An Improved TIN Compression Using Delaunay Triangulation

Yang-Soo Kim, Dong-Gyu Park, Ho-youl Jung, Hwan-Gue Cho
Department of Computer Science, Pusan Nation University
Jang-Jun Dong, Keum-Jung Ku, Pusan, 609-735, Korea
e-mail: {yskim,dongupak,hyjung,hgcho}@pearl.cs.pusan.ac.kr

Abstract

In this paper we introduce a new compression technique for the connectivity of a Triangulated Irregular Network (TIN) by using Delaunay triangulation. The strategy in this paper is based on that most of 2.5 dimensional terrain connection are very similar to plane-projected Delaunay triangulation of their vertex set. We only need to keep the vertex coordinates and a few edges that are not included in the Delaunay edges. We present an efficient encoding method for the set of edges using vertex reordering and general bracketing algorithm. Also we propose progressive visualization method for web-based GIS.

In the experiments, the method was examined on several TIN data with various resolutions, which were generated by five typical terrain simplification algorithms. The connectivity information of common terrain data could be compressed by 0.23 bits per vertex.

1 Introduction

Modern Geographic Information System (GIS) requires a huge size of terrain data. Many problems are arising in storage, transmission, and visualization of terrain models. One of the objectives in managing GIS data is attempting to manipulate the terrain data easily and efficiently. We also need a special compression technique for this terrain data, because the transmitting terrain data is crucial in the web-based GIS.

A terrain is a 2-dimensional surface in 3-dimensional space with a special property: every vertical line intersects a plane in a point. In other words, it is the graph of a function \( f : D \subseteq \mathbb{R}^2 \rightarrow \mathbb{R} \) that assigns a height \( f(p) \) to every point \( p \) in the domain, \( D \), of the terrain. So its projection on a plane is a planar subdivision whose bounded faces are triangles.

For rendering purposes, it is convenient to model a terrain as a collection of disjoint triangles. Such a representation is called a Triangulated Irregular Network (TIN) in GIS. We present a new compression algorithm for triangulated terrain model by using Delaunay triangulation.

2 Previous Work

In order to store, transmit, and give a multiresolution TIN files, the files need to be kept in a compressed form. So far, there have been several compression techniques for planar graph model, especially for planar triangulated graph structure. In this section, the previous compression techniques are reviewed.

General Mesh Compression

Compression for the planar graph and 3D geometry was introduced by Turan [10] and Keeler [6]. Turan has shown that a planar graph can be encoded in at most \( 12n \) bits, where \( n \) is the number of vertices in the graph [10]. Keeler introduced space-efficient encoding schemes for planar graphs and maps [6]. He proved that an arbitrary planar graph \( G \) with \( m \) edges can be encoded in \( m \log_2 \theta + O(1) \) bits. Deering has proposed a generalized triangle mesh (GTM) representation, which consists of triangle strips and mesh buffers to minimize a vertex revisit [2]. Java 3D API uses Deering’s GTM method for geometry compression. This algorithm requires \( (1/8 \log_2(n)) + 8n \) bits to represent the connectivity information for a given triangle mesh [2, 5]. Chow improved the performance of GTM using some heuristics [1].

IBM Algorithm

Taubin and Rossignac proposed an efficient method for compressing geometry connectivity [8]. Their method decomposes a mesh into a set of spanning trees of triangles and vertices. These spanning trees are encoded separately so that both connectivity and vertex position are compressed easily. They were able to reduce connectivity information to 2 bits per triangle.

Touma’s Mesh Compression

Touma has suggested a new triangle mesh compression
method by exploiting the triangle marching structure [9]. He used a vertex connectivity code for compressing triangulation. The connectivity codes consists of “add”, “split” and “merge”. He encoded mesh connectivity as a list of vertex degrees in a special order, and this procedure requires a vertex stack (called active list) to make final triangles, where the edge connectivity is encoded by 1.5 bits per vertex.

Gumhold’s Method
Gumhold has presented a local compression and decompression algorithm, which is fast enough for real time applications [12]. They used cut-border data structure for compression and decompression. The connectivity is encoded at about 1.7 bits per vertex and requires $O(\sqrt{n})$ memory during compression and decompression.

Table 1 shows the comparison of performances of each compression algorithm.

