An Efficient Information Hiding Algorithm for Polygon Models

Chung-Ming Wang
Yu-Ming Cheng
cmwang@cs.nchu.edu.tw
ssmtt.cym@msa.hinet.net
Institute of Computer Science, National Chung Hsing University, Taichung, Taiwan

Abstract
We present an efficient digital steganographic technique for three-dimensional (3D) triangle meshes. It is based on a substitutive blind procedure in the spatial domain. The basic idea is to consider every vertex of a triangle as a message vertex. We propose an efficient data structure and advanced jump strategy to fast assign order to the message vertex. We also provide a Multi-Level Embed Procedure (MLEP), including sliding, extending, and rotating levels, to embed information based on shifting the message vertex by its geometrical property. Experimental results show that the proposed technique is efficient and secure, has high capacity and low distortion, and is robust against affine transformations (which include translation, rotation, scaling, or their combined operations). The technique provides an automatic, reversible method and has proven to be feasible in steganography.

Categories and Subject Descriptors: D.2.11 [Software Engineering]: Information Hiding

1. Introduction
The goal of information hiding is to conceal messages inside other harmless messages in a way that does not allow any enemy to even detect that there is a second secret message present [JJ98], [PAK99]. Whereas classical cryptography is about protecting the content of messages, information hiding, or steganography, is about concealing their very existence [JJ98], [PAK99], [KP00].

Digital media can be transmitted conveniently over networks. Therefore, protecting secret messages during transmission is an important issue with the development of internet technologies. In computer-based steganography, images, audio files, documents, and even three-dimensional (3D) models may all serve as innocuous-looking hosts for secret messages. Recently, 3D polygon models have become of great interest since 3D hardware has become much more affordable than ever, allowing the wide use of such models. Because of the natural differences between 3D polygonal models and traditional cover media, such as 2D images, traditional schemes of steganography are not very well suited to 3D polygon files. How to fully exploit the advantage of 3D polygonal models is an important research issue. Preferred steganography algorithms need to satisfy three basic requirements; namely security, high capacity, and low distortion. Performance is also an important trade-off attribute, especially when trying to recover hidden messages [KP00], [BW02].

This paper presents an efficient, high capacity, secure, low distortion, and robust steganographic technique for 3D triangle meshes. We achieve the efficiency by relying on a hierarchical kd-tree [Ben75], [FBF77], [Ben90], [AM93], a triangle neighbor table, and an advanced jump strategy. We propose a Multi-Level Embed Procedure (MLEP) that can embed at least three bits per vertex. In addition, a secret key is used on the MLEP for more security. We define a specific metric for distortion evaluation that has been validated by many tests. We can easily control the distortion rate by fine-tuning those parameters and anticipate possible distortion rate before embedding, which can effectively decrease the time of each trial. Finally, our algorithm is robust against affine transformations. Experimental results show significant improvements in terms of capacity and computation time with respect to the most recent, advanced technique, e.g. that of Cayre and Macq [CM03].

The rest of this paper is organized as follows. In section 2, related work is described. Then, our algorithm, including the information embedding process and extraction procedure, is presented in section 3. Experimental results are shown in section 4, followed by a brief conclusion and suggestions for future work in section 5.

2. Related Work
There are several information hiding schemes for 3D models that have been proposed either in the frequency domain [KDK98], [PHF99], [OTM*01], [OMT02], [CWP*04] or in the spatial domain [OMA97], [OMA98a], [OMA98b], [CM03]. These schemes often embed messages by relying on additive or substitutive schemes. Since we are interested in maximizing capacity and assume no...
robustness requirements, we prefer to develop a technique based on a substitutive blind scheme in the spatial domain. How to exploit the advantage of 3D more from a capacity point of view is one of our main goals here.

Ohbuchi et al. [OMA98b] discuss techniques for embedding data into 3D polygonal models of geometry. They introduced the concept of data embedding into such models. The several simple data embedding algorithms described in [OMA98b] are, however, merely examples of how to provide information channels in a 3D polygonal model.

Cayre and Macq [CM03] presented a steganography algorithm recently. Theirs is a blind scheme in the spatial domain, where the key idea is to consider a triangle as a two-state geometrical object, which means the position of the orthogonal projection of the triangle summit on the bottom edge can be divided into a “0” or “1” state according to the bit to be hidden. Furthermore, their scheme extends and enriches one of the simplest techniques, called the triangle strip peeling sequence (TSPS) [OMA97], [OMA98a]. The basic idea behind the TSPS algorithm is the insertion of bits while moving on the mesh. As a result, their scheme achieves a more secure algorithm and a fully automatic implementation.

