Technical section

Steganography on point-sampled geometry

Chung-Ming Wang\textsuperscript{a,*}, Peng-Cheng Wang\textsuperscript{a,b}

\textsuperscript{a}Institute of Computer Science, National Chung Hsing University, Taichung, Taiwan 402, ROC
\textsuperscript{b}Department of Information Management, Hsiuping Institute of Technology, Taichung, Taiwan 412, ROC

Abstract

We present two new schemes for digital steganography of point-sampled geometry in the spatial domain. To the best of our knowledge, our schemes are the first in the literature for such types of geometry. Both schemes employ a principal component analysis to translate the points’ coordinates to the new coordinate system, resulting in a blind approach. In the first scheme, we establish a list of intervals for each axis according to the secret key. We then embed a bit into each interval by changing the points’ position. In the second scheme, we locate a list of macro embedding primitives (MEPs), and then embed \(c\) bits (\(2 \leq c \leq 6\)) at each MEP, instead of a single bit as in the first approach. We validate our schemes with various models in terms of capacity, complexity, visibility, and security. Experimental results demonstrate that the proposed schemes are robust against translation, rotation, and scaling operations. In addition, our schemes are fast and can achieve high data capacity with insignificant visual distortion in the stego models.

\(\textcopyright\) 2006 Elsevier Ltd. All rights reserved.

Keywords: Steganography; Point-sampled geometry; Principal component analysis

1. Introduction

Steganography uses digital multimedia data such as movies, music, images, and 3D models as cover media to embed hidden information, which are usually denoted as the payload [1,2]. There are many applications for which steganography is a suitable solution, including in-band captioning, covert communication, tamper proofing, authentication, embedded control, and revision tracking. Preferred data hiding algorithms embed as many payloads as possible in the cover media, forming the stego model with as little distortion as possible. Therefore, the two most important criteria in evaluating the technique of data hiding are the amount of payload and the degree to which cover media get distorted.

Point-sampled geometries have emerged in recent years as a versatile representation for geometric models [3]. The surface of a 3D object is described by a set of sample points without further topological information. Numerous point-based rendering systems have been developed to display point-sampled models. To take advantage of such models, many geometry processing applications have also been introduced recently. Unfortunately, research in steganography has not kept pace with the advances of point-sampled geometries. Even though some data hiding and watermarking schemes have been presented for conventional 3D polygonal models [4–13], only one watermarking scheme for point-sampled geometry has been proposed so far [14]. Data hiding schemes remain largely unexplored. This has significantly limited the applications of steganography. Therefore, the need to provide data hiding details for point-sampled geometry is clear.

In general, data hiding in the spatial domain increases capacity but leads to relatively poor robustness. In
watermarking, transform domains offer better robustness. Since we are interested in maximizing capacity, we plan to embed the information into the spatial domain and assume no robustness requirements, except for the similarity transformations like translation, rotation, and scaling. In this paper, we present two novel schemes to tackle the digital steganography of point-sampled geometry.

This paper is organized as follows: Section 2 surveys related works. Section 3 presents in detail the proposed algorithms. Section 4 shows the visual effect for several models and describes experimental results. Conclusions and future work are described in the final section.

2. Related works

Because we have yet to find any related works for data hiding on point-sampled geometry, we first present some data hiding and watermarking schemes of 3D polygonal models in the spatial domain [4,5] and in the transform domain [6,8,9]. We then survey a watermarking scheme recently proposed for point-sampled geometry [14].

The most recent research on data hiding on 3D polygonal models is by Cayre and Macq [4]. There the authors describe a blind scheme in the spatial domain. The key idea is to consider a triangle as a two-state geometrical object. This data hiding method first establishes a list of triangles of the model that will contain the payload. Each triangle that can be embedded with a bit is called an admissible triangle. Then, a vertex’s position at each admissible triangle is modified or not according to the embedding bits. This scheme is robust against translation, rotation, and scaling operations. However, it is not robust against remeshing, mesh simplification, mesh altering, cropping, and other operations.

