Deep learning-based Point Cloud Compression

Maurice Quach

université PARIS-SACLAY

Deep learning-based Point Cloud Compression

Compression de nuages de points par apprentissage profond

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat : Traitement du signal et des images
Graduate School : Informatique et sciences du numérique
Référent : CentraleSupélec

Thèse préparée dans l'unité de recherche Laboratoire des signaux et systèmes (Université Paris-Saclay, CNRS, CentraleSupélec), sous la direction de Frédéric DUFAUX, directeur de recherche et le co-encadrement de Giuseppe VALENZISE, chargé de recherche

Thèse soutenue à Paris-Saclay, le 6 juillet 2022, par

Maurice QUACH

Composition du jury

Géraldine MORIN Professeure, IRIT, Université de Toulouse, Toulouse, France	Présidente & Examinatrice
Mylene FARIAS Professeure associée, Department of Electrical Engineering, Universidade de Brasília, Brazil	Rapporteur & Examinatrice
Fernando PEREIRA Professeur, Instituto de Telecomunicações, Instituto Superior Técnico, Lisbon, Portugal	Rapporteur & Examinateur
Marius PREDA Maître de conférences, ARTEMIS, Telecom SudParis, IP Paris, France	Examinateur
Frédéric DUFAUX Directeur de recherche, CNRS, CentraleSupélec, Université Paris-Saclay (L2S)	Directeur de thèse
Giuseppe VALENZISE Chargé de recherche, CNRS, CentraleSupélec, Université Paris-Saclay (L2S)	Coencadrant

THESE DE DOCTORAT
NNT : 2022UPASG051

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Compression de nuages de points par apprentissage profond

Mots clés : nuage de points, apprentissage profond, compression, évaluation de la qualité

Résumé : Les nuages de points deviennent essentiels dans de nombreuses applications et les progrès des technologies de capture conduisent à des volumes de données croissants. La compression est donc essentielle pour le stockage et la transmission. La compression des nuages de points peut être divisée en deux parties : la compression de la géométrie et des attributs. En outre, l'évaluation de la qualité des nuages de points est nécessaire afin d'évaluer les méthodes de compression des nuages de points. La compression de la géométrie, la compression des attributs et l'évaluation de la qualité constituent les trois parties principales de cette thèse.

Le défi commun à ces trois problèmes est la parcimonie et l'irrégularité des nuages de points. En effet, alors que d'autres modalités telles que les images reposent sur une grille régulière, la géométrie des nuages de points peut être considérée comme un signal binaire parcimonieux dans un espace 3D et les attributs sont définis sur la géométrie qui peut être à la fois parcimonieuse et irrégulière.

Dans un premier temps, l'état de l'art des méthodes de compression de la géométrie et des attributs est passé en revue, en mettant l'accent sur les approches basées sur l'apprentissage profond. Les défis rencontrés lors de la compression de la géométrie et des attributs sont examinés, avec une analyse des approches actuelles pour les résoudre, leurs limites et les relations entre l'apprentissage profond et les approches traditionnelles. Nous présentons nos travaux sur la compression de la géométrie : une approche de compression de la géométrie avec perte basée sur la convolution avec une étude sur les facteurs de performance clés pour ces méthodes et un modèle génératif pour la compression de la géométrie sans perte avec une variante multi-échelle atténuant ses problèmes de complexité. Ensuite, nous présentons une approche basée sur le pliage pour la compression d'attributs qui apprend un mapping du nuage de points à une grille 2D afin de réduire la compression d'attributs de nuages de points à un problème de compression d'images. De plus, nous proposons une métrique de qualité perceptive profonde différentiable qui peut être utilisée pour entraîner des réseaux de compression géométrique de nuages de points avec perte tout en étant corrélée avec la qualité visuelle perçue, ainsi qu'un réseau neuronal convolutif pour l'évaluation de la qualité des nuages de points basé sur une approche d'extraction de patchs. Enfin, nous concluons la thèse et discutons des questions ouvertes dans la compression des nuages de points, des solutions existantes et des perspectives. Nous soulignons le lien entre la recherche actuelle sur la compression des nuages de points et les problèmes de recherche dans des domaines adjacents, tels que le rendu dans l'infographie, la compression des maillages et l'évaluation de la qualité des nuages de points.

Title: Deep Learning-based Point Cloud Compression

Keywords: point cloud, deep learning, compression, quality assessment

Abstract: Point clouds are becoming essential in key applications with advances in capture technologies leading to large volumes of data. Compression is thus essential for storage and transmission. Point Cloud Compression can be divided into two parts: geometry and attribute compression. In addition, point cloud quality assessment is necessary in order to evaluate point cloud compression methods. Geometry compression, attribute compression and quality assessment form the three main parts of this dissertation.

The common challenge across these three problems is the sparsity and irregularity of point clouds. Indeed, while other modalities such as images lie on a regular grid, point cloud geometry can be considered as a sparse binary signal over 3D space and attributes are defined on the geometry which can be both sparse and irregular.

First, the state of the art for geometry and attribute compression methods with a focus on deep learning based approaches is reviewed. The challenges faced when compressing geometry and attributes are considered, with an analysis of the current approaches to address them, their limitations and the relations between deep learning and traditional ones. We present our work on geometry compression: a convolutional lossy geometry compression approach with a study on the key performance factors of such methods and a generative model for lossless geometry compression with a multiscale variant addressing its complexity issues. Then, we present a folding-based approach for attribute compression that learns a mapping from the point cloud to a 2D grid in order to reduce point cloud attribute compression to an image compression problem. Furthermore, we propose a differentiable deep perceptual quality metric that can be used to train lossy point cloud geometry compression networks while being well correlated with perceived visual quality and a convolutional neural network for point cloud quality assessment based on a patch extraction approach. Finally, we conclude the dissertation and discuss open questions in point cloud compression, existing solutions and perspectives. We highlight the link between existing point cloud compression research and research problems to relevant areas of adjacent fields, such as rendering in computer graphics, mesh compression and point cloud quality assessment.

Acknowledgements

I would like to thank my PhD advisors, Frédéric Dufaux and Giuseppe Valenzise, for their continued support throughout this journey. During these three years, I very much enjoyed our passionate discussions and your amazing guidance. I feel very fortunate and honored to have worked with such kind and exceptional advisors.

I would like to thank the members of the jury, Géraldine Morin, Mylene Farias, Fernando Pereira, Marius Preda and Dong Tian, for the detailed reviews, insightful comments and inspiring exchanges. In particular, special thanks for their attention to this work and the great discussions that ensued.

I am very grateful to Dong Tian, Jiahao Pang and Muhammad Asad Lodhi at InterDigital for the exciting internship on Point-based LiDAR Point Cloud Compression during the last year of the thesis. Especially, I would like to thank Dong Tian for his persistence in making the internship possible and providing me with this valuable opportunity. In addition, many thanks to Giuseppe, Frédéric, Stéphanie Douesnard, Audrey Bertinet and everyone who helped make it possible. Also, thanks to Aladine Chetouani, Dat Thanh Nguyen, Pierre Duhamel, Emin Zerman, Ali Ak, Patrick Le Callet, Hai Dang Nguyen, Nan Ding and Meta Can Kaya for the wonderful and inspiring collaborations.

I would like to thank my friends and colleagues, Milan Stepanov, Abhishek Goswami, Florent Chiaroni and Maxim Karpushin for the stimulating discussions.

Most importantly, I would like to thank my parents and my family for their unconditional love and support. In particular, I would like to thank my wife, Di Ju, who accompanied me and supported me during these three years.

I would also like to thank the ANR ReVeRy national fund (REVERY ANR-17-CE23-0020) which funded this thesis.

Contents

List of Tables

List of Figures

Acronyms

Notation	Description
APQI	Aggregated Patch Quality Index
BCE	Binary Cross Entropy
BD-PSNR	Bjontegaard-delta PSNR
BDBR	Bjontegaard-delta bitrate
bpov	bits per occupied voxels
BSDF	Bidirectional Scattering Distribution Function
CNN	Convolutional Neural Network
CTC	Common Test Condition
D1	Point-to-point metric
D2	Point-to-plane metric
DCM	Direct Coding Mode
DCT	Discrete Cosine Transform
FL	Focal Loss
FR	Full Reference
G-PCC	Geometry-based Point Cloud Compression
GFT	Graph Fourier Transform
GQI	Global Quality Index
JPEG	Joint Photographic Experts Group
KLT	Karhunen-Loève Transform
LiDAR	Light Detection And Ranging

Notation	Description
LoD	Level of Detail
LSTM	Long Short-Term Memory
MLP	Multi Layer Perceptron
MOS	Mean Opinion Score
MPEG	Moving Picture Experts Group
MSE	Mean Squared Error
MVUB	Microsoft Voxelized Upper Bodies
naBCE	neighborhood adaptive BCE
NR	No Reference
OR	Outlier Ratio
PC	Point Cloud
PCAC	Point Cloud Attribute Compression
PCC	Point Cloud Compression
PCGC	Point Cloud Geometry Compression
PCQA	Point Cloud Quality Assessment
PLCC	Pearson Linear Correlation Coefficient
PQI	Patch Quality Index
PSNR	Peak Signal-to-Noise Ratio
RAHT	Region-Adaptive Hierarchical Transform
RD	Rate-Distortion
ReLU	Rectified Linear Unit
RGBD	Red, Green, Blue, Depth
RMSE	Root Mean Square Error
RR	Reduced Reference
SLAM	Simultaneous Localization And Mapping

Notation	Description
SROCC	Spearman Rank Order Correlation Coefficient
SVR	Support Vector Machine
TDF	Truncated Distance Field
TSDF	Truncated Signed Distance Fields
TSP	Traveling Salesman Problem
V-PCC	Video-based Point Cloud Compression
WBCE	Weighted Binary Cross Entropy

1 - Introduction

1.1 . Context

Recent advances have increased the accuracy and availability of 3D capture technologies making point clouds an essential data structure for transmission and storage of 3D data. 3D point clouds are crucial to a large range of applications such as virtual reality [33], mixed reality [63], autonomous driving [194], construction [183], cultural heritage [172], etc. In such applications, large scale point clouds can have large numbers of points. Compression is thus essential for storage and transmission of point clouds.

Point clouds (Figure 1.1) are sets of points with x, y, z coordinates and associated attributes such as colors, normals and reflectance. Point clouds can be split into two components: the geometry, the position of each individual point, and the attributes, additional information attached to each of these points. A point cloud with a temporal dimension is referred to as a dynamic point cloud, and without a temporal dimension as a static point cloud.

The Moving Picture Experts Group (MPEG) has promoted two standards [155]: MPEG Geometry-based Point Cloud Compression (G-PCC) and MPEG Video-based Point Cloud Compression (V-PCC). These two standards have reached the Final Draft International Standard stage. G-PCC makes use of native 3D data structures such as the octree to compress point clouds while V-PCC adopts a projection-based approach based on existing video compression technologies. In addition, the Joint Photographic Experts Group (JPEG) has issued a call for evidence [2] on PCC.

Research on PCC can be categorized along two dimensions. On one hand, one

Figure 1.1: Point clouds visualizations. From left to right, "Dourado Site" [201], "soldier" [56], "phil" [116] and "Arco Valentino" [9]

can either compress point cloud geometry, i.e., the spatial position of the points, or their associated attributes. On the other hand, we can also separate works focusing on compression of dynamic point clouds, which contain temporal information, and static point clouds.

Due to the properties of point cloud capturing techniques, point clouds are typically sparse. This makes point clouds different from other modalities such images which have a dense regular geometry in the form of a regular grid. As a result, point cloud compression is a challenging problem.

1.2 . Objectives and contributions

The main objective of this thesis is to investigate the use of deep learning techniques for PCC. Specifically, we research the application of deep learning techniques for the following problems: lossy geometry compression, lossless geometry compression, lossy attribute compression and point cloud quality assessment. Indeed, deep learning for these problems was previously unexplored in the literature. In this thesis, we propose learning-based methods addressing these problems. In addition, using this broad perspective, we highlight the underlying challenges these closely related problems have in common namely point cloud sparsity and irregularity. We then expand this perspective into an extensive background on PCC and show how the sparsity and irregularity of point clouds can be resolved or alleviated to propose novel deep learning based approaches to these problems.

We first show how convolutional neural networks can be used for both lossy and lossless geometry compression. Decoding as classification is a flexible interpretation that is suitable for both lossy and lossless geometry compression. Hence, we propose a novel lossless compression approach based on masked convolution and then extend it with an efficient multiscale formulation. We also investigate deep learning based attribute compression and address the sparsity and irregularity of the geometry by reducing the problem to a image compression. Indeed, we propose a learning based approach to map point cloud attributes to a 2D regular grid. This enables the use of any image processing technique to point cloud attributes and we demonstrate how image compression methods can be used in such a context. Furthermore, we explore a differentiable deep perceptual point cloud quality metric to estimate the perceived visual quality of a point cloud from its geometry. This is strongly related to geometry compression as this metric is also suitable as a loss function for training lossy geometry compression networks. Indeed, the loss functions used in the literature tend to be weakly correlated with perceived visual quality. Loss functions that are more correlated with perceived visual quality could improve the performance of learning based lossy compression methods. In addition, we also propose a convolutional approach for point cloud quality assess-

ment that handles sparsity and irregularity via a projection approach. Indeed, the attributes lie on a sparse and irregular geometry which is a challenge for the design of objective quality metrics. The sparsity and irregularity can be addressed via a 2D projection approach. This then enables the use of 2D convolutional neural networks for quality assessment which are well studied for images.

1.3 . Structure

First, we present the state of the art for PCC in Chapter 2. Then, we address learning-based geometry compression with a lossy convolutional approach in Chapter 3. The interpretation of decoding as classification paves the way for lossless compression with an auto-regressive generative model in Chapter 4. However, such models exhibit high complexity due to the sequential generation of symbol probabilities. We address this issue with a multiscale prediction approach in Chapter 5. In Chapter 6, we propose a learning-based attribute compression method which is based on image compression and a learning-based mapping of point cloud attributes to a 2D grid. In lossy geometry compression methods such as the one present in Chapter 3, the neural network is jointly optimized for rate and distortion via a loss function. Thus, this formulation of this loss function is of great importance and for human visualization, the measure of distortion should ideally be strongly related to perceived visual quality. In Chapter 7, we propose a differentiable perceptual quality metric that is well correlated with perceived visual quality. Then, we address learning-based point cloud quality assessment with a projection based approach in Chapter 8. Overall, these different approaches have in common that the sparsity and irregularity of point clouds is a key problem across geometry compression, attribute compression and point cloud quality assessment. In the different chapters, we demonstrate how this problem can be resolved or attenuated for the different problems. Finally, we present conclusions, perspectives on open problems and the list of publications in Chapter 9.

2 - State of the Art on Point Cloud Compression

2.1 . Introduction

In this chapter, the state of the art for geometry and attribute compression methods with a focus on deep learning based approaches is reviewed. The challenges faced when compressing geometry and attributes are considered, with an analysis of the current approaches to address them, their limitations and the relations between deep learning and traditional ones. Current open questions in point cloud compression, existing solutions and perspectives are identified and discussed. Finally, the link between existing point cloud compression research and research problems to relevant areas of adjacent fields, such as rendering in computer graphics, mesh compression and point cloud quality assessment, is highlighted.

Common capturing techniques for point clouds include camera arrays [164], LiDAR sensing [69] and RGBD cameras [28]. It is important to consider capturing techniques as the resulting point clouds will exhibit specific characteristics which can be exploited for compression. Among these characteristics, the geometry resolution and the number of points are key for point cloud compression algorithms. The ratio between the number of points and the geometry resolution is the surface sampling density of the point cloud.

For example, camera arrays tend to produce point clouds that are dense, regularly sampled and amenable to 2D projections. RGBD cameras produce point clouds that can be represented on a 2D image with depth and color components; the resulting point clouds are dense, regularly sampled and susceptible to occlusions. Spinning LiDAR sensors are commonly used for autonomous driving, they tend to produce point clouds that can be approximately represented on a 2D image with depth and reflectance components. Such LiDAR point clouds are sparse and unevenly sampled in euclidean space but more dense and evenly sampled in spherical space which is an approximation of the sensing mechanism. In Figure 1.1, we show point clouds captured with aerial LiDAR ("Dourado Site"), camera arrays ("soldier" and "phil") and fused terrestrial LiDARs ("Arco Valentino").

The target application must also be considered. In the case of human visualization, a point cloud combined with a rendering method is a simplified model for a plenoptic function [108] which results in six degrees of freedom. Therefore, we aim to maximize the perceived visual quality given a rate constraint. For autonomous driving, we instead aim to maximize the performance of the autonomous driving function given a rate constraint. Note that it is actually possible to improve quality with lower bitrate in the case of point clouds. For example, sparse samplings of surfaces are more costly to code compared to dense samplings (watertight at

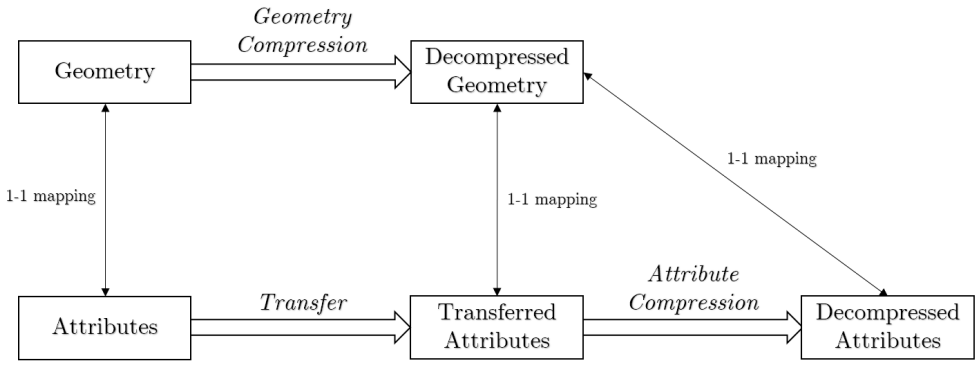


Figure 2.1: Point Cloud Compression by separating geometry and attribute compression.

a given resolution). In addition, compared to dense samplings, sparse samplings can result in inferior renderings and for autonomous driving, in safety issues due to occupancy false negatives (not sampled).

This can be explained by the sampling process. The actual state of each voxel can actually be considered as ternary instead of binary with unseen, empty and near surface states [50]. As as result, deciding the occupancy of unseen voxels is important when performing compression as a dense surface is less costly to code than a sparsely sampled surface. In particular, compression methods based on surface models and deep learning tend to complete missing points from surfaces in sparser point clouds.

Existing literature reviews provide an overview of MPEG standardization efforts [155, 71, 35], a classification taxonomy [138] and an analysis based on the coding dimensionality [34]. In this review, we will provide a general overview of PCC approaches with a focus on deep learning-based methods.

Point clouds can be divided into two components: the geometry and the attributes. The geometry is represented by points at given positions and the attributes attach information to each of these points. These two components can be coded separately, although it is necessary to transfer attributes if we code the geometry in a lossy manner as shown in Figure 2.1.

For geometry, one of the main difficulties is the sparsity of the signal. Geometry can be considered as a binary signal on a regular voxel grid. We define sparsity as the ratio between the number of points and the number of voxels. In addition, we introduce the concepts of global and local sparsity when evaluating point cloud sparsity at different Level of Details (LoDs). A point cloud at its original LoD has certain precision and coarser LoDs are obtained by reducing this precision usually via quantization. Local sparsity refers to sparsity localized at

finer LoDs and global sparsity to sparsity localized at coarser LoDs. Note that local and global sparsity are not mutually exclusive. The localization of sparsity in specific LoDs has been exploited for better entropy modeling of octree occupancy codes at different LoDs [95]. Such LoDs can be defined in duality with an octree representation by assuming the voxel grid dimensions are a common power of two. Then, quantization by a power of two and duplicate removal is identical to octree pruning. Following this, coarser LoDs are typically denser than finer LoDs for locally sparse point clouds which is important as performance of compression approaches depends on sparsity characteristics.

For attributes, the sparsity of the geometry which forms the support for the attributes is also one of the difficulties. An additional difficulty is the irregularity of the support: while geometry is a sparse signal on a regular support, attributes form a signal on an irregular support that may also be locally sparse, globally sparse or both. Irregularity can thus be defined as the absence of a regular support structure such as the regular grid sampling of images. Essentially, regular supports can be defined as the cartesian product of one or multiple regularly sampled intervals coresponding to different dimensions (horizontal, vertical, temporal, ...).

In the case of text, audio, images and video, the attributes lie on a regular grid, and the geometry, which represents a negligible amount of information. In the case of point clouds, we can consider that the information lies on a sparse subset of a regular grid and coding this sparse subset (the point positions, the geometry) is expensive. In addition, the information now lies on a sparse, irregular domain with irregular neighborhoods making attribute compression more difficult.

2.2 . Point Cloud Compression

We consider geometry and attribute compression separately and for each of them, we separate approaches tailored to LiDAR point clouds as they tend to have very specific structures.

2.2.1 . G-PCC and V-PCC standards

The MPEG G-PCC standard [3] features numerous techniques in order to handle different types of point clouds. It separates geometry coding and attribute coding following the scheme in Figure 2.1. Geometry can be coded using two modes: octree coding and predictive geometry coding. Octree coding is a general compression approach while predictive geometry coding targets LiDAR point clouds. In addition, compression of LiDAR point clouds using a sequential structure [129] and compression of point cloud sequences with inter prediction [128] are being investigated as exploratory experiments.

The MPEG V-PCC standard aims to compress point cloud sequences using video coding techniques. The point cloud is divided into patches, they are then

packed into a 2D grid and the geometry and attributes are compressed separately with video codecs [5]. Preprocessing and postprocessing operations such as smoothing, padding and interpolation are applied to improve reconstruction quality.

2.2.2 . Geometry Compression

Geometry compression can be considered as the compression of a sparse binary occupancy signal over a regular voxel grid. One of the main difficulties in compressing geometry is the sparsity of the signal which can be global, local or both.

By definition, global sparsity occurs when large volumes of space are entirely empty, e.g. when the point cloud is a scene composed of multiple objects. It can also be a result of the sampling process which samples only the surfaces of real world volumes resulting in the inside of the volumes being empty. Local sparsity can occur when the sampling density, the number of points per unit of volume, is low compared to the considered precision or voxel size. For example, this can happen in fused LiDAR point clouds of monuments and buildings as LiDAR sensors provide high precision captures with a sampling density that can be comparatively low and/or non uniform.

The structure of the point cloud is also primordial. Spinning LiDAR point clouds [113] are both globally and locally sparse in 3D space but with a model of the sensor, such as the spherical coordinate system, it becomes more regular, uniformly distributed and dense.

Regular point clouds

Octree Regular point clouds are quite dense allowing compression with a large range of operations. Among these operations, the simplest is a hierarchical voxel grid representation with the octree data structure. The geometry can be locally simplified by fitting a primitive such as planes, triangles, etc. Transform coding and dimensionality reduction are also possible by changing the voxel grid basis to another basis, for example, with projection to 2D or 1D spaces.

The most common geometry compression algorithm is octree coding [153]. 3D space is recursively divided into 8 equal subvolumes until the desired LoD is achieved. This is equivalent to encoding an occupancy map, i.e. a binary signal on a voxel grid, in a multiresolution manner. When a containing volume is empty, all its subvolumes are empty. Therefore, only the occupancy of valid voxels, whose parents are occupied, needs to be encoded. This explains the octree-voxel duality as each level of an octree can be associated with a voxel grid as shown in Figure 2.2.

