Data Hiding on 3D Polygonal Meshes

Y. Maret*
Signal Processing Institute
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland

T. Ebrahimi
Signal Processing Institute
Swiss Federal Institute of Technology
1015 Lausanne, Switzerland

ABSTRACT

This paper presents a high-capacity method to embed information into the geometry of a 3D polygonal mesh. The method extends a previously reported work, to which several improvements have been brought. By construction, the new embedding algorithm is robust against rotation, scaling and translation attacks. Analysis of the proposed method on different meshes showed that its embedding capacity compares to that of previous high-capacity works. Furthermore, the new technique introduces only negligible amount of distortions to the cover mesh.

Categories and Subject Descriptors

E.m [**Data**]: Miscellaneous

General Terms

Algorithms, Performance

Keywords

Data Hiding, Polygonal Mesh, High-Capacity

1. INTRODUCTION

Hiding information into images or sound files is a relatively common operation. There are many applications of such techniques, ranging from content annotation to the secret transmission of critical data. Data hiding on digital media is believed to have been used in both legal and illegal circumstances. An interesting uses has 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 Millennium Copyright Act, which made the distribution of the DeCSS source code in a compilable form illegal) by distributing the code hidden inside an image [12]. Other applications have made use of cover data, ranging from physical media such as paper and wood, to digital content in form of speech, audio, images and video.

Three dimensional models are good candidates to serve as cover data. Despite the rapid evolution of dedicated hardware and methods to display and process 3D models efficiently, only a few data hiding methods have been proposed so far. Among the existing methods, an adaptation of the classic spread-spectrum watermarking technique has been proposed [9], which is robust but involves a multi-resolution decomposition of the model, which can be rather computationally expensive in some cases. Another approach is based on a spectral decomposition of the model [8], but such a method has a high complexity. In another work [3], the information is hidden by modifying the mesh geometry to create an unique path on the mesh. A more recent watermarking approache [7] uses principal component analysis to reorient the model. In another recently published work [5], the message is hidden within the model topology.

2. BASIC DEFINITIONS

This section gives some basic definitions used in this paper, such as the formal definition of the three-dimensional meshes.

Let us formally define the support information, i.e. the discrete surfaces. The 3D smooth surfaces is approximated by a polygonal mesh that will be represented by a graph with a coordinate associated to each vertices. To that end, let a discrete surface $\mathcal{M}$ be defined by three sets: $\mathcal{M} = (\mathbb{N}_n, \mathcal{C}, \mathcal{P})$, the set of positive integers used to label each of the n vertices, the edge set $\mathcal{C} = \{e\}$ composed of pairs $e = \{i, j\}$, $i \neq j \in \mathbb{N}_n$, indicating the existence of an edge between the vertices i and j, and finally, the point set $\mathcal{P} = (\mathbf{p}_i)_{i=1}^n$ associating a vector $\mathbf{p}_i \in \mathbb{R}^3$ to each vertex. The sets $\mathbb{N}_n$ and $\mathcal{C}$ define the model's topology (through a graph), while the set $\mathcal{P}$ describes its geometry. Let us further define the 1-ring $\mathcal{S}_i$ of a vertex i as

$$\mathcal{S}_i = \{j \mid e = \{i, j\} \in \mathcal{C}\}, \tag{1}$$

The set $\mathcal{S}_i$ contains all vertices directly connected to the vertex i. In the following, the topological part of the model $\mathcal{M}$ will only be used through its 1-rings $\mathcal{S}_i$.

3. PROPOSED METHOD

The details of the *embedding algorithm* are presented in this section. The method extends previous works [1, 13]. The first paper [13] proposed a method exhibiting several limitations among which the most important one, for data hiding, was its low capacity. In the second paper [1], the

*For more information about this paper, feel free to contact us at the following address: yannick.maret@epfl.ch

MM-SEC'04, September 20-21, 2004, Magdeburg, Germany.

method was extended to actually embed non-trivial information into the model, increasing the embedding capacity. The current paper proposes an extension that significantly increases the embedding capacity and reduces the extraction complexity. The increased capacity is achieved by adapting the embedding process to the sample distribution in the similarity-transform invariant space. The complexity reduction of message extraction is produced by making use of a similarity invariant space, while in [1] the transform was only partially invariant to rotation. In addition, this paper describes the message embedding and extraction method from a new perspective, which leads to a better theoretical understanding of its different aspects.

3.1 Message Embedding

This section provides a quick overview of the message embedding method. It consists of a three-stage algorithm. In the first stage, the model geometry is transformed into a similarity-invariant space resulting in a non-uniformly sampled function on the unit sphere. The second stage modifies the resulting samples in order to embed the message. The final stage computes the necessary modifications in the model geometry. The next subsections explain in details the above mentioned stages of the algorithm.

3.1.1 Similarity-Invariant Space

The embedding method should be resilient to non-malicious transformations of the model. In the case of 3D surfaces, such operations are mainly similarity transforms (rotation, scaling, translation or any combinations of the latter). Therefore, working in a space invariant to similarity transforms will permit an embedding method intrinsically resilient to such transforms.

This similarity-invariant space is built gradually. The discrete normals set $\mathcal{N}$ is obtained from the geometry set $\mathcal{P}$. The set $\mathcal{N}$ is invariant to any translation of the model. The set $\mathcal{N}$ is further modified to produce the relative normals set $\mathcal{R}$, which is in addition invariant to any scaling. Finally, $\mathcal{R}$ is placed in a *given orientation*, to obtain the similarity invariant set $\mathcal{I}$, whose elements are invariant to any translation, scaling and rotation.

