Coding Topological Structure of 3D CAD Models

Hiroshi Masuda
Department of Environmental and Ocean Engineering,
The University of Tokyo,
Bunkyo-ku, Tokyo 113 Japan,

Ryutarou Ohbuchi and Masaki Aono
IBM Research, Tokyo Research Laboratory
1623-19, Shimo-tsuruma, Yamato-shi, Kanagawa 242 Japan

Abstract

This paper proposes a loss-less method to encode and compress three-dimensional (3D) geometric models which may contain non-simple topological structures. Any combination of wireframe, surface, and solid components can be encoded and compressed by the method. Furthermore, our method is able to handle models that contain various non-simple topological structures that are likely to exist models generated by using 3D Computer Aided Design (CAD) systems. Previous methods were only able to handle limited classes of relatively simple topological types and thus were not able to handle most 3D CAD models.

Given a model that may include a wide range of non-simple topological types, our method transforms it to a "reduced" model that only contains simple topological types. This topological transformation is performed by applying a sequence of Euler operators to the given model. Both the transformation operation and the reduced model are encoded and compresed to produce compressed data. We adopted a powerful existing method by Taubin et al to encode triangular mesh, the result of reductioin.

We implemented and evaluated our compression algorithm. Experiments with 3D models of mechanical parts showed that, in addition to being able to handle a diverse and complex topological types, our method achieves excellent compression ratios.

Keywords: Data compression, three-dimensional geometric modeling, solid model, computer aided design (CAD).

1 INTRODUCTION

Increasing number of mechanical products are developed and manufactured as collaborative endeavors of multiple entities, for example, among multiple companies or multiple divisions within a company. Consequently, sharing of design and manufacturing data among these entities by using the Internet and other medium have become quite important. Activities to ensure exchangeability of design data, for example the ISO 10303 [16], informally known as STEP, are intended to facilitate such data sharing.

Exchangeability of design data alone, however, is not sufficient to promote sharing. Recent advent of three-dimensional (3D) Computer Aided Design (CAD) systems allowed us, for example, to create a design consisting of a large number of complex solid models. This trend resulted in explosions in data sizes of such 3D models. The explosion in data size significantly increased demand for communication bandwidth and storage space.

Bandwidths of commnication media for data exchange, such as the Internet, have not kept pace with the increase in model data size, however. Consequently, exchanging design data of size tens to hundreds of megabytes over a network takes significant amount of time. The shortage of communication bandwidth has become a major impediment to the widespread use of collaborative design.

An obvious approach in alleviating network bandwidth shortage is to compress data to be transferred, which is, in our case, 3D CAD models.

Numerous compression methods for such data objects as text, sound, image, and movie have been studied extensively for years. It is only recently that several compression methods for 3D models have been proposed [8, 6, 12, 9]. Triangle strip and Triangle fans, included in IRIS GL and OpenGL can also be viewed as methods to compactly encode triangular meshs. While these algorithms achieved siginificant compression, they alrogithms are not very effective in compressing 3D geometric models created by CAD systems. The major impediment here is the lack of supports in these compression methods for diverse and complex topological types that may be included in 3D CAD models.

For example, Deering[8], Hoppe[6] and Li[12] support only triangular meshes. Taubin and Rossignac extended the domain to polygonal models with non-manifold and with non-orientable conditions [9]. However, Taubin's method still has not been able to handle such non-simple topological structures as holes in faces, cavities in solids, strut edges, and self loops. (These topological constructs are illustrated in Figure1.)

To accept models produced by contemporary 3D geometric CAD systems, a compression method must extend its domain so that diverse range of topological types including those illustrated in Figure 1 are included.

It has been known that an arbitrary topological modification can be realized by applying a sequence of Euler oprators to a model. Furthermore, original topology of the model can be recovered by applying reverse Euler operators in an inverted sequence [1]. Our new compression method for 3D models with arbitrary topological strucutre utilizes this property.

Our compression method first transform an original 3D model by using a sequence of Euler operators so that the result of the transformation contains only simple topological types. Both the recorded sequence of Euler operators used for the transformation and the transformed model, which is a triangular mesh, are encoded, compressed and stored and/or transmitted.

In order to efficiently encode topology of the reduced

model, we employed a polygonal meshe encoding method developed by Taubin and Rossignac [9]. Encoded topology of the transformed mesh and the Euler operator sequence used for the transformation are then processed by using entropy coders appropiate to respective data types. Similary, attributes, which may include color and geometry (e.g., coordinates of vertices) of the model are also processed by using appropriate entropy coders. Throghout the process, correspondence of topological entities and their attributes are maintained by using adjacency of the topological components.

Decoder of the compressed model data employs a process which is essentially an inverse of what is used for encoding. Entropy decoders are followed by reconstruction of original topology by using an inverted sequence of reverse Euler operators. Attributes (including geometry) and their relactions to topological components are maintained and recovered. Attributes themselves are also decompressed and added to the recovered topology.

