A GEOMETRIC COMPRESSION ALGORITHM FOR MASSIVE TERRAIN DATA USING DELAUNAY TRIANGULATION *

Sung-Soo Kim, Yang-Soo Kim, Mi-Gyung Cho, Hwan-Gue Cho

Graphics Application Laboratory,
Department of Computer Science,
Pusan National University,
Kum-Jung-Ku, Pusan 609-735, Korea.
E-mail: {sskim, yskim, mgcho, hgcho}@pearl.cs.pusan.ac.kr

ABSTRACT

In this paper we introduce a new compression technique for a large triangulated terrain using Delaunay triangulation. Our compression technique decomposes a triangulated mesh into two parts. One is a point set whose connecting structure is defined implicitly by Delaunay edges. The other is the set of edges which cannot be recovered by the implicit Delaunay triangulation rule. Thus we only need to keep the whole vertex coordinates and a few edges which is not included in the Delaunay edges. For the vertex coordinate, we apply "entropy coding" given by [Costa98], and we store only the edges not included in Delaunay triangulation.

In experiments, we prepared several TIN data set with various resolutions, which were generated by five typical algorithms for terrain simplification. Those algorithms include progressive meshing, vertex decimation and incremental greedy insertion etc. We found that most of terrain triangulations are quite similar(= nearly 93%) to the plane-projected Delaunay triangular mesh. This experimental work shows that more than 93% edges of a common terrain data is included in Delaunay triangulation. By exploiting this result, we can compress the common terrain data by 1.2 bits per vertex. Another advantage of our Delaunay compression approach is that we can reconstruct the original terrain structure locally, since we can easily determine if an edge is included in Delaunay edges by observing only a few local surrounding vertices.

Keywords: Data Compression, Delaunay Triangulation, Terrain Modeling

1 INTRODUCTION

One of the main problems in GIS is to visualize and analyze the terrain surface, e.g., finding the geodesic shortest path, hidden surface removal, presenting a selectively refinement view and constructing a virtual environment. One of objectives in handling GIS data is how to handle the terrain data easily and efficiently since a common terrain file is very large. We also need a special compression technique for this GIS terrain data file since the transmitting terrain data is crucial in the Web-based GIS. Terrain models are usually specified with *triangle-based* surface meshes, and nowadays a key issue is how to store, access and visualize possibly in real time hundreds of thousands of faces[HSurvey97].

Much of the current research has been focused on managing these large datasets. Two well known approaches are *surface simplification* and *geometry compression*. The former allows an object to be viewed at different details depending on viewing distance from the object. These techniques reduce the number of vertices in the mesh by altering the model's connectivity and by the possibly adjusting the position of the remaining vertices to minimize the error produced by the simplification[HSurvey97]. The latter is a general space-time trade-off, and offers advantages at every level of the memory/interconnect hierarchy:

*This work was supported by KOREA Research Foundation(1998.)

less storage space is needed on disk, less transmission time to transmit over the network.
A *terrain* is the graph of a continuous function that assigns every point on the plane to an elevation. 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 geographic information systems and a *polyhedral terrain* in computational geometry.

In this paper we focus on *space efficiency* for multiresolution model. We present a new compressed representation of triangulated terrain model using Delaunay triangulation.

2 RELATED WORKS

2.1 Terrain Simplification Algorithms

The goal of terrain simplification–or more generally: surface simplification– is to represent a surface with fewer vertices, but to keep a good approximation of the original surface. The advantages of working with a simplified version include reduced storage space, shorter times to construct the data structures, and faster visualization of the surfaces.

Vertex Decimation

Schröder has proposed the vertex decimation to simplify complex triangulations[Schroe92]. This method takes a triangulated surface as input, typically a manifold with boundary. The algorithm makes multiple passes over the data until the desired error is achieved. On each pass, all vertices that are not on a boundary that has error below the threshold are deleted, and their surrounding polygons are retriangulated. The error at a vertex is the distance from the point to the approximating plane of the surrounding vertices. This method is fast, but it does not assure bounded approximation [HSurvey97].

