Joint synchronization and high capacity data hiding for 3D meshes

Vincent Itier, William Puech, Gilles Gesquière, Jean-Pierre Pedeboy

Joint synchronization and high capacity data hiding for 3-D meshes

Vincent Itier,[a,c] William Puech,[a] Gilles Gesquière,[b] Jean-Pierre Pedeboy[c]

[a]LIRMM, UMR 5506 CNRS, University Montpellier, 860 rue de St Priest, Bat. 5, 34095 Montpellier, FRANCE

[b]LIRIS, UMR 5205 CNRS, University Lumière Lyon 2, Bat C 5 avenue Pierre Mendès,69676 Bron, FRANCE

[c]STRATEGIES S.A, Parc d'Affaires Icade, Immeuble Amsterdam, 56 Rue dArcueil, 94578 Rungis, FRANCE

ABSTRACT

Three-dimensional (3-D) meshes are already profusely used in lot of domains. In this paper, we propose a new high capacity data hiding scheme for vertex cloud. Our approach is based on very small displacements of vertices, that produce very low distortion of the mesh. Moreover this method can embed three bits per vertex relying only on the geometry of the mesh. As an application, we show how we embed a large binary logo for copyright purpose.

Keywords: 3-D data hiding, high capacity, geometry synchronization

1. INTRODUCTION

In the context of 3-D data exchange, it is necessary to enhance security, control and copyright aspects. Indeed, with technologies like internet, viewer or editor software, it is easy to share, transmit, manipulate, modify and print any 3-D mesh. Owners of a media would like to protect it against utilization, selling, claiming of properties or other attacks aiming its rights. There are two kinds of methods which are encryption and data hiding. Encryption is the most secure method since it is based on cryptography's principles introduced by Kerckhoffs.[14] Cryptography can be used for visual confidentiality, but when the media is decrypted there is no more security. Steganography and watermarking are methods for embedding hidden data. Steganography is "defined as the practice of undetectably communicating a message in a cover object",[8] thus it is not designed for a security purpose. Watermarking is used in order to provide ownership to ensure integrity or to add content on a media. These methods can be blind, if they do not need previous knowledge about the media, semi-blind or non-blind. Obviously blind methods offer better security. Furthermore, embedding data in a media is dependent of a compromise among capacity, invisibility and robustness. In this compromise one always required property is the invisibility. Indeed, the media should not look like it has been altered and designers do not want to see huge modification. A lot of watermarking method focuses on robustness like Wang *et al.*,[23] contrary to fragile watermarking which is used for integrity. Fragile watermarking has to detect the smallest modification of the media and if possible locates and characterizes it like in Yeo and Yeung.[24] More information on 3-D watermarking can be found in Rondao *et al.* and by Wang *et al.*[19,22] surveys. Numerous methods are proposed for very low capacity insertion.

The proposed method focuses on the capacity issue. Our approach relies on a stage that orders the vertices thanks to a Hamiltonian path.[12] This stage called synchronization, is crucial to perform the same path between the coding and the decoding stages. The embedding stage is done on the three dimension by slight vertex displacements, in order to move the vertex in an interval that corresponds to the data to be embedded. This method looks like the Quantization Index Modulation (QIM) method proposed by Chen and Wornell.[5] The capacity of the proposed method is near to 3 bits per vertex.

Further author information: (Send correspondence to V.I.)
V.I.: E-mail: vincent.itier@lirmm.fr

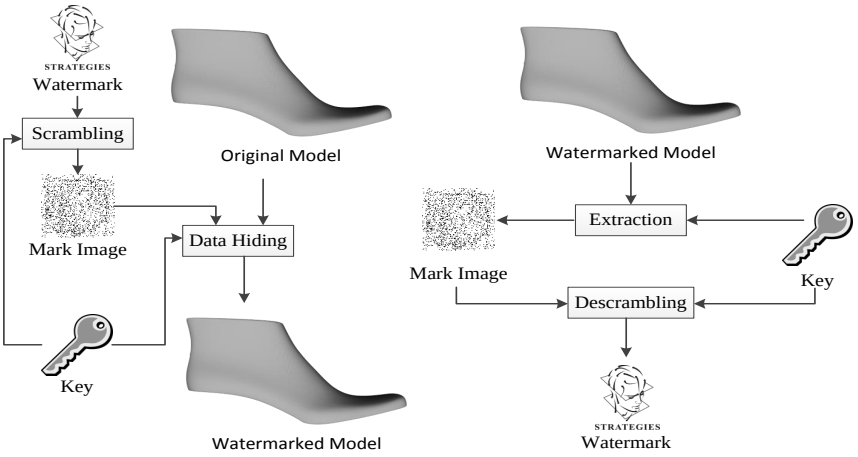


Figure 1: Scheme of logo embedding for copyright application.

In this paper, our goal is to propose a method with a high capacity data hiding. We first present a brief state of the art of 3-D data hiding in Section 2. In Section 3, we develop our method based on joint synchronization and data hiding. Then, we explain the extraction stage and we discuss about security aspects. Section 4 presents experimental results and some improvements. Finally, Section 5 concludes the paper and lists our future objectives.

2. RELATED WORKS

There are three ways to embed data in a mesh which are on topology, on geometry or on both. Early methods had inserted new edges and new vertices as in Ohbuchi *et al.*[18] which is visible and easily detectable. Moreover, the size of the mesh increases as a function of the message size. Some methods want to keep the vertex position unchanged so data are embedded in topology as in Amat *et al.*[1] The watermark is fragile and have a low capacity up to 0.14 $bits/vertex$. Contrary, some methods focus on geometry by modifying the vertex norm as in Cho *et al.*,[6] Luo and Bors[17] and in the method based on QIM proposed by Vasic and Vasic.[21] The first two methods have very low capacity $\simeq 64$ bits and the third a low capacity. In order to embed a message in a mesh, methods are based on a general scheme presented in Fig. 1. Here a logo is encrypted with a secret key, then it is inserted in the mesh by using a data hiding method. The security of the method is ensured by a secret key. Then, in order to authenticate a mesh an owner can verify the presence of his logo with the secret key.

