REDUNDANCY IN 3D POLYGON MODELS AND ITS APPLICATION TO DIGITAL SIGNATURE

S. Ichikawa, H. Chiyama, and K. Akabane[1]

Department of Knowledge-based Information Engineering
Toyohashi University of Technology
Hibarigaoka, Tempaku, Toyohashi, Aichi, 441-8580,
JAPAN
Email: ichikawa@tutkie.tut.ac.jp
URL: http://meta.tutkie.tut.ac.jp/~ichikawa/index-e.html

ABSTRACT

This paper introduces new watermarking schemes for 3D polygon models. The schemes presented here use just the redundancy in model description, hence essential model data such as vertex coordinates and topology are left intact. Proposed Full Permutation Scheme (FPS) and Partial Permutation Scheme (PPS) embed information by permuting the order of vertices or faces, and Polygon Vertex Rotation scheme (PVR) embeds information by rotating the vertices of faces. The digital signature procedure for verification purposes is also presented, which works in cooperation with popular public-key cryptography. A modified PVR scheme (Packet PVR) is then proposed for more robust signature. Evaluation results show that our schemes can embed 0.2% (by Packet PVR) – 2.8% (by FPS) of information compared to the original model file size. Our methods are orthogonal and complementary to the preceding methods that use geometrical and topological domain.

Keywords: Watermarking, Polygon Models, Data Embedding, Model Verification, Public-key Cryptography

1 INTRODUCTION

With the explosive spread of computers and networks, digital multimedia materials have become commonplace and widespread. There is an urgent demand to protect these digital assets from unfavorable use. The digital watermarking technique is regarded as a potential solution to this problem, and is widely researched on various kinds of multimedia contents. In particular, watermarking for audios, digital still images, and video images have been extensively investigated. More information may be found in the references [Yeung98b] [Hartu99], which provide a good overview and survey on multimedia watermarking techniques.

In this paper, new watermarking schemes for 3D polygon models are proposed and examined. The purpose of this research is to provide the means for creators to make a digital signature [Garfi95] [Schne96] on their products, along with providing the means for users to validate the integrity of the polygon data provided by various creators.

For example, let us consider a vendor, wishing to distribute a reference 3D model of a mechanical part to customers. The vendor wants customers to use the model data intact, because the use of modified or corrupt data can cause serious mismatches between the vendor and customer later. Customers will also benefit from using certified model data to avoid unexpected modification, damage, and confusion of models.

It is well known that such a certification mechanism is provided by digital signature with public-key cryptography [Garfi95] [Schne96]. The problem is how to sign a model and how to manage the signature with the model. It is trivial to write a signature on a model *file*, but it is not simply

[1]Presently with Tokai Co. Ltd.

applicable to structured models, which are often built by gathering ready-made components. In such cases, we wish to preserve the original signatures of components. Simply by signing a model file, the final structured model can be certified, but certificates of each component are purged. Maybe we can insert a signature into a model file as a comment, but such comments are not necessarily preserved by CAD tools. Thus, we must find a way to embed the signature into the model itself, instead of writing a signature in the model file.

This study focuses on how to embed/retrieve a signature and how to provide a feasible system to 3D model users. Our method is *fragile* and *public* in watermarking terminology, and is not intended to be robust as it is. Nevertheless, our scheme does not prevent using other robust watermarking technology at the same time, because our method is transparent to any geometrical and topological watermarking method.

In the next section, the preceding watermarking techniques for polygon models are examined to clarify the difference and original contribution of this paper. Then, a new watermarking scheme and its application are described in Section 3. This method utilizes the redundancy in polygon models, leaving essential model data intact. Section 4 presents another redundancy to embed information, which could be used with the scheme described in Section 3.

2 RELATED WORKS

Watermarking on 3D models is a relatively new research theme. The first watermarking research on 3D models was presented by Ohbuchi et al. [Ohbuc97] [Ohbuc98] They presented three techniques: Triangle Similarity Quadruple Embedding (TSQ), Tetrahedral Volume Ratio Embedding (TVR), and Mesh Density Pattern Embedding (MDP). TSQ and TVR embed watermarks in the geometrical domain, while MDP uses a topological domain by remeshing.