Let us first define the discrete normals set $\mathcal{N}$ as

$$\mathcal{N} = (\mathbf{n}_i)_{i=1}^{n} \text{ with } \mathbf{n}_i = \frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} (\mathbf{p}_j - \mathbf{p}_i) \tag{2}$$

where $\mathcal{S}_i$ is the 1-ring of vertex i (Eq. 1), and $\mathbf{p}_i \in \mathcal{P}$. Although the $\mathbf{n}_i$ are called discrete normals, they are not normals in the geometrical sense of the term because they do not have unit length; moreover if $\mathbf{p}_i$ and all $\mathbf{p}_j \in \mathcal{S}_i$ lie on the same plane, then $\mathbf{n}_i$ will also lie on this plane, which contradicts basic properties of conventional normals. However they have the interesting property to be invariant to any translation of the model, that is, for all $\mathbf{t} \in \mathbb{R}^3$

$$\mathcal{P}' = \mathcal{P} + \mathbf{t} \Rightarrow \mathcal{N}' = \mathcal{N}$$

Moreover, it is possible to recover the geometry from this set, as it will be explained in the subsection 3.1.2.

Let us add scaling invariance by defining the relative normals set $\mathcal{R}$ as

$$\mathcal{R} = (\mathbf{r}_i)_{i=1}^{n} \text{ with } \mathbf{r}_i = \mathbf{n}_i / \bar{n} \tag{3}$$

where $\bar{n} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{n}_i$, and $\mathbf{n}_i \in \mathcal{N}$. As stated before, the relative normals are, additionally, invariant to any non-zero scaling of the model, that is, for all $\mathbf{t} \in \mathbb{R}^3$ and $a \in \mathbb{R}^\star$

$$\mathcal{P}' = a \cdot \mathcal{P} + \mathbf{t} \Rightarrow \mathcal{R}' = \mathcal{R}.$$

Moreover, a rotation around any arbitrary axis of the model is a linear operation on the relative normals, that is, for all rotation matrices $\mathbf{R}$, for all $\mathbf{t} \in \mathbb{R}^3$ and $a \in \mathbb{R}^\star$

$$\mathcal{P}' = a \cdot \mathbf{R} \cdot \mathcal{P} + \mathbf{t} \Rightarrow \mathcal{R}' = \mathbf{R} \cdot \mathcal{R}$$

This last property will allow for the definition of a new set, whose elements will be invariant to any similarity transform applied to the model. To that end, let us define two parameter vectors, $\hat{\mathbf{f}}_0$ and $\hat{\mathbf{f}}_1$, of unit-length, non-zero, and orthogonal to each other. These vectors will be used as parameters during the construction of the similarity invariant space. Two characteristic vectors, of unit-length and orthogonal to each other, are formed from the set $\mathcal{R}$. A new set $\mathcal{I}$ is then computed by rotating $\mathcal{R}$, such that its characteristic vectors are aligned with the parameter vectors.

The characteristic vectors, $\hat{\mathbf{c}}_0$ and $\hat{\mathbf{c}}_1$, are defined by

$$\hat{\mathbf{c}}_0 = \mathbf{c}_0 / \|\mathbf{c}_0\| \text{ where } \mathbf{c}_0 = \sum_{i \in \mathbb{N}_n} \mathbf{r}_i \tag{4}$$

$$\hat{\mathbf{c}}_1 = \mathbf{c}_1 / \|\mathbf{c}_1\| \text{ where } \mathbf{c}_1 = \sum_{i \in \mathbb{N}_n} \frac{\mathbf{r}_i - (\mathbf{r}_i \bullet \hat{\mathbf{c}}_0) \cdot \hat{\mathbf{c}}_0}{\|\mathbf{r}_i - (\mathbf{r}_i \bullet \hat{\mathbf{c}}_0) \cdot \hat{\mathbf{c}}_0\|} \tag{5}$$

where $\mathbf{r}_i \in \mathcal{R}$. Additionally, a general rotation of the model is a linear operation on them. That is, for all general rotation matrices $\mathbf{R}$, for all $\mathbf{c} \in \mathbb{R}^3$ and $a \in \mathbb{R}^\star$

$$\mathcal{P}' = a \cdot \mathbf{R} \cdot \mathcal{P} + \mathbf{c} \Rightarrow \hat{\mathbf{c}}_0' = \mathbf{R} \cdot \hat{\mathbf{c}}_0 \text{ and } \hat{\mathbf{c}}_1' = \mathbf{R} \cdot \hat{\mathbf{c}}_1.$$

Based on the above observations, a rotation matrix $\mathbf{R}_{\mathcal{R} \to \mathcal{I}}$ can be found such that when applied to the set $\mathcal{R}$ its characteristic vectors will be aligned to the parameter vectors $\hat{\mathbf{f}}_0$ and $\hat{\mathbf{f}}_1$. The rotation matrix, $\mathbf{R}_{\mathcal{R} \to \mathcal{I}}$, is found by first constructing two matrices, $\mathbf{F}$ and $\mathbf{C}$. They are given by[1]

$$\mathbf{F} = (\hat{\mathbf{f}}_0 \; \hat{\mathbf{f}}_1 \; \hat{\mathbf{f}}_0 \times \hat{\mathbf{f}}_1) \tag{6}$$

$$\mathbf{C} = (\hat{\mathbf{c}}_0 \; \hat{\mathbf{c}}_1 \; \hat{\mathbf{c}}_0 \times \hat{\mathbf{c}}_1) \tag{7}$$

The rotation matrix $\mathbf{R}_{\mathcal{R} \to \mathcal{I}}$ is then found by solving the linear system $\mathbf{F} = \mathbf{R}_{\mathcal{R} \to \mathcal{I}} \cdot \mathbf{C}$ according to the following equation

$$\mathbf{R}_{\mathcal{R} \to \mathcal{I}} = \mathbf{F} \cdot \mathbf{C}^{-1}. \tag{8}$$

The similarity-invariant space $\mathcal{I}$ is then simply given by rotating each element of the relative normals set $\mathcal{R}$, that is

$$\mathcal{I} = (\mathbf{i}_i)_{i=1}^{n} \text{ with } \mathbf{i}_i = \mathbf{R}_{\mathcal{R} \to \mathcal{I}} \cdot \mathbf{r}_i, \tag{9}$$

where $\mathbf{r}_i \in \mathcal{R}$. The above construction method is valid as long that both characteristic vectors $\mathbf{c}_0$ and $\mathbf{c}_1$ are non-zero, which can happen for highly symmetric meshes.