Interest in high capacity data hiding has recently grown. In steganography for example, Chao *et al.*[4] proposed an approach where between 21 to 31 bits per vertex can be hidden. They order vertices using topology. Then, they embed bits by displacing each vertex on x, y and z into calculated regions and they can recursively divide regions to embed more bits. Their method produces nearly invisible distortion. But as the authors have mentioned, the vertex scheduling stage can hinder the method robustness. Another steganography method with very high capacity, 50 bits per feature vertex, has been proposed by Li *et al.*[16] The method adds lots of vertices which contain the payload, respecting the shape of the mesh. They also add vertices to make the mesh unsuspicious but its size increases. For robustness purpose, Gao *et al.*[9] proposed a semi-blind watermarking which is based on affine-invariants. Pieces of watermark are inserted by modifying the length ratio of each vertex. This is done by replacing the decimal notation in a given range. Their method can withstand majority of classical attacks, and the capacity depends on the range which is proportional to distortion. According to their parameters the capacity can be easily between 1 to 2 bits per vertex. Nevertheless, they have to assign an unique index to each piece of the message to resynchronize it. This synchronization issue prevents to embed more useful bits.

The synchronization part is crucial in 3-D application, because it is necessary to retrieve the same sequence between the coding and the decoding stages. Rossignac has proposed a widely used method for compression which orders the facets of the mesh.[20] In data hiding, modifications can occur, they can be malicious or not, so the synchronization has to be kept safe enough. Moreover, we think that an embedding method on a domain needs a synchronization method on the same domain.

3. PROPOSED DATA-HIDING METHOD

In this Section, we present a new high capacity data hiding scheme. First, in Section 3.1 we propose a new synchronization method based on the path constructed with the mesh vertices. Then, in Section 3.2, we present our data hiding scheme and we show how we use this synchronization for jointly embed a message. In Section 3.3, we analyze the proposed method and we present improvements for preserving the synchronization in Section 3.3.1 and for increasing the quality of the extracted message in Section 3.3.2. Finally, in Section 3.4, we show how the message can be extracted.

3.1 Synchronization stage

We introduce a new synchronization method that depends only on the mesh geometry, not on the relation given by the edges. Thus, a connectivity attack cannot cause the loss of the order defined on the vertices. The main idea is to build an ordered structure on the vertices like it is done by Amat *et al.*[1] They build an Euclidean Minimum Spanning Tree (EMST) to cover all the vertices. Their method has two major drawbacks, complexity and robustness, but it has some good properties: it gives an implicit order, it does not deteriorate the mesh, it does not depend on connectivity and it is unique. In this paper, we propose to build a Hamiltonian path as it has been proposed for the synchronization of vertex clusters in.[13] This method reduce the time complexity and a Hamiltonian path is more stable than an EMST. For a mesh of size n in number of vertices, the Hamiltonian path is built over the graph of the vertices with the aim to keep the interesting qualities pointed out. The graph is denoted $G_n = (V_n, E_m)$ and it is defined as the complete graph on all the vertices of the mesh. E_m represents the edges of the graph which are different to the topology of the mesh and $m = n(n-1)/2$. The path constructed at the step i is a sub-Hamiltonian path P_i on the set $Vi = \{v_0, \dots, v_i\}$, which is made up of the chosen edges $\{e_0, \dots e_{i-1}\}$. All the vertices are either in V_i, either in $V_n \backslash V_i = \{v_{i+1}, \dots, v_n\}$ the set of the non-visited vertex, these sets are illustrated Fig. 2.

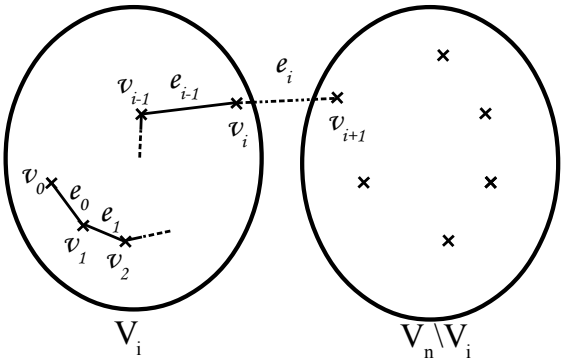


Figure 2: State of the sets at step i.

We use a secret key to choose the first vertex v_0 over the n vertices to build a Hamiltonian path which is a path that visits each vertex exactly once. In order to build it, we recursively choose the nearest neighbor v_{i+1} of the current vertex v_i, $i \in [0, n-1]$ by choosing the minimal edge $e_i \in E_m$ in euclidean distance between v_i and its non-visited neighbors. Obviously, after the last step, P_n is a Hamiltonian path on G_n.

3.2 Embedding Scheme

In this Section, we present how the edges of the Hamiltonian path built in previous step are used to embed the hidden data.

We propose to use the construction of the Hamiltonian path to obtain a relation between two vertices. This relation is given by the edges of the path. For each iteration i of the construction, data are embedded by slightly moving the vertex v_{i+1}, relative to its predecessor v_i, to its new position v'_{i+1}. The embedding process is joined with the proposed synchronization, in order to know the new vertex position at each iteration and keep the path unchanged. We propose to displace the vertex on ρ, θ and ϕ after converted its Cartesian coordinates to spherical ones. There, we explain the embedding process on ρ since the same process can be applied on θ and ϕ.

For defining the value of an edge e_i which corresponds to ρ_i, we partition it by a given step Δ. Each interval corresponds to a bit of the hidden message, which is alternately a 0 or a 1. We can extract the value b_i of the initial vertex v_{i+1} with:

$$b_i = \left\lceil \left(\frac{\rho_i}{\Delta} \right) \right\rceil \%2, \tag{1}$$

where $\rho_i = \|v_i, v_{i+1}\|_2$, is the euclidean distance between v_i and v_{i+1}. Fig. 3 illustrates the edge division and the value corresponding to the initial position of the vertex v_{i+1}.

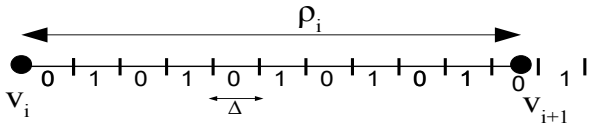


Figure 3: Dividing an edge for embedding on ρ_i.

Since we know the value of b_i we can compare it with the bit of the message m_i to be inserted. There are two cases, bits can be equal or different. If they are equal, we just compute the new ρ'_i value of the vertex v_{i+1} which corresponds to the center of the current interval and compute the new position v'_{i+1} of v_{i+1}. If the bits are different, we move the vertex v_{i+1} to the nearest adjacent interval center to have its new position v'_{i+1}. The Fig. 4 illustrates the two cases, in blue the equality in red the difference.

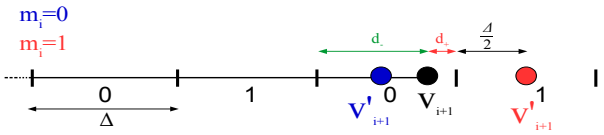


Figure 4: Vertex displacement on ρ_i.

