LEARNING-BASED LOSSLESS POINT CLOUD GEOMETRY CODING USING SPARSE TENSORS

Dat Thanh Nguyen and André Kaup

Chair of Multimedia Communications and Signal Processing
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Erlangen, Germany

ABSTRACT

Most point cloud compression methods operate in the voxel or octree domain which is not the original representation of point clouds. Those representations either remove the geometric information or require high computational power for processing. In this paper, we propose a context-based lossless point cloud geometry compression that directly processes the point representation. Operating on a point representation allows us to preserve geometry correlation between points and thus to obtain an accurate context model while significantly reduce the computational cost. Specifically, our method uses a sparse convolution neural network to estimate the voxel occupancy sequentially from the x, y, z input data. Experimental results show that our method outperforms the state-of-the-art geometry compression standard from MPEG with average rate savings of 52% on a diverse set of point clouds from four different datasets.

Index Terms— Point Cloud, Sparse Convolution, Deep Learning, G-PCC, VoxelDNN, SparseVoxelDNN.

1. INTRODUCTION

A Point Cloud (PC) is a set of discrete 3D points that represent 3D scenes or objects. Typical PCs contain millions of points and each point is represented by spatial coordinates (x, y, z) and attributes (e.g. color, velocity, etc.) which require a huge amount of storage. Recently, point clouds are becoming the common data structure in many 3D applications, and as a result, efficient point cloud compression (PCC) methods are highly required.

The most well-known methodologies were developed by the Moving Picture Expert Group (MPEG) [1]. Two approaches have been proposed: Video-based PCC (V-PCC) focuses on dynamic point clouds and Geometry-based PCC (G-PCC) targets static content. V-PCC projects 3D point clouds onto 2D planes and utilizes a 2D video codec to encode the projected videos. On the other hand, G-PCC

operates in 3D space and uses 3D tools to encode the point cloud. In most of the methods, geometry and attributes are encoded independently. In this paper, we focus on the point cloud geometry (PCG) compression leaving the compression of attributes for future study.

The raw representation of a point cloud geometry is a point representation with millions of x, y, z values. Typically, those coordinate values are first quantized to integer precision prior to geometry coding. At this stage, the point cloud geometry can be converted to a voxel or octree representation. To obtain a voxel representation, the 3D volume is first divided into a fixed number of cubes in each dimension, each cube is called a voxel and a binary value is assigned to each voxel to indicate whether that voxel is occupied or not. The number of voxels per each dimension defines the resolution as well as the bit depth of a voxel block (1024 voxels = 1024 resolution = 10 bits depth). Geometric information (i.e., planes, surfaces, etc.) are preserved in a voxel representation. However, applying conventional operations such as convolution or 3D transforms on a voxel block requires significant computational power as voxel representations bring a large amount of redundancy regardless of the voxels' occupancy state. By recursively splitting a voxel block into eight sub-cubes and using bit 1 to signal an occupied sub-cube, bit 0 to signal an empty sub-cube until the desired precision, we obtain an octree representation. Octree representation lower the geometry bit rate compared to point-based or voxel-based representation, however, geometric information is lost in the octree representation.

In this work, we propose a point-based method for lossless point cloud geometry coding utilizing sparse convolution operation (SparseVoxelDNN). On the one hand, a point-based representation (which we represent using sparse tensors) eliminates the sparsity while retaining the geometric information of the point clouds and thus can be exploited by our context model. On the other hand, with sparse tensor in combination with sparse convolution operation we can easily process point cloud data with low computational complexity. We demonstrate experimentally that our SparseVoxelDNN outperforms G-PCC and the state-of-the-art learning-based geometry compression method by a large margin.

2. RELATED WORK

Most of the existing point cloud geometry compression methods including MPEG G-PCC are based on an octree structure [2–5] and a trisoup representation. In the MPEG lossless geometry coder, to encode the octant value of each node, G-PCC exploits correlation within the octree structure by introducing many tools such as neighbour-dependent entropy context [6], intra prediction [7], or planar/angular coding mode [8, 9]. MPEG G-PCC is widely used for PCC comparison because of its efficiency and regular updates.

Other context modeling methods based on simpler approaches have been proposed [3, 4], aiming at predicting the occupancy pattern using the information from parent nodes, current node position or corresponding nodes at the previous time frame. However, those approaches introduce frequency tables which are collected from the higher octree level or the previous time frame and must be sent to the decoder along with the actual geometry bitstream.

