FDM-based Global Motion Estimation for Dynamic 3D Point Cloud Compression

So Myung LEE[1], Li Cui[1], Tianyu Dong[1], Eun-Yong Chang[2], Jihun Cha[2] and Euee S. Jang[1]

[1]Department of Computer and Software, Hanyang University ,Seoul 04763, Korea
[2]Electronics and Telecommunications Research Institute (ETRI), Daejeon 34129, Korea
Keywords: dynamic point cloud compression, global motion estimation, fast distortion measurement

ABSTRACT

In this paper, we propose a fast global motion estimation (GME) for dynamic 3D point cloud compression (PCC). We applied fast distortion measurement method(FDM) to replace and reduce the computational complexity of GME. The experimental results show that the proposed method is two times faster than MPEG V-PCC.

1 INTRODUCTION

Augmented reality/virtual reality(AR/VR) applications are becoming more popular in recent years, which includes dynamic 3D point clouds. These applications often face difficulty in storing or transmitting point cloud data due to huge size requirement. Therefore, MPEG, which is developing a set of standards for video compression, uses video codec like Video-based Point Cloud Compression (V-PCC) to compress dynamic 3D point cloud data [1]. Additionally, there is Geometry-based Point Cloud Compression(G-PCC) that deals with static and dynamic acquisition 3D point cloud in MPEG. If motion estimation/compensation is adopted in G-PCC, it may be possible to efficiently compress dynamic 3D point clouds.

There are some global motion estimation (GME) studies for dynamic 3d point clouds, one of which is the study for graph-based motion estimation/compensation (ME/MC) [2]. The graph-based ME/MC study, represented the time-fluctuation of the sequences using graphs, and the positions and color data of point cloud were considered to be the peak signal on the vertices of the graph. ME was depicted as a feature matching between successive graphs and it was used for color compensation in 3D point cloud compression. Our proposed method deals with GME algorithms simply in a different way such as sorting each pixel in ascending order and counting the number of identical pixels, which results in low computational complexity.

We tried GME in another way to speed up the computational complexity of the solution. The computational complexity of the original GME method is too high, because it needs to compare each pixel positions(X,Y,Z) between frames one by one. But the proposed method in this study is to sort pixels of the previous frame and the next frame in ascending order and count the number of identical pixels between two frames. Calculating the number of identical pixels as this proposal, it is much simpler and faster than that of the original method.

In this paper, we first used the fast distortion measurement (FDM) [3] method to reduce computational complexity. The FDM method is used to removed the duplicate points of two frames and the performance of the method is compared with that of the existing method. Experimental results in [3] has shown that the total time computation of distortion measurement has reduced between 47% and 74% . Because the time has been reduced successfully, we have applied GME to FDM.

The rest of this paper is organized as follows. In section 2, we introduce the overall background for GME. In section 3, we describe an explanation of the FDM on which this work was based on and the proposed method, GME. Results are presented in section 4 and conclusion is given in section 5 briefly.

2 BACKGROUND

Generally, GME is an important task in video compression. GME is useful for video content analysis such as video processing, video object segmentation and background modeling [4]. However, GME has high computational complexity. So, the need for more research in GME is on the rise to reduce computational complexity. Although GME is costly in computational complexity, it still poses a good position to provide further compression efficiency. Thus, we applied GME to point cloud, which is expected more complex than that in the video.

GME in point cloud compression also has significant computational complexity. We can depict a simple GME encoding process. We start from the basic fact on points in a dynamic point cloud: *there is no relation between points temporally*. This facts implies that a GME process in point cloud compression not only needs to measure the distance between two adjacent point clouds, but also needs to find out one-to-one mapping between (nearest) points of two adjacent frames. This is where the huge complexity comes from. The GME in video coding computes N comparisons if there are N pixels in a video frame. If there are M different positions (or movement of a frame over the other), there could be MN computations altogether. However, the GME in point cloud compression adds additional computation: one-to-one

mapping between points (i.e., N^2). The computation order of GME in point cloud compression becomes $MN + N^2$. Therefore, the computational complexity of GME process in PCC is dominantly determined how fast one-to-one mapping can be processed.

The way to show the best GME results in MPEG PCC is to use the most optimal method, PC_ERROR. PC_ERROR is a software to measure the geometry distortion of the point cloud used in MPEG [5]. The method for measuring distortion is to determine how close the point positions of the original point cloud data and the compressed one are in Euclidean distances [3]. In PC_ERROR, it identifies the corresponding point from the other point cloud for every point in one point cloud to measure the distances. In MPEG, PC_ERROR is used in two different ways to measure distortion as shown in Fig.1 [5].