Kanai et al. [Kanai98] proposed a method based on wavelet transform and multiresolution representation. Praun et al. [Praun99] also presented a method based on multiresolution surface representations. These techniques embed data in the frequency domain by perturbing geometrical data.

Benedens [Bened99] [Bened00] presented Normal Bin Encoding (NBE) and Affine Invariant Embedding (AIE), both of which are based on mesh surface normals. Wagner [Wagne00] embedded watermarks in the relative length of vector norms, where only vertex coordinates are modified. Ohbuchi et al. [Ohbuc01] presented a watermarking technique in mesh spectral domain using eigen vector decomposition.

All these methods are geometrical or topological, and modify the original model data to embed the watermark by modifying vertex coordinates imperceptively or changing the construction of faces by remeshing. Our method, on the other hand, preserves model data as is, except for the harmless change in the order of description.

Another difference is that most of these methods are intended to be robust, or resistant to intentional and unintentional modification. A robust watermark will persist despite attempts at removal, but there are some applications that do not require such robustness. For example, it is enough for verification purposes that the modification on the original can be detected, even if the watermark is lost by this modification. This kind of watermark is called “fragile”.

Fragile watermarking for 3D models was first proposed by Yeung and Yeo [Yeung98a] [Yeo99] for verification purposes. In their method, a watermark is embedded by perturbing the coordinates of objects. The watermark is encoded by using a secure (secret) key before embedding. Benedens [Bened00] presented a Vertex Flood Algorithm (VFA) for model authentification. VFA embeds the watermark into the distances between vertices and the center of mass of a start triangle by modifying vertex coordinates. In short, all these methods [Yeung98a] [Yeo99] [Bened00] modify geometrical data to embed information.

Both our method and the preceding ones [Yeung98a] [Yeo99] [Bened00] are fragile and intended for verification use. However, our method uses neither geometrical nor topological data for embedding, hence preserving the essential model data intact.

3 WATERMARKING BY PERMUTATION

3.1 Redundancy in Polygon Models

There are many kinds of polygon model representations, but the fundamental form is the same. A polygon model consists of vertices, faces, and additional information. Vertices are defined by coordinate values, and faces are defined by a sequence of vertices. Other information such as texture is also included. An example of a polygon model skeleton is shown below, which is described in VRML [Int97].

```
IndexedFaceSet {
  coord Coordinate {
    point [
      $x_0\ y_0\ z_0$,
      $x_1\ y_1\ z_1$,
      ...
    ]
  }
  coordIndex [
    $v_i$, $v_j$, $v_k$, $-1$,
    ...
  ]
}
```

Each vertex (*point*) is represented by three sets of coordinate values (x_i, y_i, z_i). Each face (*polygon*) is represented by a sequence of vertex indices $(v_i, v_j, v_k, ...)$, which is terminated by -1.

There is no required order of vertices. If we have n vertices, we have $n!$ options to describe the set of n vertices. This means that $\lfloor \log_2 n! \rfloor$ bits of information can be carried by selecting one of the options. The same redundancy exists in face description, as no specific order is required. We can carry $\lfloor \log_2 n! \rfloor$ bits by n faces in the same way as in vertices. Face descriptions depend on vertex indices, so one must adjust the description of faces according to the selected order of vertices. However, this does not prevent utilizing the redundancy in face description. After adjusting vertex indices, one can choose any order of faces to carry information.

3.2 Full / Partial Permutation Scheme

A serious drawback of this approach is that it takes a rather long time for a large n to encode/decode information to/from a permuted sequence. The way to improve it is to cut up the set of n items (vertices or faces) into small chunks. Let k items make up a chunk of items. We encode/decode information per k items, and repeat it for $\lfloor n/k \rfloor$ chunks. This scheme gives up some of the possible capacity, while shortening the processing time.