Progressive Meshes

Hoppe introduced a new multiresolution model called *Progressive Meshes*(PM)[HH96]. The progressive meshes are built from a simplification algorithm that is based on a single operation, the *edge collapse*. The inverse of edge collapse operation is the *vertex split* operation. The PM scheme is composed of a coarse low resolution mesh M_0, together with the sequence of refinement records obtained by inverting the simplification steps operated to build M_k from the input mesh M_0. These refinement records allow the incremental refinement of M_k into a mesh M_i at whatever precision, with interactive times. During simplification, the selection of the edges to be collapsed is done via a priority queue, ordered by the energy cost improvement estimated for each edge collapse[HSurvey97].

Simplification Using Quadric Error Metrics

The surface simplification using quadric error metrics method is based on the iterative contraction of edge pairs, which also allows to join unconnected regions of the model. The atomic operation, *vertex pairs contraction*, may be conceived as a less general vertex clustering operation[Heck97]. As simplification proceeds, a geometric error approximation is maintained at each vertex of the current model. The error approximation is represented using quadric metrics, extending a previous approach[Garland95]. The main advantages of this approach are the computational efficiency and generality. It allows the simplification of disconnected or non-manifold meshes.

Greedy Insertion Method

Heckbert has proposed the *greedy insertion* algorithm to refine a coarse triangulation[Garland95]. This work has explored fast and accurate variations of the greedy insertion algorithm. This method uses two optimizations of the most basic greedy insertion algorithm. First, they exploited the locality of mesh changes and only recalculated the errors at input points for which the approximation changed. Second, they used a heap to permit the point of highest error to be found more quickly. *SCAPE* software was implemented by this algorithm.

2.2 Geometric Compression

In order to store, transmit and give a multiresolution TIN files, we need to keep them in a compressed form. So far, there are a few compression techniques for planar graph model, especially for planar triangulated graph structure. In this section, we review previous compression techniques.

General Mesh Compression

Compression for planar graph and 3D geometry was introduced by [Turan84, Kenn95]. 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[Turan84]. Keeler introduced space-efficient encoding schemes for planar graphs and maps[Kenn95]. He has proved that an arbitrary planar graph G with m edges can be encoded in $m\log 12 + O(1)$ bits.

Deering has proposed a *generalized triangle mesh* representation[Deer95]. First, 3D geometry is represented as a *generalized triangle mesh*, a data structure that encodes both position and connectivity of the geometry efficiently. Next, the positions, normals, and colors are quantized to less than 16 bits/vertex. Chow has also described some heuristic algorithms which generate codes where each vertex of a typical mesh appears only 1.3 times on the average[Mchow97].

IBM Mesh Compression

Extending Deering's main idea, Taubin and Rossignac gave an efficient method for compressing geometry connectivity[Taubin96]. Their method decomposes a mesh into a set of spanning tree of triangles and vertices. These 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. However, one disadvantage of this method is that the decompression stage is complicated by large memory requirements and is not amenable to cost-effective hardware implementations[Mchow97].

Costa's Mesh Compression

Costa has announced a new triangle mesh compression method by exploiting the triangle marching structure[Costa98]. He uses a more sophisticated prediction scheme based on local surface properties, requiring only approximately 9 bits per vertex for typical inputs. Mesh connectivity encoding is performed using active list which is a list of vertex degrees in a special order. To compress the vertex coordinate data, they use the "parallelogram" rule for geometric prediction.
The main disadvantage is that compressed models do not support selective refinement as necessary for view-dependent visualization.

3 DELAUNAY RULE-BASED COMPRESSION

In this section, we propose a new geometric data compression algorithm for TIN data using Delaunay triangulation. The basic idea of our algorithm is that we encode the TIN topology with a "rule", so TIN is decomposed into two parts; a rule(=Delaunay Triangulation) and elementary data(= vertex coordinates and some different edges).

3.1 Implicit Delaunay Triangulation

A Delaunay triangulation is desirable for approximation because of its general property that most of its triangles are nearly equiangular. Note that this generates a unique triangulation for a given set of points P. And the faces of $DT(P)$ are called *Delaunay triangles*. Various algorithms to construct DT are invented because of its many applications. Prevailed algorithms are divide-and-conquer and incremental method. Both can construct DT in $O(n\log n)$ time when the number of points is n.

