The Compression of the Normal Vectors of 3D Mesh Models Using Clustering

Deok-Soo Kim, Youngsong Cho, and Donguk Kim

Department of Industrial Engineering, Hanyang University, 17 Haengdang-Dong, Sungdong-Ku, Seoul, 133-791, South Korea
dskim@hanyang.ac.kr, {ycho1971, donguk}@ihanyang.ac.kr

Abstract. As the transmission of 3D shape models through Internet becomes more important, the compression issue of shape models gets more critical. The issues for normal vectors have not yet been explored as much as it deserves, even though the size of the data for normal vectors can be significantly larger than its counterparts of topology and geometry. Presented in this paper is an approach to compress the normal vectors of a shape model represented in a mesh using the concept of clustering. It turns out that the proposed approach has a significant compression ratio without a serious sacrifice of the visual quality of the model.

1. Introduction

As the use of Internet is an every day practice, the rapid transmission of data becomes one of the most critical factors to make the business successful. To transmit data faster through network, the compression of the data is the fundamental technology to obtain. Since the advent of its concept, the compressions of text, sound and image have been investigated in depth and we enjoy the result of research and development of such technologies in everyday life. The research on the compression of shape model, however, has been started very recently. In 1995, Deering published the first and noble paper discussing the issue [3].

The issues related to the topology and/or geometry of shape models have been extensively investigated. However, its counterpart for normal vectors has not been explored as much as it deserves, even though the size of the data for normal vectors can be significantly larger than its counterparts of topology and geometry.

Normal vectors may or may not exist in a model even though they are necessary for the visualization of the model. If normals do not exist in a model, usually rendering software creates the normals from the face definitions. However, there are frequently cases that normals are necessary. Once normal vectors exist in a shape model, the file size of the normals is quite big as shown in Fig. 1 compared to the size of topology and/or geometry data. In the examples shown in the figure, normals take almost half of whole data of shape models. It should be noted that the ratio may vary depending on the model and the implementation of the authoring tools. The models used in this paper are represented in VRML and were created using a CAD system called ProEngineer.

P.M.A. Sloot et al. (Eds.): ICCS 2002, LNCS 2330, pp. 275–284, 2002.

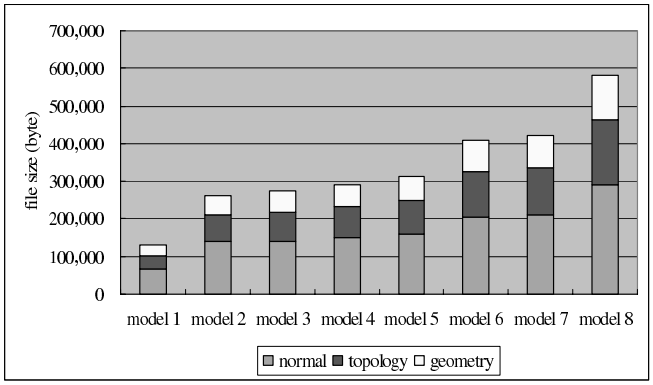


Fig. 1. The sizes of geometry, topology and normal vectors for various mesh models represented in VRML

A normal vector usually consists of three real numbers represented by float numbers, and a float number uses 32-bits in IEEE standard. It has been known that two normals with a discrepancy of 2^{-46} radian can be distinguished with normals represented in the float numbers, and it is generally agreed that this is too much detail information than necessary for the general graphics objects [3]. Hence, the compression of the normal vectors is an important issue for the exchange of a shape model through Internet or graphics pipeline.

The presented algorithm takes advantage of the well-known technique of clustering. The normals of a model are automatically grouped in a set of clusters where each cluster has a representative normal which is computed as a mean value of normals in the cluster. Each representative normal vector has a unique identification associated with it and the normals in a cluster are represented by the cluster identification for its use.

It turns out that the proposed algorithm compresses the normal vectors of a model in less than 10 % of the size of the original model without a serious sacrifice of the visual quality of the appearance of the models.

