Journal of Southeast University (English Edition) Vol. 21, No. 2, pp. 165 - 169 June 2005 ISSN 1003—7985

Compression of finite element hybrid mesh

Zeng Jianjiang Chen Wenliang Zhai Jianjun

(College of Aerospace Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract: A method for encoding and compressing finite element models is proposed. The model may be various non-simple topological structures and contain any combinations of beams, triangular elements and quadrilateral elements. First the model is subdivided into simple meshes that are orientable and manifold. Based on the Edgebreaker algorithm, 13 labelled pairs are introduced for quadrilateral meshes and five other labelled pairs are introduced for triangles. Then the connectivity information of mixed triangle/quadrilateral meshes is coded in a direct manner. Two other bits are used to record the wireframe information. For the pure wireframe model, Taubin's method is extended to compress it. The compression algorithm is implemented and evaluated. Experiments with several models show that the method achieves excellent compression ratios.

Key words: finite element model; mesh; compression

Finite element analysis is time consuming. For a complex model, it will take lots of time to solve it. Zeng[1] shows us that a web based system can shorten the time. In such a system, the finite element model (FEM) is prepared on the client side, and then the result of the finite element model can be sent to an FEM solver on the server side. Benefitting from the high performance of a server machine or such technology as parallel computing, the result will be obtained quickly and transferred to the client. Such a system lowers the requirements of performance of the client machine.

Due to the increase of the size and the complexity of these models, the large storage space consumption required for the standard representation of meshes makes the manipulation and exchange of the models over the Internet more and more problematic. In general, a finite element model includes geometrical data, element information and attributes.

For the compression of finite element models, many mesh compression methods can be referred to[2-5]. The methods proposed by Gumhold and Strasser[3], and Rossignac[5] only encode connectivity. The method proposed by Touma and Gotsman[2] predicts geometrical properties better, and the method proposed by Li and Kuo[4] improves on the entropy encoding of prediction errors. Bajaj et al.[6] proposed another method to encode single-resolution triangular meshes. It is based on a decomposition of the mesh into rings of triangles originally used by Taubin and Rossignac[7] in their compression algorithm, but with a different and more complex encoding. All of these schemes require $O(n)$ total bits of data to represent a single-resolution mesh in compressed form.

Due to the following reasons, the above methods cannot directly be applied to general finite element compression:

- In finite element models, there exist several kinds of elements simultaneously: triangular elements, quadrilateral elements, tetrahedral elements, hexahedral elements and beams etc. Therefore, the geometric model is a combination of wireframes, triangles and quadrilaterals etc.
- The topology of the finite element models is complex. A finite element model may be non-manifold and nonorientable.

1 FEM Mesh Topological Subdividing

Triangle and quadrilateral meshes are compressed by region growing in our methods. The method asks for an orientable and manifold model. So for a nonorientable and non-manifold mesh, the mesh will be subdivided into several orientable manifold meshes which will be compressed separately.

If the original mesh is non-manifold, singular points will be found by checking the connective relation of all vertices. The singular points will be duplicated and recorded, then the original mesh can be divided into several manifold meshes as shown in Fig. 1(a). The nonorientable mesh can be divided into several orientable meshes in the same way. The orient-collision

Received 2004-08-16.

Biography: Zeng Jianjiang (1971—), male, associate professor, ezengjj@yahoo.com.cn.

edges will be split and the left and right triangles will not share these edges. Finally an orientable mesh is obtained (shown in Fig. 1(b)).

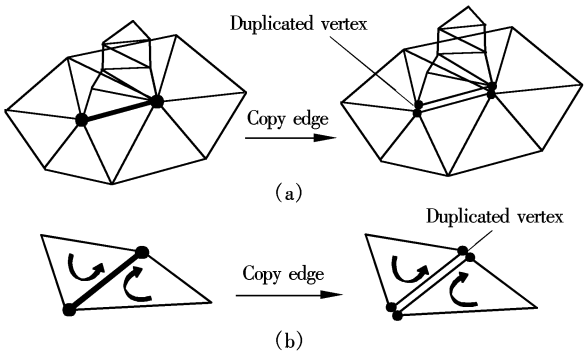

Fig. 1 Subdivide complex meshes into simple meshes by splitting edge. (a) Non-manifold mesh; (b) Nonorientable mesh

It is obvious that the topology of the model has been changed after the original mesh is subdivided into an orientable manifold. In order to keep the original topology, all the copied vertices should be recorded and appended to the compressed file. After decoding the mesh, all the duplicated vertices should be merged to recover the topology.

