Enhancement Layer Inter Frame Coding for 3D Dynamic Point Clouds

Shishir Subramanyam
Distributed and Interactive Systems
Centrum Wiskunde & Informatica
Amsterdam, Netherlands
s.subramanyam@cwi.nl

Pablo Cesar
Distributed and Interactive Systems
Centrum Wiskunde & Informatica
Amsterdam, Netherlands
p.s.cesar@cwi.nl

***Abstract*—In recent years, Virtual Reality (VR) and Augmented Reality (AR) applications have seen a drastic increase in commercial popularity. Different representations have been used to create 3D reconstructions for AR and VR. Point clouds are one such representation that are characterized by their simplicity and versatility, making them suitable for real time applications. However, point clouds are unorganized and identifying redundancies to use for compressing them is challenging. For the compression of time varying or dynamic sequences it is critical to identify temporal redundancies that can be used to describe predictors and further compress streams of point clouds. We use a point cloud codec that encodes additional information in an enhancement layer. We propose adding inter prediction to the enhancement layer in order to gain further bit rate savings.**

***Index Terms*—Point cloud compression, octrees, enhancement layer, interframe coding**

I. Introduction

In the last decade VR and AR have enjoyed increased mainstream commercial popularity. This has been possible due to the increased availability of affordable sensors, increased computational power of commodity hardware and the miniaturization of electronics; making AR and VR more viable and usable. In order to deliver immersive VR and AR experiences it is necessary to create photorealistic reconstructions of a natural scene or object. Recent advances in depth sensors have made it possible to perform such reconstructions in real time.

3D point clouds have become a popular format to represent such reconstructions. They are not dependent on the capture methodology. They contain a set of coordinates describing the location of points that together represent the geometry as well as a set of attributes associated with each point such as color, normals and transparency.

Point clouds can be used in a host of new VR and AR applications that allow the user to experience and interact with the reconstruction in new ways. Some examples of such applications are real time 3D immersive telepresence [1], 3D free viewpoint sports replays and cultural heritage applications among others.

Point clouds are characterised by their simplicity and versatility. They do not require any triangulation computation like meshes, they do not require heavy pre-processing and are resilient to noise. They are thus suitable for real time applications. Point clouds are versatile as they have no restrictions on the additional attributes that can be stored at each point location.

However, point clouds are also challenging to compress. As they do not have connectivity and topology (or surface information), it is difficult to identify spatial redundancies and occlusions for compression within one frame (intra frame). There are also no explicit point correspondences between frames or even fixed bounding boxes which makes it challenging to identify temporal redundancies for compression across a sequence of frames (inter frame).

In recent years, standardisation activities for point cloud compression have been ongoing [2] [3]. The groups that conduct the call for proposals are comprised of major academic and industry partners who have contributed datasets for the testing and evaluation of submitted proposals. These datasets are more dense and photorealistic than those used in previous research [1] [4]. They contain 800,000 to 1 million points (see figure 1).

A popular datastructure used to code point clouds is the octree. An octree is used to partition the 3D space by recursively subdividing it into 8 octants as shown in figure 2. For point clouds this recursive subdivision is done for all regions that are occupied. The primary reason octrees have been used for point cloud compression is because they enable a more efficient representation of the point cloud geometry by regularizing the structure and using occupancy codes for partitioned spaces.

The compression process is repeated for each of the leaf nodes in the octree. The number of leaf nodes in an octree are exponentially related to the tree depth. The computational complexity is therefore exponential with respect to the octree depth or level of detail, as the number of leaf nodes that need to be coded increase by a factor of 8 each time subdivision occurs. This problem has a bigger impact on the photorealistic datasets used by ongoing standardisation activities [2] [3], as the density of point clouds is significantly high, requiring a higher level of detail (LoD) to maintain quality. This results in a deeper octree with exponentially more leaf nodes. One approach to address this problem was proposed by Ainala et al [7] where the authors suggest using the octree up to a certain fixed level of detail and to encode additional detail in an enhancement layer. We propose extending this approach by adding inter frame prediction to the enhancement layer