2. Related Works

Ever since Deering's noble work on the compression of shape model, there have been several researches on various aspects of shape model compression. One line of researches is the compression of topology [2][3][5][8][10][11][13], geometry [3][9][11][13], colors [12], normal vectors [3][12], and so on.

Among several algorithms regarding on the topology compression, Edgebreaker [10] reserves a special attention since our algorithm uses the information produced by Edgebreaker. Edgebreaker transforms a mesh structure into a string of C, L, E, R, and S symbols and redefines the sequence of faces in the original mesh model without creating any change in the model.

The research for normal vector compression has not yet been explored as much as it deserves. Deering's approach is the table-based approach to allow any useful normal to be represented by an index. To convert a normal vector into an index of normal on the unit sphere, the unit sphere is split up into 48 segments with identical surface area. In the each segment, the normal vectors are represented by two *n*-bits integers which two angles using spherical coordinates are converted into [3].

In Taubin's proposal for VRML compressed binary format, normal vectors are quantified with a subdivision scheme. An octant of the base octahedron is divided by recursively subdividing the base triangle *n* times. A normal vector is then encoded into a sequence of $3+2n$ bits, where *n* is the subdivision level [12].

In Hoppe's Progressive mesh, he introduced a wedge to represent discontinuities of normals on mesh. It is a set of vertex-adjacent corners whose attributes are the same. In vertex split recodes, a field of 10 bit encodes the wedges to which new corners created by vertex split are assigned and the deltas of normals is encoded [6].

3. Representations of normal vectors in VRML model

Among the several representations of shape models, a mesh representation, a triangular mesh model in particular, is the main representation in this paper, and the mesh model is assumed to be orientable manifold.

A mesh model is usually obtained from a surface model or the boundary of a solid model, and normal vectors are assigned at the vertices or faces of the model depending on the requirements of the visual appearance of the model. The principle use of normals in a shape model lies in the rendering process so that a model can be visualized more realistically. Generally, the shading algorithms, such as Gouraud shading and Phong shading [4], require the normal be known for each vertex of a shape model.

Usually there are four different ways to assign normal vectors to a mesh model reflecting two factors. First, normal vectors may be assigned at the vertices or faces. Second, the coordinate values of the normal vectors may be explicitly represented where they should appear or the indices of the vectors may be used instead of the coordinates themselves. In fact, this categorization lies under the design concept of VRML itself.

The node to define a triangular mesh in VRML 97 is *IndexedFaceSet*, and the node has two fields named *normalPerVertex* and *normalIndex* [1]. If the value *normalPerVertex* field is TRUE, the normal vectors in the mesh model are assigned at the vertices of the model. Otherwise, the normal vectors are defined at the faces. If the *normalIndex* field exists in the mesh definition, the coordinate values of a vertex may be defined only once and they can be referenced as many as needed via the indices. Hence, the possible ways that normals are related to a mesh model in VRML 97 can be illustrated as shown in Fig. 2. Even though Fig. 2(a) and (c) look similar, they are in fact different method in the sense that (a) does not use indices while (c) does.

Among these four possible configurations of assignments, we will be presenting a compression algorithm and file format for Case 4 that the normal vectors are assigned

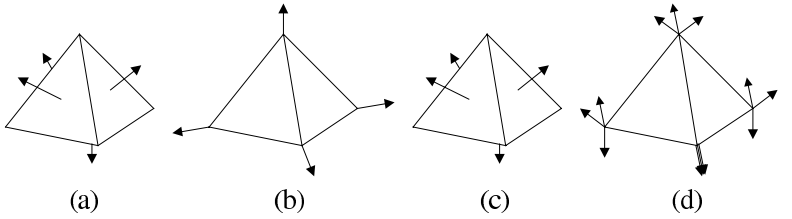


Fig. 2. Categorization of normal vector representation in VRML (a) Case 1 : *normalPerVertex* (FALSE) *normalIndex*(non-existing) (b) Case 2 : *normalPerVertex*(TRUE) *normalIndex*(non-existing) (c) Case 3 : *normalPerVertex*(FALSE) *normalIndex*(existing) (d) Case 4 : *normalPerVertex*(TRUE) *normalIndex*(existing)

at the vertices and the normal indices are used. Note that this case is the most general configuration among the four cases.