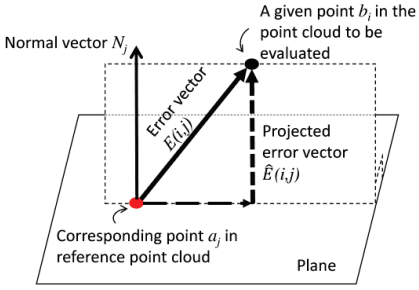

Fig. 1 Point-to-point distances vs. point-to-plane distance [5]

In point-to-point distance measurement, it uses equation (1) to measure the average or maximum in Euclidean distances between a pair of points and point-to-plane distance calculates as shown in equation (2) to project the error vector $E(i,j)$ along the normal direction N_j to obtain a new error vector $\hat{E}(i,j)$ [5].

$$e_{A,B}^{c2c} = \frac{1}{N_A} \sum_{\forall a_j \in A} \|E(i,j)\|_2^2 \qquad (1)$$

$$e_{A,B}^{c2p} = \frac{1}{N_A} \sum_{\forall a_j \in A} \left(E(i,j) \cdot N_j\right)^2 \qquad (2)$$

The peak signal-to-noise rate(PSNR) can be calculated from these obtained values using equation (3).

$$PSNR_{A,B} = 10 \log_{10} \frac{p^2}{e_{A,B}} \qquad (3)$$

Dynamic 3d point cloud also uses PC_ERROR with this distortion measurement method. Because it is recommended to use the same matrix applied to PC_ERROR. So we measured the results of this study with the same algorithm used by PC_ERROR. The GME method using PC_ERROR has its pros and cons. It guarantees the optimal result given the search range. Yet, the computational complexity remains very high (i.e. *MN* + N^2).

3 EXPERIMENTS

3.1 Fast Distortion Method

In this section, we introduce the current method of acquiring distortion using PC_ERROR with two point cloud data and the FDM method of obtaining distortion through PC_ERROR with two changed point cloud data after point deletion process. FDM is used to figure out the distortion of the closest neighbors of the points between two frames to reduce complexity [3] . First, the pixel points (X,Y,Z axis) of each frames was sorted in ascending order [3]. A method for sorting is as follows. First of all, let the same x-values sort in a bundle. Then, the remaining y-values and z-values from the bundle of x-values are sorted in this way [3]. In this alignment, the values of x, y, and z must be sorted in ascending order. Second, we designed a fast searching algorithm with arranged points in ascending order shown as Fig. 2 [3].

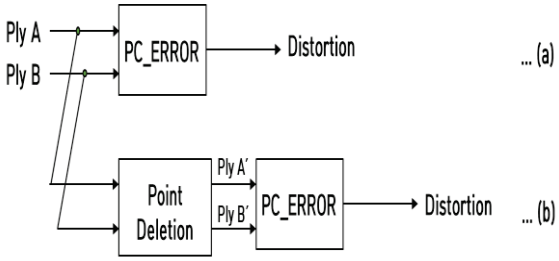

Fig. 2 Comparison between PC_ERROR and proposed method [3]

Fig. 2-(a) measures distortion through PC_ERROR with given original ply A and B. The proposed method is shown as Fig. 2-(b). Fig. 2-(b) performs a duplicate point deletion before doing PC_ERROR. Point deletion process is shown as Fig. 3.

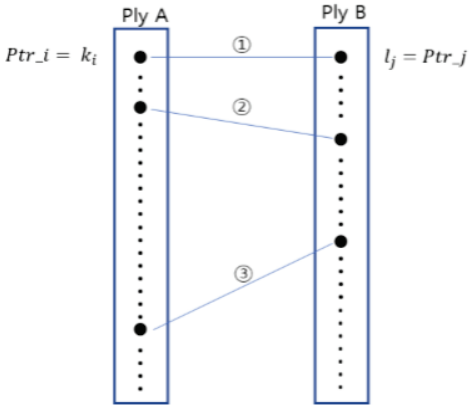

Fig. 3 Process of Point Deletion in proposed method [3]

In Fig. 3, ply A and B with sorted in ascending order perform the point deletion process. First, k_i is the pointer Ptr_i of ply A and l_j is the pointer Ptr_j of ply B [3]. As in the case of ①, if x, y and z coordinates of k_i and l_j match each other, execute the point deletion process [3].

After the process, each k_i and l_j pointer increase by one for finding the next matching coordinates. In case of ②, if k_i pointer is larger than l_j pointer, l_j pointer will be increased until the coordinates of two pointers are equal. And the case ③ is the opposite of case ② [3].

After point deletion process, ply A and B become ply A' and B' with duplicate point deleted. Ply A' and B' also measures distortion through PC_ERROR [3]. FDM method reduces the computational complexity from N^2 to $N \log N$.

3.2 Global Motion Estimation

We adopted the merit of FDM for GME design. In GME, as shown in Fig. 4, **Ply 1** extended the range of pixel points(x,y,z) by using a global search for each pixel point; then, generates a total of n different combinations. On the contrary, **Ply 2** is an original ply file without a global search.