Earlier, Benedens [5] proposed an algorithm that embeds private watermarks by altering normal distribution. It works in the spatial domain. This algorithm achieves robustness against randomization of points, mesh altering, and polygon simplification. However, the algorithm is not robust against cropping attacks, as the normal distribution is calculated for the entire model.

Kanai et al. [6] were the first to apply a transform-domain watermarking approach on a 3D polygonal model. Their algorithm is a non-blind scheme, which decomposes an original polygonal model by applying wavelet transform several times and then embeds watermarks into the wavelet coefficient vectors. This algorithm is robust against affine transformation, random noise added to vertex coordinates, and other attacks. Their method requires the mesh in a 3D polygonal model to have 4-to-1 subdivision connectivity.

Ohbuchi et al. [8,9] proposed a non-blind algorithm that works in the transform domain. It embeds the watermarks into a 3D polygonal model by deforming the low-frequency components of the shape using mesh spectral analysis. This scheme achieves robustness against similarity transformation, cropping, random noise, mesh simplification, and other operations.

Cotting et al. [14] recently reported a non-blind watermarking approach for point-sampled geometry. It extends existing watermarking algorithms [8,9] designed for 3D polygonal models to point-sampled geometry. Their approach embeds watermarks into the low-frequency components, and employs statistical methods based on correlation to analyze the extracted watermarks. This scheme is robust against low-pass filtering, resampling, affine transformations, cropping, additive random noise, and combinations of the above.

This brief summary of the limited available literature indicates that current algorithms presented for data hiding or watermarking focus on 3D polygonal models, not on point-sampled geometry. In this paper, we intend to be the first to present a data hiding approach for point-sampled geometry.

3. The proposed technique

This section presents two algorithms for hiding data. Our algorithms are designed to accept a point-sampled model (the cover model), a sequence of binary bits, referred to as the payload, and the secret key. The proposed scheme is shown in Fig. 1.

3.1. Symmetrical swap algorithm (SSA)

SSA is the first scheme we propose for digital steganography of point-sampled geometry in the spatial
domain. The SSA embedding algorithm embeds a payload according to the three steps below.

1. Constructing principal component analysis (PCA) axes: We adopt PCA [15] for the point-sampled model to produce three principal axes, constructing a new coordinate system. We then translate the coordinates of the original points to the new coordinate system. Obviously, this new coordinate system has a new origin, which is a gravity center of the point-sampled model; it also has three basis vectors, which are the three principal axes, being referred to as the X-, Y-, and Z-axis, respectively.

2. Creating three sorting lists: Next, we sort the points’ coordinates for each axis. This produces three lists containing sorted indices.

3. Embedding data into lists: We then embed a bit into each interval of the X–Y–Z axes decided by the secret key, which we employ to generate a random sequence with integer values. These values represent the index in our algorithm for embedding bits at corresponding intervals. This operation is called a symmetrical swap procedure [4,7].

For simplicity, we only use the X-axis to illustrate the basic idea. Suppose \( X_1 \ldots X_m \) are X-coordinate values of sorted points \( P_1 \ldots P_m \), where \( m \) is the number of points in the point-sampled model, and each interval (which consists of two values, \( x_n \) and \( x_{n+2} \) \( (1 \leq n \leq m-2) \) on the X-axis) is considered as a two-state object. We define the state of the interval by the X-coordinate value \( x_{n+1} \) of the point \( P_{n+1} \). We divide the interval \( x_n x_{n+2} \) into two subsets \( S_0 \) and \( S_1 \). If \( x_{n+1} \in S_0 \), then we consider that the interval is in a “0” state; otherwise, \( x_{n+1} \in S_1 \), and the interval is in a “1” state, as shown in Fig. 2 [4]. To set the interval in the \( i \) \( (i = 0 \) or \( 1) \) state, either one of these two cases must occur: First, if \( x_{n+1} \in S_0 \), then no modifications need be processed. Second, if \( x_{n+1} \notin S_0 \), then \( P_{n+1} \) has to be shifted toward \( P'_{n+1} \) using the subinterval border as a symmetry axis orthogonal to \( x_n x_{n+2} \) so that \( x_{n+1} \in S_0 \). In other words, the legitimate point \( P_{n+1} \) can be geometrically modified or not, depending on the bit value to be hidden and the initial state of the interval.