For each octree node, coding can be done bitwise (per voxel) but also bytewise

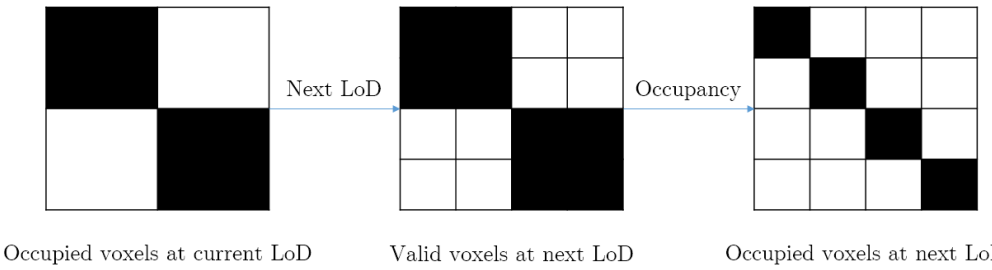


Figure 2.2: Simplified 2D view of an octree construction. Octrees can be seen as hierarchical binary voxel grids. An occupied voxel has at least one point within its volume and a valid voxel is possibly occupied while invalid voxels must be empty. Only valid voxels need to be coded at each LoD.

(per octree node). When coding occupancy bits bytewise, it is possible to reorder bits to improve compression performance. Specifically, approaches using tangent planes to guide reordering have been explored in the literature [153, 89, 95]. In addition, an octree node can have between 1 and 8 occupied voxels. The number of occupied voxels can be encoded to serve as a context for encoding occupancies [153]. For context based entropy modeling, the estimated occupancy probabilities can also be conditioned on the previously coded occupancies in the current octree node [3]. The structure of an octree node requires that at least one of eight occupancy bits is occupied ($256 - 1 = 255$ combinations) and the voxel grid interpretation enables the use of regular neighborhoods as contexts.

Taking advantage of the hierarchical nature of the octree, it is possible to use information from the parent node to code voxel occupancies at the current LoD Garcia and de Queiroz [66]. Such hierarchical contexts improve the compression performance compared to using no context. Furthermore, the neighbors of the parent node are also used as context in G-PCC neighbor-dependent entropy context [3]. Based on the duality between each octree level and a voxel grid, a regular neighborhood can be used to construct a voxel-based context using both current and parent level(s).

Transforms and dimensionality reduction Point clouds can be represented using 2D surfaces fitted to the point cloud. In addition, the points can also be projected onto 2D surfaces either using predefined surfaces (such as axis aligned planes) or fitted surfaces. Reducing the dimensionality to a single dimension makes the point cloud into a point sequence.

For dense point clouds, assuming that points are sampled from surfaces is common. Therefore, local point distributions can be modeled using surface models

such as planes [59], triangles [58], Bézier patches [49], etc. Surfaces can also be modeled implicitly with Bézier volumes [105] as the level set of a volumetric function. Sim et al. [160] also represent point clouds as a tree of spheres.

The dimensionality of point cloud compression can also be reduced using projection methods. For example, geometry can be compressed with dyadic decomposition Freitas et al. [62]; instead of an octree, a binary tree of 2D binary images is encoded. A simplified approach based on a single projection has also been studied [177]. In Kaya et al. [99], the point cloud is partially reconstructed from two depthmaps and the missed points are encoded separately. Projections can also be performed adaptively at a local level. For example, it is possible to project points on a 2D plane and encode a height field on this 2D plane [135]. Similarly, the MPEG Video-based Point Cloud Compression standard (V-PCC) projects the point cloud geometry to a 2D grid and performs compression with video compression technologies [5]. [112] improve upon V-PCC with occupancy-map-based rate distortion optimization and partition, [189] uses occupancy map information for fast decision of coding units and fast mode decision. Similar to V-PCC, [199] also propose a view-dependent video coding approach but focus on quality of experience for streaming and show competitive results in their subjective quality experiments. We notice that modeling point clouds with surface models is the same as projecting the points onto 2D surfaces with an height field that is zero everywhere i.e. the third dimension is entirely discarded.

The dimensionality can be further reduced by considering point clouds as sequences of points. Given a sequence of points and a prediction method, it is interesting to ask what is the optimal ordering for compression. However, this is difficult to compute due to the combinatorial natural of the problem. A proxy criterion for the reordering is a formulation as the Traveling Salesman problem which is considered in Zhu et al. [198] with binary tree based block partitioning.

Graph transforms have also been applied to geometry compression. When applying the graph transform for attribute compression, the geometry forms a support on which a graph and a corresponding basis can be built. In such approaches, the graph and a corresponding basis are built from a support and a graph transform is applied to a function lying on this support. However, the difficulty for geometry compression is that the geometry is the support itself or the voxel grid is the support for binary occupancy. In the first case, the support is not available at the decoder and, in the second case, the dimensionality is too high. This conundrum can be resolved by encoding a lower resolution point cloud (base layer) and upsampling it via surface reconstruction (upsampled base layer) [52]. This upsampled base layer can then be used as a support and the original geometry becomes an attribute on this support which resembles residual coding.

LiDAR point clouds

LiDAR point clouds are typically sparse in 3D space and can exhibit non uniform density due to their specific structures. Due to the scanning mechanism, spinning LiDAR point clouds are usually denser near the sensor and sparser far from the sensor. Such structures can be exploited for compression as these point clouds typically lie on a lower dimensional manifold in 3D space.

Point clouds captured with LiDAR sensors can often be represented as a sequence. LASzip [91] is a lossless, order-preserving codec specialized for LiDAR point clouds treating them as sequences and using metadata related to the LiDAR scanning mechanism [12]. In the special case of spinning LiDAR sensors, Song et al. [162] separate the point cloud into ground, object and noise layers. Typically, most points belong to the ground, and they form circular patterns which can be efficiently compressed. The point cloud formed by objects is usually globally sparse: the objects form clusters of points sparsely located in the scene.

LiDAR point clouds can be transformed to depth images by approximating the LiDAR sensor mechanism with quantized spherical angular coordinates. However, this transformation is often lossy on most LiDAR sensors albeit some new sensors make it lossless [137] by outputting a range image directly. The distortion introduced by this transformation will be acceptable or unacceptable depending on the application. In Ahn et al. [18], range images are compressed with predictive coding in hybrid coordinate systems: spherical, cylindrical and 3D space. Using mode selection, the range is predicted either directly, via the elevation or using a planar estimation. In Sun et al. [165], the range image is segmented into ground and objects, followed by predictive and residual coding for compression.

LiDAR point cloud packets can be directly compressed in some cases. The point cloud in this format has a natural 2D arrangement with the laser index and rotation. Full LiDAR point cloud frames are produced at 10Hz, while packets are typically produced at frequencies greater than 1000Hz. As a result, directly compressing LiDAR packets is more appealing for low-latency scenarios than full frames. Such packets can then be compressed using image compression techniques [176]. In addition, Tu et al. [173] propose the use of Simultaneous Localization And Mapping (SLAM) to predict and compress dynamic LiDAR point clouds.

In three dimensions, the structure of the LiDAR sensor can be used as a prior for octree coding. Specifically, sensor information such as the number of lasers, the position and angle of each laser and the angular resolution of each laser per frame can be encoded and exploited to improve compression. Only valid voxels, or plausible voxels according to the acquisition model, must be encoded when the acquisition process is known. This has been implemented as the angular coding mode in G-PCC [3].

2.2.3 . Attribute compression

The geometry forms a support for the attributes. The sparsity and the irregularity of this support is a source of difficulty for compressing point cloud attributes. Indeed, classical compression problems on images, videos, audios have attributes supported on regular grids. This results in regular neighborhoods for each sample leading to regular inputs for context modeling. The main difference between point cloud compression and these well studied compression problems is that the geometry, i.e. the support of the attributes, is sparse and irregular instead of being a regular grid. We show that the different point cloud attribute compression approaches actually all attempt to address sparsity, irregularity or both at once. Hence, sparsity and irregularity of this support are key challenges for point cloud attribute compression.

In [195], sparsity and irregularity are also referred to as unstructured (absence of a grid structure) and addressed with a graph transform. [54] specifically aim to address irregularity and [48] focus on sparsity after irregularity has been addressed with a graph transform [195]. We also note that for some types of point clouds such as LiDAR point clouds, the point cloud structure, e.g. acquisition structure, can be used to reduce sparsity and irregularity.

Regular point clouds

Point cloud attributes can be compressed using a graph transform [195]. We can construct a graph based on the point cloud geometry to define a fourier transform on graphs. This addresses the sparsity and irregularity issue as we are now in the graph frequency domain instead of the spatial domain. However, building such graphs and defining the specific transform for compression is a complex problem. Cohen et al. [48] focus on sparse point clouds and propose a method to compact attributes and generating efficient graphs for compression. de Queiroz and Chou [53] partition the point cloud into blocks and perform transform coding on point cloud attributes. Specifically, they propose a Gaussian Process Transform which is a variant of the Karhunen-Loève Transform (KLT) that assumes points are samples of a 3D zero-mean Gaussian Process. Such approaches resolve the irregularity by working in the spectral domain defined by a graph transform.

The Region-Adaptive Hierarchical Transform (RAHT) [54] is akin to an adaptive variation of a Haar wavelet for point cloud attribute compression. In addition, it has been found that reordering of the RAHT coefficients significantly improves the performance of the run-length Golomb-Rice coding compared to other entropy coding schemes [152]. Souto et al. [163] improve upon this RAHT with Set Partitioning in Hierarchical Trees [149]. Chou et al. [46] generalize the RAHT with volumetric functions defined on a B-spline wavelet basis. They show that the RAHT can be interpreted as a volumetric B-Spline of first order and that higher

order volumetric B-Splines eliminate blocking artifacts. In this case, the irregularity is resolved by considering the point cloud in an octree structure. Indeed, the geometric structure is irregular but the hierarchical relations are regular.

Sandri et al. [151] propose a number of extensions to the RAHT in order to compress plenoptic point cloud attributes. In plenoptic point clouds, the color of a point varies with the viewing direction. In particular, they find out that the combination of the RAHT and the KLT performs best: this method is referred to as RAHT-KLT. Krivokuća and Guillemot [103] compress plenoptic point clouds using a combination of clustering and separate encoding of diffuse and specular components using the RAHT-KLT. Krivokuća et al. [106] extend this approach to incomplete plenoptic point clouds (6-D) where a point may not have attributes for some viewpoints as it is not visible. This extends the RAHT to plenoptic point clouds whose plenoptic component is irregular. That is, for a given point, some viewpoints may not be valid and may not have attributes resulting in a 6-D structure.

G-PCC also proposes an attribute compression scheme based on prediction and lifting operations [3]. Multiple LoDs are constructed and predictors are built based on the attributes of nearest neighbors. This shows that attribute compression schemes need not follow the same structure as geometry compression albeit doing so may provide some complexity advantages (single pass). Indeed, by assuming that geometry is fully available when decoding attributes, any type of division, segmentation can be performed using this information. Attributes tend to be spatially correlated, hence a subset of point attributes can form a good basis for estimating remaining attributes. Similarly to RAHT, while the geometry is irregular, here a regular hierarchical structure is built.

Mekuria et al. [120] propose a complete point cloud codec for both geometry and attribute compression. It is also known as the "MPEG Anchor" as it was used as a comparison basis in early stages of the MPEG standardization process for point cloud compression. Attributes are projected to a 2D image and compressed using the JPEG codec. Zhang et al. [197] cluster the point cloud with mean-shift clustering and compress the attributes by projecting them on a 2D plane and performing a 2D Discrete Cosine Transform (DCT). Gu et al. [74] also explore the use of a 1D DCT to compress point cloud attributes. First, they group the points by blocks and order them by their Morton codes [127]. Then, the 1D DCT is applied and entropy coded. In these approaches, the attributes are mapped on a 2D or 1D regular grid in a spatially contiguous manner. As a result, the irregularity of the geometry is resolved or at least reduced.

Hou et al. [85] make use of sparse representations via a virtual adaptive sampling. Indeed, point cloud attributes have an irregular structure, reducing a block of voxels (voxel grid) to the subset of occupied voxels can be viewed as a vir-

tual adaptive sampling process. Gu et al. [75] improve upon this approach with inter-block prediction and run-length encoding. Shen et al. [157] also make use of this virtual adaptive sampling formulation and propose learning a multi-scale structured dictionary.

V-PCC [13] compresses attributes from 2D projections and video coding. The attributes are mapped to a 2D image, the image is then smoothed using different methods and compressed using video compression method [5]. The smoothing aims to maximize rate-distortion performance by completing values of empty pixels. Similarly, He et al. [83] propose an approach based on spherical projections to project both geometry and attributes upon a 2D image. The attribute image can then be compressed using image compression techniques.

In de Queiroz et al. [55], the authors propose a model description for volumetric point clouds that is independent of lighting and camera viewpoint. This generalizes the concept of plenoptic point clouds as the point cloud is no longer a simplified model for the plenoptic function Landy and Movshon [108]. Instead, the visual aspect of the point cloud can vary depending on the characteristics of the scene. This is related to the Bidirectional Scattering Distribution Function (BSDF) [26] which models how light is reflected or transmitted by a given surface. In perspective, by rendering point clouds with surfaces or volumes, it is possible to apply rendering techniques developed for meshes such as Physically Based Rendering Greenberg [72] approaches. This is an important consideration for attribute compression as it changes the nature and structure of the attributes being compressed.

LiDAR point clouds

LASzip [91] is a specialized codec for LiDAR point clouds. The point cloud is considered as a sequence and the attributes are compressed using delta coding, predictive coding and context-based entropy coding. Specifically, the LAS format [12] contains the number of returns and the return number for an emitted laser pulse. This information is used to perform context-based entropy coding and improve delta coding by selecting reference values that are more likely to be similar to the current value.

He et al. [83] use equirectangular projection to produce a 2D image from a LiDAR point cloud. Then, the authors test different image compression algorithms on color and reflectance images. Similarly, Houshiar and Nüchter [86] compressed color images using the PNG format.

Yin et al. [193] improve the predicting transform of G-PCC by exploiting normals. Specifically, the normals of two points are used to improve G-PCC mode selection. This demonstrates that attributes can be conditioned on the geometry by their position on the local characteristics of the geometry such as normals.

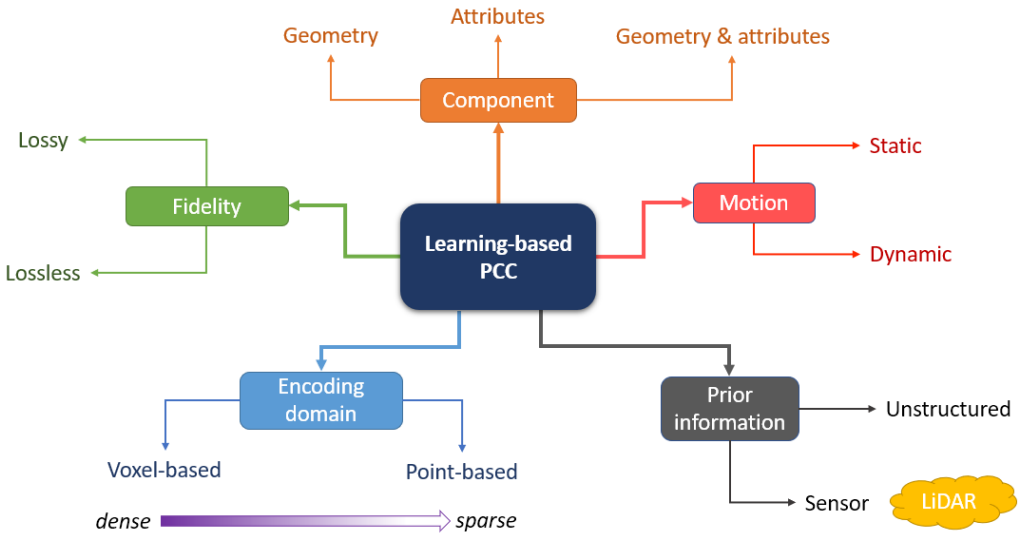


Figure 2.3: A taxonomy of Learning-based Point Cloud Compression.

2.3 . Deep learning based Point Cloud Compression

In Figure 2.3, we present a taxonomy of learning-based approaches for point cloud compression. A point cloud is composed of two components, geometry and attributes and can include motion (dynamic) or not (static). The components can be compressed separately or jointly and in a lossy or lossless manner.

We also differentiate approaches based on their encoding domain. Voxel based approaches are typically more suited to dense point clouds and point-based approaches to sparse point clouds. Indeed, while the complexity of voxel based approaches depends on the dimension of the voxel grid (precision), the complexity of point based approaches depends on the number of points.

It is also important to consider prior information. The most common source of structure in point clouds is usually the sensor. Spinning LiDAR sensors can be approximately modeled using a spherical coordinate system, point clouds from camera arrays have a representation as multiple 2D images, RGBD cameras produce a point cloud that can be stored on a single 2D image etc. Taking this prior information into account can yield significant compression gains but requires specific processing for each type of prior.

2.3.1 . Geometry compression

Lossy compression

By considering point clouds as a binary signal on a voxel grid, we can use Convolutional Neural Networks (CNNs) for point cloud geometry compression. Specifically,

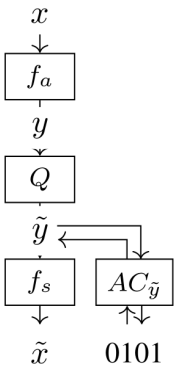


Figure 2.4: Simple autoencoder architecture for compression. f_a is an analysis transform, f_s a synthesis transform, Q refers to quantization and AC to arithmetic coding with its associated entropy model.

CNN based autoencoder architectures aim to transform the input into a lower dimensional latent space and reconstruct an output identical to the input. This can be interpreted from the point of view of transform coding with an analysis f_a and synthesis transform f_s as shown in Figure 2.4. The quantization part is also essential in that it must be differentiable in order to train the neural network. It is shown in [24] that additive uniform noise models quantization well during training and is differentiable. Additionally, the distribution of the latent space is modeled as part of the network and the resulting entropy enables rate distortion optimization of the network for specified rate distortion tradeoffs. The probabilities predicted by the entropy model can then be used by an arithmetic coder in order to encode and decode the latent space.

The decoding of a point cloud can be cast as a binary classification problem in a voxel grid. However, point clouds tend to be extremely sparse and this causes a class imbalance problem; this problem is resolved with the use of the focal loss [114]. Also, [77] propose a neighborhood adaptive loss function as an alternative to the focal loss.

A drawback of CNNs on voxel grids is the significant temporal and spatial complexity. This can be alleviated by using block partitioning [180, 79]. Wang et al. [182] go further along this path by making use of sparse convolutions [47] which reduces the temporal and spatial complexities even further. However, the disadvantage is that compression performance on sparse point clouds is degraded. Furthermore, Killea et al. [101] propose the use of lightweight 1D+2D separable convolutions for point cloud geometry compression resulting in less operations and network parameters with almost equivalent performance to their baseline model.

Overall, more elaborate neural network architectures are explored in Wang et al. [180, 182]. Wang et al. [182] propose a multiscale approach using sparse

convolutions. This approach is extended in Wang et al. [181] by making use of both the previous LoD and the current LoD by progressively building the current LoD. In particular, the authors study different levels of progressivity: n-stage with a sequential dependency chain, 8-stage which unpacks a voxel into 8 subvoxels one by one and 3-stage which unpacks a voxel dimension-wise into 2, 4 and finally 8 voxels. The authors also make use of the lossless-lossy compression scheme to adapt their method to point clouds with different levels of density.

Since the decoding can be cast as a classification problem, the compression performance can be improved significantly with adaptive thresholding approaches. Indeed, the density of the training dataset and the test dataset may vary. This can cause a mismatch between the distribution of predicted occupancy probabilities and the actual occupancies. This mismatch can be corrected with adaptive thresholding. For example, Wang et al. [180] encode the number of points as a threshold so that the reconstruction has the same number of points as the original. Guarda et al. [76] perform mode selection on models trained for different densities in order to achieve adaptive behavior.

A key observation is that CNNs for point cloud geometry compression tend to be biased towards a particular density depending on the training dataset and loss function parameters. Adaptive thresholding can correct this bias and can be based on factors such as the number of points and the distortion metric. Using mode selection on models targeting different densities is another way of making a density adaptive solution. The neighborhood adaptive loss function also removes the need to adjust focal loss parameters.

Commonly, one neural network is trained for each Rate-Distortion (RD) tradeoff. In that way, each network is optimized for a RD tradeoff at the cost of additional training time. Different approaches have been proposed to reduce the training time. Guarda et al. [80] propose the use of implicit-explicit quantization of the latent space which reduces the number of models to be trained. Implicit quantization controls the RD tradeoff with a scalar RD tradeoff parameter in the loss function when training. On the other hand, explicit quantization controls the RD tradeoff by varying the quantization step on the latent space. Surprisingly, the performance of implicit-explicit quantization is higher compared to implicit-only quantization.

In [111], CNNs are used for block prediction. More precisely, the blocks are predicted from already decoded neighboring blocks. Lazzarotto and Ebrahimi [110] perform residual coding and encode a residual between a ground-truth block and a degraded block compressed with G-PCC. [19] perform super-resolution using CNNs which improves the reconstruction quality significantly without any additional coding cost.

Lossless compression

Lossless compression algorithms can be extended using deep learning based entropy models. These entropy models predict occupancy probabilities which are then fed into an entropy coder. Different types of contexts can be used to make such predictions.

In the context of octree coding, parent node information can be used as context. Huang et al. [87] improve the coding of an octree by proposing a neural network based entropy model for octree codes. Specifically, this neural network takes the location and information from the parent node to predict occupancy probabilities. Biswas et al. [29] extend this to dynamic point clouds and propose the use of continuous convolutions to improve probability predictions using already decoded frames. However, such a context only provides limited information.

By introducing a sequential dependency in the voxel grid, one can use voxels at the current LoD as context. The neural network predicts the occupancy probability of each voxel, and the probabilities are then fed to an arithmetic coder.

Also, we can entirely remove the sequential dependencies by predicting the voxel occupancy using only the parent LoD. Que et al. [146] propose a neural network which takes a voxel grid context from the parent LoD to predict occupancy probabilities. These probabilities are then used for entropy coding. In addition, they predict an offset for each point in order to further refine coordinates. Although such approaches have a reasonable complexity, they do not take into account already decoded voxels at the current LoD which limits compression performance.

Kaya and Tabus [98] propose an approach that makes use of the current LoD while reducing complexity. Such approaches typically exhibit a sequential dependency in the current LoD. To alleviate this issue, the authors propose to process the current LoD by slice. In this way, the probabilities can be estimated slice by slice instead of voxel per voxel. The resulting parallelization significantly reduces complexity while preserving most of the context in the current LoD.

Point-based approaches

Point-based approaches process the point cloud as a set of points instead of a binary occupancy signal over a voxel grid. Yan et al. [190] proposed the use of a PointNet-like [37] autoencoder architecture for point cloud geometry compression. Wen et al. [186] proposes a similar scheme improved with curvature-based adaptive octree partitioning and clustering. Gao et al. [64] propose a more elaborate neural network architecture with a novel neural graph sampling module. Wiesmann et al. [187] propose a point-based neural network for LiDAR point cloud compression. The authors propose a convolutional autoencoder architecture based

on the KPConv [170] and a novel deconvolution operator to compress point cloud geometry.

LiDAR specific approaches

With some sensors, the LiDAR point cloud frame is constructed from packets. A packet is arranged as 2D data with respect to the laser ID and emission. Tu et al. [174] exploit this 2D structure to compress point cloud geometry with a convolutional Long Short-Term Memory (LSTM) neural network [84, 159] and residual coding; this is an extension of Tu et al. [173]. Tu et al. [175] take inspiration from video coding and divide dynamic point cloud frames into I-frames and B-frames [5]. The B-frames are predicted with U-net [148] based flow computation and interpolation extending work in Tu et al. [176].

Assuming that LiDAR point clouds can be represented as a 2D range image, Sun et al. [166] also take inspiration from video coding and propose the use of a convolutional LSTM neural network to compress the range images.

2.3.2 . Deep learning based attribute compression

One key difficulty for point cloud attribute compression is the irregularity of the geometry. Indeed, the geometry forms the support of the attributes. Hence, irregular geometry results in irregular neighborhoods which in turn make context modeling and prediction schemes difficult. Existing works attempt to resolve this irregularity by substituting the support.