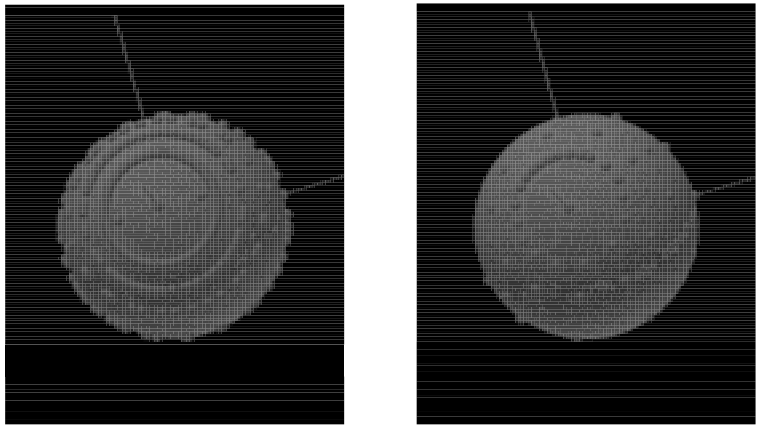
4. Clustering of the normal vectors

The approaches to compress normal vectors should be investigated in three aspects: bit-efficiency, the amount of information losses, and the decompression time.

Note that the approaches by Deering and Taubin yield a number of surface (this surface is a vector surface on which the end points of normals lie) segments with identical area. Provided that the normal vectors are usually unevenly distributed in the space (in its angular distribution), some of the surface segments are less frequently used than the others. Both approaches also use a fixed length code to reference a surface segment and therefore the bit-efficiency can decrease significantly if the distribution of normals is seriously skewed.

Based on this observation, the bit-efficiency can be improved in one of two ways: With surface segments with identical area, the entropy encoding using Huffman code may be used so that the frequency of the normal occurrences in the segment may be accounted. The other way could be to make a number of regions so that each region may contain normal vectors with identical or closer to identical number of normals. In this case, hence, the areas of related surface regions may differ. In this way, the bit-utilization of fixed length code may be maximized. Theoretically speaking, both approaches will reach similar compression ratio, except that the first approach needs Huffman tree itself to be stored. In addition, the first approach needs to navigate Huffman tree in the decompression process.

Considering these factors, we propose an approach that reflects the concept to assign identical number of normals to a segment. In our algorithm, clustering technique is used to compress the normal vectors. We use the standard and simplest K-means algorithm for the clustering. In the future, it may be necessary to analyze the pros and cons of different clustering algorithms for the compression of normal vectors. K-means algorithm is an iterative clustering algorithm. Suppose that the number of clusters and the initial mean of each cluster are given. K-means algorithm then assigns each data to the closest cluster based on a dissimilarity measure and



(a) (b)

Fig. 3. (a) Normal vectors of original model (b) Cluster means after clustering

updates cluster mean values by incorporating the new data until the cluster means are stabilized [7]. The dissimilarity measure is the angle between normal vectors. The cluster mean is the axis direction of the smallest enclosing cone.

Shown in Fig. 3 (a) is 1,411 normal vectors from a bolt model. The small balls on a larger sphere represent the end points of unit normal vectors. Note that several normals may coincide at a ball on the sphere. Fig. 3 (b) illustrates the mean normal vectors of each cluster, where all normal vectors are grouped into 64 clusters.

5. Normal Vector Compression

As was discussed, we will be presenting a compression algorithm for Case 4 that the normal vectors are assigned at the vertices and the normal indices are used.