For these reasons, we further look at the research done by Cayre and Macq [CM03], which provides a good digital steganographic property for 3D triangle meshes. Their algorithm requires two steps: a stencil and a macro embedding procedure. A stencil is the list of triangles of the mesh that will contain the message, which usually does not fill all models fully, especially in models that have holes. A macro embedding procedure, which can normally embed one bit per vertex, considers each triangle as a two-state object and embeds messages by shifting a vertex if needed.

Although their scheme provides a good property, their algorithm still suffers from two major drawbacks: it doesn’t have a large enough capacity and it takes up too much time. First, as to capacity, it doesn’t take full advantage of vertices in 3D polygonal models and it doesn’t employ the full capacity of the cover media. They derived their scheme from the quantization index modulation (QIM) concept, which normally hides one bit per vertex and usually doesn’t fill all models fully, since the way they created the stencil can not guarantee a visit to every triangle mesh. For instance, if a triangle is just connected with one other, namely, only one neighbor of it, when the stencil is traced on it from this one neighbor, the stencil will stop, since no other triangle can be traced. Second, their scheme is inefficient because it takes too much preprocessing time to create the stencil, which exponentially grows when the filling rate reaches 80%. Huge retraced triangles happen with respect to the filling rate and exponentially grow when the filling rate approximates the upper bound of the capacity. Obviously, it seriously affects the complexity of the models. Embedding a message on larger models by their scheme is almost impossible.

Cayre et al. [CDS04] proposed a framework for fragile watermarking of 3D triangle meshes for authentication and integrity. Their scheme uses the so-called indexed localization technique. The purpose of such a localization is to explicitly embed the number of the watermark bit along with its value through a kind of geometrical Division Multiple Access (DMA). In particular, it embeds watermark values (bits) through a geometrical invariant in every triangle, say triangle ΔABC, along a special traversal of the mesh connectivity graph. This scheme is exactly the same as in [CM03]. The watermark number, however, is embedded through another geometrical invariant, which is the ratio of the area of the imaginary square drawn from AB and the area of the triangle ΔABC, in other words, the height H perpendicular to the reference edge AB. In addition, Cayre et al. proposed a fast mesh traversal method, which is inspired by [TG98]. Their method grows and propagates edge cycles on the connectivity graph that finally conquers every vertex. No specific traversal order is needed at decoding to cope with mesh cropping. Every triangle is processed one after one another, independently of any connectivity relationship when decoding, resulting in a linear time mesh traversal.

Although the framework proposed by Cayre et al. is aimed at fragile watermarking [CDS04], we think their ideas could apply to data hiding after some proper modifications. However, extending their framework for data hiding has two drawbacks. First of all, it doesn’t take into consideration the full capacity of a 3D polygonal model. It utilizes only two independent dimensions of every triangle in a 3D model for data embedding. This results in hiding two bits per vertex. In contrast to their approach, our scheme can hide three bits per vertex, taking full vertex advantage of a 3D model. Second, the data extracted might be incorrect. If we embed watermark values, instead of watermark numbers, into the height H perpendicular to the reference edge AB, it is unclear how the watermark values can be extracted correctly without considering any mesh connectivity relation in a 3D polygonal model.

Therefore, in section 3, we propose an effective way of modifying the message vertex by geometrical properties. Considering capacity, our approaches fill all models fully, and we give an upper bound on the maximal capacity of our approaches. We decrease visual degradation by relying on smaller moves with respect to the human visual system (HVS). The geometric distortion from the original polygon due to the embedding can be forecasted and controlled in the proposed approaches. We improve performance by relying on a hierarchical kd-tree [Ben75], [FBF77], [Ben90], [AM93], which is a hierarchical data structure for efficiently finding nearest neighbors in multi-dimensional space. We also enhance performance by using a triangle neighbor table and an advanced jump strategy. We estimate the complexity of our approaches by giving common processing times for a well known model. Finally, we improve security by relying on a secret key. We prove the security of our approaches by giving an estimate, since a practical verification of the theoretic paradigms is untraceable in our case.
3. The Proposed Technique