Let us call the original scheme "Full Permutation Scheme" (FPS), and the new scheme "Partial Permutation Scheme" (PPS). The information capacity $F(n)$ that is carried by n items with FPS is given by Eq. 1, and the capacity $P(n,k)$ of PPS is given in Eq. 2.

$$F(n) = \lfloor \log_2 n! \rfloor, \quad (1)$$
$$P(n,k) = \left\lfloor \frac{n}{k} \right\rfloor \lfloor \log_2 k! \rfloor. \quad (2)$$

Fig. 1 displays $F(n)$ and $P(n,k)$, where $k = 8$. A chunk of 8 items can carry 15 bits. Encoding is easily realized by preparing a look-up table of 2^{15} entries. Decoding is also easy by preparing an 8-stage decision tree.

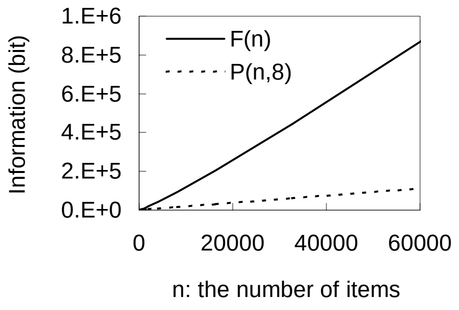

Figure 1: Capacity of Permutation

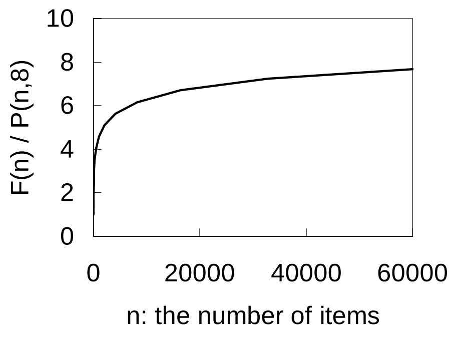

Figure 2: The Ratio of F(n) to P(n,8)

Though the capacity of PPS looks poor compared to FPS, there is little difference between them. Fig. 2 shows that $F(n)/P(n,8) < 8$ holds for $n < 60000$, and that $F(n)/P(n,8)$ does *not* grow in proportion to n. Rather, it is proportional to the logarithm of n, which grows slowly. This is rationalized by the following explanation.

The value of $n!$ is approximated by Stirling's formula:

$$n! \approx \sqrt{2\pi n} \cdot n^n \cdot e^{-n}. \quad (3)$$

According to Eq. 1 and Eq. 3, $F(n)$ is $O(n \log n)$. On the other hand, $P(n,k)$ is $O(n)$ from Eq. 2, provided that k is constant. Thus $F(n)/P(n,k)$ is $O(\log n)$.

3.3 Evaluation

The previous section outlined the fundamentals of our idea. Now, the idea has to be examined on real 3D models. We pick up five polygon models, that are shown in Table 1. All these models are originally acquired from WWW as public VRML model file (.wrl) [Int97], and then converted to DirectX model file (.x) by Crossroads 3D [Rule97] and Direct3D Retained Mode library [Micro97] in our development environment for later processing.

File name	Vertex	Face	Size (byte)
x_wing.x	3099	6084	506107
satelite.x	4304	4572	583692
Briefcase.x	4786	6180	654457
purple-walter.x	5314	5312	715340
panzer.x	20424	42333	3746982

Table 1: 3D Models

Table 2 shows the evaluation results on these model files. F_v and F_f are the FPS capacity (bit) on vertices and faces, respectively. P_v and P_f are that of PPS of $k = 8$. In Table 2, WD means Watermark Density, that is the ratio of total watermark capacity ($F_v + F_f$ or $P_v + P_f$) to file size. As seen from Table 2, the watermark capacity is about 2–3% by FPS, and about 0.4% by PPS. Watermarks of 10^5–10^7 bits are generally more than suffice for verification purposes.

3.4 Embedding and Retrieving Digital Signature

This section outlines how to embed and retrieve data. As described in Section 1, the digital signature technique of public-key cryptosystem [Garfi95] [Schne96] is adopted here. It is natural to use the appropriate one-way hash function jointly, as we can carry limited capacity of information with polygon models.