The set $\mathcal{I}$ can be seen as the (non-uniformly) sampled version of a function over an unit sphere, as shown by

$$\mathcal{I}^{\mathsf{S}} = \left(\mathbf{i}_i^{\mathsf{S}}\right)_{i=1}^{n} \text{ with } \mathbf{i}_i^{\mathsf{S}} = (\rho_i, \theta_i, \phi_i) \tag{10}$$

where each element $\mathbf{i}_i = (x_i, y_i, z_i)$ of the set $\mathcal{I}$ is converted into its spherical coordinate system equivalent, and where $\rho_i = \|\mathbf{i}_i\|$ denotes the radius, $\theta_i = \arccos(z_i/\rho_i)$ the angle to the z-axis, and $\phi_i = \arctan(y_i/x_i)$ the angle to the x-axis

[1] The vectors $\mathbf{c}_0 \times \mathbf{c}_1$ and $\mathbf{f}_0 \times \mathbf{f}_1$ are necessary to ensure that there are enough equations for the linear system solution to be unique.

(on the xy-plane), and with the conventions $\theta \in [0, 2\pi)$ and $\phi \in [0, \pi]$. The pairs (θ_i, ϕ_i) can be used as the coordinates of the samples on the unit sphere and the ρ_i as their values.

3.1.2 Inverse Transform

This section describes how to recover the geometric information from the similarity-invariant space. The inverse transform is the last step of the embedding algorithm and therefore applies to a modified version of set $\mathcal{I}$, denoted by $\tilde{\mathcal{I}}$. The modified geometry set $\tilde{\mathcal{P}}$ is computed from the set $\tilde{\mathcal{I}}$. To this end, the sequence of operations follows a reverse order to that in Sec. 3.1.1 (i.e. $\tilde{\mathcal{I}} \to \tilde{\mathcal{R}} \to \tilde{\mathcal{N}} \to \tilde{\mathcal{P}}$).

The sequence $\tilde{\mathcal{I}} \to \tilde{\mathcal{R}} \to \tilde{\mathcal{N}}$ is quite easy to obtain because the rotation matrix $\mathbf{R}_{\mathcal{R}\to\mathcal{I}}$ and the mean normal length $\bar{n}$ were already computed during the transform step. Moreover, since $\mathbf{R}_{\mathcal{R}\to\mathcal{I}}$ is a rotation matrix, its inverse is simply given by transposition, i.e. $\mathbf{R}^{-1}_{\mathcal{R}\to\mathcal{I}} = \mathbf{R}^{T}_{\mathcal{R}\to\mathcal{I}}$. The relative normals set $\tilde{\mathcal{R}}$ is simply obtained by applying $\mathbf{R}^{-1}_{\mathcal{R}\to\mathcal{I}}$ to all elements of $\tilde{\mathcal{I}}$. The discrete normals set $\tilde{\mathcal{N}}$ is computed by multiplying the elements of $\tilde{\mathcal{R}}$ by $\bar{n}$, that is

$$\tilde{\mathcal{N}} = (\tilde{\mathbf{n}}_i)_{i=1}^{n} \text{ with } \tilde{\mathbf{n}}_i = \bar{n} \cdot \mathbf{R}^{T}_{\mathcal{R}\to\mathcal{I}} \cdot \tilde{\mathbf{i}}_i, \tag{11}$$

where $\tilde{\mathbf{i}}_i \in \tilde{\mathcal{I}}$. Finding the geometry set $\tilde{\mathcal{P}}$ from the discrete normals set $\tilde{\mathcal{N}}$ is not possible because there are infinitely many geometry sets producing the same discrete normals. However, it is possible to compute a unique solution for $\tilde{\mathcal{P}}$ by constraining one, or more, of its elements to have certain *a priori* values. More precisely, given a non-empty subset $\mathcal{E} \subset \mathbb{N}_n$, define the following linear system

$$\begin{cases} \tilde{\mathbf{n}}_i = \frac{1}{|\mathcal{S}_i|} \sum_{j \in \mathcal{S}_i} (\tilde{\mathbf{p}}_j - \tilde{\mathbf{p}}_i) & \text{for } i \in \mathbb{N}_n \setminus \mathcal{E} \\ \tilde{\mathbf{p}}_i = \mathbf{p}_i & \text{for } i \in \mathcal{E} \end{cases} \tag{12}$$

where $\tilde{\mathbf{n}}_i \in \tilde{\mathcal{N}}$ and $\mathbf{p}_i \in \mathcal{P}$. This system is solved to find a unique solution $\tilde{\mathcal{P}} = (\tilde{\mathbf{n}}_i)_{i=1}^{n}$. The elements of the set $\mathcal{E}$ need to be carefully chosen, so as not to interfere with the embedding process. In fact, constraints on some points on the model may remove changes induced by the embedding process to their corresponding normals. Further information for the choice of the set $\mathcal{E}$ will be given in the next section.