Most of current terrain data format is ASCII-based which contains a large amount of redundancy. It contains vertex coordinates and connectivity informations(triangles). Connectivity encoding techniques attempt to reduce the redundancy inherent to many popular representations of triangular meshes.

In this paper we restrict input geometric models as triangulated terrain model, especially, 2.5D terrain data. Our basic assumption is that most terrain connectivity of typical TINs is quite similar to the corresponding Delaunay triangulation of the given point set.

A terrain model G is denoted by $T_i(V, E_i)$, where V is a vertex set, E_i is an edge set and T_i is triangulation. Let an original triangulation be $T_o(V, E_o)$ and Delaunay triangulation of G be $T_d(V, E_d)$. Then, T_o and T_d are two different triangulations of the same point set P. We define exclusive-OR operation '$\oplus$' for two triangulations which computes the intersecting graph, ΔT, between $T_o(V, E_o)$ and $T_d(V, E_d)$. This operation is represented in the following:

$$T_o \oplus T_d = \Delta T(V, \Delta E), where \qquad (1)$$

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

An Implicit Delaunay Triangulation, $IDT(V, \Delta E)$, is composed of vertex coordinate set V, and different edge set ΔE. Our compression algorithm exploits Lemma 1.

Lemma 1 *Let T_o and T_d be two different triangulations of the original terrain.(T_o : original triangulation, T_d : 2D Delaunay triangulation). Let a new graph ΔT be defined $\Delta T(V, \Delta E) = T_o \oplus T_d$. Then we can reconstruct the original triangulation T_o by the following equation.*

$$T_o = T_d \oplus \Delta T \qquad (2)$$

Proof : Assume that T_o and T_d is an original triangulation and the corresponding 2D Delaunay triangulation, respectively. By definition, we know that $T_o \oplus T_d = \Delta T(V, \Delta E)$. So by associative property of exclusive-OR, we apply '$\oplus T_d$' to the both sides of $T_o = T_d \oplus \Delta T$.

$$T_o \oplus T_d \oplus T_d = \Delta T \oplus T_d.$$

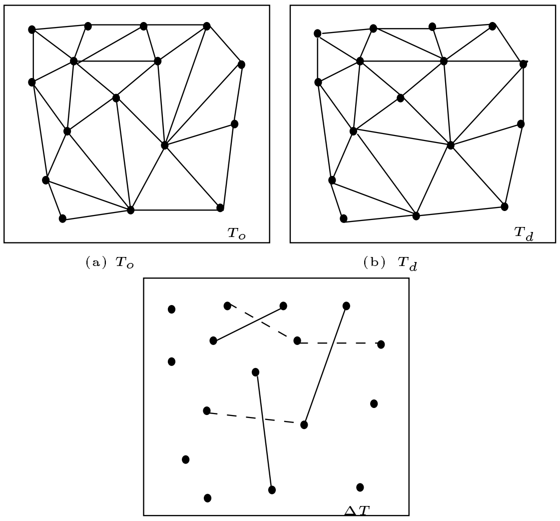
Since $T_d \oplus T_d = \phi$, so we have

$$T_o \oplus \phi = \Delta T \oplus T_d$$

Finally, we got $T_o = \Delta T \oplus T_d$. $\square$

Consequently, IDT should contain vertices and a small amount of connectivity(ΔT). The 2D DT is contained implicitly after model is encoded.

Fig. 1 shows one example of ΔT resulted from equation (1). The algorithm for computing ΔT goes follows:



(a) T_o (b) T_d

(c) $\Delta T(V, \Delta E)$

Figure 1: An example of $\Delta T(V, \Delta E)$: (a) T_o (original terrain), (b) T_d(delaunay triangulation), (c) $T_o \oplus T_d = \Delta T(V, \Delta E)$: solid and dashed edges are included in ΔE.