In order to find the nearest adjacent interval center that minimize $\|v_{i+1}, v'_{i+1}\|_2$, we calculate d_+ and d_- which are the distances which separate the vertex v_i from the interval limits:

$$d_+ = \lceil (\frac{\rho_i}{\Delta}) \rceil \times \Delta - \rho_i, \tag{2}$$

$$d_- = \rho_i - \lfloor (\frac{\rho_i}{\Delta}) \rfloor \times \Delta. \tag{3}$$

Then we obtain d_m:

$$d_m = min(d_+, d_-) + \frac{\Delta}{2}. \tag{4}$$

The last stage is to make the inverse transform of v_{i+1} between spherical to Cartesian coordinates. Furthermore, by using the same strategy on θ and ϕ we can embed 3 *bits/vertex* and as we can repeat the embedding for $i \in [0, n-1]$, we can embed $3 \times (n-1)$ bits in a 3-D object of size n vertices. Nevertheless, an issue can occur when we move a vertex, indeed the displacement can change the synchronization at the extraction stage. For solving this problem, we propose in Section 3.3, to improve the embedding data jointly with synchronization process.

3.3 Displacement analysis

Fig. 5 illustrates the proposed embedding scheme, there are three major stages. These stages are jointly repeated for each vertex $v_i, i \in [0, n-1]$. Algorithm 1 describes the proposed watermarking method. Input are the initial vertex v_0 which is given by the secret key k, the message to be inserted M and the Hamiltonian path P on G_n presented in Section 3.1. We also declare an empty path P. The first stage consists in the searching of nearest neighbor which is done by the function $getNearestNeighbor()$ at line 4 in Algorithm 1. The second

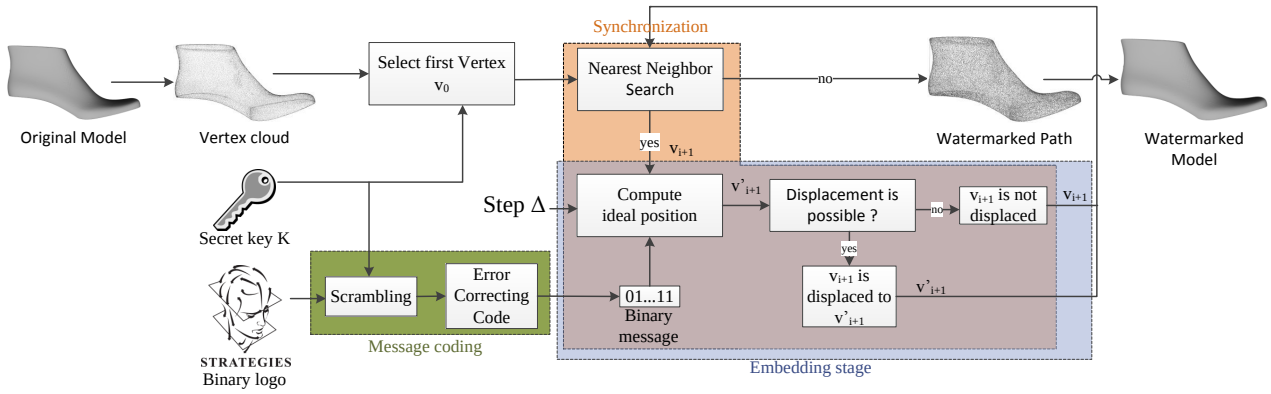


Figure 5: Joint synchronization and embedding.

Algorithm 1 Joint Synchronization and Watermarking

Require: $P = (E, V_{in}), V_{out}, k, M, G_n = (E_m, V_n)$
1: $V_{out} = V_n$
2: $v_1 \leftarrow v_k \in V_{out}$
3: **while** $V_{out} \neq \emptyset$ **do**
4: $v_2 \leftarrow getNearestNeighbor(v_1)$
5: **if** $check()$ **then**
6: $v_2 \leftarrow newPos(v_2, M)$
7: **end if**
8: $V_{in} \leftarrow \{v_1\}$
9: $V_{out} \leftarrow V_{out} \setminus \{v_1\}$
10: $E \leftarrow \{v_1, v_2\}$
11: $v_1 \leftarrow v_2$
12: **end while**

stage is the embedding process described in Section 3.2 and named $newPos()$ line 6. The last one is a checking stage to deal with desynchronization issues which is the function $check()$ at line 5.

3.3.1 Synchronization preservation

The checking stage is made to preserve the initial sub-Hamiltonian path P_{i+1}, when we move the current vertex v_{i+1}. Indeed, at the decoding stage we want to perform the same path along the vertices of the mesh. At the iteration i, a vertex is either in the path, or either in the set of vertex to be added to the path. A vertex $v_j, j \in [0, n]$ is in P_i if $j \in [0, i]$. Thus, there are two cases for a vertex v_j, $j \neq i + 1$:

(i). $j \in [0, i] \Leftrightarrow v_j, \in V_i$,

(ii). $j \in [i + 2, n] \Leftrightarrow v_j, \in V_n \backslash V_i$.

For the first case (i), the vertex v_{i+1} could be moved too close of the sub-path P_i and a vertex of the sub-path $v_j \in P_i, j \neq i$ could become its predecessor, as illustrated Fig. 6. The vertex v_{i+1} is moved in a new position v'_{i+1} near to v_j, so the distance $d_i = \|v_i, v'_{i+1}\|_2$ of e_i became greater than the distance $d_j = \|v_j, v'_{i+1}\|_2$ of e_j. As consequence at the decoding stage, the edge e_j between v_j and v'_{i+1} would be chosen and will change the Hamiltonian path.

For the second case (ii), the vertex v_{i+1} is moved to a new position v'_{i+1} which is too far from it predecessor and another vertex $v_j, \in V_n \backslash V_i$ becomes nearest to v_i. This case is illustrated Fig. 7 which shows that another vertex will be chosen at the decoding stage.

We propose here a strategy for the checking stage, which is to not move the vertex and potentially lose one bit of the message but the synchronization will be preserved. In order to compute if a vertex displacement is

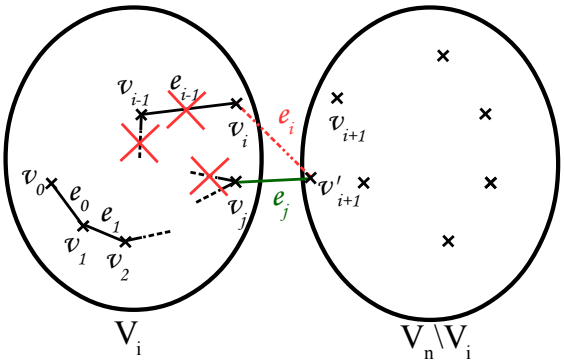


Figure 6: v_{i+1} is moved too close.