First, it is possible to assume that the point cloud is a sampling of a 2D manifold and that it has a 2D parameterization that allows attributes to be projected onto a 2D image. This is similar to the concept of UV texture maps [36] in Computer Graphics except that here we seek to recover such a parameterization from a point cloud.

Alternatively, attributes can be directly mapped onto a voxel grid. Alexiou et al. [22] extend convolutional neural networks used for geometry compression to attribute compression. Specifically, they propose joint compression of geometry and attributes defined on a voxel grid.

Another approach is to design neural networks that embrace this irregularity. This can be done with point-based neural networks and convolutions expressed directly on points (point convolutions). Sheng et al. [158] propose a point-based neural network for point cloud attribute compression. In particular, they propose a second-order point convolution which improves upon the general point convolution. Biswas et al. [29] also make use of point convolutions for inter-frame coding of intensities for intensity prediction in dynamic LiDAR point clouds.

Isik et al. [92] compress point cloud attributes by representing them with volumetric functions. This is strongly related to Chou et al. [46] where attributes

are encoded using volumetric functions parameterized with a B-spline basis. In Isik et al. [92], volumetric functions are parameterized with coordinate based networks instead and the latent vectors of a hierarchy of volumetric functions are compressed with RAHT.

Overall, existing works handle the irregularity of the support, the geometry, by mapping attributes onto a 2D plane, using CNNs to compress attributes on a voxel grid or using point convolutions to directly define convolutions on the points.

2.4 . Discussion

2.4.1 . Geometry compression

In Table 2.1, we show an overview of geometry compression approaches for point clouds. Methods are divided into traditional and deep learning based. Different data structures employed for geometry compression: octree, surface models, sequence, graphs, voxel grids, range images etc.

For lossless octree coding, we can differentiate approaches depending on whether they exploit the previous LoD, the current LoD or both as a context. Using only the octree node information of the previous LoD presents a complexity advantage in that the information can be retrieved easily. Using the previous LoD (in a voxel grid sense) allows for parallel probability estimations on the current LoD that are more precise than when restrained to the octree node in the previous LoD. When employing the current LoD, a sequential constraint is necessary in that we can only use already decoded occupancies as a context. This results in a more precise entropy model at the cost of complexity. It is also possible to combine both the previous and the current LoD. The first advantage is that the previous LoD can provide coarse occupancy information for areas of the current LoD that are not yet decoded. The second advantage is that the sequential constraint can be loosened to form a hybrid approach that balances coding performance and complexity.

Projection and surface models transforms the data into another space which typically has a lower dimensionality or a different structure. Typical surface models include planes, triangles etc... Point based deep learning approaches can be thought of as projection based approaches in that the points are represented by a codeword in a latent space of lower dimensionality. The point cloud information is aggregated by a pooling operation which can be a max-pooling, an adaptive pooling operation or other types of pooling. Here, we refer to adaptive pooling as a pooling which aggregates information from points in a way that depends on the neighborhood structure. This is typical in point based convolution operators [170].

Point clouds can also be compressed as a sequence of points. This can be interesting as it allows for formulating point cloud compression as a variant of the

Category	Traditional	Deep Learning
Octree entropy coding		
Previous LoD, octree node	Garcia and de Queiroz [66]	Huang et al. [87]
Previous LoD	Schnabel and Klein [153]	Que et al. [146]
Current LoD		
Previous + current LoD	G-PCC [3]	Kaya and Tabus [98]
Projection / Surface models / Point based	Dricot and Ascenso [58, 59], Cohen et al. [49], Krivokuća et al. [105], Sim et al. [160], Freitas et al. [62], Tzamarias et al. [177], Kaya et al. [99], Ochotta and Saupe [135], VPC [13]	Yan et al. [190], Wen et al. [186], Gao et al. [64], Wiesmann et al. [187]
Sequence	Zhu et al. [198], Isenburg [91]	
Graph	de Oliveira Rente et al. [52]	
Voxel grid		
Vanilla		Wang et al. [180], Guarda et al. [79], Lazzarotto et al. [111], Lazzarotto and Ebrahimi [110]
+ Octree (previous LoD)		Wang et al. [182]
+ Octree (previous + current LoD)	G-PCC [3]	Wang et al. [181]
LiDAR		
2D range image model	Ahn et al. [18], Sun et al. [165]	Sun et al. [166]
2D packets	Tu et al. [173]	Tu et al. [174]
Temporal prediction	Tu et al. [176]	Tu et al. [175]
Entropy modeling with sensor model	G-PCC [3]	

Table 2.1: Approaches to point cloud geometry compression divided into traditional and deep learning based methods.

Traveling Salesman Problem (TSP) [198]. In addition, for LiDAR point clouds, a natural 1D structure exists and compression these points clouds as a sequence preserves this natural structure [91].

Voxel-based deep learning based methods are the most common approaches for geometry compression. Specifically, CNN autoencoders have been used to compress point cloud geometry as a binary signal over a regular voxel grid. We notice that an octree level can be represented by a voxel grid and that the two representations are interchangeable. In particular, the octree encodes binary voxel grids at multiple LoDs while eliminating sparsity of the binary signal and reducing the redundancy between LoDs; if the parent voxel is empty, all its children are empty and need not be encoded. In addition, block prediction and residual coding have also been explored using CNN architectures. In general, such networks predict a field of probabilities. Since the decoding can be seen as a classification problem, these approaches can be employed as both lossy (thresholding) and lossless (thresholding + residual or entropy coding) codecs.

The loss function is a crucial factor when training neural networks for geometry compression. While for lossless compression, the binary cross-entropy is suitable as it is directly related to the bitrate. For lossy compression, it is important that the loss function is suitable to the application. For example, loss functions correlated with perceived visual quality would be relevant for applications with human visualisation. A limitation of current approaches is the use of simple loss functions such as the binary cross-entropy which may not adequately model perceived visual quality. An open question is how to best design such loss functions for the best rate-distortion performance.

de Oliveira Rente et al. [52] codes geometry by starting from a low resolution point cloud, performing surface reconstruction on this low resolution point cloud and then coding a residual between this surface reconstruction and the original point cloud. As Lazzarotto and Ebrahimi [110] has explored point cloud residual coding with neural networks, the intersection of deep learning based residual coding and surface reconstruction can be an interesting avenue of research.

LiDAR point clouds tend to have very specific structures that typically lie in 1 or 2 dimensions rather 3 dimensions. As such, specialized approaches have been developed to compress LiDAR point clouds. LiDAR point clouds can be compressed as 2D range images, as 2D LiDAR packets but also applying specific temporal prediction techniques such as SLAM or flow estimation. This is applicable both for learning-based [175, 174] and handcrafted methods [176, 173].

Compression performance

MPEG [130] performed a comparison in the performance of different state of the art point cloud compression approaches. Overall, CNN based approaches [76, 181,

Category	Traditional	Deep Learning
Graph	Zhang et al. [195], Cohen et al. [48], de Queiroz and Chou [53]	Sheng et al. [158]
Octree	de Queiroz and Chou [54], Sandri et al. [152], Souto et al. [163], Chou et al. [46]	Isik et al. [92]
Octree (plenoptic point clouds)	Sandri et al. [151], Krivokuća and Guillemot [103], Krivokuća et al. [106]	
Voxel grid	Hou et al. [85], Gu et al. [75], Shen et al. [157]	Alexiou et al. [22]
2D grid	Mekuria et al. [120], VPC [13], Zhang et al. [197]	
Sequence	Gu et al. [74], Isenburg [91]	

Table 2.2: Approaches to point cloud attribute compression divided into traditional and deep learning based methods.

98] outperform the G-PCC codec with the drawback of additional computational complexity. CNN based methods seem to perform better on denser point clouds. Indeed, a difference is observed between 10-bit dense (watertight) point clouds (longdress, redandblack, loot, soldier) from [56] and sparser 12-bit point clouds (house without roof, statue klimt) which require downscaling the voxel grid in order to make the point cloud denser and obtain competitive compression performance. Also, CNN based methods using sparse convolution [181] achieve state of the art compression performance with lower computational complexity compared to dense convolutions. As described in [130], the runtime on GPU is faster to the G-PCC runtime on CPU. However, on a CPU it is still significantly slower (3.65 times slower for encoding and 37.1 times slower for decoding).

2.4.2 . Attribute compression

Traditional approaches handle the irregularity of the geometry by making use of the octree structure to build a multiscale decomposition of the attributes [54]. Also, we can build a graph and use the Graph Fourier Transforms (GFTs) or similar transforms to compress attributes [195]. In addition, attributes can be mapped onto a 2D grid and then compressed using image compression methods [120]; and similarly, points can be ordered into a sequence and compressed as such [74]. Another type of approach is to modelize the irregularity of the geometry with

a virtual adaptive sampling process [85]. Essentially, we modelize the attributes as a 3D field over a voxel grid and this field is sampled at occupied voxels.

3D approaches

Deep learning based methods handle the irregularity of the geometry by using a 3D regular space (voxel grid) [22], by mapping attributes onto a 2D grid or with the use of point convolutions to define CNNs that operate directly on the points [158]. Note that such point convolutions can often be seen as graph convolutions with the topology of the graph built from the point cloud geometry and its neighborhood structure.

The essence of attribute compression is handling the irregularity of the geometry. One approach is to map attributes into n-dimensional regular grids where we usually have $1 \leq n \leq 3$. In addition to this mapping, with the virtual adaptive sampling hypothesis, we can extrapolate attribute values to non-occupied voxels to simplify the problem to compression of a 3D regularly sampled field (instead of irregularly sampled). Another approach is to embrace the irregularity of the geometry and make use of the same octree structure that is commonly used for compression of the geometry.

In Table 2.2, we can see that deep learning based point cloud attribute compression is still relatively unexplored. Current deep learning based approaches are not competitive with G-PCC or V-PCC [3, 13]. One of the key difficulties is how to handle the irregularity of the geometry and from Table 2.2 we can see a few interesting points.

Alexiou et al. [22] has explored the use of CNNs on voxel grids for point cloud attribute compression. However, the combination between such approaches and the virtual adaptive sampling hypothesis explored in numerous works [85, 75, 157] for learning sparse dictionaries has not been explored. Indeed, we can interpret point cloud attributes as a color field defined everywhere on a voxel grid but only sampled at occupied locations. Then, these attributes could be compressed by first extrapolating the color field values everywhere on the voxel grid and compressing the resulting regularly sampled color field.

Octree-based approaches, such as the RAHT [54], are an interesting basis for deep learning based approaches. Specifically, sparse convolutions [47] may be a possible avenue to express convolutions on an octree structure as is done for geometry compression in Wang et al. [182]. As there is a duality between octree and voxel grids, the same question about extrapolation applies. It is an open question whether good convolutional transforms can be learned to compress attributes that lie on an irregular geometry and thus have irregular neighborhoods. In addition, inspired by RAHT, Isik et al. [92] propose a deep learning attribute compression approach based on volumetric functions parameterized by coordinate

based neural networks. They adopt a hierarchical structure of latent vectors which are compressed with RAHT.

Compressing attributes as volumetric functions is similar in spirit to virtual adaptive sampling and extrapolation. All three of these approaches remove the irregularity by assuming that the attributes are samples of an attribute field defined over 3D space (a volumetric function) which can be seen as a virtual adaptive sampling process. In other words, this is also extrapolating missing values from the available samples. The final result is that the attributes are now defined over the entire 3D space removing the sparsity and irregularity which may arise from the geometry.

Dimensionality reduction

Point cloud attribute compression is difficult due to the irregularity and sparsity of the support. It is possible to attenuate the irregularity with projections to lower dimensions which makes the support denser and more regular. In addition, techniques to remove the irregularity and sparsity in the 3D domain are actually applicable in all dimensions. This is interesting as point clouds are known to exhibit certain structures. For example, LiDAR point clouds have a 1D structure and most point clouds have a 2D manifold or surface structure.

Numerous approaches have explored the use of 2D images for point cloud attribute compression [120, 13, 197]. Mekuria et al. [120] have proposed a simple mapping procedure that makes use of Morton ordering and VPC [13] a more advanced procedure that segments the point cloud according to their normals and performs bin packing to store them in a 2D image. To the best of our knowledge, the combination of such handcrafted mappings and deep learning based image compression methods [24] has not been explored yet and may be promising.

Point cloud attributes can also be compressed as a sequence of points. This is especially suitable for spinning LiDAR point clouds which have a clear 1D structure. In Isenburg [91], both LiDAR point cloud geometry and attributes such as reflectance are compressed as a sequence. Currently, while research is considering compression with 2D structures [174], deep learning based approaches have not been explored for 1D structures yet.

Dimensionality reduction for point cloud attribute compression essentially exploits the spatial structure to alleviate irregularity and sparsity. How to find a new geometry (support) and a mapping from the original geometry to the new geometry that results in the best rate-distortion performance remains an open question. A limitation of existing approaches is that the new geometry and its mapping are fixed and not optimized for rate-distortion.

Graph based approaches

Graph based approaches have been used to compress point cloud attributes with the GFT [195, 48, 53]. Point based approaches can be considered as a special case of graph based approaches where the graph is defined by the point cloud geometry. In Sheng et al. [158], point convolutions are proposed for the compression of attributes. Such convolutions have been shown to perform well for tasks such as classification and segmentation but the compression performance is currently lacking compared to traditional approaches such as the G-PCC implementation of RAHT [3]. Point based neural networks are interesting as the complexity of these approaches depends on the number of points and not on the number of voxels. The drawback is that we are not working on a regular voxel grid anymore which makes the design of such networks more difficult. As such, a currently open question is how to build high performance point convolutional neural networks for attribute compression?

Compression performance

Currently, deep learning based attribute compression approaches [92, 158, 22] are not yet competitive with the latest G-PCC codec. However, it is shown in [92] and [158] that these approaches are actually competitive with RAHT which is used in G-PCC but improved with additional coding tools such as predictive RAHT, context adaptive arithmetic coding and joint entropy coding of color as discussed in [92]. In addition, [92] actually show performance close to earlier versions of G-PCC at the cost of high complexity.

2.4.3 . LoD decomposition and compression

For both geometry and attribute compression, it is interesting to consider the point cloud decomposed into multiple LoDs. Specifically, with LoDs based on octree decomposition, we can consider sparsity for a given LoD as the number of occupied voxels divided by the number of valid voxels at this LoD. Then, we can consider point clouds whose coarse LoDs are sparse as globally sparse: it can be the case in a point cloud composed of multiple objects. And we can consider point clouds whose finer LoDs are sparse as locally sparse: this typically happens in LiDAR point clouds or when the geometry resolution is too high compared to the number of points. This is important as certain methods are known to work better on dense point clouds (such as CNN based methods) than sparse ones.

In addition, decomposing the point cloud into multiple LoDs also allows us to mix differents compression methods. A common scheme is to perform block partitioning before applying compression methods that are more computationally expensive. This can be seen as compressing a set of block coordinates, which is by definition a point cloud, and then compressing a point cloud for each block.

Such decompositions can be performed according to LoDs which makes it possible to code sparse LoDs separately (with different methods) from dense LoDs. A very common situation is that the first LoDs are dense and the last few LoDs are sparse; this typically happens when point cloud geometry is described with a precision that is high compared to the number of points or when the sampling is not uniform (such as with LiDAR point clouds).

An interesting property of such decompositions is that it is always possible to use a lossless geometry compression approach followed by a lossy one. For example, when using block partitioning, the compression of block coordinates can be performed with not only octree coding but any kind of lossless point cloud geometry compression method. This is particularly useful combined with the observation that point clouds can be sparse at different LoDs and that certain compression methods are known to work better with certain densities. Considering a LoD-based point decomposition, it is thus interesting to consider compression schemes that select the most suitable compression method for each LoD based on their density. In addition, the geometric distortion introduced by such lossless-lossy schemes is bounded by the volume of losslessly coded voxels.

At a given LoD, the relation between compression algorithm, point cloud characteristics at a given LoD and compression performance remains an open question. How to best select and combine compression algorithms to adapt to different point cloud characteristics is also mostly unexplored. A limitation of the current state of the art is that algorithms tend to be specialized or manually tuned towards specific characteristics: dense, sparse, spinning LiDAR, etc.

The contents of this chapter have been published in M. Quach, J. Pang, D. Tian, G. Valenzise, and F. Dufaux, "Survey on Deep Learning-Based Point Cloud Compression," *Frontiers in Signal Processing*, vol. 2, 2022.

3 - Convolutional Neural Networks for Lossy Point Cloud Geometry Compression

3.1 . Introduction

In this chapter, we focus on the lossy compression of static point cloud geometry. In PCC, a precise reconstruction of geometric information is of paramount importance to enable high-quality rendering and interactive applications. For this reasons, lossless geometry coding has been investigated recently in MPEG, but even state-of-the-art techniques struggle to compress beyond about 2 bits per occupied voxels (bpov) [66]. This results in large storage and transmission costs for rich point clouds. Lossy compression proposed in the literature, on the other hand, are based on octrees which achieve variable-rate geometry compression by changing the octree depth. Unfortunately, lowering the depth reduces the number of points exponentially. As a result, octree based lossy compression tends to produce "blocky" results at the rendering stage with medium to low bitrates. In order to partially attenuate this issue, [104] proposes to use wavelet transforms and volumetric functions to compact the energy of the point cloud signal. However, since they still employ an octree representation, their method exhibits rapid geometry degradation at lower bitrates. While previous approaches use hand-crafted transforms, we propose here a data driven approach based on learned convolutional transforms which directly works on voxels.

Specifically, we present a method for learning analysis and synthesis transforms suitable for point cloud geometry compression. In addition, by interpreting the point cloud geometry as a binary signal defined over the voxel grid, we cast decoding as the problem of classifying whether a given voxel is occupied or not. We train our model on the ModelNet40 mesh dataset [188, 156], test its performance on the Microsoft Voxelized Upper Bodies (MVUB) dataset [116] and compare it with the MPEG anchor [120]. We find that our method outperforms the anchor on all sequences at all bitrates. Additionally, in contrast to octree-based methods, ours does not exhibit exponential diminution in the number of points when lowering the bitrate. We also show that our model generalizes well by using completely different datasets for training and testing.

Then, we propose a set of contributions to improve RD performance and accelerate model training. We then present an ablation study identifying key performance factors for deep learning-based point cloud geometry compression. In particular, we start from a baseline model [141] and we consider the following improvements:

- *Entropy modeling*: we consider an hyperprior model to improve entropy coding.

- *Deeper transforms* that compensate downsampling with progressively higher numbers of filters.

- *Changing the balancing weight in the focal loss*: similar to [141], we cast decoding as an unbalanced classification problem by optimizing a focal loss [114]. Hence, we study the RD performance impact of the focal loss α parameter.

- *Optimal thresholding for decoding*: in order to classify voxels as occupied or not, we propose an optimal thresholding approach that minimizes a given distortion metric (instead of a fixed threshold as in [141]).

- *Sequential training*: in order to reduce the computational complexity of training a network for each RD tradeoff, we propose a sequential training procedure. That is, we train a network corresponding to a given RD point by fine tuning the network trained from the previous RD point. This makes training times up to 8 times faster compared to training independently and improves RD performance.

- *Ablation study* An extensive ablation study evaluating the impact of each factor mentioned above on RD performance. The evaluated conditions are detailed in Table 3.1.

- *Octree partitioning* An efficient octree partitioning algorithm that is significantly faster compared to recursive octree partitioning.

3.2 . Related Work

Our research is related most closely to three research areas: static point cloud geometry compression, deep image compression and deep point cloud compression.

Static point cloud geometry compression methods are usually based on the octree structure [153]. Indeed, octrees provide an efficient way of partitioning the 3D space and representing point clouds. In particular, they are especially suitable for lossless coding in combination with octree entropy models [66]. However, lossy compression using octrees alone has poor performance as pruning octree levels decreases the number of points exponentially resulting in significant distortion. To alleviate this issue, many solutions have been proposed such as triangle [58] surface models, planar [59] surface models, graph-based enhancement layers [52] and volumetric functions [104]. The core idea is that by encoding approximations along a coarse octree, we can alleviate the shortcomings of the octree structure.

Different from previous work in this area, we study learned approximation models based on deep neural networks.

Deep image compression considers the use of deep neural networks for image compression. An end-to-end image compression solution with joint RD optimization along with a learned entropy model has been proposed in [24], which also replaces (non-differentiable) quantization with uniform noise at training time. As a follow-up of that work, a scale hyperprior model has been proposed in [25]. The scale hyperprior enables the modeling of spatial correlations in the latent space; for each element, it uses a Gaussian distribution whose standard deviation is predicted by a dedicated network. We design models for PCC using these learning-based entropy modeling techniques.

DPCC is a recent research avenue exploring the use of deep neural networks for PCC. For lossy geometry coding, voxel-based DPCC methods have been shown to outperform traditional methods significantly [141, 180, 78]. For lossless geometry coding, deep neural networks have been used to improve entropy modeling [87]. Also, DPCC for attributes has been explored by interpreting point clouds as a 2D discrete manifold in 3D space [140]. Closely related to our study, the behavior and performance of DPCC methods has been investigated in [78]. However, this particular study investigates the characteristics and RD impact of the latent space. In contrast, we seek to understand and identify key performance factors for rate-distortion (RD) performance on a larger scale.

3.3 . Basic Approach

3.3.1 . Proposed method

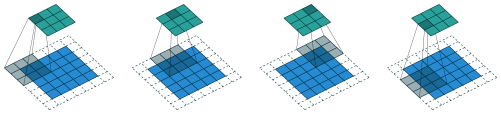
In this section, we describe the proposed method in more details.

Definitions

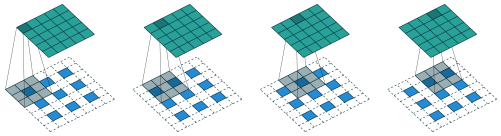
First, we define the set of possible points at resolution r as $\Omega_r = [0\,..\,r]^3$. Then, we define a point cloud as a set of points $S \subseteq \Omega_r$ and its corresponding voxel grid v_S as follows:

$$v_S \colon \Omega_r \longrightarrow \{0,1\},$$
$$z \longmapsto \begin{cases} 1, & \text{if } z \in S \\ 0, & \text{otherwise.} \end{cases}$$

For notational convenience, we use s^3 instead of $s \times s \times s$ for filter sizes and strides.



(a) Strided convolution on a 5^2 input with a 3^2 filter, 2^2 strides and same padding. The shape of the output is 3^3.



(b) Strided transpose convolution on a 3^2 input with a 3^2 filter, 2^2 strides and same padding. The shape of the output is 5^2.

Figure 3.1: Strided convolution and strided transpose convolution operations. Illustrations from [60].

Model

We use a 3D convolutional auto-encoder composed of an analysis transform f_a, followed by a uniform quantizer and a synthesis transform f_s.

Let $x = v_S$ be the original point cloud. The corresponding latent representation is $y = f_a(x)$. To quantize y, we introduce a quantization function Q so that $\hat{y} = Q(y)$. This allows us to express the decompressed point cloud as $\hat{x} = \hat{v}_S = f_s(\hat{y})$. Finally, we obtain the decompressed point cloud $\tilde{x} = \tilde{v}_S = round(\min(0, \max(1, \hat{x})))$ using element-wise minimum, maximum and rounding functions.

In our model, we use convolutions and transpose convolutions with same padding and strides. They are illustrated in Figure 3.1 and defined as follows :

- Same (half) padding pads the input with zeros so that the output size is equal to the input size.
- Convolution performed with unit stride means that the convolution filter is computed for each element of the input array. When iterating the input array, strides specify the step for each axis.
- Convolution can be seen as matrix multiplication and transpose convolution can be derived from this. In particular, we can build a sparse matrix C with non-zero elements corresponding to the weights. The transpose convolution, also called *deconvolution*, is obtained using the matrix C^T as a layout for the weights.

Using these convolutional operations as a basis, we learn analysis and synthesis transforms structured as in Figure 3.2 using the Adam optimizer [102] which is based on adaptive estimates of first and second moments of the gradient.

