Mesh Collapse Compression

MARTIN ISENBURG, JACK SNOEYINK

University of North Carolina, Sitterson Hall, Chapel Hill, NC 27599, USA
{isenburg | snoeyink}@cs.unc.edu

Abstract. We present a novel algorithm for encoding the topology of triangular meshes. A sequence of edge contract and divide operations collapses the entire mesh into a single vertex. This implicitly creates a tree with weighted edges. The weights are vertex degrees and capture the topology of the unlabeled mesh. The nodes are vertices and capture the labeling of the mesh. This weighted-edge tree has a very compact encoding.

1 Introduction

Efficiently encoding the topology of triangular meshes has recently been the subject of intense study [3, 4, 5, 6, 1] and many representations have been proposed. The sudden interest in this area is fueled by the emerging demand for transmitting 3D data sets over the Internet (e.g. VRML). Since transmission bandwidth is a scarce resource, compact encodings for 3D models are of great advantage. For a survey, see the paper of Rossignac [4].

We present a novel algorithm for encoding the topology of triangular meshes. Our scheme performs a sequence of edge contract and divide operations that collapse the entire mesh into a single vertex. For edge contract operations we store the degree of the removed vertex. For divide operations we store start and end symbols. This uniquely determines the inverse operations. For meshes homeomorphic to a sphere, the algorithm is especially simple. However, the algorithm also encodes surfaces of higher genus at the expense of a few extra bits per handle. A video demonstration of Mesh Collapse Compression can be found in [2].

2 Compression Method

As input, we take a mesh, without boundary but of any genus, that is composed of topological triangles. An arbitrary vertex is chosen to be the *mc-vertex*, or *mesh collapse vertex* and an arbitrary directed edge leaving the mc-vertex is chosen to be the *mc-edge*, or *mesh collapse edge*. We cut and open the mesh along the mc-edge, which creates a new face that is bounded by only two edges. For easier illustration we arrange this face to be the outer face as shown in Figure 1a. The resulting configuration is called a *digon*. This is a triangulation with exception of the outer face, which is bounded by only two edges.

We distinguish between *simple digons*, *complex digons*, and *trivial digons*: A digon is *simple* when only the two bounding edges connect the two vertices of the outer face. A digon is *complex* when there are more than two edges. Each additional edge is a *dividing edge*. A complex digon with d dividing edges can be divided into $d + 1$ simple digons. A digon is *trivial* when it has only three vertices (see Figure 1b).

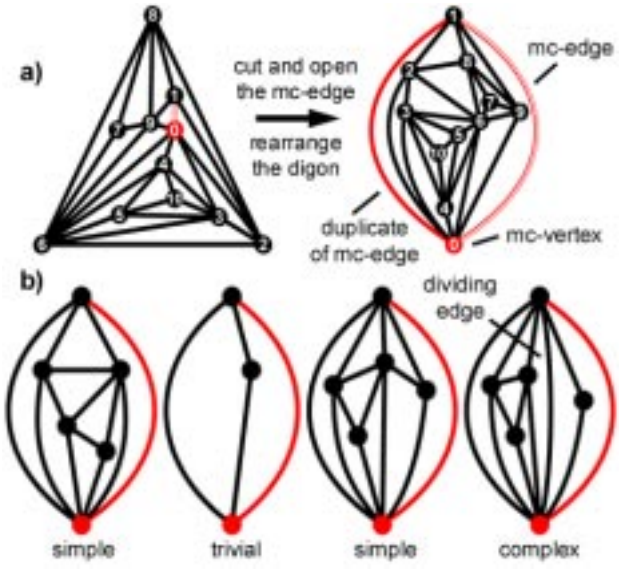

Figure 1: (a) Cutting and opening the mc-edge turns the mesh into a digon. (b) Simple, trivial, and complex digons.

We use two invertible operations that decompose the initial digon into one or more trivial digons. The *mc-contract* operation takes a simple digon as input. It first contracts the current mc-edge, then deletes the resulting loop, and finally selects the next edge counterclockwise around the mc-vertex as the new mc-edge. This removes one vertex and two triangles. The resulting digon is either simple or complex. The *mc-divide* operation takes a complex digon as input. It divides the complex digon along a dividing edge into two digons. One of the resulting digons is simple, the other is either simple or complex.

Starting with the initial digon we repeatedly apply the mc-contract operation until either a complex or a trivial digon is encountered. For each mc-contract operation we record the removed vertex and its degree. When we encounter a complex digon we apply the mc-divide operation. We record

a start symbol and mc-compress the resulting simple digon by recursion. Subsequently we continue on the resulting other digon. When we encounter a trivial digon we record its three vertices and an end symbol and stop. The recorded information is sufficient to invert each operation.

This encoding process implicitly creates an *mc-tree*, a vertex tree with weighted edges. The mc-tree tends to be skinny. Only complex digons lead to branches in the tree and the number of complex digons encountered in practice is small. For very regular meshes the mc-tree can be trivial. See Figure 2 for an example.