Algorithm ComputeDeltaT
Input : $T_o(V, E_o)$ - Original triangulated terrain.
Output : $\Delta T(V, \Delta E) = T_o \oplus T_d$

1. Create new TMesh $T_d(V, E_d)$;
   Create new EdgeList ΔT;
   $T_d := \phi$; $\Delta T := \phi$;
   $T_d :=$ Delaunay2D(T_o.VertexList);
   /* perform Delaunay triangulation of T_o */
2. Sort $e_i \in E_o$ lexicographically
   and put them in List L_o;
   Sort $e_j \in E_d$ lexicographically
   and put them in List L_d;
3. Compare two Lists L_o and L_d
   by head merging algorithm;
      if ($e_i \in E_d$ and $e_i \notin E_o$)
         then $\Delta T := \Delta T \cup e_i$;
      if ($e_i \notin E_d$ and $e_i \in E_o$)
         then $\Delta T := \Delta T \cup e_i$;

In above algorithm "TMesh" means an abstract data structure for any triangulated mesh and "EdgeList" means an edge list in triangulation. Let us consider the time complexity of above algorithm. Suppose that $|V| = n$, $|E_d| = 3n$. Delaunay2D function can be done in $\Theta(n\log n)$ by divide and conquer method or simply $O(n\log n)$ in average if we use a randomized incremental algorithm. Then in order to compute the difference between E_o and E_d, we make L_o and L_d in $O(n\log n)$ time. And we compare L_o to L_d by merging them, which can be done in $O(n)$. So the total time complexity of the algorithm for computing ΔT is $O(n\log n)$.

3.2 Delaunay Compression Algorithm

Our compression algorithm is stated as follows:

Algorithm Delaunay Encoding
Input : Original Triangulation $T_o(V, E_o)$
Output : $IDT_c(V_c, E_c)$

1. Delaunay2D(T_o.VertexList);
2. Compute $\Delta T(V, \Delta E) = T_o \oplus T_d$;
3. V_c := Compress V using Costa's method;
4. E_c := Compress ΔE using IBM algorithm;
5. return $IDT_c(V_c, E_c)$;

Vertex coordinates of an original terrain model are compressed by Costa's vertex coordinate compression method[Costa98] and edge information is compressed by IBM algorithm[Taubin96]. Costa's compression method encodes the vertex coordinate in 9 bits per vertex on average and IBM algorithm encodes the edge information in 4 bits per vertex on average. Finally, compressed terrain model should contain the compressed point set(V_c) and the compressed edge information(E_c) instead of all mesh connectivity. Fig. 2 shows an overview of our compression scheme.

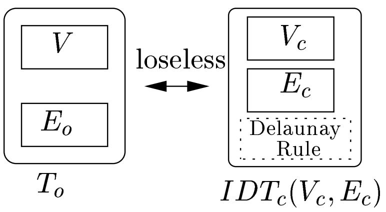


Figure 2: Overview of our compression scheme ; T_o denotes an original triangulated terrain and $IDT_c(V_c, E_c)$ denotes a compressed model representation.

3.3 Delaunay Decompression Algorithm

In this section we introduce a new subgraph structure for decompressing $IDT_c(V_c, E_c)$. Let us define a Smallest Containing Subgraph(SCS) $SCS(e_i)$ as a smallest subgraph containing e_i. A *Smallest Containing Subgraph* is a shortest cycle in T_d which surrounds the edge $e(x,y)$ intersected edge e in T_o. Fig. 3 shows a smallest containing subgraph containing $e(x,y)$. In this paper, we should compute the smallest containing subgraph to decompress a IDT. Fig. 4 explains the decompression procedure. First, we find the SCS for all edges in ΔE. Second, Delaunay edges(dashed edges) in SCS are removed. Finally, the removed Delaunay edges in SCS are substituted with edges of ΔE. For each edge e in ΔE, the number of intersected edges of Delaunay triangulation $T_d(V, E_d)$ is expected to be a constant.

