Preprocessing Using Classification for Lossless Compression of 3D Geometry Data

Miryeong Ha and Hoon Yoo*
Department of Computer Science, Sang Myung University, Jongno-gu, Seoul, South Korea.
Associate Professor, Department of Media Software Sang Myung University, Jongno-gu, Seoul, South Korea.
*Corresponding author

Abstract
This paper describes a preprocessing method for lossless compression of 3D geometry data. Geometry data consists of vertices that are the three-dimensional location data in 3D space. Normally, a 3D scanner produces a huge data including 3D geometry data from an object. Thus, a compression method is required to transmit 3D data over networks and state-of-the-art techniques have been studied. In this paper, we propose a method to improve the existing methods. To do so, a simple and efficient classification method is introduced. The classification in the proposed method can reduce the number of vertices. The overhead data from the classification are compressed by a lossless compression such as an arithmetic coder. Experimental results indicate that the proposed method is superior to the existing methods. In addition, our method can be combined with any existing method.

Keywords: Preprocessing, Classification, Lossless compression, 3D geometry data.

INTRODUCTION
Vertex data is the three-dimensional location data in 3D space. A set of vertices for a 3D object is sometimes called as three-dimensional geometry data, and it is the key data to construct a set of mesh data. Those mesh data constructed from vertex data play a main role in the 3D graphic applications such as medical, engineering, and scientific data visualization [1-4].

Technical development on acquiring three-dimensional data provides a lot of 3D data in various areas. Normally, a 3D scanner produces a huge data including 3D geometry data from an object. For example, millions vertices and many millions meshes can be obtained from a complicated object. This hugeness of 3D data makes big problems in delivering them over networks, storing local disks, and loading them into the main memory of computing systems. Thus a compression technique on the 3D data should be required in many applications.

In general, 3D data consists of two types: one is the vertex data, the 3D geometry data, and the other is the mesh data that are the linking information among the vertex data and mesh property data. Thus, each data compression technique has been discussed in separable fashion. That is mesh connectivity compression [1, 5-7], 3D geometry compression [8-11], and mesh property compression [12-15].

Among these compression methods, the 3D geometry compression is the heart of the 3D compression technique since the 3D geometry data is the basic data for 3D data. For example, the mesh connectivity is the information for three vertices thus mesh only can be prepared when vertices are scanned in advance.

The 3D geometry data is normally represented by a floating-point format. A typical format is the 32-bit IEEE floating point format. And also the data from high-technology science or very accurate engineering are required to be transmitted or stored without any loss. A lossless compression method is preferable in 3D geometry data compression. Some lossless compression methods have been addressed for this reason [16-17]. These methods are considered to be the state-of-the-art compression technique for lossless 3D geometry compression.

In the existing compression methods, a value which is formatted by the IEEE floating-point format is divided by three components for compression. Those components are sign, exponent, and mantissa. The three-components are compressed by context-modeling and optional prediction followed by the arithmetic coding [17].

In this paper, a preprocessing method is proposed to improve the existing method in terms of compression performance. Our method is based on a classification method [18]. Classification in the proposed method requires some overhead information to reduce the number of vertices. Those overhead data are compressed by a lossless compression such as the arithmetic coder. In addition, the proposed method as a preprocessing method can be compatible to any existing method.

RELATED WORK

![Diagram of compression methods](image)

Figure 1: Existing methods (a) FPG method [16] (b) PFPG method [17]

The existing compression methods consist of splitting three components, prediction, and context-based arithmetic coder. The coding unit is each coordinate value whose format is the
IEEE floating-point format and a vertex is composed of three coordinates \((x, y, z)\). Each coordinate value is divided by three components which are the sign, exponent, and mantissa part. The three-components are compressed by context-modeling and optional prediction followed by the context-based arithmetic coding [16-17]. The battery of sensor nodes in the wireless sensor network become rechargeable, the network design changes fundamentally. Designing the network that reduces the cost for power recharging is an important one.

Figure 1(a) shows the existing method in [16] which utilizes context-based arithmetic coding for each component in a separable fashion. Normally, the three components sign, exponent, and mantissa have very different statistics. For example, the sign is formatted with 1 bit, exponent is with 8 bits, and mantissa is with 23 bits or 56 bits. Thus different context-based arithmetic coders are naturally used in the method.

Figure 1(b) shows the exiting method in [17] which is the enhanced version of the method in [16]. This method introduced a prediction method to improve the compression performance. Also it used combined context for sign and exponent.

**Proposed System**

A preprocessing method is proposed to improve the existing methods in terms of compression performance. A classification method is introduced in our method. The classification is assigned for every floating-point value. That is, each coordinate value is compared with the previous coordinate value. If the current and previous values are the same, the current value does not need to save but send 1-bit information to indicate that. This is considered as the overhead information, one bit per coordinate value.