We handle quantization similarly to [25]. Q represents element-wise integer rounding during evaluation and Q adds uniform noise between -0.5 and 0.5 to

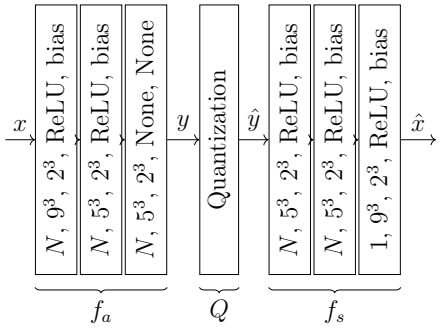


Figure 3.2: Neural Network Architecture. Layers are specified using the following format: number of feature maps, filter size, strides, activation and bias.

each element during training which allows for differentiability. To compress $Q(y)$, we perform range coding and use the Deflate algorithm, a combination of LZ77 and Huffman coding [90] with shape information on x and y added before compression. Note however that our method does not assume any specific entropy coding algorithm.

Our decoding process can also be interpreted as a binary classification problem where each point $z \in \Omega_r$ of the voxel grid is either present or not. This allows us to decompose $\hat{x} = \hat{v}_S$ into its individuals voxels z whose associated value is p_z. However, as point clouds are usually very sparse, most $v_S(z)$ values are equal to zero. To compensate for the imbalance between empty and occupied voxels we use the α-balanced focal loss as defined in [114]:

$$FL(p_z^t) = -\alpha_z(1 - p_z^t)^\gamma log(p_z^t) \tag{3.1}$$

with p_z^t defined as p_z if $v_S(z) = 1$ and $1 - p_z$ otherwise. Analogously, α_z is defined as α when $v_S(z) = 1$ and $1 - \alpha$ otherwise. The focal loss for the decompressed point cloud can then be computed as follows:

$$FL(\tilde{x}) = \sum_{z \in S} FL(p_z^t). \tag{3.2}$$

Our final loss is $L = \lambda D + R$ where D is the distortion calculated using the focal loss and R is the rate in number of bits per input occupied voxel (bpov). The rate is computed differently during training and during evaluation. On one hand, during evaluation, as the data is quantized, we compute the rate using the number of bits of the final compressed representation. On the other hand, during training, we add uniform noise in place of discretization to allow for differentiation. It follows that the probability distribution of the latent space $Q(y)$ during training is a continuous relaxation of the probability distribution of $Q(y)$ during evaluation

which is discrete. As a result, entropies computed during training are actually differential entropies, or continuous entropies, while entropies computed during evaluation are discrete entropies. During training, we use differential entropy as an approximation of discrete entropy. This makes the loss differentiable which is primordial for training neural networks.

3.3.2 . Experimental results

We use train, evaluation and test split across two datasets. We train and evaluate our network on the ModelNet40 aligned dataset [188, 156]. Then, we perform tests on the MVUB dataset and we compare our method with the MPEG anchor [120].

We perform our experiments using Python 3.6 and Tensorflow 0.12. We use $N = 32$ filters, a batch size of 64 and Adam with $lr = 10^{-4}$, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. For the focal loss, we use $\alpha = 0.9$ and $\gamma = 2.0$.

To compute distortion, we use the point-to-plane symmetric PSNR computed with the *pc_error* MPEG tool [171].

Dataset

The ModelNet40 dataset contains 12,311 mesh models from 40 categories. This dataset provides us with both variety and quantity to ensure good generalization when training our network. To convert this dataset to a point cloud dataset, we first perform sampling on the surface of each mesh. Then, we translate and scale it into a voxel grid of resolution r. We use this dataset for training with a resolution $r = 64$.

The MVUB dataset [116] contains 5 sequences captured at 30 fps during 7 to 10 seconds each with a total of 1202 frames. We test our method on each individual frame with a resolution $r = 512$. In other words, we evaluate performance for intra-frame compression on each sequence.

We compute RD curves for each sequence of the test dataset. For our method, we use the following λ values to compute RD points : 10^{-4}, 5×10^{-5}, 10^{-5}, 5×10^{-6} and 10^{-6}. For each sequence, we average distortions and bitrates over time for each λ to obtain RD points. For the MPEG anchor, we use the same process with different octree depths.

Evaluation

To evaluate distortion, we use the *point-to-plane symmetric PSNR* [171] $e_{symm}(A, B) = \min(e(A, B), e(B, A))$ where $e(A, B)$ provides the point-to-plane PSNR between points in A and their nearest neighbors in B. This choice is due to the fact that original and reconstructed point clouds may have a very different number of

points, e.g., in octree-based methods the compressed point cloud has significantly less points than the original, while in our method it is the opposite. In the rest of this section, we refer to the point-to-plane symmetric PSNR as simply PSNR.

Results

Our method outperforms the MPEG anchor on all sequences at all bitrates. The latter has a mean bitrate of 0.719 bpov and a mean PSNR of 16.68 dB while our method has a mean bitrate of 0.691 and a mean PSNR of 24.11 dB. RD curves and the Bjontegaard-delta bitrates (BDBR) for each sequence are reported in Figure 3.3. Our method achieves 51.5% BDBR savings on average compared to the anchor.

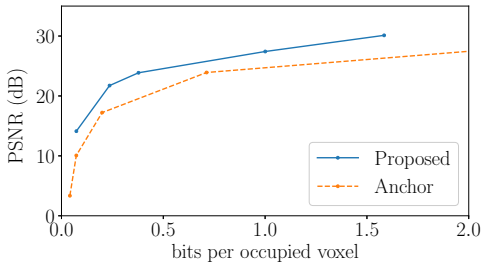
In Figure 3.4, we show examples on the first frame of the Phil sequence. Our method achieves lower distortion at similar bitrates and produces more points than the anchor which increases quality at low bitrates while avoiding "blocking" effects. This particular example shows that our method produces 218 times more points than the anchor at similar bitrates. In other words, both methods introduce different types of distortions. Indeed, the number of points produced by octree structures diminishes exponentially when reducing the octree depth. Conversely, our method produces more points at lower bitrates as the focal loss penalizes false negatives more heavily.

In this work, we use a fixed threshold of 0.5 during decompression. Changing this threshold can further optimize rate-distortion performance or optimize other aspects such as rendering performance (number of points).

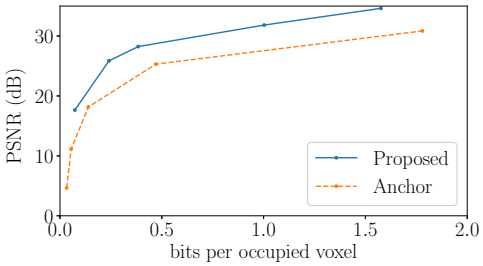
3.4 . Improved Deep Point Cloud Compression

3.4.1 . Proposed Improvements

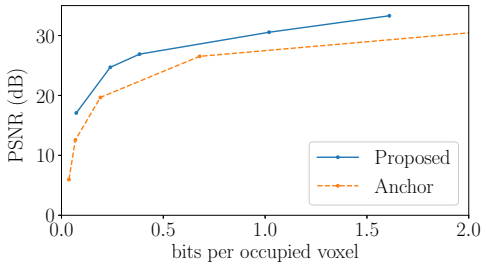
In this section we present different strategies to improve DPCC. We consider as baseline the network proposed in our preliminary work [141] (denoted as **c1** in the following). In that work, we relied on shallow transforms to compress entire point clouds at once. However, this has a fundamental limitation in terms of memory usage, as it does not allow to compress large point clouds as those commonly used in MPEG CTCs. Therefore, in this work we make use of octree partitioning to partition point clouds into blocks of size $64 \times 64 \times 64$ voxels, which we have found to be a good trade-off between memory usage and coding performance. We denote the different considered improvements with **c2**,..., **c6**, which are summarized in Table 3.1.



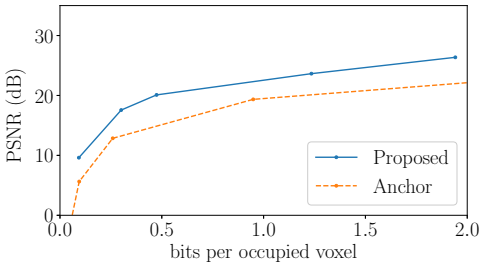
(a) Andrew sequence (−47.8% BDBR)



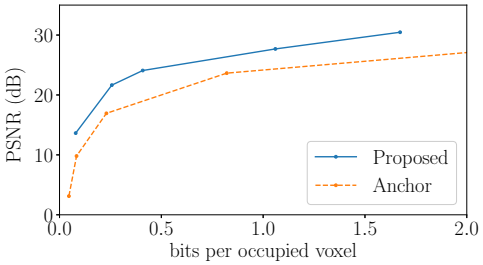
(b) David sequence (−55.7% BDBR)



(c) Phil sequence (−49.0% BDBR)



(d) Ricardo sequence (−52.7% BDBR)



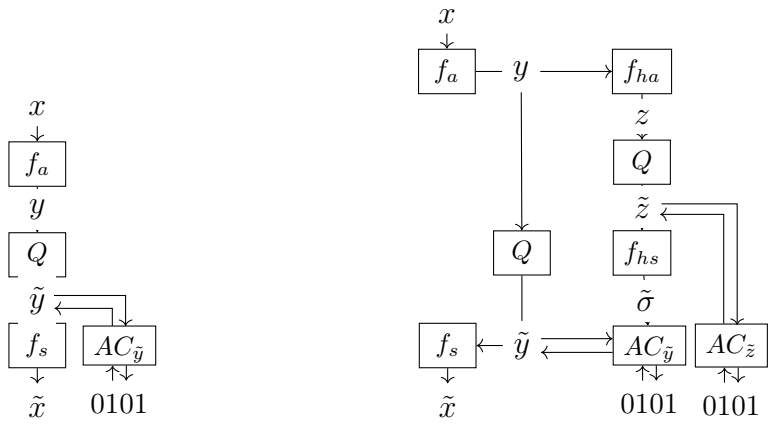
(e) Sarah sequence (−52.4% BDBR)

Figure 3.3: RD curves for each sequence of the MVUB dataset. We compare our method to the MPEG anchor.

Figure 3.4: Original point cloud (left), the compressed point cloud using the proposed method (middle) and the MPEG anchor (right). Colors are mapped using nearest neighbor matching. Our compressed point cloud was compressed using $\lambda = 10^{-6}$ with a PSNR of 29.22 dB and 0.071 bpov. The anchor compressed point cloud was compressed using a depth 6 octree with a PSNR of 23.98 dB and 0.058 bpov. They respectively have 370,798; 1,302,027; and 5,963 points.

Table 3.1: Experimental conditions evaluated for point cloud geometry compression. Each condition is an improvement over the previous one.

Name	Model	Transforms	α	Threshold	Training
c1	Baseline	Shallow	0.90	Fixed	Independent
c2	Hyperprior	—	—	—	—
c3	—	Deep	—	—	—
c4	—	—	0.75	—	—
c5	—	—	—	Optimal	—
c6	—	—	—	—	Sequential



(a) Baseline model. (b) Hyperprior model.

Figure 3.5: Entropy models considered for point cloud geometry compression. The f functions are learned transforms, Q refers to quantization and AC to arithmetic coding with its associated density model.

Entropy modeling (c2)

We consider models that take the voxelized point clouds x and $\tilde{x}$ as input and output. In particular, we consider a baseline model (Fig. 3.5a) and an hyperprior model (Fig. 3.5b).

The baseline model is based on an autoencoder architecture with an analysis f_a and a synthesis transform f_s [24]. y is modeled using a learned entropy model for each feature map. The baseline model is expressed as follows

$$y = f_a(x) \qquad \tilde{y} = Q(y) \qquad \tilde{x} = f_s(\tilde{y}). \tag{3.3}$$

We consider a scale hyperprior model [25] as a better entropy model for $\tilde{y}$. Specifically, we model y with a zero-mean gaussian density model $\mathcal{N}(0, \tilde{\sigma}^2)$ where standard deviations $\tilde{\sigma}^2$ are predicted from y with $\tilde{\sigma} = f_{hs}(Q(f_{ha}(y)))$. As a result, the spatial dependencies can be modeled better compared to the learned entropy model. The hyperprior model is expressed as follows

$$y = f_a(x) \qquad \tilde{y} = Q(y) \qquad \tilde{x} = f_s(\tilde{y}) \tag{3.4}$$

$$z = f_{ha}(y) \qquad \tilde{z} = Q(z) \qquad \tilde{\sigma} = f_{hs}(\tilde{z}) \tag{3.5}$$

where z is modeled with a learned density model for each feature map.

The compression model is trained using joint RD optimization with the loss function $R + \lambda D$. For each RD tradeoff, we train a model with the corresponding λ value resulting in transforms and entropy models specialized for this particular

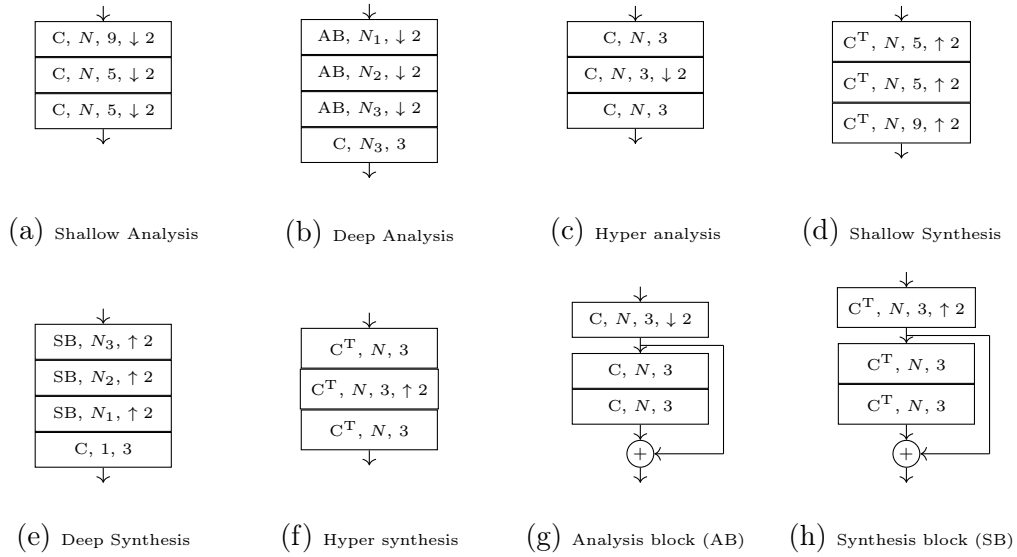


Figure 3.6: Transform types. Each layer is specified as follows: convolution type (C refers to convolution, C^T to transposed convolution), number of filters, filter size and strides.

tradeoff. The entropy R is computed on $\tilde{y}$, and $\tilde{z}$ for the hyperprior model, using their associated entropy models. Since the quantization operation Q is not differentiable, we use additive uniform noise during training in place of quantization as originally proposed in [24].

Deeper transforms (c3)

We compare shallow and deep transforms for analysis and synthesis, as illustrated in Fig. 3.6. Specifically, we focus on analysis and synthesis transforms and use shallow hyper-analysis and hyper-synthesis transforms (Fig. 3.6c and 3.6f).

The transforms based on 3D convolutions and 3D transpose convolutions introduced in [141] are referred to as shallow transforms (Fig. 3.6a and 3.6d). We introduce *deeper* variants of shallow transforms (Fig. 3.6b and 3.6e), which we refer to as deep transforms. These transforms are composed of residual [82] blocks (Fig. 3.6g and 3.6h) which use skip-connections to prevent issues such as exploding or vanishing gradients. The skip-connections act as "shortcuts" in the network allowing gradients to backpropagate through shorter paths. We also make them *progressive* by increasing the number of filters progressively as $N_1/4 = N_2/2 = N_3$. The rationale behind this choice is that the number of filters should compensate the downsampling along the spatial dimensions. In that way, the capacity at a given layer $W \times H \times D \times N$ decreases more slowly which allows the network to

compress information more easily. In our experiments, we set $N_3 = 64$.

Changing the balancing weight in the focal loss (c4)

When considering point clouds as voxel grids, we observe that most of the space is empty (usually $> 95\%$). This large class imbalance between occupied voxels and unoccupied voxels is a barrier to effective training. Indeed, without any counter-measures, the network would converge towards empty outputs only. In order to resolve this class imbalance issue, we adopt the focal loss [114] as our distortion loss.

The focal loss is well suited for point clouds since it addresses the class imbalance issue with α-balancing. Moreover, the focal loss differentiates between easy and hard examples using the γ parameter. Specifically, the higher γ is, the more hard examples are emphasized. With $\gamma = 0$, the focal loss becomes equivalent to the weighted binary cross-entropy.

For conciseness, we adopt the following notation. If $x = 1$, then $x_t = x$, $\alpha_t = \alpha$ and $\tilde{x}_t = \tilde{x}$; otherwise $x_t = 1 - x$, $\alpha_t = 1 - \alpha$ and $\tilde{x}_t = 1 - \tilde{x}$. We then define the focal loss as

$$\mathrm{FL}(x, \tilde{x}) = \alpha_t x_t (1 - \tilde{x}_t)^\gamma \log(\tilde{x}_t). \tag{3.6}$$

We study the impact of the focal loss α parameter on RD performance. The α parameter governs the attention given to occupied voxels and empty voxels. A high α value makes marking occupied voxels as empty more costly than marking empty voxels as occupied and results in denser reconstructions. Originally, we picked the same α value (0.90) as in [141]. This was motivated by the fact that point clouds are often comprised of more than 95% of empty space.

However, we found experimentally that lower α values can actually provide better coding gains. We hypothesize that this is due to the fact that the default $\gamma = 2$ in the focal loss emphasizes hard examples (occupied voxels) more than easy examples (empty voxels). Thus, $\gamma = 2$ already alleviates the class imbalance issue which explains this phenomenon.

Optimal thresholding for decoding (c5)

For each block, after decoding y (and z for the hyperprior model) into $\tilde{x}$, we need to convert $\tilde{x}$ into binary values in order to obtain the decompressed point cloud. The baseline method (**c1**) employs a fixed threshold $t = 0.5$. In contrast, we perform this conversion by finding optimal thresholds for each block of voxels. This threshold is transmitted as side information in the bitstream with a small overhead in terms of bitrate.

We formulate optimal thresholding as the problem of finding an optimal threshold $t^\star$ such that

$$t^\star = \arg\min_t d(x, H(\tilde{x} - t)) \tag{3.7}$$

where d is a distortion metric and $H(x)$ is the heaviside step function (equal to 1 when $x \geq 0$ and 0 otherwise).

Sequential training (c6)

We train compression models for each RD tradeoff using a corresponding λ value. This allows for transforms and entropy models to be specialized for this particular tradeoff resulting in better RD performance. Unfortunately, using this independent training scheme, we need to train one model for each tradeoff.

To alleviate this issue, we propose a novel sequential training scheme that speeds up training significantly and improves RD performance. The core idea of this scheme is to use previously trained neural network weights as a starting point for new neural networks. Essentially, given a set of λ tradeoffs, we first train λ_1. Then, for each subsequent model, we train λ_i using the trained weights of λ_{i-1}.

In this training scheme, we proceed to train the different tradeoffs in descending order. That is, we first train a low distortion, high bitrate model. Then, for each subsequent model, we progressively lower the bitrate while trying to minimize the increase in distortion.

3.4.2 . Experiments

We evaluate the six different improvement strategies described in subsection 3.4.1 and summarized in Table 3.1. The BD-PSNR gains are reported in Table 3.2.

Experimental setup

We perform our experiments on four point clouds specified in the MPEG CTCs [1, 56]. Namely, "longdress_vox10_1300", "loot_vox10_1200", "redandblack_vox10_1490", "soldier_vox10_0690" which we refer to as "longdress", "loot", "redandblack" and "soldier".

We train our models on a subset of the ModelNet40 dataset. First, we sample the dataset into voxelized point clouds with resolution 512 and select the 200 largest point clouds. Then, we divide these point clouds into blocks with resolution 64 and select the 4000 largest blocks. This produces a small dataset containing rich point clouds, accelerates dataset loading time and reduces memory footprint when training. We perform training with λ values ranging from 5×10^{-6} to 3×10^{-4}.

We evaluate the different conditions using G-PCC trisoup and octree as baselines. Specifically, we use G-PCC v10.0 (released in May 2020) with the included

Table 3.2: RD performance for each experimental condition. We specify BD-PSNR values (dB) compared to G-PCC trisoup and octree in each cell (trisoup BD-PSNR / octree BD-PSNR). The greatest values for trisoup and octree are indicated in **bold** and the second greatest in *italic*. **c6** consistently outperforms all other conditions.

		Experimental conditions					
Point cloud	Metric	**c6**	**c5**	**c4**	**c3**	**c2**	**c1**
loot	D1	**5.91** / **7.00**	*5.84* / *6.89*	4.05 / 5.06	2.03 / 3.67	−0.27 / 2.26	−0.72 / 1.88
	D2	*6.85* / **6.12**	**6.90** / *6.11*	4.10 / 3.33	1.44 / 1.23	−1.81 / −0.81	−2.60 / −1.40
redandblack	D1	**5.02** / **6.50**	*4.81* / *6.30*	3.28 / 4.71	1.43 / 3.45	−0.19 / 2.58	−0.59 / 2.01
	D2	**5.93** / **5.65**	*5.74* / *5.46*	2.91 / 2.62	0.55 / 0.79	−1.73 / −0.46	−2.42 / −1.18
longdress	D1	**5.54** / **6.94**	*5.41* / *6.79*	3.75 / 5.10	1.81 / 3.82	−0.26 / 2.64	−0.79 / 2.10
	D2	**6.59** / **6.01**	*6.52* / *5.91*	3.90 / 3.30	1.41 / 1.36	−1.37 / −0.34	−2.20 / −1.09
soldier	D1	**5.55** / **6.91**	*5.49* / *6.88*	3.76 / 5.11	1.88 / 3.88	−0.28 / 2.60	−0.77 / 2.13
	D2	**6.54** / **6.02**	*6.52* / **6.02**	3.86 / *3.36*	1.39 / 1.45	−1.54 / −0.40	−2.31 / −1.06
Average	D1	**5.50** / **6.84**	*5.39* / *6.71*	3.71 / 5.00	1.79 / 3.71	−0.25 / 2.52	−0.72 / 2.03
	D2	**6.48** / **5.95**	*6.42* / *5.87*	3.69 / 3.15	1.20 / 1.21	−1.61 / −0.50	−2.38 / −1.18

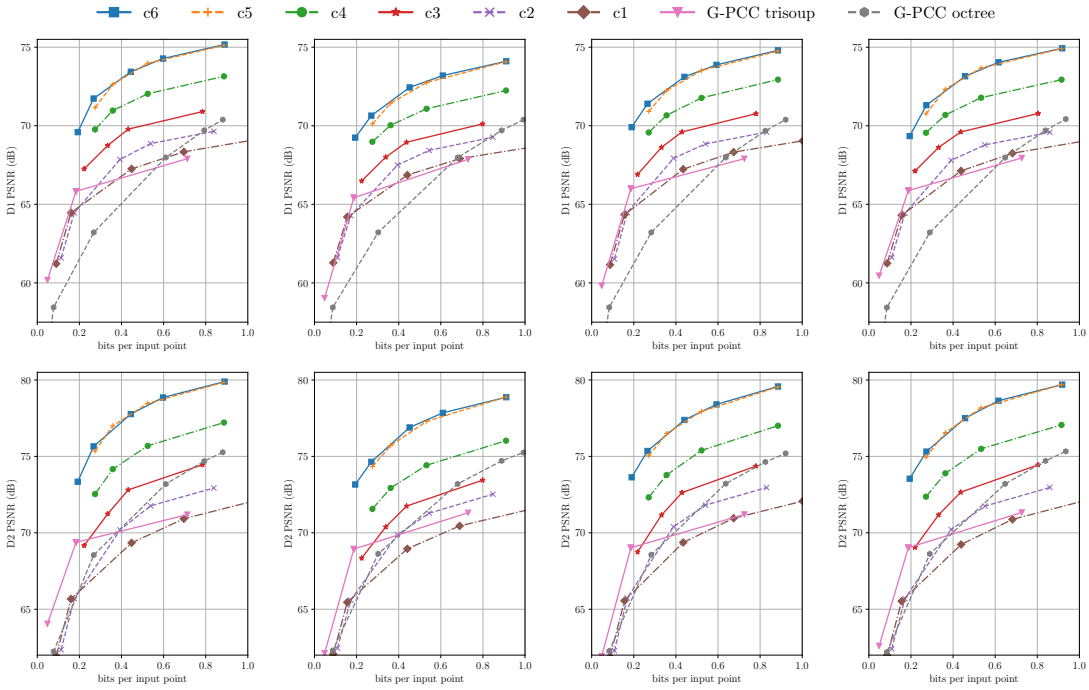
configurations, "mpeg-pcc-dmetric" v0.12.3 for D1 and D2 metrics, Python 3.6.9 and TensorFlow 1.15.0 with the Adam optimizer [102].