Digital signature on polygon model is performed as follows.

1. Sort vertices in geometrical order.
2. Calculate vertex signature value of sorted vertex data.
3. Encode vertex signature with creator's secret key.
4. Permute (or renumber) vertices according to encoded vertex signature data.
5. Adjust face descriptions according to renumbered vertices.
6. Sort faces by vertex indices.
7. Calculate face signature of sorted face data.
8. Encode face signature with creator's secret key.
9. Permute faces according to encoded face signature data.

In fact, there are many options in this procedure. For example, sorting of vertices and faces could be in any order, if reproducible on the consumer side.

Verification is realized by the following process.

1. Sort vertices in geometrical order.
2. Calculate vertex signature value of sorted vertex data.
3. Retrieve encoded vertex signature value from unsorted vertex data.
4. Decode the encoded vertex signature with creator's public-key to obtain original vertex signature.
5. Verify the calculated vertex signature against the original vertex signature.
6. Sort faces by vertex indices.
7. Calculate face signature value of sorted face data.
8. Retrieve encoded face signature value from unsorted face data.
9. Decode this encoded face signature with creator's public-key to obtain original face signature.
10. Verify the calculated face signature against the original face signature.

One should recall that our method is "fragile". Renumbering of vertices and faces easily destroys the encoded signature, but it is still detected as signature mismatch or signature format error. Addition or deletion of vertices and faces will also be detected, because such operations destroy the embedded signature. Modification of coordinates or vertex indices results in a different signature from the encoded one, thus detected. Forgery of a signature is practically impossible, as long as

File name	Full Permutation Scheme (FPS)				Partial Permutation Scheme (PPS)			
	F_v	F_f	$F_v + F_f$	WD (%)	P_v	P_f	$P_v + P_f$	WD (%)
x_wing.x	31477	67711	99188	2.45	5805	11400	17205	0.42
satelite.x	45753	49000	94753	2.03	8070	8565	16635	0.36
Briefcase.x	51609	68918	120527	2.30	8970	11580	20550	0.39
purple-walter.x	58104	58080	116184	2.03	9960	9960	19920	0.35
panzer.x	262973	589572	852545	2.84	38295	79365	117660	0.39

Table 2: Watermarking by Permutation

the adopted public-key cryptosystem is safe and the secret-key of creator is properly protected.

Our method only guarantees that the model is intact or modified. It can not determine how much the model is modified or where. If such information is important, the method of Yeo and Yeung [Yeo99] is worth considering, because it can give hints of how much and where. Yeo and Yeung's method embeds data by perturbing vertex coordinates, so it presumably fits with our scheme that does not use geometrical domain at all.

One problem remains with our scheme. We adopted a public-key cryptosystem for the digital signature. The creator's public-key is necessary to verify the model. It does not make sense to encode the public-key of the creator in the model, because an embedded public-key has to be verified against the creator's public-key anyway to rely on the verification result. Thus, users must access the creator's public-key by another reliable means by themselves. This means that user must know the creator's information along with the model itself. It is impossible to access a creator's public-key without knowing who the creator is.

In the next section, we propose another scheme to encode information in polygon models. We can incorporate some additional information by this method; e.g. creator's name, version, release date, copyright notice, license condition, and contact address.

4 WATERMARKING BY POLYGON VERTEX ROTATION

This section describes another redundancy of the polygon model, which exists in polygon descriptions. The new technique is entirely orthogonal to the FPS and PPS described in the previous section. Thus, it may be used jointly.

4.1 Redundancy in Face Description

There is no one way to describe a polygon. For example, all of three descriptions (v_i, v_j, v_k), (v_j, v_k, v_i), and (v_k, v_i, v_j) represent the same triangle. As the faces have a front side and back side, three equivalent descriptions (v_i, v_k, v_j), (v_j, v_i, v_k), and (v_k, v_j, v_i) are different from (v_i, v_j, v_k). In general, a polygon with n vertices can be represented in n ways, as rotating vertices of the polygon. We can embed information by utilizing this redundancy. Let us call this scheme Polygon Vertex Rotation (PVR).