<table>
<thead>
<tr>
<th>Method</th>
<th>bits/vertex</th>
<th>compression algorithm</th>
<th>progressive transmission</th>
<th>vertex compression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deering [2]</td>
<td>$1/8 \log n + 8$</td>
<td>GTM</td>
<td>not support</td>
<td>delta encoding</td>
</tr>
<tr>
<td>Taubin [8]</td>
<td>4</td>
<td>topological surgery</td>
<td>not support</td>
<td>linear prediction</td>
</tr>
<tr>
<td>Hoppe [5]</td>
<td>$\log n + 5$</td>
<td>progressive mesh</td>
<td>support</td>
<td>–</td>
</tr>
<tr>
<td>Chow [1]</td>
<td>$1/8 \log n + 8$</td>
<td>GTM</td>
<td>not support</td>
<td>variable quantization</td>
</tr>
<tr>
<td>Touma [9]</td>
<td>1.5</td>
<td>active list</td>
<td>not support</td>
<td>parallelogram prediction</td>
</tr>
<tr>
<td>Gumhold [12]</td>
<td>1.7</td>
<td>cut-border</td>
<td>not support</td>
<td>–</td>
</tr>
</tbody>
</table>

**Table 1. Performance comparison of compression algorithms.**

3 Delaunay Compression

In our previous work, we proposed a new geometric compression scheme for TIN data using Delaunay triangulation [13]. We define $T_0$ and $V_o$ as original terrain model and its vertex set. $T_p$ is defined as the plane projected terrain model of $T_o$ and $V_p$ is defined as the vertex set of $T_p$. Kim’s study showed that all triangles in $T_p$ are very similar to the Delaunay triangulation of $V_p$ [13].

In this paper we restrict input geometric models as to the triangulated terrain model, especially, 2.5 dimensional terrain data. It is not necessary to encode all triangles in $T_p$, because most triangles can be reconstructed by making Delaunay triangulation with $V_p$. We only need to store different parts of Delaunay triangulation and $T_p$ to recover the original data.

In this section, we introduce Implicit Delaunay Triangulation and propose an optimized compressing method, and we explain the general bracketing method.

3.1 Implicit Triangulation

In the following, the faces of $DT(P)$ are called as Delaunay triangles of the point set $P$. Various algorithms to construct $DT(P)$ have been invented to meet several applications. There are two major trends in completing $DT(P)$: divide-and-conquer and incremental method. Both can construct a $DT(P)$ in $O(n \log n)$ time where the number of points is $n$.

The terrain model $G$ is denoted by $T(V, E)$, where $V$ is a vertex set, $E$ is an edge set, and $T$ is a triangulation. Let an original triangulation be $T_o(V, E_o)$ and the Delaunay triangulation of $G$ be $T_d(V, E_d)$. Then, $T_o$ and $T_d$ are the two different triangulations of the same point set $P$. We define exclusive-OR operation ‘$\oplus$’ for two triangulations, that computes the intersecting graph, $\Delta T$, between $T_o(V, E_o)$ and $T_d(V, E_d)$. This operation is represented as follows:

$$\Delta T = T_o \oplus T_d, \text{where }$$

$$V = \{v_i \mid (v_i, v_j) \in E_o\}$$

$$\Delta E = \{(v_i, v_j) \mid (v_i, v_j) \in E_o, (v_i, v_j) \notin E_d\}$$

The Implicit Delaunay Triangulation, $IDT(V, \Delta E)$, is one of representation scheme, which is composed of the vertex coordinate set $V$, and the different edge set $\Delta E$. Consequently, $IDT(V, \Delta E)$ should contain vertices and a small amount of connectivity information. The rule of Delaunay triangulation is implicitly contained after the model is encoded.

3.2 $\Delta E$ Compressing

$\Delta T$ is composed of the vertex set $V$ and the edge set $\Delta E$ which is not included in the Delaunay triangulation. Fig. 2(d) shows the example of $\Delta T$. In this section, we propose a method to efficiently encode $\Delta E$.