Fig. 1. Dense photorealistic point cloud contributed by 8i Technologies [5] to ongoing standardisation activities [2] [3]

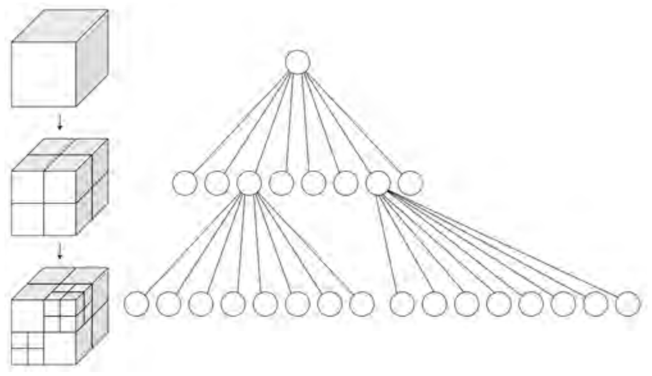

Fig. 2. 3D space partitioning using octrees and accompanying occupancy codes [6]

to exploit temporal redundancies across frames of a dynamic sequence.

The rest of this paper is organised as follows, in section II we outline the related work in point cloud compression. In Section III, we describe the compression framework where we implemented our solution. In Section IV, we describe the enhancement layer inter prediction approach. In section V, we discuss our results followed by conclusions in section VI.

II. Related Work

Research into point cloud compression in the past can be broadly placed in two main categories. The first category is signal processing based approaches. Zhang et al. [8] proposed a methodology to compress point cloud attributes using a graph fourier transform. They assume that an octree has been created and separately coded for geometry prior to coding attributes. They then construct the graph, similar to [9], and calculate a precision matrix for each macro-block and perform an eigen decomposition. They use this to quantize each colour component and entropy code the results. Ricardo Queroz et al. [10] used a region adaptive hierarchical transform in order to use the colors of nodes in lower levels of the octree to predict the colors of nodes in the next level. Each voxel is assigned a weight based on the number of voxels combined to generate them and a transform is used based on these weights. The results are quantized and entropy coded using arithmetic coding. Thanou et al. [9] found matches between points in consecutive frames and warped the previous frame to the current frame. Point cloud attributes like color have an irregular domain of definition and as a result are suitable for applying a graph transform, however this approach requires repeated eigen decomposition of the graph laplacian. As a result this approach is not suitable for dynamic sequences of point clouds that are used in real time applications.

The second category of codecs are based on concepts from 2D image and video codecs. These approaches are block based and hierarchical. A popular data structure used to spatially partition 3D point clouds is the octree. This is a 3D extension of the quadtree used in video coding.

Intra Frame coding in octrees can be achieved by entropy coding the occupancy codes as shown in [1] to compress the geometry. JPEG compression is used on the colors after they have been mapped to a 2D grid. Robert Cohen et al. [11] created a methodology that applies 3D intra prediction between macroblocks similar to video coders like AVC and HEVC. They used a nearest neighbour interpolation/extrapolation to overcome the problem of sparse macroblocks in point cloud octrees and to identify the value of an attribute at the boundary between macroblocks. These values are used to predict attributes in each macroblock and the residuals are coded using a shape adaptive DCT.

Inter frame coding with octrees was demonstrated by Julius Kammerl et al [4] where an XOR operation was performed between occupancy codes of consecutive octree frames and the results were coded using an entropy coder. Mekuria et al [1] used a block based approach to calculate a rigid transform between compatible macro blocks in consecutive frames using the iterative closest points algorithm. This approach involves encoding at multiple levels of detail, thus allowing progressive decoding and also predicts both geometry and attribute (color) values. Ainala et al [7] presented an improvement to this technique for individual point cloud frames using an enhancement layer. After a fixed octree depth they switch to a new layer that uses plane projection approximantion.