Experimental results

In Fig. 3.7 and Table 3.2, we observe that each condition is a net improvement over previous ones. **c6** outperforms G-PCC trisoup with an average BD-PSNR of 5.50 dB on D1 and 6.48 dB on D2 and outperforms G-PCC octree with an average BD-PSNR of 6.84 dB on D1 and 5.95 dB on D2. Note that the lowest bitrate point for **c6** is not included in BD-PSNR computations in order to keep integration intervals consistent and keep BD-PSNRs comparable across different conditions.

We also observe that **c5** (optimal thresholding) is especially beneficial for the point-to-plane metric (D2) with an improvement of 1.68 dB for D1 and 2.73 dB for D2 compared to **c4**. Indeed, optimal thresholding provides optimal sets of thresholds for D1 and D2 yielding two separate reconstructions.

In Fig. 3.8, we provide qualitative results on "soldier_vox10_0690". We observe that shapes and local point densities are reproduced more accurately compared to G-PCC trisoup. Overall, our method results in lower distortions at similar bitrates.



(a) "loot" (b) "redandblack" (c) "longdress" (d) "soldier"

Figure 3.7: RD curves for each condition in Table 3.2. **c6** consistently outperforms G-PCC trisoup and G-PCC octree.

(a) Original (b) **c6** (D1 69.59 dB, 0.194 bpp) (c) G-PCC Trisoup (D1 65.87 dB, 0.188 bpp)

Figure 3.8: Qualitative evaluation on "soldier_vox10_0690". For **c6** and G-PCC Trisoup, we show the decompressed point cloud and its D1 squared errors. The errors are displayed according to the color scale on the right and are truncated to the 99th percentile (3.0). In parentheses, we specify the D1 PSNR along with the number of bits per input point (bpp).

Ablation study

In this subsection, we present BD-PSNR values when compared to G-PCC trisoup. The hyperprior model (**c2**) results in an improvement of 0.47 dB for D1 and 0.77 dB for D2 compared to **c1**. Adding deep transforms (**c3**) further improves D1 by 2.04 dB and D2 by 2.81 dB compared to **c2**.

In Table 3.3, we observe that setting $\alpha = 0.75$ for D1 and $\alpha = 0.50$ for D2 increases RD performance significantly for all point clouds. The average BD-PSNR for $\alpha = 0.75$ is 3.71 dB for D1 and 3.69 dB for D2. Also, the average BD-PSNR for $\alpha = 0.50$ is 3.70 dB for D1 and 6.07 dB for D2. Indeed, higher α values lead to denser reconstructions which are favored by D1 and lower α values to sparser ones which are favored by D2. We select $\alpha = 0.75$ (**c4**) as we have found experimentally that it performs better when associated with optimal thresholding. Compared to $\alpha = 0.90$ (**c3**), $\alpha = 0.75$ brings an improvement of 1.92 dB for D1 and 2.49 dB for D2 .

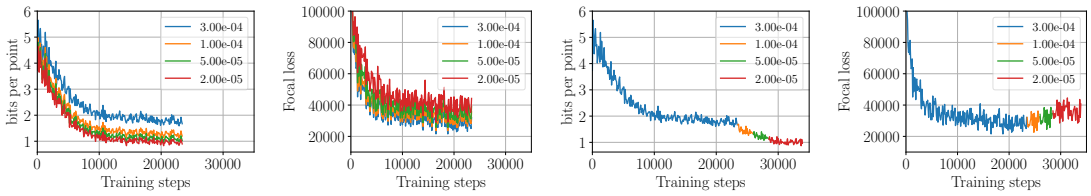
Then, we use optimal thresholding (**c5**) with the point-to-point (D1) and point-to-plane (D2) objective metrics. As a result, we obtain two point clouds respectively optimized with D1 and D2. Also, we encode thresholds on 8 bits with 256 uniformly distributed threshold values between 0 and 1. Optimal thresholding

Table 3.3: Impact of the focal loss α parameter on RD performance. We specify BD-PSNR values (dB) compared to G-PCC trisoup for different α values. The greatest values are indicated in **bold** and the second greatest in *italic*. $\alpha = 0.75$ outperforms all other α values.

		α			
Point cloud	Metric	0.90	0.75	0.50	0.25
loot	D1	2.03	*4.05*	**4.41**	1.24
	D2	1.44	4.10	**6.47**	*4.19*
redandblack	D1	1.43	**3.28**	*2.24*	−3.70
	D2	0.55	*2.91*	**5.19**	1.63
longdress	D1	1.81	**3.75**	*3.66*	−0.11
	D2	1.41	*3.90*	**6.28**	3.88
soldier	D1	1.88	*3.76*	**4.48**	1.68
	D2	1.39	3.86	**6.32**	*4.45*
Average	D1	1.79	**3.71**	*3.70*	−0.22
	D2	1.20	*3.69*	**6.07**	3.54

(**c5**) results in an improvement of 1.68 dB for D1 and 2.73 dB for D2 compared to **c4**.

Training DPCC models is time consuming as shown in Fig. 3.9. Indeed, the **c5** condition requires 4 hours of training resulting in a total of 16 hours for four models on an Nvidia GeForce GTX 1080 Ti. With sequential training (**c6**), these models train in 30 to 60 minutes instead of 4 hours which is up to 8 times faster. In addition, this results in an improvement of 0.11 dB for D1 and 0.06 dB for D2 compared to **c5**.



(a) Independent training.

(b) Sequential training.

Figure 3.9: Bits per point and focal loss when training independently and sequentially. Sequential training is more efficient as it reuses previously trained models to train subsequent ones.

Sparser point clouds

In Guarda et al. [76] (Figure 10), the authors evaluate the performance of different deep learning approaches on sparser point clouds. The method proposed in this chapter performs competitively compared to the proposed adaptive approach in [76] on sparser point clouds. This shows that adaptive thresholding successfully adapts the network output to different input densities.

3.5 . Conclusion

First, we present a novel data-driven point cloud geometry compression method using learned convolutional transforms and a uniform quantizer. Our method outperforms the MPEG Anchor on the MVUB dataset in terms of rate-distortion with 51.5% BDBR savings on average. Additionally, in constrast to octree-based methods, our model does not exhibit exponential diminution in the number of output points at lower bitrates. This work can be extended to the compression of attributes and dynamic point clouds.

Then, we propose a set of key performance factors for such methods and we present an extensive ablation study on the individual impact of these factors. More precisely, we provide insights on the individual impact of scale hyperprior models, deep transforms, the focal loss α value, optimal thresholding and sequential training. We analyze each of these factors in order to provide a better understanding about why they improve RD performance. The final model (**c6**) outperforms G-PCC trisoup with an average BD-PSNR of 5.50 dB on D1 and 6.48 dB on D2 and outperforms G-PCC octree with an average BD-PSNR of 6.84 dB on D1 and 5.95 dB on D2.

Code is available at `https://github.com/mauriceqch/pcc_geo_cnn` and `https://github.com/mauriceqch/pcc_geo_cnn_v2`.

The contents of this chapter have been published in M. Quach, G. Valenzise, and F. Dufaux, "Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 4320–4324 and M. Quach, G. Valenzise, and F. Dufaux, "Improved Deep Point Cloud Geometry Compression," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2020, pp. 1–6.

4 - Deep Generative Model for Lossless Point Cloud Geometry Compression

4.1 . Introduction

In the previous chapter, decoding in the context of lossy point cloud geometry compression was cast as classification. Indeed, the grid of probabilities resulting from decoding are subject to binary classification via thresholding. In this chapter, we instead focus on *lossless coding* of point cloud geometry which can be achieved by considering these probabilities as input to an entropy coder. In particular, we consider the case of *voxelized* point clouds. Voxelization consists in pre-quantizing the geometric coordinates of the point cloud prior to coding in order to represent the geometry with integer precision. This operation is common in many coding scenarios, e.g., when dealing with dense point clouds such as those produced by camera arrays. After voxelization, the point cloud geometry can be represented either directly in the voxel domain or using an octree spatial decomposition. PCs are divided into a fixed number of cubes, which defines the resolution (e.g., 10 bit = 1024 cubes per dimension). Each cube is called a voxel. If a voxel contains at least one point, it is called an occupied voxel. Usually, very few voxels are occupied and a large part of the volume is empty. An octree representation can be obtained by recursively splitting the volume into eight sub-cubes until the desired precision is achieved. Then, occupied blocks are marked by bit 1 and empty blocks are marked by bit 0. Consequently, at each level, the generated 8 bits represent the occupancy state of an octree node (octant). Our method operates in both the voxel and octree domain. On the one hand, the octree representation can naturally adapt to the sparsity of the point cloud, as empty octants do not need to be further split; on the other hand, in the voxel domain convolutions can be naturally expressed, and geometric information (i.e., planes, surfaces, etc.) can be explicitly processed by a neural network.

In this work, we propose a deep-learning-based method (named VoxelDNN) for lossless compression of static voxelized point cloud geometry. Our main contributions are:

- We employ for the first time a deep generative model in the voxel domain to estimate the occupancy probabilities sequentially using a masked 3D convolutional network. The conditional distribution is then used to model the context of a context-based arithmetic coder.
- We propose an optimal rate-driven partitioning and context selection algorithm. The partitioning algorithm adapts to the point cloud sparsity by

employing a hybrid octree/voxel representation while the context to encode each block is expanded to the neighboring blocks and the expansion size is optimally selected.

- We propose specific data augmentation techniques for 3D point clouds coding, to increase its generalization capability.

We demonstrate experimentally that the proposed solution outperforms the state-of-the-art MPEG G-PCC lossless codec in terms of bits per occupied voxel over a set of point clouds with varying density and content type. The rest of the chapter is structured as follows: Section 4.2 reviews the related work; the proposed method is described in Section 4.3; Section 4.4 presents the experimental results; and finally Section 4.5 concludes the chapter.

4.2 . Related work

Relevant work related to this chapter includes state-of-the-art PC geometry coding and learning-based methods in image and point cloud compression.

4.2.1 . MPEG G-PCC and Conventional Lossless Codecs

Most existing methods that compress point cloud geometry, including MPEG G-PCC, use octree coding [153, 120, 96, 65, 66, 67, 87, 29] and local approximations called “triangle soups” (trisoup) [153, 58]

In the G-PCC geometry coder, points are first transformed and voxelized into an axis-aligned bounding box before geometry analysis using trisoup or octree scheme. In the trisoup coder, geometry can be represented by a pruned octree plus a surface model. This model approximates the surface in each leaf of the pruned octree using 1 to 10 triangles. In contrast, the octree coder partitions voxelized blocks until sub-cubes of dimension one are reached. First, the coordinates of isolated points are independently encoded to avoid "polluting" the octree coding (Direct Coding Mode - DCM) [7]. To encode the occupancy pattern of each octree node, G-PCC introduces many methods to exploit local geometry information and obtain an accurate context for arithmetic coding, such as Neighbour-Dependent Entropy Context [10], intra prediction [8], planar/angular coding mode [11, 6], etc. The lossless geometry coding mode of G-PCC is based on octree coding only.

In order to deal with the irregular point space, many octree-based lossless PCC methods have been proposed. In [153], the authors proposed an octree-based method which aims at reducing entropy by employing prediction techniques based on local surface approximations to predict occupancy patterns. Recently, more context modeling based approaches are proposed [65, 66, 67]. For example, the intra-frame compression method P(PNI) proposed in [67] builds a reference octree by propagating the parent octet to all children nodes, thus providing 255 contexts

to encode the current octant. Octree coding allows for a progressive representation of point clouds since each level of the octree is a downsampled version of the point cloud. However, a drawback of octree representation is that, at the first levels of the tree, it produces "blocky" scenes, and geometry information of point clouds (i.e., curve, plane) is lost. The authors of [29] proposed a binary tree based method which analyzes the point cloud geometry using binary tree structure and realizes an intra prediction via the extended Travelling Salesman Problem (TSP) within each leaf node. Instead, in this chapter, we employ a hybrid octree/voxel representation to better exploit the geometry information. Besides, the methods in [65, 66, 67] produce frequency tables which are collected from the coarser level or the previous frame and must be transmitted to the decoder. Our method predicts voxel distributions in a sequential manner at the decoder side, thus avoiding the extra cost of transmitting large frequency tables.

4.2.2 . Generative Models and Learning-based Compression

Estimating the data distribution from a training dataset is the main objective of generative models, and is a central problem in unsupervised learning. It has a number of applications, from image generation [169, 73, 136, 150], to image compression [185, 24, 121] and denoising [38]. Among the several types of generative models proposed in the literature [68], auto-regressive models such as PixelCNN [136, 150] are particularly relevant for our purpose as they allow to compute the exact likelihood of the data and to generate realistic images, although with a high computational cost. Specifically, PixelCNN factorizes the likelihood of a picture by modeling the conditional distribution of a given pixel's color given all previously generated pixels. These conditional distributions only depend on the possible pixel values with respect to the scanned context, which imposes a *causality* constraint. PixelCNN models the distribution using a neural network and the causality constraint is enforced using masked filters in each convolutional layer. Recently, this approach has also been employed in image compression to yield accurate and learnable entropy models [121]. This chapter explores the potential of this approach for point cloud geometry compression by adopting and extending conditional image modeling and masking filters into the 3D voxel domain.

Inspired by the success in learning-based image compression, deep learning has been widely adopted in point cloud coding both in the octree domain [87, 29], voxel domain [79, 81, 141, 142, 180, 81] and point domain [190, 88, 182]. Recently, the authors of [87] proposed an octree-based entropy model that models the probability distributions of the octree symbols based on the contextual information from octree structure. This method only targets static LiDAR point cloud compression. The extension version for intensity-valued LiDAR streaming data using spatio-temporal relations is proposed in [29]. However, these methods target dynamically acquired point clouds, while in this chapter we mainly focus on dense static point clouds.

Working in the voxel domain enables to easily extend most 2D tools, such as convolutions, to the 3D space. Many recent 3D convolution based autoencoder approaches for lossy coding [141, 142, 180, 81] compress 3D voxelized blocks into latent representations and cast the reconstruction as a binary classification problem. The authors of [190] proposed a pointnet-based auto-encoder method which directly takes points as input rather than voxelized point cloud. To handle sparse point clouds, recent methods leverage advances in sparse convolution [47, 70] to allow point-based approaches [88, 182]. For example, the proposed lossy compression method in [182] progressively downscale the point cloud into multiple scales using sparse convolutional transforms. Then, at the bottleneck, the geometry of scaled point cloud is encoded using an octree codec and the attributes are compressed using a learning-based context model. In contrast, in this chapter, we focus on dense voxelized point clouds and losslessly encode each voxel using the learned distribution from its 3D context. In addition, we apply this approach in a block-based fashion, which has been successfully employed in traditional image and video coding.

4.3 . Proposed method

4.3.1 . System overview

In this work, we propose a learning-based method for lossless compression of point cloud geometry. We aim at minimizing the encoded rate measured by the number of bits per occupied voxel (bpov) by exploiting the spatial redundancies within point cloud. The general scheme of our method is shown in Figure 4.1. A point cloud voxelized over a $2^n \times 2^n \times 2^n$ grid is known as an n-bit depth PC, which can be represented by an n level octree. In this work, we represent point cloud geometry in a hybrid manner, by combining the octree and voxel domains. We coarsely partition an n-depth point cloud up to level $n-6$. This allows to coarsely remove most of the empty space in the point cloud. As a result, we obtain a $n-6$ level octree and a number of non-empty binary blocks v of size $2^6 \times 2^6 \times 2^6$ voxels, which we refer to as resolution $d = 64$ or simply block 64 (Figure 4.1(a)). Blocks 64 can be further partitioned at resolution $d = \{64, 32, 16, 8, 4\}$ corresponding to maximum partitioning level $maxLv = \{1, 2, 3, 4, 5\}$ as detailed in Section 4.3.3. Figure 4.1(b) shows the multi-resolution encoder with $maxLv = 2$. A block of size d can be encoded as a single block (b2) or partitioned into 8 sub-cubes (b1). We then encode each non-empty block (blocks in blue in the figure) using the proposed method in the voxel domain (Section 4.3.2) and select the partitioning mode resulting in the smallest bpov. The overview of a single block encoder is shown in Figure 4.1(c). Our context model predicts the distribution of each voxel given all encoded voxels and pass it to an arithmetic coder to generate the final bit-

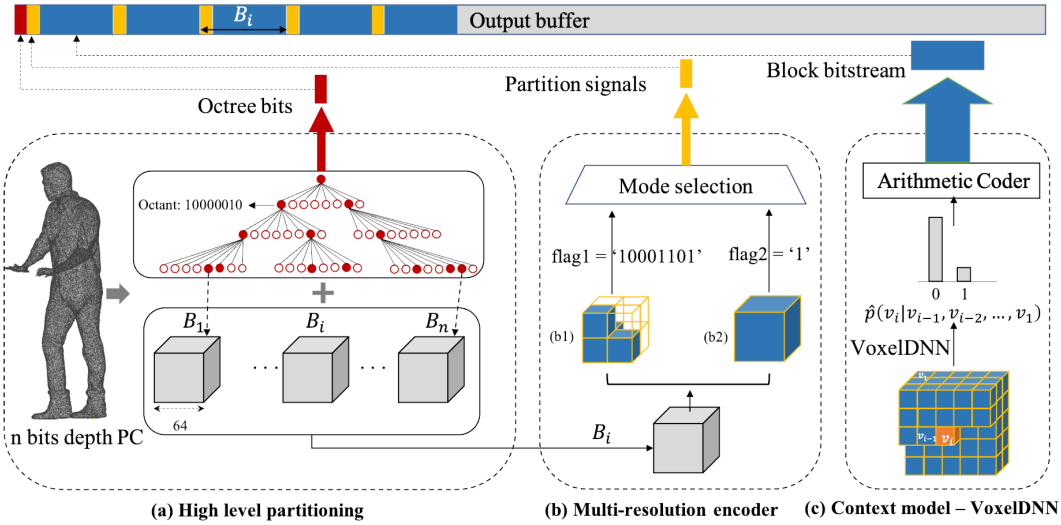


Figure 4.1: Overview of the proposed method. (a): a n bit depth point cloud is partitioned down to the $n-6$ octree level, yielding occupied blocks of size $64 \times 64 \times 64$. (b): We encode each block of 64^3 voxels as a single block (b1), or divide it into 8 children blocks (b2), depending on the total number of bits of each solution (partitioning level = 2). This procedure is repeated recursively for increasing partitioning levels up to 5. (c): For each occupied block of size d, the context model estimates the distribution of each voxel given the previously encoded voxels.

stream. The context is chosen adaptively following a rate optimization algorithm (Section 4.3.3). The high-level octree, partitioning signal, selected context as well as the depth of each block are converted to bytes and signaled to the decoder as side information. For ease of notation, we index all voxels in block v at resolution d from 1 to d^3 in raster scan order with:

$$v_i = \begin{cases} 1, & \text{if } i^{th} \text{ voxel is occupied} \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

4.3.2 . VoxelDNN

Our method losslessly encodes the voxelized point cloud using context-adaptive binary arithmetic coding. Specifically, we focus on estimating accurately a probability model $p(v)$ for the occupancy of a block v composed by $d \times d \times d$ voxels. We factorize the joint distribution $p(v)$ as a product of conditional distributions $p(v_i|v_{i-1}, \ldots, v_1)$ over the voxel volume:

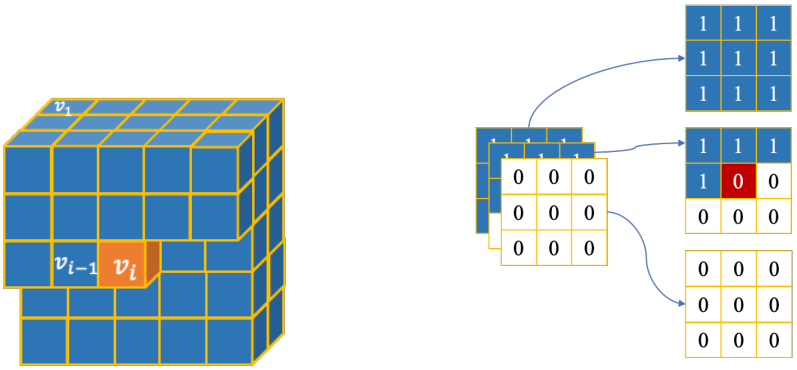
$$p(v) = \prod_{i=1}^{d^3} p(v_i|v_{i-1}, v_{i-2}, \ldots, v_1). \tag{4.2}$$

Each term $p(v_i|v_{i-1}, \ldots, v_1)$ above is the probability of the voxel v_i being occupied given the occupancy of all previous voxels, referred to as a context. Figure 4.2(a) illustrates such a 3D context. We estimate $p(v_i|v_{i-1}, \ldots, v_1)$ using a neural network which we dub **VoxelDNN**.

The conditional distributions in (4.2) depend on previously decoded voxels. This requires a *causality* constraint on the VoxelDNN network. To enforce causality, we extend to 3D the idea of masked convolutional filters, initially proposed in PixelCNN [136]. Specifically, two kinds of masks (A or B) are employed. Type A mask is filled by zeros from the center position to the last position in raster scan order as shown in Figure 4.2(b). Type B mask differs from type A in that the value in the center location is 1 (colored in red). Type A masks are used in the first convolutional filter to remove the connections between all future voxels and the voxel currently being predicted. From the second layer, the value of the current voxel is not used in its spatial position and is replaced by the result of the convolution over previous voxels. As a result, from the second convolutional layer, type B masks are applied which relaxes the restrictions of mask A by allowing the connection from the current spatial location to itself.

In order to learn good estimates $\hat{p}(v_i|v_{i-1}, \ldots, v_1)$ of the underlying voxel occupancy distribution $p(v_i|v_{i-1}, \ldots, v_1)$, and thus minimize the coding bitrate, we train VoxelDNN using cross-entropy loss. That is, for a block v of resolution d, we minimize :

$$H(p, \hat{p}) = \mathbb{E}_{v \sim p(v)} \left[\sum_{i=1}^{d^3} - \log \hat{p}(v_i) \right]. \tag{4.3}$$



(a) 3D voxel context (b) 3D type A mask

Figure 4.2: (a): Example 3D context in a $5 \times 5 \times 5$ block. Previously scanned elements are in blue. (b): $3 \times 3 \times 3$ 3D type A mask. Type B mask is obtained by changing center position (marked red) to 1.

It is well known that cross entropy represents the extra bitrate cost to be paid when the approximate distribution $\hat{p}$ is used instead of the true p. More precisely, $H(p, \hat{p}) = H(p) + D_{KL}(p\|\hat{p})$, where D_{KL} denotes the Kullback-Leibler divergence and $H(p)$ is Shannon entropy. Hence, by minimizing (4.3), we indirectly minimize the distance between the estimated conditional distributions and the real data distribution, yielding accurate contexts for arithmetic coding. Note that this is different from what is typically done in learning-based *lossy* PC geometry compression, where the focal loss is used [141, 142]. In this lossy context, the motivation behind using focal loss is to cope with the high spatial unbalance between occupied and non-occupied voxels. The reconstructed PC is then obtained by hard thresholding $\hat{p}(v)$, and the target is thus the final classification accuracy. Conversely, here we aim at estimating accurate soft probabilities to be fed into an arithmetic coder.