Let us further analyze the linear system to solve. It can be written in three equations systems $\tilde{\mathbf{n}}_x = \mathbf{A} \cdot \tilde{\mathbf{p}}_x$, $\tilde{\mathbf{n}}_y = \mathbf{A} \cdot \tilde{\mathbf{p}}_y$ and $\tilde{\mathbf{n}}_z = \mathbf{A} \cdot \tilde{\mathbf{p}}_z$, where the matrix $\mathbf{A}$ has size $n \times n$, and is often sparse because the 1-ring of a vertex generally contains only a small number of elements. In our implementation, this sparsity is taken into account to resolve the linear system. The matrix $\mathbf{A}$ is decomposed, using an adapted QR-method [10], and the obtained decomposition is used to easily resolve the system to find the solutions $\tilde{\mathbf{p}}_x$, $\tilde{\mathbf{p}}_y$ and $\tilde{\mathbf{p}}_z$. The inverse transform is computationally more complex than the direct transform because there is no linear system to resolve in the latter.

3.1.3 Bitstream Embedding

This subsection details the information embedding process. To embed a message into a model, its similarity-invariant set $\mathcal{I}^{\mathsf{S}}$ is modified in order to produce a new set $\tilde{\mathcal{I}}^{\mathsf{S}}$ containing the embedded message. The embedded message is represented as a bitstream given by $\mathcal{B} = (w_i)_{i=1}^{N}$, where the w_i are N binary words of W bits each. An ordering on the similarity-invariant set $\mathcal{I}^{\mathsf{S}}$ is defined, to link each word w_i to a number of elements in the set $\mathcal{I}^{\mathsf{S}}$. The modification is performed by incorporating the word w_i in the binary representations of those linked elements.

The ordering of the elements of $\mathcal{I}^{\mathsf{S}}$ is defined as follows. To this end, a curve $\mathsf{c}(t)$ on the unit sphere going from the 'north' pole and ending at the 'south' pole of the sphere, circling around the z-axis, is defined. In the $\theta\phi$-plane, the curve is given by

$$\mathsf{c}(t) = \big(\theta(t), \phi(t)\big) = \left(t(\text{mod } 2\pi), \frac{t}{2\mathfrak{m}}\right),$$

where $t \in [0, 2\pi\mathfrak{m}]$ and $\mathfrak{m} \in \mathbb{N}$ is the number of revolutions around the z-axis. Elements $\mathbf{i}_i^{\mathsf{S}}$ of $\mathcal{I}^{\mathsf{S}}$ are projected on the curve $\mathsf{c}(t)$, and a corresponding t_i on $\mathsf{c}(t_i)$ is assigned. An example of such a curve is given in Fig. 1. The set $\mathcal{I}^{\mathsf{S}}$ is then scanned following an ordered list $(t_i)_{i=1}^{n}$.

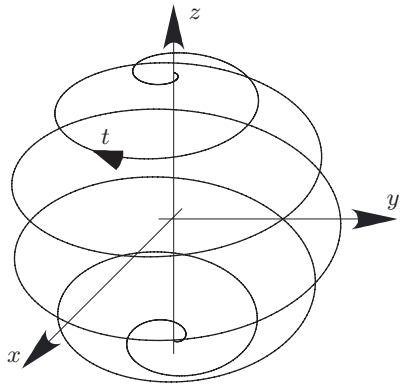

Figure 1: Example of a projection curve $\mathsf{c}(t)$ with $\mathfrak{m} = 6$.

More precisely, the ordering of the elements of the set $\mathcal{I}^{\mathsf{S}}$ is defined as a permutation vector Π, given by

$$\Pi = (\sigma_i)_{i=1}^{n} \tag{13}$$

with the following constraints

$$\begin{aligned} \sigma_i &\in \mathbb{N}_n, \sigma_i \neq \sigma_j, \forall i \neq j \\ \mathsf{t}(\theta_{\sigma_j}, \phi_{\sigma_j}) &\leq \mathsf{t}(\theta_{\sigma_{j+1}}, \phi_{\sigma_{j+1}}) \end{aligned}$$

for all $(\rho_i, \theta_i, \phi_i) \in \mathcal{I}^{\mathsf{S}}$. The scanning function $\mathsf{t}(\theta, \phi)$ is derived from $\mathsf{c}(t)$ and can be defined as

$$\mathsf{t}(\theta, \phi) = \left\lfloor \frac{\phi \cdot \mathfrak{m}}{\pi} + \frac{1}{2} - \frac{\theta}{2\pi} \right\rfloor 2\pi + \theta, \tag{14}$$

where $\mathfrak{m} \in \mathbb{N}$ is a parameter of the embedding algorithm, which defines the density of the scanning function $\mathsf{t}(\theta, \phi)$. An ordered sequence $\mathcal{O}$ of integers is defined as

$$\mathcal{O} = (\lfloor \rho_{\sigma_i} \cdot \mathfrak{r} \rfloor)_{i=1}^{n}, \tag{15}$$

where $\sigma_i \in \Pi$, and $(\rho_j, \theta_j, \phi_j) \in \mathcal{I}^{\mathsf{S}}$, and $\mathfrak{r} \in \mathbb{N}$ is a parameter of the embedding algorithm. The parameter $\mathfrak{r}$ is a scaling factor, which controls the precision of the binary representation of the radii.

The embedding algorithm requires a mapping function, which will put into correspondence each elements of w_i from $\mathcal{B}$ to elements from $\mathcal{O}$. To map $\mathcal{B}$ onto $\mathcal{I}^{\mathsf{S}}$, a redundancy parameter r is first computed. This parameters provides

the number of elements of $\mathcal{O}$ affected to the same w_i, and is given by

$$r = \left\lfloor \frac{n}{2N} \right\rfloor \tag{16}$$

where n is the number of vertices and N the number of words in the message. Let $\mathcal{W}_i$ denote the set of the indices of the elements from $\mathcal{O}$ that are put into correspondence with the word w_i

$$\mathcal{W}_i = \{j \mid j \in \mathbb{N}_n \text{ and } 2ir - 2r < j \leq 2ir - r\}. \tag{17}$$

The vertices in the model have not been all used for embedding. All remaining ones are included in the set $\mathcal{E}$ (cf. Eq. 12), that is

$$\mathcal{E} = \mathbb{N}_n \backslash \left(\bigcup_{i=1}^{N} \mathcal{W}_i \right). \tag{18}$$

If the number of vertices is a multiple of $2r$ then $|\bigcup_{i=1}^{N} \mathcal{W}_i| = |\mathcal{E}| = n/2$, and if not, then $|\mathcal{E}| = n - rN > n/2$.

