On the Non-Redundancy of Split Offsets in Degree Coding

Martin Isenburg
isenburg@cs.unc.edu

Jack Snoeyink
snoeyink@cs.unc.edu

University of North Carolina at Chapel Hill

Abstract

The connectivity coder by Touma and Gotsman encodes a planar triangulation through a sequence of vertex degrees and occasional "split" symbols that have an associated offset value. We show that the split offsets of the TG coder are not redundant by giving examples of degree sequences that have two different decodings if the split offsets are not specified. Surprisingly, such examples are rare and a large number of encodings remain unique.

1 Introduction

Recent years have seen a number of schemes that compactly encode triangle mesh connectivity by a sequence of symbols that specify how to grow a "compression boundary" enclosing an already encoded region, one triangle at a time. A popular scheme is the Triangle Mesh Compression method by Touma and Gotsman [7], or TG coder for short. For planar triangulations, the TG coder generates a sequence of vertex degrees that usually contains a few "split" symbols with associated offset values.

There has been speculation that it might be possible to modify the TG coder to operate without explicitly storing the offsets values. The Cut-border Machine [3] and Dual-Graph Method [5] explicitly include split offsets, but the otherwise identical Edgebreaker [6] and Face-Fixer [4] schemes avoid them, getting by only with the "end" symbols in the code sequence. However, the TG coder does not store explicit "end" symbols. It maintains more state information on the compression boundary than Edgebreaker or Face Fixer that—together with explicit offsets—makes "end" symbols implicit. But if we omit the offsets we can find sequences with two valid decodings even if we add explicit "end"s to the code sequence.

2 Connectivity coding with the TG Coder

To encode a triangulation, the TG coder [7] grows an encoded region, maintaining one or more compression boundaries into which it includes triangle after triangle. It usually includes the triangle adjacent to the *gate* edge, which advances in clockwise order around the *focus* vertex. However, it immediately includes any triangle that shares two edges with the compression boundary.

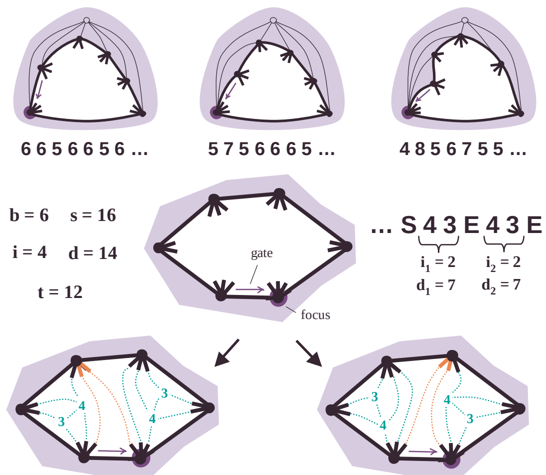


Figure 1: The smallest scenarios where an offset-less encoding with "split" and "end" symbols is not unique occur in triangulations of 11 vertices. The top right example, however, is unique.

The encoder starts with an compression boundary of length two around an arbitrary edge, records the degree of the two initial boundary vertices, and sets their *slot count* to be one less than their degree. Whenever all boundary vertices have a slot count of one or higher, the triangle adjacent to the gate shares only one edge with the compression boundary. Usually the third vertex of the triangle at the gate is a previously unprocessed vertex, and the encoder simply "adds" this vertex as a new boundary vertex and records its degree. Occasionally, the third vertex of this triangle is already on the boundary, and the encoder splits the compression boundary into two loops, temporarily stores one on a stack, and continues encoding on the other. Here the encoder records a split symbol and *offset*, which is the number of slots that are on the boundary part that is stored on the stack, from which it can derive how many slots are clockwise along the boundary between the gate and the split vertex.

In the full paper we consider four offset-less encodings, each with less information about the split operations that occur during the encoding process. The strongest is if we omit the offset, but still record "end" symbols that mark the completion of a boundary loop. Figure 1 illustrates the smallest examples for which this encoding is non-unique: in triangulations with 11 vertices there are two possible ways of splitting the boundary of length 6 that has its 16 slots distributed in this particular configuration and where both resulting boundary parts still enclose two unprocessed vertices of degree 3 and 4. However, there are not always two valid decodings in this scenario. Sometimes the focus is already connected to other vertices of the boundary through previously decoded triangles. This places additional constraints on the possibilities for splitting the boundary. The top-right triangulation from Figure 1, for example, has only one valid decoding.