A limitation of octree based point cloud compression is that as the depth d of the octree increases the computational cost increases exponentially (upto 8^d). This problem has become more acute with new dense photorealistic datasets. In order to overcome this issue our approach is to terminate the octree at a fixed level of detail that is computationally viable. Additional information is stored in an enhancement layer (based on the approach of Ainala et al [7]) and will only be used if network conditions and computational resources allow it. The enhancement layer is based on a plane projection approximation where the geometry coordinates are transformed, sorted and differentially coded while the color attributes are placed in a 2D image grid using the same sort order for raster scanning. The resulting images are stacked and compressed using JPEG compression. In this way the bond between geometry and attributes are maintained. The transformation is determined using Principal Component Analysis.

III. Compression Framework

The work presented in this paper was developed using the compression of framework created by Mekuria et al [1] available at https://github.com/cwi-dis/cwi-pcl-codec. The enhancement layer approach proposed by Ainala eta al [7] was implemented here and extended with inter prediction.

A. Compression framework

We use the framework of Mekuria et al [1] (PCC codec) in our implementation. This framework is built using the Point Cloud Library (PCL) [12]. The PCC codec improves the PCL codec primarily by adding a color coder and an inter prediction scheme that also includes predicting attributes (color). The intra frame coder encodes geometry at fixed levels of detail by differentially encoding voxel local points from the PCL codec. This process is repeated at different pre-specified levels of detail (LoD) so that the decoding process can be progressive and a final level of detail can be chosen by the decoder based on prevailing network conditions. A carry less byte range coder is then used for statistical compression (entropy coding) of the final bitstream. The range coder has been preferred over arithmetic coding as it operates at a byte level making it much faster on general microprocessors. The intra frame coder also encodes colors by performing a depth first traversal of the octree and scanning the colors to a structured JPEG image grid using a zig zag pattern to further exploit the correlations among co-located points.

The inter frame coder in their codec reuses the intra frame coder in combination with a lossy prediction scheme based on the iterative closest points (ICP) algorithm from PCL. This is shown in figure 3. Macroblocks are defined at a predefined number of levels above the final level of detail or voxel size. The inter frame coder identifies shared macroblocks in the previous frame and the current frame that are comparable and eligible for prediction. This eligibility is based on the macroblock point count and color contrast changes. The ICP algorithm then identifies a rigid transform between shared macroblocks in both frames and if the algorithm converges the transform is used as a predictor. The framework can optionally be used to calculate a color offset between corresponding macroblocks across frames.

B. Enhancement layer

Ainala et al proposed adding an enhancement layer to the PCC codec [7]. They implemented a proof of concept and found that the results were promising. We first implemented

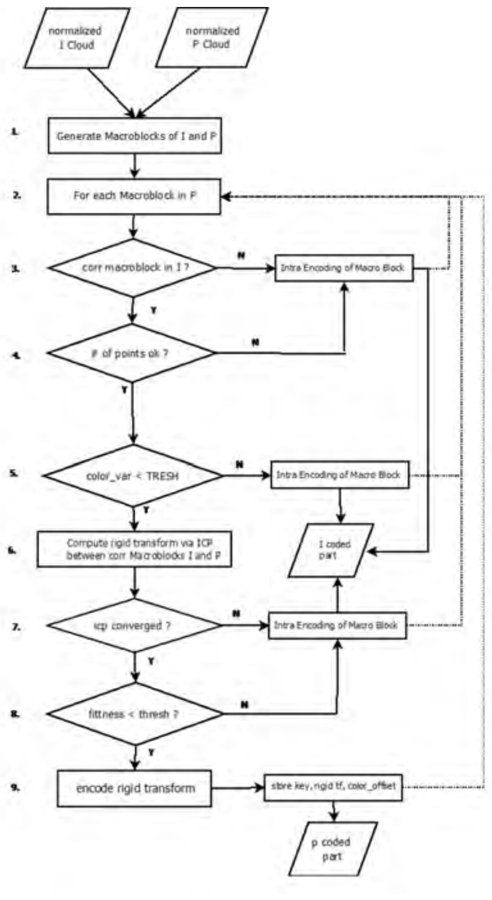

Fig. 3. Inter frame prediction by Mekuria et al [1]

and integrated their proposal with the PCC codec in order to evaluate its actual suitability and effectiveness in compressing the dense point clouds used in ongoing standardization activities [2].