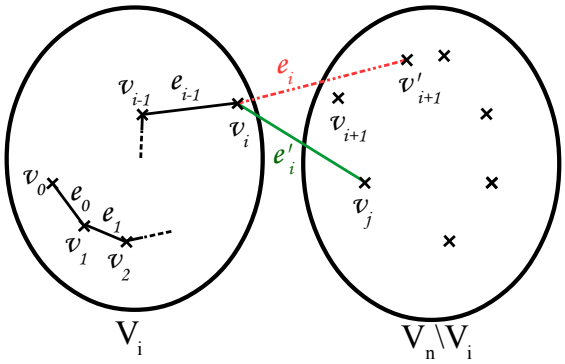


Figure 7: v_{i+1} is moved too far.

allowed, we define a radius r_i for each vertex v_i which corresponds to a sphere where the vertex can be safely moved. This radius is given by:

$$r_{i+1} = min(r^+_{i+1}, r^-_{i+1}), \tag{5}$$

where r^- and r^+ are the cases illustrated respectively in Fig. 6 and Fig. 7 and are given by:

$$r^-_{i+1} = min(\|v_j, v_{i+1}\|_2 - \|v_j, v_{j+1}\|_2),\ j \in V_i,\ j \neq i, \tag{6}$$

$$r^+_{i+1} = min(\|v_i, v_j\|_2) - \|v_i, v_{i+1}\|_2,\ j \in V_n \backslash V_i,\ j \neq i{+}1. \tag{7}$$

If a vertex displacement is out of the vertex safety sphere, we choose to disallow the displacement.

3.3.2 Error Correcting Code

In Section 3.3.1, we allow to lose a bit in order to keep the synchronization unchanged between the coding and the decoding stages. The loss of a bit of the message could produces a lost of information and a noise on the extracted logo. We propose to use an Error Correcting Code, (ECC) to correct the errors made during the embedding. We use the perfect $(23, 12, 7)$ Golay code,[10] which adds 11 corrector bits for 12 message bits, to produce the message to be embedded. After the extraction stage, we decode the message by using a fast method proposed by Chang *et al.*[3] that allow to correct up to 3 bits per message block. Obviously, the size of the "useful" message is reduced and its size s_m is given by:

$$s_m = \lfloor \frac{3(n-1)}{23} \rfloor \times 12. \tag{8}$$

Furthermore, using a pseudo-random key to scramble the message leads to a uniform error distribution, which improves the correction rate. Indeed, errors are generated by the checking stage that depends on the neighborhood of a vertex, since mesh can have different densities we can say that errors are not uniformly distributed in the watermarked model.

3.4 Extracting scheme

Fig. 8 illustrates the extracting scheme. Knowing the initial vertex, the chosen step Δ and the pseudo-random key used to encrypt the message, it is easy to extract the embedded message by constructing the Hamiltonian path and "read" the three bits in each vertex using the equation Eq.1 presented in Section 3.2. Then, the whole message is decrypted using the secret key.

Establish the security of the method only on the first vertex is not good enough to ensure a high security level. But we think that the encryption of the message is sufficient enough to ensure a good security level even if the attacker knows the step Δ, due to the complexity. Nevertheless, we think that evaluate more precisely the security of our method, according to the work done by Cayre *et al.*,[2] is a major objective.

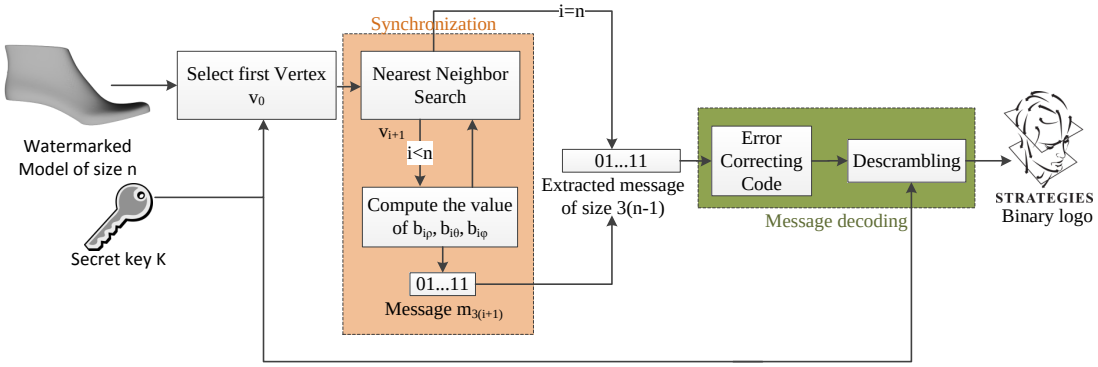


Figure 8: Extracting the embedded message.

4. EXPERIMENTAL RESULTS

In this Section, we present the results of embedding a logo in a mesh thanks to the proposed data-hiding method. In Section 4.1, we analyze the behavior of the watermarking process on a full example. In Section 4.2, we discuss about the proposed improvements. Finally, a full experimentation on many different objects, is proposed in Section 4.3.

4.1 Application example

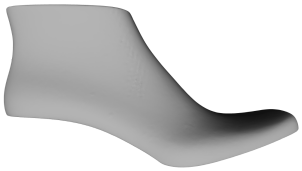


Figure 9: ShoeS mesh with 45002 vertices.

In this Section, we present a detailed example of the proposed method applied to an inside shape of a shoe. Fig. 9 presents the mesh ShoeS which is used to illustrate results. This object is a simplified version of an original one* and has 45002 vertices and 90000 facets. We use a simplified mesh because the algorithm has a complexity of $\mathcal{O}(n^3)$, due to the checking stage in the construction of the path. Furthermore, it is normalized in order to compare results obtained on different meshes, with the same parameters. The normalization is given by a scaling factor k which is a function of the size of the bounding box:

$$k = max\{x_{max} - x_{min}, y_{max} - y_{min}, z_{max} - z_{min}\}. \tag{9}$$

The capacity of the data-hiding scheme allows to embed a square binary logo of size:

$$s = \lfloor\sqrt{3(n-1)}\rfloor \times \lfloor\sqrt{3(n-1)}\rfloor. \tag{10}$$

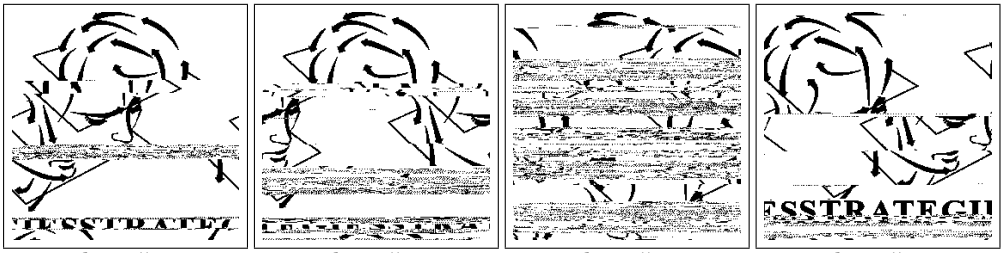
Fig. 10 presents the logo to hide, its size is given by Eq. 10:

In order to well understand the major aspect of the synchronization, Fig. 11 presents the results of the method without the checking stage. We use the proposed watermarking with a step of 10^{-6} for embedding the logo in the mesh and we extract it. It shows the importance of the synchronization and also the choice of the first vertex, in the case of desynchronization.