Theorem. *For the TG coder without split offsets, but with end symbols, there exist two different triangulations that have the same offset-less encoding.*

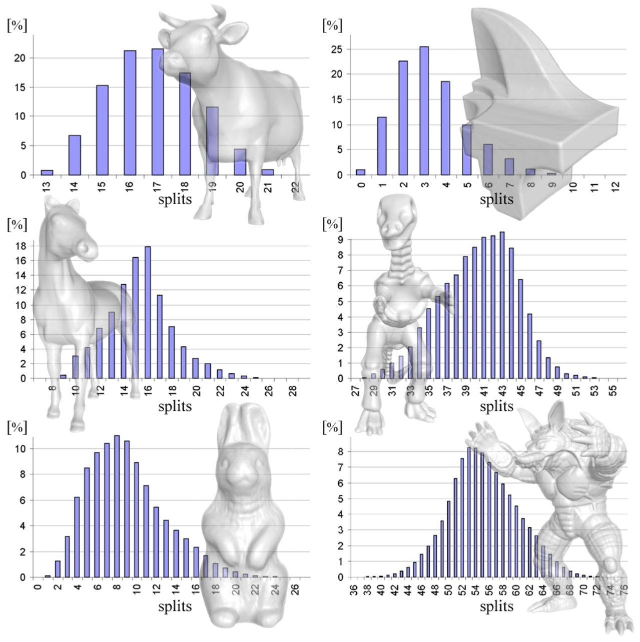
3 Searching for valid decodings

In order to find all valid decodings of an offset-less encoding we search through all possibilities of performing split operations. For each attempt it recursively starts to decode the first boundary part and in case this is successful does the same for the second. Only if both recursions are successful it returns a success, otherwise it tries out the next possibility or returns a failure if there are none left. A few observations help us to immediately eliminate some splits from further consideration.

Initial experiments seemed to indicate that the split offsets of the TG coder might in fact be replaced by "end"s. On our standard set of example meshes the search for split offsets would find the correct answer every run we tried. Table 1 shows that non-unique encoding are surprisingly rare. The full paper has further experiments showing that only a small fraction of random triangulations have non-unique encodings.

4 Closing discussion

There have been attempts to establish a guaranteed bound on the coding costs of the TG coder. However, the infrequently occuring "split" symbols and their offsets made this a difficult task. Our work shows that these split offsets are not completely redundant. There remains the task of determining if any degree-based coder can avoid offsets. Alliez and Desbrun [1] suggested an adaptive traversal heuristic that lowered the number of split operations and the remaining number of "splits" seemed negligibly small. Therefore the authors restricted their worst case analysis to the vertex degrees. But Gotsman [2] has shown that the entropy analysis of Alliez and Desbrun includes many degree distribution that do not correspond to actual triangulations, and that there are fewer valid permutations of degrees than triangulations and that additional information is necessary to distinguish between. So split information does contribute a small but necessary fraction to the encoding.



meshes			splits			non-unique
name	vertices	encodings	min	max	avg	encodings
cow	2,904	17,412	13	22	16.8	0
fandisk	6,475	38,838	0	12	3.3	0
horse	48,485	290,898	7	29	15.4	9
dinosaur	56,194	337,152	27	56	40.4	10
rabbit	67,039	402,222	0	27	9.0	56
armadillo	172,974	1,037,832	36	76	55.2	146

Table 1: We show several meshes with their histograms of splits. The table lists numbers of vertices and encodings, split statistics, and number of non-unique encodings for each mesh.

References

[1] P. Alliez and M. Desbrun. Valence-driven connectivity encoding for 3D meshes. In *Proceedings of Eurographics'01*, pages 480–489, 2001.

[2] C. Gotsman. On the optimality of valence-based connectivity coding. *Computer Graphics Forum*, 22(1):99–102, 2003.

[3] S. Gumhold and W. Strasser. Real time compression of triangle mesh connectivity. In *Proceedings of SIGGRAPH'98*, pages 133–140, 1998.

[4] M. Isenburg and J. Snoeyink. Face Fixer: Compressing polygon meshes with properties. In *Proceedings of SIGGRAPH'00*, pages 263–270, 2000.

[5] J. Li and C. C. Kuo. A dual graph approach to 3D triangular mesh compression. In *Proceedings of ICIP'98*, pages 891–894, 1998.

[6] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Transanctions on Visualization and Computer Graphics*, 5(1):47–61, 1999.

[7] C. Touma and C. Gotsman. Triangle mesh compression. In *Proceedings of Graphics Interface'98*, pages 26–34, 1998.