2 Quadrilateral/Triangle Mesh Compression

Now the model is manifold and orientable. For the mixed triangle/quadrilateral meshes (QT-meshes), one simple way to compress a quadrilateral mesh or a mesh containing other polygons is to split all polygons into triangles and to apply one of the triangle mesh compression algorithms. To maintain compatibility with the original mesh, decompression may need to delete the added edges and restore the original quads. As a result, compressing these meshes by triangulating all polygons may require encoding additional information describing how to recover the original polygon mesh structure from the stream of compressed triangles.

Our approach to encoding QT-meshes is a generalization of the Edgebreaker method for encoding triangle meshes[8,9]. King generalizes the algorithm to quads by defining a more complex set of labels to represent the edges and vertices which have been previously visited. For the polygons, the number of combinations may be computed via a recurrence relation based on the fact that no unvisited vertex may be incident to a previously visited edge. The solution to this recurrence for a polygon with n edges is the Fibonacci number $F(2n-1)$. For a quadrilateral, therefore, there are $F(7) = 13$ possible combinations of visited and not-yet-visited edges and vertices.

King labels each quad with the two CLERS labels of the triangles resulting from the split. The splitting rule therefore leaves only 13 possible label pairs: CR, CC, LE, CS, SC, LC, SE, LL, LR, LS, SL, SR, and SS. For comparison, consider that there are 23 possible pairs of adjacent symbols in the Edgebreaker code for a triangle mesh, only 18 of which correspond to adjacent triangles in the mesh, since any pair starting with E includes two triangles from different branches of the spanning tree.

Fig. 2 shows the compression process traversing a small region of a quad mesh and splitting the resulting quads. The region pictured is a portion of a larger Q-mesh. The current quad is light gray; previously visited quads are either dark gray or not pictured; not-yet-visited quads are white; and quads pushed onto the stack after each S have a diagonal pattern. The resulting CLERS string for this region is CCSESELCCRCRSESECRCRSRLESELE. The growing label strings are displayed adjacent to the corresponding compression stages. Further details of the compression process are given in Ref. [9].

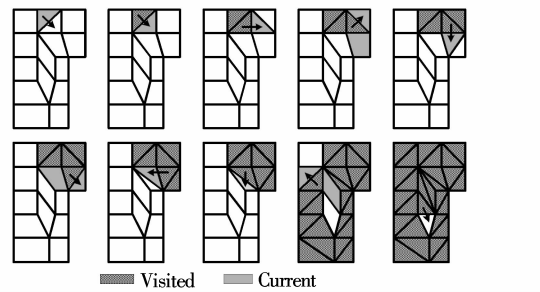

Fig. 2 Encoding a Q-mesh

In practice, many QT-meshes have triangles as only a small fraction of their faces, with 8% to 12% triangles common in finite element meshes. We can introduce a sixth label, T, to identify a triangle face, thus representing each triangle as a label pair TC, TL, TE, TR, or TS as shown in Fig. 3. By ensuring that each label pair represents a single face, this approach simplifies the design of the decoder and it makes it easy to exploit the relationships between adjacent labels and adjacent quads. This method is the most effective for such predominantly quadrilateral meshes in which the cost of the extra T labels is low. Since the Q-mesh compression algorithm above uses the same CLERS labels as Edgebreaker uses for triangles, it may be generalized to meshes containing both quads

and triangles by a single CLERS label for each triangle and using one of the 13 CLERS label pairs above for each quad.

