Steganography for Three-Dimensional Polygonal Meshes

N. Aspert, E. Drelie, Y. Maret and T. Ebrahimi

Signal Processing Institute
Swiss Federal Institute of Technology
1015 Lausanne
Switzerland

ABSTRACT

This paper proposes a method to embed information into a 3D model represented by a polygonal mesh. The approach used consists in slightly changing the position of the vertices, influencing the length of approximation of the normals to the surface. This technique exhibits relatively low complexity, and offers robustness to simple geometric tranformations. In addition, it does not introduce any visible distortion to the original model.

Keywords: steganography, watermarking, discrete surfaces, polygonal meshes

1. INTRODUCTION

Hiding information into images or sound files is a relatively common operation, called steganography. There are many applications of such techniques, ranging from content annotation to the secret transmission of critical data. A rather unexpected application has also been shown in relation to the *DeCSS* case (from the name of the algorithm used to decrypt DVDs), offering a way to bypass the DMCA (Digital Millenium Copyright Act, which made the distribution of the DeCSS source code in a compilable form illegal) by distributing the code hidden inside an image.[1] However, despite the rapid evolution of dedicated hardware and methods to display and process 3D models efficiently, only a few steganography/watermarking methods have been proposed so far. Among the existing methods, an adaptation of classic spread-spectrum watermarking techniques has been proposed[2] in 1999, which is very robust but involves a multi-resolution decomposition of the model, which can be a rather difficult operation in some cases. Another approach is based on a *spectral decomposition* of the model,[3] but such a method has an almost prohibitive complexity. In this paper, we propose a method that extends a previous approach[4] to embed information into a 3D model, offering a reduced complexity.

The paper is organized as follows : section 2 introduces the notations used through the paper, and presents the information embedding and extraction algorithms. Section 3 gives the details about implementation, complexity, and presents the results obtained in term of distortion introduced in the models by the embedding. Conclusions are drawn in section 4.

2. PROPOSED METHOD

2.1. Notations

In this section we define the notations that will used throughout the paper.

A *discrete surface* $\mathcal{M}$, may be defined by three sets: $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{P})$, where $\mathcal{S}$ is a set of indices $i = 1, \ldots, N$ (N is the number of vertices); $\mathcal{I}$ is a set of index pairs $\big\{(i,j)\big\}$ each describing the edge between the vertices of index i and j. Finally $\mathcal{P}$ consists in a set of points $p_i \in \mathbb{R}^3$, giving the coordinates of each vertex. For each vertex p, the vector from the origin of the coordinate system to this point will be denoted by $\boldsymbol{p}$. The usual euclidean norm of a vector $\boldsymbol{v}$ will be denoted by $\|\boldsymbol{v}\|$. The number of elements in a set $\mathcal{G}$ will be denoted by $|\mathcal{G}|$. Let $a \in \mathbb{R}$; the term $\langle a \rangle$ will denote the integer that is closest to a. Let us also denote the *1-ring* of p_i by $\mathcal{E}_i$, which is the set of all vertices sharing an edge with p_i, i.e. :

$$\mathcal{E}_i = \big\{ j \mid (i,j) \in \mathcal{S} \big\}.$$

Further author information: (Send correspondence to N. Aspert)
E-mail: Nicolas.Aspert@epfl.ch, Telephone: +41 21 693 3632, Fax: +41 21 693 7600

2.2. Information Embedding

In this section, we present the details of the embedding algorithm. The proposed method is based on an algorithm proposed by M. Wagner.[4] In the original paper, the only information embedded into a discrete surface was a function defined on the unit sphere. We have extended and adapted this technique to embed more useful information (i.e. a string of characters).