This section describes the proposed algorithm for efficient information hiding for polygon models. Our algorithm consists of two separate procedures: an embedding procedure and an extraction procedure, both containing four stages (see Figure 1). The time used to embed and to retrieve the message is the same, as both operations are symmetrical. The embedding process proceeds as follows. In a preprocessing step, the initial triangle for embedding is resolved by principal component analysis (PCA) [Ren02]. A sequence list of triangles in the cover model that will contain the message is established starting from the initial triangle. It depends on a secret key and accelerated reliance on the hierarchical kd-tree, triangle neighbor table, and advanced jump strategy, details of which are described in section 3.1.2. Every triangle of the sequence list is modified or not according to the binary symbol it has to convey. Finally, the resulting stego model and recovery key are obtained after the embedding step. During extraction, the following steps are performed. The potentially embedded model is analyzed using the PCA technique in the preprocessing step. Then, a sequence list of triangles in the stego model is established, starting from the initial triangle. After establishing the sequence list, the message is extracted using a Multi-Level Embed Procedure (MLEP). Finally, the stego model is recovered optionally, producing the recovery model, which is the same as the original cover model.

3.1. Information Embedding

The embedding procedure of our steganographic system proceeds in multiple steps as described in the next subsections.

3.1.1. Preprocessing

Information hiding schemes often can be processed in the spatial domain or in the frequency domain, and they embed messages by relying on additive or substitutive schemes. As mentioned earlier, we have chosen a substitutive blind scheme. To use 3D to get greater capacity, we treat every vertex of a 3D polygonal model as a message vertex which can be represented by at least three bits in the 3D space; using all vertices of the model, we can easily rely on our steganographic approach. Principal component analysis (PCA) determines the initial triangle; it gives three principal axes centered on the gravity center of the model, whose intersections with the mesh lead to a small number of possible initial triangles. Our second step requires a sequence list of triangles in the cover model.

3.1.2. Sequence List

To embed information, we must first choose a sequence list (called a variable stencil in the following), all the processes here being processed automatically. The length of the variable stencil depends on the embedding rate we desire, the maximal rate being \((3(n-2)-1)/n\times100\%\), where \(n\) is the vertex numbers of the 3D model. After the initial triangles are decided, the next triangles in the list are selected depending on the bit value of the binary pseudo-noise sequence generator. If the bit value is “0”, then the next triangle is the first one of the initial triangle in clockwise order; otherwise, the next triangle is the second one in clockwise order (see Figure 2). In addition, if the next triangle decided upon has been traced before, we simply go on through the triangle with respect to the bit value, without considering the triangle for embedding or decoding. Moreover, if a triangle cannot be traced, the next triangle is selected.

![Figure 1](image1.png)

**Figure 1:** An overview of the embedding procedure and the extraction procedure.

**3.1. Information Embedding**

The embedding procedure of our steganographic system proceeds in multiple steps as described in the next subsections.

**3.1.1. Preprocessing**

Information hiding schemes often can be processed in the spatial domain or in the frequency domain, and they embed messages by relying on additive or substitutive schemes. As mentioned earlier, we have chosen a substitutive blind scheme. To use 3D to get greater capacity, we treat every vertex of a 3D polygonal model as a message vertex which can be represented by at least three bits in the 3D space; using all vertices of the model, we can easily rely on our steganographic approach. Principal component analysis (PCA) determines the initial triangle; it gives three principal axes centered on the gravity center of the model, whose intersections with the mesh lead to a small number of possible initial triangles. Our second step requires a sequence list of triangles in the cover model.

**3.1.2. Sequence List**

To embed information, we must first choose a sequence list (called a variable stencil in the following), all the processes here being processed automatically. The length of the variable stencil depends on the embedding rate we desire, the maximal rate being \((3(n-2)-1)/n\times100\%\), where \(n\) is the vertex numbers of the 3D model. After the initial triangles are decided, the next triangles in the list are selected depending on the bit value of the binary pseudo-noise sequence generator. If the bit value is “0”, then the next triangle is the first one of the initial triangle in clockwise order; otherwise, the next triangle is the second one in clockwise order (see Figure 2). In addition, if the next triangle decided upon has been traced before, we simply go on through the triangle with respect to the bit value, without considering the triangle for embedding or decoding. Moreover, if a triangle cannot be traced, the next triangle is selected.
found in clockwise order, we simply get a new triangle from another one of the current triangles. If some triangles do not have the next triangle on the first or second side, we can go back to the previous triangle and trace another side for the next triangle, and so on until the embedding rate meets our needs.