The rest of this paper is structured as follows. In Section 2, we present an overview of our compression method, followed in Section 3 by an description of Euler operators we employed. We will present, in Section 4, a method to maintain correspondence between topological entity and geometry and/or attributes. Next, we will describe coding method for the reduced topology in Section 5, and coding method for geometry in Section 6. In Section 7, we will present results of experiments that applied our coding method to several 3D CAD models. We conclude in Section 8 with summary and remarks on future work.

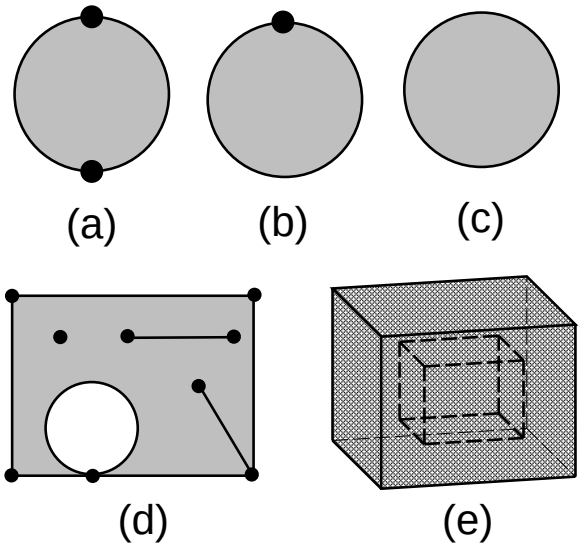

Figure 1: Examples of non-simple topological structures used by 3D CAD models; degenerate polygons (a, b, and c), an isolated vertex, a strut edge, an isolated edge, and a self loop on a surface (d), and a cavity in a solid (e).

2 OVERVIEW OF ENCODING AND DECODING PROCESS

This section presents an overview of the encoding and decoding steps of the 3D model compression method. Details of each step will be explained in the following sections.

The diagram in Figure 2 illustrates the encoding and decoding process. Decoding of compressed data is essentially and inverse of encoding so that this section only describes the encoding process. In this figure, the downward flow of data is for encoding and the upward flow of data is for decoding. We assume that the encoder accepts as its input 3D models that may include solid, surfaces, and/or wireframe models.

Processing required for encoding can be summarized as follows;

- *Simplifying topological structures:* Simplify topology of the input model down to triangular meshes while recording the sequence of Eular operators used for the simplification.
- *Encode reduced topological entities:* Encode and compress the sequence of Eular operators used for the simplification.
- *Encode reduced triangular mesh:* Encode and compress topology of the triangular meshes, which are the results of the simplification.
- *Encode geometry and attributes:* Encode and compress the attributes and geometry of the simplified triangular meshes.

In addition, steps above requires several bookkeeping operations.

- A unique identifier need to be assigned to each topologycal entity of the triangular meshes, the result of simplificatoin, so that the entities can be correctly associated with their geometry and attributes.

The remaining of this section briefly describes each of the steps and bookkeeping operations.

2.1 Simplifying Topological Structures

In order to encode a 3D CAD model, solid and surface models components of the orignal model are topologically transformed so that they become simple polyognal models. At the same time, cavities in solids, holes in faces, strut edges, self-loops, and other non-simple topological structures in the original model are removed. Both of these transformations are performed by using a sequence of Euler operators. Be it solid, surface, or wireframe, the end product of this transformation step is one or more triangular meshes. The kinds, operands, and application seqence of Euler operators used for the transformation are recorded, encoded, compressed, and packed with the reduced simple polygonal mesh model to produce a compressed model.

2.2 Ordering and Numbering Topological Entities

A unique sequential number is attached to each topological entity of the triangular meshes and/or wireframe models. By also ordering geometry and attribute components, the numbering is used to maintain correspondence between topology and its geometry and/or attribute components. Here, geometries could be vertex coordinates of polygonal meshes or coordinates of control points parametric surfaces, and attributes could be stiffness and masses of solids.

At the same time, geometry (e.g., vertex coordinates of triangular meshes or control point coordinates of parametric

surfaces) and attiributes (e.g., stiffnesses and masses of solid objects) that are associated with the topological entities are also numbered. These numbering are necessary to maintain association of topological entities with their corresponding geometry and attiributes.

If vertices are uniquely and sequentially numbered, they can be associated with their coordinates by simply storing both in the same order. Our algorithm, however, encode topology differently from geometry and/or attributes since these two groups have quite different characteristics and thus require different coding approaches. Numbers attached to both topological entity and their geometry and/or attributes are used to later recover their correspondences upon decompression.

2.3 Encoding Reduced Topological Entities

At this stage, topology data contains both triangular meshes and wireframe models. Due to their different topological properties, the two are encoded by using different methods. Topology of triangular meshes are encoded by using the method developed by Taubin and Rossignac [9], which is probably the most efficient method developed so far.

To encode wire frame models, spanning trees of their vertices are computed first. A spanning tree here is defined as a tree that covers every vertices on the wire frame. Then, edges that are on the spanning tree are encoded separately from edges that are not included in the spanning tree. This increases coding efficiency.