This was done by specifying the depth at which the octree based PCC codec is terminated and the enhancement layer takes over. The goal is to use the octree based codec as long as it is computationally viable and encode additional details in an enhancement layer. During the subdivision process, once the octree target level of detail is reached, we move to the enhancement layer. Internally the enhancement layer is treated as an additional level of detail with the octree subdivision function overwritten to terminate when the flatness criteria is met. This is done by first calculating the covariance matrix of points within a tree node. The centroid of points is used as the mean.

The flatness criteria, shown in equation 1, is then calculated as the ratio of the eigenvalues of the covariance matrix within a node ($\lambda 1$, $\lambda 2$ and $\lambda 3$). A comparison between the smallest eigenvalue (direction of minimum variation) and the sum of all eigenvalues is used as an indicator of flatness. If the node is flat enough we then apply principal component analysis (PCA) on all voxels within the node. This is done by using the eigenvectors corresponding to the eigenvalues calculated in the previous step to create a 3X3 rotation matrix. We then rotate the point coordinates to remove linear correlations. This orthogonal transformation converts the x,y,z coordinates to u,v,w in decreasing order of variance. The point coordinates are then sorted based on equation 2, differentially coded and quantized to integer values. The resulting bit stream is further compressed using lossless statistical compression by using the same range coder that is used by the PCC codec.

$$\theta = \frac{min(\lambda 1, \lambda 2, \lambda 3)}{(\lambda 1 + \lambda 2 + \lambda 3)} \tag{1}$$

$$index = sort(\alpha * u + v) \tag{2}$$

Color is then coded for each flat node by using the index from equation 2 to map points onto a 2D grid of fixed width. The last row is padded with zeros to complete the image. All flat nodes are then scanned in depth first order, the images corresponding to the nodes are stacked together and compressed using JPEG compression. The bond between geometry and color is maintained as we use the same sort order to encode both (see equation 2).

IV. Enhancement layer inter prediction

Inter-prediction in the enhancement layer is performed by using the double buffered octree structure from PCL [4]. The octree subdivision criteria is changed from voxel occupancy to flatness in the enhancement layer. Octree nodes are terminated when they are sufficiently flat. The rotation performed during plane projection for each node is stored as a quaternion and added to the bitstream. The images generated while compressing the color attribute are associated to their corresponding nodes in the double buffer. If a node in the enhancement layer of the current frame has the same flatness profile as an enhancement layer node in the previous frame we mark the node for inter prediction. We compare the flatness θ as well as the eigen values of the covariance matrix.

The node is identified by the k(x,y,z) macroblock key associated with it. Each component of the key is stored using 16 bit integer values. For enhancement layer nodes that are marked for prediction, the key of the corresponding node from the previous frame is added to the bitstream. While decoding the 2D image of colors, transformed geometry coordinates (after plane projection) and the rotation matrix associated with the macroblock (represented as a quantised quaternion) from the previous frame are reused for the current frame.

The decoder then skips processing these nodes and places them in the depth first scan order. They are then recombined with the intra coded blocks and the decoded point cloud is constructed. The algorithm we implemented is shown in figure 4. Some of the modules are explained in the following subsections.

A. Frame header

At the start of the bit stream associated with every frame we add the frame header. This is used to apply pre-specified codec settings. The settings are read from a configuration file that is required as one of the inputs to the codec. The frame header contains all the fields required by the PCC codec available at https://github.com/cwi-dis/cwi-pcl-codec such as the bit allocation for macroblock size, prediction method, color bit allocation and mapping mode. We extend the header to include settings for the enhancement layer. This includes the level of detail where the enhancement layer begins, the flatness threshold θ used during subdivision and lastly the mapping mode used during raster scan of color attributes to 2D images in the enhancement layer.To generate the results presented in the next section we used the JPEG lines mapping scheme implemented by Mekuria et al [1]. We also include the threshold values used while comparing the flatness of corresponding enhancement layer nodes in successive frames for inter prediction.