The construction of the modified sequence $\tilde{\mathcal{O}}$ is straightforward. W contiguous bits of the binary representation of $\rho_j \in \mathcal{O}$ are replaced by the binary representation of w_i to obtain $\tilde{\rho}_j \in \tilde{\mathcal{O}}$. This replacement is performed is such a way that the least significant bit of the w_i replaces the $\mathfrak{b}^{th}$ least significant bit of ρ_j, $\mathfrak{b} \in \mathbb{N}$. The latter, $\mathfrak{b}$ defines another embedding algorithm parameter, which affects its strength. Finally, the modified similarity-invariant set $\tilde{\mathcal{I}}^{\mathsf{S}}$ is given by

$$\tilde{\mathcal{I}}^{\mathsf{S}} = \left(\tilde{\mathbf{i}}_i^S \right)_{i=1}^{n} \text{ with } \tilde{\mathbf{i}}_i^S = (\tilde{\rho}_{j(i)}, \theta_i, \phi_i), \tag{19}$$

where $\tilde{\rho}_{j(i)} \in \tilde{\mathcal{O}}$ and $j(i)$ is such that $\sigma_j = i, \sigma_j \in \Pi$.

The coding by repetitions, explained above, may seem inefficient as shown by Shannon in his original paper [11]. However, in this particular framework, we will show (in Sec. 3.2) that this approach has its own advantages.

Since the average normal length $\bar{n}$ (computed during the direct transform, cf. Sec. 3.1.1) might change during the embedding process, a rescaling parameter $\tilde{\mathfrak{r}}$ must be sent to the extraction algorithm, given by

$$\tilde{\mathfrak{r}} = \left\lfloor \mathfrak{r} \frac{\tilde{\bar{n}}}{\bar{n}} \right\rfloor, \tag{20}$$

where $\tilde{\bar{n}}$ is the average normal length computed after the inverse transform has been performed. In some applications, transmitting the parameter $\tilde{\mathfrak{r}}$ might not be desirable since it depends on the model. However, knowing the parameter $\mathfrak{r}$ is sufficient to recover the hidden message, at the cost of several decoding rounds. Indeed, $\tilde{\mathfrak{r}}$ is close to $\mathfrak{r}$ because the perturbations are usually small. The decoder can simply 'try' different values for $\tilde{\mathfrak{r}}$ into the interval $[\mathfrak{r} - 2^{W+\mathfrak{b}-1}, \mathfrak{r} + 2^{W+\mathfrak{b}-1}]$, an chose the extracted message bearing the less variance.

3.2 Message Extraction

In this section, the algorithm used to extract a message $\tilde{\mathcal{B}}$ from a model $\tilde{\mathcal{M}}$ is explained. To be able to extract the message from the model, the message extractor will need to know the message length N and the rescaling parameter $\tilde{\mathfrak{r}}$. In addition, the message extractor needs to use the same parameters used during the embedding process, namely, the number of bits W in each word, the position $\mathfrak{b}$ of the modified bits, the two parameter vectors $\mathbf{f}_0$ and $\mathbf{f}_1$, and finally the positive integer $\mathfrak{m}$ defining the density of the scanning function.

The first part of the message extraction follows exactly the operations during the embedding. The model $\mathcal{M}$ is first transformed into a similarity-invariant space $\tilde{\mathcal{I}}^{\mathsf{S}}$ as explained in subsection 3.1.1. The permutation vector $\tilde{\Pi}$ is then reconstructed, and used to produce the integer sequence $\tilde{\mathcal{O}}$ as defined in subsection 3.1.3.

The uncoded message $\mathcal{U}$ is extracted using a simple masking technique on $\tilde{\mathcal{O}}$, to produce

$$\mathcal{U} = (u_1, u_2, \cdots, u_n), \tag{21}$$

where n is the number of vertices in the model, and $u_i \in [0, 2^N - 1]$. Ideally, $\mathcal{U}$ would have the following structure

$$\mathcal{U} = (\underbrace{w_1, \ldots, w_1}_{r}, \underbrace{\xi_1, \ldots, \xi_r}_{r}, \underbrace{w_2, \ldots, w_2}_{r}, \underbrace{\xi_{r+1}, \ldots, \xi_{2r}}_{r}, \ldots, \underbrace{w_N, \ldots, w_N}_{r}, \underbrace{\xi_{(N-1)r+1}, \ldots, \xi_{Nr}}_{r}),$$

where the w_i are part of the message, and the ξ_i denote elements without message (i.e. the ξ_i are random variables without any correlation with the hidden message). Finally, the hidden message $\tilde{\mathcal{B}} = (w_1 w_2 \ldots w_N)$ is reconstructed making use of a simple histogram approach [1].

The extraction of the hidden message, as explain in the previous paragraph, needs to become more robust in real applications. In fact, several factors can corrupt the ideal $\mathcal{U}$. The choice of the constrained points, in the embedding algorithm, could affect the normal values $\tilde{\mathbf{n}}_i$. In addition, the model $\tilde{\mathcal{M}}$ can undergo several modifications between the embedding and extraction steps. Such factors would result in an undecoded message $\mathcal{U}$ which could be shifted or even locally corrupted. It is therefore important that embedding and extraction algorithms exhibit a certain degree of resiliency to such modifications. The choice of coding by repetition, rather than conventional error correcting codes, seems adequate for a first approach.

3.2.1 Synchronized Decoding

In this section, a more robust technique for message decoding is presented, when compared to histogram approach. The proposed technique takes advantage of the particular structure of an ideal undecoded message $\mathcal{U}$.