Deep learning has been widely applied in image/video compression as well as for point cloud data. Similar to image/video compression, most of the learning-based lossy point cloud geometry compression methods are based on an auto-encoder architecture [10–12]. Specifically, in [10], 3D voxelized blocks are first encoded into a low dimensional latent space and the reconstruction is cast as a binary classification problem. Recently, an autoregressive-based lossless compression method VoxelDNN has been proposed in [13] which was further developed to a multi-scale approach by the authors [14]. Both approaches outperform G-PCC by a large margin on dense point clouds. However, using a voxel representation as input results in high computational costs even with a multi-scale version.

Sparse tensors and sparse convolution have been studied and applied to point cloud tening at reducing the complexity. However, most of the applications are in lossy geometry PCC scenarios [15–17]. In this work, we propose an approach to extract features from a sparse representation and then build an accurate context to estimate the voxel occupancy using autoregressive modeling. Autoregressive models have been proven to be the state-of-the-art likelihood estimation methods [18, 19], principally resulting in an accurate occupancy distributions. However, a drawback of autoregressive models is its computational cost. We hence employ sparse tensor and sparse convolution operation to partially reduce the cost.

3. PROPOSED METHOD

The details of our novel method SparseVoxelDNN will be presented in this section. Our aim is to encode occupancy voxel block v at resolution d containing very few occupied voxels. Instead of processing a voluminous and redundant block v, we represent block v in a sparse tensor. We first define a 3D raster scan order that scans voxel by voxel in depth, height and width order. For ease of notation, we index all voxels in block v at resolution d from 1 to d^3 in raster scan order with:

$$v_i = \begin{cases} 1, & \text{if } i^{th} \text{ voxel is occupied} \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

A voxel v_i with $i = x_i d^2 + y_i d + z_i$ is corresponding to a point at x_i, y_i, z_i in sparse tensor. For easier interpretation, we will use v_i to represent a point as well as a voxel in the following sections.

3.1. Occupancy distribution modeling

SparseVoxelDNN losslessly encodes a voxel block v of size d using a CABAC coder. Our aims is to build an accurate probability model to predict the joint probability $p(v)$ that can adapt to local occupancy state. We factorize the joint distribution $p(v)$ as a product of conditional distributions over the spatial dimension:

$$p(v) = \prod_{i=1}^{d^3} p(v_i | v_{i-1}, v_{i-2}, \dots, v_1). \quad (2)$$

Each term on the right hand side is the probability of voxel v_i being occupied given the occupancy state of all previous voxels in the 3D raster scan order including $v_{i-1}, v_{i-2}, \dots, v_1$. In [13], the authors directly estimate $p(v_i | v_{i-1}, \dots, v_1)$ as binary classification problem using a neural network called VoxelDNN. Instead, inpired by the work from Pixelcnn++ [19], VAE [20], we model the distribution in a natural way as a continuous distribution, by assuming there is a latent occupancy intensity γ with continous distribution, in which the occupancy distribution can be easily extracted. This also enables us to model other channels with a lot of flexibility. Specifically, we model the occupancy intensity of each voxel as a mixture of L logistic distributions which allows us to easily calculate the probability on the observed discretized value as:

$$\gamma \sim \sum_{l=1}^{L} \pi_l \text{logistic}(\mu_l, s_l)$$

$$\hat{p}(v_i = 0 | \pi, \mu, s) = \sum_{l=1}^{L} \pi_l \left[\sigma((x + 0.5 - \mu_l)/s_l) \right] \quad (3)$$

$$\hat{p}(v_i = 1 | \pi, \mu, s) = 1 - \hat{p}(x = 0 | \pi, \mu, s)$$

where π_l is the one hot coding weight which will determine whether the lth distribution to be picked, μ_l and s_l are the mean and scale parameter of a logistic distribution, while $\sigma()$ is the logistic sigmoid function (i.e. the cdf function of a logistic distribution). The better the distribution estimation is, the smaller will be the coding bitrate we obtain. Therefore,
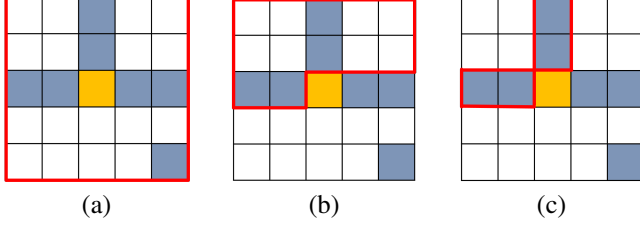
Fig. 1 Illustration of context in a 5×5 image. White pixels correspond to empty voxels, grey pixels correspond to occupied voxels and the yellow pixel corresponds to currently processed voxel. The area inside the red boundaries is the receptive field of a convolution operation of size 5×5. (a): Conventional convolution without masks (violating the causality constraint). (b): Conventional convolution with masks [13]. (c): Sparse convolution with our sparse masks.

we train our SparseVoxelDNN to estimate π_l, μ_l and s_l by minimizing the negative log-likelihood of $\hat{p}$.

