7th WSEAS Int. Conf. on MATHEMATICAL METHODS and COMPUTATIONAL TECHNIQUES IN ELECTRICAL ENGINEERING, Sofia, 27-29/10/05 (pp223-228)

An Efficient Compression Method for Triangular Meshes Used in Engineering

SEBASTIAN KRIVOGRAD, MLADEN TRLEP, BORUT ŽALIK
Faculty of Electrical Engineering and Computer Science
University of Maribor
Smetanova ulica 17, SI - 2000 Maribor
SLOVENIA

Abstract: - This paper presents a new approach for the lossless compression of engineering data, represented by triangles. The method works in two steps - a topology compression followed by an entropy compression. The topology compression is based on the five states; one is automatically recognized by the compressor/decompressor and the remaining is coded by four commands. This approach for topology compression has been compared with other methods and turns out to be highly competitive. Geometric data and application-specific engineering data are also prepared for compression during the topology compression. Namely, the compressed topological data contributes just a few percent to the normal size of the geometric and application-specific data. General purpose compression methods are usually used for these. We compared our method with the popular PkZip and we achieved considerably better results.

Key-Words: - FEM triangular mesh, compression, topology, geometry

1 Introduction

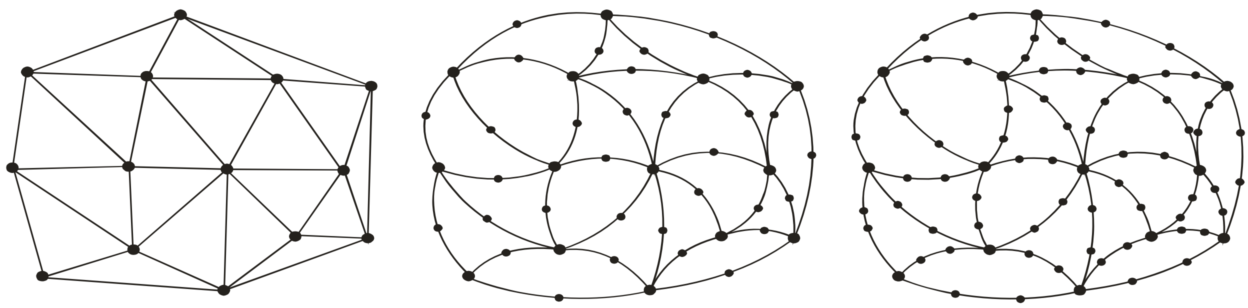
Free-form geometric shapes produced by various CAD systems, 3D scanners or quality mesh generators are usually represented by triangular meshes [1]. Triangles are also very popular in the Finite Elements Method (FEM), which is the most popular and investigated method for various engineering and scientific calculations [2]. Today triangles are accepted directly by most graphic accelerators and, therefore, triangular meshes can be manipulated and visualized in real-time even on low-cost machines. However, a problem appears when such huge triangulated models have to be stored or, even worse, when they have to be transferred over the internet. Compression methods have to be applied, in such cases.

Data compression is amongst the oldest challenges in computer science. Many different approaches have been suggested up to now [3]. However, the classical methods, which minimize the redundancies in alphanumeric streams, are not optimal for compression of the triangular meshes. Therefore, the compression of triangular meshes has become a popular topic over the last few years, started by Turan's pioneering work [4]. A further achievement in this area was the work of Touma-Gotsman [5]. Their method was later ameliorated by Alliez-Desbrun and Isenburg [6, 8]. They improved the compression rate by a simple heuristic (it will be briefly explained in the next section). Up to now, methods for compressing triangular meshes have been intended for describing the boundaries of geometric objects and their visualization, and not for engineering applications. In FEM, for example, different types of elements may mutually exist, and each of them is equipped by additional application-specific numerical information.

In this paper we present a new method for the lossless compression of triangular meshes, as they appear in engineering (Electromagnetic FEM data were used (for the case study)). The presented method, contrary to others methods, accepts triangles of arbitrary orders (see Fig. 1) and efficiently handles engineering application-specific numerical information. Using experimentation we confirmed that the proposed method provides very promising results and that it is superior to those general-purpose compression methods, which are usually used in practice.