2.2.1. Information embedding in the normal vectors

In image watermarking, one of the most common techniques is to embed the information by modifying the value of the pixels or their transform. To watermark 3D models represented by a mesh, Wagner proposed to modify slightly the positions of the vertices composing the model, embedding the information in the length of the *normal vectors*, defined for each vertex. Those vectors are not really normal vectors in the classical meaning of differential geometry. Let us consider a model $\mathcal{M}$. The normal vector $\boldsymbol{n}_i$ associated to vertex p_i is defined by :

$$\boldsymbol{n}_i = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} (\boldsymbol{p}_j - \boldsymbol{p}_i). \tag{1}$$

Although the vector $\boldsymbol{n}_i$ can be considered as an approximation of a normal on the discrete surface of the model at p_i, equation (1) does not guarantee this vector to have a unit length. The core of the information embedding process relies on this fact, and the information is hidden into the relative length of these vectors in order to obtain the invariance versus translation, scaling and rotation. The term *relative length* makes sense when defining the *average normal vector length* d as follows :

$$d = \frac{1}{N} \sum_{i \in \mathcal{S}} \|\boldsymbol{n}_i\| .$$

Then, we can associate an integer n_i to each normal using the following relation :

$$n_i = \left\langle \frac{c}{d} \|\boldsymbol{n}_i\| \right\rangle , \tag{2}$$

where c is a user-defined parameter. An interesting property of the n_i is that they are invariant with respect to translation, rotation or isotropic scaling.

In the original approach, the information to be hidden is defined on a sphere and maps each unit vector $\boldsymbol{v}$ to a real number :

$$w = f(\boldsymbol{v}),\ 0 \leq w \leq 1,$$

where f is a function defined on the unit sphere. In order to embed a textual information into the model, we have chosen to use a piecewise-constant function f. The details concerning the choice of f are given in section 2.2.2. The embedding procedure starts by associating an integer w_i to each normal in the following way :

$$w_i = \left\langle 2^b f\left(\frac{\boldsymbol{n}_i}{\|\boldsymbol{n}_i\|}\right) \right\rangle , \tag{3}$$

where b denotes the number of bits used to represent w_i. The next step of the encoding procedure, is to replace b bits of n_i by the b bits of w_i, generating a new integer $\tilde{n}_i$. It is then possible to define some modified normal vectors $\tilde{\boldsymbol{n}}_i$ using the following relation :

$$\tilde{\boldsymbol{n}}_i = \frac{\tilde{n}_i d}{c} \frac{\boldsymbol{n}_i}{\|\boldsymbol{n}_i\|},$$

and finally the new point coordinates $\tilde{p}_i$ are found by solving the system of N linear equations defined by :

$$\tilde{\boldsymbol{n}}_i = \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} (\tilde{\boldsymbol{p}}_j - \tilde{\boldsymbol{p}}_i). \tag{4}$$

Generally, this set of equations has a rank lower than N. Solving the system gives the new points $\tilde{p}_j$. Let us denote by $\tilde{\mathcal{P}}$ the set formed by these new vertices. A new discrete surface $\tilde{\mathcal{M}}$ is defined by $\tilde{\mathcal{M}} = (\mathcal{S}, \mathcal{I}, \tilde{\mathcal{P}})$. Note

that since the normal vectors are modified, a parameter $\tilde{c}$, computed at embedding time has to be passed to the decoder (cf. section 2.3).

$$\tilde{c} = c\frac{\tilde{d}}{d},$$

where $\tilde{d}$ is the new average normal vector length.

2.2.2. Piecewise-constant functions on the sphere

In the original method, the information to embed is represented by a function on the unit sphere. While this is conceptually interesting, such an information has almost no practical use. One would prefer to be able to embed directly a textual message M into the cover-model. Therefore, we have to define a function on the sphere that is unambiguously linked to the message M. The message to embed, having N_a symbols can be written as $M = S_1 S_2 \dots S_{N_a}$, where S_i is a set of atoms having values inside an alphabet $\mathcal{A}$. Let us denote by $\hat{\boldsymbol{n}}_i$ the unit normal, associated to the vertex p_i, i.e. : $\hat{\boldsymbol{n}}_i = \boldsymbol{n}_i / \|\boldsymbol{n}_i\|$. We will denote by (φ_i, z_i) the spherical coordinates of $\hat{\boldsymbol{n}}_i$. The unit sphere is split into quadrants $\mathcal{Q}_{k,l}$, with $(k,l) \in \{0, \dots, N_m - 1\}^2$, where N_m^2 is the total number of these quadrants. A normal $\hat{\boldsymbol{n}}_i$ belongs to $\mathcal{Q}_{k,l}$ if its coordinates (φ_i, z_i) satisfy :

$$\begin{array}{rcccll} k\Delta\varphi & \leq & \varphi_i & < & (k+1)\Delta\varphi & \text{with} \quad \Delta z = 2/N_m \\ l\Delta z & \leq & z_i & < & (l+1)\Delta z & \text{with} \quad \Delta\varphi = 2\pi/N_m. \end{array}$$

