Progressive Geometry Compression for Meshes

Xinguo Liu[1], Hujun Bao[1], PhengAnn Heng[2], TienTsin Wong[2], Hanqiu Sun[2], Qunsheng Peng[1]
[1]State Key Lab of CAD&CG, Zhejiang Univ. China
[2]Department of Computer Science and Engineering, The Chinese University of Hong Kong

Abstract A novel progressive geometry compression scheme is presented in the paper. In this scheme a mesh is represented as a base mesh followed by some groups of vertex split operations using an improved simplification method, in which each level of the mesh can be refined into the next level by carrying out a group of vertex split operations in any order. Consequently, the PM representation can be effectively encoded by permuting the vertex split operations in each group. Meanwhile, a geometry predicator using Laplacian operator is designed to predict each new vertex position using its neighbors. The correction is quantized and encoded using Huffman coding scheme. Experimental results show that our algorithm obtains higher compression ratios than the previous work. It is very suitable for progressive transmission of geometry models over Internet.

Keywords Triangular Mesh Geometry Compression Manifold Surface Progressive Mesh

1 Introduction

Polygonal meshes more and more widely used as 3D geometry model in computer graphics applications are always expensive to store in disk or transmit over network, since they consist of large number of geometry and vertex connectivity information. Therefore it is very significant to compress polygonal meshes. Recently, many peoples have contributed to this issue [1~7]. With the rapid development of Internet, it is useful and necessary to share 3D models through Internet. However, due to the limit bandwidth huge models are very awkward and time consuming to transmit, especially when people prefer a simplified one rather the whole model. So it is much better to transmit 3D models in a progressive way so that people can get the data he really want. Many compression methods before are not applicable for this kind of progressive transmission. To address this problem, we develop a progressive geometry compression method. It first converts the mesh into a special progressive mesh (PM), which is mutiresolution representation suitable for progressive transmission [8]. In PM, the original model is represented as a very simple base mesh followed by sequence of vertex split operations. Then it encodes the operation (edge collapse / vertex split) to save the data.

2 Simplification Scheme for Progressive Compression

Among the large number of simplification methods, any gradually edge collapse method can be used to create PM. However, in order to meet the requirement of progressive compression and transmission, the gradually edge collapse method should own the following advantages: (1) very fast; (2) The simplified meshes approximate the original very well; (3) The resulted PM can be effectively encoded; (4) The position of each split vertex can be predicted as accurate as possible. We found that the key point for such method is to determine edge collapse order. In our method the collapsed edge is determined in step. Vertices are sorted firstly according to its local size measured by the weighted average of its minimum and maximum incident edge. Then under the constraint of independent vertex set, starting from the vertex with small local size, we continue to select an incident edge with smallest collapse volume to collapse as illustrated in Fig 1. So the vertex V is merged to the corresponding neighbouring vertex U_k. This process is stopped due to the constraint or a satisfactory base model is obtained. If the former case, we re-evaluate and re-order the remained vertices and restart the simplification again, other wise the simplification is finished. Experiments show that this simplify algorithm can produce simplified models very closely to the original model.

3 Progressive Geometry Compression

Starting with the base mesh, we can recover the original model by a sequence of vertex-split operations. Since the base model are very small and consists of a little number of vertices and triangles and can be compressed using any single resolution geometry compression method, the most important point for the whole compression is to efficiently encode vertex split operation. Each split operation has three parameters: the split vertex U, left vertex L and right vertex R of the new edge, as illustrated in Fig 2. Since L and R are correlated and limited in the neighbours of U, so the number of the possible combination are $m(m+1)/2$, and L, R can be encoded with average 5 bits or so assuming that vertices degree m are average 6. The first parameter U is more difficult to encode than R and L. Reviewing our simplification method, under the constraint of independent set, very time when we re-evaluate the remained vertices, the decimated group of vertices $\{V_0, V_1, \ldots, V_N\}$ belongs to an

independent vertex set. . Let $\{U_0, U_1, \ldots, U_N\}$ be the corresponding split vertex group. This shows that we can split the vertex of the group in any order to recover the model. Therefore, without loss of generality, let $U_i<U_{i+1}$. If we encode the split vertex group by group, then the encoding problem is converted to encode a monotonic increasing integer sequence. One of the most efficient and simplest way is to encode the difference sequence $\{U_0, U_1 - U_0, \ldots, U_{i+1} - U_i, \ldots, U_N - U_{N-1}\}$. A statistic results show that the vertex number of an independent set is about ¼ of the original model's vertex number. Therefore, due to our greedy scheme in simplification step, the differences $U_{i+1} - U_i$ are very small and about 4 or so. With Huffman code, each split vertex can be encoded using less than 2 bits. Implementation result in Table 1 also shows that the average cost for connectivity is about less than 3.5bits per triangle.