Although this scheme can trace all the available triangles, it is still time consuming. In fact, this processing time is the most costly of our scheme. As noted before, generating the sequence list requires large search and compare operations to decide the next triangles. These operations are the main cause of the problem. For that reason, we devised three ways to improve performance: a hierarchical kd-tree, a triangle neighbor table, and an advanced jump strategy. Using these three methods together speeds up the entire process, and a faster steganographic approach means that it is possible to carry more information using a larger model.

First, to offset the added expense of time involved in tracing triangles, we create a hierarchical kd-tree structure—even though we will use it later. Assume a set \( P \) of vertices in 3D space. These vertices are preprocessed into the kd-tree structure, so that given any query vertex \( q \), those vertices of \( P \) nearest to \( q \) can be reported efficiently. The kd-tree only finds the nearest vertices in 3D space. Therefore, we need to treat every triangle as a vertex to efficiently find the candidate neighbors of every triangle. To do so, we use the gravity center of a triangle instead of the triangle itself. This limits the number of candidate triangles.

Second, we construct a triangle neighbor table (TNT). This table indicates the neighbors and connective relations of the available triangles. By itself, however, the TNT is still time consuming when using a large model, because constructing the table requires large search and compare operations. This is where the kd-tree comes into play: it offsets the expense of those operations by decreasing the number of candidate triangles needed to make the comparison.

Third, we also propose an advanced jump strategy (AJS) to improve performance. Tracing triangles over and over takes a great deal of processing time; towards the end of the tracing process even more time is needed as the process nears the limit of its capacity. We avoid this problem altogether. Instead of repeatedly retracing the same triangles, the AJS chooses a new starting triangle based on PCA ordering. This ordering is decided according to the position of the orthogonal projection of the gravity center of every triangle on the first principal axis. The AJS then links this new ordering to the tail of the previous variable stencil, and it does so automatically. The resulting triangle has never been traced before, and it only has one vertex that has never been used before. In other words, there are two vertices of this triangle that have been used already. Consequently, the AJS efficiently avoids retracing triangles that have already been traced. This greatly improves performance, especially near the capacity limit.

In short, these three strategies effectively decrease overall processing time. Once the sequence list is ready, we can begin to embed the information. Since all procedures of the algorithm are symmetrical, we can easily maintain the sequence list in both the embedding and extracting procedures.

3.1.3. Multi-Level Embedding Procedure (MLEP) Since, as we mentioned, our steganographic technique is based on a substitutive blind procedure in the spatial domain, we consider every vertex of a triangle as a message vertex. To achieve this, we propose a Multi-Level Embedding Procedure (MLEP) to embed information by shifting the message vertex. Every level of the MLEP can be used singly or all simultaneously, since these approaches are independent. These levels are called Sliding, Extending, and Rotating. In MLEP, we embed the information by modifying the message vertex based on geometrical properties; it guides the change of the position of the orthogonal projection of the message vertex on the bottom edge, height of the triangle, and angle of the triangle plane.

![Figure 3: Decomposition of the edge AB into two interleaved sets M₀ and M₁, both sets having order/2 subsets.](image)

In the sliding level, each triangle is easily treated as a two-state geometrical object; this method is similar to Cayre and Macq [CM03]. The position of the orthogonal projection of the triangle summit \( C \) on the entry edge \( AB \) is denoted as \( P(C)_{|AB} \). Extending the QIM concept to 3D triangle meshes, the \( AB \) interval is divided into two sets (\( M₀ \) and \( M₁ \)), and both sets have order/2 subsets, as shown in Figure 3. If \( P(C)_{|AB} \in M₀ \), then we consider the triangle is in a “0” state; otherwise, \( P(C)_{|AB} \in M₁ \), and the triangle is in a “1” state (see Figure 4). To embed messages \( i (i=0 \text{ or } 1) \) in the triangle, two cases occur:

- \( P(C)_{|AB} \in M₀ \): No modifications are needed.
- \( P(C)_{|AB} \notin M₀ \): \( C \) has to be shifted toward \( C' \) so that \( P(C')_{|AB} \in M₀ \).

![Figure 4: Sliding level of the MLEP.](image)

The \( C \rightarrow C' \) mapping has to be reversible and invariant through affine transformations. Moreover, we extend this partition of the entry edge to the whole line defined by \( A \)
and B. In this way, we can handle triangles for which the projection of C falls out of the segment AB.