2 The Algorithm

The FEM triangular meshes carry three types of data: geometric data for each vertex (coordinates represented by floating-point numbers), vertex indices needed for defining the topology (each triangle has $3 \cdot o$ indices, where o defines the triangle order (indices are represented by integers)) and application-specific data associated with vertices and triangles (represented by integer and/or floating-



Fig, 1: FEM data of a) first, b) second and c) third order

point numbers). The proposed compression algorithm works in two steps:

1. Firstly, the topology is compressed and, simultaneously, geometric and application-specific data are properly arranged in the auxiliary lists.
2. Secondly, entropy coding of the data stored in the auxiliary lists is performed.

2.1 Topology compression

Each vertex in a triangular mesh generally defines more triangles. The number of triangles determined by a vertex is considered a vertex degree. At the beginning, the triangular mesh is analyzed, and the vertex degrees are determined.

In manifold triangular meshes, each edge is either shared by two triangles or is part of the mesh boundary (see Fig. 2). The border edges are considered as *active edges* and stored in the list of active edges (*ListOfActiveEdges*). They form one or more mutually non-connected loops (Fig. 2). If the mesh represents the closed surfaces, there are no bordering edges. In this case, an arbitrary triangle is selected and extracted from the mesh. Its edges become members of the *ListOfActiveEdges*. The vertices defining the active edges are named *active vertices* and are placed in the *ActiveVertices-Structure*.

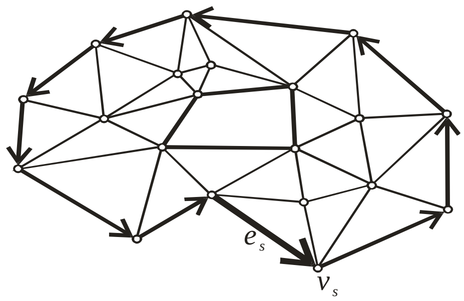


Fig. 2: Input manifold triangular mesh

The compression process starts by selecting of one of the active edges. It is named a *selected edge* e_s. The selection will be described later. The remaining active edges in the same loop are then oriented according to the first selected edge (Fig. 2). The end vertex of the selected edge is treated as *selected vertex* v_s (Fig. 2).

The main idea of the algorithm is to close the sequence of triangles around the v_s starting from e_s (Fig. 3). Configuration of the triangles around the e_s and v_s defines the characteristic states of the algorithm (the states of the algorithm will be explained later). The codes defining the states are stored in the list of commands (*ListOfCommands*). Triangles surrounding the selected vertex are compressed and removed from the triangular mesh. Their application-specific information are stored in *ListOfTriangles*. After that the *ListOfActiveEdges* and *ActiveVerticesStructure* are updated. The algorithm is terminated when the *ListOfActiveEdges* is empty.

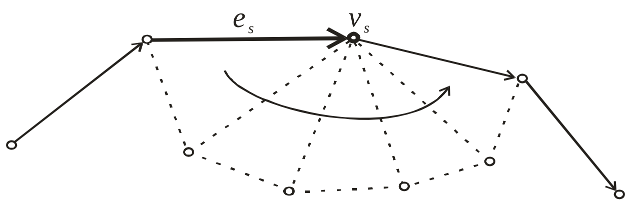


Fig. 3: The compression process wraps around selected vertex

The efficiency of the algorithm is highly depended on the lengths of the same commands. To increase this length, Alliez-Desbrun proposed a heuristic for selecting the e_s as follows: for all active vertices, the number of non-compressed surrounded triangles t_s are determined [6]. If there are more vertices with the same t_s, their immediate neighbours are visited. For each neighbour, the number of non-compressed surrounding triangles is determined and this number is added to the corresponding t_s. This procedure continues until only one vertex with minimal t_s remains, and it determines the next selected edge e_s. Our implementation is simpler and also faster. The hash table is used for selecting the next selected edge. Vertex position in the hash table depends only on the number of non-compressed triangles surrounding the vertex. Selection of the next e_s is now very fast: the first vertex in the hash table with the lowest number of non-compressed triangles, determines the next e_s.

2.1.1 States of the compression process

The presented approach uses five different states. One of them is solved automatically, the rest are expressed by only four commands (Gumhold-Strasser, for example, uses seven commands in his approach [7]). The reason for reducing the number of commands is the wish for short codes describing the commands and for long series of the same commands, for efficient entropy coding.