As shown before, the ideal undecoded message $\mathcal{U}$ alternates r equal symbols (the w_i) with r random ones (the ξ_i). The goal of this approach is to find N distinct subsequences in the undecoded message $\mathcal{U}$, such that the elements of each subsequence bear the largest number of equal elements. To extract the decoded message $\tilde{\mathcal{B}}$, the most frequent symbol in each subsequence is chosen.

In the following, the undecoded sequence $\mathcal{U}$ is converted in W binary sequences $\mathcal{U}^{(j)}$. Let the $u_i \in \mathcal{U}$ be defined by its binary representation, that is $u_i = \sum_{j=0}^{W-1} 2^j u_i^{(j)}$. Furthermore, let $\mathcal{U}^{(j)}$ denotes the sequence of the j^{th} bit of each element in $\mathcal{U}$, that is

$$\mathcal{U}^{(j)} = \left(u_1^{(j)}, u_2^{(j)}, \ldots, u_n^{(j)} \right). \tag{22}$$

To determine the N subsequences bearing the highest number of equal bits, local average sequences $\mathcal{A}^{(j)}$ for each

of the $\mathcal{U}^{(j)}$ is obtained according to the following

$$\mathcal{A}^{(j)} = \left(a_i^{(j)}\right)_{i=1}^{n} \text{ with } a_i^{(j)} = \frac{1}{r}\sum_{j=i}^{i+r}\left(u_i^{(j)} - \frac{1}{2}\right) \qquad (23)$$

where $u_i^{(j)} = 0$ if $i > n$, and the $a_i \in [-0.5, 0.5]$.

First, let us consider the ideal case where the index i coincides with the first element w_k of a sequence of r equal symbols. It is easy to see that $a_i^{(j)} \in \{-0.5, 0.5\}$, and $a_i^{(j)} + 1/2 = w_k^{(j)}$. Second, let us consider the ideal case where the index i coincides with the first elements ξ_k of a sequence of r random symbols. Under the assumptions that the ξ_k are independent from each other and distributed according a symmetrical PDF, one can state that $\mathbf{E}\left[a_i^{(j)}\right] = 0$.

Based on the above ideal cases, the proposed decoding technique can be derived as follows. Let $(m_k)_{k=1}^{N}$ gives the first indices of subsequences bearing the highest number of equal bits. The m_k are computed recursively, starting with m_1

$$m_1 = \arg_i\left(\max_{1\le i\le \frac{r}{2}}\left|\sum_{j=0}^{W-1} a_i^{(j)}\right|\right), \qquad (24)$$

and then

$$m_k = \arg_i\left(\max_{m_{k-1}+r+\frac{r}{2}\le i\le m_{k-1}+3r-\frac{r}{2}}\left|\sum_{j=0}^{W-1} a_i^{(j)}\right|\right), \qquad (25)$$

for $k = 2, \ldots, N$. It could be that there are more than one local maxima found. In this case, the maximum corresponding to the index i that is closest to the expected index $i_e = m_{k-1} + 2r$ is retained.

The decoded message is finally given by

$$\tilde{\mathcal{B}} = (\tilde{w}_1\tilde{w}_2\ldots\tilde{w}_N) \text{ with } \tilde{w}_k = \sum_{j=0}^{W-1} 2^j \cdot \left\lfloor a_{m_k}^{(j)} + \frac{1}{2}\right\rfloor, \qquad (26)$$

where $\left\lfloor a_{m_k}^{(j)} + \frac{1}{2}\right\rfloor$ represent the most frequent binary values (at position j) in the subsequences.

4. RESULTS

This section presents the results obtained with the algorithm proposed in this paper. Section 4.1, will introduce the parameters and models used during the experiments. Section 4.2 will report the result of study for embedding capacity. Embedding and extracting computational complexities are discussed in Sec. 4.3. Section 4.4 will analyze the amount of distortion created by message embedding. Comments concerning the robustness of the proposed method are discussed in Sec. 4.5.

4.1 Choices of Parameters and 3D Models

Some parameters of the proposed algorithm will be kept constant throughout the experiments. These parameters and their values are listed below. The parameter vectors used during the construction of the similarity-invariant space (cf. subsection 3.1.1) are kept constant, i.e. $\hat{\mathbf{f}}_0 = (0, 0, 1)$ and $\hat{\mathbf{f}}_1 = (0, 1, 0)$. The other constant parameters are those used during the embedding (cf. subsection 3.1.3), namely the number of bits in the message words $W = 8$, the position of the least significant bit $\mathfrak{b} = 15$, and the integer $\mathfrak{r} = 2 \cdot 10^8$. The density scanning parameter was chosen to be $\mathfrak{m} = \left\lfloor\sqrt{N}\right\rfloor$, since it gave good results during the experiments. The only parameter that will vary is the message length N. The message extractor needs the following three entities: the model $\mathcal{M}$, the message length N and the rescaling parameter $\tilde{\mathfrak{r}}$.

To compare the results obtained to those in [1], the relevant parameters were chosen to be, $(1, 1, 1)$ for cover direction, $c = 2 \cdot 10^8$, each word was represented by $b = 8$ bits, and the position of the least significant bit was $d = 15$.

During the different experiments, four different models were used, namely, body, pieta, david and venus. In order to speed up the simulation time coarser versions of the models have been created using M. Garland's *QSLIM* [6] software. The characteristics of the coarser models are shown in Tab. 1. All experiments were conducted with a numerical precision of 7 digits on the vertices coordinates.

Table 1: Characteristics of the models used during the experiments.

Models	Number of vertices n	Number of triangles
body	711	1396
pieta	3476	6976
david	24085	47753
venus	50002	100000

4.2 Embedding Capacity

For data hiding applications, one of the most important criteria (together with distortion) is the embedding capacity. In this section, the embedding capacity limits of the proposed algorithm are measured.