B. Principal Component Analysis (PCA)

The plane projection done in the enhancement layer is done using PCA. Principal component analysis uses a set of orthogonal transformations to convert a set of linearly correlated input to linearly uncorrelated variables. We implement PCA by using the Eigen library from PCL [12]. The covariance matrix is calculated and eigen decomposition is performed for all points within an enhancement layer node. Each node can be processed independently and therefore can be run in parallel. We sort the eigenvalues of a flat enhancement layer node in ascending order, we then use the corresponding eigenvectors to form the rotation quaternion needed for plane projection. The geometry of points after rotation are stored by differentially coding the coordinates and applying a quantization scheme. The amount of information retained per coordinate, after quantization is proportional to the eigenvalues corresponding to them. The same sort order is maintained for geometry and attributes.

C. Quaternion coding

Quaternions are a number system that extend complex numbers. They can be represented in the form a + bi + cj + dk. Where a,b,c and d are real numbers and i, j, k are quaternion units. The properties of quaternions are shown in equation 3. A crucial property of quaternions is that quaternion multiplication is noncommutative. The relationship between i, j and k are thus similar to the cross product rules for unit cartesian vectors. As a result quaternions can be used to efficiently express a rotation in three dimensional space. In our implementation we store the orthogonal transformation used for plane projection in the form of a quaternion. This is then quantized and stored as three numbers occupying 16 bits each. Each quaternion is associated with the macroblock key of the corresponding enhancement layer node. During the decoding process we use the quaternion to rotate the coordinates back to their original values.

$$i^2 = j^2 = k^2 = ijk = -1 \tag{3}$$

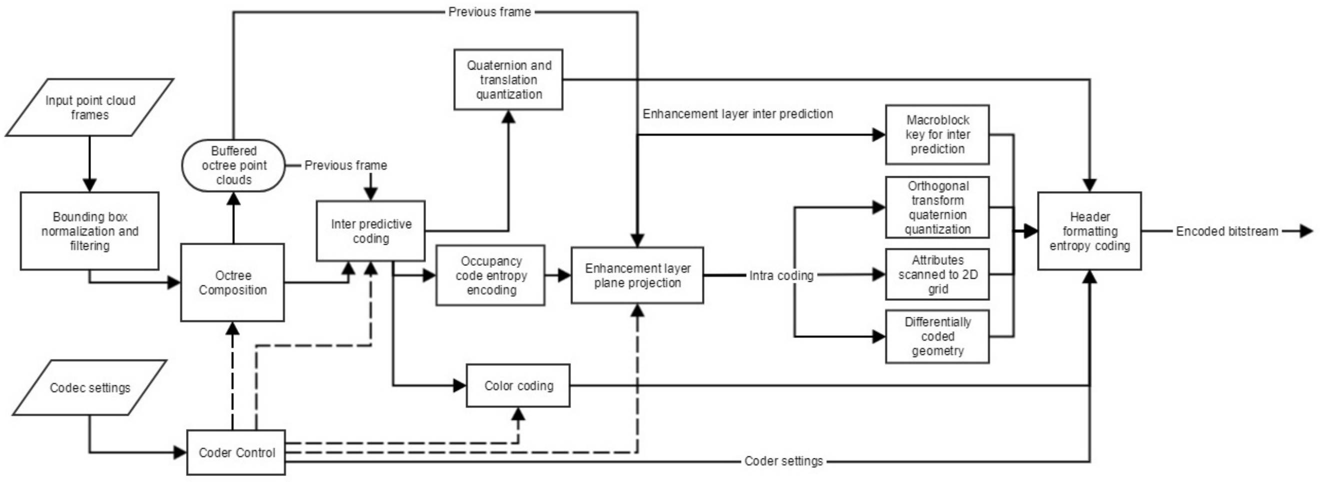
Previous frame
Input point cloud frames
Quaternion and translation quantization
Enhancement layer inter prediction
Macroblock key for inter prediction
Buffered octree point clouds
Previous frame
Bounding box normalization and filtering
Inter predictive coding
Orthogonal transform quaternion quantization
Header formatting entropy coding
Encoded bitstream
Octree Composition
Occupancy code entropy encoding
Enhancement layer plane projection
Intra coding
Attributes scanned to 2D grid
Differentially coded geometry
Codec settings
Color coding
Coder Control
Coder settings