2.4 Encoding Geometry and Attribute

Non-topological entities such as geometry and attribute that are associated with a topological entity are encoded by using methods appropriate for each one of them. In this paper, we assume geometry as the representative of the non-topological entity for explanation.

To encode geometry, we employ a loss-less compression technique. While lossy compression is known to yield a large compression ratio, lossy compression is not torelated for CAD applications. For example, a boolean operation (e.g., difference) of a computational solid geometry modeler will fail if geometries of the models to be operated have errors.

2.5 Entropy Coding

Finally, all encoded data are compressed by using an entropy coding method. For the experiment reported in this paper, we employed a general-purpose entropy coding tool based on a dictionaly based compression algorithm. Further increase in compression efficiency is expected if we were to fine tune this entropy coding stage.

3 TOPOLOGICAL MODIFICATION

3.1 Euler operators

As outlined before, our method first reduces solid and surface models down to a set of triangular meshes by applying a sequence of Euler operators. The reduction process is illustrated in Figure 3. Names of Euler operators employed in this figure are defined in Table 3.1, in which (a) lists a set of Euler operators for encoding, and (b) lists a set of Euler operators for decoding, each of which are inverse their counterpart in (a).

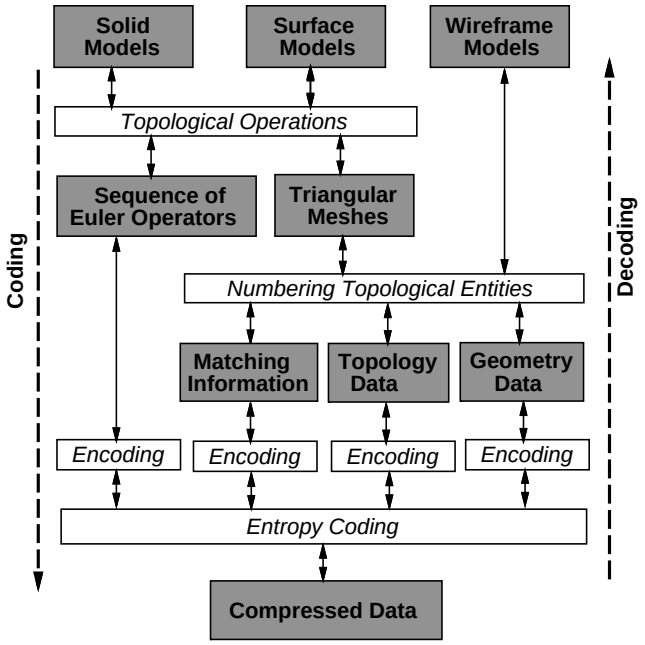

Figure 2: 3D geometric model encoding process.

This reduction process is reviersible so that if inverse of each operator is applied in an inverted sequence, reduced model can be transformed back to the original. This reduction enables us to apply efficient coding methods developed for triangular meshes.

1. *Separation of shells:* Solids with cavities are separated into multiple shells so that cavities are eliminated, by using the Kill Cavity and Make Model (KCMM) operator. Figure 3.1 (a) illustrates this transformations. After this transformation, solids can be treated as sets of orientable connected faces (that are, shells).

2. *Subdivision of Surface Edges:* In order to remove degenerate topological entities, by using Split Edges (SE) operator, edges in loops are segmented by inserting vertices so that every loop has more than three vertices. In Figure 3 (2), segmentation of a circle by adding two vertices illustrates an example of this transformation. This transformation is necessary since CAD systems routinly generates such degenerate topological entities, which can not be encoded by simply applying existing coding methods, such as Taubin's. In addition, if two edges share two vertices and thus can not be distinguished from each other, one of the edges is segmented by inserting a vertex so that the two can be distinguished uiniquely by their starting and ending vertices. In Figure 3 (2), insertion of a vertex into an oval with two vertices illustrates an example of this transformation.

3. *Removal of Holes and Strut Edges:* A surface with holes consists of more than one loops. Similarly, a surface with strut edges consists of more than one loops. In these cases, holes and strut edges are removed by applying Make Edge and Kill Ring (MEKR) operator, as illustrated in Figure 3 (3). After this transformation, all the entities are consisted of only one loop.

4. *Triangulation:* Every face, now consisting of single loop, is tessellated into triangles, as illustrated in Figure 3 (4), by using an operator called Make Edge and Face (MEF). This tessellation can be performed quickly, since most of it is purely topological and need not consider geometric factors, such as avoidance of triangles with high aspect ratio.

   A care must be take, however, so that less than three vertices are shared by any pair of triangles. To achieve this, we employ a constrained Delaunay triangulation. In the example of Figure 4 (a), two adjacent rectangles (one of them concave) are wrongly tesselated into four triangles in which two adjacent triangles share all their vertices. Such triangular mesh can not be handled properly by the following triangular mesh coding stage. Our constrained Delaunay triangulation will produce a tessellation like the one shown in Figure 4 (b).

