called *embedding primitives* in this paper.

2.2.1. Geometrical primitives

Geometrical values - specifically, the coordinates of points and vertices - can be modified to embed data. Modifications of coordinates change scalar or vector quantities, some of which are classified below according to the transformations to which they are invariant. The *invariance* property is quite useful in constructing data embedding algorithms that are robust against a given class of transformations.

- Altered by all the transformations listed below
  - Coordinates of a point
- Invariant to translation and rotation
  - Length of a line
  - Area of a polygon
- Invariant to rotation, uniform-scaling, and translation
  - Two quantities that define similar triangles (e.g., two angles)
- Invariant to affine transformation
  - Ratio of the lengths of two segments of a straight line
  - Ratio of the volumes of two polyhedrons
- Invariant to projection transformation
  - Cross-ratio of four points on a straight line [Farin96]

Embedding modifies these quantities in such a manner that the modification do not affect the intended uses (e.g., viewing by a browser) of models. For geometrical primitives, this usually means that the amount of a coordinate displacement must remain small. However, this is not always the case. If curved surfaces, instead of polygonal surfaces, are input to an embedding algorithm, the algorithm could have a large degree-of-freedom in number and positions of vertices in output polygonal meshes. The algorithm could exploit this freedom of vertex placement and features of the surfaces (e.g., curvatures) to produce robust watermarks.

2.2.2. Topological primitives

Watermarks can be embedded in the topology of a model. Modifications in this class involve changes in topology, although the geometry might also be changed as a result. Simple examples of topological embedding primitives include encoding of a binary symbol by using two alternative ways of triangulating a quadrilateral, ⧄ and ⧅, or two different mesh sizes, □ and ⊞.

2.3. Arranging embedding primitives

A lone embedding primitive usually can not encode meaningful amount of data. By arranging a set of embedding primitives into an ordered arrangement, a significant amount of information can be embedded into the arrangement of primitives. Examples of arrangements are 1D sequences generated by sorting triangles according to their areas, and 2D arrangements of embedding primitives based on the connectivity of triangles in an irregularly tessellated triangular mesh. A set of arranged primitives could be used to encode an ordered sequence of symbols, such as a character string and binary-number digits, or geometrical patterns, for example, shapes of letters. Note that, conventional targets of data embedding, such as audio and image data, fortunately have implicit ordering. For example, a 2D image has pixels arranged in a regular 2D array. To embed data into 3D polygonal models, however, primitives in the models must be arranged explicitly by embedding algorithms.

Arrangement of primitives can be achieved by the following two methods:

a. **Topological arrangement** employs topology (e.g., connectivity of vertices) to arrange embedding primitives. Topological arrangement requires topology among the embedding primitives. This arrangement is applicable to both topological and geometrical primitives. This arrangement can survive almost any geometrical transformation.

b. **Quantitative arrangement** sorts primitives into arrangement by using inequality relations among quantities associated with embedding primitives. This arrangement is applicable only to geometrical primitives, since it requires quantities for comparison. In order for the arrangement (and thus the watermark) to survive, the quantity used must withstand expected disturbances, such as a class of geometrical transformation.

It is often critical to find an *initial condition* - for example, the first primitive of a one-dimensional arrangement - in order to start an arrangement. The initial condition must be invariant to expected disturbances (e.g., affine transformation.); if the initial condition is changed irrecoverably, watermarks are lost.

Arrangements of embedding primitives can also be classified by their locality into *global*, *local*, and *subscript arrangements.*

a. Global arrangement arranges a set of all the embedding primitives.
b. Local arrangement subdivides the embedding primitives into disjoint subsets, and arranges each subset.
c. Subscript arrangement is similar to local arrangement with very small subsets (e.g., a few primitives per subset). Each subset in this case is called a *macro-embedding-primitive*, which embeds {data-symbol, subscript} pair so that the subscript identifies data symbol's position in an arrangement.

In the latter two arrangement methods, a set of primitives are grouped into a subset by proximity, either topological or quantitative, of their members.

3. Embedding algorithms

In the previous section, we have discussed fundamental methods for embedding data into 3D polygonal models. These fundamental methods can be combined to generate practical data-embedding algorithms. This section describes two such algorithms.

Both algorithms are implemented by using a kernel for a non-manifold modeler [Masuda96]. Using *radial edge* structure [Weiler86] to represent topological relationships among vertices, edges, faces, and regions, the modeler efficiently performs such operations as computation of the spanning tree of vertices on a triangular mesh.