According to this result, we present other experiments using the checking stage to keep the synchronization safe. Fig. 12 presents the extracted binary logos. We analyze the quality of them by comparing extracted ones with the original.

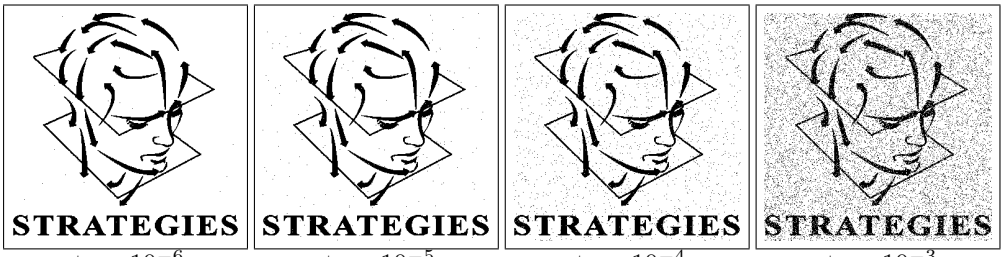
*STRATEGIES S.A. http://www.cadwin.com

Figure 10: Resized logo 367×367 pixels.



key #1 key #2 key #3 key #4

Figure 11: Extracted logo for different key vertices, step: 10^{-6}.



step: 10^{-6} step: 10^{-5} step: 10^{-4} step: 10^{-3}

Figure 12: Extracted logo.

Table 1 presents the Normalized Cross Correlation[11] (NCC) between the original logo and the extracted ones. The more it is close to 1 the more the logos are correlated, contrary more it is close to 0 the less it is. In this context, we choose to present results in terms of NCC because it is design for image and indicates if the image is still recognizable. Contrary, the Bit Error Rate (BER) is more interesting in signal processing to compare bits flows.

Table 1: NCC between the original logo and the extracted ones.

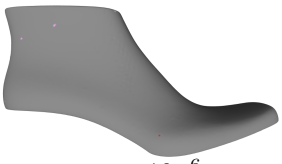
Step Δ	10^{-6}	10^{-5}	10^{-4}	10^{-3}
NCC	0.9997	0.9973	0.9755	0.8032

We can see on the Fig. 12 that the extracted messages are noisy, this is due to the checking stage that disallows some vertex moving. But it keeps the synchronization unchanged and the logo is still recognizable. Moreover, the Table 1 presents good results in terms of NCC for steps Δ lower than 10^{-3}. However, the logo is still recognizable for $\Delta = 10^{-3}$. The limit of the step depends on the mean distance between two vertices, noted md, in the Hamiltonian path of a mesh. In this example, the mesh presented in Fig. 9 has his $md = 4.40368 \times 10^{-3}$. Fig. 13 illustrates the difficulties to move a vertex when the step is too close to md.

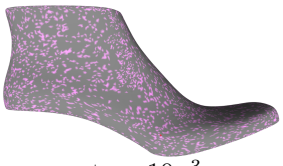
We can see that numerous vertices cannot be moved and that they are not distributed with the same probability along the mesh. The percentage of non-moved vertices for each step Δ is presented in the Table 2. We note that the number of vertices, which cannot be moved, is directly influenced by the value of Δ.

4.2 Improved method with ECC

To improve these results, we add the ECC presented in Section 3.3.2, to the embedded message to reduce the noise. The payload changes because a part of the placements available for embedding, is used to add redundant



step: 10^{-6}



step: 10^{-3}

Figure 13: Vertices that cannot be moved appear in pink.

Table 2: Percentage of non-moved vertices as a function of Δ.

Step Δ	10^{-6}	10^{-5}	10^{-4}	10^{-3}
Percentage of non-moved vertices	0.035	0.253	1.857	16.957

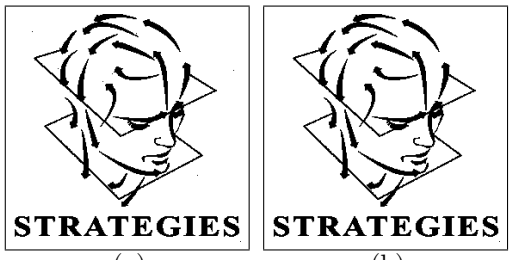
information. The number of message bits given by Eq. 8 is used in order to compute the size s of the square logo to embed:

$$s = \lfloor \sqrt{\lfloor \frac{3(n-1)}{23} \rfloor \times 12} \rfloor \times \lfloor \sqrt{\lfloor \frac{3(n-1)}{23} \rfloor \times 12} \rfloor. \tag{11}$$

Fig. 14 presents the resized logo with Eq. 11.

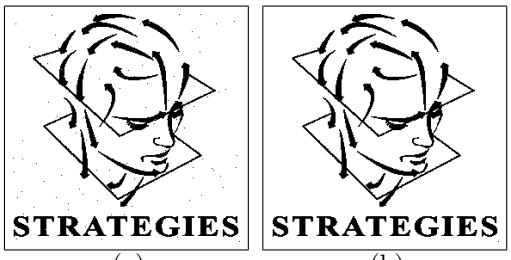
Figure 14: Resized logo 265×265 pixels.

Fig. 15 presents results of the extraction of the binary logo presented in Fig. 14. For each step Δ, we extract the logo and apply the ECC. For visualization comparison, we present two extracted logos for each step Δ. Logo denoted (a) is not corrected while the other denoted (b) corresponds to the decoded message with ECC.