All the edges in $\Delta E$ can be classified into three categories: single edge, tree, and graph. The graph $E_g$ consists of a set of vertices and a set of edges joining these vertices. $E_g$ has cycle edges. The tree $E_t$ is a connected graph
that has no cycle edges. The single edge \( E_s \) is an isolated edge that consists of only two vertices. All the \( E_s \) are encoded by reordering the vertices of \( E_s \) at the head of the vertex sequence and adding the number of \( E_s \). The tree information of \( E_t \) can be encoded by 2 bits per vertex using general bracketing method. We decompose the tree information into a set of brackets and a vertex sequence. The encoding results of \( E_t \) are a bitstream (set of brackets) and reordered vertices. To encode \( E_g \), the cycle edges are removed from \( E_g \). The number and the vertex indices of the cycle edges are added to the result. The \( E_g \) where cycle edges are removed can be also encoded by using general bracketing method. Additional indices for cycle edges are needed (log \( n \) bits) but cycle edges seldom appear in \( \Delta E \). The algorithm for compressing \( \Delta E \) is as follows:

**Algorithm**: EncodeDeltaE

**Input**: \( \Delta T(V_r, E_c) \)

**Output**: \( IDT_r(V_r, E_t) \)

1. Classify edges of \( \Delta E \);
2. Reorder vertices of single edge \( E_s \);
   - Add the number of \( E_s \);
3. Remove cycle edge from graph \( E_g \);
   - Add the number of cycle edges;
   - Add indices of cycle edges;
   - Encode tree \( (E_g - \text{cycle edge}) \) using the general bracketing method;
4. Encode tree \( E_t \) using the general bracketing method;

Fig. 1 shows the overview of \( IDT_r(V_r, E_t) \). Vertices are reordered to encode \( E_s \), \( E_t \), and \( E_g \) which are shown in (a). Fig. 1(b) shows the results of compressing \( \Delta E \). Each information is needed for encoding \( E_s \), \( E_g \), and \( E_t \).

Fig. 2 shows the example of the compressing \( \Delta T \). Fig. 2(a) shows \( T_o \) that is the original terrain and Fig. 2(b) shows the Delaunay triangular mesh obtained from \( T_o \). Fig. 2(d) shows \( \Delta T \) that is composed of the vertices and \( \Delta E \) edges. The edges in \( \Delta E \) are classified, \( a \) and \( b \) are a single edge respectively, \( c \) is a graph, and \( d \) is a tree. The results of encoding single edges, \( a \) and \( b \), are the reordered vertices, “1 7 4 9”, and the number of single edges, 2 in the bitstream. To encode graph \( c \), the cycle edge \( e_c \) was removed from the graph \( c \). The number of cycle edges, 1, and indices of cycle edges, “6 7”, are added on the bitstream. These indices, “6 7”, are not the indices of the original vertex list \( V_o \) but the indices of reordered vertex list \( V_r \), “14 16”. After removing \( e_c \), the graph \( c \) is reconstructed as a tree structure, so \( c \) can be encoded by using the general bracketing method. The results of encoding \( c \) are the vertices, “8 14 16 11”, and the sequence of brackets, “0 0 0 1 1 0 1 1”. The tree \( d \) is encoded, and the results are “19 23 20” and “0 0 0 1 1 1”. These results of compressing \( \Delta T \) are shown in Fig. 2(e). Most of \( \Delta E \) edges can be encoded with the reordered vertices and bitstream (0 or 1), but the cycle edges in \( E_g \) need their indices (log \( n \) bits). So, the cycle edges have a significant influence on the compression ratio.

Table 2 shows the number of edge types in \( \Delta E \). The cycle edges are less than 0.6% of \( \Delta E \) in this table. Because the ratio of cycle edges is very small, they have little influence on the compression ratio.

<table>
<thead>
<tr>
<th>edges</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>( [\Delta E] )</td>
<td>8,837 10,847 2,699 8,748 2,798</td>
</tr>
<tr>
<td>( #E_s )</td>
<td>3,649 3,782 1,881 3,580 1,778</td>
</tr>
<tr>
<td>( #(E_g + E_c) )</td>
<td>5,188 7,065 818 5,168 1,020</td>
</tr>
<tr>
<td>#cycle edge</td>
<td>44 64 2 45 6</td>
</tr>
</tbody>
</table>

Table 2. The number of edges in \( \Delta E \) for testing TIN files which the number of vertices are 50,000 and the number of total edges are about 150,000.

### 3.3 General Bracketing Method

Fig. 3 illustrates how \( E_t \) is encoded from the original tree. Each vertex of \( E_t \) can be represented as a pair of brackets. The bracket on the left is denoted as 0 and the bracket on the right as 1. The tree information is decomposed into a set of brackets; it also means that the connectivity information can be reconstructed with a set of brackets. All the vertices of \( E_t \) are reordered with the tree hierarchy, namely, level ordering.
Figure 2. An example of compressing terrain data: (a) $T_o$ is an original terrain, (b) $T_d$ is the Delaunay triangular mesh obtained from $T_o$, (c) $T_o \oplus T_d$: dashed edges are included in $T_d$, but not in $T_o$, (d) $\Delta T$ is composed of $V$ and $\Delta E$, (e) The compression results are reordered vertices and bitstream of $\Delta E$. 