To reduce the number of data itself, classification is introduced, as shown in Fig. 2. A typical floating-point value is represented with 32 bits or 64 bits. After classification, the overhead of 1 bit/data should be added in the resulting data. However, the size of shorten data is enough to compensate the overhead. For example in Fig. 2, the size of input floating-point data is 320 bits. The size of overhead is 10 bits and the reduced size of the shortened data is 192 bits, thus the size of resulting data is 202 bits, which means that saving bits is 118 bits and its reduction ratio is 36.9%.

Table 1 shows the reduction ratio for each coordinate data from a real 3D image set. The table says that some data can be reduced dramatically, although the other is not shortened. Note that the reduction ratio somewhat is different with respect to each coordinate data. For example, the x-coordinate data of Teeth are reduced by a ratio of 79%. This means that only 21% of the data is left by our classification.

Note that there can be no data reduction such as the Bun_zipper and Lusy data in Table 1. In this case the overhead is only left without any data reduction. However, additional bits can be very small if an entropy coding method is applied to the overhead data [19-20]. For example, the overhead should be all zero if the reduction ratio is zero. This means that the entropy of the all zero overhead is also zero. Thus the amount of additional bits required in this case is zero if an efficient entropy coder is utilized.

**Table 1: Reduction ratio for a 3D image set (%)**

<table>
<thead>
<tr>
<th>Image</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bun_zipper</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Teeth</td>
<td>79</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Lucy</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>4</td>
<td>30</td>
<td>31</td>
</tr>
</tbody>
</table>

**Table 2: Illustration of proposed classification and data reducing**

Floating Point Data

| 2.4 | 5.1 | 5.1 | 3.3 | 3.3 | 3.3 | 4.0 | 2.4 | 3.1 | 3.1 |

Overhead

| 0   | 0   | 1   | 0   | 1   | 1   | 0   | 0   | 0   | 1   |

Data

| 2.4 | 5.1 | X   | 3.3 | X   | X   | 4.0 | 2.4 | 3.1 | X   |

Overhead

| 0   | 0   | 1   | 0   | 1   | 1   | 0   | 0   | 0   | 1   |

Shortened data

| 2.4 | 5.1 | 3.3 | 4.0 | 2.4 | 3.1 |

**Figure 2:** Illustration of proposed classification and data reducing

The overall proposed method is described in Fig. 3. Our preprocessing method can be combined with the state-of-the-art existing methods to improve them in terms of compression performance. The overhead data, 1-bit data, is coded by a binary arithmetic coder. This enables the amount of overhead to be minimal.
The complexity of our method can be low. The classification requires one comparison operation and one memory buffer. And a binary arithmetic coder is added to compress 1-bit overhead data. The added complexity is much less than the existing methods that require three context calculators and three arithmetic coders (binary, 8-bit, and 23-bit arithmetic coders) just for a 32-bit floating-point value.

![Figure 4: Test data for 3D geometry compression](image)

**Table 2: Compression results (bit/vertex)**

<table>
<thead>
<tr>
<th>Image</th>
<th>ZIP</th>
<th>FPG [16]</th>
<th>PFPG [17]</th>
<th>Pro-posed +FPG</th>
<th>Pro-posed +PFPG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bun_zipper</td>
<td>85.490</td>
<td>71.323</td>
<td>70.396</td>
<td>71.315</td>
<td>70.395</td>
</tr>
<tr>
<td>Teeth</td>
<td><strong>51.221</strong></td>
<td>54.202</td>
<td>53.355</td>
<td>53.453</td>
<td>53.388</td>
</tr>
<tr>
<td>Lucy</td>
<td>77.181</td>
<td><strong>52.504</strong></td>
<td>52.679</td>
<td><strong>52.504</strong></td>
<td>52.680</td>
</tr>
<tr>
<td>Happy Buddha</td>
<td>47.951</td>
<td>48.417</td>
<td>46.112</td>
<td>46.586</td>
<td><strong>46.064</strong></td>
</tr>
<tr>
<td>Average</td>
<td>65.461</td>
<td>56.611</td>
<td>55.635</td>
<td>55.964</td>
<td><strong>55.632</strong></td>
</tr>
</tbody>
</table>

**CONCLUSION**

A lossless compression method for 3D geometry data has been proposed based on preprocessing. Our method can be easily combined with the existing methods and thus the performance of compression improved in terms of compression ratio.

**ACKNOWLEDGEMENT**

This work was supported by the research fund of Signal Intelligence Research Center supervised by Defense Acquisition Program Administration and Agency for Defense Development of Korea.

**REFERENCES**


