3D Point Cloud Sequence Compression Using Block Skipping

Ji-Su Kim, Jae-Han Lee, and Chang-Su Kim
Korea University
{jisukim, jaehanlee}@mcl.korea.ac.kr, changsukim@korea.ac.kr

***Abstract*—The novel compression algorithm for point cloud sequence is proposed in this paper. We improve the efficiency by adopting a new inter-frame coding algorithm: the block skipping. First, we convert each point cloud sequence to an octree data structure. Then, for compressing the geometry difference between consecutive frames, we build a dual octree data structure. We find that in most areas of each frame have little color difference with the adjacent frame. Therefore, we divide the frame into blocks and perform inter-frame coding by adaptively allocating the number of bits. Finally, we use arithmetic coder to compress geometry and color difference of consecutive frames. Experimental results show that the proposed algorithm compress point cloud sequence more efficiently than intra-frame coding.**

Keywords—3D point cloud sequence; compression; skipping

I. INTRODUCTION

A 3D point cloud is one of the 3D data representations. The point cloud is composed of many points, and each point includes geometry and color information. Much information of the point cloud requires a huge amount of memory and time. Therefore, an efficient compression algorithm is essential. Fig. 1 shows the point cloud samples.

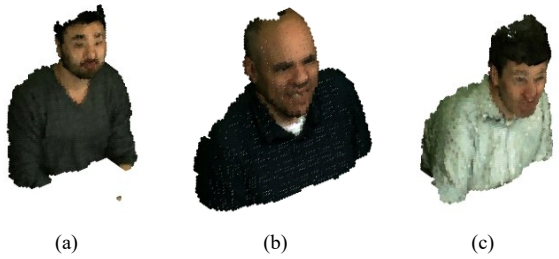
There are various attempts [1, 2, 3, 4, 5] to compress the point cloud. To compress the geometry information, Schnabel and Klein [1], and Huang et al. [2] adopt the octree data structure. Ochotta and Saupe [3] use the 2D wavelet transform based scheme, and Thanou et al. [4] use a graph-based scheme for compression. Kim et al. [5] adopt the dual octree data structure and approximate color difference. Each algorithm has some problems. [1, 2] focus on geometry information only, [3, 4] require a lot of memory and time to compress, and [5] depends on the heuristic factor.

We propose a novel 3D point cloud sequence compression algorithm. First, we decompose point cloud to the octree. Second, we use the dual octree to obtain residual geometry information between consecutive point cloud frames. Third, we use block skipping algorithm to obtain residual color information between consecutive point cloud frames. Finally, we adopt arithmetic coder to compress residual information efficiently. Experimental results demonstrate that the proposed algorithm compress point cloud sequence more efficiently than the intra-frame coding algorithm based on [5].

II. PROPOSED ALGORITHM

A. Octree Data Structure

The octree is an efficient binary description of 3D point cloud [7]. To build the octree, we divide 3D space into 8 blocks of the same size. Each divided block corresponds to a tree node. We allocate a binary value to the tree node. If the divided block contains at least one point, we allocate 1 to the tree node. Otherwise, we allocate 0 to the tree node. In the case of the former, we repeat the dividing process. We divide 3D space until octree depth equal to 7.



(a) (b) (c)

Fig. 1. Point cloud sequence samples: (a) david, (b) ricardo, and (c) andrew

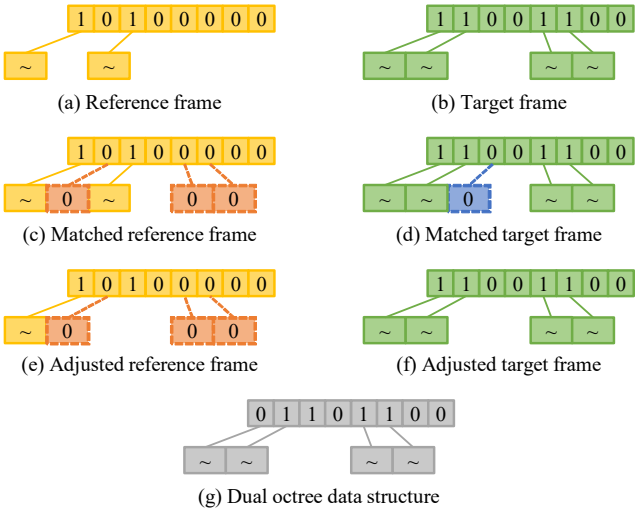


Fig. 2. Sample building process of dual octree data structure

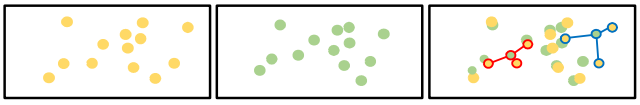
B. Dual Octree Data Structure

For inter-frame coding, we compress the difference of consecutive frames: the reference frame F_t and target frame F_{t+1}. Each frame has a different total point number. Because each frame contains a different number of points, compressing the difference between frame is challenging. To adjust the size of data between adjacent frames equally, we build a dual octree data structure which is adopted in [4, 5]. Fig. 2 shows the building process of the dual octree.