Similarly, we apply the same basic idea to the extending level and embed messages in the height of the triangle; this idea is somewhat similar to [CD04] even though theirs is aimed at fragile watermarking. First, let a vertex V and the line defined by V and B be orthogonal to line AB; we define the state of the triangle by the position of the orthogonal projection of the triangle summit C on the virtual edge VB, which position is denoted as \( P(C) \). We also divide the VB interval into two sets \( M_0 \) and \( M_1 \) (see Figure 5). In addition, we define \( \Delta T \) as the distance threshold ratio, \( DT \) as the distance threshold, and \( L \) as the length of entry edge AB of the initial triangle. We replace \( P(C) \) by \( D \) in the following explanation. For robust reasons, we prefer to use the ratio of edge AB of the initial triangle to control movement of the message vertex here (see equation 1); in this way we can guard against affine transformations correctly.

\[
\Delta T = DTR \times L \tag{1}
\]

To embed messages \( i \) (\( i=0 \) or \( 1 \)) in the triangle, two cases occur:

- \[ \left[ \begin{bmatrix} \frac{\Delta T}{\Delta L} \\frac{\Delta T}{\Delta L} \end{bmatrix} \right] \text{ mod } 2 \in M_i : \text{ No modifications are needed.} \]
- \[ \left[ \begin{bmatrix} \frac{\Delta T}{\Delta L} \\frac{\Delta T}{\Delta L} \end{bmatrix} \right] \text{ mod } 2 \notin M_i : \text{ C has to be shifted toward} \]

\( C'' \) by adding or subtracting extending vector \( \overrightarrow{EV} \).

\[
\overrightarrow{EV} = p \times \Delta T \times \left( \begin{bmatrix} CD \\ CD \end{bmatrix} \right), \tag{2}
\]

where the \( p \in \{-1, 1\} \), which is the pseudo random number sequence (PRNS) generated from a known secret key; this way can improve security well. Equally, the \( C \rightarrow C'' \) mapping has to be reversible and invariant through affine transformations.

**Figure 5: Extending level of the MLEP.**

Finally, we apply the same concept to the rotating level and embed messages in the degree of the angle between both triangle planes. First, let a reference vertex G of this level be the gravity center of the initial triangle. The vertices of the initial triangle are not used for embedding messages on this level. When we process another triangle \( \Delta ABC \), both triangles \( \Delta ABC \) and \( \Delta ABG \) form two individual planes. We define the degree of the angle between the two planes as \( \theta \). Based on this angle, we can simply embed or extract our secret messages. Similarly, we divide the range of the degree of the angle into two sets \( M_0 \) and \( M_1 \) (see Figure 6). Let \( E_3 \) be a plane with the normal vector \( \overrightarrow{AB} \) and \( C, D, C''' \) be the vertices on the same \( E_3 \) plane. Then, we assume that the vertex \( D \) is the center of a sphere and the radius of it is \( |\overrightarrow{DC}| \) (replaced by \( r \) in the following explanation). We can fine tune the degree of the angle by adding or subtracting the degree of the angle \( \varphi \), which depends on the equation of the inner product of the vectors. As a result, we get the following system of equations:

\[
\begin{align*}
\overrightarrow{AB} \cdot (x-x_c, y-y_c, z-z_c) &= 0 \\
\overrightarrow{DC} \cdot \overrightarrow{DC} &= \overrightarrow{DC}^2 \\
\cos \varphi &= r^2 \cos \varphi, \\
\text{where } r &= |\overrightarrow{DC}| = |\overrightarrow{DC}|
\end{align*} \tag{3}
\]

To embed messages \( i \) (\( i=0 \) or \( 1 \)) in the triangle, two cases occur:

- \[ \left[ \begin{bmatrix} \theta \\ \varphi \end{bmatrix} \right] \text{ mod } 2 \in M_i : \text{ No modifications are needed.} \]
- \[ \left[ \begin{bmatrix} \theta \\ \varphi \end{bmatrix} \right] \text{ mod } 2 \notin M_i : \text{ C has to be shifted toward} \]

\( C''' \) by adding or subtracting the degree of the angle \( \varphi \).

Equally, the \( C \rightarrow C''' \) mapping has to be reversible and invariant through affine transformations. Every vertex of the model should perform this level except vertex \( C \) of the initial triangle, since the reference vertex \( G \) is the gravity center of the initial triangle.