PVR can cooperate with FPS or PPS described in Section 3, if the sorting of faces is properly performed in FPS and PPS. In step 6 of the embedding procedure (Section 3.4), the sorted result of faces must be insensitive to the vertex rotation of each face. Such sorting procedure can be realized as follows: (1) Normalize vertex rotation first, (2) then sort faces according to vertex indices. A face description can be normalized by rotating vertices until the vertex of the lowest number of the index reaches first place. To embed data by PVR with FPS/PPS, a new step 6' is required between step 6 and step 7 in Section 3.4.

6' Embed data by rotating vertices of each face.

Data retrieval from PVR is performed as follows.

1. Sort faces by normalized vertex indices.
2. Retrieve data from vertex indices of faces, according to the sorted order.

It should be noted that the retrieval from PVR is independent of PPS or FPS. Thus, PVR can be used to carry various information such as creator information for PPS/FPS. Forgery of PVR data is impossible, if PVR is used with PPS or FPS. PPS and FPS protect the face description from modification by the digital signature, so any attempt to change PVR data is detected by PPS or FPS as a signature mismatch.

4.2 Full PVR and Partial PVR

Let Φ be the set of faces of a polygon model, and $\nu(\phi)$ the number of vertices of a face $\phi \in \Phi$. Taking all faces into account, the total degree of freedom of the model is given by $\prod_{\phi\in\Phi} \nu(\phi)$. Thus, the information capacity $\Theta(\Phi)$ of a face set Φ is given by the following equation:

$$\Theta(\Phi) \quad = \quad \left\lfloor \textstyle\sum_{\phi\in\Phi} \log_2 \nu(\phi) \right\rfloor . \qquad (4)$$

Let us call this method "full" PVR (FPVR). FPVR has the same problem as FPS. For example, we must handle such high numbers as 3^{100} to encode/decode data for a model consisting of 100 triangles. Hence we examine "partial" PVR (PPVR), which partitions Φ into chunks of faces.

Partitioning of Φ is simple, where the model consists of the same kind of polygons: e.g., only triangles. In this case, any chunk of k faces carry $\lfloor k \ \log_2 \nu(\phi) \rfloor$ bit, since $\nu(\phi)$ is constant. However, general polygon models consist of various kinds of polygons. In this general case, we have many options to partition Φ. This makes the capacity of each chunk different, and thus the total capacity is dependent on the way of partitioning of Φ.

Now, let us concentrate here on the case of $k = 2$ for simplicity. Given two faces (ϕ_1, ϕ_2), the chunk capacity is calculated as $\lfloor \log_2\{\nu(\phi_1)\nu(\phi_2)\} \rfloor$. Let $\theta(\Phi)$ be the total capacity of PPVR for Φ. $\theta(\Phi)$ is represented as follows, if the set of (ϕ_1, ϕ_2) for Φ is given.

$$\theta(\Phi) \quad = \quad \sum_{(\phi_1,\phi_2)} \lfloor \log_2 \nu(\phi_1) + \log_2 \nu(\phi_2) \rfloor . (5)$$

The partitioning of Φ must be reproducible on the customer's side. Under this condition, basically any way of partitioning Φ is acceptable.

We adopt the following heuristics to partition Φ in the following discussion.

1. Sort faces according to normalized vertex indices. [2]
2. Mark all faces "unused".
3. Take two faces that are topologically adjoining from the top of sorted face list. Mark selected pair of faces "used".
4. Repeat Step 3 and take as many pairs as possible.

[2]See Section 4.1 for normalization of vertex indices.

This is a kind of greedy algorithm, and does not guarantee the quality of the solution. Also, it can leave some faces unused. However, it is still regarded good enough according to the evaluation results shown in Table 3.