The \( P_{n+1} \rightarrow P'_{n+1} \) mapping has to be invariant through similarity transformations. In addition, if we prefer to recover the original state of a legitimate point, the mapping has to be reversible. To avoid visual degradation of the point-sampled model, we can subdivide the interval \( x_n x_{n+2} \) into \( 2k \) \( (k \geq 1) \) subintervals so that geometrical distortion \( |x_{n+1} - x_{n+1}^\prime| \) gets smaller as \( k \) increases. On the other hand, a larger \( k \) can cause bit retrieval errors due to limited machine precision. The \( P_{n+1} \rightarrow P'_{n+1} \) mapping is a symmetry across the closest axis orthogonal to \( x_n x_{n+1} \) that intersects the border of the closest subinterval belonging to \( S_i \).

If we prefer to recover the original state of a legitimate point, it is necessary to store an extra bit for every bit of the payload, and its size is exactly the same as the payload size. The extra bit is set to 1 if the state of the interval is changed and 0 otherwise. The extra bits are constructed during embedding. In this way, exact reversibility is only granted to the holder of extra bits and secret keys.

Using our SSA Extraction Algorithm, we proceed to data extraction with a stego model, the secret key, three principal axes, and the gravity center of the cover model, as shown in Fig. 1. We take the following steps for each axis:

1. Find an interval \( x_n x_{n+2} \) by interpreting the secret key.
2. Extract a data bit from the interval by the X-coordinate value \( x_{n+1} \).
3. Repeat (1) and (2) above for all the legitimate points \( P_{n+1} \) on a given stego model.

In this approach, we can embed a bit at each interval, resulting in a total of \( m/2 \) bits for each axis. Clearly, the data capacity is \( 1.5 m \) for the X–Y–Z axes.

3.2. Encoded swap algorithm (ESA)

ESA is the second scheme we propose for digital steganography of point-sampled geometry in the spatial domain. The macro embedding primitive (MEP) denotes the locality where the payload can be embedded. Note that there are three sorted lists in the ESA scheme, each of which contains \( m/2 \) intervals in the X-, Y-, and Z-axis, where \( m \) is the number of points in the point-sampled model. Moreover, in the ESA scheme, each MEP contains three intervals in the X-, Y-, and Z-axis, and is referred to by an exclusive integer. Fig. 3 shows five MEPS, represented by MEP1, ..., MEP5. The basic concept behind the ESA scheme is that we embed \( c \) bits \( (2 \leq c \leq 6) \) at each MEP, instead of a single bit as in the
SSA approach. This means that in the ESA scheme, when embedding the payload, we consider the X-, Y-, and Z-intervals simultaneously in an MEP, instead of treating them independently as in the SSA scheme.

The ESA embedding algorithm first performs the steps (1) and (2) as in the SSA scheme. Similar to the SSA approach, we first locate an MEP according to the secret key. In this MEP, we then embed 2 bits into three intervals in the X-, Y-, and Z-axis. Similar to the SSA scheme, we define an MEP to have two equal parts, indicating either at the “0” or “1” state. When the intervals in the X-axis, Y-axis, or Z-axis are in the “1” state, we embed the data bits “01”, “10”, and “11” at the corresponding interval, respectively. However, an MEP is embedded with the “00” bits by setting all three intervals in the “0” state. Thus, we can embed 2–6 bits at a single MEP, depending on possibly flipping the states of the intervals with the order from the X-axis, Y-axis, to the Z-axis. Obviously, by referring to the sequence of the payloads, we can embed different numbers of bits into different MEPS. Fig. 3 shows a simple example of the ESA scheme that embeds a payload with the 18 bit string “10011010001111011” in five consecutive MEPS. The left and right sides of the diagram represent the states before and after the embedding. For example, according to the definition of the three axes, the first six bits in the payload, “100110”, represent the order Y-axis, X-axis, Y-axis. Thus, we can only embed the first two bits “10” into the Y-axis in MEP1. This is achieved by flipping the state in the X-axis and keeping the state in the Z-axis unchanged. Similarly, all the next six bits “011011” can be embedded into MEP2 since the order coincides with our definition. Clearly, MEP2 has the maximum data embedding. The next six bits, “000111”, contain two bits “00” at the beginning. According to our definition, they can be embedded into MEP3 by flipping all three intervals to be exclusively at the “0” states, leading to the minimum two bits data embedding. Once again, under this embedding process, we can embed 2–6 bits into an MEP.