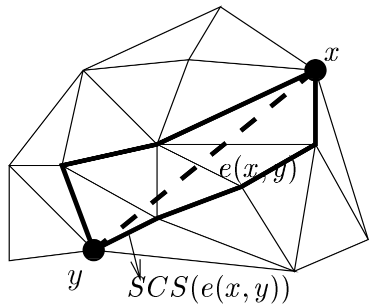


Figure 3: A Smallest Containing Subgraph, SCS, for edge $e(x,y)$

Our experiment shows that it is less than 2, about 1.12. See Table 2. So the number of vertices and edges in $SCS(e(x,y))$ is also a constant on average. Therefore we can find $SCS(e(x,y))$ in constant time by breadth-first searching from x and y. The V_c and E_c are decompressed by Costa's and IBM algorithm before decompression procedure is applied. The algorithm for decompressing IDT is given as follows:

Algorithm Delaunay Decoding
Input : $IDT(V, \Delta E)$: Implicit DT.
Output : $T_o(V, E_o)$: original triangulated terrain.

```
  Create new TMesh T_o;
T_o := Delaunay2D(V);
for each edge g ∈ ΔE {
   E_g = FindSCS(g);
   for each edge h ∈ E_g {
     if ( h intersects g) then discard h;
       else T_o := T_o ⊕ h;
   }
   T_o := T_o ∪ g;
```

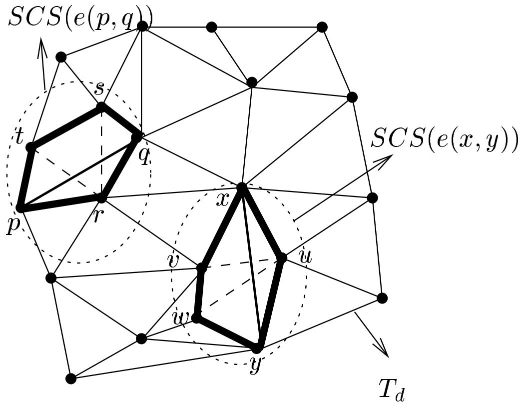


Figure 4: T_d and SCS; dashed edges are included in Delaunay triangulation, and they are stored in $IDT(V, \Delta E)$.

```
}
```

Experimentally, we found that the average number of edges in smallest containing subgraph is about 1.12 as shown in Table 2. So the time complexity for decompressing is $O(p \cdot C_0 \cdot n) = O(0.112 \cdot n)$, where p is ratio of the number of different edges between DT and TIN and C_0 is $|SCS(e(x,y))|$, if DT is given.

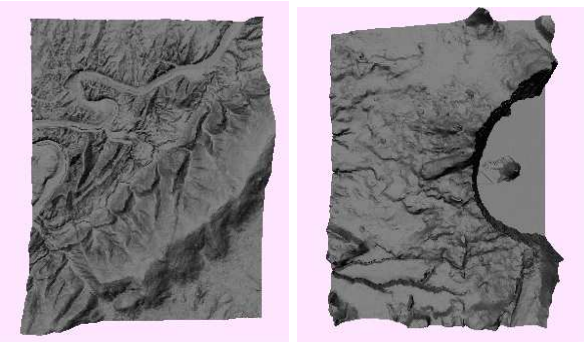
4 EXPERIMENTS

4.1 Similarity between DT and real TIN

Our experiment shows the similarity between original triangulation and 2D Delaunay triangulation. We define the degree of similarity as follows:

$$S = 1 - \frac{m}{n} \ , \ (0 \leq S \leq 1) \qquad (3)$$

where n is the total number of edges in T_o and m is the number of edges in ΔT. We used five general data sets for the experiments. Fig. 5 visualizes two terrains. We tested our algorithm on data sets based on 1:250,000 scale digital elevation models obtained from U.S. Geological Survey. The Crater Lake (Oregon, USA) data was used by [Garland95] for their experiments with the refinement algorithm in the $SCAPE$ system. We measured the similarity of multiresolution models according to simplification methods, such as progressive meshing, vertex decimation, quadric error metrics, greedy insertion and degree method. Table 1 shows that similarity ratio is quite high, namely more than 93%.



(a) Ashby (99468 faces) (b) Craterlake (99472 faces)

Figure 5: Two sample terrains

Faces	Data				
	Ashby	Crater	Ozark	Ntc	Spokane
100%	0.94	0.93	0.94	0.98	0.98
90%	0.94	0.94	0.97	0.98	0.93
70%	0.93	0.94	0.95	0.96	0.92
50%	0.93	0.93	0.93	0.95	0.92
30%	0.93	0.94	0.91	0.94	0.91
10%	0.93	0.95	0.89	0.93	0.89

(a) Similarity ratio to *Progressive Meshes*.