Table 3 presents some evaluation results of PVR. Θ is the capacity of FPVR, and θ is the capacity of PPVR of $k = 2$ with greedy partitioning. It is apparent that Θ is the upper bound of θ, but the ratio θ/Θ shows that PPVR utilizes 80–100% [3] of the possible capacity $\Theta(\Phi)$. Watermark Density (WD) is the ratio of θ to the corresponding file size. WD of PPVR is almost half of the WD of PPS, compared to Table 2.

4.3 Packet PVR

As mentioned earlier, FPVR and PPVR are not robust. At the end of this paper, we present a modified PVR scheme more robust than FPVR and PPVR.

In a communication area, a *packet* is organized to make data transfer more reliable. Usually, the packet consists of a preamble, sequence number, contents, check sum, etc. Longer data are split into a couple of packets, and transferred as units of original data. If some of the packets are damaged or lost, the corresponding packets are retransmitted and reconstructed with other packets. We may apply this idea to polygon model. Let us call this scheme "packet PVR". Fig. 3 illustrates the concept of the polygonal packet. Let

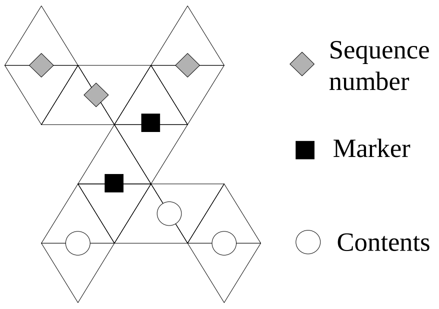

Figure 3: Polygonal Packet

[3]In purple-walter.x, θ is equal to Θ. This is not a mistake, but caused by rather exceptional conditions. First, our greedy algorithm counted all faces for pairs. Second, this model consists of only tetragons. In such a case, $\Theta = \theta$ holds, because $\log_2 \nu(\phi)$ is exactly 2.

File name	θ (bit)	Θ (bit)	θ/Θ	File size (byte)	WD (%)
x_wing.x	8790	9642	0.91	506107	0.22
satelite.x	8022	8751	0.92	583692	0.17
Briefcase.x	7932	9795	0.81	654457	0.15
purple-walter.x	10624	10624	1.00	715340	0.19
panzer.x	66714	73251	0.91	3746982	0.22

Table 3: Watermarking by Polygon Vertex Rotation

us make things simple and consider a model consisting of just triangles. In Fig. 3, 16 triangles are used to form a polygonal packet. The encoding scheme is based on PPVR of $k = 2$. Since $3^2 = 9 > 8 = 2^3$, three bits of information (0–7) are encoded in a pair of triangles. Moreover, one code is left for control use (*void*), because a pair of triangles can represent 9 codes.

In Fig. 3, eight symbols are indicated. Each represents 3 bits of information encoded between two triangles. A marker has a *void* value, and corresponds to the preamble or trailer of communication packet. Two markers announce the existence of a packet. Contents are represented by 3 pairs of triangles, and thus contain 9 bits of data. One of 9 bits is used as a check bit to distinguish the contents and sequence number. If the check bit is zero, the other 8 bits are sequence number. If the check bit is one, the others are regarded as contents. A sequence number is used to reconstruct the original message from the contents of many packets.

In packet PVR, the encoding procedure is easy. We just scan the model and find 16 unused triangles that are connected in the specified relationship; then we embed the markers, sequence number, and contents. To decode, we have to scan the value of triangle pairs exhaustively. If we find two *void* values in the specified locations, and if all 16 triangles exist in the specified relationship, we decode the sequence number and contents according to check bit value.

Packet PVR is not perfect, but more robust than FPVR, PPVR, FPS, and PPS. These four schemes are easily destroyed by renumbering, adding or deleting vertices and faces, because they depend on the order of description. On the other hand, packet PVR allows renumbering and removal of vertices and faces to some extent.

The evaluation results of packet PVR are shown as ψ in Table 4. The models examined here are partially different from the models in previous sections, because the packet PVR presented here can only handle polygon models consisting only of triangular faces. The results of FPVR (Θ in Fig. 4) and those of PPVR (θ) are also shown for comparison. The capacity of packet PVR is about one sixth that of PPVR, but this would be a reasonable drawback of robustness.