The conditional distribution in (2) requires a causality constraint on our SparseVoxelDNN network. In Algorithm 1 we show the type A and type B mask construction which will be applied on sparse convolution filter at the first layer and the subsequent layers to enforce causality, respectively. Note that in sparse convolution implemented by Nvidia [21] $kernel\_size = k_D k_H k_W$ where k_D, k_H, k_W are the 3D kernel sizes. Simple examples of context in 2D space with and without masks are shown on Figure 1. Our masks strictly enforce the causality while eliminating all the redundancy within the receptive area and thus providing an accurate context.


Algorithm 1: Mask construction and application

Input: $mask\_type$
1   $kernel\_size \leftarrow k_D * k_H * k_W$
2   $mask \leftarrow ones\_like(kernel)$
3   $mask\left[kernel\_size//2 + mask\_type == B, :, :\right] = 0$
4   $kernel = kernel * mask$

Our SparseVoxelDNN network architecture for a point cloud at resolution d is shown in Figure 2. From the first layer to the "Sparse to Dense" layer, we use sparse tensor representation. Therefore, we have to make sure that the coordinate corresponding to the next voxel in 3D raster scan order can be generated from the previously decoded coordinates regardless of its occupancy state. We enforce generation of new coordinates on the first layer in such a way that the output coordinates are the outer product of the kernel size and the input coordinates. Next, we employ two residual blocks before converting a point cloud from a sparse tensor to a voxel representation in the "Sparse to Dense" layer. The last layer is a $1 \times 1 \times 1$ 3D convolution and thus does not violate the causality. The number of filters h in the last layer is the number of parameter in (3). For example, in our experiments, with a mixture of 5 logistics, we have $h = 15$.
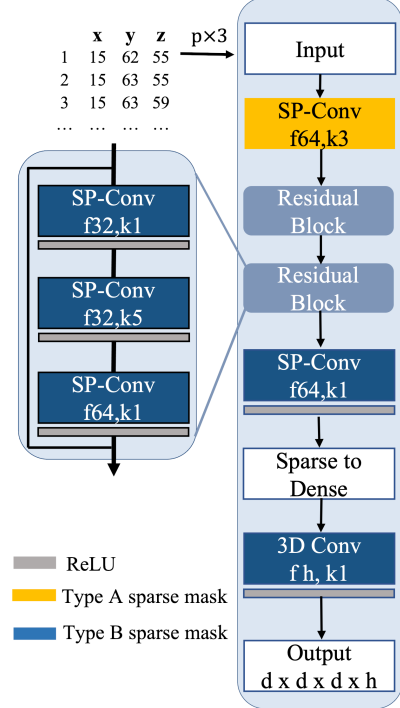
Fig. 2 SparseVoxelDNN architecture. A sparse type A mask is applied in the first layer (yellow block) and sparse type B masks afterwards. 'f64,k3' stands for 64 filters, kernel size 3. White blocks are Input/Output or converting layer and thus do not contain any trainable parameters.

3.2. SparseVoxelDNN Coder

Ultilizing a sparse tensor does not harm the computational cost as voxel representation but it is still not feasible to process the whole point cloud as a single block. Therefore, prior to the actual encoding process, we partition the point cloud into blocks of size 64 and use an octree to signal the partitioning. By doing this, most of the empty space in the point clouds can be removed. Experimental results show that the octree signal costs less than 2% of the bitstream. We encode each block of size 64 using our SparseVoxelDNN combined with a CABAC coder. Specifically, we encode voxel by voxel sequentially in 3D raster scan order. Everytime a voxel distribution is obtained, we pass the distribution to an arithmetic coder. The voxel is then fed back to SparseVoxelDNN (in the sparse form) to predict the occupancy distribution of the next voxel. Sequential prediction is a general drawback of all autoregressive-based approaches. We minimize the cost by ultilizing sparse tensor and sparse convolution in our distribution model to have a lighter and faster prediction.

