Compression of Point Cloud Geometry with Random Access

Emanuele Palma and Ioan Tabus
Tampere University of Technology
Tampere, Finland
emanuele.palma@tut.fi, ioan.tabus@tut.fi

I. Introduction

The compression of point cloud geometry is a primary task in most immersive image compression technologies. The current standardization projects for compression of immersive image and video modalities show the importance foreseen for point cloud compression applications. In this poster we show results for the compression of point cloud geometry, when one imposes additional functionality requirements in the form of random access to various parts of the scene or object inside one point cloud. We first propose a method for segmenting a point cloud image based on the color attributes at each node, using the mean shift algorithm on several 2D projections, first independently, and then using a fusion approach, so that in the end we obtain a relatively small number of regions, i.e., clusters of 3D points. The next task is to compress the point cloud, with random access to each segment, by encoding independently in a lossless mode, using one octree for each segment. Finally we present results of compressing separately the disjoint point clouds as compared to the performance of compressing the initial point cloud in its entirety at once by a single octree.

II. Point cloud segmentation

The segmentation of objects from a point cloud representation is an important topic of image analysis per se, and for a recent work in the area one with an extended review of the literature one can see e.g., [1]. In here we are not interested in the accuracy of the split into objects, but rather are interested in decomposing the scene in a number of point clusters and analyze the performance of compression with random access. One recent work, [2], uses the segmentation of point clouds into a large number of clusters with the goal of improving the efficiency of coding the color attributes. In here we are interested in a much smaller number of objects, and the focus is on encoding the underlying geometry of the point cloud. The pointcloud can be segmented in several ways, e.g., using only the geometry information, or using only the color attributes at each pixels, or both the geometry and color information, see e.g., [2]. In here we use for segmentation the color information, and perform the segmentation in a number of steps, described next. The first segmentation step consists in performing the segmentation of a number of 2D projections of the pointcloud. We consider as an example the point cloud with a high voxel density, represented in Figure 1, where the space is split into 1024^3 voxels, out of which the occupied voxels represent a tiny part, of about 800000 voxels. We first perform two projections of the point cloud to XY plane, one from front, and one from the back, see Figure 2. Each projection from Figure 2 is an RGB image with a good resolution, 1024×1024 pixels, and we perform the mean shift segmentation [3] of the images using the parameters $h_r = 13$, $h_s = 13$, and impose a minimum size of $N_{reg} = 800$ for declaring a region. The results of the segmentation are shown in Figure 3 a) and b), having a number of 11 and 8 regions, respectively. In a second segmentation step we register the regions of the segmentations into a single segmentation, and at the same time we impose constraints on the number of regions in the final segmentation, obtaining the refined segmentations from Figure 4, which has $M = 6$ regions, $\Omega_1, \ldots, \Omega_M$. The labels in the two 2D images are traced back to the initial point cloud, resulting in the incomplete labeling of the point cloud from Figure 5a). Since the front and back xy projections contain only approximately half of the point clouds (the other points are occluded in these two projections) one has to fill-in labels for the occluded points. The fill-in process is as follows: we take the $M = 6$ regions obtained in the XY projections, and fit by $k-$ means clustering a number of $k = 10$ clusters for each region Ω_i, resulting in $kM = 60$ cluster centers, $\{C_1, \ldots, C_{kM}\}$, where each center C_i is labeled by $\lambda_i \in \{1, \ldots, M\}$. We now traverse all the points of the point cloud that do not have yet a label. For each such point we take its color attribute and find its closest cluster center C_i and finally we label the point by λ_i. After this stage, each point of the point cloud is labeled with one of the labels $\{1, \ldots, M\}$, with the result shown in Figure 5 b).

III. Compression of the point cloud geometry

The encoding of point cloud geometry by octrees is a well established technique, for extensive references see, e.g., [4]. We developed our own routines for octree building and encoding, where we implement a number of variations of the standard octree technique. The initial point cloud is represented as an octree $\mathcal{T}$. We denote "Direct octree" the codelength per occopied voxel, $R^{DO} = \mathcal{L}^{DO}/N$, where N is the number of occupied voxels and $\mathcal{L}^{DO}$ is obtained as the direct coding of octree, where each pattern of 8 bits from an octree node is written to the bitstream $\mathcal{B}$, having length $\mathcal{L}^{DO}$. We note that the final codelength obtained is

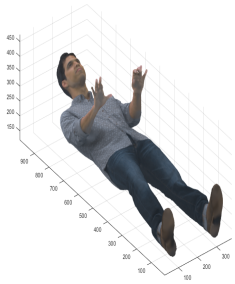

Fig. 1. Initial point cloud with 1024^3 voxels out of which 744000 are occupied voxels

a) b)