Therefore, a piecewise constant function f on the sphere can be defined by :

$$f(\hat{\boldsymbol{n}}_i) = f_{k,l} \in \mathbb{R}$$

for all $\hat{\boldsymbol{n}}_i$ belonging to $\mathcal{Q}_{k,l}$. A simple way to embed a textual message using those piecewise is to use the ASCII code of each character of the message as value of $f_{k,l}$. If the distribution of the normals on the unit sphere is uniform, it is be sufficient to have $N_m = \lceil \sqrt{N_a} \rceil$. However, it is not possible, without having a priori knowledge on the model, to determine the repartition of the unit normals on the sphere. Therefore, all the pieces cannot be treated identically, as shown in the next section.

2.2.3. Normal acceptability criteria

Let us now introduce the concept of acceptability of a normal. A normal vector is called *acceptable* if an information can be embedded into the normal. On the contrary, an *unacceptable* normal will have no information embedded into its relative length. If a normal $\tilde{\boldsymbol{n}}_l$ is marked as *unacceptable*, the corresponding equation in (4) is replaced by a new equation that considers the associated vertex as fixed, i.e. :

$$\tilde{p}_l = p_l. \tag{5}$$

The acceptance criteria are based on the normal length and on the redundancy of the information. In order to obtain a watermarked model very similar to the original, some normals are marked as unacceptable in order to leave the associated vertices unchanged. This implies that some acceptable normals could be marked as unacceptable to reach a fixed ratio of unchanged vertices versus N. This is also useful to make the system of equation (4) more easy to solve (since it usually has a rank smaller than N). The normal length acceptance criterion takes into account that we cannot know *a priori* the number of bits needed to code the integers n_i. Thus, we suppose that there exists $c \in \mathbb{N}$, such that each n_i can be coded entirely by m bits. In order to obtain the integers $\tilde{n}_i$, we have to replace b contiguous bits of n_i by the b bits of w_i. This operation is illustrated by figure 1. Let be $d < (m - b)$, then we have:

$$\tilde{n}_i = \left\lfloor \frac{n_i}{2^{d+b}} \right\rfloor + w_i 2^d + \mathrm{mod}_{2^d}(n_i), \tag{6}$$

where $\mathrm{mod}_b(a)$ stands for a modulo b, and $\lfloor a/b \rfloor$ the euclidean division of a by b. If $\lfloor n_i / 2^{d+b} \rfloor = 0$ in equation (6), the modification on n_i would be too important, since this condition means that bits $d+b$ to $m-1$ are zero. In such a case, the information would be embedded in the most significant bits of n_i. In order to overcome this

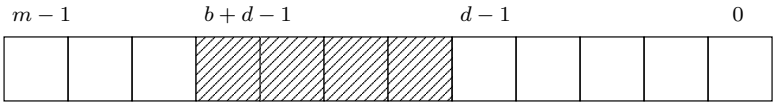


Figure 1. Position of the bits of w_i in n_i. The modified bits are shown by gray squares

effect, we decide that the normal $\boldsymbol{n}_i$ is *unacceptable* if $n_i < 2^{b+d} - 1$. Using this criterion adds noise to the embedded information but introduces a smaller amount of distortions in the model $\tilde{\mathcal{M}}$.

In order to be able to extract the information properly, it is necessary that a minimum number of normals lies in each piece of the sphere $\mathcal{Q}_{k,l}$, as defined in section 2.2.2. Let $n_{k,l}$ the number of normals belonging to $\mathcal{Q}_{k,l}$. This piece will be acceptable if :

$$n_{k,l} \geq r, \tag{7}$$

where r is the minimum redundancy should be achieved. According to equation (7), a second acceptance criterion is defined as follows : if $\mathcal{Q}_{k,l}$ is acceptable, a certain information will be attributed to this piece and the normals that are associated to it will be considered acceptable. Finally, the number of constant pieces ${N_m}^2$ has to be defined. It has to be bigger than the number of atoms N_a containing the message. Since the number of pieces that will be considered acceptable is not known a priori, N_m is defined as :

$$N_m = \left\lceil \sqrt{\alpha N_a} \right\rceil, \tag{8}$$

with $\alpha > 1$. Typically, $\alpha = 1.5$ is a sound value, which means that there will be 50% more pieces than atoms. The $\mathcal{Q}_{k,l}$ that are not used will be treated as unacceptable, and all related vertices will be treated accordingly.