3.3. Experimental setup

As training data we consider point clouds from four different datasets, including Microsoft Voxelized Upper Bodies (MVUB) [22] - dynamic dense voxelized upper-body point clouds, MPEG CAT1 - static sparse point clouds for cul-

Table 1: Number of blocks in the training sets of each model

	MVUB	8i	CAT1	Owlii	Total
Block 64	317763	235369	2499	358834	914465

Table 2: Comparions of our method against G-PCC v14 and VoxelDNN

		G-PCC	VoxelDNN		SparseVoxelDNN	
	Point Cloud	bpov	bpov	Gain over G-PCC	bpov	Gain over G-PCC
MVUB	Phil	1.15	0.82	-28%	0.40	-66%
	Ricardo	1.07	0.75	-29%	0.35	-67%
	Average	**1.11**	**0.79**	**-28.5%**	**0.37**	**-66.4%**
8i	Redandblack	1.09	0.70	-36%	0.30	-64%
	Loot	0.95	0.61	-36%	0.33	-65%
	Thaidancer	1.00	0.66	-34%	0.32	-68%
	Boxer	0.95	0.59	-38%	0.30	-68%
	Average	**1.00**	**0.64**	**-36.0%**	**0.34**	**-66.5%**
CAT1	Frog	1.92	1.70	-10%	1.23	-36%
	Arco Valentino	4.85	4.99	+3%	3.03	-38%
	Shiva	3.67	3.50	-4%	2.66	-28%
	Average	**3.48**	**3.40**	**-3.8%**	**2.31**	**-33.6%**
USP	BumbaMeuBoi	5.41	5.07	-6%	2.52	-53%
	RomanOiLight	1.86	1.62	-13%	1.52	-19%
	Average	**3.64**	**3.35**	**-9.6%**	**2.02**	**-35.8%**

Table 3: Average runtime (in seconds) of different encoders comparing with G-PCC

	G-PCC	VoxelDNN	SparseVoxelDNN
(Enc)	1.6	355	7.2
(Dec)	1.1	330	229

sparse point clouds with around 66% of bitrate reduction on MVUB and 8i. This could be explained by the fact that the context model can efficiently exploit the relation between points and predict a more accurate probability on dense point clouds than on sparse point clouds. However, despite the different content type and density of input point clouds, we obtain a significant bitrate reduction of 33.6% and 35.8% in average on CAT1 and USP compared to G-PCC, respectively. Compared to VoxelDNN, our method obtains, on average, a further bitrate reduction of 29% on all tested point clouds. VoxelDNN does not achieve a clear gain on sparse point clouds from CAT1 and USP, as voxel blocks contain a large amount of redundancy and thus the context model is less efficiency compared to the context model with sparse representation (as shown in Figure 1).

Table 3 reports the encoding, decoding time in seconds of our method, G-PCC, and VoxelDNN [13]. Our approach obtains a significant bitrate gain over G-PCC and the price to be paid for this performance improvement is a higher computational cost with average encoding time of $4.7\times$ over G-PCC. Compared to VoxelDNN, SparseVoxelDNN is $49\times$ faster in encoding runtime. There is still the common decoding complexity issue of autoregressive-based approaches (VoxelDNN and SparseVoxelDNN), however the problem can be alleviated with multiscale structures as disscussed in [14, 19].

5. CONCLUSIONS

In this paper, we have presented a novel lossless point cloud geometry compression method called SparseVoxelDNN. We represent the point cloud using both octree, voxel and point representation in a way that is most suitable to their characteristics. We build a 3D autoregressive model to predict continuous occupancy distribution by deploying 3D sparse convolution masks in a point-based neural network. This approach enables us to accurately model the occupancy distribution and obtain significant gains, with average rate savings of 52% over G-PCC while avoiding expensive computations. We are now developing lossy point cloud compression methods using sparse convolution and sparse tensors.

tural heritage and other 3D photography applications, MPEG Owlii [23] and 8i [24] - dynamic dense voxelized full-body point clouds. To enforce the fairness between the datasets in which we select point clouds for testing, point clouds from MPEG CAT1 are sampled to 10 bit precision as in MVUB, Owlii and 8i. To train a SparseVoxelDNN model of size $d = 64$, we divide all selected PCs into occupied blocks of size $d \times d \times d$. Table 1 reports the number of blocks from each dataset for training.