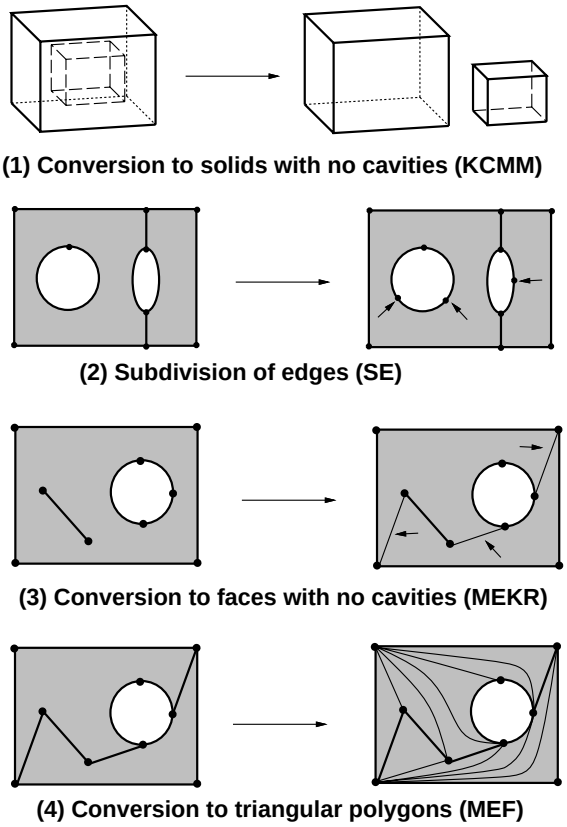

Figure 3: Operations for topological transformation.

3.2 Encoding reverse Euler operators

An original model is converted to a reduced model by applying, at each step, a specific Euler operator to a specific topological entity. For a bidirectional conversion between the original model and the reduced model the following must be recoded;

- The kind of each Euler operator,
- the operand of each Euler operator, i.e., the topological entity the operator is applied to, and

Operations	Meaning
KCMM	kill cavity and make model
SE	split edge
MEKR	make edge and kill ring
MEF	make edge and face

(a) Euler Operators for Coding.

Operation	Meaning
MCKM	make cavity and kill model
ME	merge edge
KEMR	kill edge and make ring
KEF	kill edge and face

(b) Reverse Euler Operators for Decoding.

Table 1: Types of Euler operations.

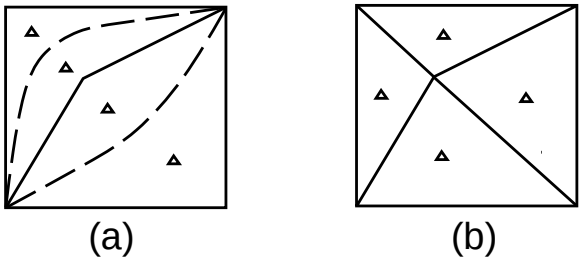

Figure 4: (a) Triangles that share all three vertices and (b) constrained Delaunay triangulation.

- the sequence of {*Euler operator, operand*} pairs as they are applied for reduction.

We assume that every topological entity at every step of conversion is uniquely identified by an identification (ID) number. This is necessary to uniquely identify the operand of each Euler operator. The ID numbers are assigned by using a method that will be described in the next section.

By using a sequence of {Euler operator, operand} pairs recorded, a reduced topology is reverted back to its original topology according to the steps below.

1. *Inverting Triangulation:* Triangulation is inverted, by deleting edges added upon reduction by using the *Kill Edge and Face (KEF)* operator, which is an inverse of the MEF operator. The operand for the KEF operator is given as an object ID number.

2. *Inverting Removal of Holes and Strut Edges:* A loop is recovered by deleting an edge that bisected the loop by using *Kill Edge and Make Ring (KEMR)* operator along with its operand, that is, an dge. In addition to the operation and operand, distinction must be made if the reconstituted loop is an outer loop or an inner loop. Outer-loop/inner-loop distinction is made by the direction of loop traversal, i.e., whether it is traversed clockwise or counter-clockwise. Thus the loop traversal direction is encoded as 1 bit flag and stored with each KEMR operation.

3. *Inverting Subdivision of Surface Edges:*

   As the model was reduced for compression, vertices and edges were inserted by using SE operator. These vertices and edges are deleted by using a sequence of Merge

Edge (ME) operator. Obviously, the object to be removed must be specified.

4. *Inverting Separation of Shells:*

   As the final step to reconstitute original topology that include cavities in solids, *Make Cavity and Kill Model (MCKM)* operator is used to combine two shells into a solid with a cavity. Each shell is identified by an ID number of a vertex included in the shell. Outer shell and inner shel of the solid with cavity is distinguished by a one bit flag attached to each shell.

(Comment(Ohbuchi11271998): This item (few paragraphs around here) must be revised and rewritten.)

In general, at each step, consequence of applying a series of Euler operators depends on the operator's application order. However, in this case, orders of Euler operator applications may be exchanged at each step, since topological entities involved in the operation can be uniquely identified as those that adjoin with each other. (COMMENT: RO So what??)

For example, two faces used by each ME operator can be identified by using a vertex identifier stored with the ME operator.

This application order independence at each step of reduction leads to very high compression ratio in encoding the operations and their operands, as explained below.