Fig. 4. Enhancement layer inter prediction added to the point cloud codec created by Mekuria et al. [1]

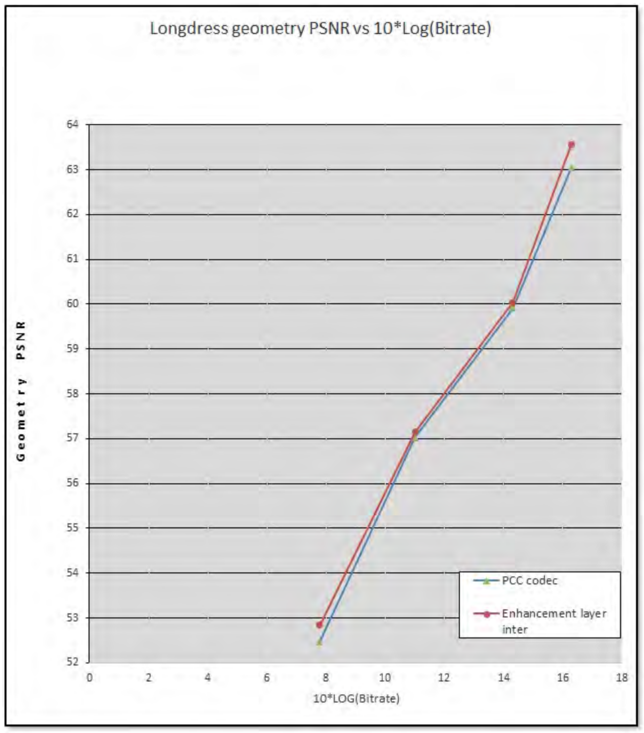

Fig. 5. Geometry distortion vs bit rate for the long dress dataset

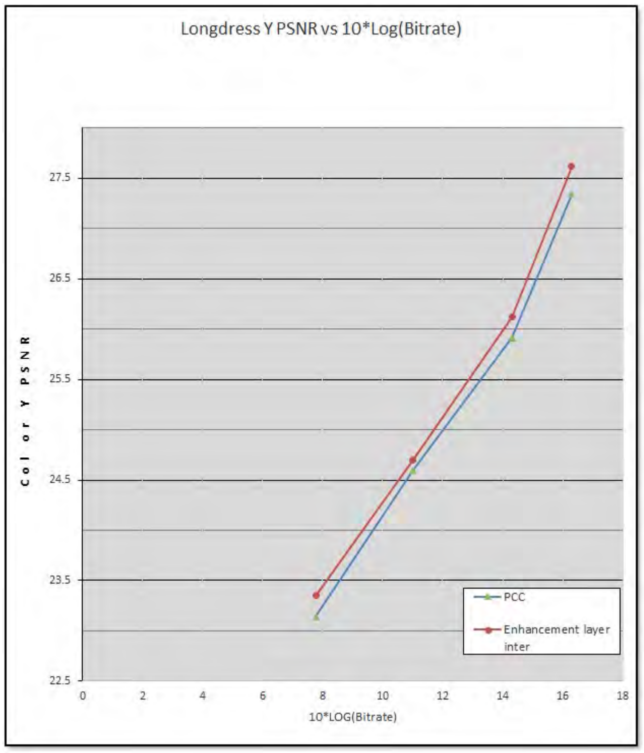

Fig. 6. Y color channel distortion vs bit rate for the long dress dataset

V. Results

In order to evaluate the performance of the codec we use the longdress dataset [5] that has been used for standardisation activities [2] [3]. This is a relatively dense point cloud dataset with approximately 800,000 points per frame. The rate distortion performance of the codec was measured against the codec developed by Mekuria et al [1] on the same dataset. We set the threshold for inter frame enhancement layer coding to 5 % change in flatness (equation 1) and compare the differences in eigen values to identify the corresponding macroblocks. In order to measure the distortion introduced by the compression process we identify corresponding points in the original and decoded point clouds. We use the implementation available in https://github.com/cwi-dis/cwi-pcl-codec and measure the geometry distortion using the point to point distance to calculate a peak signal to noise ratio (PSNR). We also measure colour distortion separately by calculating a PSNR for the

Y,U and V colour channels. The octree and enhancement layer configuration is modified so that we generate a decoded point cloud at the same bit rate points for the implementation of Mekuria et al [1] (PCC) and our extension of this codec (enhancement layer inter) with a 10% tolerance. The results for geometry distortion are shown in figure 5 and for distortions in the Y color channel are shown in figure 6.