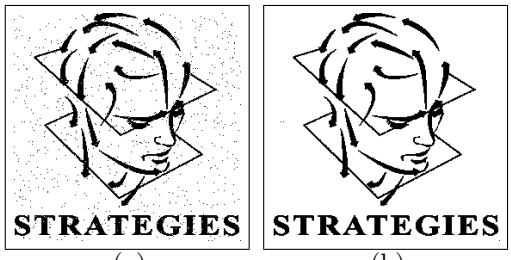
(a) (b)

step: 10^{-6}



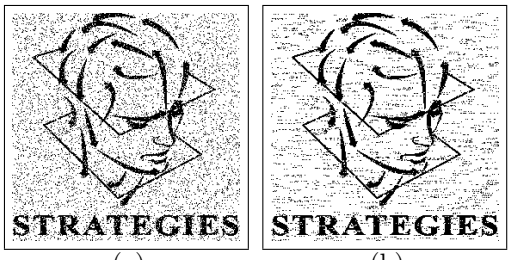
(a) (b)

step: 10^{-5}



(a) (b)

step: 10^{-4}



(a) (b)

step: 10^{-3}

Figure 15: Extracted logos: a) Without ECC, b) With ECC.

To stress the results, Table 3 presents the NCC between the original logo and the extracted ones.

We see that without the add of an ECC, results are good for $\Delta < 10^{-3}$ but they are almost perfect by using

Table 3: NCC between the original logo and the extracted ones.

Step Δ	10^{-6}	10^{-5}	10^{-4}	10^{-3}
without ECC	0.9998	0.9974	0.9764	0.8051
with ECC	1	1	0.9999	0.8796

the ECC. For $\Delta = 10^{-3}$, the NCC decreases because of the number of non-moved vertices. Fig. 16 illustrated the ratio between the NCC and the percentage of non-moved vertices. Obviously, this percentage is linked with the value of the step Δ. We see that the use of an ECC tends to stabilize the curve and the decrease is less important than without apply it.

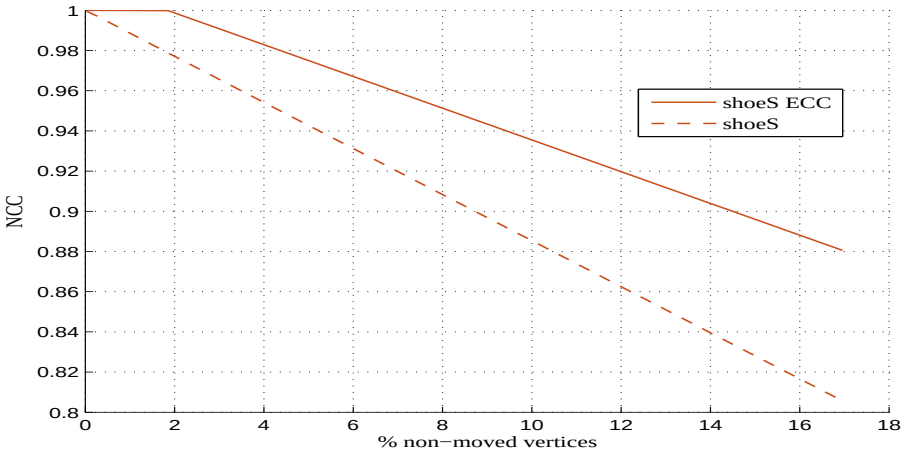


Figure 16: NCC regarding the number of non-moved vertices.

We are also interested in the distortions of the mesh which are produced by embedding the hidden data. Table 4 presents the difference between the original mesh and watermarked ones. Results are presented with three metrics, the Hausdorff distance and the RMSE (root mean square error) using Metro[7] and MSDM2 by Lavoué[15] which is more correlated with human perception.

Table 4: Difference between the original mesh and watermark ones.

Step Δ	Hausdorff	RMSE	MSDM2
10^{-6}	1.10^{-6}	0	0.00259
10^{-5}	10.10^{-6}	0	0.00310
10^{-4}	82.10^{-6}	2.10^{-6}	0.03556
10^{-3}	724.10^{-6}	16.10^{-6}	0.16246

Those results show that the proposed embedding scheme is almost invisible. Indeed, the classic distances are very low and MSDM2 distances are close to zero which corresponds to almost invisible distortion for SVH. The invisibility of the method is shown on an example in Fig. 17.

Furthermore, in Fig. 18, we analyze the distribution of vertices displacement for each step Δ. The displacement distance is noted negative if its direction is oriented toward the center of gravity of the mesh. Note that the vertex displacement are evenly distributed around $y - axis$. Moreover, a large step Δ produces a large variance but more displacement are near to zero since more vertices cannot be moved.

4.3 Full experimentation

In this Section, we present a full experimentation on various meshes from different sources, Armadillo[†], bunny[†], dragon[†], hand[‡], horse[‡], rabbit[‡], shoeS* and shoe*. The mesh shoeS is the simplified version presented in

[†]http://www-graphics.stanford.edu
[‡]http://www-rech.telecom-lille1.eu/madras

(a) (b)

Figure 17: a) Original mesh, b) Watermarked one with non-moved vertices in pink.

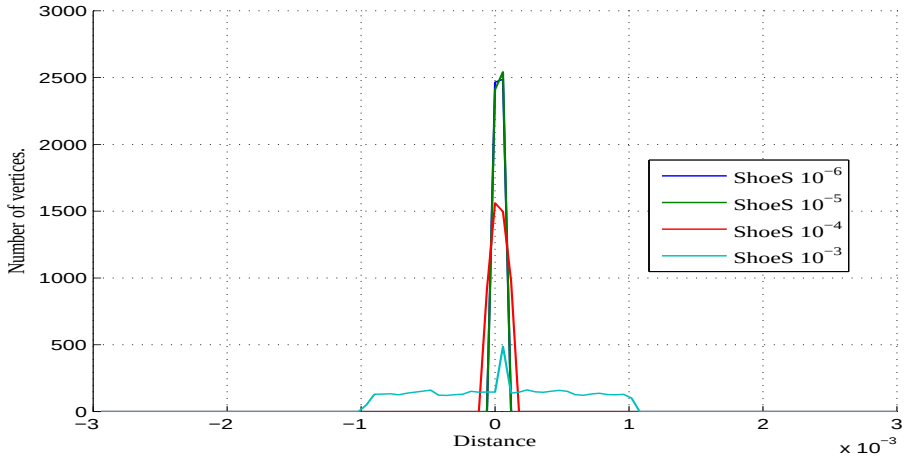


Figure 18: Displacement distribution.