<table>
<thead>
<tr>
<th>Vertices:</th>
<th>Bitstream:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 7 4 9</td>
<td>2</td>
</tr>
<tr>
<td>8 14 16 11 19 23 20</td>
<td>1</td>
</tr>
<tr>
<td>2 3 5 6 10 12 13 15 17</td>
<td>6 7</td>
</tr>
<tr>
<td>18 21 22 24 25</td>
<td>0 0 0 1 1 0 1 1</td>
</tr>
<tr>
<td>// vertices of $E_s$</td>
<td>// number of single edges</td>
</tr>
<tr>
<td>// vertices of $E_g$ and $E_t$</td>
<td>// number of cycle edges</td>
</tr>
<tr>
<td>// other vertices</td>
<td>// indexes of cycle edge</td>
</tr>
<tr>
<td></td>
<td>0 0 0 1 1 1</td>
</tr>
<tr>
<td></td>
<td>// sequence of brackets</td>
</tr>
</tbody>
</table>

(c) Compression results
The results of encoding $E_t$ are a set of brackets (bitstream of 0 and 1), and the reordered vertices. Therefore, the connectivity of $E_t$ can be encoded with 2 bits per vertex.

### 3.4 Delaunay Decompression

The compressed results of our algorithm are a set of vertices and encoded $\Delta E$ edges. The encoded $\Delta E$ edges are the reordered vertices and bitstream of the number of single edges, the number of cycle edges, the indices of cycle edges and the sequence of brackets (see Fig. 2). At first, the $\Delta E$ edges should be reconstructed from the encoded bitstream. The $\Delta E$ edges are the constrained edges of the Delaunay triangulation. Therefore, the original terrain can be reconstructed from the constrained edges and the vertex sequence using the constrained incremental Delaunay triangulation [14, 15].

### 4 Progressive Transmission

The web-based terrain data transmission is also an important issue on GIS area. When the terrain data is transmitted over a communication line, one would like to show progressively better approximations to the model as data is incrementally received. To archive progressive transmission, we implement the vertex evaluation function, the vertex reordering function, the pipeline transmission structure between a client and a server, and Constrained Incremental Delaunay Triangulation (CIDT).

If all the vertices in the original mesh are sorted by their evaluation criteria and reordered, simple Incremental Delaunay Triangulation (IDT) representation is enough to archive the progressive transmission. The geometry server will transmit the vertices one by one, and the client will reconstruct the triangle using IDT rule. The algorithm, however, contains the $\Delta E$ edges to reconstruct the original triangle. The $\Delta E$ edges are the constraint edges of the triangle mesh. Because the average number of $\Delta E$ edges are about 10% of the original mesh, the vertices in $\Delta E$ edges are more than 40% of the total vertices.

We used Schroeder's evaluation function for approximation. The evaluation function uses a distance to a vertex and its average plane. The average plane is constructed using the triangle normals, centers and areas. Furthermore, we used single edge evaluation function using a distance to the center of the edge and its average plane. Fig. 4 shows the snapshot of the average plane and single edge.

If $p_i$ and $p_j$ are the vertices in the single edge, the average plane surrounding single edge and the middle point of $p_i$ and $p_j$ can be obtained. The single edges are sorted and reordered by the distance between the single edge and average planes. Also, vertices, not included in $\Delta E$, are sorted and reordered by the evaluation criteria.

Fig. 5 and Fig. 6 are the progressive visualization of Crater Lake and Ashby. The visualization of the terrain is implemented by Java 3D API and progressive transmission with RMI (Remote Method Invocation). Fig. (a) in each figure is the transmission result of the single edge $E_s$. Fig. (b) in each figure is the transmission result of the $\Delta E$ edge. Fig. (c) in each figure is the transmission result of the $\Delta E$ edge and other reordered vertices. Fig. (d) is the final transmission of total triangle mesh.

