ELSEVIER

Computer-Aided Design 37 (2005) 869–877

COMPUTER-AIDED DESIGN

www.elsevier.com/locate/cad

Lossless compression of predicted floating-point geometry

Martin Isenburg[a,*], Peter Lindstrom[b], Jack Snoeyink[a]

[a]*Department of Computer Science, College of Arts and Sciences, University of North Carolina at Chapel Hill, Campus Box 3175, Sitterson Hall, Chapel Hill, NC 27599-3175, USA*

[b]*Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550, USA*

Accepted 20 September 2004

Abstract

The size of geometric data sets in scientific and industrial applications is constantly increasing. Storing surface or volume meshes in standard uncompressed formats results in large files that are expensive to store and slow to load and transmit. Scientists and engineers often refrain from using mesh compression because currently available schemes modify the mesh data. While connectivity is encoded in a lossless manner, the floating-point coordinates associated with the vertices are quantized onto a uniform integer grid to enable efficient predictive compression. Although a fine enough grid can usually represent the data with sufficient precision, the original floating-point values will change, regardless of grid resolution.

In this paper we describe a method for compressing floating-point coordinates with predictive coding in a completely lossless manner. The initial quantization step is omitted and predictions are calculated in floating-point. The predicted and the actual floating-point values are broken up into sign, exponent, and mantissa and their corrections are compressed separately with context-based arithmetic coding. As the quality of the predictions varies with the exponent, we use the exponent to switch between different arithmetic contexts. We report compression results using the popular parallelogram predictor, but our approach will work with any prediction scheme. The achieved bit-rates for lossless floating-point compression nicely complement those resulting from uniformly quantizing with different precisions.

Keywords: Mesh compression; Geometry coding; Lossless; Floating-point

1. Introduction

Irregular surface or volume meshes are widely used for representing 3D geometric models. These meshes consists of mesh *geometry* and mesh *connectivity*, the first describing the positions in 3D space and the latter describing how to connect these positions into the polygons/polyhedra that the surface/volume mesh is composed of. Typically there are also mesh *properties* such as colors, pressure or heat values, or material attributes.

The standard representation for such meshes uses an array of floats to specify the positions and an array of integers containing indices into the position array to specify the polygons/polyhedra. A similar scheme is used to specify the various properties and how they attach to the mesh. For large and detailed models this representation results in files of substantial size, which makes their storage expensive and their transmission slow.

The need for more compact mesh representations has motivated researchers to develop techniques for compression of connectivity [6–8,10,14,19,24], of geometry [6,9,10,23,24], and of properties [2,15,16,22]. The most popular compression scheme for triangulated surface meshes was proposed by Touma and Gotsman [24]. It was later generalized to both polygonal surface and hexahedral volume meshes [8–10]. It tends to give very competitive bit-rates and continues to be the accepted benchmark coder for mesh compression [11]. Furthermore, this coding scheme allows single-pass compression and decompression for out-of-core operation on gigantic meshes [12].

While connectivity is typically encoded in a lossless manner, geometry compression tends to be lossy. Current schemes quantize floating-point coordinates and other properties associated with the vertices onto a uniform

* Corresponding author.
E-mail address: isenburg@cs.unc.edu (M. Isenburg).