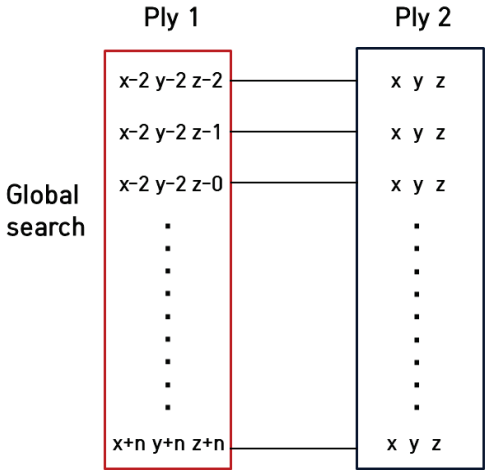

Fig. 4 Ply 1 with global search and original ply 2

The n combinations of these generated **Ply 1** and **Ply 2** are compared through PC_ERROR as shown in Fig. 5-(1), the reference method, to extract the most highest PSNR-value out of n-combinations. The proposed method is shown in Fig. 5-(2) and it performs a global search as shown in Fig. 5-(1) and then sort the pixel points of **Ply 1** and **Ply 2** in ascending order, respectively. It then compares each pixel point of **Ply 1'** and **Ply 2** to count the number of identical points. The process of sorting is like the point deletion process in Fig. 3. After the sorting **Ply 1'** and **Ply 2** in the same way as the Point Deletion process in Fig. 3, it counts the number of identical points between **Ply 1''** and P**ly 2'** instead of deleting duplicate points.

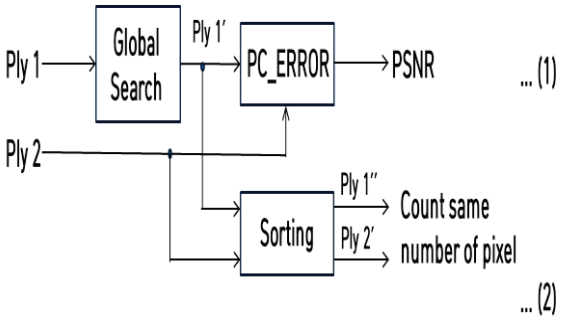

Fig. 5 An algorithm for reference method and proposed method with global search

After this process, it compares the largest number of identical pixels as a result of Fig. 5-(2) and the highest combination of PSNR as a result of Fig. 5-(1) to identify how the accuracy of the two results is consistent. We also measured the time and PSNR about the above results.

4 EXPERIMENTAL RESULTS

For the evaluation of the proposed method, we used one of the datasets provided by MPEG [6]. The dataset is soldier ply files as shown in Fig. 6 with integerized values of which each ply is made up of 300 frames.

Fig. 6 A sequence of soldier ply

We compared them with the previous and current frame of the same ply and measured the average time and PSNR through PC_ERROR. We designated a global search range of -2 to +2 in each axis for each previous frame and generated a total of 125 (= 5^3) combinations. Therefore, the total of 125 combinations were executed per frame through PC_ERROR.

Table 1 shows how the corresponding combinations in the original method and the proposed method match each other. The sequence of soldier is 78.6% with high accuracy. Table 2 shows the average time and PSNR of soldier sequence. The average time shows that the

proposed method is about twice as fast as the original bur for PSNR, the proposed method is lower than the reference.

Table 1. Accuracy for reference and proposed method results

Sequence	Accuracy for reference and proposed method results
soldier	78.6%

Table 2. Average time and PSNR for reference method and proposed method

Reference method		Proposed method	
Time	PSNR	Time	PSNR
383.2497	67.0002	241.5067	66.9525

5 CONCLUSION

In this paper, we propose a fast global motion estimation (GME) for dynamic 3D point cloud compression (PCC). The key point is to sort pixels of the previous and next frame in ascending order then count the number of same pixels between two frames. Thus, the time computation of the proposed method has decreased as shown in results. This method could give a new approach on GME algorithm to compress dynamic 3D point cloud data. In the future, other methods such as MEMC algorithm with local search can be performed to better compress dynamic 3d point cloud data.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No. 2017-0-00067, Development of ICT Core Technologies for Safe Unmanned Vehicles)

REFERENCES

[1] ISO/IEC JTC1/SC29/WG11 MPEG2017/w16631. Use cases for Point Cloud Compression. (Geneva, January 2017).

[2] Dorina Thanou, Philip A. Chou and Pascal Frossard, “GRAPH-BASED MOTION ESTIMATION AND COMPENSATION FOR DYNAMIC 3D POINT CLOUD COMPRESSION," Proc. ICIP'15, pp.3235-3239 (2015).

[3] Yousun Park and Euee S. Jang, “Fast distortion measurement method for MPEG point cloud compression,”Proc. The Institute of Electronics and Information Engineers'18, pp. 825-828 (2018).

[4] Yue-Meng Chen and Ivan V.Bajic, “Motion Vector Outlier Rejection Cascade for Global Motion Estimation," J. IEEE'09, Vol. 17, No. 2, pp. 197-200 (2010).

[5] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen and Anthony Vetro, “GEOMETRIC DISTORTION METRICS FOR POINT CLOUD COMPRESSION”, Proc. ICIP'17, pp. 3460-3464 (2017).

[6] ISO/IEC JTC1/SC29/WG11 MPEG2017/w18474. Common test conditions for point cloud compression. (Geneva, March 2019).