2.2.4. Resistance to rotation

In the original method, an adjustment in the normal computation is needed, in order to make it robust with respect to affine transformation. While this approach is sufficient for the extraction of a simple function defined to a sphere, it would fail for the piecewise function we have proposed. While a translation or an isotropic scaling may not cause any trouble, a rotation performed on the model would shuffle the pieces on the sphere, making the message difficult to retrieve. As a consequence, a technique that solves this problem is proposed.

First, the model $\mathcal{M}$ is translated in order to make its "center of mass" coincide with the origin. We denote the translated model $\mathcal{M}_0$. Secondly, we compute a *privileged direction*, denoted by $\hat{\boldsymbol{d}}$:

$$\hat{\boldsymbol{d}} = \frac{\sum_{i \in \mathcal{S}} \hat{\boldsymbol{n}}_i}{\left\| \sum_{i \in \mathcal{S}} \hat{\boldsymbol{n}}_i \right\|},$$

which is roughly the direction where the normal density is the most important. Clearly, "real-world" models are unlikely to have a null $\hat{\boldsymbol{d}}$. In fact, such a case could mainly occur with analytic models (for instance a sphere). Finally, the model $\mathcal{M}_0$ is rotated in order to make $\hat{\boldsymbol{d}}$ coincide with a *cover direction*, denoted by $\boldsymbol{d}_{cover}$, which is set by the user, but must be known at extraction time. The resulting model, denoted by $\mathcal{M}_{cover}$ is the one on which the embedding is performed. The final marked model $\tilde{\mathcal{M}}$ is obtained by performing the inverse rotation and inverse translation on $\mathcal{M}_{cover}$.

2.3. Information Extraction

To extract the information hidden in $\tilde{\mathcal{M}}$, the following parameters are needed:

- $\tilde{c}$, the modified main parameter;
- N_a and α, the length of the embedded message and parameter needed to compute N_m;
- the position of bits of w_i in n_i, i.e. the parameters b and d;

- $\boldsymbol{d}_{cover}$, the cover direction.

Such informations represent a very small amount of data and therefore can easily be transmitted along with the marked model, for instance in a texture image, or in another attribute of the model.

Prior to any other operation, the transforms described in section 2.2.4 are applied to $\tilde{\mathcal{M}}$ in order to find the model $\tilde{\mathcal{M}}_{cover}$, aligned and positioned as $\mathcal{M}_{cover}$, in which the information was embedded. The integers $\tilde{w}_i$ are then extracted from the normals $\tilde{\boldsymbol{n}}_i$, by using the equations (2) and (6). For each piece of index (k,l) of the unit sphere, we have $\hat{n}_{k,l}$ samples, forming a set $\underline{\hat{w}}^{k,l} = \left\{\hat{w}_1^{k,l}, \ldots, \hat{w}_{n_{k,l}}^{k,l}\right\}$. The information extraction is based on the variance $s_{k,l}^2$ of this set :

$$s_{k,l}^2 = \frac{1}{n_{k,l}} \sum_{i=1}^{n_{k,l}} (\hat{w}_i^{k,l} - \bar{w}^{k,l})^2,$$

where $\bar{w}^{k,l}$ is the average value of the samples in $\underline{\hat{w}}^{k,l}$. The set $\underline{\hat{w}}^{k,l}$ represents an atom of the message if $n_{k,l}$ is big enough, since according to equation (7), the acceptance criterion is $n_{k,l} \geq r$. Moreover, the relevance of a piece of index (k,l) can be estimated by comparing its variance $s_{k,l}^2$ to a threshold, i.e: $s_{k,l}^2 \leq s_{min}^2$.