5.1 Encoding of normal vector

Consider an example that has three faces and six normals as illustrated in Fig. 4. Suppose that the clustering process yielded three clusters A, B, and C, where $\mathbf{n}_1$ and $\mathbf{n}_4$ are in the cluster A, $\mathbf{n}_2$ is in the cluster B, and the others, $\mathbf{n}_3$, $\mathbf{n}_5$, and $\mathbf{n}_6$, are in the cluster C. In Fig. 4, f_1 is initially related with three normals $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_3$, and this fact is reflected in the first three integers, 1, 2, and 3, respectively, in *Normal Index* array. These integers denote that f_1 is related with the first, the second, and the third vectors in *Cluster Pointer* array which again point to the cluster mean values, $\overline{\mathbf{n}}_A$, $\overline{\mathbf{n}}_B$, and $\overline{\mathbf{n}}_C$, of clusters A, B, and C, respectively.

Suppose that *Cluster Pointer* array is rearranged via a sorting operation so that the key values of the elements in the array are in an ascending order. Then, the representation of the model is identical to the one before the sorting was applied, if the index values in *Normal Index* array are appropriately adjusted.

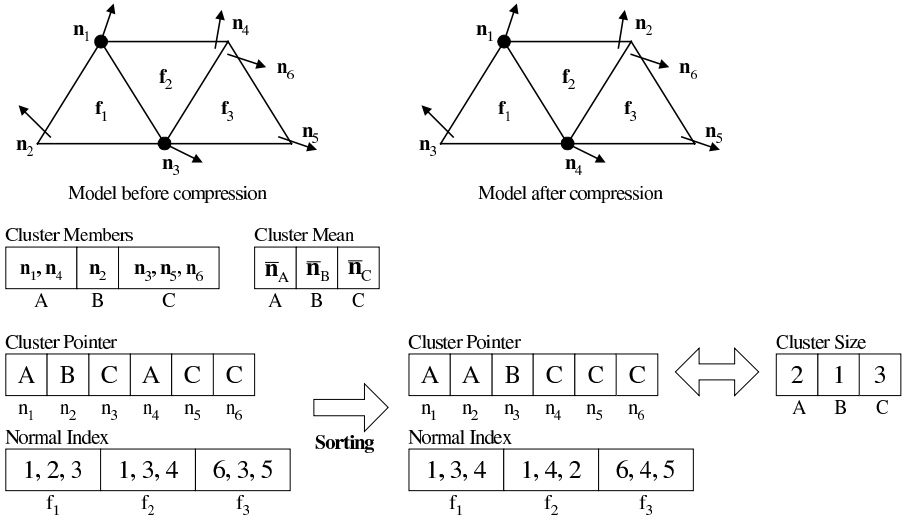


Fig. 4. Encoding of normal vector

This observation is well-illustrated in the figure. Suppose that *Cluster Pointer* array *before* sorting is rearranged to *Cluster Pointer* array *after* sorting. Then, the information related to a face f_1 remains identical if the normals $\mathbf{n}_2$ and $\mathbf{n}_3$ (represented by integers 2 and 3) in *Normal Index* array *before* sorting is changed to $\mathbf{n}_3$ and $\mathbf{n}_4$ in the rearranged *Normal Index* array *after* sorting, respectively. Hence, f_1 is related with $\mathbf{n}_1$, $\mathbf{n}_3$, and $\mathbf{n}_4$ after the rearrangement of *Cluster Pointer* array without any topological change in the model.

Then, we further encode *Cluster Pointer* array by devising another array named *Cluster Size* which contains the number of occurrence of each cluster in a sequential order as shown in the figure. In the given example, *Cluster Size* array contains three elements where the first, the second, and the third elements indicate the occurrences of normals associated with the clusters A, B, and C, respectively.