3.1. Triangle similarity quadruple embedding

A pair of dimensionless quantities, for example, {b/a, c/a}, {S/{a*a}, b/c}, or $\{q_1,q_2\}$ in Figure 3a, defines a set of similar triangles. One of these pairs can be used as a primitive to watermark triangular meshes so that the watermarks are robust against translation, rotation, and uniform-scaling transformations. Combining this primitive with subscript arrangement and repeated embedding of a message, the watermark becomes resistant to resection or local deformation of the stego-3D-models.

In order to realize subscript ordering, a quadruple of adjacent triangles that share edges (Figure 3b) is used as a macro embedding primitive that stores a pair of data symbols, a marker, and a subscript together. A marker is a special value (or values) that identifies macro embedding primitives. This algorithm, which is called Triangle Similarity Quadruple (TSQ) algorithm, embeds a message according to the following steps.

(1) Find a macro embedding units on the input triangular mesh. In doing so, avoid triangles that have already been used for the watermark. Avoid triangles that are unfit for stable embedding, e.g., if its two dimension-less quantities are either too small.
(2) For each macro embedding unit, embed *{subscript, mark, data1, data2}* quadruple by displacing vertices by small amount.
(3) Repeat (1) and (2) above until all the data symbols of a message are embedded.

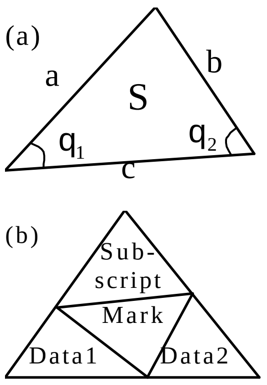


Figure 3. (a) A pair of dimension-less quantities defines a set of similar triangles. (b) A macro-embedding-primitive.

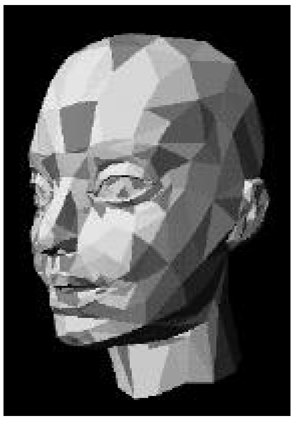


Figure 4. Triangles in dark gray are the macro-embedding-primitives.

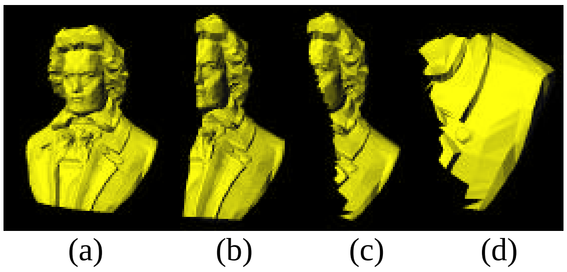


Figure 5. A model of a Beethoven's bust (4889 triangles) resected repeatedly.

	No. of triangles	Data remained intact
(a)	4889	6 copies, 132 bytes each
(b)	2443	132/132 bytes
(c)	1192	102/132 bytes
(d)	399	85/132 bytes

Table 1. Data loss due to resection in the example shown in Figure 5.

In order to embed multiple copies of a message, steps (1) to (3) are repeated many times. Each repetition of a message is embedded in a topological proximity.

The TSQ extraction algorithm does not require the original cover-3D-model for extracting embedded message. The algorithm traverses all the triangles in the model to find triangles with markers. Each marker identifies a macro-embedding-primitive, which contains two data symbols and a subscript. The subscript puts the symbol pair at a proper place in the sequence of extracted message symbols.

Figure 5a shows a model of Beethoven's bust (4889 triangles, 2655 vertices) in which six identical copies of a message, each of which consists of 132 bytes, have been embedded by using the TSQ algorithm. Figure 5b-c shows the result of resection by arbitrary planes (the first one just happened to cut the model in half at the medial line.) As shown in Table 1, cutting the model in half left the message intact. After the model has been roughly quartered 102 bytes out of 132 bytes remained. Note that, since a subscript arrangement was used, intact characters still tended to be in the correct positions within the message string.

3.2. Tetrahedral volume ratio embedding

A ratio of volumes of a pair of tetrahedrons is the embedding primitive for the Tetrahedral Volume Ratio (TVR) embedding algorithm described in this section. The algorithm is designed to accept triangular meshes as input and embedding primitives are ordered topologically into a global one-dimensional arrangement to embed a sequence of symbols. The algorithm does not require cover-3D-model for extraction, and the watermark survives affine transformation. The TVR algorithm, however, is not robust with respect to topological modifications such as resection and remeshing.