If the piece (k,l) is acceptable, the information embedded is given by:

$$w_{k,l} = \text{maj}\left(\underline{\hat{w}}^{k,l}\right),$$

i.e. by associating the value in $\underline{\hat{w}}^{k,l}$ having the highest probability to $f_{k,l}$. Finally, by taking the $f_{k,l}$ in the same order as they were inserted, extracting only the $w_{k,l}$ with a variance $s_{k,l}^2$ sufficiently low and by taking into account only the normals $n_{k,l}$ satisfying the condition of equation (7), the message $\hat{M}$, identical to the hidden message M, can be extracted (if no heavy modification has been done on the $\tilde{\mathcal{M}}$). In case where the model $\tilde{\mathcal{M}}$ has been rotated around the cover direction, the alignment with $\boldsymbol{d}_{cover}$ may not be sufficient. In order to avoid the effects of such a rotation the "proper" rotation angle γ around $\boldsymbol{d}_{cover}$ has to be estimated. This is done by rotating $\tilde{\mathcal{M}}_{cover}$ around $\boldsymbol{d}_{cover}$ with an angle $\gamma_i = i\Delta\gamma$ (using a step $\Delta\gamma$ small enough). For each i, the total variance ${}^i v_{tot}$ is computed using the following relation :

$${}^i v_{tot} = \sum_{k,l} \left({}^i s_{k,l}^2\right)$$

The best angle γ_i is the one minimizing the quantity ${}^i v_{tot}$. The final message extraction is performed on the model, rotated around the cover direction using this angle.

3. RESULTS

This section presents the results obtained with the algorithms detailed in section 2. When dealing with watermarking or steganography, several criteria are relevant to assess the performances of a method. In this paper, only on a subset of these criteria will be studied. The details concerning the implementation and complexity of the algorithms are presented in section 3.1 and 3.2 respectively. The distortion introduced by the information added to the model will be studied in section 3.3. The robustness with respect to basic attacks will be developed in section 3.4.

3.1. Implementation

The linear system from equation (4) can be rewritten as follows (taking into account the cases where normals are not acceptable, as stated in equation (5)) :

$$\boldsymbol{A}\boldsymbol{x}_{x,y,z} = \boldsymbol{b}_{x,y,z}.$$

$\boldsymbol{A}$ is a $N \times N$ square matrix, where N is the number of vertices in the model, $\boldsymbol{x}$ the unknown column vector formed by the coordinates of the new positions of the vertices, and $\boldsymbol{b}$ a known column vector, formed by the

initial positions of the vertices. In fact, the system has to be solved three times, once for each coordinates (x, y, z) of the vertices. In the proposed method, the bottleneck in terms of complexity is clearly the resolution of the system. In order to take into account the sparseness of $\boldsymbol{A}$, and also the fact that $\boldsymbol{A}$ may have a rank lower than N, the method proposed by T. Robey and D. Sulsky,[5] which performs a QR decomposition of the matrix, has been chosen. All the algorithms have been implemented using the C and C++ languages. The user-defined parameters (except the embedded message) have been kept constant for all our tests. The main parameter was set to $c = 2.10^7$, the cover direction to $(1, 1, 1)$. The w_i were represented by $b = 8$ bits, and embedded in the n_i (which are 32 bits integers) between the 16th and 24th bits (i.e. $d = 16$ in figure 1). The redundancy parameter r has been set to 10, and 20% of the total number of vertices were unchanged.

3.2. Complexity

Given the relative simplicity of the extraction operation, the complexity study will focus on the embedding operation. As stated in the previous subsection, the overall complexity of this operation is mostly due to the QR decomposition performed on $\boldsymbol{A}$. The reader is referred to the original paper[5] for a more detailed analysis of the complexity of the decomposition. We have chosen to evaluate the overall complexity of the embedding numerically, by varying the number of vertices of the cover models. Tables 2 and 3 present the embedding and extraction times for two test messages (cf. table 1). The models used to perform these tests are the *Venus* model, and the *BallJoint* model. The *Venus* model was originally made of 268 686 triangles and the *BallJoint* model had 274 120 triangles. Coarser versions of the models have been created using M. Garland's *QSLIM*[6] software. The execution times have been measured on a computer with a 1.8 GHz Intel Pentium 4 CPU.

	length (bytes)	message text
message 1	33	`a77e0b997bcd0dee22a8e8f4be1ce6ef`
message 2	75	"There is no vestige of a beginning, no prospect of an end" (Hutton, 1795)
message 3	33	`605b4d749aa9aea21d5d5a1d8fd98eb0`