- **ADD.** This command is used when more than one triangle originates in vertex v_s regarding the edge e_s (Fig. 4a). In this case we compress all uncompressed triangles surrounding vertex v_s in one step. In the situation shown in Fig. 4, command **ADD** inserts coordinates of the vertices and all application-specific information related to the vertices v_i, v_j and v_k into *ListOfVertices*, and their vertex degrees into *ListOfDegrees*. When there are triangles of second and higher orders then the inner vertices are inserted in the *ListOfVertices* in the order in which they are met. In Fig. 4 we have triangles of second order so additionally stored vertices are v_{m1}, v_{m3}, v_{m2}, v_{m4}, v_{m5}, v_{m6}.. In this way, the topological data (stored in *ListOf-Commands*, *ListOfDegrees* and *ListOfNumbers*) do not change. At the end the compressed triangles are marked as used and their application specific data are inserted into *ListOfTriangles*. The *ListOfActiveEdges* and the hash table storing active vertices are updated to describe the situation shown in Fig. 4b. The command **ADD** is the most frequently used, all following commands just solve special cases.

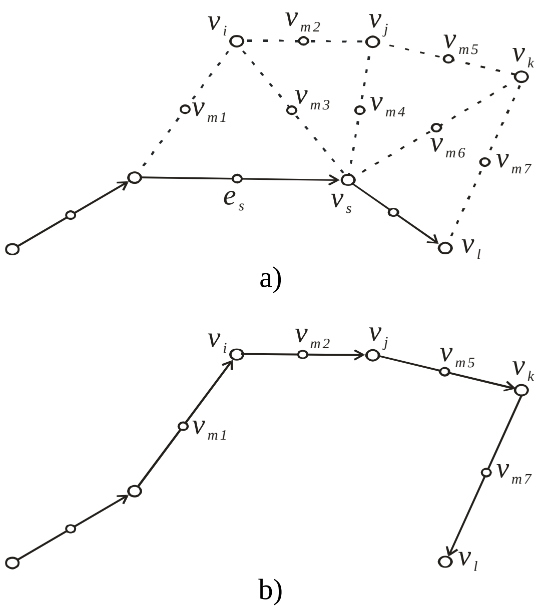


Fig. 4: Command **ADD**: beginning state (a), ending state (b)

- **ADD ONE.** It is used when more than two triangles originate in the selected vertex v_s, and any of the vertices around v_s has already been used, except the first one. In Fig. 5a, vertex v_j has already been used and, therefore, the command **ADD** cannot be applied. In such a case, only the first triangle determined by vertex v_i is compressed.

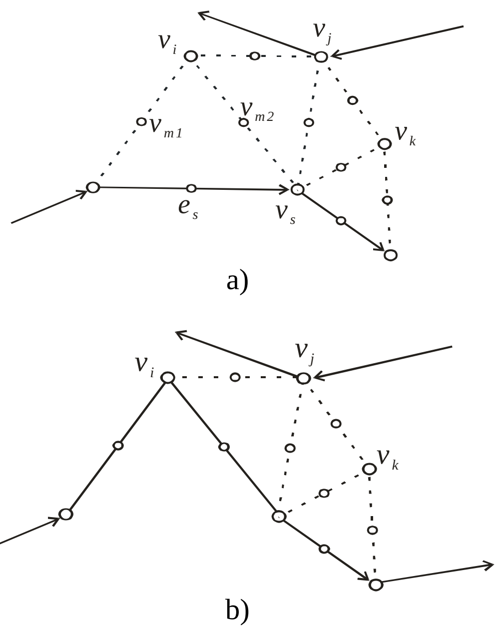


Fig. 5: Command **ADD ONE**: detection of the state (a), result (b)

- **SPLIT MERGE.** This command is applied in two cases:
  - The loop has to be split when:
    - more than one triangle originates in the selected vertex v_s,
    - the vertex of the first triangle has already been used (vertex v_i in Fig. 6a), and
    - v_i is a member of the same loop as v_s.

