View-Dependent Coding of 3D Mesh Sequences

Semih Çelik
Istanbul Technical University

Uluğ Bayazıt†
Istanbul Technical University

ABSTRACT

Visibility computations are commonly used in computer graphics applications. This paper presents a new view-dependent compression technique for 3D mesh sequences. The approach consists of geometry coding of visible parts and region descriptions of changes in visible regions. The proposed view-dependent compression method yields significant improvements in compression performance over compression method without visibility awareness with gains up to 47%.

Index Terms: Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Animation

1 INTRODUCTION

Mesh sequences are popularly used for visualization of moving synthetic objects in computer simulation, film and game industries. View-dependency is used in many application areas for 3D meshes such as visualization and lightening. The purpose of this paper is adapting view-dependency to compression of 3D mesh sequences. View-dependency has not been applied to 3D mesh sequences in the literature. For static 3D meshes, two types of visibility based compression methods have been applied.

Removal of invisible vertices has been applied to the compression of static meshes in some studies [1, 2, 3]. Additionally, some studies allocate low bits to invisible parts of static meshes [4, 5, 6].

In our study, we assume that the viewpoint is known at the encoder. Another assumption is that the connectivity of the mesh is constant over time.

The proposed method is based on removing invisible vertices in each frame of a 3D mesh sequence. The decoder must have all visible parts in each frame. There are three considerations of bit rate change when visibility detection is incorporated into the proposed system.

1- Bit reduction with removal of invisible vertices.
2- Required bits for region descriptions, which become visible or invisible in current frame.
3- Some vertices become visible in the current frame. These vertices have no temporal reference.

The first two factors directly affect the bit rate, while the third factor increases bit rate as much as the difference in efficiency between temporal and spatial compression methods.

2 PROPOSED METHOD

Our proposed system is composed of visibility detection, region description and geometry coding. Edgebreaker ([7]) is applied to first frame of mesh sequences to encode the constant connectivity.

2.1 Visibility Detection and Region Definitions

Ray-triangle intersection [8] is applied to detect which vertices/faces are visible in the current frame. Each frame might be divided into two regions as visible (R_f^u) or invisible (R_f^i) and four regions as in Figure 1 based on changes of visibility (from f-1 to f):

- R_f^u: visible in both previous and current frame
- R_f^i: visible parts, which are invisible previously
- R_f^i: invisible parts, which are invisible previously
- R_f^u: visible in both previous and current frame

In coding phase of any frame f, the encoder and the decoder has the information of R_f^u and R_f^i. Thus, describing one of R_f^u and R_f^i also describes the other region. Similarly, describing one of R_f^i and R_f^i is sufficient to describe the other region.

Regions, which are visible or invisible in both frames f-1 and f, are greatly larger than regions, which become visible or invisible from frame f-1 to frame f. It is beneficial to represent visibility with descriptions of R_f^u and R_f^i.

2.2 Region Description

A modified version of the simple and efficient Edgebreaker algorithm [7, 9, 10] is developed to represent regions. Edgebreaker traverses a mesh by region growing. It encodes one of five symbols {center (C), left (L), right (R), split (S) and end (E)} for each face included into the region. The symbols indicate the direction of traversal for following faces with respect to a reference edge (gate). Edgebreaker algorithm requires less than 2n (guaranteed higher bound) bits and 1.7n bits on the average to encode connectivity of a mesh with n triangles.

In order to describe any region R, we developed a modified version of Edgebreaker. The modified version employs a reduced alphabet of symbols split (S), right (R), left (L) and end (E). In our coding scheme, symbol R replaces symbol C in Edgebreaker. Starting with a gate g and a triangle X, coding decisions in possible cases are shown in Figure 2.

Figure 1: Region definitions for transition from frame f-1 to f.

In Figure 2(a), symbol R replaces symbol C of the original Edgebreaker. In Figure 2(b), 2(c), 2(d) and 2(e), the modified version uses the same symbols of the original Edgebreaker.

Any face that lies in R_f^{f-1} cannot be included into R_f^{f}. Likewise, faces in R_f^{f-1} are excluded from R_f^{f}.

Traversal algorithms visit all faces of a region once. In the traversal of R_f^{f}, the faces of R_f^{f-1} and the already visited regions of
In this study, we presented a predictive compression of 3D mesh sequences exploiting visibility of meshes to reduce bit rate significantly (up to 47% experimentally). In the future, our system could be modified with prediction of next viewpoints and extending the visible area to cover future visibility of 3D mesh sequences.

REFERENCES