The first integer 2 in *Cluster Size* array, for example, indicates that the first two elements in *Cluster Pointer* array are pointing to the cluster A. Note that this also means that the cluster A consists of two normal vectors that are $\mathbf{n}_1$ and $\mathbf{n}_2$ from the clustering operation. Similarly, the second integer 1 means that the cluster B has one normal that is $\mathbf{n}_3$. Then, it is easy to verify that model before compression and model after compression altogether defines identical model without any ambiguity as shown in Fig. 4.

5.2 Encoding of normal index

Our algorithm assumes that topology of the mesh model is compressed by Edgebreaker. Note that Edgebreaker redefines the sequence of faces of the original mesh model and the changes should be reflected in the indices of the normals. In addition, it should be reminded that the normal indices were changed once more when the sorting operation was performed.

It is interesting to observe that a normal index is more likely to appear within a few *topological distances* after its first appearance. The topological distance is defined here as the integer that represents the relative distance between the currently recurring normal index value and the latest occurrence of the same normal index value in *Normal Index* array.

Suppose that a triangle in a model has an associated unique normal vector. Then, normal indices at three vertices will refer to an identical normal, and the index values will be identical. The normal indices of a face may show up in a consecutive order in *Normal Index* array. Also, suppose that three triangles share one vertex, and there is a unique normal at the vertex (which is frequently the case in many models). Then, *Normal Index* array will have an identical index value three times with two topological intervals in-between. Since Edgebreaker creates a triangle chain by producing a new neighboring triangle, a normal index at the vertex of a face is likely to appear after one or two (or more) triangles are created. We separate the normal indices into two distinct groups: *absolute* and *relative* indices.

Suppose that an index to a particular normal vector appears at a particular position in *Normal Index* array. When the second occurrence of the index happens within a prescribed maximum topological distance, denoted as r, the second occurrence of the index is represented by relative index that has the topological distance as the value. On the other hand, the second occurrence of the index may happen beyond r or a new index, not existing beforehand, the index is represented by absolute index and has the index itself as the value.

Fig. 5 (a) is *Normal Index* array after sorting in Fig. 4. Suppose that $r = 4$. Normal indices of 1 and 4 in f_2 have topological distance 3 and 2, respectively, and normal index of 4 in f_3 has topological distance 3. These three indices are represented by relative indices as shown at the gray cells in Fig. 5 (b).

On the other hand, the normal indices of 1, 3, and 4 in f_1, 2 in f_2, and 6 in f_3 are completely new indices in the array. In other words, they do not exist in the array beforehand. Hence, they have to be represented by absolute indices. In the case of the normal index 3 (the very last integer in the array) in f_3, the topological distance is 7 since $\mathbf{n}_3$ was shown up as the second integer in the array which is far beyond the predefined r (=4) and therefore it is also represented by an absolute index.

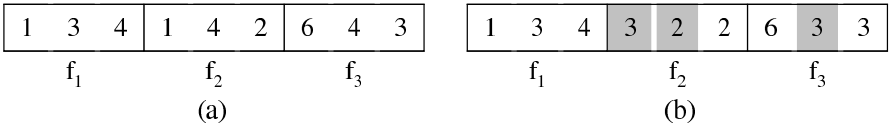
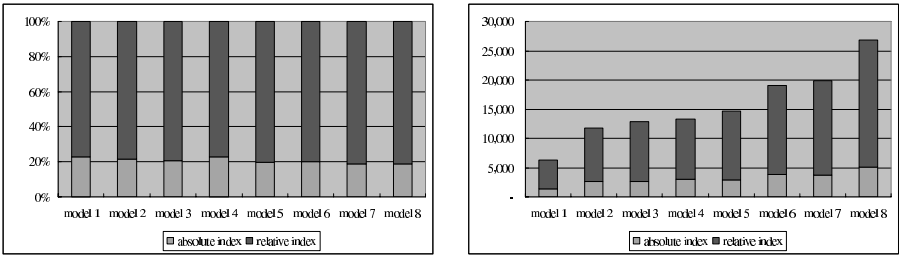


Fig. 5. (a) usage of normal indices (b) mixed usage of absolute and relative index ($r = 4$)

To effectively use the mixed indices of absolute and relative, it is necessary to consider two factors: the bit-size of relative indices and the coverage of indices with relative indices. Depending on these two factors, the mixed use may compress the model or may not.