Faces	Data				
	Ashby	Crater	Ozark	Ntc	Spokane
100%	0.94	0.93	0.94	0.98	0.98
90%	0.88	0.92	0.92	0.96	0.96
70%	0.87	0.89	0.88	0.91	0.92
50%	0.87	0.86	0.86	0.87	0.87
30%	0.86	0.85	0.85	0.85	0.84
10%	0.85	0.84	0.83	0.84	0.83

(b) Similarity ratio to *Quadric Error Metrics(QSLIM)*.

Faces	Data				
	Ashby	Crater	Ozark	Ntc	Spokane
100%	0.94	0.93	0.94	0.98	0.98
90%	0.94	0.93	0.95	0.98	0.98
70%	0.95	0.94	0.96	0.99	0.99
50%	0.96	0.95	0.96	0.99	0.99
30%	0.98	0.96	0.98	0.99	0.99
10%	0.99	0.98	0.99	0.99	0.99

(c) Similarity ratio to *Greedy Insertion(SCAPE)*.

Faces	Data				
	Ashby	Crater	Ozark	Ntc	Spokane
100%	0.94	0.93	0.94	0.98	0.98
90%	0.91	0.9	0.91	0.94	0.94
70%	0.85	0.85	0.85	0.88	0.89
50%	0.8	0.8	0.81	0.82	0.83
30%	0.76	0.75	0.77	0.77	0.77
10%	0.72	0.71	0.73	0.72	0.72

(d) Similarity ratio to *Vertex Decimation*.

Faces	Data				
	Ashby	Crater	Ozark	Ntc	Spokane
100%	0.94	0.93	0.94	0.98	0.98
90%	0.94	0.93	0.94	0.97	0.98
70%	0.92	0.92	0.92	0.94	0.95
50%	0.92	0.91	0.91	0.95	0.94
30%	0.89	0.89	0.89	0.9	0.90
10%	0.86	0.85	0.85	0.86	0.86

(e) Similarity ratio to *Degree Method*.

Table 1. The similarity between DT and several TINs generated by the recently proposed algorithms.

Fig. 6 shows the similarity between Delaunay triangulation and Greedy($SCAPE$) triangulation. Note that the similarity increases with smaller number of vertices(=coarser terrain model).

Figure 6: Similarity of Greedy Insertion to DT

Info.	Data				
	Ashby	Crater	Ozark	Ntc	Spokane
$\lvert\Delta E\rvert$	8837	10847	8748	2699	2798
$\lvert E(T_d)\rvert$	149467	149471	149467	149549	149609
F_i=1	7932	10020	7819	2586	2582
F_i=2	786	687	807	112	201
F_i=3	106	127	112	1	15
F_i=4	13	12	10	0	0
F_i=5	0	1	0	0	0
AVG	1.12	1.09	1.12	1.04	1.08

Table 2. The number of intersections between ΔE and T_d. F_i denotes the total number of edges having i-times intersection in ΔE. AVG is the average number of intersections.

Table 2 shows the statistics of edge intersection between ΔE and T_d. This result shows that most edges in ΔE intersects to edges in T_d only once. We can compute $SCS(e(x,y))$ in constant time, since $|SCS(e(x,y))|$ is a constant size. The time complexity of decompressing consists of the time for constructing Delaunay triangulation and $|\Delta E| \cdot C_e$, where C_e is the time for finding SCS for each edge e in ΔE. Therefore, total time for decompressing depends on the performance of Delaunay triangulation only. It takes a few seconds to reconstruct Delaunay triangulation

4.2 Expected Performance of Delaunay Compression

Now we give an expected performance of our Delaunay compression for triangulated terrain data. By conducting several experiments we found that about more than 90% of edges are common in DT and real TIN data. Let m denote the number of edges in a triangulation $T_o(V, E_o)$ of a terrain data. Suppose that about $(1-p) \cdot m$ edges of $T_o(V, E_o)$ are common in DT , where p is ratio of the number of different edges between DT and TIN. Then we need to store the edge information about ΔE, where $\Delta T(V, \Delta E) = T_o \oplus T_d$. So the number of different edges is $|\Delta E|$. And we compress the vertex coordinate information by using Costa's entropy coding. In order to encode ΔE we adopt the IBM connectivity encoding algorithm for the $\Delta T(V, \Delta E)$. According to their work, each vertex can be encoded 4 bits on average.