  Because the decompressing algorithm does not know which vertex is vertex v_i, its variable index is inserted into *ListOfNumbers*. The loop is split by inserting a chain of vertices starting from vertex v_u and ending in vertex v_v (Fig. 6a). The number of vertices in the chain is inserted into *ListOfNumbers* and the information about the vertices into *ListOfVertices* and *ListOf-Degrees*. In this way, triangles shaded in Fig. 6a are compressed causing division of the loop (see Fig. 6b). All information about

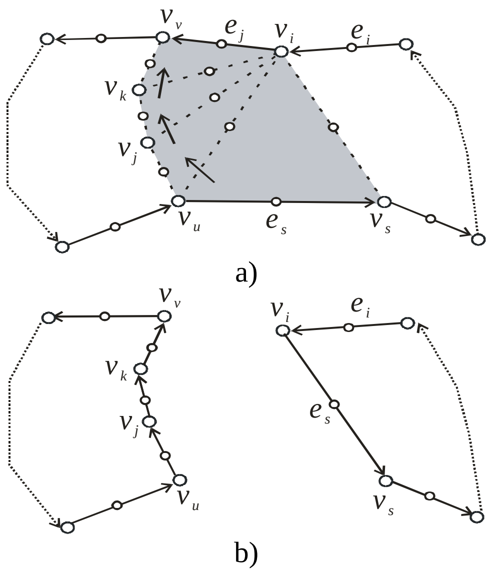


Fig. 6: Split by command **SPLIT MERGE**: chain of triangles (a), result (b)

compressed triangles are inserted into *ListOfTriangles* as they are processed.

- The equivalent situation is shown in Fig. 7, where two loops are merged. In this case v_i is not a member of the same loop as v_s. Both loops have to be oriented in the same way.

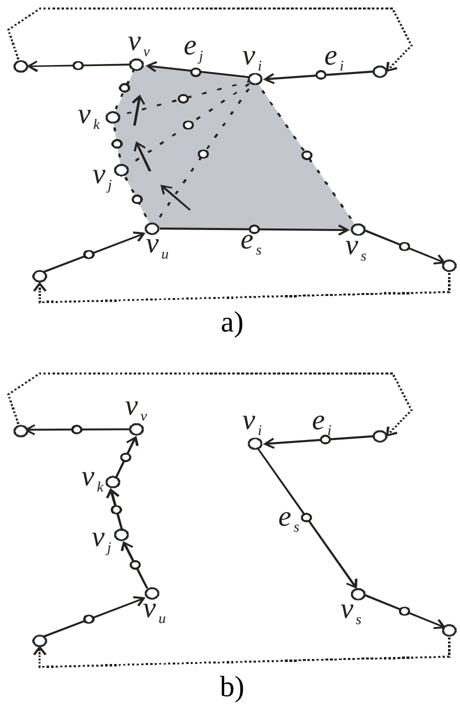


Fig. 7: Merge by command **SPLIT MERGE**: chain of triangles (a), result (b)

- **SKIP.** This command can only be caused by the command **SPLIT MERGE**. Namely, it could happen that a vertex in the chain of vertices has already been used. In this case, the selected edge is left in *ListOfActiveEdges* and the next edge from this list is selected.
- The algorithm is able to automatically close the round around any vertex when its vertex degree number becomes 1. Fig. 8a shows this case and the result is presented in Fig. 8b. In this way, the number of used commands is reduced.

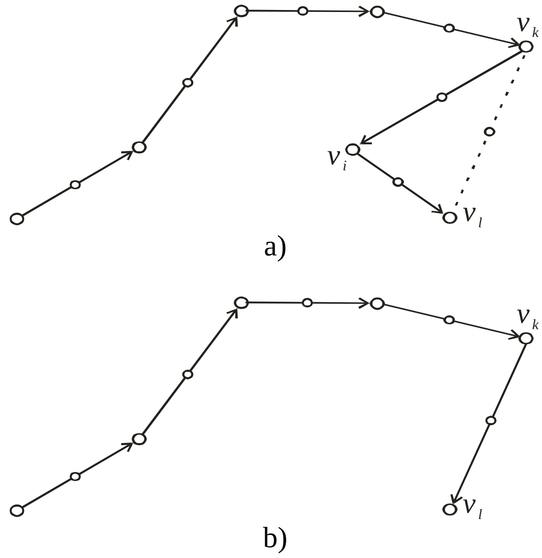


Figure 8: Automatic closure: detection of the state (a), result (b)

These commands are also sufficient for processing non-manifold triangular meshes. Namely, such meshes can be transformed into manifold meshes as described in [7].

2.2 Entropy coding

In the second step all lists, except *ListOfActiveEdges* and *ActiveVerticesStructure*, are compressed further. Namely, *ListOfActiveEdges* and *ActiveVertices-Structure* change dynamically during compression and these changes must be reproduced exactly by the decompressor from the information obtained from other list.