Figure 4.3 shows our VoxelDNN network architecture for a block of dimension d. Given the $d \times d \times d$ input block, VoxelDNN outputs the predicted occupancy probabilities of all input voxels. Our first 3D convolutional layer uses $7 \times 7 \times 7$ kernels with a type A mask. Type B masks are used in the subsequent layers. To avoid vanishing gradients and speed up the convergence, we implement two residual blocks [82] with $5 \times 5 \times 5$ kernels. Since type A masks are applied at the first layer, identity skip connection of residual block does not violate the causality constraint. Throughout VoxelDNN, the ReLu activation function is applied after each convolutional layer, except in the last layer where we use softmax activation. Using more filters generally increases the performance of VoxelDNN, at the expense of an increase in the number of parameters and computational complexity. After experimenting with various number of filters, we concluded that for input voxel

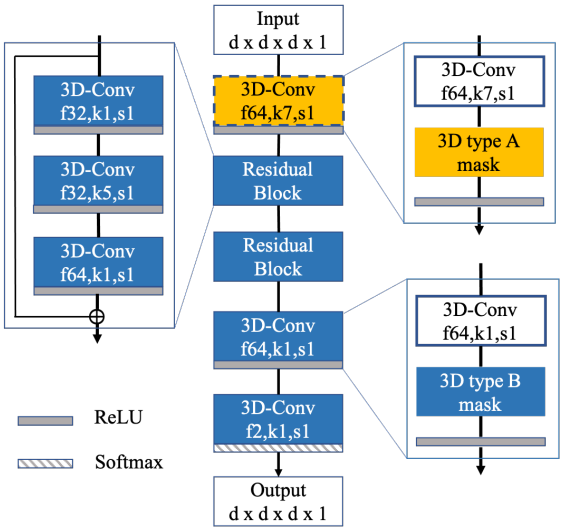


Figure 4.3: VoxelDNN architecture, d is the dimension of the input block, masked layers are colored in yellow and blue. A type A mask is applied to the first layer (dashed borders) and type B masks afterwards. 'f64,k7,s1' stands for 64 filters, kernel size 7 and stride 1.

block ($d \times d \times d \times 1$) which only has a single feature, 64 convolutional filters give a good trade-off between complexity and model performance.

4.3.3 . Multi-resolution encoder and adaptive partitioning

We use an arithmetic coder to encode the voxels sequentially from the first voxel to the last voxel of each block in a generative manner. Specifically, every time a voxel is encoded, it is fed back into VoxelDNN to predict the probability of the next voxel. Then, we pass the probability to the arithmetic coder to encode the next symbol.

However, applying this coding process at a fixed resolution d (in particular, on larger blocks) can be inefficient when blocks are sparse, i.e., they contain only a few occupied voxels. This is due to the fact that in this case, there is little or no information available in the receptive fields of the convolutional filters. To overcome this problem, we propose to optimize the block size based on a rate-optimized multi-resolution splitting algorithm as follows. We partition a block into 8 sub-blocks recursively and signal the occupancy of sub-blocks as well as the partitioning decision (0: empty, 1: encode as a single block, 2: further partition). The partitioning decision depends on the bit rate after arithmetic coding. If the total bitstream of partitioning flags and occupied sub-blocks is larger than encoding the parent block as a single block, we do not perform partitioning. The details of this process are shown in Algorithm 1. The maximum partitioning level or

the maximum number of block sizes is controlled by $maxLv$ and partitioning is performed up to $maxLv = 5$ corresponding to a smallest block size of 4. Depending on the output bits of each partitioning solution, a block of size 64 can contain a combination of blocks with different sizes. Figure 4.4 shows 4 partitioning examples for an encoder with $maxLv = 4$. Note that VoxelDNN learns to predict the distribution of the current voxel based on previously encoded voxels. As a result, we can use a bigger model size to predict the probabilities for smaller input block size.

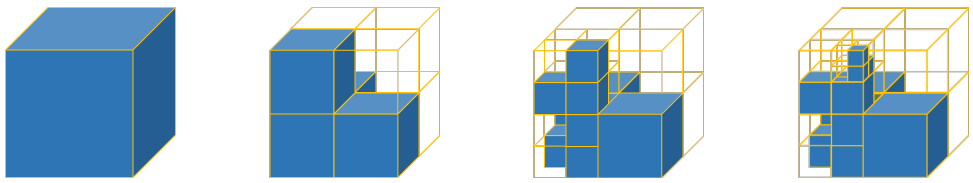
Algorithm 1: Block partitioning selection

Input: block: B, current level: $curLv$, max level: $maxLv$
Output: partitioning flags: fl, output bitstream: $bits$

Function `partitioner(`$B, curLv, maxLv$`):`
  $fl2 \leftarrow 2$; `// encode as 8 child blocks`
  for *block b in child blocks of* B **do**
    if *b is empty* **then**
      $child\_flag \leftarrow 0$;
      $child\_bit \leftarrow empty$;
    else
      if $curLv == maxLv$ **then**
        $child\_flag \leftarrow 1$;
        $child\_bit \leftarrow$ singleBlockCoder(b);
      else
        $child\_flag, child\_bit \leftarrow$ partitioner(b, $curLv + 1$,$maxLv$);
      end
    end
    $fl2 \leftarrow [fl2, child\_flag]$;
    $bit2 \leftarrow [bit2, child\_bit]$;
  end
  $total\_bit2 = sizeOf(bit2) + len(fl2) \times 2$;
  $fl1 \leftarrow 1$; `// encode as a single block`
  $bit1 \leftarrow$ singleBlockCoder(B);
  $total\_bit1 = sizeOf(bit1) + len(fl1) \times 2$;
  `/* partitioning selection */`
  if $total\_bit2 \geq total\_bit1$ **then**
    return $fl1, bit1$;
  else
    return $fl2, bit2$;
  end

4.3.4 . Context extension

We have discussed our multi-resolution encoder with multiple block sizes to adapt to the point cloud structure. However, with smaller block sizes, an implicit context model (using the content of the block) will be less efficient because the



(a) (b) (c) (d)

Figure 4.4: Partitioning a block of size 64 into: (a) a single block of size 64, (b): blocks of size 32, (c): 32 and 16, (d): 32, 16 and 8. Non-empty blocks are indicated by blue cubes.

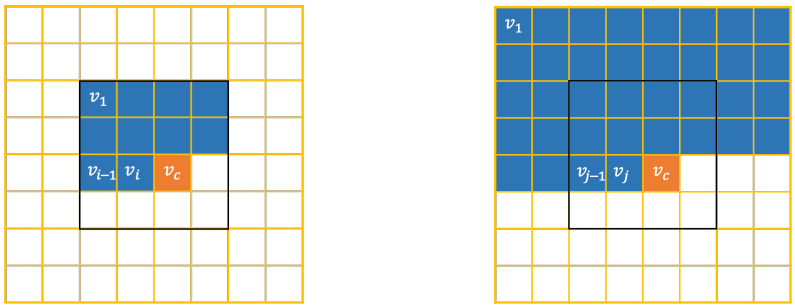
context may be too small. Therefore, we extend the context of each block to the encoded voxels that are above and on the left of the current voxel (causality constraint). Figure 4.5 illustrates the context before and after extension. Before extending the context, to encode voxel v_c, only voxels from v_1 to v_{i-1} in Figure 4.5(a) are considered as contexts. After extending the context to the bigger block, the context is now composed of all voxels in the blue area in Figure 4.5(b). The white area represent inactive voxels, i.e., not used in Eq. (4.2). Extending the context does not change the partitioning algorithm discussed above, although it might change the optimal selected partitions. Also, the causality is still enforced as long as we use masked filters in our network.

However, extending to a larger context is not always efficient when the extension area is sparse or contains noise, therefore we employ a rate-optimized block extension decision. To limit the computational complexity, we only allow certain combinations of block sizes and extension sizes, as shown in Table 4.1. To encode a block with context extension, in Algorithm 1, we encode a block with all the possible extension sizes and select the best one in terms of bpov. In total, we build 5 models for 5 input sizes which are $\{128, 64, 32, 16, 8\}$ in the context extension mode.

4.3.5 . Data augmentation

Table 4.1: Extending block size

Block size	**Extending block size**
64	128,64
32	64,32
16	64,32,16
8	64,32,16,8



(a) (b)

Figure 4.5: 2D illustration of context extension from block 4×4 to block 8×8. (a): Before extension, (b): after extension. Blue squares are active voxels in the context, voxels in the white area are ignored by masks or from the bigger block.

In order to train more robust probability estimation models and to increase the generalization capabilities of our model, we employ data augmentation techniques specifically suited for PCC. In particular, we observed that methods based on convolutional neural networks are especially sensitive to changes in PC density and acquisition noise. Therefore, in addition to typical rotation and shifting data augmentation used for other PC analysis tasks [23, 182], we also consider here alternative techniques, such as downsampling. Note that even though our VoxelDNN operates on voxel domain, to reduce the complexity, all input pipelines process point clouds in the form of x, y, z coordinates before converting into dense block in the final step. Specifically, for each generated block from the training datasets, we rotate them by an angle θ around each x, y, z axis. In addition, to adapt to varying density levels of the test point clouds, we randomly remove points from the original block as well as rotated blocks with the sampling rate f_s ($f_s \in [0, 1]$) over the total points. Figure 4.6 shows our data augmentation methods applying on Longdress point cloud from MPEG.

4.4 . Experimental Results

4.4.1 . Experimental Setup

Training dataset

We consider point clouds from different and varied datasets, including ModelNet40 [188] which contains 12,311 models from 40 categories and three smaller datasets: MVUB [116], MPEG CAT1 [1] and 8i [56]. We uniformly sample points from the mesh models from ModelNet40 and then scale them to voxelized point clouds with

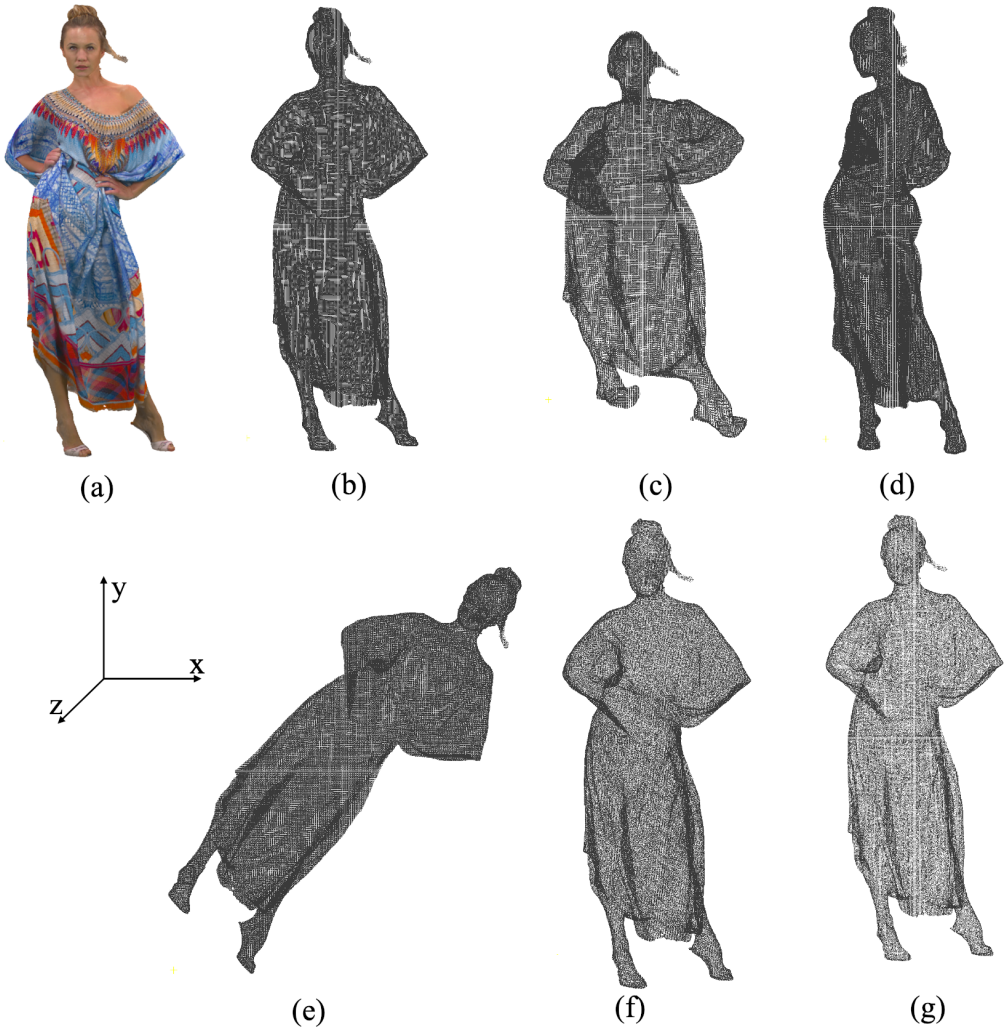


Figure 4.6: Example of data augmentation applied on the Longdress point cloud. (a) Original; (b) After removing color attributes; (c),(d),(e) Rotation with $\theta = 45°$ on x, y and z axis; (f),(g) Sampling rate $f_s = 0.7$ and $f_s = 0.4$

Table 4.2: Training and Testing Point Clouds

Training Set			Test Set	
Point Cloud	# Fr	ρ	Point Cloud	ρ
MVUB, 10 bits			*MVUB, 10 bits, dynamic upper body*	
Andrews	6	1.70	Phil	1.64
David	5	1.65	Ricardo	1.77
Sarah	6	1.72		
8i, 10 bits			*8i, 10 bits, dynamic full body*	
Soldier	9	1.51	Redandblack	1.49
Longdress	9	1.52	Loot	1.43
			Thaidancer	1.68
CAT1, 10 bits			Boxer	1.56
Facade	1	1.20	*CAT1, 10 bits, static cultural heritage*	
Egyptian mask	1	0.12	Frog	1.13
Statue klimt	1	0.89	Arco Valentino	0.45
Head	1	1.43	Shiva	0.88
House w/o roof	1	1.21		
ModelNet40, 9 bits			*USP, 10 bits, static cultural heritage*	
200 largest PCs	200	1.53	BumbaMeuBoi	0.18
			RomanOilLight	0.94

9 bit precision. To enforce the fairness between the smaller datasets in which we select point clouds for testing, point clouds from MPEG CAT1 are sampled to 10 bit precision as in MVUB and 8i. In addition, we measure the *local density* ρ of a point cloud, computed as the average portion of occupied voxels in the blocks of size 64, that is:

$$\rho = \frac{1}{N_{\mathcal{B}}} \times \sum_{\mathcal{B}_i \in \mathcal{B}} \frac{100 \times \text{number of points in } \mathcal{B}_i}{64^3} \quad (\%) \tag{4.4}$$

where $\mathcal{B}$ is the set of occupied blocks of size 64, and $N_{\mathcal{B}}$ is the cardinality of $\mathcal{B}$. The higher the value of ρ is, the denser the point cloud. The selected point clouds, number of frames as well as ρ of the training data are shown in Table 4.2.

To train a VoxelDNN model of size d we divide all selected PCs into occupied blocks of size $d \times d \times d$. Table 4.3 reports the number of blocks from each dataset for training, with the majority coming from the ModelNet40 dataset. For the models trained with data augmentation, we apply rotation with $\theta = 45°$ on x, y, z axis and then sampling from all blocks with sampling rate $f_s = [0.7; 0.4]$. In total, we

Table 4.3: Number of blocks in the training sets of each model.

	MVUB	8i	CAT1	ModelNet40	Total
Model 128	1516	1101	677	2860	6154
Model 64	5777	4797	2777	11147	24498
Model 32	22082	20436	15243	50611	108372
Model 16	87578	86106	45626	224951	444261
Model 8	354617	349760	180037	986253	1870667

augment from each block to 12 variations in terms of density and rotation which significantly increase the volume and diversity of our training set.

Test data

We evaluate the performance of our approach on a diverse set of point clouds in terms of spatial density and content type. All selected point clouds are either used in MPEG Common Test Condition or JPEG Pleno Common Test Condition to evaluate point cloud compression methods. As shown in Table 4.2 the test PCs can be categorized into four sets:

- **MVUB**: Microsoft Voxelized Upper Bodies [116] - a dynamic voxelized point cloud dataset containing five subjects. For testing, we randomly select 2 frames from *Phil* (frame number 10) and *Ricardo* (76) sequences which are both very dense (high ρ) with smooth surfaces.

- **8i**: Dense point clouds from 8i Labs. They are also dynamic voxelized point clouds but each sequence contains the full body of a human subject. In the test set, *loot* (1000) and *redandblack* (1510) are from 8i Voxelized Full Bodies (8iVFB v2) [56] while *boxer* and *thaidancer* are selected and downsampled to 10 bits from 8i Voxelized Surface Light Field (8iVSLF) dataset.

- **CAT1**: static point clouds for cultural heritage and other related 3D photography applications [1]. We select *Arco_Valentino_Dense_vox12, Frog_00067_vox12, and Shiva_00035_vox12* from this dataset and downsample to 10 bits to validate the performance of our method. PCs from this dataset are less dense compared to the previous two datasets. *Frog_00067* has smoother surfaces compared to the other two PCs which contain rough surfaces.

- **USP**: an inanimate dataset from the University of São Paulo, Brazil, related to cultural heritage with 10 bits geometry precision [201]. *BumbaMeuBoi* and *RomanOilLight* are two selected point clouds from this dataset. PCs from

USP dataset have simple shape with smooth surfaces. *BumbaMeuBoi* is the sparsest PC in our test set with the smallest ρ.

Figure 4.7 illustrates the test point clouds.

Training procedure

We train 5 models for 5 input block sizes, i.e., 128, 64, 32, 16, 8. The mini-batch sizes are 1, 8, 64, 128, 128, respectively. Our models are implemented in TensorFlow and trained with Adam [102] optimizer, a learning rate of 0.001 for 80 epochs on a GeForce RTX 2080 GPU.

4.4.2 . Performance evaluation and ablation studies

In the following, we evaluate the performance of the proposed approach as well as the impact of its various components. We start with models without data augmentation nor context extension in order to study the optimal maximal partitioning depth for our method and establish a baseline for the evaluation. Next, on top of the best encoder in this experiment (**Baseline**), we separately add data augmentation (**Baseline + DA**) and context extension (**Baseline + CE**). Finally, **Baseline + DA + CE** incorporates both data augmentation and context extension. We compare our method against the state-of-the-art point cloud compression method G-PCC from MPEG [71] which has a dedicated lossless geometry mode for static point clouds. We report the number of bits per occupied voxel (bpov) for each test point cloud and the average per dataset.

In all experiments, the high-level octree plus partitioning signal are directly converted to bytes without any compression. For the encoders with context extension, we signal the selected size using two bits (maximum 4 options on block 8). This information is also directly converted to bytes in the bitstream. On average, signaling bits account for 2.44% of the bitstream.

Optimal maximum partition depth

To evaluate the effectiveness of the partitioning scheme, we increase the maximum partitioning level from 1 to 5, corresponding to a minimum block size of 64, 32, 16, 8, and 4. As 3D convolution is not able to efficiently exploit voxel relations on a very small receptive field, we do not train a separate model for block 4. Instead, we use the model trained on blocks of size 8 to predict its probabilities.

Table 4.4 shows the average bpov of our encoder on the 4 test datasets at 4 partitioning levels. The results with 5 partitioning levels are identical to 4 partitioning

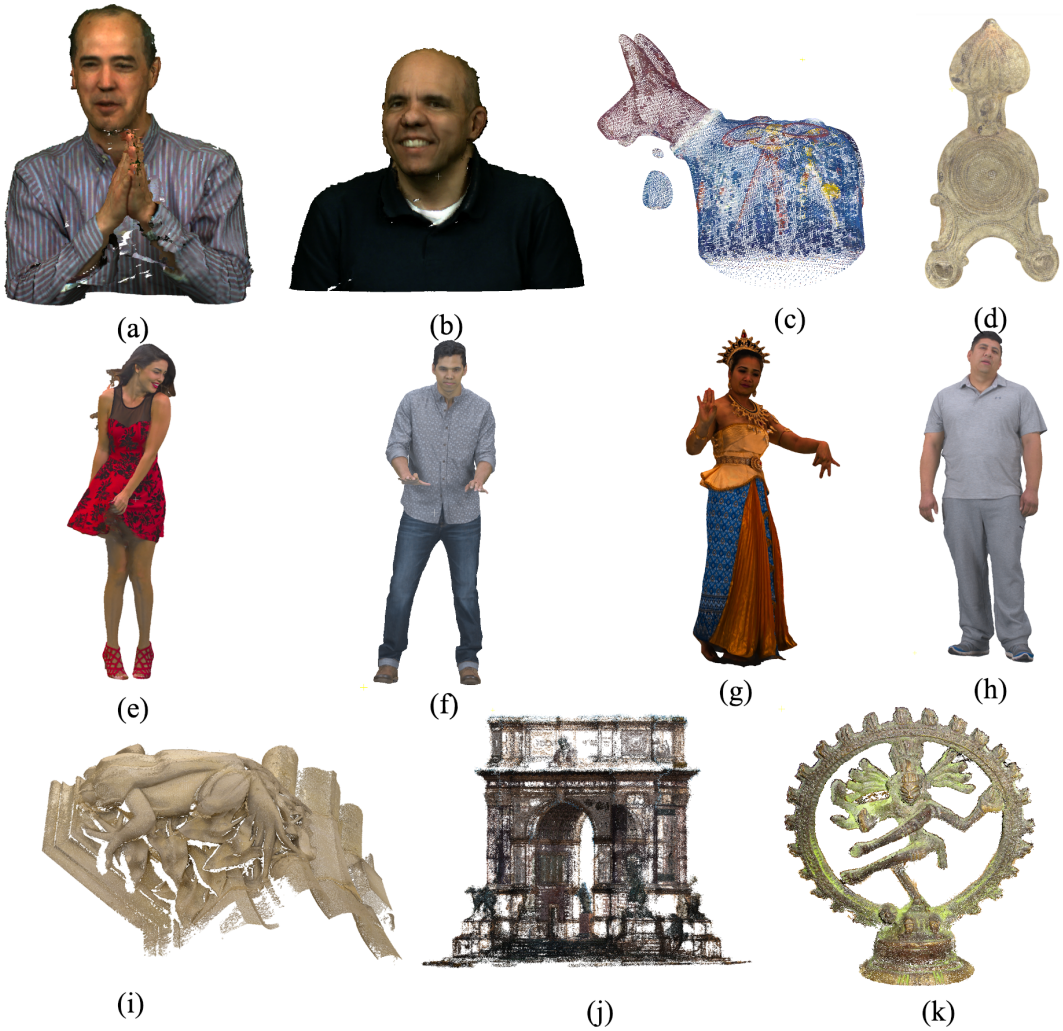


Figure 4.7: Point clouds in the test set. (a) Phil, (b) Ricardo (c) BumbaMeuBoi (d) RomanOilLight, (e) Redandblack, (f) Loot, (g) Thaidancer (h) Boxer, (i) Frog, (j) Arco Valentino, (k) Shiva.

Table 4.4: Average rate in bpov per dataset at different partitioning levels and the gain over the encoder with 1 partitioning level.