We add data augmentation techniques in the training phase to increase the generalization of our network. Specifically, rotation and sub-sampling are applied on training blocks. The training and testing process are performed on a a GeForce RTX 3090 GPU with Adam optimizer with learning rate 10^{-4} and accumulating gradients. We set the batch size to 8, accumulating step to 16 and early stopping callback with patience of 5 epochs. We evaluate our geometry compression methods on a set of 10 bit point clouds from MVUB, 8i, CAT1 which are not yet used in the training phase and two inanimate sparse point clouds from the University of São Paulo, Brazil (USP) [25].

4. EXPERIMENTAL RESULTS

We compare the encoding bitrate measured in average bit per occupied voxel (bpov) of our method with MPEG G-PCC v14 [1] and the state-of-the-art autoregressive-based method VoxelDNN [13]. Table 2 reports the *bpov* of these approaches as well as the gain of SparseVoxelDNN and VoxelDNN over G-PCC for each point cloud and the average per dataset.

First, we observe that SparseVoxelDNN outperforms both G-PCC and VoxelDNN by a large margin in all test point clouds. Our method works better on dense point clouds than

6. REFERENCES

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and*

Selected Topics in Circuits and Systems, vol. 9, no. 1, pp. 133–148, 2018.

[2] R. Schnabel and R. Klein, "Octree-based point-cloud compression.," *Proceedings of the 3rd Eurographics/IEEE VGTC Conference on Point-Based Graphics*, vol. 6, pp. 111–120, 2006.

[3] D. C. Garcia and R. L. de Queiroz, "Context-based octree coding for point-cloud video," in *Proc. IEEE International Conference on Image Processing (ICIP)*, September 2017, pp. 1412–1416.

[4] D. C. Garcia and R. L. d. Queiroz, "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry," in *Proc. IEEE International Conference on Image Processing (ICIP)*, October 2018, pp. 1807–1811.

[5] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression," *arXiv:2005.07178*, May 2020.

[6] "Neighbour-dependent entropy coding of occupancy patterns," in *TMC3, ISO/IEC JTC1/SC29/WG11 input document m42238, Gwangju, Korea, January 2018*.

[7] "Intra mode for geometry coding," in *TMC3, ISO/IEC JTC1/SC29/WG11 input document m43600, Ljubljana, Slovenia, July 2018*.

[8] "Planar mode in octree-based geometry coding," in *TISO/IEC JTC1/SC29/WG11 input document m48906, Gothenburg, Sweden, July 2019*.

[9] "An improvement of the planar coding mode," in *ISO/IEC JTC1/SC29/WG11 input document m50642, Geneva, CH, Oct 2019*.

[10] M. Quach, G. Valenzise, and F. Dufaux, "Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression," in *Proc. IEEE International Conference on Image Processing (ICIP)*, September 2019, pp. 4320–4324.

[11] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned point cloud geometry compression," *arXiv preprint arXiv:1909.12037*, 2019.

[12] A. F. Guarda, N. M. Rodrigues, and F. Pereira, "Point cloud geometry scalable coding with a single end-to-end deep learning model," in *Proc. IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3354–3358.

[13] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4617–4629, 2021.

[14] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Multiscale deep context modeling for lossless point cloud geometry compression," in *Proc. IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 2021, pp. 1–6.

[15] T. Huang and Y. Liu, "3d point cloud geometry compression on deep learning," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 890–898.

[16] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," *arXiv preprint arXiv:2011.03799*, 2020.

[17] W. Yan, S. Liu, T. H. Li, Z. Li, G. Li, et al., "Deep autoencoder-based lossy geometry compression for point clouds," *arXiv preprint arXiv:1905.03691*, 2019.

[18] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," *arXiv:1601.06759 [cs]*, August 2016.

[19] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications," *arXiv:1701.05517 [cs, stat]*, January 2017.

[20] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," *Advances in neural information processing systems*, vol. 29, 2016.

[21] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[22] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft voxelized upper bodies - a voxelized point cloud dataset," in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M72012*. May 2016.

[23] Yi Xu, Yao Lu, and Ziyu Wen, "Owlii dynamic human mesh sequence dataset," .

[24] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset," in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*. Geneva, January 2017.

[25] M. Zuffo, "UNIVERSITY OF SAO PAULO POINT CLOUD DATASET," Nov. 05 2021[Online].