So we finally got two compressed files, one for vertex coordinates and the other for ΔE). It is easy to see that the number of bits to represent the connectivity information for a TIN graph is $(4 \cdot p \cdot m)$. Thus the expected bit per vertex in our Delaunay encoding scheme is $4 \cdot p \cdot m = (4 \cdot 0.1 \cdot 3n)/n = 1.2$ bits.

5 CONCLUDING REMARKS

We propose a new geometric compression method for triangulated terrain model. Our experiment shows that the typical terrain data e.g., TIN is quite similar to the Delaunay triangulation of the same point set. Several experiments proved that more than 90% of edges are common in DT and any TIN terrain. This means that most of the edges in a plain TIN data could be implicitly defined by Delaunay triangulation rule. By applying this idea, the whole topology of a terrain mesh is encoded with two parts, one is a rule(Delaunay Triangulation) and the other is a small portion of data(vertex coordinates).

A simple analysis shows that this compressing strategy gives 1.2 bits per vertex compressing rate, which is a quit competitive result. Decompressing completes in a few seconds, since our decompressing algorithm depends on the performance of a Delaunay triangulation algorithm only. Since as far as we know there had been lots of very fast Delaunay triangulation implementation, our propose is practical.

One advantage of our algorithm is that we can recover a part of terrain easily since Delaunay triangulation could be done locally due to its generic property. If you want to modify mesh connectivity, you only need to add some edge information to $\Delta T(V, \Delta E)$.

In the future, we have to find a good encoding scheme for ΔE, by exploiting the fact that it is smaller portion comparing with the whole edge set.

REFERENCES

[HH96] Hugues Hoppe, *Progressive Meshes*, SIGGRAPH '96 Proc., pp. 99-108, Aug., 1996.

[HH97] Hugues Hoppe, *View-dependent refinement of progressive meshes*, SIGGRAPH '97 Proc., pp. 189-198, July, 1997.

[HSurvey97] Paul S. Heckbert and Michael Garland, *Survey of Polygonal Surface Simplification Algorithms*, SIGGRAPH '97 Course Notes., May, 1997.

[Schroe92] William J. Schroeder and Jonathan A. Zarge and William E. Lorensen, *Decimation of triangle meshes*, SIGGRAPH '92 Proc., July, pp. 65-70, 1992.

[Turan84] Gyorgy Turan, *On the Succinct Representation Of Graphs*, Discrete Applied Mathematics. pp. 289-294, 1984.

[Kenn95] Kenneth Keeler, Jeffery Westbrook, *Short encodings of planar graphs and maps*, Discrete Applied Mathematics. pp. 239-252, 1995.

[Geom94] Joseph O'Rourke, *Computational Geometry In C*, CAMBRIDGE University Press., 1994.

[Deer95] Michael Deering, *Geometry Compression*, SIGGRAPH '95 Proc. pp. 13-20, Aug., 1995.

[Mchow97] Mike M. Chow, *Optimized Geometry Compression for Real-time Rendering*, Visualization '97 Proc. pp. 347-354, IEEE Computer Society Press., 1995.

[Costa98] Costa Touma, Craig Gotsman, *Triangle Mesh Compression*, Proc. of Graphics Interface '98, pp. 26-34, 1998.

[Taubin96] G. Taubin, J.Rossignac, *Geometric compression through topological surgery*, Research Report RC-20340, IBM Research Division, 1996.

[Garland95] Michael Garland and Paul S. Heckbert, *Fast Polygonal Approximation of Terrains and Height Fields*, CMU-CS 95-181, CS Dept., Sept, 1995.

[Heck97] Michael Garland and Paul S. Heckbert, *Surface Simplification using Quadric Error Metrics*, SIGGRAPH '97 Proc., pp. 209-216, July., 1997.

[JR96] J. R. Shewchuk, *A Two-Dimensional Quality Mesh Generator and Delaunay Triangulator*, http://www.cs.cmu.edu/quake/triangle.html, July, 1996.