Shown in Fig. 6 is the frequency distribution of normals that can be covered by the absolute and the relative indices. Note that $r = \infty$ for the relative indices in this experiment meaning that all recurring indices are counted. As shown in the figure,

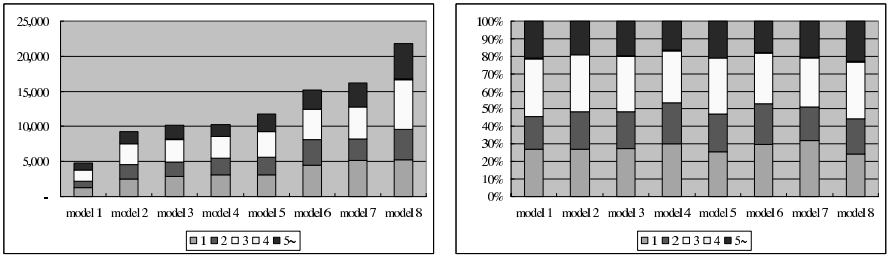
approximately 80 % of the normal indices are recurring. Hence, the normal indices can be better compressed if the recurring normals can be more efficiently encoded, if the appropriate *r* is prescribed.



(a) (b)

Fig. 6. Distribution of absolute index vs. relative index (a) Frequencies, (b) Percentiles

To decide appropriate number for r, we performed another experiment that is summarized in Fig. 7. The figure shows the coverage of relative indices which can be covered by the topological distances 1, 2, 3, 4, and higher than 5. Note that the relative indices with topological distances 1, 2, and 3 altogether cover roughly 80% of total relative indices. This observation suggests that most of the relative indices can refer to the latest occurrence of the same normal index using 2-bits of memory.



(a) (b)

Fig. 7. Occurrence of relative indices with topological distance 1, 2, 3, 4 and, 5+ (a) Frequencies, (b) Percentiles

6. Experiment

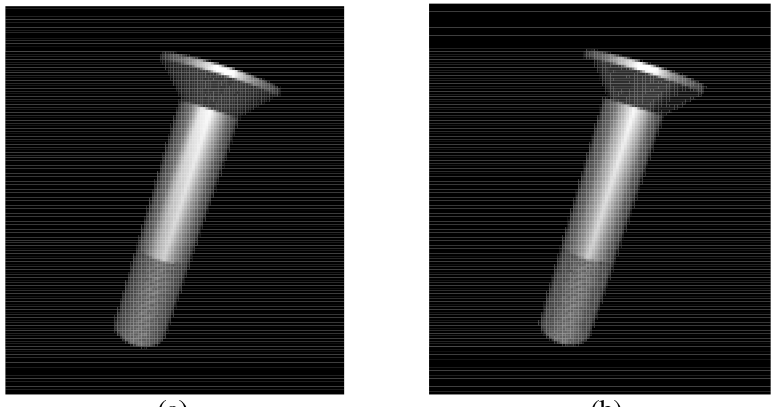
The models used in this paper are created using a commercial CAD system called ProEngineer. Table 1 shows the comparison between file sizes compressed by different approaches. Column A shows the size of VRML file in ASCII which contains the normal vectors only, and the column B is the size of compressed files using an application called WinZip. Min Required Memory, denoted as MRM, means the theoretical minimum memory size to store normal vectors in a binary form.