With our ESA Extraction Algorithm, inputs to this algorithm are the same as those employed in the SSA extraction algorithm. At each MEP, we extract 2–6 bits, depending on the states detected in the X-, Y-, Z-axis. This extraction algorithm takes the following steps:

1. Find an MEP by interpreting the secret key.
2. Extract 2–6 data bits from this MEP by referring to the states of the three intervals in the X-, Y-, and Z-axis.
3. Repeat (1) and (2) for all the MEPS on a given stego model.

We now derive the lower and upper bounds for the ESA scheme. Let z represent the total data capacity that can be embedded into a point-sampled model with m points. At each MEP, in the worst case, we can embed two bits into an MEP, while in the best case, six bits “011011” are simultaneously embedded. This indicates that the embedding data capacity satisfies the statement $m \leq z \leq 3m$ since there are $m/2$ MEPS in total. We now consider the average case of the data capacity when randomly embedding $c (2 \leq c \leq 6)$ bits into an MEP. We can construct a 3-combination of a 4-set {“00”, “01”, “10”, “11”} by choosing three different or identical elements from the 4-set. This results in a total of 64 3-combinations of the 4-set, as shown in Table 1 with the indices numbered from 1 to 64. Also shown in this table are all the cases of embedding between two and six bits into an MEP. Accumulating the embedded bits at each index leads to a total of (154/64) bits, which are the average data capacity that can be embedded into an MEP. This indicates that the average data capacity is approximately $1.203m$ for a point-sampled model containing $m/2$ MEPS.

We establish evaluation metrics in terms of the imperceptibility, security, capacity, and complexity for the proposed methods. In a still image, one may generally want to evaluate the image distortion by using

![Fig. 3. An illustration of the states before (left) and after (right) the embedding. A bit string “10011010001111011” is embedded into MEPS for X–Y–Z axes. The dotted line subdivides an MEP into two equal parts where the state “0” is defined as on the left part.](image-url)
the signal-to-noise ratio (SNR) computations. However, this metric is dedicated to signals that are regularly sampled, which is not suitable for 3D models. Some proposals have been made for 3D polygonal models, mostly being based on the Haussdorff distance and Laplacian [4]. In the 3D point-sampled geometry, we employ a simple evaluation metric, the root mean square (RMS) ratio, to measure the 3D model distortion.

Security is achieved by using a key to hide a secret payload in an undeterministic way. Since our main goal is to focus on capacity, we assume that no robustness is required, except for some basic operations, such as translation, rotation, and scaling. For simplicity, we divide the interval into two subintervals to hide one bit per interval in the SSA approach. Nevertheless, our method is able to hide more than one bit per interval. In general, when we divide the interval into $2^d$ subintervals, we can embed $d$ bits into each interval. Clearly, the value of the parameter $d$ depends on the machine precision. Finally, we estimate the computation complexity of our scheme by giving the execution time for various models.

4. Experimental results

We implemented the proposed two techniques using Microsoft Visual C++ programming language. We also performed experiments to validate the feasibility of our algorithms. Results were collected on a personal computer with a Pentium IV 2.8 GHz processor and 512 MB memory.