Random messages $\mathcal{B}$ were embedded in a given model $\mathcal{M}$ to measure the embedding capacity. Extracted messages $\tilde{\mathcal{B}}$ were examined in terms of the ratio between the number of binary errors, at extraction, and the total number of bits in the message as a function of the message length N. The experiments were ran several times for each length (each time for a different random message). Finally, by taking the median and average values of the different ratios at a given length, one was able to give two estimates of the probability of error when embedding a message of a given length in a given model.

For the models body (pieta), the results are shown in Fig. 2(a) and (Fig. 2(b)). The experiments were run 25 times for each length. There was no extraction error for messages of lengths up to 60 (200) when using the proposed method, and up to 20 (75) when using the method in [1]. It should be noted that it is not possible to embed a message of length exceeding 35 (175) using the method [1] due to the fixed division of the unit-sphere used therein.

Based on these above results, one can state that the proposed method has an embedding capacity of about three times that of the previous algorithm [1].

Similar experiments were run using the proposed algorithm for larger models, i.e. david and venus. The experiments were run 25 times for each length. The results, together with those of body and pieta, are shown on Fig. 3. Simulation results showed that the embedding capacity of the proposed technique is about 0.5 bit/vertex for an error-free extraction. It was further observed that the error-free extraction capacity per vertex tends to reduce as the models became larger, a plausible explanation could be that the

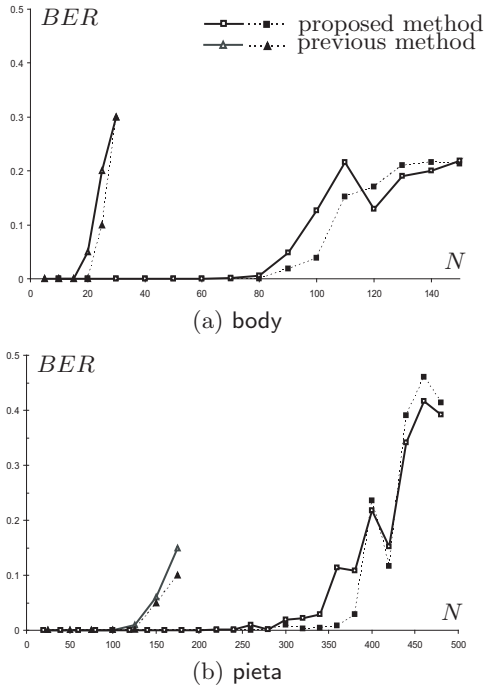

Figure 2: Embedding capacity of the proposed method (squares) compared with a previous work (triangles) on two models. A dashed curve denotes a median estimate, while a continuous one denotes an average estimate. The x-axis denotes the length N of the embedded message, and an estimation of probability of binary error is represented on the y-axis.

larger the model the higher probability of corruption of a decoded symbol, and the higher probability of skipping or adding symbols. This effect could be reduced be making use of error correcting codes inside embedded messages.

In conclusion, the proposed method significantly improves the embedding capacity when compared to the previous method [1]. The embedding capacity is also comparable to that of [5] and [3], where it was about 0.4 to 0.8 and 0.1 to 0.3 bit/vertex, respectively.

4.3 Complexity

In many applications computation time is an important factor. Most applications require an as fast as possible message extraction technique. Some others could have a further requirement on the speed of the embedding stage.

As seen in Sec. 3, there is a complexity asymmetry between message embedding and message extraction steps. In [1], it was experimentally found that the embedding complexity was around $O(n^2)$, where n is the number of vertices in the model.

To estimate the complexity, several messages of fixed length were embedded in a given model, and the computation time for embedding and extraction steps were measured. In Tab. 2 typical embedding and extraction times, both for the previous algorithm [1] and the proposed method, are reported. Because the proposed embedding method is computationally close to the previous technique, it is expected that they also compare in terms of embedding complexity. Concerning

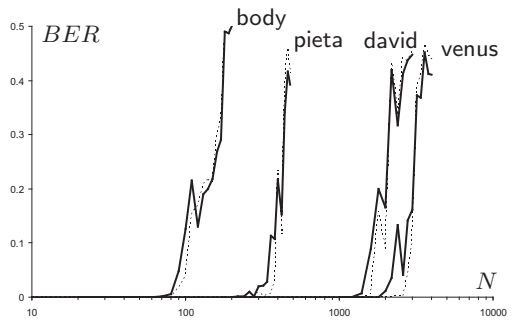

Figure 3: This graph compares the embedding capacity of the proposed method on different models. A dashed curve denotes a median estimate, while a continuous one denotes an average estimate. The x-axis denotes the length N of the embedded message, and an estimation of probability of binary error is represented on the y-axis.

the computational complexity of extraction, one would expect to achieve considerable improvements when compared to the previous algorithm. Indeed, in the previous method the message was embedded in a partially rotation-invariant space, and during extraction it was thus necessary to extract messages at different orientations around a particular axis and finally pick the one with the smallest variance. Based on these results, one can conclude that the proposed method, when compared to the previous one, brings a reduction in computational complexity up to 80% during extraction and does not influence significantly the embedding complexity.

Table 2: Typical embedding and extraction times for the proposed (previous) method. We embedded several random messages in different models. The performance were measured on a Pentium-M at 1.6GHz and with 512Mb of memory.

models	N	embedding time, s	extraction time, s
body	20	0.12 (0.13)	0.01 (0.02)
pieta	75	0.45 (0.4)	0.02 (0.04)
david	200	16.2 (15.3)	0.11 (0.43)
venus	500	38 (37)	0.4 (2.5)

4.4 Distortion

An important characteristic of a data hiding 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. 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* [2] software, which is publicly available.