The lists which are compressed, store either integer or floating-point numbers. In engineering, floating-point numbers are represented by exponential notation consisting of mantissa and exponent. In our case, the exponents and the mantissas are considered individually. The exponents are integer numbers so they do not need any additional preprocessing. Mantissas are floating-point numbers represented by 32(64) bits. We consider the 32 bit floating-point number as 32 bit unsigned integer. For example 1.91452 is represented by hexadecimal code 0x3FF50F1F what is considered as unsigned integer 1073024799, or -7.48013 is converted into 3236912442 (0xC0EF5D3A) [9]. In this way, all floating-point numbers can be compressed by the same algorithms as integer numbers.

Ultimately, in the end each list is sent through the entropy coding algorithm. The minimum value of the whole list is found, stored and subtracted from all other elements in the list. If it happens that all elements are equal, all differences are 0. In this case, entropy coding is terminated, otherwise each number is divided into bytes and coded either by Huffman coding, adaptive Huffman coding, arithmetic coding, or RLE [3].

3 Results

Two types of tests were performed:

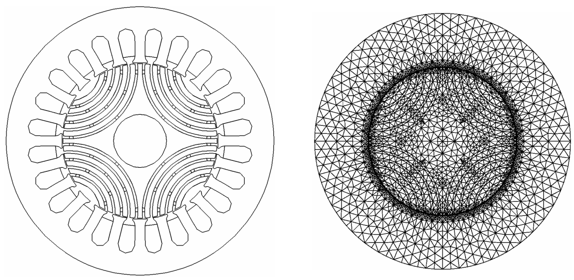
- Firstly, the efficiency of our topology compression algorithm was tested against the Touma-Gotsman and Isenburg algorithms [5, 8]. These two methods are considered as the most efficient algorithms for topology compression available at the moment. This comparison is possible as triangles of the second and the higher orders do not influence the topological information.
- A comparison of compression of the complete data set of the engineering data (triangular FEM meshes from electromagnetic analysis were used) was made against the popular and widely used *PkZip* package (WinZip 9.0 [10]). Namely, Touma-Gotsman and Isenburg implementations used lossy compression of geometric data, which is enough for visualization [5, 8]. They also do not support the compression of engineering application specific data.

3.1 Comparison of topology compression

For comparison of topology compression the non-engineering data were used, where the total number of bytes needed for compressing the topology were compared. The proposed implement-tation is, on average, 4% better than the Touma-Gotsman algorithm and around 3% worse than the Isenburg algorithm (see Table 1) [5, 8].

However, topological data represent just a small part of data being compressed. Geometric data and engineering-specific data require much more space. FEM data from electromagnetic (see Fig. 10) was used in the experiments. In this case, vertices and triangles are associated with the following data:

- vertices: geometric data (x and y coordinates represented by floating-point numbers), value of unknown function of electric or magnetic potential in this vertex (floating-point number), and type of boundary condition (integer number).
- triangles: indices of vertices defining the triangle (topology information compressed according to the description in Section 2), type of material covered by this triangle (integer number), property of the used material (floating-point number) and source-value of the electro-magnetic field (two floating-point numbers).



Model 2 Triangulated model 2

Fig. 10: Real FEM examples

The results are summarized in Table 2, where **DAT**: represents the original size of the input data stored in ASCII file, **DAT.ZIP**: means the size of compressed input file with *PkZip*, **BDAT**: represents the size of the input data using binary file. **BDAT.ZIP**: means the compressed size of the input binary file using *PkZip*, and **CDAT**: represents the compressed input data using the proposed method.

As can be seen in Table 2, the proposed method is very efficient as the compressed file is less than 5% of the original data stored in ASCII. It is also much better than *PkZip*. If the input data are in ASCII format, the proposed method gives 360% better results that *PkZip*. 225% better compression rates were also achieved than *PkZip* if the input data are in binary file.