Another part we need to deal with in geometry compression is the position of the split vertex, which is the major data cost for meshes and therefore is very crucial to a progressive compression scheme. During our simplification step, when a vertex is decimated, its neighbours are locked. So when the vertex is to be recovered, all its neighbours are already known and can help to determine the split vertex position. Consider a split vertex U, let V be a new vertex split out from U, we first predict the position of vertex V using the neighbour vertices with the following formulation:

$$w = \frac{1}{m}\sum_{j=1}^{m}\left(u_j + \frac{1}{m_j}\sum_{k=1}^{m_j}(u_{j,k} - u_j)\right)$$

Where m and m_j are respectively the degree of V and U_j, U_j and $U_{j,k}$ are respectively the 1-ring neighbors of vertex U and U_j. Then the correction term V-W is quantized using the user specified precision, for instance 10 bits precision in Table 1. Finally the quantized correction terms are encoded using Huffman algorithm. Fig. 3 shows that our prediction method is very effective, and the correction terms become less and less as more and more vertices are recovered. Therefore the average bit cost per vertex decrease. Other geometry properties, such as color, normal, texture coordinate etc. if any, can be compressed in a similar way. Although the progressive geometry compression method in the paper is very efficient, there are still some overhead bits for vertex connectivity compared with the single resolution compression methods, which is undesirable for progressive transmission especially when the whole model is transmitted. Therefore, How to reduce those overheads is worthwhile for further study. Additionally, as the main cost of mesh, geometry information also needs more efficient compression.

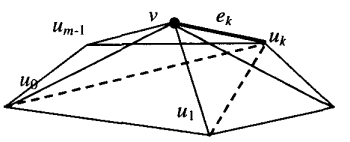


Fig 1. The collapse volume of an edge

$$collapsed\_volume(e_k) = \sum((u_j - u_k) \otimes (u_{j+1} - u_k)) \cdot (v - u_k)$$

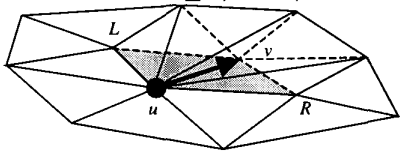
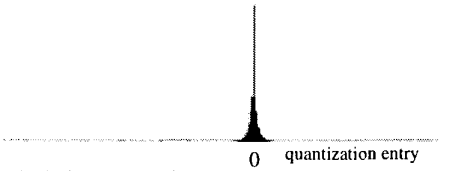


Fig 2. Vertex split. *v* is split frome vertex *u and introduce a new edge* <*u*,*v*> whose right and left vertex *right* and *left*.

(a) Distribution of the quantized correction coordinates;



(b) the histogram of the corrections in (a).

Fig 3. Quantized vertex position correction of the bunny model (10 bits quantization precision).

Table 1. Compress result for a Bunny model. dV and dT are the increment of vertices and triangles, respectively. c/dT is the connectivity bits per triangle, and g/dV is the connectivity bits per vertex. c+g/dT is the total bits per triangle .

unny	0	1	2	3	4	5	6	7	8
DV	127	31	46	67	97	143	189	269	353
DT	236	62	92	134	193	286	376	537	702
c/dT	-	3.69	3.46	3.63	3.53	3.58	3.44	3.57	3.51
g/dT	-	16.9	15.7	15.5	14.7	12.9	12.0	11.2	10.5
C+g/dT	-	20.6	19.2	19.1	18.2	16.5	15.4	14.8	14.0

9	10	11	12	13	14	15	16	17	verage
501	710	1011	1383	1956	2713	3795	5240	8868	-
998	1412	2016	2760	3903	5411	7573	0460	7680	-
3.52	3.49	3.44	3.42	3.50	3.47	3.48	3.44	3.14	3.346
9.62	8.89	8.26	7.62	7.00	6.46	5.86	5.32	4.09	5.670
13.1	12.4	11.7	11.0	10.5	9.93	9.34	8.76	7.23	9.016

References

[1] M. Deering. Geometry compression. In *Proceedings of SIGGRAPH'95*, pp. 13-20, 1995.

[2] Gabriel Taubin and Jarek Rossignac. 3D geometric compression. In *SIGGRAPH'98 Course Notes 21*. ACM SIGGRAPH, 1998.

[3] C. Touma and C. Gotsman. Triangle mesh compression. In *Proceedings of Graphics Interface '98*, pp. 26-34, 1998.

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

[5] M. Isenburg and J. Snoeyink. Mesh collapse compression. In *Proceedings of SIGGRAPHI'99 - 12th Brazilian Symposium on Computer Graphics and Image Processing*, pp.27-28, 1999.

[6] Taubin et al. Progressive Forest Split. In *Proceeding of SIGGRAPH'98*, pp.123-132, 1998.

[7] G. Taubin. A signal processing approach to fair surface design. In *Proceedings of SIGGRAPH '95*, pp. 351-358, 1995.

[8] Hugues Hoppe. Progressive meshes. In *Proceedings SIGGRAPH 96*, pp. 99–108. ACM SIGGRAPH, 1996.