**Figure 6: Rotating level of the MLEP.**

In fact, every method of these levels is not limited to embedding one bit per vertex. These methods really can partition more delicately to get a larger capacity; the real limitation is computer precision. Since the jump variable stencil created by our approach easily contains all vertices of a model, if we simply embed one bit per vertex on every level, we can define a simple upper bound for the capacity of our scheme. Let \( M_{maxbit} \) be the maximal number of bits actually embedded and \( N_{vertices} \) be the number of vertices in the mesh. We can then state the following equation on our scheme:
\[ M_{\text{maxbits}} = 3(N_{\text{vertex}} - 2) - 1. \]  

(4)

Every vertex corresponds to at least three bits of information, except the vertices \( A \) and \( B \) of the initial triangle. These two vertices are never used for embedding messages, since they are bases of the extraction procedure. In addition, vertex \( C \) of the initial triangle is only used for embedding messages in the sliding and extending level, since the reference vertex \( G \) of the rotating level is obtained from the gravity center of the initial triangle \( \Delta ABC \). So the theoretical upper bound for the capacity of our scheme is approximately triple the number of vertices inside the cover mesh.

Next, we turn to the stego model and the recovery key.

3.1.4. Data Storage In this final step, the stego model and the recovery key are created. Our approach achieves reversibility based on a recovery key which is constructed during embedding and kept private for retrieving the perfect original 3D models. The key is a string containing bits indicating whether a change was needed for every shifted message vertex. In this way, exact reversibility is only granted to the holder of the recovery and secret key.

The MLEP algorithm was developed to decrease distortion and increase capacity with respect to the HVS. To deal with distortion, we define a maximal distortion for the MLEP; this helps us to understand the reasons for the distortion. After this analysis, we can easily control the distortion rate by fine tuning those parameters. Recall that the sliding level of the MLEP embeds messages by shifting the message vertex \( C \) toward \( C' \); the maximal distance of the shift is denoted as \( \lambda_{\text{max}} \). The maximal distance of the shift of the extending level and rotating level is \(|EV|_{\text{max}}\) and \(2r_{\text{max}}\cos((\pi - \phi)/2)\). As a result, the message vertex will be shifted toward the diagonal vertex in a box in the worst case. Here \( S_{\text{dist}} \) denotes the maximal distance of the shift:

\[ S_{\text{dist}} = \sqrt{\lambda_{\text{max}}^2 + (|EV|_{\text{max}}^2 + (2r_{\text{max}}\cos((\pi - \phi)/2))^2} \]  

(5)

Fortunately, the \( S_{\text{dist}} \) is always much smaller than the real distortion (see Table 1), since the probability of the three worst cases happening at the same vertex is rare and never happens in every vertex. In fact, the \( S_{\text{dist}} \) is somewhat similar to the Hausdorff distance, but always larger, and it is useful for forecasting and controlling the distortion rate. The MLEP is better than only using the method of a single level, since the \( S_{\text{dist}} \) is always much smaller than \( \lambda_{\text{max}} + |EV|_{\text{max}} + 2r_{\text{max}}\cos((\pi - \phi)/2) \). For example, the length of a diagonal line of a box is always less than the sum of the three edges which construct the box, and the capacity is almost triple that of using the method of a single level. If we only use the method of level 1, every bit needs to shift \( \lambda_{\text{max}} \) in the worst case and \( 3\lambda_{\text{max}} \) is needed for three bits of message, for we know that when \( \lambda_{\text{max}} \geq |EV|_{\text{max}} \) and \( \lambda_{\text{max}} \geq 2r_{\text{max}}\cos((\pi - \phi)/2) \), the MLEP will get less distortion, which is the radical of \( 3\lambda_{\text{max}}^2 \).

3.2. Information Extracting

Since all procedures of the algorithm are symmetrical, we easily extract the message using the method mentioned above if we have the help of the recovery key. The recovery model, created optionally in the last step, is desirable since 3D models are sometimes the result of measures or specifications for industrial processes.

4. Experimental Results

We implemented the proposed technique 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 2GHz processor and 256 MB memory. The analyses of our steganographic system are based on the following four goals: increased capacity, invisibility, security, and heightened performance.