		1 level	**2 levels**		**3 levels**		**4 levels**	
Dataset	Point Cloud	bpov	bpov	Gain	bpov	Gain	bpov	Gain
MVUB	Phil	0.8943	0.8295	-7.25%	0.8206	-8.24%	0.8205	-8.25%
	Ricardo	0.8109	0.7511	-7.37%	0.7440	-8.25%	0.7440	-8.25%
	Average	**0.8256**	**0.7903**	**-7.31%**	**0.7823**	**-8.25%**	**0.7823**	**-8.25%**
8i	Redandblack	0.7920	0.7269	-8,22%	0.7191	-9.20%	0.7190	-9.22%
	Loot	0.7017	0.6347	-9.56%	0.6271	-10.63%	0.6271	-10.63%
	Thaidancer	0.7941	0.7360	-7.32%	0.7298	-8.10%	0.7297	-8.11%
	Boxer	0.6462	0.5960	-7.77%	0.5901	-8.68%	0.5900	-8.70%
	Average	**0.7335**	**0.6734**	**-8.22%**	**0.6665**	**-9.15%**	**0.6665**	**-9.16%**
CAT1	Frog	1.9497	1.8406	-5.60%	1.8216	-6.57%	1.8214	-6.58%
	Arco Valentino	5.4984	5.2947	-4.52%	5.2051	-5.33%	5.2050	-5.34%
	Shiva	3.7964	3.6632	-3.51%	3.6400	-4.01%	3.6403	-4.11%
	Average	**3.7482**	**3.5845**	**-4.54%**	**3.5569**	**-5.31%**	**3.5556**	**-5.34%**
USP	BumbaMeuBoi	6.3618	5.8235	-8.46%	5.7305	-9.92%	5.7305	-9.92%
	RomanOilLight	1.8708	1.7157	-5.14%	1.7030	-5.84%	1.7030	-5.84%
	Average	**4.0853**	**3.7696**	**-6.80%**	**3.7168**	**-7.88%**	**3.7168**	**-7.88%**

levels and are not shown in the table. We observe that, as partitioning levels increases, the corresponding gain over single-level also increases. However, adding the 3^{rd} and 4^{th} level yields only a slight improvement compared to adding the 2^{nd} level. This can be explained with Figure 4.8 showing the percentages of occupied voxels in each partition size. We observe that most voxels are encoded using blocks 64 and 32, while very few voxels are encoded using blocks of smaller size. Adding more partitioning levels enables to better adapt to point cloud geometry, however, this is not often compensated by a bitrate reduction of the sub-blocks, since in the smaller partitions the encoder has access to limited contexts, resulting in less accurate probability estimation. However, there is an increase in the portion of block 32 and 16 on CAT1 and USP compared to MVUB and 8i. This reflects the density characteristics of each dataset: on sparser datasets (CAT1 and USP), the algorithm tends to partition point cloud into smaller blocks to eliminate as much empty space as possible. Based on these observations, we use a maximum of 4 partitioning levels for our baseline codec in later experiments.

Comparison with G-PCC

In Table 4.5, we report the output bitrate of our methods to compare with MPEG G-PCC. Both our method and G-PCC perform better on dense PCs while having higher rates on sparser PCs. Compared to G-PCC, the **Baseline** encoder obtains

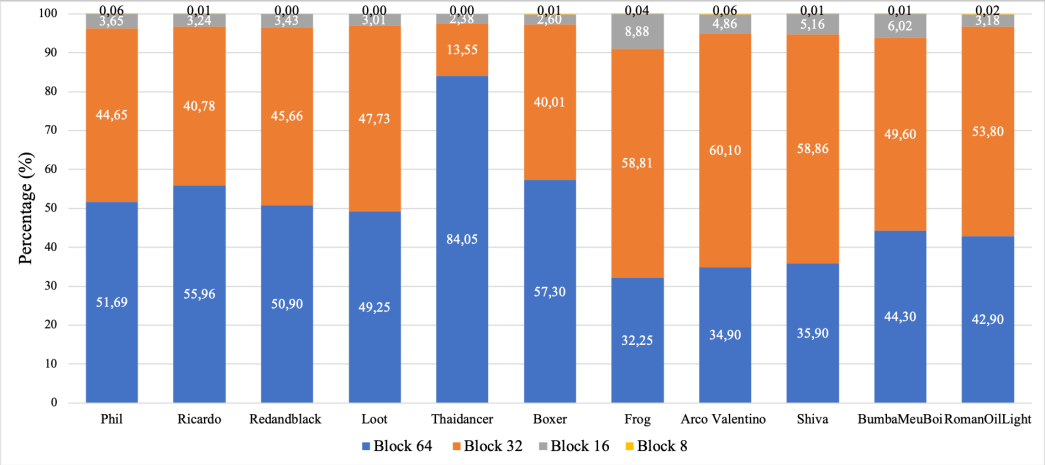


Figure 4.8: Percentage of occupied voxels encoded in each partition size. From top to bottom: block 8, 16, 32, 64. Most of occupied voxel are encoded in block 64 and block 32.

a significant gain – over 29% bitrate reduction on dense point clouds from MVUB and 8i dataset. On CAT1 and USP datasets, our method achieves a comparable rate with G-PCC. In particular, for Arco Valentino and BumbaMeuBoi, the two point clouds having the lowest ρ, our baseline codec yields a rate higher than G-PCC (+8.17% and +5.10%, respectively). For point clouds with high local density levels, our VoxelDNN could efficiently leverage the relations between voxels and predict more accurate probability. In contrast, probability prediction is less accurate on sparser point clouds.

This can be partially solved by adding data augmentation during training. Indeed, by random subsampling the point clouds in the training set, VoxelDNN learns to predict more accurate probabilities when the point cloud is less dense. **Baseline + DA** yields higher gains over G-PCC on CAT1 and USP compared to **Baseline**, with average bitrate reductions of about 1.54% and 4.09%, respectively. On the other hand, we observe a small degradation of the performance compared to **Baseline** for denser datasets, such as MVUB and 8i dataset. This is somehow expected, as data augmentation increases the generalization capability of VoxelDNN, which instead is more adapted to denser PCs in the baseline mode.

The encoder with context extension, **Baseline + CE**, obtains a better rate on all test point clouds compared to the **Baseline** encoder, regardless of the density, with an average further bitrate reduction of 4.8% over G-PCC. The cost to be paid for this performance improvement is a higher computational complexity in the encoding process.

The last two columns of Table 4.5 show the experiment results for the encoder

Table 4.5: Average rate in bpov of proposed method and percentage gains compared with MPEG G-PCC (negative percentages mean bitrate reduction).

		G-PCC	**Baseline**		**Baseline + DA**		**Baseline + CE**		**Baseline + DA + CE**	
Dataset	Point Cloud	bpov	bpov	Gain over G-PCC	bpov	Gain over G-PCC	bpov	Gain over G-PCC	bpov	Gain over G-PCC
MVUB	Phil	1.1617	0.8205	-29.37%	0.8954	-22.92%	0.7601	-34.57%	0.8252	-28.97%
	Ricardo	1.0672	0.7440	-30.28%	0.8235	-22.84%	0.6874	-35.59%	0.7572	-29.05%
	Average	**1.1145**	**0.7823**	**-29.83%**	**0.8595**	**-22.88%**	**07238.**	**-35.06%**	**0.7912**	**-29.01%**
8i	Redandblack	1.0899	0.7190	-34.3%	0.7772	-28.69%	0.6645	-39.03%	0.7003	-35.75%
	Loot	0.9524	0.6271	-34.16%	0.6282	-34.04%	0.5766	-39.46 %	0.6084	-36.12%
	Thaidancer	0.9985	0.7297	-26.92%	0.7253	-27.36%	0.6769	-32.21%	0.6627	-33.63%
	Boxer	0.9479	0.5900	-37.76%	0.6573	-30.66%	0.5503	-41.95%	0.5906	-37.69%
	Average	**0.9972**	**0.6665**	**-33.22%**	**0.6870**	**-30.19%**	**0.6171**	**-38.12%**	**0.6405**	**-35.77%**
CAT1	Frog	1.9085	1.8214	-4.56%	1.7662	-7.64%	1.6971	-11.08%	1.7071	-10.55%
	Arco Valentino	4.8119	5.2050	+8.17%	5.0639	+5.24%	4.9923	+3.75%	4,9900	+3.70%
	Shiva	3.6721	3.6403	-0.87%	3.5838	-2.04%	3.4619	-5.72%	3.5135	-4.32%
	Average	**3.4642**	**3.5556**	**+0.91%**	**3.7413**	**-1.54%**	**3.3838**	**-2.32%**	**3.4035**	**-3.72%**
USP	BumbaMeuBoi	5.4522	5.7305	+5.10%	5.3831	-1.27%	5.3580	-1.73%	5.066	-7.08%
	RomanOiLight	1.8604	1.7030	-8.46%	1.7319	-6.91%	1.6130	-13.30%	1.6231	-12.76%
	Average	**3.6563**	**3.7168**	**-1.68%**	**3.5575**	**-4.09%**	**3.4855**	**-7.51%**	**3.4855**	**-9.91%**

incorporating both data augmentation and context extension, **Baseline + DA + CE**. On average, we have a higher gain than **Baseline** and **Baseline + DA** because of the Context Extension. As expected, comparing with **Baseline + CE**, **Baseline + DA + CE** has increasing gains on CAT1 and USP datasets while obtaining a lower gain on MVUB and 8i datasets. Despite the different performance trends for different densities of the input point clouds, we obtain, on average, a bitrate reduction of 20.17% compared to G-PCC. Note that, in practice, if the characteristics of point cloud to be coded are known in advance, our approach is flexible, in that we could deploy different models targeting a specific application (cultural heritage, tele-immersive conferencing, etc.) and content type to obtain the best compression rate.

Effect of PC content and density on coding performance

In order to better understand the performance of our codec for different types of content, we plot in Figure 4.9 the average bpov as a function of the percentage of occupied voxels for each block 64 of *Phil, Loot, Arco Valentino* and *BumbaMeuBoi* with the **Baseline + DA + CE** encoder. Notice that each block 64 can be split up to different partition levels, indicated by the size of the dots in the figure. The distribution of the density of blocks 64 is shown in the top panel.

From this figure, we can draw some observations. First, most of blocks are partitioned into 3 levels (smallest dots) and the majority of the remaining blocks

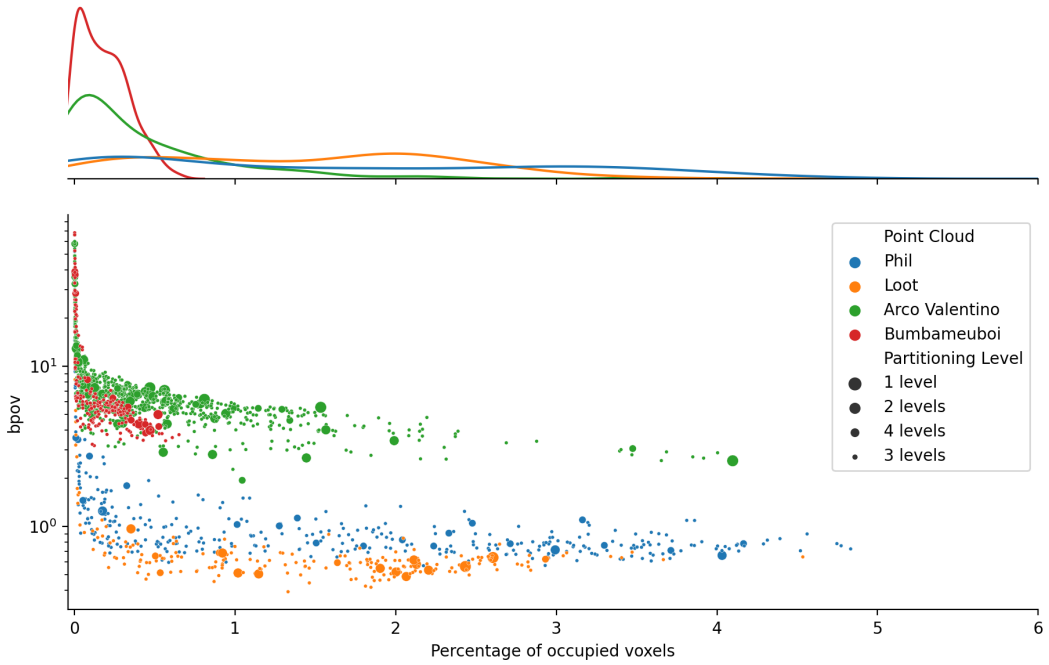


Figure 4.9: Performance on block 64 on four test point clouds. Each point corresponds to a block 64 with percentage of occupied voxels (ρ) and bpov (log scale) performance of that block. The size of each point indicates the partitioning level and each partitioning level was sized according to its frequency. Higher points indicate that VoxelDNN is performing worse. The marginal distributions of occupied voxels for each point cloud are on the top of the scatter plot.

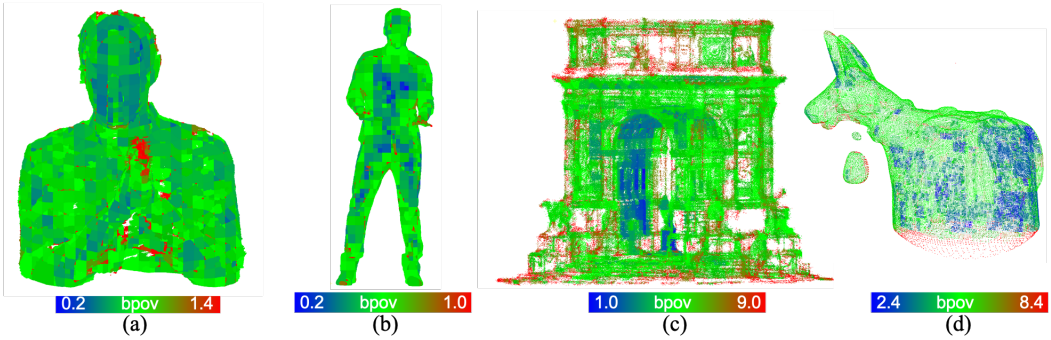


Figure 4.10: Output geometry bitrate in bpov per block. (a) Phil, (b) Loot, (c) Arco Valentino, (d) BumbaMeuBoi. The heatmap bar below each subfigure shows the minimum and maximum bpov and the corresponding color.

are partitioned into 2 or 4 levels. Second, in each point cloud, denser blocks are easier to compress, as mentioned before, due to the better capabilities of convolution to capture spatial relations. On the other hand, our approach becomes inefficient when the blocks are less dense, and the bitrate associated to the very sparse blocks rapidly grows by an order of magnitude compared to the rest. This phenomenon is true for all kinds of contents, although it has a stronger effect when the block density distribution is skewed to the left, such as for *Arco Valentino* or *BumbaMeuBoi*, which have the highest bitrates in our experiments.

We can also observe a content-dependence trend in the figure, which appears like a vertical offset for different PCs. *Arco Valentino* and *RomanOilLight* overall have higher bpov compared to *Phil* and *Loot* with the same number of occupied voxels. This suggests that local density alone is not the only factor affecting the performance of our approach, but that somehow higher-order statistics enter into play. We will speculate more about this behaviour when discussing the bitrate allocation in Figure 4.10. Further analysis of this trend, as well as how to take better into account the PC characteristics to improve coding performance, are left to future work.

Selection of context extension and impact on the partitioning

Figure 4.11 shows how many times an extended block size is selected in the **Baseline + DA + CE** experiments. First, it can be seen that in most cases our encoder choose to extend the context to encode the current block, and mostly the immediate larger size is selected. By extending context to exploit geometry infor-

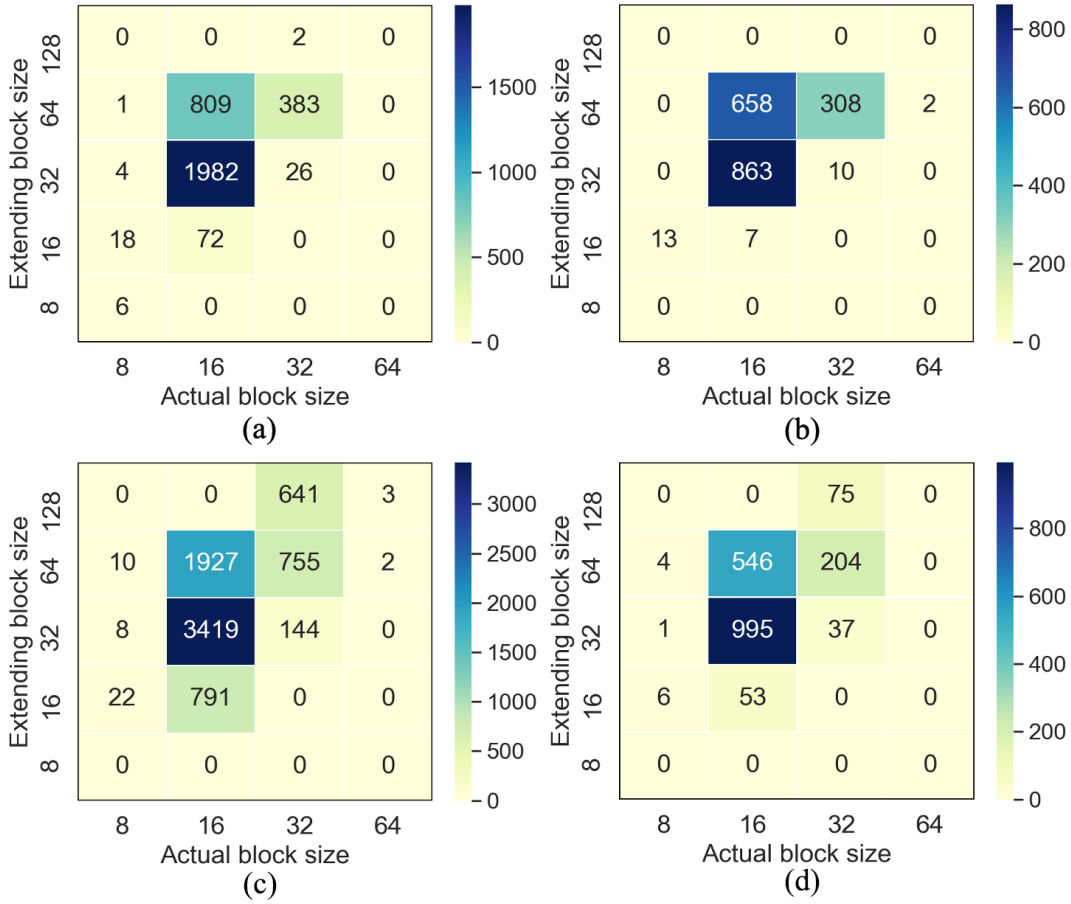


Figure 4.11: Number of extending block size for each block. (a) Phil, (b) Loot, (c) Arco Valentino, (d) BumbaMeuBoi. Most of the time, the encoder extend the context to neighboring voxels instead of independently encoding a block.

mation from the neighboring voxels, VoxelDNN can leverage a larger amount of information and predict a better probability. In most cases where the encoder does not extend the context, the blocks are on the border of the volume, corresponding to a mostly empty extending area.

By summing the quantities in each column, we obtain the number of blocks which are encoded using each block size and we observe that large parts of the point cloud are partitioned into block 32 or 16. This is in contrast with the previous observation on baseline experiments where the most frequent partitions are 64 and 32 (Figure 4.8). This has an intuitive explanation: without context extension, small block sizes of 32 or 16 were insufficient to provide a representative enough context for VoxelDNN in most of the cases, even if they would better adapt to areas with low point density. Conversely, the context extension allows to

compensate for the small block dimension and renders these modes competitive. As a result, context extension significantly affects the optimal partitioning and enables VoxelDNN to adapt better to local sparsity while still providing enough contextual information to predict accurate probabilities.

Using multiple models for the context

For the multi-resolution encoder, instead of using a separate model for each block size, VoxelDNN can use only a single neural network to predict the distribution. Specifically, we place small blocks (8, 16, 32) into a block of size 64 and then use the network for block 64 to predict and extract the corresponding distributions. This method of computing the occupancy distribution is different from Context Extension in that the surrounding voxels are always set to 0. In Table 4.6, we compare the performance of using a single model with **Baseline**, which is a multi-models encoder. In this experiment, both encoders have 4 maximum partitioning levels and use the same model 64. On average, by having a separate model for each block size, a multi-model encoder obtains about 1% additional gain over G-PCC compared to the single model encoder. This amount of gain indicates that the bigger VoxelDNN model can predict the conditional distribution on smaller blocks as efficiently as using a separate model for each block size. However, model 64 is trained on blocks of size 64 only, and learns features at that scale. In general, a model trained on small blocks could better capture the context from small input blocks and thus provides a higher gain in some circumstances.

Visualization of the bitrate allocation on coded PCs

The bpov heatmaps of 4 point cloud are shown in Figure 4.10. The blocks in the figures reflect the optimal partitioning obtained by the algorithm. First, we visually confirm what found in Figure 4.9, i.e., VoxelDNN performs better, i.e., achieves a small bitrate, in the smooth and dense areas of the point cloud. Conversely, in the noisy areas (*Phil*'s hand, *Loot*'s hand), sudden holes (*Arco Valentino*) or very sparse regions (edges in *Arco Valentino*, the bottom part of *BumbaMeuBoi*), which are indicated in red, the performance is worse. We can argue that the density of a point cloud, together with the smoothness and noise characteristics of the content, are among the main factors that influence the performance of VoxelDNN. On the other hand, we can argue that noisy and very sparse areas are intrinsically difficult to code in general, and indeed also the MPEG G-PCC codec requires a large number of bits to encode point clouds such as *BumbaMeuBoi* and *Arco Valentino*.

4.4.3 . Computational complexity analysis

Table 4.6: Single model and multi-models comparison.

		G-PCC	**Single model**		**Multi-models**	
	Point Cloud	bpov	bpov	Gain over G-PCC	bpov	Gain over G-PCC
MVUB	Phil	1.1617	08312	-28.45%	0.8205	-29.37%
	Ricardo	1.0672	0.7541	-29.34%	0.7440	-30.28%
	Average	**1.1145**	**0.7927**	**-28.89%**	**0.7823**	**-29.83%**
8i	Redandblack	1.0899	0.7320	-32.84%	0.7190	-34.3%
	Loot	0.9524	0.6403	-32.77%	0.6271	-34.16%
	Thaidancer	0.9985	0.7305	-26.84%	0.7297	-26.92%
	Boxer	0.9479	0.6008	-36.62%	0.5900	-37.76%
	Average	**0.9972**	**0.6759**	**-32.27%**	**0.6665**	**-33.22%**
CAT1	Frog	1.9085	1.8433	-3.42%	1.8214	-4.56%
	Arco Valentino	4.8119	5.2173	+8.42%	5.2050	+8.17%
	Shiva	3.6721	3.6595	-0.34%	3.6403	-0.87%
	Average	**3.4642**	3.5734	+1.56%	**3.5556**	**+0.91%**
USP	BumbaMeuBoi	5.4522	5.7501	+5.46%	5.7305	+5.10%
	RomanOiLight	1.8604	1.7094	-8.12%	1.7030	-8.46%
	Average	**3.6563**	3.7298	-1.33%	**3.7168**	**-1.68%**

A well-known drawback of auto-regressive generative models such as PixelCNN and VoxelDNN is the sequential generation of the symbol probabilities. This requires to run the network for each voxel, which has a complexity that increases linearly with the number of voxels. Therefore, VoxelDNN has a computational complexity which is 3 orders of magnitures bigger than G-PCC.

Table 4.7 reports the encoding and decoding time of our **Baseline** and **Baseline + CE**. Tests are benchmarked on an Intel(R) Xeon(R) Silver 4110 CPU @ 2.10GHz machine with an Nvidia GeForce GTX 2080 GPU and 16 GB of RAM, running Ubuntu 16.04. Our encoding time is highly dependent on the number of blocks and the number of voxels within each block. Besides, the number of modes in the partitioning algorithm and context extension also influence the complexity. The **Baseline + CE** encoder tries all the extending modes and selects the best one, thus its average encoding time is higher than **Baseline** – an increase of about 182%. The reason why the encoding time for the **Baseline** codec is lower than the decoding time is purely implementative: at the encoder it is possible to predict the whole block probabilities in a single batch on a GPU, while in a realistic scenario, at the decoder side the voxels need to be individually decoded.

Table 4.7: Average runtime (in seconds) of different encoders comparing with G-PCC.