When a mesh has boundaries or holes, we patch them with dummy vertices in a preprocessing step. When a mesh has handles, we encode as before; whenever a handle is broken, we obtain a complex digon whose by dividing edges separated components are still connected along the handle. This can be detected, as the recursive call for encoding one component works its way along the handle and also encodes the other. The only change is that we record some additional information when an mc-contract operation encounters the other mc-edge and its duplicate. The details are in the full paper.

3 Summary and Acknowledgments

We presented a novel encoding scheme for mesh topology. Our algorithm is simpler than approaches by [4, 6, 1] and produces a code sequence similar to [6]. Subsequent run-length and/or entropy encoding results into very compact bit streams of 1 to 4 bits per vertex.

The first author thanks the International Computer Science Institute (ICSI) at Berkeley for the research opportunity during the summer of 1998. This work was done while both authors were at the University of British Columbia and was supported by NSERC, IRIS, and a UBC Graduate Fellowship.

References

[1] L. de Floriani, P. Magillo, and E. Puppa. A simple and efficient sequential encoding for triangle meshes. In *15th European Workshop on Computational Geometry*, 1999.

[2] M. Isenburg and J. Snoeyink. Mesh collapse compression video. In *Video Review of ACM Symposium on Computational Geometry*, 1999.

[3] K. Keeler and J. Westbrook. Short encodings of planar graphs and maps. In *Discrete Applied Math*, pages 239–252, 1995.

[4] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. In *Technical Report GIT–GVU–98–35, Georgia Tech*, page 15, 1998.

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

[6] C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface*, pages 26–24, 1998.

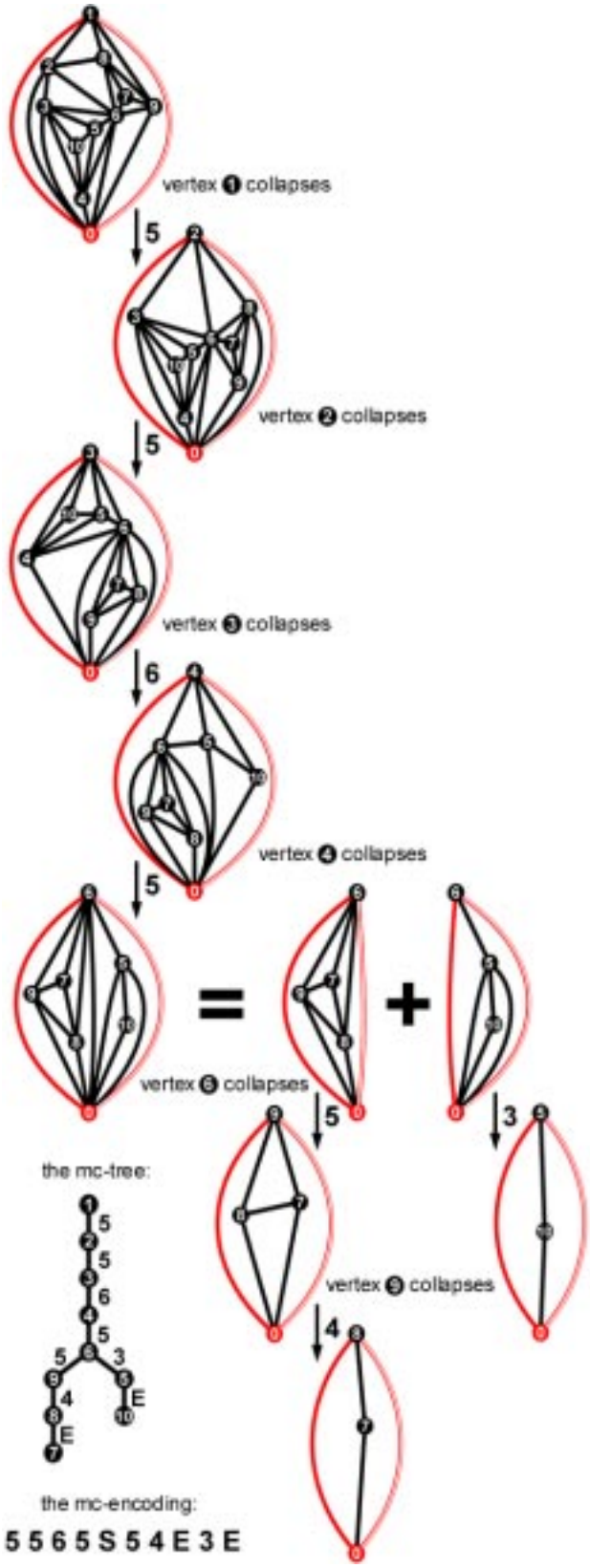

Figure 2: An example of mc-compression: The fourth mc-contract operation results in a complex digon that is divided into two simple digons.