Fig. 4 demonstrates a shading display for two cover models, Bunny (35947 points) and Venus1 (33591 points). Fig. 5 compares the visual effects between the cover and stego models using the SSA approach. Note that no visual distortion can be perceived between the cover and stego models. For clarity, we employed different colors to indicate positions of data hiding in the stego models; we used red to represent the legitimate

---

Table 1: An illustration of 64 3-combinations and embedded bits

<table>
<thead>
<tr>
<th>Index</th>
<th>3-combinations</th>
<th>Embedded bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000000</td>
<td>11</td>
</tr>
<tr>
<td>1</td>
<td>000001</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>000010</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>000100</td>
<td>01</td>
</tr>
<tr>
<td>4</td>
<td>001000</td>
<td>01</td>
</tr>
<tr>
<td>5</td>
<td>010000</td>
<td>01</td>
</tr>
<tr>
<td>6</td>
<td>100000</td>
<td>01</td>
</tr>
</tbody>
</table>

Fig. 4. A shading display for the cover models Bunny (left) and Venus1 (right).
points on which their positions were geometrically modified; we used blue to represent the positions of the legitimate points that remained unchanged. Note that we did not embed any payloads at the corresponding legitimate points inside an interval/MEP when the distance between legitimate points and its neighbor points in the same interval was smaller than a given threshold. This avoided possible extraction errors due to

![Cover models (left) and stego models (right) of the SSA approach for Horse1, Bunny, Venus1, Venus2, Elephant, and Horse2.](image)

Red represents the positions of the legitimate points that were modified during the embedding process, while blue indicates those positions that remained unchanged.
Fig. 5. (Continued)
machine imprecision. Finally, we used white to indicate points where no data was embedded.

Fig. 6 illustrates the visual effect of the ESA approach for payloads with the average and best case. In the average case, we encoded the payloads randomly, while in the best case, we constructed the payloads purposefully according to the order in the $X$-, $Y$-, and $Z$-axis. Once again, though the data capacity is 1.203 $m$ and 3 $m$ in both cases, we detected insignificant visual distortion in the stego models.

Table 2 contains the experimental results of the SSA approach. We list, for various point-sampled models, their data capacities, RMS ratio, and execution time for PCA, embedding, and extraction. As expected, the data...
capacity is nearly 1.5 times the number of points when we hide one bit per interval. (The RMS ratio is the RMS values, derived from the data embedding, over the diagonal length of the bounding volume for a point-sampled model.) The small RMS ratios indicate insignificant position changes during the data hiding scheme.
The embedding time was much longer than the extraction time because we needed to sort points, according to the coordinate values of the X-, Y-, and Z-axis, so as to construct intervals and their corresponding legitimate points. Note the embedding time is proportional to the number of points in the point-sampled models.

Table 3 illustrates the experimental results using the ESA approach. We present model size, embedding data capacity, RMS ratio, and execution time for payloads with the average and best cases. As expected, the data capacity is close to 1.2 and 3 times the number of points for these two cases, respectively. We detected no errors when extracting the payloads. Again, RMS ratios are small, indicating minute position movements in data hiding. The embedding time is still much longer than the extraction time due to the sorting operations.

5. Conclusions and future work

Several data hiding schemes have been presented in the spatial domain for polygonal models. However, to our knowledge, no data hiding algorithm for a point-sampled geometry has ever been presented in the literature. This paper presented two novel schemes, SSA and ESA, for data hiding on such point-sampled geometry using the PCA. Our schemes are a blind approach in the spatial domain; they are simple to implement; and they recover the original model with the exact information that has been stored. Our approaches are robust against translation, rotation, and scaling operations. For the SSA approach, the data capacity in bits achieves 1.5 times the number of points in the cover models. The capacity is increased considerably when we divide the interval into $2^d$ subintervals and embed $d$ bits into each interval. For the ESA approach, the data capacity in bits achieves nearly 1.2 and 3 times the number of points when using payloads with the average and best cases, respectively. To address the issue of imperceptibility, we used a metric, RMS ratio, to measure distortion. RMS ratios and visual appearance of images showed insignificant distortion for the stego models. Security was also achieved by using a key to embed the data. We believe that extracting the data