Since each reduction step automatically identifies the operator of the step, only operands, that are, entity ID numbers, need to be encoded at each step. Entity ID numbers are sorted and their first-order difference are computed. The first-order difference of the ID numbers consists mostly of small intergers e.g., 0s, 1s, and 2s. In addition, these numbers are repetitious. Consequently, entity ID numbers can be compressed very efficiently by using an entropy coding method, such as an arithmatic coding or dictionary based coding algorithms.

4 MAINTAINING TOPOLOGY-GEOMETRY CORRESPONDENCE

For proper decoding, correspondence between topological data and geometric data must be maintained. We attached to entity identifier to vertices, edges and faces, in order to specify operand of Euler operators. The identifier is also used to maintain correspondence of geometry and topolocy. If a topological entity has an ID number n, its geometry can be obtained as the n-th geometry data. Attribute data associated with the entity can also be found in a similar manner.

One-dimensional ordering of topological entities can be created by using spanning tree, which is defined as a tree that covers every nodes in a connected graph. In our algorithm, entities are ordered, by using spanning trees of vertices as the starting point.

In general, a model may contain multiple disconnected regions and thus require more than one spanning trees to cover all the vertices in the model. Figure 5 (a) shows an example of a spanning tree that covers a surface model. In the figure, the black square indicates the starting (root) vertex and white circles indicates ending vertices (leaves).

A spanning tree is generated given a pair of initial condition, a starting vertex and a starting edge. In case of surface and solid models, given a "current" vertex, the edge to be followed to reach the next vertex is chosen as the first edge, in clockwise order about the current edge, from the edge that connected the current and previous vertices. If the chosen edge made a loop back to the vertices already in the tree, it is unselected and next edge in clockwise direction is selected. In case of wireframe models, an arbitrary edge that does not make a loop back to vertices in the existing tree is selected as the edge that ledges to the next vertex. The process continues until there is no vertex reacheable from the current vertex. After a spanning tree is generated, the tree is traversed, depth first, to produce one-dimensional ordering. If a model requires more than one spanning trees to cover all the vertices, each of tree requires a new pair of initial conditions.

Figure 5 (b) shows the spanning tree whose vertices are numbered following a depth-first traversal.

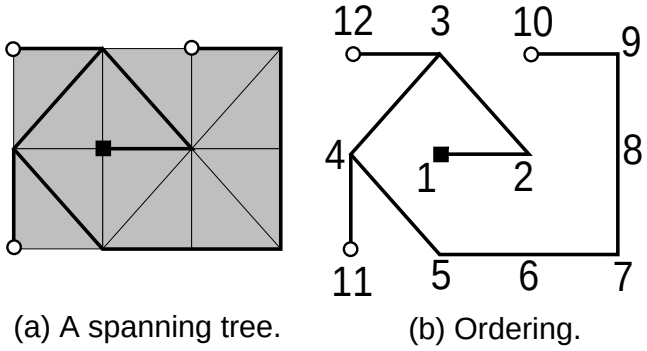

Figure 5: Ordering vertices using a spanning tree.

Sequential numbers of edges and faces are determined by using ID numbers of vertices. We identify an edge $[v_i, v_j](v_i < v_j)$ by vertex IDs at its both ends. We define an ordering relation among edges as $[v_{i1}, v_{j1}] < [v_{i2}, v_{j2}]$ if $v_{i1} < v_{i2}$ or $v_{i1} = v_{i2}$. Then, edges can be sorted and their sequential numbers are obtained. Faces can be sorted in a similar manner, by identifying a face by using an orderd set of vertices $[v_i, v_j, ...]$ and defining an ordering relation based on the ordering of the vertices.

A method to encode a set of spanning trees is necessary to properly assign sequential ID numbers to toplogical entities. A method to encode spanning trees is explained in the next section.

5 COMPRESSING REDUCED MODEL

5.1 Compressing triangular mesh

Once arbitrary topological structures are reduced to triangular meshes, we can apply existing mesh topology compression methods such as Taubin's [9], Deering's [8] or Hoppe's [6]. In our implementation, we adopted Taubin's method due to its very high compression rate.

Let us explain the Taubin's triangular mesh topology compression method by using the example of 5.

In order to compress the model shown in Figure 5, the model is cut through the edges in the spanning tree, as shown in Figure 6. In other words, edges in the spanning tree is split into two edges. This cutting produces triangle strips. In 6, the triangle strips are indicated by the dotted arrows starting from triangular symbols. Each triangle strip can be encoded by the start triangle and a sequence of 1 bit flags that indicate whether the next triangle is adjascent to either right (R) or left (L) edge of the current triangle.

For example, the longest strip in Figure 6 is encoded as *LLRLLLLLL*.

This way, topological structure of triangular meshes can be stored by encoding spanning trees and triangle strips. Recovery of the original triangle meshes can be accomplished by decoding triangle strip from the left/right flag sequence, and then "stich" the strips together along the edges of the spanning tree.