### 5 Experiments

#### 5.1 Similarity between DT and Real TIN

We have prepared five general data sets for the experiments. We tested our algorithm on data sets based on 1:250,000 scale digital elevation models obtained from U.S. Geological Survey.
The experiment shows the similarity between the original triangulation and 2D Delaunay triangulation. The degree of similarity is defined as follows:

$$S(A, B) = 1 - \frac{n|\Delta E_{(A,B)}|}{n|E_A|}, \quad (0 \leq S(A, B) \leq 1) \quad (2)$$

where A and B are terrain model, $n|\Delta E_{(A,B)}|$ is the number of delta edges between A and B, $n|E_A|$ is the number of edges in A. The similarity of multiresolution models was measured according to the simplification methods, such as progressive meshes, vertex decimation, quadric error metrics, greedy insertion and degree method [5, 7, 4, 3, 11].

Table 3 and 4 show that the similarity ratio is quite high for all testing data, that is more than 93%.

### 5.2 Compression Performance

Table 5 shows the performance of our compression algorithm on each terrain model. Each test model has 50,000 vertices (nearly 100,000 faces). The results show that the connectivity information can be encoded by 0.23 bits per vertex and 0.12 bits per face, on average. Comparing to Touma, Gumhold and other known algorithms [9, 12, 1, 5], the compression ratio outperforms the other compression algorithms.

The terrains are optimized for the rendering efficiency and frequently replaced by approximated terrains. We also measured the compression ratio of various resolution models that are simplified by the five simplification methods.
Table 4. Similarity ratio to Degree Method.

The test models are simplified as 10, 30, 50, 70, and 90% by each simplification method. Table 6 and 7 show the compression ratio of the simplification results. The experimental results show that the compression algorithm provides the outstanding compression ratio.

The similarity ratio of the models depends on the number of faces. As faces decrease, the similarity ratios decrease, as shown in the Table 3 and 4. Consequently, the compression ratio also decreases, but efficient compression ratio is still maintained within 1.3 bits per vertex. Fig. 7 shows the average compression ratio of our compression algorithm for the terrain model on its simplification level. The compression ratio of the graph is obtained from the average ratio of five test data. Fig. 7 shows that our compression algorithm can be applied to simplified models efficiently.

<table>
<thead>
<tr>
<th>model</th>
<th>vertex</th>
<th>face</th>
<th>conn. (bits)</th>
<th>bits/vert.</th>
<th>bits/face</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ashby</td>
<td>50,000</td>
<td>99,468</td>
<td>15,728</td>
<td>0.31</td>
<td>0.16</td>
</tr>
<tr>
<td>Crater</td>
<td>50,000</td>
<td>99,472</td>
<td>21,118</td>
<td>0.42</td>
<td>0.21</td>
</tr>
<tr>
<td>Ntc</td>
<td>50,000</td>
<td>99,550</td>
<td>2,428</td>
<td>0.05</td>
<td>0.02</td>
</tr>
<tr>
<td>Ozark</td>
<td>50,000</td>
<td>99,468</td>
<td>15,696</td>
<td>0.31</td>
<td>0.16</td>
</tr>
<tr>
<td>Spokane</td>
<td>50,000</td>
<td>99,610</td>
<td>3,098</td>
<td>0.06</td>
<td>0.03</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td>0.23</td>
<td>0.12</td>
</tr>
</tbody>
</table>

Table 5. Experimental results of optimized Delaunay compression.

6 Conclusions

Many GIS and graphics application programs are developed to handle the large dataset. We proposed a new algorithm for compressing/decompressing large TIN models.

Our compression algorithm is based on that 2.5D TIN data are quite similar to the Delaunay triangulation of the same point set. The experiment proved that more than 90% of edges are common in DT and arbitrary TIN data. This implies that only 10% of edges (as we defined to \( \Delta E \) edges) are needed to restore original TIN from the arbitrary point set. Original TIN was decomposed from the reordered point set and the \( \Delta E \) edges. The \( \Delta E \) edges can be encoded by the general bracketing algorithm and vertex reordering.

Several test data are obtained from USGS and simplified by 10 steps using 5 simplification rules. Each experimental result shows that our algorithm maintains the good compression ratio. The average compression ratio is 0.23 bits per vertex, which is quit competitive results compared to the existing TIN compression algorithm.

The decompressing algorithm depends on the performance of the Delaunay triangulation algorithm only. The time complexity of Delaunay triangulation is \( O(n \log n) \), where \( n \) denotes the number of vertices.

There are several areas for future works, including:

1. Application of geometry compression to 3D polyhedral object compression.
2. Compressing the vertex coordinate.
Figure 7. Compression ratio of the terrain models on its simplification level.

References