Section 4.1 and obtained from the shoe mesh. These meshes used for experiment have wide variety, different shapes, some are semi-regular like Shoe meshes. They are presented in Table 5, first row gives the number of vertices, the second the mean distance between two vertices in the Hamiltonian path, and the third the time to the whole process. As we can see the mean distance is proportional with the number of vertex. This is due to the normalization of the meshes given by Eq. 9. While, the time increases faster than the number of vertices since the time complexity of the algorithm is cubic.

Table 5: Computational results on meshes.

Mesh	Horse	Bunny	Hand	ShoeS	Dragon	Rabbit	Armadillo	Shoe
# vertices	5002	34834	36616	45002	50000	70658	172974	188609
$md \times 10^{-3}$	14.4384	6.2245	11.3314	4.4227	8.9123	6.0781	3.4081	2.2879
time	$4.5s$	$8m34s$	$9m58s$	$14m53s$	$24m25s$	$36m57s$	$3h52m$	$4h42m$

The results in term of NCC are presented in Fig. 19 which allows to compare the NCC of all extracted logos for each step for each mesh. For all meshes there are two curves, the dashed line correspond to the extracted logo without using the ECC. The solid line of the same color is the extracted logo after applying ECC.

Note that the use of ECC is almost always better. Moreover, the NCC is greater to of 0.97 for $\Delta < 10^{-3}$. For small meshes, results are still good with $\Delta = 10^{-3}$ because of their density, indeed their md value is higher so their vertices can be moved easily. At contrary, the huge meshes for which the algorithm does not allow a lot of displacement have a low NCC however thanks to the size of the hidden logo, the extracted logo is still visible but it is very noisy. As we show in Section 4.2, the value of the NCC is correlated with the number of non-moved vertices. The percentage of non-moved vertices depends on the step Δ and Fig. 20 gives it for each

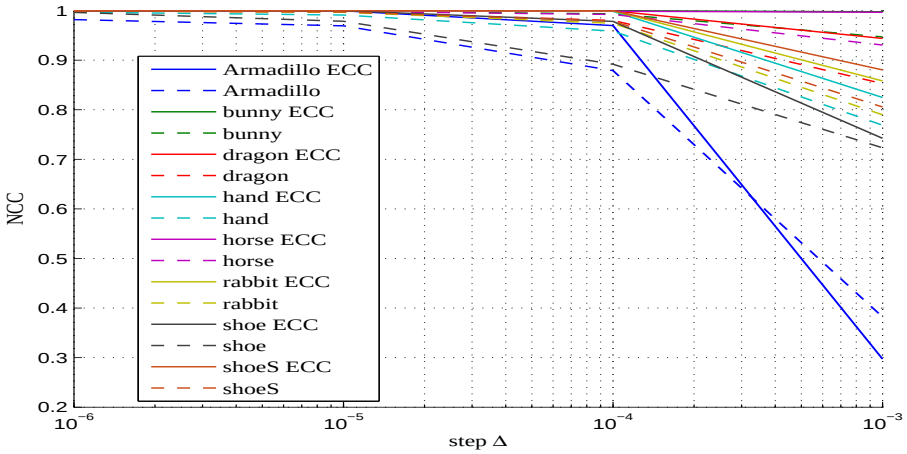


Figure 19: NCC between original logo and the extracted one.

studied mesh.

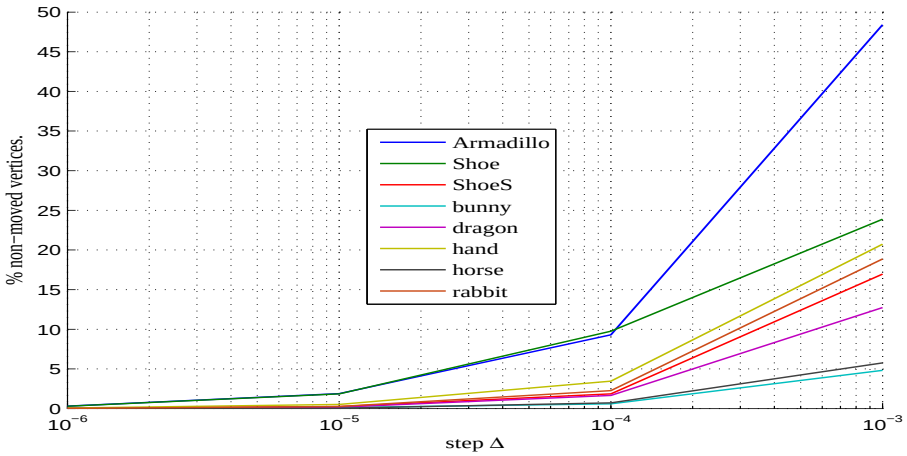


Figure 20: Percentage of non-moved vertices.

When we analyze results presented by Fig. 19 and Fig. 20 we see that the NCC decreases in the same way as the percentage of non-moved vertices increases which confirms the correlation assumptions. In order to show how the embedding stage modifies the vertex cloud, Fig. 21 presents the distortion in terms of Hausdorff distance between the original mesh and the watermarked one.

Distortions are the inserted ones, as a vertex is moved in function of the step Δ. This result is very interesting because it allows to choose the distortion of the mesh. Here the object surface is not visibly distorted. In order to compare our method with the previous ones, Table 6 presents the results of the methods in terms of capacity, Hausdorff distance and $PSNR_1$,[4] for the bunny model which has 34834 vertices.

Table 6: Comparison to previous methods on Bunny model.

	Capacity	Hausdorff Distance	$PSNR_1$
Chao *et al.*[4]	940464	×	100.57
Gao *et al.*[9]	51408	548.10^{-6}	×
Ours with $\Delta = 1.10^{-6}$	54289	1.10^{-6}	127.3

We note that for an equivalent capacity our method distort less the model than the method of Gao *et al.*. And we can see that our capacity is less than that of Chao *et al.* but our PSNR is higher. We also compute some results for other models with $\Delta = 1.10^{-6}$, results are presented in Table 7.

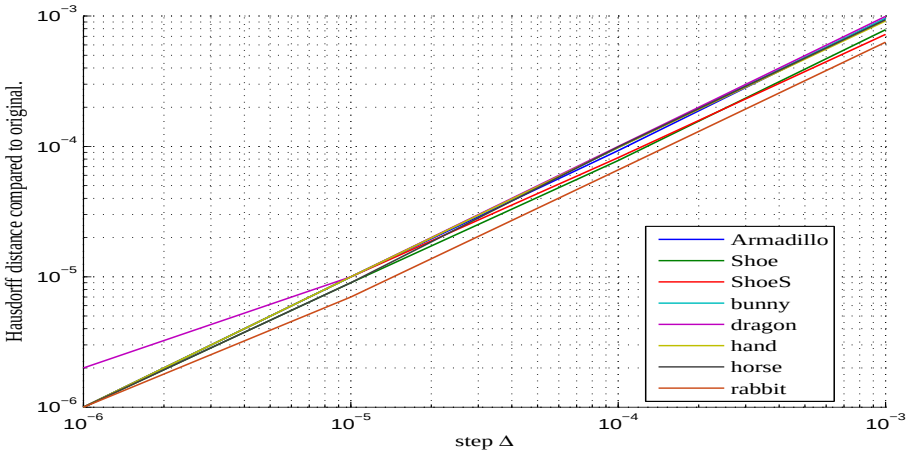


Figure 21: Distorsion vertex.