The crux of the TVR algorithm is establishing global one-dimensional ordering of embedding primitives. This is accomplished in accordance with the following steps;

(1) Find a spanning tree of vertices *Vt*, called *vertex tree*, on the input triangular mesh *M*, given an initial condition *Ivt* for *Vt*. Convert *Vt* into a sequence of triangles *Tris*, called a *triangle sequence.*

(2) Convert *Tris* into a sequence of tetrahedrons *Tets*, called a *tetrahedron sequence*. To do this, compute a common apex as the centroid of the coordinates of a few triangles selected from the triangle sequence. The selected triangles are removed from the triangle sequence so that their coordinates are not modified by embedding.

(3) Convert *Tets* into a sequence of ratios of volumes *Vrs*. To do this, a volume of a tetrahedron (e.g., the first one) in *Tets* is selected as a common denominator of all the ratios, and volumes of the remaining tetrahedrons are used for numerators.

(4) Embed a symbol into each ratio by displacing vertices of numerator-tetrahedrons. The vertex displacements for the current symbol must not interfere with modifications of the previously embedded symbols. (In Figure 8, triangles that are used for embedding, which are colored dark gray, do not share edges because of this constraint.)

We now explain details of the first step, starting with the method to create a triangle sequence, and later come back to explain how to find an initial condition *Ivt*.

Generating vertex tree *Vt* from an input triangular mesh requires that the input mesh is an orientable manifold. To generate *Vt*, traverse vertices from a given initial vertex in the initial traverse direction, starting with the *Vt* initialized to empty. At each vertex, by scanning the edges in counter-clockwise order, find an edge that is not a member of *Vt* and does not loop back to any of the vertices covered by *Vt*. If such an edge is found, add it to *Vt*. Figure 6 shows an example of a vertex tree, in which vertex 1 is the initial vertex.

The vertex tree *Vt* is converted into the triangle sequence *Tris* as a set of edges *Tbe*, called a Triangle Bounding Edge (TBE) set is constructed. The *Tbe* is initialized to a set

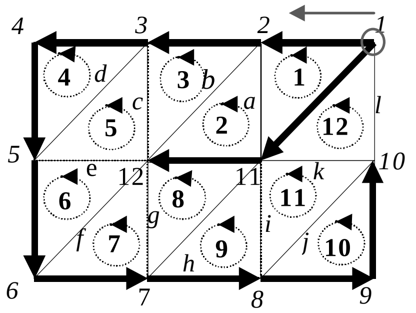


Figure 6. An example of a vertex tree (vertices are numbered), a triangle bounding edge set (edges are numbered as ↙1), and a triangle sequence (triangles are numbered as ①) for a triangular mesh.

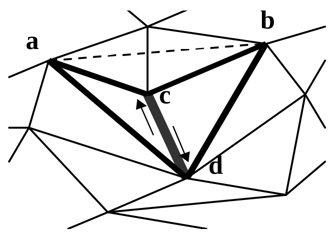


Figure 7. Volume of a tetrahedron a-b-c-d, which is subtended by two triangles a-d-c and b-c-d that are adjacent to the edge c-d, is computed. (Two arrows indicate possible alternative initial traverse directions given an initial edge c-d.)

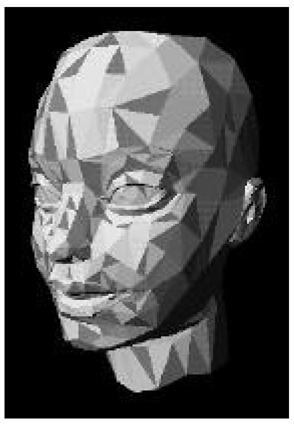


Figure 8. Triangles used for embedding by the TVR algorithm are shown in dark gray.

of edges that connect vertices in the *Vt*. To add an edge to the *Tbe*, vertices are traversed according to the *Vt*, starting from the root. At each vertex, all the edges adjacent to the vertex are scanned clockwise, and the scanned edge is added to the *Tbe* if it is not a member of the *Tbe*. A new triangle is added to the *Tris*, which started as an empty sequence, if all three edges of the triangle are in the *Tbe* for the first time, and the triangle is not already in the *Tris*. In the example shown in Figure 6, edges (except the initial entries of the *Tbe*) are marked by alphabets in the order of addition to the *Tbe* using alphabetical ordering. In the figure, members of the *Tris* are marked by numbers in circles.