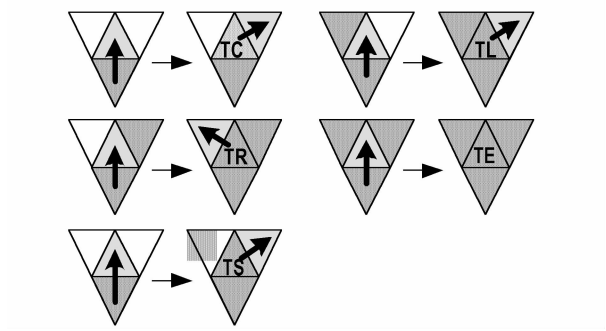

Fig. 3 Label pairs for the triangle

Finally we apply entropy coding to take advantage of different frequencies of the symbols. Here a predefined codebook is adopted. Due to the topological properties of manifold meshes, an upper bound on the total length of the mesh code is achieved. This optimization for a triangle mesh is similar to that described in Ref. [8], so we elaborate on a quad mesh with a minority of triangles, which is a case frequently encountered in finite element models.

3 Wireframe Compression

If there are beam elements in a finite element model, the beam elements are treated as a wireframe. Many finite element models include beams. For example, a stringer, bar of beam and rib of the flight should be treated as beam elements. When the meshes are traversed, the most wireframe will be visited. As shown in Fig. 4, when introducing a new triangle, at most two wireframes will be introduced. We can express all the four cases by two bits. 00 means no wireframe, 01 means the left edge is a wireframe, 10 means the right edge is a wireframe and 11 means both edges are wireframes. For E triangle is introduced, no wireframe is introduced since all neighbors have been visited.

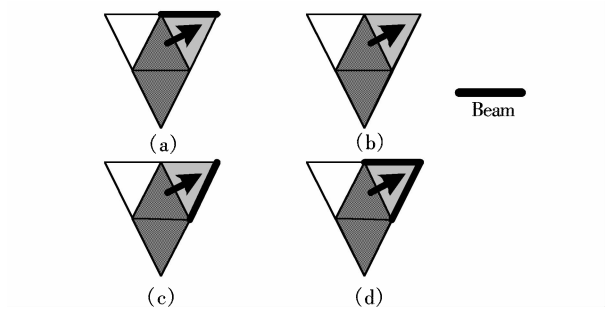

Fig. 4 Four possible cases when a new triangle is introduced. (a) No wireframe is introduced; (b) Left edge is a wireframe; (c) Right edge is a wireframe; (d) Both edges are wireframes

For the quadrilateral mesh, two bits just deal with half of the quadrilateral. Fig. 5 shows an example of a Q-mesh with wireframes.

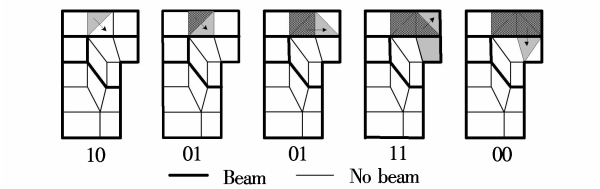

Fig. 5 Encoding the Q-mesh with wireframes

However, there are pure beam or bar element finite element models. For this kind of model, compression of wireframes is performed in a simple manner[10]. Taubin's method is extended to deal with wireframe compression.

Edges on a spanning tree are encoded as follows by using the vertices' IDs at both ends. For explanation, we use the example of Fig. 6(a), and assume Fig. 6 is a wireframe, not a triangular mesh, and that its spanning tree is also shown in Fig. 6. This spanning tree can then be represented as 1 2 3 4 5 6 7 8 (9 10 11 12 13 14 15(16 17 18 (19) 20)21) 22). 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 8(7(3(1) 1)1); each number means length of a "run" of vertex IDs before it is interrupted by either the symbol "(" or ")". The run-length-encoded representation of the tree is then encoded by an entropy coder.