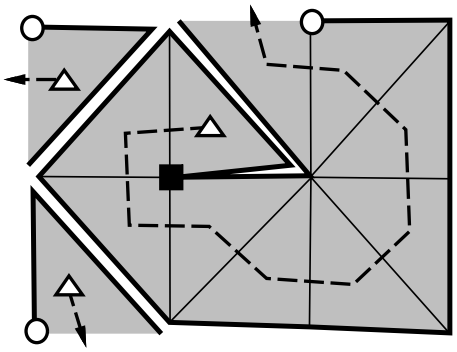

Figure 6: A method for coding triangular meshes.

5.2 Compressing wireframe

Compression of wireframes is performed in a similar manner. Each edge in a wireframe model is classified into one of two subtypes, depending on whether the edge is included in a spanning tree. These two subtypes are compressed separately.

Edges on a spanning tree is encoded as follows by using the vertices IDs at both end. For explanation, we use the example of 5(a), but assume this time that 5(a) is a wire frame, not a triangular mesh, and its spanning tree is as shown in 5(b).

This spanning tree can then be represented as 123(4(5678910)11)12). In this notation, numbers indicate vertex IDs, the symbol '(' indicates that the tree has a non-traversed branch at the vertex immediately before, and the symbol ')' indicates that the ID immediately before the symbol is a leaf. Notice that the IDs are ordered in a strictly ascending order with an increment of one. Borrowing the idea of run-length coding, this tree can then be coded as 3(1(6(1)1); each number means length of a "run" of vertex IDs before it is interrupted by either '(' or ')' symbols. The run-length-encoded representation of the tree is then encoded by using an entropy coder.

If an edge is not on the spanning tree, the edge is represented by a pair of vertices at both ends of the edge that are sorted in an ascending order, i.e., $[sv_i, ev_j](i = 1, 2, .., n)$. We then take first order difference of the sorted vertex IDs $[sv_i - sv_{i-1}, ev_i - sv_i](i = 1, 2, .., n.sv_0 = 0)$. The difference is then encoded by using an entropy coder.

While this compression scheme is straightfoward, it is quite effective in most of the cases we tried. Chances are that adjascent vertices have IDs close to each other. If first order difference is taken, most of the differences are small numbers, e.g., 0, 1, or 2. Such distribution of numbers can be encoded quite efficiently by using any entropy coder. Furthermore, topologically similar partial shapes (topologically repetitive features) can also be captured by using certain types of entropy coder, for example, a dictionary based entropy coder. Consequently, repetitive features can be compressed very efficiently.

6 COMPRESSING GEOMETRY

Vertices, edges, and faces may have geometric components, that are, coordinate values, curves, and surfaces, respectively. These geometric components are corrections of floating point numbers, such as coordinates of vertices or control points of curves and surfaces, combined with a bit of additional information, such as the degree of a polynmial curve. In the current implementation, in order to compress geometric components, we simply remove redundancy by using a loss-less entropy coding method.

Compression methods can be classified into either loss-less or lossy methods. A loss-less compression maintains allows exact recovery of values after decompression, but its compression ratio can not be very high. A lossy compression grants loss of information in the original data, but significant data reduction is possible. Choice of compression algorithms depends on applications of the data to be compressed.

Most CAD applications demands exact reproduction of original data after decompression. For example, Boolean operations in a Computational Solid Geometry (CSG) modeler are quite sensitive to geometric errors. Such applications require loss-less compression methods for geometry. On the other hand, if the application is to simply view CAD models, or to perform interference checks, accuracy of double precision floating point numbers are in general not necessary. A lossy compression method will suffice for this kind of applications. Examples of lossy compression methods for geometry can be found in [14, 13].

While both lossy and loss-less compression methods need be considered depending on applications, we will discuss a loss-less compression method intended for general CAD applications.

In this paper, we further restrict the type of geometry to coordinate values, which are represented by three-tuple (x,y,z) of double precision floating point numbers. Note that such coordinate values, used for vertex coordinates and control points for curves and surfaces, are the dominant component in terms of data size among various object types in geometry.

Coordinate values of vertices are encoded by taking advantage of the one-dimensional ordering of vertices mentioned before. Each curves and surface may contain either one-dimensional or two-dimensional list of such coordinate values. These ordered coordinate values can be treated similary to the vertex coordinates.

Several methods have been proposed to encode sequences of coordinate values, for example, linear predictive coding and predictive residual vector quantization [9, 15]. We observed that these methods are effective if values are normalized and quantized to fixed precision integer values, that is, when the method is lossy.

THIS PORTION ON FLOATING POINT NUMBER ENCODING IS A BIT IMPRECISE / WEAK.

Previous algorithms to compress coordinate values rely on the coherence of the data values, that is, continuous and smooth changes in coordinate values. They are most effective when the mesh is dense so that the coordinate values have high degree of coherence, and when the values can be normalized to integer values with 16 to 24 bits precision. Most 3D CAD systems, on the contrary, tend to employ

coarser meshes (each of which is a curved surfaces, for example) that has smaller degree of coherence and require that their coordinate values to be represented in 64 bit floating point formats with 52 bit or so of mantissa. Consequently, previous algorithms listed above that are used to compress coordinate sequences were considered not appropriate for 3D CAD data. We thus employed a rather simple (near) loss-less scheme to compress coordiante values.