Table 1. Comparison of file size

	VRML File (A)	Zip of A (B)	MRM (C) (calculated)	Proposed Algorithm (D)	D/A	D/B	D/C
model 1	68,392	18,730	41,892	5,934	8.68%	31.68%	14.16%
model 2	132,491	36,888	77,952	10,706	8.08%	29.02%	13.73%
model 3	139,411	38,637	82,524	11,450	8.21%	29.63%	13.87%
model 4	150,678	40,990	89,736	11,781	7.82%	28.74%	13.13%
model 5	158,900	43,664	92,796	13,001	8.18%	29.78%	14.01%
model 6	205,110	54,436	122,172	16,218	7.91%	29.79%	13.27%
model 7	210,446	55,578	124,500	17,190	8.17%	30.93%	13.81%
model 8	291,132	77,532	168,348	24,789	8.51%	31.97%	14.72%

The compression ratio of the proposed algorithm is approximately 10 % and 30 % of VRML files in ASCII and WinZip compressed files of corresponding VRML files, respectively.

Fig. 8 shows the visual difference between the original model and the model with normal vectors compressed by the proposed algorithm. Note that original model and compressed model have the same geometry information and the same topology information. Fig. 8 (a) shows the visualization of original model 5 with 2,861 normal vectors and Fig. 8 (b) shows the visualization of model 5 compressed by the proposed algorithm and the normal vectors in model 5 are represented by 64 cluster means. It is not easy to find the remarkable distinction between (a) and (b).



(a) (b)

Fig. 8. Difference of visual quality (a) model 5 before compression (158,900 bytes) (b) model 5 after compression (13,001 bytes)

7. Conclusion

Normal vectors are necessary to create more realistic visualization in a shape model. When the normals exist in the definition of model, the file size could be significantly

large compared to the size of geometry and topology data. Hence, the file size can be an obstacle to transmit the model data seamlessly through a network.

The proposed compression technique obtains a significant compression ratio of roughly 10% for original normal vectors without a serious sacrifice of the visual quality of the model. In particular, we have presented techniques related to the absolute and relative indices for compressing normal vectors.

However, there are a few issues to be further studied. Using a better algorithm than K-means algorithm for the clustering may be one of the important topics. In addition, we believe that the mean values of the representative normal vectors in the clusters could be also compressed.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Ceramic Processing Research Center(CPRC) at Hanyang University.

Reference

1. Carey, R., Bell, G., Marrin, C.: ISO/IEC DIS 1477-1: 1997 Virtual Reality Modeling Language(VRML97). The VRML Consortium Incorporated (1997)
2. Chow, M.M.: Optimized Geometry Compression for Real-Time Rendering. In Proceedings of IEEE Visualization '97 (1997) 347-354
3. Deering, M.: Geometry Compression. In Proc. SIGGRAPH' 95 (1995) 13-20
4. Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics : Principles and practice 2nd edn. Addison Wesley (1987)
5. Gumhold, S., Strasser, W.: Real-Time Compression of Triangle Mesh Connectivity. In Proceedings of ACM SIGGRAPH '98 (1998) 133-140
6. Hoppe, H: Efficient implementation of progressive meshes: Computers and Graphics, Vol. 22, No. 1 (1998) 27-36
7. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall (1988)
8. Kim, Y.S., Park, D.G., Jung, H.Y., Cho, H.G.: An Improved TIN Compression Using Delaunay Triangulation. In Proceedings of Pacific Graphics '99 (1999) 118-125
9. Lee, E.S., Ko, H.S.: Vertex Data Compression For Triangle Meshes. Eurographics 2000, Vol. 19, No. 3 (2000) 1-10
10. Rossignac, J.: Edgebreaker: Connectivity Compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 1 (1999) 47-61
11. Taubin, G., Rossignac, J.: Geometric Compression Through Topological Surgery. ACM Transactions on Graphics, Vol. 17, No. 2 (1998) 84-115
12. Taubin, G., Horn, W.P., Lazarus, F., Rossignac, J.: Geometric Coding and VRML. Proceedings of the IEEE, Vol. 86, Issue 6 (1998) 1228 -1243
13. Touma, C., Gotsman, C.: Triangle Mesh Compression. In Proceedings of Graphics Interface '98 (1998) 26-34