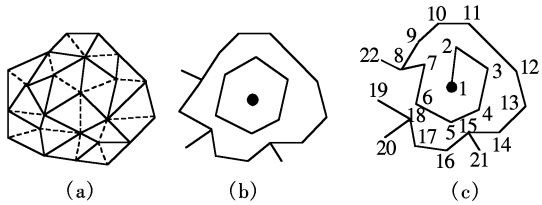

Fig. 6 Ordering vertices by spanning tree

While our method for encoding and compressing wireframes is simple, it is quite effective in most of the data we tried. A symbol set with such a skewed distribution can be encoded quite efficiently by using any entropy coder. In addition, certain entropy coders, such as dictionary-based algorithms, can recognize and compress topologically repetitive features in the models quite effectively.

4 Coordinate of Node Compression

Our algorithm assumes that a vertex coordinate is represented by a three-tuple $p = \{x; y; z\}$ of double precision floating-point numbers. The current implementation of our algorithm employs a simple loss-less compression method. In terms of data size, such coordinate values are the dominant component among geometrical data. In order to expose coherence that resides in data, our geometrical compression algorithm relies again on the one-dimensional ordering of vertices created by means of the vertex spanning tree.

The algorithm compresses vertex coordinate x, y, and z independently of each other. After the vertices have been one-dimensional ordered, the algorithm computes, for each coordinate, the first-order difference of coordinate values in order to reduce their dynamic range. Let us explain this for the case of coordinate x. Given a vertex p_n, a vertex adjacent to it on the vertex tree is examined. Among the vertices adjacent to p_n, the previous vertex p_m is selected, and the difference $dx(n, m) = (xn - xm)$ is computed. The list of the first-order difference values are then entropy coded. In our experiment, we used gzip, a popular general-purpose dictionary-based compression tool, for the entropy coding.

We note, for the coarse models, small geometric errors can be tolerated. Lossy surface compression methods may be useful in this case. Such lossy predictive encoding methods as Ref. [11] can achieve higher compression ratios.

5 Experiments and Conclusion

We have implemented the algorithms described in this paper and run them on some real-world models. Fig. 7 shows the models that are used to evaluate our 3-D model coding algorithm. The model of a car door in Fig. 7(a) includes triangle and quadrilateral elements. The model of a car body in Fig. 7(b) is more complex and includes triangle and quadrilateral elements. The model of an aircraft includes triangle and quadrilateral and beams elements.

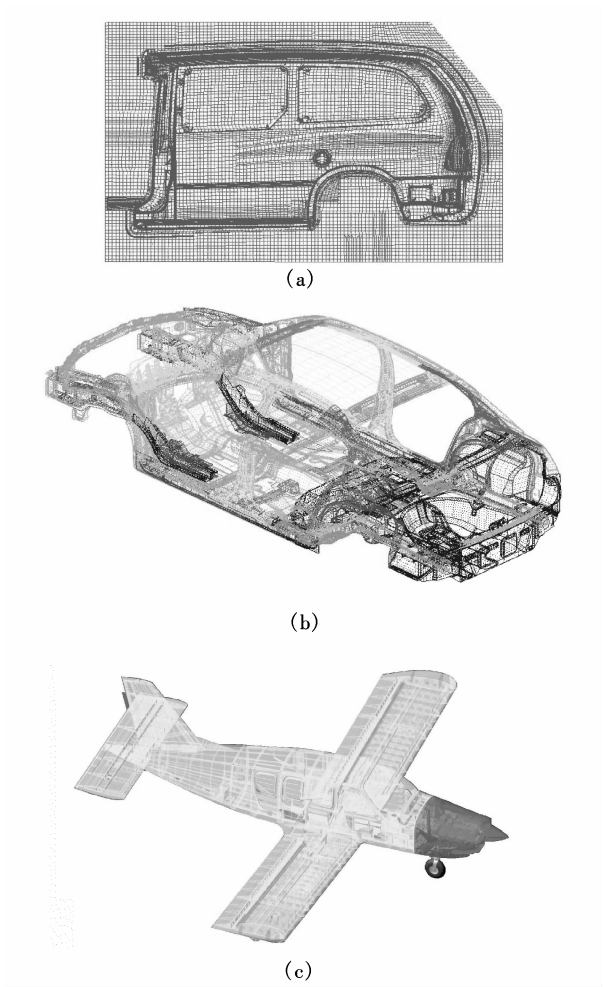

Fig. 7 Finite element model. (a) Car door; (b) Car body; (c) Aircraft

Tab. 1 shows the results of compression experiments using the three models. Data sizes are indicated in kilobytes, and compression ratios are indicated in percentiles. We compared our method and gzip of Nastran file for the compression ratios. As seen in Tab. 1, the algorithm performed well for all of the models, achieving high compression ratios.