Table 7: Capacity, Hausdorff Distance and $PSNR_1$ for some models with $\Delta = 1.10^{-6}$.

Model	Size	Capacity	Hausdorff Distance	$PSNR_1$
Horse	5002	7744	1.10^{-6}	127.635
Dragon	50000	77841	2.10^{-6}	132.247
Rabbit	70658	110224	1.10^{-6}	130.002
Venus	100759	157609	1.10^{-6}	133.582
Buddha	144628	225625	1.10^{-6}	111.985
Shoe	188209	294849	1.10^{-6}	126.087

This results show the very low distortions that are produced by our data hiding method while maintaining an interesting capacity.

5. CONCLUSION

In this paper, we have proposed a blind high capacity data hiding method which is designed for embedding additional information. We use a new synchronization based on the geometry of the mesh which orders the vertices by building a Hamiltonian path. The synchronization algorithm is improved in order to jointly embed the data to be hidden. The message is embedded by slightly moving a vertex from its predecessor in the path. This scheme allows to use $n-1$ vertices as medium for hidden data, where n is the number of vertices of the mesh. Furthermore, our method displaces a vertex by moving, in appropriate intervals, its spherical components v_ρ, v_θ and v_ϕ. So we achieve a capacity near of 3 bits per vertex. The method is nearby invisible for a small step Δ. Moreover, the method achieves a good security since the payload cannot be extracted without the knowledge of the secret key.

The proposed method has interesting properties, but we list possible tracks for future work. The first improvement will be to compute the step Δ reliably and automatically way. We will propose another normalization which is linked with our method to compare meshes only by their number of vertex. We also want to improve the performance in term of time complexity. Another improvement could be to steganalyze a watermarked mesh which amounts to distinguish between cover and stego objects. In order to know if the method could be used as a high capacity steganography method. On the contrary, we want to improve the robustness of the synchronization in the purpose that it makes the possible attacks fail. Another possibility is to hugely improve the capacity of payload by changing the embedding process of the proposed method.

REFERENCES

[1] P. Amat, W. Puech, S. Druon, and J. Pedeboy. "Lossless 3D steganography based on mst and connectivity modification". *Signal Processing: Image Communication*, 25(6):400–412, 2010.

[2] F. Cayre, C. Fontaine, and T. Furon. "Watermarking security: theory and practice". *IEEE Transactions on Signal Processing,*, 53(10):3976–3987, 2005.

[3] H.-C. Chang, H.-P. Lee, T. Lin, and T. Truong. "A weight method of decoding the (23, 12, 7) golay code using reduced table lookup". pages 1–5, 2008.

[4] M.-W. Chao, C.-H. Lin, C.-W. Yu, and T.-Y. Lee. "A high capacity 3D steganography algorithm". *IEEE Transactions on Visualization and Computer Graphics*, 15(2):274–284, 2009.

[5] B. Chen and G. W. Wornell. "Quantization index modulation methods for digital watermarking and information embedding of multimedia". *ournal of VLSI signal processing systems for signal, image and video technology*, 27(1/2):7–33, 2001.

[6] J. Cho, R. Prost, and H. Jung. "An oblivious watermarking for 3D polygonal meshes using distribution of vertex norms". *IEEE Transaction on Signal Processing*, 55(1):142–155, 2007.

[7] P. Cignoni, C. Rocchini, and R. Scopigno. "Metro: Measuring error on simplified surfaces". Technical report, 1996.

[8] J. Fridrich. *[Steganography in Digital Media: Principles, Algorithms, and Applications]*. 1st edition, 2009.

[9] X. Gao, C. Zhang, Y. Huang, and Z. Deng. "A robust high-capacity affine-transformation-invariant scheme for watermarking 3D geometric models". *ACM Transaction on Multimedia Computing, Communications and Applications*, 8(2S):34:1–34:21, 2012.

[10] M. J. E. Golay. "Notes on digital coding". *IEEE Proc.*, 37:657, 1949.

[11] R. C. Gonzalez and R. E. Woods. *[Digital Image Processing]*. 3rd edition, 2006.

[12] W. R. Hamilton. "On the icosian calculus". *Proceedings of the Royal Irish Academy*, 6:462, 1853.

[13] V. Itier, W. Puech, J.-P. Pedeboy, and G. Gesquiere. "Construction of a unique robust hamiltonian path for a vertex cloud". pages 105–110, 2013.

[14] A. Kerckhoffs. "La cryptographie militaire". *Journal des sciences militaires*, IX:5–38, 1883.

[15] G. Lavoué. "A multiscale metric for 3D mesh visual quality assessment". *Computer Graphics Forum*, 30(5):1427–1437, 2011.

[16] M. T. Li, N. C. Huang, and C. M. Wang. "A novel high capacity 3D steganographic algorithm". 7(3):1055–1074, 2011.

[17] M. Luo and A. Bors. "Surface-preserving robust watermarking of 3-D shapes". *IEEE Transactions on Image Processing*, 20(10):2813–2826, 2011.

[18] R. Ohbuchi, H. Masuda, and M. Aono. "Watermaking three-dimensional polygonal models". 16:261–272, 1997.

[19] P. Rondao Alface and B. Macq. "From 3D mesh data hiding to 3D shape blind and robust watermarking: A survey". 4499:91–115, 2007.

[20] J. Rossignac. "Edgebreaker: Connectivity compression for triangle meshes". *IEEE Transactions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[21] B. Vasic and B. Vasic. "Simplification resilient LDPC-coded sparse-QIM watermarking for 3D-meshes". *IEEE Transactions on Multimedia*, 15(7):1532–1542, 2013.

[22] K. Wang, G. Lavoué, F. Denis, and A. Baskurt. "A comprehensive survey on three-dimensional mesh watermarking". *IEEE Transactions on Multimedia*, 10(8):1513–1527, 2008.

[23] K. Wang, G. Lavoué, F. Denis, and A. Baskurt. "Robust and blind mesh watermarking based on volume moments". *Computer Graphics*, (35(1)):1–19, 2011.

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