File name	vertex	face	Θ	θ	ψ
scud.x	136	240	380	360	64
tiger.x	303	602	954	903	152
chess1.x	389	772	1223	1158	216
gumby.x	406	760	1204	1140	208
x_wing.x	3099	6084	9642	9126	1680

Table 4: Watermarking by Packet PVR

5 CONCLUSION

The many studies on watermarking of 3D polygon models have used geometrical or topological domain to embed data. In the present investigation, we presented new watermarking schemes that utilize the redundancy of polygon models to carry information. The framework for digital signature on polygon models was also presented. To our knowledge, no researcher has pointed out this kind of watermarking techniques to date. This is the original contribution of this paper.

Five watermarking schemes (FPS, PPS, FPVR, PPVR, and Packet PVR) are presented and evaluated on real 3D polygon models. FPS and PPS can be used to protect the integrity of model data with the digital signature of a public-key cryptosystem. Three PVR schemes can carry additional information, and work with FPS and PPS.

These schemes only use redundancy of description, so other watermarking schemes that use geometrical and topological domain are also usable with our schemes.

ACKNOWLEDGMENTS

This work was partially supported by a grant from the Telecommunications Advancement Foundation and a Grant-in-Aid from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

REFERENCES

[Bened99] O. Benedens. Geometry-based watermarking of 3D models. *IEEE Computer Graphics and Applications*, 19(1):46–55, 1999.

[Bened00] O. Benedens and C. Busch. Towards blind detection of robust watermarks in polygonal models. In *Proc. EUROGRAPHICS 2000*, pages C199–C208. Blackwell, 2000.

[Garfi95] S. Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly, 1995.

[Hartu99] F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(7):1079–1107, July 1999.

[Int97] International Standard ISO/IEC 14772-1:1997. *The Virtual Reality Modeling Language*, 1997. http://www.web3d.org/fs_specifications.htm.

[Kanai98] S. Kanai, H. Date, and T. Kishinami. Digital watermarking for 3D polygons using multiresolution wavelet decomposition. In *Proc. Sixth IFIP WG 5.2 International Workshop on Geometric Modeling: Fundamentals and Applications (GEO-6)*, pages 296–307, 1998.

[Micro97] Microsoft Corp. *Direct3D*, 1997.

[Ohbuc97] R. Ohbuchi, H. Masuda, and M. Aono. Embedding data in 3D models. In *Proc. European Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS '97)*, LNCS 1309. Springer, 1997.

[Ohbuc98] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygonal models through geometric and topological modifications. *IEEE Journal on Selected Areas in Communications*, 16(4):551–560, 1998.

[Ohbuc01] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama. Watermarking 3D polygonal meshes in the mesh spectral domain. In *Proc. Graphics Interface 2001*, pages 9–17. Morgan Kauffman, 2001.

[Praun99] E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. In *Proc. SIGGRAPH 1999 Conf. Computer Graphics*, pages 49–56. ACM, 1999.

[Rule97] K. Rule. *Crossroads 3D*, 1995–1997. http://home.europa.com/~keithr/Crossroads/.

[Schne96] B. Schneier. *Applied Cryptography (2nd Ed.).* Wiley, 1996.

[Wagne00] M. G. Wagner. Robust watermarking of polygonal meshes. In *Proc. Geometric Modeling & Processing 2000*, pages 201–208. IEEE Computer Society, April 2000.

[Yeo99] B-L. Yeo and M. M. Yeung. Watermarking 3D objects for verification. *IEEE Computer Graphics and Applications*, 19(1):36–45, 1999.

[Yeung98a] M. Yeung and B-L. Yeo. Fragile watermarking of three dimensional objects. In *Proc. 1998 Int'l Conf. Image Processing, ICIP98*, volume 2, pages 442–446. IEEE Computer Society, 1998.

[Yeung98b] M. M. Yeung(Ed.). Digital watermarking. *CACM*, 41(7):30–77, July 1998.