Here, we give some of the results that we obtained. Model details, distortion, and processing time are detailed in Table 1. Since our scheme does not change the topology, we use an estimation of the RMS (Root Mean Square) and the \( S_{\text{dist}} \) we defined to evaluate the distortion induced by the embedding process. Table 1 demonstrates that our approach works well in triangle meshes. We exploit the advantage of 3D more: every vertex of a 3D polygonal model can be treated as a message vertex, and can be represented by at least three bits in 3D space, which means the filling rate of our method approximates 299.9%. Not only can every model have embedded messages in the upper bound of the capacity easily and soon, but we can also forecast and control the distortion simply by fine tuning those parameters. Although the MLEP used for embedding produces little distortion, we still give full control over these parameters to the user. Visual results of the cover and stego model of the horse model are shown in Figure 7, and stego models of other testing models are shown in Figure 8.

From the security point of view, finding the initial triangle, getting the sequence list over the mesh, and obtaining the \( p \) value of PRNS are the challenges for an attacker. Just by looking at these issues, it is clear that our scheme is secure in the cryptographic sense. It is resistant against exhaustive search. Retrieving the message without the key is virtually impossible. This problem is NP-hard with respect to the number of triangles in the mesh.

Figure 9 shows the advanced analyses of the relations between the message length and preprocessing time (time for generating jump variable stencils) and the message length and processing time (time for the MLEP procedure). Figure 9a shows the relation between message length and preprocessing time without being accelerated by our scheme, and Figure 9b shows the relation with acceleration by our scheme. It is clear that our accelerating scheme greatly improves performance. For example, to create the sequence list in the house model fully without any accelerating scheme needs 762,609 seconds, but to create the list with our accelerating scheme only needs 3,219 seconds.
seconds, an improvement of performance approaching 23690.87%! When the filling rate reaches around 50%, the time cost of creating the sequence list without acceleration soon increases, and exponentially grows when the filling rate reaches around 94%. Large retraced triangles happen with respect to the filling rate and exponentially grow when the filling rate approximates to the upper bound of the capacity (see Table 2). To create a sequence list with our accelerating scheme only needs a little setup time to construct a hierarchical kd-tree structure and create the TNT; this efficiently decreases the processing time in the overall preprocess. The AJS also achieves the goal of improved performance after a filling rate over 50%. Time cost grows very slowly even if the filling rate reaches around 94%; the overall time cost really shows a great decrease. In addition, our accelerating scheme only needs a little setup time when starting to embed a message; then, time cost grows slowly even if the filling rate reaches around 94%, for example, embedding a message in the model of bunny, where the filling rate is 87.7%. The method of Cayre and Macq [CM03] needs more than 120 seconds, but our scheme only needs 1.409 seconds; it improves performance more than 85 times (testing on an AMD Duron 800 MHz processor and 128 MB memory).

Obviously, a larger model would get larger improvement, the time cost grows very slowly even if the filling rate reaches around 94%; the overall time cost really shows a great decrease. In addition, our accelerating scheme only needs a little setup time at the beginning of the process when embedding the message. It is also worth exploring the characteristics of the HVS in order to decrease visual degradation. Finally, future research has to further explore and analyze the relationship between message length, visual effect, and the resulting robustness.

5. Conclusion and Future Work

In this paper we have presented an efficient digital steganographic technique for 3D triangle meshes. Our technique provides information hiding with efficiency, high capacity, security, low distortion, automaticity, reversibility, and robustness against affine transformations. Not only can every model have embedded messages in the upper bound of the capacity easily and soon, but we can also forecast and control the distortion simply by fine tuning those parameters. We have demonstrated the feasibility of our technique for steganographic applications.

The main difference with previous works such as [CM03] and [CDS*04] is that we exploit the advantage of 3D space more for larger capacity. We do so by relying on three independent degrees of freedom, which is based on our MLEP. We shift the message vertex with respect to the HVS, and we can easily forecast and control the distortion rate. This process naturally leads to higher capacity and lower distortion. Furthermore, we improve performance based on the hierarchical kd-tree, triangle neighbor table, and advanced jump strategy; this means that it is possible to carry more information by larger models, and it will make this kind of application more widely available. In addition, the secret key was used on three level embedding procedures for more security. Recovering messages without assistance of the secret key is really impossible.

Even though our approach already delivers good results, some further improvements are conceivable. The main limitation of this approach is machine precision errors when considering small triangles. Since the definitive capacity limit is reached when machine precision errors occur, one could use other approaches to divide reference edges or even use different approaches and rules to embed messages for increased capacity. In order to improve the performance of this approach, much faster search algorithms and much smarter stencil rules are needed to decrease processing time at the end of the process when reaching the capacity limit. Another kind of improvement is to investigate how to preserve the quality of polygon models when embedding the message. It is also worth exploring the characteristics of the HVS in order to decrease visual degradation. Finally, future research has to further explore and analyze the relationship between message length, visual effect, and the resulting robustness.