In this paper, a method is developed to deal with finite element model composed of triangle/quadrilateral meshes and wireframes. The complex model is transformed into the simple meshes firstly. Then the connectivity information is coded in a direct manner. Another two bits will record the wireframe information.

Tab. 1 Experimental results of the algorithm

Model	Number				Size of file/KB		
	Vertices	Triangle	Quadrilateral	Beam	Nastran	Gzip of Nastran	Our method
Car door	3 832	294	3 834	0	518	87.1	19.6
Car body	148 922	16 528	134 927	0	26 123	3 895	927
Aircraft	32 845	5 692	62 812	20 684	9 262	1 035	352

References

[1] Zeng J. A web-based CAD system [J]. *Journal of Materials Processing Technology*, 2003, **139**(1 - 3): 229 - 232.

[2] Touma C, Gotsman C. Triangle mesh compression [A]. In: Davis W, Booth K, Fourier A, eds. *Proceedings of the* 24*th Conference on Graphics Interface* [C]. San Francisco, 1998. 26 - 34.

[3] Gumhold S, Strasser W. Real time compression of triangle mesh connectivity [A]. In: *Proceedings of SIGGRAPH* [C]. Orlando, USA, 1998. 133 - 140.

[4] Li J, Kuo C. Progressive coding of 3D graphics models [J]. *Proceedings of the IEEE*, 1998, **86**(6): 1052 - 1063.

[5] Rossignac J. Edgebreaker: connectivity compression for triangular meshes [J]. *IEEE Transactions on Visualization and Computer Graphics*, 1999, **5**(1): 47 - 61.

[6] Bajaj C, Pascucci V, Zhuang G. Single resolution compression of arbitrary triangular meshes with properties [A]. In: *IEEE Data Compression Conference* [C]. San Francisco, 1999. 307 - 316.

[7] Taubin G, Rossignac J. Geometric compression through topological surgery [J]. *ACM Transactions on Graphics*, 1998, **17**(2): 84 - 115.

[8] King D, Rossignac J. Guaranteed 3. 67V bit encoding of planar triangle graphs [A]. In: *Proceedings of the* 11*th Canadian Conference on Computation Geometry* [C]. Vancouver, British Columbia, Canada, 1999. 146 - 149.

[9] King D, Rossignac J, Szymczak A. Connectivity compression for irregular quadrilateral meshes TR-99-36 [R]. USA: Georgia Institute of Technology, 1999.

[10] Masuda H, Ohbuchi R. Coding topological structure of 3D CAD models [J]. *Computer Aided Design*, 2000, **32**(5): 367 - 375.

[11] Taubin G, Gueziec A, Horn W, et al. Progressive forest split compression [A]. In: *ACM SIGGRAPH* [C]. Orlando, 1998. 123 - 132.

有限元混合网格的压缩

曾建江　陈文亮　翟建军

(南京航空航天大学航空宇航学院, 南京 210016)

摘要: 提出了一个对有限元模型进行编码压缩的方法. 该模型的拓扑结构可以是任意型式, 允许包含四边形单元、三角形单元和梁(杆)单元. 有限元模型首先分解成一系列的可定向的流形模型. 基于 Edgebreaker 算法, 针对四边形网格遍历的情况引入 13 对标记, 同时对混合网格中的三角形用 5 对标记来表示. 这样, 混合网格的连接信息可以采用一种直接的方式进行编码. 然后再使用 2 比特位记录模型中的线框信息. 对于完全线框模型, 采用扩展后的 Taubin 方法进行压缩. 该压缩算法已经实现并进行了测试. 多个复杂模型的压缩实验表明该方法具有很好的压缩效率.

关键词: 有限元模型; 网格; 压缩

中图分类号: TP391. 72