Comparative results of the measured mean symmetrical Hausdorff distance, in percentage of the bounding box, using the proposed (previous) methods are shown in Tab. 3

for the models body, pieta, david and venus. It can be seen that the mean symmetrical Hausdorff distances between the embedded models and the originals are of the same order of magnitude for the proposed and previous methods. It is interesting to note that when using the proposed method, the mean symmetrical Hausdorff distance between embedded and original models are quasi constant even when embedding larger messages. This constant distortion behavior arises because the number of unchanged vertices is constant for any message lengths (cf. section 3.1.3).

Table 3: Distortions using the proposed (previous) method for different models and message lengths N, measured using mean symmetrical Hausdorff distances, and given in percentage of the bounding box.

body	$N = 1$	$N = 15$	$N = 30$
Distortions$\cdot 10^{-3}$	0.3 (0.2)	0.24 (0.3)	0.25 (0.6)
pieta	$N = 75$	$N = 150$	$N = 175$
Distortions$\cdot 10^{-3}$	0.08 (0.11)	0.08 (0.15)	0.07 (0.17)
david	$N = 100$	$N = 200$	$N = 500$
Distortions$\cdot 10^{-3}$	0.97 (1.1)	0.98 (1.2)	0.98 (1.3)
venus	$N = 100$	$N = 500$	$N = 1000$
Distortions$\cdot 10^{-3}$	0.21 (0.21)	0.22 (0.22)	0.22 (0.24)

4.5 Robustness

While the message can be extracted after basic geometric transformations such as translation, isotropic scaling, an rotations, it does not resist more invasive techniques such as surface subdivision, simplification and compression. This relative sensitivity is due to the fact that the embedding technique slightly modifies the length of the normals by moving its surrounding vertices. Section 4.4 showed that these displacements had a small amplitude. Thus, performing simplification, compression or any other techniques that disturb such displacements with an amplitude larger than that introduced by the embedding algorithm will virtually "erase" the stego-information.

5. CONCLUSIONS AND FUTURE WORK

This paper presented an efficient data hiding tool to embed messages in 3D polygonal meshes. Its embedding capacity was comparable with that of other high-capacity techniques [3, 5], and provided a three-fold improvement compared to its predecessor [1]. Few distortions were introduced to the cover mesh by the embedding, furthermore the amount of distortions was independent of the message length. The message extraction complexity was reduced by a factor four, which could be of interest in applications where a real time message extraction is required. The proposed technique is thus well suited for content annotation applications, for which high-capacity and low distortions are the key requirements. In addition, the method could be used for secret communication applications.

The imperceptibility of the hidden messages is one of the key requirements in steganography. One direction of research is thus to prove, or at least assess, the hidden message imperceptibility. Another direction of work would be to test the embedding robustness by using the evaluation methodology proposed in [4].

6. ACKNOWLEDGMENTS

This research was partially founded by the Swiss National Science Foundation for the promotion of scientific research – "Multimedia Security", grant number 200021-1018411. The work presented was developed within VISNET, a European Network of Excellence (http://www.visnet-noe.org), funded under the European Commission IST FP6 programme. The model venus is courtesy of Cyberware, Inc. The models pieta and david are courtesy of the "Digital Michelangelo Project". The authors would like to acknowledge F. Vermont for implementation of parts of the presented algorithm, as well as N. Aspert, M. Corsini and U. Hoffmann for fruitful discussions and insights.

7. REFERENCES

[1] N. Aspert, E. Drelie, Y. Maret, and T. Ebrahimi. Steganography for Three-Dimensional Polygonal Meshes. In *SPIE 47th Annual Meeting*, pages 705–708, Seattle, July 2002.

[2] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. MESH: Measuring Error between Surfaces using the Hausdorff distance. In *IEEE International Conference on Multimedia and Expo 2002*, pages 705–708, 2002.

[3] O. Benedens. Two high capacity methods for embedding public watermarks into 3d polygonal models. In *Proceedings of the Multimedia and Security-Workshop at ACM Multimedia*, pages 95 – 99, Orlando, Florida, 1999.

[4] O. Benedens, J. Dittmann, and F. A. P. Petitcolas. Three-dimensional Watermarking Design Evaluation. In *Security and Watermarking of Multimedia Contents V*, SPIE Proceedings Series 5020, pages 337 – 348, 2003.

[5] F. Cayre and B. Macq. Data Hiding on 3D triangle meshes. In *IEEE Transactions on signal Processing*, volume 51, pages 939–949, 2003.

[6] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH 97 Proceedings*, pages 209–216, August 1997. `http://graphics.cs.uiuc.edu/~garland/software/qslim`.

[7] A. Kalivas, A. Tefas and I. Pitas. Watermarking of 3D Models using Principal Component Analysis. In *ICME*, 2003.

[8] R. Ohbuchi, A. Mukaiyama, and S. Takahashi. A Frequency-Domain Approach to Watermarking 3D Shapes. In *Computer Graphics Forum*, volume 21, 2002.

[9] E. Praun, H. Hoppe, and A. Finkelstein. Robust Mesh Watermarking. In *Computer Graphics (SIGGRAPH'99 Proceedings)*, pages 69–76, 1999.

[10] T. Robey and D. Sulsky. Row Ordering for Sparse QR Decomposition. *SIAM Journal on Matrix Analysis and Applications*, 15(4):1208–1225, October 1994. `http://www.ahpcc.unm.edu/~trobey`.

[11] C. E. Shannon. A mathematical theory of communication. In *Bell System Technical Journal*, volume 27, pages 379–423, July and October 1948.

[12] D. S. Touretzky. Gallery of CSS Descramblers, 2000. `http://www.cs.cmu.edu/~dst/DeCSS/Gallery`.

[13] M. Wagner. Robust Watermarking of Polygonal Meshes. In *Proceedings of Geometric Modeling and Processing 2000*, pages 201–208, 2000.