Table 1. Test messages embedded. The message 1 is the MD5 sum of the file containing the original *Venus* model. Message 2 includes the quotation marks and the brackets. Message 3 is the MD5 sum of the *BallJoint* model.

		embedding time	extraction time
original (ca. 134k vertices)	message 1	519.83 s	9.82 s
	message 2	520.23 s	9.92 s
30k vertices	message 1	57.79 s	3.76 s
	message 2	64.86 s	3.84 s
15k vertices	message 1	9.63 s	2.23 s
	message 2	11.94 s	2.29 s
5k vertices	message 1	2.64 s	1.13 s
	message 2	3.47 s	1.18 s
2.5k vertices	message 1	0.42 s	0.4 s
	message 2	0.51 s	0.44 s

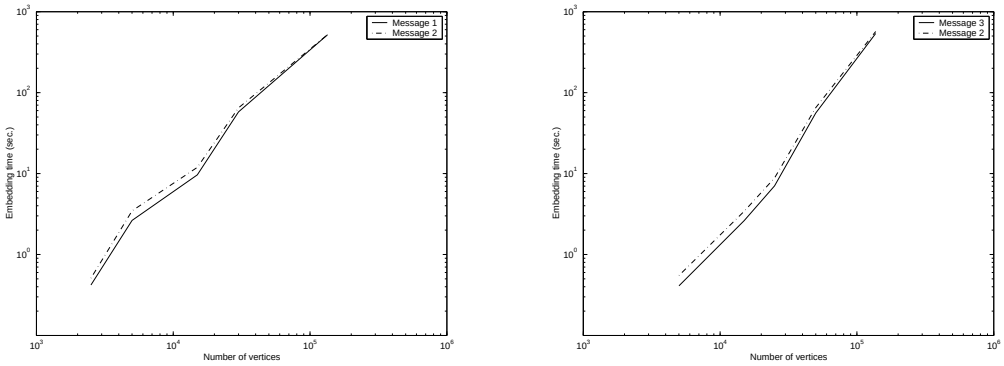
Table 2. Embedding and extraction times for the *Venus* model.

Figure 2 presents graphically the results of tables 2 and 3.

The estimated complexity is $\mathcal{O}(N^{1.7})$ for the *Venus* model, and $\mathcal{O}(N^{2.2})$ for *BallJoint*, which tends to prove that the algorithm has a complexity close to $\mathcal{O}(N^2)$.

		embedding time	extraction time
original (ca. 137k vertices)	message 3	537.7 s	10.18 s
	message 2	568.67 s	10.26 s
50k vertices	message 3	56 s	3.69 s
	message 2	65.17 s	3.77 s
25k vertices	message 3	7.04 s	1.86 s
	message 2	8.81 s	1.94 s
15k vertices	message 3	2.62 s	1.13 s
	message 2	3.42 s	1.19 s
5k vertices	message 3	0.41 s	0.4 s
	message 2	0.55 s	0.45 s

Table 3. Embedding and extraction times for the *BallJoint* model.



(a) (b)

Figure 2. Embedding time vs. number of vertices for the *Venus*(a) and *BallJoint*(b) models

3.3. Distortion

An important characteristic of a steganography or watermarking algorithm is the distortion that is added to the cover model. The ideal algorithm would maximize the information embedded into the cover model while minimizing the distortion induced. In the case of algorithms using images as cover information, the distortion can be measured using the RMSE or an equivalent pixel-based method, or using more sophisticated perception-driven measurements (which are often used as a criterion to embed the stego-information). In the case of discrete triangulated surfaces, a distortion measurement similar to the RMSE can be constructed using an approximation of the symmetrical Hausdorff distance. All measurements have been performed using the *MESH*[7] software, which is publicly available.

Figure 3 shows the evolution of the distortion induced by the embedding of the message versus the length of the message. There is no trivial influence of the message length on the distortion in the marked model. However, figure 3 clearly shows that RMSE remains small (ca. 0.002% of the length of the bounding box diagonal). Such a distortion is close to that induced by compression/decompression, using for instance a Touma-Gotsman[8] encoder, i.e. visually lossless.