<table>
<thead>
<tr>
<th>Cover</th>
<th>Number of points</th>
<th>Data capacity (bits)</th>
<th>RMS ratio</th>
<th>PCA execution time (seconds)</th>
<th>Embedding execution time (seconds)</th>
<th>Extraction execution time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse1</td>
<td>48485</td>
<td>72456</td>
<td>0.0000082</td>
<td>0.047</td>
<td>411.390</td>
<td>0.016</td>
</tr>
<tr>
<td>Bunny</td>
<td>35947</td>
<td>53862</td>
<td>0.0000113</td>
<td>0.032</td>
<td>232.969</td>
<td>0.016</td>
</tr>
<tr>
<td>Venus1</td>
<td>33591</td>
<td>50310</td>
<td>0.0000114</td>
<td>0.032</td>
<td>208.046</td>
<td>0.016</td>
</tr>
<tr>
<td>Venus2</td>
<td>19847</td>
<td>28482</td>
<td>0.0000159</td>
<td>0.016</td>
<td>68.281</td>
<td>0.015</td>
</tr>
<tr>
<td>Elephant</td>
<td>19753</td>
<td>29301</td>
<td>0.0000195</td>
<td>0.015</td>
<td>66.812</td>
<td>0.015</td>
</tr>
<tr>
<td>Horse2</td>
<td>3000</td>
<td>4434</td>
<td>0.000133</td>
<td>0.000</td>
<td>0.75</td>
<td>0.000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cover</th>
<th>Number of points</th>
<th>Data capacity (bits)</th>
<th>RMS ratio</th>
<th>PCA execution time (s)</th>
<th>Embedding execution time (s)</th>
<th>Extraction execution time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horse1</td>
<td>48485</td>
<td>57900</td>
<td>0.0000078</td>
<td>0.047</td>
<td>373.422</td>
<td>0.016</td>
</tr>
<tr>
<td>Bunny</td>
<td>35947</td>
<td>41780</td>
<td>0.0000230</td>
<td>0.047</td>
<td>206.328</td>
<td>0.016</td>
</tr>
<tr>
<td>Venus1</td>
<td>33591</td>
<td>40300</td>
<td>0.0000109</td>
<td>0.032</td>
<td>191.172</td>
<td>0.016</td>
</tr>
<tr>
<td>Venus2</td>
<td>19847</td>
<td>23080</td>
<td>0.0000168</td>
<td>0.016</td>
<td>61.562</td>
<td>0.015</td>
</tr>
<tr>
<td>Elephant</td>
<td>19753</td>
<td>23920</td>
<td>0.0000200</td>
<td>0.016</td>
<td>58.172</td>
<td>0.015</td>
</tr>
<tr>
<td>Horse2</td>
<td>3000</td>
<td>3668</td>
<td>0.0001328</td>
<td>0.000</td>
<td>0.703</td>
<td>0.000</td>
</tr>
</tbody>
</table>

Table 2
Experimental results using the SSA approach

Table 3
Experimental results using the ESA approach. For each model, we illustrate the payloads with the average case (top) and those with the best case (bottom)
without the key is virtually impossible. This problem is NP-hard in the cryptographic sense [4]. In addition, we estimated the complexity of our scheme by giving execution times for various models. Our scheme is fast; for a model with around 36,000 points, the time required to embed data is approximately 233 and 207 s for the proposed schemes. Finally, we demonstrated the feasibility of our algorithms for a number of point-sampled models. We believe our schemes are appropriate for most point-sampled models.

In this paper, we only hid one bit per interval for the SSA approach. In the future, we intend to improve the hiding capacity with \( d (d > 1) \) bits per interval by dividing the interval into \( 2^d \) subintervals.

Acknowledgments

We would like to thank Mr. Monty Vierra for proofreading the original manuscript and providing many valuable suggestions to clarify this paper’s presentation. This research was supported by the National Science Council (NSC, Taiwan) under the grant number NSC 93-2213-E-005-018.

References