C. Color Estimation

We also adjust the amount of color information between the adjacent frames. From F_t, we estimate $\hat{F}_{t+1}$ that has the same amount of information of target frame F_{t+1}.

We denote a n^{th} point of reference frame as p_n, a m^{th} point of target frame as q_m and color information of each of them as $c_{t,n}$, $c_{t+1,m}$. We search the *K*-nearest neighbor points in the reference frame based on the location corresponding to



(a) Reference frame (b) Target frame (c) K-nearest neighbor

Fig. 3. Sample estimating process of color information

q_m. Here, we set K to 3. Then, the average color of the searched points is defined as estimated color information $\hat{c}_{t+1,m}$. Fig. 3 shows the process of color information adjustment.

D. Block Skipping Algorithm

Usually, adjacent frames rarely change, it means the majority of color difference is 0. Therefore, compressing all color residual information is waste. We propose block skipping algorithm to compress blocks that have large enough change.

We divide 3D space into $512(8 \times 8 \times 8)$ blocks of the same size. Then, we compute a number of bit b_i used for i^{th} block compressing and b_i^{skip} used for i^{th} block skipping as

$$b_i = 1 + BN_i, \tag{1}$$

$$b_i^{\text{skip}} = 1, \tag{2}$$

where N_i is the number of points and B is the quantization bit. Empirically, we set B to 7. Next, we compute the color distortion for two cases: d_i for an ordinary case and d_i^{skip} for a block skipping.

$$d_i = \sum_{n=1}^{N_i} (c_{t+1,n} - \hat{c}_{t+1,n})^2, \tag{3}$$

$$d_i^{\text{skip}} = \sum_{n=1}^{N_i} (c_{t+1,n} - c_{t,n})^2. \tag{4}$$

We compute compressing score s_i and skipping score s_i^{skip} to distinguish which block should be skipping as

$$s_i = \alpha b_i + (1 - \alpha) d_i, \tag{5}$$

$$s_i^{\text{skip}} = \alpha b_i^{\text{skip}} + (1 - \alpha) d_i^{\text{skip}}. \tag{6}$$

where α is a weight to control the relative influence of compression rate and quality. In experiments, we set α as 0.8. For each block, we compare s_i^{skip} and s_i. If $s_i^{\text{skip}} < s_i$, then we skip that block. Otherwise, we compress the color difference by using B bits.

III. Experimental Results

Dataset: For experiments, we adopt the JPEG Pleno Database, captured according to [7], which contains 10 sequences. Each sequence consists of hundreds of frame. Each sampled frame has about 8,000 points whose average data size is 5.39MB.

Quantitative Results: We compare encoded data file size per point under similar decoded image quality. We use bits per vertex(bpv) to compute encoded data file size per point and peak point and peak signal-to-noise ratio(PSNR) to detail decoded point cloud quality. Table 1 shows quantitative comparison results between intra-frame coding algorithm and the proposed algorithm. Note that the proposed algorithm can compress 13.48 times better than the intra-frame coding algorithm with higher PSNR.

(a) (b) (c)

Fig. 4. Qualitative results of the proposed algorithm: (a) reference frame, (b) reconstructed frame, and (c) target frame

Table 1. Quantitative Results

	Intra-frame coding [5]	Proposed
Size of 1 frame	22.5KB	1.66KB
bpv	21.56	1.60
PSNR	20.52	20.93

Qualitative Results: Fig. 4 shows the reference frame, the reconstructed target frame and the original target frame. For original target frame and reconstructed target frame, there is no noticeable image distortion.

IV. Conclusion

We proposed an effective algorithm to compress point cloud sequence. We first model point cloud sequence by using octree data structure. Then, to compute geometry difference information and color difference information, we adopt dual octree data structure and block skipping algorithm respectively. Experimental results show that the proposed algorithm outperforms the intra-frame coding algorithm.

Acknowledgment

This work was supported by 'The Cross-Ministry Giga KOREA Project' grant funded by the Korea government(MSIT) (No.GK18P0200, Development of 4D reconstruction and dynamic deformable action model based hyper-realistic service technology)

References

[1] R. Schnabel and R. Klein. Octree-based point-cloud compression. In *SPBG*, 2006.

[2] Y. Huang, J. Peng, C. C. J. Kuo, and M. Gopi. A generic scheme for progressive point cloud coding. *VCG*, 14(2):440–453, 2008.

[3] T. Ochotta and D. Saupe. Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In *PBG*, 2004.

[4] D. Thanou, P. A. Chou, and P. Frossard. Graph-based motion estimation and compensation for dynamic 3D point cloud compression. In *ICIP*, 2015.

[5] J.-S. Kim, S. Lee, J.-H. Lee, and C.-S. Kim. Inter-frame compression of 3D point cloud sequences. In *ICEIC*, 2018.

[6] D. Meagher. Geometric modeling using octree encoding. *CGIP*, 19(2):129–147, 1982.

[7] C. Loop, C. Zhang, and Z. Zhang. Real-time high-resolution sparse voxelization with application to image-based modeling. In HPGC, 2013.