The algorithm selects an initial *edge*, instead of an initial vertex and an initial traverse direction, using the method described below. Using an initial edge as initial condition allows two possible alternatives to start traverse of vertices to construct a vertex tree. This ambiguity is resolved by a trial-and-error method. The algorithm extracts sequence of symbols by using both alternatives, and choose the direction that yielded correct predetermined lead-in symbol sequence.

To select an initial edge, the algorithm computes, for every edge in the model, the volume of the tetrahedron subtended by the two triangles that are adjacent to the edge (Figure 7). The algorithm selects, as the initial edge, the edge for which tetrahedron’s volume is the largest. (Note that these tetrahedrons are different from the ones used to embed symbols.) Since the initial edge with the largest volume may be incorrect due to noise and other reasons, the TVR algorithm employs the trial-and-error method again. The algorithm tries multiple candidate edges until it finds the correct lead-in sequence.

The TVR algorithm can be made resistant to resection and local deformation by using a local or subscript arrangement method combined with repeated embedding of a message. Examples shown in Figure 1 used a strengthened version of the TVR algorithm by using a local arrangement and repeated embedding of a message.

4. Summary and Future Work

We have applied the concept of data embedding to 3D polygonal models. While data-embedding techniques have previously been studied for text, image, and audio data objects, to the author’s knowledge, our work is the first to study data embedding into 3D geometrical models.

After describing the background and requirements, we argued that geometry is the best data type in a 3D model for embedding. We then chose polygonal models as targets of embedding for this study. We presented fundamental methods for embedding data into polygonal models, namely, geometrical and topological modification primitives and methods for introducing ordering into a set of modification primitives. Finally, we presented two data-embedding algorithms and gave examples of their execution, thus demonstrating that data embedding into 3D polygonal models is a practical technique.

Much work in this area remains to be done. Our embedding algorithms are meant to be examples of simply embedding data into 3D polygonal models. Ideas such as spread spectrum communication [Cox96, Smith96], scrambling of symbol sequences by pseudorandom number sequence, and others may prove valuable in order to improve security, reliability, and robustness of embedding algorithm. We would also like to investigate data embedding into targets other than polygonal models, for example, control parameters of curved surfaces. In doing so, we need to understand requirements of applications other than viewing; for example, a manufacturing CAD system should have different criteria for unobtrusiveness than a simple browser of 3D polygonal models.

References

[Berghel96] H. Berghel and L. O'Gorman, Protecting Ownership Rights Through Digital Watermarking, *IEEE Computer*, July 1996, pp101-103.

[Braudway96] 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.

[Cox95] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, *Technical Report 95-10*, NEC Research Institute, 1995.

[Farin96] G. Farin, *Curves and Surfaces for CAGD: a Practical guide*, 4th edition, Academic Press, 1996.

[ISO96] ISO/IEC JTC1 SC24/N1596 CD #14772 Virtual Reality Model Language (VRML 2.0)

[Masuda96] 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).

[O'Rourke94] J. O'Rourke, *Computational Geometry in C*, Cambridge University Press, 1994.

[O'Ruanaidh96] J. J. K. O'Ruanaidh, W. J. Dowling, and F. M. Boland, Watermarking digital images for copyright protection, *IEE Proceedings on Vision, Image, and Signal Processing*, Vol. 143, No. 4, pp.250-256.

[Pfitzmann96] B. Pfitzmann, Information Hiding Terminology, in R. Anderson, Ed., Lecture Notes in Computer Science No.1174, pp347-350, Springer-Verlag, 1996.

[Schneier96] B. Schneier, *Applied Cryptography*, Second edition, John Wiley & Sons, Inc., New York, 1996.

[Smith96] J. R. Smith and B. O. Comiskey, Modulation and Information Hiding in Images, in R. Anderson, Ed., Lecture Notes in Computer Science No.1174, pp207-296, Springer-Verlag, 1996.

[Tanaka90] K. Tanaka, Y. Nakamura, and K. Matsui, Embedding Secret Information into a Dithered Multilevel Image, *Proc. 1990 IEEE Military Communications Conference*, pp216-220, 1990.

[Walton95] S. Walton, Image Authentication for a Slippery New Age, *Dr. Dobb's Journal*, April 1995.

[Weiler86] K. Weiler, The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Modeling, *Geometric Modeling for CAD Applications*, North Holland, pp3-36, May 1986.

[Zhao96] J. Zhao and E. Koch, Embedding Robust Labels into Images for Copyright Protection, *Proc. of the Int. Congress on Intellectual Property Rights for Specialized Information, Knowledge, and New Technologies*, Vienna, August 1995.