Name	**No. of vertices**	**No. of triangles**	**TG**		**Isenburg**		**Our method**		**% of TG**	**% of Isenb.**
			B	**b/v**	**B**	**b/v**	**B**	**b/v**		
Snail	760	720l	87	0.9158	75	0.7895	52	0.5474	59.77	69.33
Puma	1,204	752	499	3.3156	328	2.1794	303	2.0133	60.72	92.38
Triceratops	2,832	5,660	767	2.1667	671	1.8955	747	2.1102	97.39	111.33
Teeth	29,152	58,300	8,176	2.2437	7,869	2.1594	8,132	2.2316	99.46	103.34
Earthing	46,625	59,680	10,274	1.7628	6,561	1.1257	5,630	0.9660	54.99	85.81
0300	90,000	178,802	32	0.0028	30	0.0027	23	0.0020	71.88	76.67
Male	109,961	219,918	27,019	1.9657	26,172	1.9041	27,005	1.9647	99.95	103.18

Table 1: Comparison of the total number of bytes needed for compression of all topological data

No. Of. Vert.	No. Of. Triang.	Order	Sizes [B] DAT	DAT.ZIP	BDAT	BDAT.ZIP	CDAT
7,771	15,482	1	1,682,170	275,267	557,840	173,747	75,959
31,023	15,482	2	3,611,769	680,534	1,115,656	456,071	233,958
54,275	15,482	3	6,733,451	1,299,409	1,673,472	734,455	373,982
10,832	21,470	1	2,336,628	340,192	774,480	226,990	82,202
43,133	21,470	2	5,016,487	901,365	1,548,936	626,695	292,422
75,434	21,470	3	9,349,508	1,763,432	2,323,392	1,019,128	487,178
13,971	27,564	1	3,004,359	456436	995,336	304,878	103,729
55,505	27,564	2	6,449,453	1,201,030	1,990,648	824,258	373,223
97,039	27,564	3	12,209,755	2,341,647	2,985,960	1,333,650	623,056

Table 2: Comparison of total sizes of FEM data

4 Conclusion

This paper introduces a new efficient approach for the compression of triangular meshes, specialized for engineering data. In this case, vertices and triangles carry additional engineering information. In addition, triangles can have different orders. The compression is divided into two parts. Firstly, topology compression is performed independently of application specific data. During topology compression these data and geometric data are prepared for the second step - entropy coding.

The topology compression was compared with the Touma-Gotsman and Isenburg approaches [5, 8]. The presented algorithm is aligned between the Touma-Gotsman and Isenburg approach (over 150 examples were estimated in our tests).

The proposed methods for triangular mesh compression have not been applied to engineering data up to now. Usually, general methods such as *PkZip* have been applied, when compression was needed. However, geometric and application-specific data require much more memory space than topology, when topology is properly compressed.

The aim of this work was to develop a method suitable for engineering applications. As shown by the experiments, this proposed method achieves stimulative results. It compresses the input ASCII file describing the triangular mesh equipped with engineering data to 5% of the initial data size. Comparison with popular *PkZip* was also carried out. The proposed method is better by 360% when the ASCII file is compressed and by 225% when data are stored in binary files.

In the future the proposed method will be extended to other types of finite elements, such as rectangles, cubes and tetrahedral.

Acknowledgements:

We are grateful to the Slovenian Research Agency for supporting this research under the project Z2-6661-0769-04/2.12 - Compression of elements appearing in the final elements methods (FEM)

References:

[1] Simpson B., Hitschfeld N., Rivera M.-C., Approximate Shape Quality Mesh Generation. *Engineering with Computers*, Vol.17, No.3, 2001, pp.287-298.

[2] Trlep M., Hribernik B., Unified Approach to Solving a Steady-state Electromagnetic Field. *IEEE Transactions on Magnetics*, Vol.33, No.2, 1997, pp. 1974-1977.

[3] Salomon D., *Data Compression: The Complete Reference*. Springer-Verlag, New York, 1997.

[4] Turan G., On the Succinct Representation Of Graphs. *Discrete Applied Mathematics*, Vol.8, No.3, 1984, pp. 289-294.

[5] Touma C., Gotsman C., Triangle Mesh Compression. *Graphics Interface'98 Conference Proceedings*, 1998, pp. 26-34.

[6] Alliez P., Desbrun M., Valence-Driven Connectivity Encoding for 3D Meshes. *Computer Graphics Forum*, Vol.20, No.3, 2001, pp. 480-489.

[7] Gumhold S., Strasser W., Real Time Compression of Triangle Mesh Connectivity. *Computer Graphics (SIGGRAPH'98)*, Vol.32, 1998, pp. 133-140.

[8] Isenburg M., Compressing Polygon Mesh Connectivity with Degree Duality Prediction. *In: Graphics Interface'02 Conference Proceedings*, 2002, pp. 161-170.

[9] IEEE Standard 754 Floating Point Numbers. *http://research.microsoft.com/hollasch/cgindex/coding/ieeefloat.html*

[10] http://www.winzip.com.