We employ 1st order difference coding in order to reduce dynamic range of the coordinate values. To represent vertex coordinate of the vertex n, differences ($x_n - x_m$ for the coordinate x, for example) of its coordinate and the coordinates of vertices adjacent to it on the vertex tree is computed. Among the differences, the largest is selected as the difference. For example, on the tree of Figure 5 (b), the coordinate value of vertex 12 is encoded by using the difference of coordinate valuese between vertex 12 and 3.

By such normalization, exponent part of the floating point representation can be eliminated in most of the cases.

An exceptional case occurs if the exponent of that difference $d_x(n,m) = x_n - x_m$ is larger than that of x_n; in such a case, $(x_n - x_m) + x_m \neq x_n$. This exception is flagged by a using a 1 bit flag, which is stored with the difference value $(x_n - x_m)$.

The sequence of difference values are then compressed by using an entropy coding scheme. In our experiment, we used `gzip`, a popular dictionary based compression tool, for the entropy coding. Repeated difference values resulting from repeated features can be compressed quite efficiently by using the entropy coder.

7 IMPLEMENTATION AND EXPERIMENTAL RESULTS

In order to handle a wide range of 3D model representations, including wireframe, surface, and solid models, the algorithm described in this paper was implemented by using a non-manifold geometric modeler [2] as the kernel. This modeler employs the radial-edge data structure and provides a complete set of Euler operations on wireframe, surface, and solid models.

Our compression/decompression modules are implemented as UNIX^TM filters, which converts ASCII files to compressed binary data and vice versa. Briefly, the ASCII file that represents the (uncompressed) 3D model has the following format.

bu

- Vertices are represented by an ordered list of coordinate values. Each coordinate consists of three floating point numbers. A vertex at nth position in the list is implicitly given an identifier n.
- An edge is represented by a pair of vertex identifiers. Each edge in the ordered list of edges is given an identifier in a similar manner as the vertices.
- A loop is represented by a list of vertex identifiers listed in counter-clockwise order. Each loop in the ordered list of loops is given an identifier in a similar manner as the vertices.
- A face is represented by a set of loops in the face. Each face in the ordered list of faces is given an identifier in a similar manner as the vertices.

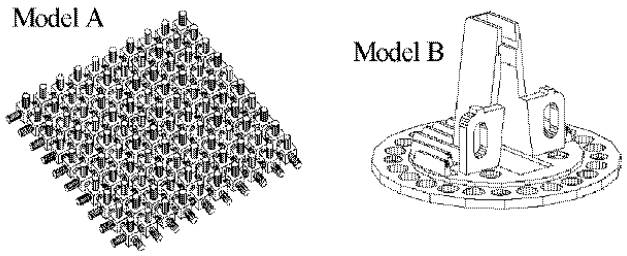

Figure 7: Examples of 3D Models.

Figure 7 shows examples to which our coding method was applied. The model A in the figure was created by unifyng 100 identical objects, while the model B in the figure was created by applying 150 form-features. Both of these models were polygonal facet models without free-form curves and surfaces; this is not a loss of generality since our algorithm compression algorithm focuses on compression of topological data. Compression methods for surfaces themselves can be found, for example, in [14] and [13].

Table 2 shows compressed sizes, in kilo byte, and compression ratios, in percentile, of the two models of the Figure 7. In applying our compression algorithm, we treated the models as both wireframe model and solid model. We simply ignored face data in the model for the wireframe models. Compression ratio in the table is defined as $(CompressedSize) \div (OriginalSize) \times 100$. We compared our method and `gzip` for the compression ratios.

As see in the table, the algorithm worked well for both of the models, achieving high compressiong ratios. Redundancy due to the repetitive topological features in both of the models were quite effectively removed by the algorithm. As expected, compression ratios for the model A are significantly higher than those of model B. This can be explained by the fact that the model A is much more repetitive in its topological features than the model B. We believe that the our compression algorithm could be used as an effective tool in storing and transferring 3D CAD data.

8 CONCLUSIONS

We have proposed a new loss-less encoding method for 3D CAD models with arbitrary topological features. The topological features may include, for example, holes, strut edges, isolated edges in faces, or cavities. In our approach, a model with arbitrary topological features is reduced, step by step, down to a simple triangular mesh model by using a sequence of Euler operators. Both topology and geometry of the reduced triangular mesh is then encoded. In the current implementation, we adopted Taubin's method[9] in order to compress topology of the resulted triagular mesh. However, other techniques can be adopted in place of the Taubin's method as other efficient techniques are discovered. Finally, geometry, essentialy an ordered list of floating point values, and the encoded topology is compressed by using an entropy coder. Throughout the process, correspondence of the geometry and topology are maintained. Recovery of the original model essentially follows the inverse of the encoding steps.