0010-4485//$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cad.2004.09.015

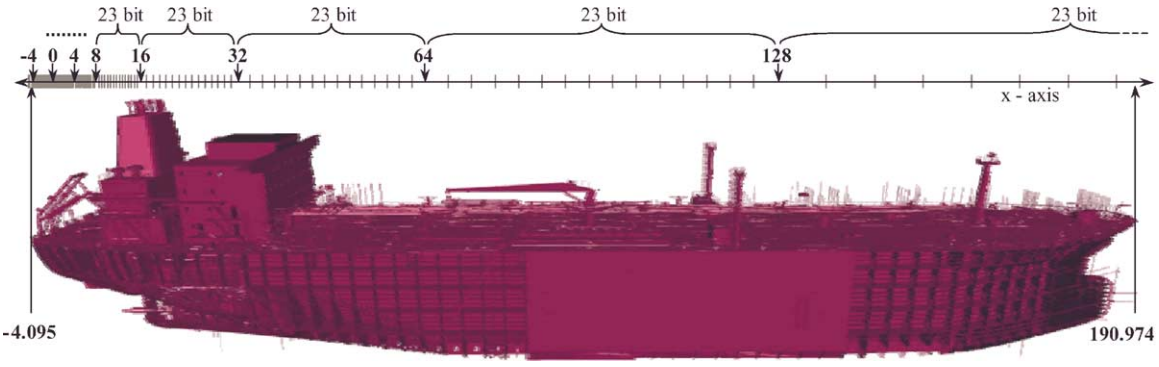

Fig. 1. The x-coordinates of this 75 million vertex Double Eagle tanker range from -4.095 to 190.974. The coordinates above 128 have the least precision with 23 mantissa bits covering a range of 128. There is 16 times more precision between 8 and 16, where the same number of mantissa bits only have to cover a range of 8.

integer grid prior to predictive compression. Usually one can choose a sufficiently fine grid to capture the entire precision that exists in the data. However, the original floating-point values will change slightly. Scientists and engineers typically dislike the idea of having their data modified by a process outside of their control and therefore often refrain from using mesh compression altogether.

A more scientific reason for avoiding the initial quantization step is a non-uniform distribution of precision in the data. Standard 32-bit IEEE floating-point numbers have 23 bits of precision within the range of each exponent (see Fig. 1) so that the least precise (i.e. the widest spaced) numbers are those with the highest exponent. If we can assume that all samples are equally accurate, then the entire *uniform* precision present in the floating-point samples can be represented with 25 bits once the bounding box (i.e. the highest exponent) is known. But if this assumption does not hold because, for example, the mesh was specifically aligned with the origin to provide higher precision in some areas, then uniform quantization is not an option.

Finally, if neither the precision nor bounding box of the floating-point samples is known in advance it may be impractical to quantize the data prior to compression. Such a situation may arise in streaming compression, as it was envisioned by Isenburg and Gumhold [12]. In order to compress the output of a mesh-generating application *on-the-fly*, one may have to operate without a priori knowledge about the precision or the bounding box of the mesh.

In this paper we investigate how to compress 32-bit IEEE floating-point coordinates with predictive coding in a completely lossless manner. The initial quantization step is omitted and predictions are calculated in floating-point arithmetic. The predicted and the actual floating-point values are broken up into *sign*, *exponent*, and *mantissa* and their corrections are compressed separately with context-based arithmetic coding [25]. As the quality of predictions varies with the exponent, we use the exponent to switch between different arithmetic contexts. We report compression results for single-precision floating-point coordinates predicted with linear predictions. However, our coding technique can also be used for other types of floating-point data or in combination with other prediction schemes. The achieved bit-rates for lossless floating-point compression nicely complement those resulting from uniformly quantizing with different precisions. Hence, our approach is a completing rather than a competing technology that can be used whenever uniform quantization of the floating-point values is—for whatever reason—not an option.

Compared to the preliminary results of this work that were reported in [13] we achieve improved bit-rates, faster compression and decompression, and lower memory requirements. Furthermore we include a detailed comparison between the proposed compression scheme, simpler predictive approaches, and non-predictive gzip compression. This comparison shows that current predictive techniques are not always the best choice. They are outperformed by gzip on data sets that contain frequently reoccuring floating-point numbers.

The remainder of this paper is organized as follows. In Section 2 we give a brief overview of mesh compression. In Section 3 we describe how current predictive geometry coding schemes operate. In Section 4 we show how these schemes can be adapted to work directly on floating-point numbers. In Section 5 we report compression results and timings. Section 6 summarizes our contributions and discusses current and future work.

2. Mesh compression

The 3D surfaces and volumes that are used in scientific simulations or engineering computations are often represented as irregular meshes. Limited transmission bandwidth and storage capacity have motivated researchers to find compact representations for such meshes and a number of compression schemes have been developed. Compression of connectivity and geometry are usually done by clearly separated, but often interwoven techniques.