3.4. Robustness

While the message can be extracted after basic geometric transformations such as translation, isotropic scaling, it does not resist more invasive techniques such as subdivision, simplification, compression. This relative sensitivity is due to the embedding technique, which slightly modifies the length of the normals by moving the

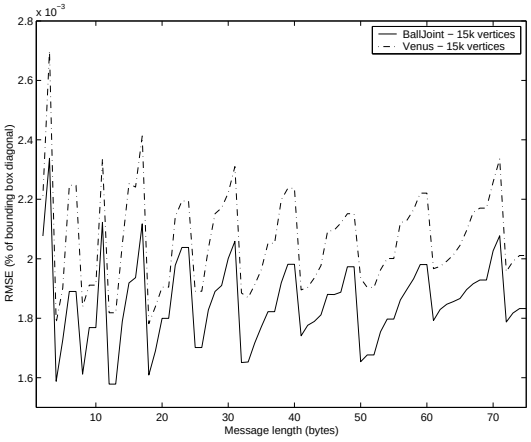


Figure 3. Distortion (RMSE in percent of the length of the bounding box diagonal) vs. the length of the embedded message

surrounding vertices. Section 3.3 showed that these displacements had a very small amplitude. Thus, performing simplification, compression or any other technique that disturbs these displacements with an amplitude greater than that introduced by the embedding algorithm will virtually "erase" the stego-information. In order to quantify more precisely this sensitivity, a *radial* perturbation has been added to the marked model. The *center of mass* of the model is defined by :

$$\tilde{\boldsymbol{p}} = \frac{1}{N} \sum_{i=0}^{N-1} \boldsymbol{p}_i.$$

A perturbation ε_i is added to each vertex p_i as follows :

$$\hat{\boldsymbol{p}}_i = \boldsymbol{p}_i \left(1 + \frac{\varepsilon_i}{\|\boldsymbol{p}_i - \tilde{\boldsymbol{p}}\|} \right),$$

where ε_i is a Gaussian noise centered on 0 of variance σ^2. Experiments showed that the limit of the noise level above which the original message is not complete is roughly $\sigma = 10^{-5}$.

4. CONCLUSION

In this paper, an original method has been proposed to embed some information into 3D models represented by polygonal meshes. The proposed method uses small displacements of the vertices of the model to embed the information. The method is robust with respect to simple geometric transformations, such as isotropic scaling, translation and rotation. However, unlike some other watermarking techniques[2] , the method will not resist to stronger attacks, such as resampling or compression. Nevertheless, it provides an interesting tool to perform content annotation and offers potentials for authentication of 3D models.

ACKNOWLEDGMENTS

This research was partially supported by the EC-project Certimark. The *Venus* and *BallJoint* models used for testing are courtesy of Cyberware, Inc.

REFERENCES

1. D. S. Touretzky, "Gallery of CSS Descramblers," 2000. `http://www.cs.cmu.edu/~dst/DeCSS/Gallery`.
2. E. Praun, H. Hoppe, and A. Finkelstein, "Robust Mesh Watermarking," in *Computer Graphics (SIGGRAPH'99 Proceedings)*, pp. 69–76, 1999.

3. R. Ohbuchi, S. Takahasi, and T. Miyazawa, "Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain," in *Proceedings of the Graphics Interface 2001*, pp. 9–17, Morgan Kaufmann Publishers, June 2001.
4. M. Wagner, "Robust Watermarking of Polygonal Meshes," in *Proceedings of Geometric Modeling and Processing 2000*, pp. 201–208, 2000.
5. T. Robey and D. Sulsky, "Row Ordering for Sparse QR Decomposition," *SIAM Journal on Matrix Analysis and Applications* **15**, pp. 1208–1225, October 1994. `http://www.ahpcc.unm.edu/~trobey`.
6. M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," in *SIGGRAPH 97 Proceedings*, pp. 209–216, August 1997. `http://graphics.cs.uiuc.edu/~garland/software/qslim`.
7. N. Aspert, D. Santa-Cruz, and T. Ebrahimi, "MESH: Measuring Error between Surfaces using the Hausdorff distance," in *IEEE Intl. Conf. on Multimedia and Expo.*, 2002. Accepted for publication, `http://mesh.epfl.ch`.
8. C. Touma and C. Gotsman, "Triangle Mesh Compression," in *Proceedings of the 24th Conference on Graphics Interface*, W. Davis, K. Booth, and A. Fourier, eds., pp. 26–34, Morgan Kaufmann Publishers, June 1998.