Fig. 2. Front view and back view obtained by projection of 3D points to xy plane

a) b)

Fig. 3. Initial segmentations by using mean shift algorithm, resulting in a) 8 classes; b) 11 classes

a) b)

Fig. 4. Refinement of the initial segmentations: a) 6 classes b) 6 classes

the same, no matter how the octree $\mathcal{T}$ is traversed. The most used entropy coding technique for improving the coding performance is to further pass the bitstream $\mathcal{B}$ through a zip encoder. However now the way the tree is traversed has a big importance. We experiment with three variants: breadth first order, depth first order, and Morton ordering [1]. The codelength per occopied voxel, $R^{GDO} = \mathcal{L}^{GDO}/N$, is defined similar to $R^{DO} = \mathcal{L}^{DO}/N$. Now $\mathcal{L}^{GDO}$ is the codelength of the gzip encoded bitstrean B, rearranged according to each traversal method.

IV. Experimental results

We show in Table I at the row "no random access" the results of compression ratios obtained when encoding independently the entire point cloud. In Table II we show the compression results obtained over each of the $M = 6$ disjoint point clouds obtained through the segmentation process of Section II. By properly weighting these results one obtaines the overall compression for the entire point cloud, but now having random access, which are shown in the row "random access" From Table I. The random access coding feature is thus shown to incure a loss of efficiency of about 0.9 bpov. Further improvement of the segmentation is expected to lead to a smaller penalty for the random access.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764951

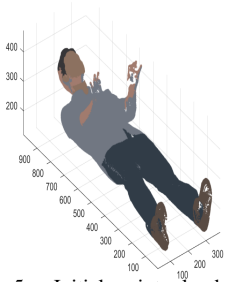

Fig. 5. Initial point cloud with only the labeled points of the Front/Back projection inserted

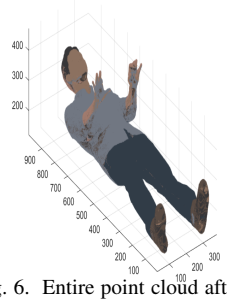

Fig. 6. Entire point cloud after the filling operation

TABLE I
Coding Results for the Entire Point Cloud

	Ordering	*Direct Octree* R^{DO} [bpov]	*Gzip Octree* R^{GDO} [bpov]
normal access	***Breadth First***	2,9791	**1,9754**
	Depth First	2,9791	1,9848
	Morton	2,9791	2,0376
random access	***Breadth First***	3,8757	**2,8445**
	Depth First	3,8757	2,8627
	Morton	3,8757	2,9124

TABLE II
Coding Results for Each Independent Point Cloud

	Ordering	*Direct Octree* R^{DO} [bpov]	*Gzip Octree* R^{GDO} [bpov]
PC label 1	***Breadth First***	6,7756	5,2201
	Depth First	6,7756	5,2116
	Morton	6,7755	5,2066
PC label 2	***Breadth First***	3,0827	2,1164
	Depth First	3,0827	2,1044
	Morton	3,0827	2,1581
PC label 3	***Breadth First***	3,4410	2,4567
	Depth First	3,4410	2,5206
	Morton	3,4410	2,5799
PC label 4	***Breadth First***	5,2810	4,2429
	Depth First	5,2810	4,2713
	Morton	5,2808	4,3100
PC label 5	***Breadth First***	4,3874	3,5462
	Depth First	4,3874	3,5169
	Morton	4,3872	3,5902
PC label 6	***Breadth First***	5,2326	4,1266
	Depth First	5,2326	4,1013
	Morton	5,2324	4,1349

References

[1] X. Lin, J.R. Casas and M. Pardas,"Temporally Coherent 3D Point Cloud Video Segmentation in Generic Scenes", IEEE Transactions on Image Processing, vol. 27, no. 6, pp. 3087-3099, June 2018.

[2] K. Zhang, W. Zhu and Y. Xu, "Hierarchical Segmentation Based Point Cloud Attribute Compression", IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 3131-3135, 2018.

[3] Comanicu, P. Meer: "Mean shift: A robust approach toward feature space analysis". IEEE Trans. Pattern Anal. Machine Intell., May 2002.

[4] R. Mekuria, K. Blom and P. Cesar, "Design, Implementation, and Evaluation of a Point Cloud Codec for Tele-Immersive Video," Circuits and Systems for Video Technology, IEEE Transactions on, vol. 27, no. 4, pp. 828-842, Apr 2017