Acknowledgements

We would like to thank Mr. Monty Vierra for proof-reading the original manuscript and providing many valuable suggestions to clarify the paper presentation. We also appreciate anonymous reviewers for their valuable comments and, in particular, pointing out Cayre et al.’s recent work. This research was supported by the National Science Council (NSC, Taiwan) under the grant numbers NSC 93-2213-E-005-018 and NSC 92-2213-E-005-021.

References


Table 1: Results of various models. For testing, we chose 256 as the order, 0.001 as the DTR, 0.01 as the distortion, and 0.000005 as the accuracy. Every model embedded messages in their upper bound for the capacity. No errors were found in the recovered messages.

<table>
<thead>
<tr>
<th>cover model</th>
<th>vertices</th>
<th>triangles</th>
<th>message</th>
<th>embedded messages</th>
<th>distortion</th>
<th>time cost (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>rabbit</td>
<td>67038</td>
<td>134074</td>
<td>67036</td>
<td>201107</td>
<td>S-dist</td>
<td>3.512</td>
</tr>
<tr>
<td>dinosaur</td>
<td>56194</td>
<td>112384</td>
<td>56192</td>
<td>168575</td>
<td>0.000000849</td>
<td>5.593</td>
</tr>
<tr>
<td>horse</td>
<td>48485</td>
<td>96966</td>
<td>48483</td>
<td>145448</td>
<td>0.000003659</td>
<td>3.434</td>
</tr>
<tr>
<td>venus</td>
<td>33591</td>
<td>67178</td>
<td>33589</td>
<td>100766</td>
<td>0.000002406</td>
<td>4.313</td>
</tr>
<tr>
<td>knots</td>
<td>23232</td>
<td>46464</td>
<td>23230</td>
<td>69689</td>
<td>0.00003869</td>
<td>3.219</td>
</tr>
<tr>
<td>bunny</td>
<td>14007</td>
<td>27826</td>
<td>14005</td>
<td>42014</td>
<td>0.00007164</td>
<td>2.953</td>
</tr>
<tr>
<td>athena</td>
<td>7546</td>
<td>15015</td>
<td>7544</td>
<td>22631</td>
<td>0.00234325</td>
<td>2.563</td>
</tr>
<tr>
<td>elephant</td>
<td>4571</td>
<td>9084</td>
<td>4569</td>
<td>13706</td>
<td>0.00077437</td>
<td>1.391</td>
</tr>
<tr>
<td>dragon</td>
<td>1257</td>
<td>2730</td>
<td>1255</td>
<td>3764</td>
<td>0.00017961</td>
<td>0.747</td>
</tr>
<tr>
<td>hand</td>
<td>1055</td>
<td>2130</td>
<td>1053</td>
<td>3158</td>
<td>0.00482970</td>
<td>0.094</td>
</tr>
</tbody>
</table>

Table 2: Results of improving performance. Every model embedded messages in their upper bound for the capacity.

<table>
<thead>
<tr>
<th>cover model</th>
<th>vertices</th>
<th>triangles</th>
<th>no accelerating strategy</th>
<th>with accelerating strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>rabbit</td>
<td>67038</td>
<td>134074</td>
<td>1981238</td>
<td>1393097</td>
</tr>
<tr>
<td>dinosaur</td>
<td>56194</td>
<td>112384</td>
<td>2344193</td>
<td>962337</td>
</tr>
<tr>
<td>horse</td>
<td>48485</td>
<td>96966</td>
<td>1684985</td>
<td>919834</td>
</tr>
<tr>
<td>venus</td>
<td>33591</td>
<td>67178</td>
<td>1456091</td>
<td>748644</td>
</tr>
<tr>
<td>bunny</td>
<td>14007</td>
<td>27826</td>
<td>635602</td>
<td>243156</td>
</tr>
</tbody>
</table>
Figure 7: The cover and stego model of the horse model (vertex: 48485; face: 96966). The stego model is quite good, since the method of MLEP used for embedding results in little distortion.

Figure 8: From top left to bottom right, stego models are listed in the following order: rabbit, dinosaur, venus, knots, bunny, athena, elephant, dragon and hand.
Figure 9: The analyses of the relations between fill rate and preprocessing time (time for generating sequence list) and message length and processing time (time for MLEP method).