We implemented and evaluated our approach, assuming that each mesh in the original topology is a polygon. The experiments showed that our method achives an excellent compression ratios. The compression ratios were significantly

Model	Original Size	Compressed by Our Method	Compressed by gzip
Wireframe (T)	153.89KB (100%)	0.23 KB (0.15%)	54.75KB (35.58%)
Wireframe (T+G)	403.49KB (100%)	3.66KB (0.91%)	87.86KB (21.78%)
Solid (T)	495.52KB (100%)	11.43KB (0.31%)	186.59KB (37.66%)
Solid (T+G)	745.12KB (100%)	14.86KB (2.00%)	219.70KB (29.49%)

(a) Compression of Model A (5520 faces, 15600 edges).

Model	Original Size	Compressed by Our Method	Compressed by gzip
Wireframe (T)	29.72KB (100%)	0.66 KB (2.22%)	12.67KB (42.63%)
Wireframe (T+G)	82.48KB (100%)	6.73KB (8.16%)	23.81KB (28.87%)
Solid (T)	93.35KB (100%)	4.21KB (4.51%)	38.80KB (41.56%)
Solid (T+G)	146.11KB (100%)	10.28KB (7.03%)	49.94KB (34.18%)

(b) Compression of Model B (1116 faces, 3304 edges).

Table 2: Compression ratios of two 3D models shown in Figure 7. In the table, letter T and G denotes topology and geometry, respectively, and T+G denotes both topoogy and geometry.

higher than those of a general purpose loss-less compression tool. It should be mentioned that the steps that compresses reduced topology and geometry can be improved to further improve compression ratio.

In the future, we intend to extend our system so that the system could handle various geometric elements, such as Coons, Bezier, B-spline, or Non-Uniform B-Spline curves and surfaces, in the original model While our current implementation could only handle simple polygons as the geometric element of the original (uncompressed) model, real-world CAD models may include various types of curves and surfaces such as those listed above.

We also intende study lossy compression algorithms for CAD models, especially for those models that include cuves and surfaces. If a model is simply viewed by human beings, for example to evaluate its shape by using a model broswer, a lossy compression algorithm with a very high compression ratio may be what is needed; a lossy compression could achieve a higher compression ratio than the one achievable by the loss-less compression.

References

[1] Mantyla, M. and Sulonen, R.: GWB: A Solid Modeler with the Euler Operators, *IEEE Computer Graphics*, Vol.2, No.7, pp.17–31, Sep. (1982).

[2] Masuda, H.: Topological Operators and Boolean Operations for Complex-Based Non-Manifold Geometric Models, *Computer Aided Design*, Vol.25, No.2, pp.119-129, Feb. (1993).

[3] Weiler, K.: Topological Structures for Geometric Modeling, *PhD. Thesis, Rensselaer Polytechnic Institute*, Aug. (1986).

[4] Schroeder, W. J., Zarge, J. A., Lorensen W. E.: Decimation of Triangle Meshes, proc. *ACM SIGGRAPH '92*, pp.55-64, (1992).

[5] Turk, G.: Re-Tiling Polygonal Surfaces, proc. *ACM SIGGRAPH '92*, pp.55-64, (1992).

[6] Hoppe, H.: Progressive Meshes, proc. *ACM SIGGRAPH '96*, pp.99-108 (1996).

[7] Kumar, S., Manocha, D. and Lastra, A. : Interactive Display of Large NURBS Models, *IEEE Transactions on Visualization and Computer Graphics*, Vol.2, No.4, pp.323-335 (1996).

[8] Deering, M. : Geometry Compression, proc. *ACM SIGGRAPH '95*, pp.13-20 (1995).

[9] Taubin, G. and Rossignac, J.: Geometry Compression Through Topological Surgery, *ACM Transactions on Graphics*, Vol. 17, No 2, pp.84-115 (1998).

[10] Taubin, G., Gueziec, A., Horn, W., and Lazarus. F.: Progressive Forest Split Compression, *ACM SIGGRAPH '98*, pp.123-132 (1998).

[11] Gumhold, H. and Strasser, W.: Real Time Compression of Triangle Mesh Connectivity, *ACM SIGGRAPH '98*, pp.??-??, (1998)

[12] Li, J-K., Li, J., and Jay Kuo, C.-C.: Progressive Compression of 3D Graphic Models, proc. *IEEE International Conference on Multimedia Computing and Systems*, pp.??-??, June (1997).

[13] Devore, R.A., Bjorn, J. and Lucier, B.J. : Surface Compression, *Computer Aided Geometric Design*, Vol.9, pp.219-239 (1992).

[14] Derose, T.,D., Lounsbery, M., and Reissel, L.: Curves and Surfaces, Wavelets and their Applications in Computer Graphics, SIGGRAPH '95 Course Notes, Fournier, A. (ed.), pp.123-154 (1995).

[15] Bossen, F.(ed) : Description of Core Experiments on 3D Model Coding, *Draft on ISO/IEC JTC1/SC29/WG11*, March (1998).

[16] International Organization for Standardization. *Industrial Automation Systems and Integration - Product Data Representation and Exchange*, 1994. (International Standard ISO 10303:1994, informally known as STEP).