Based on the rate distortion curve, we calculate a Bjontegaard delta PSNR for the longdress dataset. We observe an improvement in our implementation in objective quality of geometry of approximately 3% at similar bitrates. Using the same procedure for each color channel we observe an improvement of at least 5%.

VI. Conclusion

The implementation of inter frame coding in the enhancement layer shows an improvement over the PCC codec. There is a significant improvement to distortion in the color channels because of the quantization scheme used in JPEG compression. We are able to retain more color information while operating at the same bit rates as compared to the anchor. Our results also indicate that many local patches in the point cloud exhibit flat characteristics, this can be a consequence of the nature of the capturing process used. The initial results presented in this paper look promising and will be investigated more thoroughly in future work. This work can be further improved by using a more efficient entropy coding technique such as the PAQ neural network used by Ainala et al [7]. The distortions introduced by the coding process can be reduced by additionally coding the residuals.

References

[1] Rufael Mekuria, Kees Blom, and Pablo Cesar, “Design, implementation and evaluation of a point cloud codec for tele-immersive video,” *IEEE Transactions on Circuits and Systems for Video Technology*, January 2016.

[2] “Call for proposals for point cloud compression iso/iec jtc1/sc29 wg11 n16732, geneva ch,” January 2017.

[3] Touradj Ebrahimi, Siegfried Foessel, Fernando Pereira, and Peter Schelkens, “Jpeg pleno: Toward an efficient representation of visual reality,” *IEEE MultiMedia*, October 2016.

[4] J Kammerl, N Blodow, R B Rusu, S Gedikli, E Steinbach, and M Beetz, “Real-time compression of point cloud streams,” *Robotics and Automation (ICRA), 2012 IEEE International Conference*, pp. 778–785, May 2012.

[5] Eugene d’Eon, Bob Harrison, Taos Myers, and Philip A. Chou, “8i voxelized full bodies - a voxelized point cloud dataset, iso/iec jtc1/sc29 joint wg11/wg1 (mpeg/jpeg) input document wg11m40059/wg1m74006, geneva,” January 2017.

[6] “Octree data structure,” https://commons.wikimedia.org/wiki/File:Octree2.svg, Accessed: 2016-10-28.

[7] Khartik Ainala, Rufael N. Mekuria, Birendra Khathariya, Zhu Li, Ye-Kui Wang, and Rajan Joshi, “An improved enhancement layer for octree based point cloud compression with plane projection approximation,” *Proc. SPIE 9971, Applications of Digital Image Processing XXXIX*, September 2016.

[8] Cha Zhang, Dinei Florencio, and Charles Loop, “Point cloud attribute compression with graph transform,” *Image Processing (ICIP), 2014 IEEE International Conference on*, October 2014.

[9] Dorina Thanou, Philip A. Chou, and Pascal Frossard, “Graph-based motion estimation and compensation for dynamic 3d point cloud compression,” *Image Processing (ICIP), 2015 IEEE International Conference on*, September 2015.

[10] Ricardo De Queiroz and Philip A. Chou, “Compression of 3d point clouds using a region-adaptive hierarchical transform,” *IEEE Transactions on Image Processing 25*, June 2016.

[11] Robert A. Cohen, Dong Tian, and Anthony Vetro, “Point cloud attribute compression using 3-d intra prediction and shape-adaptive transforms,” in *DCC*, 2016.

[12] Radu Bogdan Rusu, “3d is here: Point cloud library,” *Robotics and Automation (ICRA), 2011 IEEE International Conference*, 2011.