	G-PCC	**Baseline**	**Baseline + CE**
Encoding	2.91	3282	9271
Decoding	2.85	6783	5765

When context extension is enabled, point clouds are partitioned into even smaller blocks, corresponding to a smaller complexity at the decoder, as a smaller number of voxels need to be decoded. On the other hand, the total parameters of each VoxelDNN model corresponds only to about 3.5 MB which is a small-size network in practice. Notice that the bottleneck in our system comes from the adoption of an auto-regressive model, which has the advantage of providing, in principle, an exact likelihood estimation of the data, though at a high computational cost. We are currently investigating the use of alternative generative approaches that avoid sequential probability estimation.

4.5 . Conclusions and future work

This chapter presents a lossless compression method for point cloud geometry. We extend a well-known auto-regressive generative model initially proposed for 2D images to the 3D voxel space, and we incorporate 3D data augmentation to efficiently exploit the redundancies between points. This approach enables to build accurate probability models for the arithmetic coder. As a result, when using an adaptive partitioning scheme and context extension, our solution outperforms MPEG G-PCC over a diverse set of point clouds.

Our analyses on the performance of the proposed method indicate at least two major avenues for improvement. On one hand, handling low-density point clouds would require to rethink the network architecture to handle sparse input data. On the other hand, a major drawback of VoxelDNN is the high computational cost of sequential probability generation, which we plan to replace in the future by a more efficient generative model.

The contents of this chapter have been published in D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Learning-Based Lossless Compression of 3D Point Cloud Geometry," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 4220–4224 and 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, Dec. 2021.

5 - Deep Multiscale Lossless Point Cloud Geometry Compression

5.1 . Introduction

In order to efficiently code point cloud geometry losslessly, it is necessary to accurately estimate the occupancy probabilities to be employed into a context adaptive arithmetic codec. In our previous work, we have modeled the voxel occupancy distributions using a likelihood-based deep autoregressive network called VoxelDNN [132], inspired by the popular PixelCNN model [136]. VoxelDNN achieves state-of-the-art gains (up to 34%) over the MPEG G-PCC reference codec.

Autoregressive models can accurately predict probability distributions. However, the decoding process using this approach is equivalent to sampling from the high-dimensional conditional distribution of voxel occupancies, which is computationally complex as it demands one network evaluation per voxel. In this work, taking inspiration from previous work in 2D image generation [147], we propose a multiscale method (named MSVoxelDNN) for lossless geometry compression of static dense point clouds which addresses the complexity problem of VoxelDNN. Our main contributions are:

- We introduce for the first time a multiscale deep context model in the voxel domain to estimate occupancy probabilities, in which higher-resolution scales are modeled conditioned on the lower-resolution ones.

- We accelerate the inference by parallelizing voxel prediction. At each scale, voxels are partitioned into groups. Voxels belonging to the same group are assumed to be conditionally independent from each other. In this way, we can predict all the voxels of the same group *simultaneously*, reducing the computation time. Instead, each group of voxels is assumed to depend on the previously decoded ones, and thus out context model can leverage dependencies between groups.

Compared to VoxelDNN, we make an approximation in that we do not utilize the statistical dependencies of voxels inside groups (due to the conditional independence assumption). We demonstrate experimentally that this approximation entails only a small loss of performance compared to the original VoxelDNN, and still outperforms significantly MPEG G-PCC in terms of bits per occupied voxel. However, in terms of complexity, MSVoxelDNN is on average 35 and 109 times faster compared to VoxelDNN for encoding and decoding, respectively. The rest

of the chapter is structured as follows: Section 5.2 reviews related work; the proposed MSVoxelDNN method is described in Section 5.3.4; Section 5.4.2 presents the experimental results; and conclusions are drawn in Section 5.5.

5.2 . Related Work

To deal with the irregular distribution of points in 3D space, many PCC methods employ octree representations [153, 120, 96, 65, 66, 67, 87] or local approximations [58]. The octree based method P(PNI) proposed in [67] builds a reference octree using an intra prediction mode. Each octant is then encoded with 255 contexts and a 255×255 frequency table must be transmitted to the decoder. In the MPEG G-PCC codec, geometry can be represented by a pruned octree plus a surface model (trisoup coder) or a full octree (octree coder). To exploit local geometry information within the octree and obtain an accurate context for arithmetic coding, the G-PCC octree coder introduces many techniques such as Neighbour-Dependent Entropy Context [10], intra prediction [8], planar/angular coding mode [11, 6], etc. Instead, in this chapter, we represent the PC geometry in a *hybrid* mode, mixing the octree and voxel domains. On the one hand, octree can adapt to the sparsity of the point cloud, as partitioning stops at the empty node; on the other hand, geometric information are kept and can be naturally processed by a neural network.

Recently, deep learning has been applied widely in point cloud coding in both the octree domain [87, 29] and especially voxel domain [79, 141, 180, 132]. A coding method for static LiDAR point cloud is proposed in [87] which learns the probability distributions of the octree based on contextual information and uses an arithmetic coder for lossless coding. In this work we focus instead on *dense* point clouds, where voxel-based approaches have shown interesting results. In particular, our recent work, VoxelDNN [132], is an auto-regressive based model which predicts the distribution of each voxel conditioned on the previously decoded voxels. VoxelDNN obtains an average rate saving of 30% over G-PCC. The auto-regressive approach of VoxelDNN is similar to PixelCNN [136] which provides accurate 2D data likelihood estimations. However, the common problem of auto-regressive models is the complexity, as these models require a network evaluation per voxel/sub-pixel. In 2D, several methods have been proposed to overcome this limitation. PixelCNN++ [150] models the joint distribution of three color channels simultaneously and proposes several optimizations to PixelCNN. Multiscale PixelCNN [147] generate pixels in certain groups in parallel. The L3C method [122] employs a latent space to facilitate the learning of conditional probability estimates at several scales. While this technique solves the *complexity* issue, the estimated probabilities are not accurate enough and the coding gains are signifi-

cantly less interesting than using Pixel CNN. In this chapter, we aim at finding a good trade-off between complexity and compression performance. Thus, we follow the principle of [147] introducing parallelization and multiscale prediction.

5.3 . Proposed Method

As mentioned before, in this chapter, we focus on voxelized point clouds. Without loss of generality, we assume the point cloud contains $2^n \times 2^n \times 2^n$ voxels. An octree is obtained by recursively spliting the voxel volume into eight sub-cubes until the desired precision is achieved. The occupied cube is marked by bit 1 and empty cube is marked by bit 0. As a result, in each octree node, the generated 8 bits represent the occupancy of the 8 child nodes. A point cloud of size $2^n \times 2^n \times 2^n$ can be represented by an n level octree. In this work, and similar to [132], we partition an n-depth point cloud up to level $n-6$, and thus obtain a $n-6$ high level octree and a number of non-empty binary blocks v of size $2^6 \times 2^6 \times 2^6$, which we refer to as resolution $d = 64$. The high-level octree allows to coarsely remove most of the empty space in the point cloud, which does not contain any useful context information to predict occupancies. All the non-empty voxel blocks are further processed with our multiscale scheme. We first define a raster scan order in the 3D space that scans one voxel at a time in depth, height and width order. We index all voxels in block v at resolution d from 1 to d^3 in 3D raster scan order with:

$$v_i = \begin{cases} 1, & \text{if } i^{th} \text{ voxel is occupied} \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

5.3.1 . VoxelDNN context model

VoxelDNN [132] factorizes the joint distribution of a voxel block into a product of conditional distributions:

$$p(v) = \prod_{i=1}^{d^3} p(v_i | v_{i-1}, v_{i-2}, \ldots, v_1). \tag{5.2}$$

Each term $p(v_i|v_{i-1}, \ldots, v_1)$ above is the occupied probability of the voxel v_i given only the occupancy of previous voxels, referred to as *causality constraint*. All factors in equation (5.2) are estimated by a neural network with masked filters to enforce causality [132]. Therefore, the inference must also proceed sequentially voxel-by-voxel and VoxelDNN performs one network evaluation per voxel.

5.3.2 . MSVoxelDNN context model

In this chapter, we predict multiple voxels in parallel. As mentioned above, this calls for relaxing some dependencies between voxels.

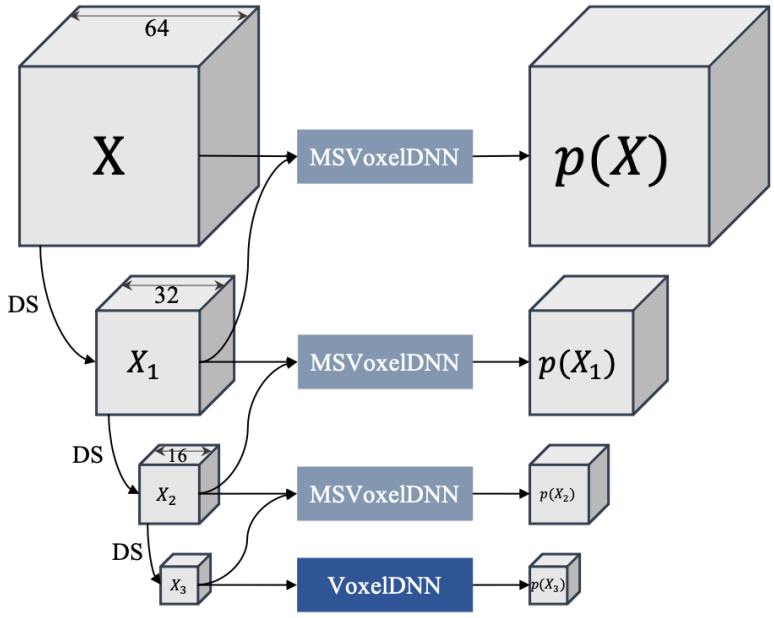


Figure 5.1: Overview of the MSVoxelDNN architecture with input block of size 64 and 3 scales. DS is the downsampling operation (max-pooling). The base resolution of size 8 is encoded using a VoxelDNN context model. The higher resolutions are predicted from lower resolution as well as encoded groups at the same scale. The predicted block probabilities on the right side are passed to an arithmetic coder for encoding voxel occupancies. The final bitstream is the concatenation of all bits at all scales.

First, we divide a voxel block into G separate groups and use v^g to represent all voxels in group g, $g = 1, \ldots, G$. We factorize the joint distribution $p(v)$ as a product of G conditional distributions $p(v^g|v^{g-1}, v^{g-2}, \ldots, v^1)$:

$$p(v) = \prod_{g=2}^{G} p(v^g|v^{g-1}, v^{g-2}, \ldots, v^1) \times p(v^1|v_{LS}). \tag{5.3}$$

Each term $p(v^g|v^{g-1}, v^{g-2}, \ldots, v^1)$ is the joint probability of all voxels in v^g being occupied given all previous groups. Compared to VoxelDNN, we have removed the dependencies of voxels within each group. In return, we are able to predict all voxels in group g in parallel. In addition, given the first group, all other groups can be autoregressively predicted. We model voxels in the first group v^1 as conditionally independent given the lower resolution v_{LS}. This procedure is applied recursively to lower resolutions until the lowest scale, which is encoded using VoxelDNN. Figure 5.1 shows the general scheme of our Multiscale VoxelDNN encoder (MSVoxelDNN). At each step of the pyramid, downsampling is obtained

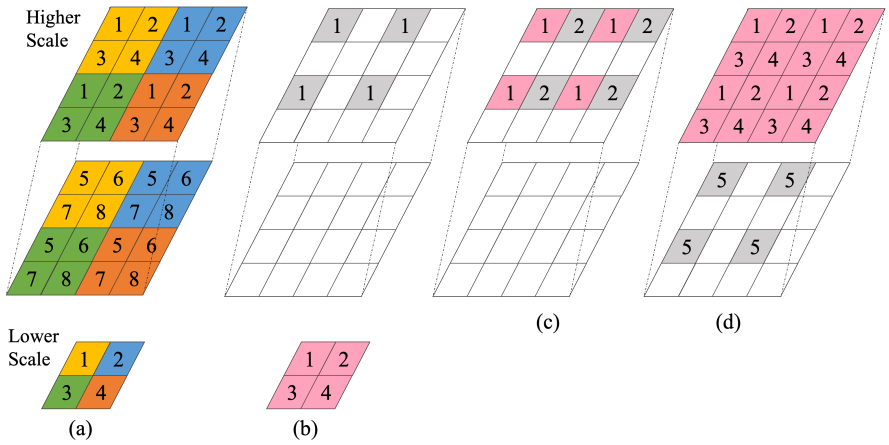


Figure 5.2: Prediction parallelization in MSVoxelDNN / (a) partitioning of a block into groups of conditionally independent voxels. For the sake of clarity and without loss of generality, we show the context modeling for a block of size $2 \times 4 \times 4$. Downsampling is achieved by applying a $2 \times 2 \times 2$ max pooling operator, i.e., the voxels in the lower scale are the maximum of all voxels having the same color on higher scale (MaxPooling operation). (b), (c) and (d) illustrate some steps of the groups prediction. The target groups are in gray, while the input (context) groups are in pink. (b): the 1^{st} group is predicted from all the groups at the lower resolution. (c) the 2^{nd} group is predicted from group 1 at the same resolution. (d) the 5^{th} group is predicted from group 1,2,3 and 4 (at the same scale).

by applying a maxpooling operation of size $2 \times 2 \times 2$ to the high resolution block, i.e., the resulting lower resolution voxel occupancy is one if at least one of the 8 higher resolution voxels is occupied. Therefore, by training the context model to predict the first group from v_{LS}, we somehow learn an inverse max pooling mapping for occupancy probabilities.

At a given scale, voxel groups are obtained by dividing the voxel block into non-overlapping $2 \times 2 \times 2$ blocks. We then select one of the 8 corners for each of $2 \times 2 \times 2$ blocks to get 8 groups. We build different models for different group predictions. Figure 5.2 shows a grouping example and prediction scheme for group 1, 2 and 5. The other groups are modeled from previous groups in a similar manner as group 2, 5.

5.3.3 . Network architecture

We employ a network structure similar to [147]. The network is composed of two stages: first, for each group prediction, we extract features using ResNet blocks. Compared to [147], we reduce the complexity of the feature extraction layer by just using 4 Resnet blocks instead of 12. The features enable to smooth

out the discontinuities in the input voxel data, due to the sampling introduced with the grouping. The so-obtained spatial feature map is then partitioned into contiguous patches, such that there are P patches for each dimension, and thus $P \times P \times P$ in total (we omit here for simplicity the number of channels in the feature space). In the second stage, each patch is input to a shallow auto-regressive model (in this case, a VoxelCNN). We can accelerate training/inference with parallel patches prediction instead of prediction on the whole spatial feature map (i.e., using $P > 1$). However, too small patches can lead to inaccurate probabilities due to limited contexts. Therefore, we use $P = 2$ which require 2^3 small network evaluations in each forward pass ($P = 4$ in [147]).

Figure 5.3 shows the network architecture to predict group 2. Given group 1 of size $D \times D \times D$, the network outputs the predicted occupancy probabilities of all voxels in group 2. First, we use 4 ResNet blocks to extract a feature map from input, in each ResNet block, a 3D convolution with $3 \times 3 \times 3$ filter size is placed between two $1 \times 1 \times 1$ convolution layers. Next, the feature map are spatially divided into 8 patches of the same size and parallelly processed by a shallow VoxelCNN to produce occupancy probabilities. The probabilities are then merged back to a block of size $D \times D \times D \times 1$ which is the size of group 2. The shallow VoxelCNN is composed by one 3D convolutional layer with type A mask, a Resnet block followed by a 3D convolution layer. In each scale, we performs one network evaluation per group and then merge all 8 group probabilities into their spatial position in the output block.

Our predicted probabilities are fed as input to an arithmetic coder for lossless coding. Therefore, to minimize the output bitrate, we train MSVoxelDNN using cross-entropy loss, which is a measurement of the bitrate cost to be paid when the approximate symbol distribution $\hat{p}$ is used instead of the true symbol distribution p.

5.3.4 . Complexity analysis

The bottleneck of VoxelDNN comes from the fact that it is necessary to apply the network on each new voxel to encode/decode. If there are d^3 voxels in a block, VoxelDNN requires $O(d^3)$ network evaluations to estimate probabilities during decoding. For VoxelDNN encoding, it is possible to partially parallelize the process by evaluating several contexts in parallel (since they are known at the encoder side), and thus divide the computational time by a constant factor. However, this does not influence the computational complexity.

Instead, MSVoxelDNN enables to reduce substantially the computational complexity compared to VoxelDNN. At the lowest resolution the block is coded using VoxelDNN with a small number of voxels ($d = 8$). Then, for each resolution level, the network is evaluated only G times, where G (the number of groups) is constant across scales. As we use a $2 \times 2 \times 2$ max pooling as downsampling operator in our

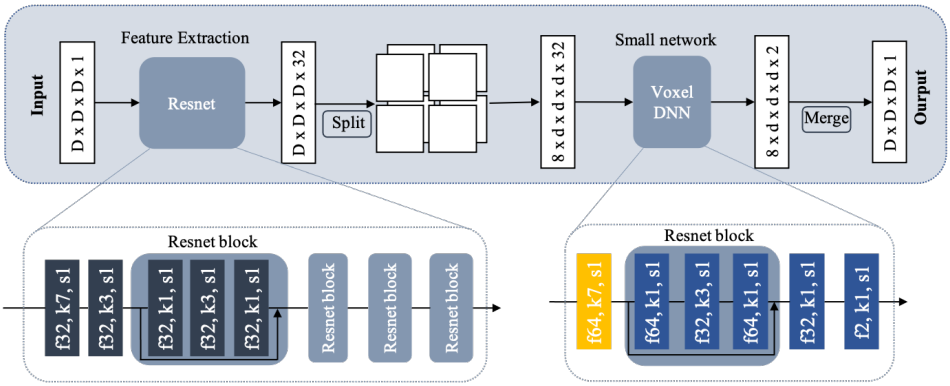


Figure 5.3: Group prediction network architecture. This network predict group 2 from group 1, corresponding to (c) in the example in Figure 5.2. The only learnable modules are Resnet and VoxelDNN. Merge and split operations only reshape data. We use a sequence of Resnet blocks to extract features from input. The features are then spatially split into smaller blocks before parallel processing by a small VoxelDNN. Black rectangular blocks are normal 3D convolution with 'f32,k7,s1' stands for 32 filters, kernel size 7 and stride 1. All convolutional blocks of VoxelDNN are masked convolutions [132], type A mask is in the first layer (in yellow), followed by type B masks.

work, the total number of levels is $\lceil \log_8 d^3 \rceil$, and thus the complexity is $O(\log d)$.

5.4 . Experimental results

5.4.1 . Experimental Setup

Training dataset: We consider point clouds from different and varied datasets, including ModelNet40 [188] which contains 12,311 models from 40 categories and three smaller datasets: MVUB [116], MPEG CAT1 [1] and 8i [56]. We uniformly sample points from the mesh models from ModelNet40 and then scale them to voxelized point clouds with 9 bit precision. To enforce the fairness between the smaller datasets in which we select point clouds for testing, point clouds from MPEG CAT1 are sampled to 10 bit precision as in MVUB and 8i.

To train a MSVoxelDNN model of at scale d we divide all selected PCs into occupied blocks of size $d \times d \times d$. Table 5.1 reports the number of blocks from each dataset for training, with the majority coming from the ModelNet40 dataset. Block 8 dataset is also used to train VoxelDNN model.

Training: We have 3 scales and at each scale we have 8 models for 8 groups and thus, in total we train 24 MSVoxelDNN models. The mini-batch sizes are 32 at scale 64 and 64 at other scale. Our models are implemented in PyTorch and

Table 5.1: Number of blocks in training sets for each block size.

Block size	MVUB	8i	CAT1	ModelNet40	Total
64	5,777	4,797	2,777	1,1147	24,498
32	22,082	20,436	15,243	50,611	108,372
16	87,578	86,106	45,626	224,951	444,261
8	354,617	349,760	180,037	986,253	1,870,667

trained with Adam optimizer, a learning rate of $1e-5$ for 100 epochs on a GeForce RTX 2080 GPU.

Experiments: We evaluate the performance of MSVoxelDNN on a set of dense point clouds from MPEG and Microsoft datasets. These PCs were not used during training. The final bitstream is composed of the bits at all scales and the bits for the high-level octree. The average bits per occupied voxel (in *bpov*) is then measured by dividing the total bits by the number of occupied voxels. We compare the performance of MSVoxelDNN, VoxelDNN and G-PCC (version 12). Note that in the VoxelDNN paper [132], we use a single model for all block sizes. However, in this chapter, we train separate VoxelDNN models for each block sizes on the same dataset as MSVoxelDNN to have a fair comparison.

5.4.2 . Experimental results

In all experiments, the high-level octree are directly converted to bytes without any compression, this part only accounts for less than 1% of the bitstream.

Rate comparison: Table 5.2 reports the rate in *bpov* of the proposed method, MSVoxelDNN, compared with G-PCC and VoxelDNN. We observe that MSVoxelDNN outperforms G-PCC on all test point clouds with rate savings from 11.13% to 26.32%. Compared to VoxelDNN, MSVoxelDNN has smaller gains over G-PCC, with a bitrate increase of 8.12% to 18.21%. This is due to the fact that MSVoxelDNN breaks some dependencies between voxels to model voxel probabilities in parallel, resulting in a less accurate context model.

Complexity comparison: Table 5.3 shows the encoding/decoding run-time for G-PCC, VoxelDNN and MSVoxelDNN. It can be seen that both VoxelDNN and MSVoxelDNN are slower than G-PCC, however there is a very large speedup of MSVoxelDNN compared to VoxelDNN. Specifically, MSVoxelDNN has a 35 and 109 times faster encoding and decoding time, respectively, compared to VoxelDNN.

Table 5.2: Average rate in bpov of MSVoxelDNN compared with MPEG G-PCC v12 and VoxelDNN. The last column shows the gain reduction of MSVoxelDNN from VoxelDNN over G-PCC.

	G-PCC	**VoxelDNN**		**MSVoxelDNN**		
Point Cloud	bpov	bpov	Gain over G-PCC	bpov	Gain over G-PCC	Rate increase over VoxelDNN
Microsoft [116]						
Phil10	1.15	0.82	-29.37%	1.02	-11.13%	+18.25%
Ricardo10	1.07	0.74	-30.28%	0.95	-11.21%	+19.07%
Average	**1.11**	**0.78**	**-28.90%**	**0.99**	**-11.17%**	**+17.73%**
MPEG [1, 56]						
Redandblack10	1.09	0.71	-34.31%	0.87	-20.18%	+14.13%
Loot10	0.95	0.62	-34.16%	0.63	-21.05%	+13.11%
Thaidancer 10	1.00	0.73	-27.00%	0.85	-15.00%	+12.00%
Boxer 10	0.90	0.59	-34.44%	0.70	-26.32%	+8.12%
Average	**1.00**	**0.67**	**-31.79%**	**0.79**	**-20.55%**	**+11.24%**

The asymmetry of coding run-time is due to the possibility to partially parallelize VoxelDNN at the encoder, as mentioned in Section 5.3.4.

5.5 . Conclusions

In this chapter, we propose a Multiscale VoxelDNN method to lossless code the geometry of dense point clouds. On this kind of content, MSVoxelDNN reduces the bitrate compared to G-PCC by up to 17% on average, while reducing by over two orders of magnitudes the decoding complexity of the state-of-the-art VoxelDNN lossless codec. This is obtained by removing some dependencies between voxels in the same group in order to process them in parallel.

The performance of MSVoxelDNN could be further improved by optimizing the grouping of voxels, in such a way to remove only those dependencies that do not contribute significantly to the estimation of conditional occupancy probabilities. Also, the MSVoxelDNN scheme (but this is a common issue of VoxelDNN as well) yield poor performance on sparse point clouds – in general MSVoxelDNN

Table 5.3: Encoding/decoding time comparison per dataset (in seconds).

	G-PCC	**VoxelDNN**	**MSVoxelDNN**
Encoding			
Microsoft	7	4,124	85
MPEG	3	2,459	54
Decoding			
Microsoft	5	10,332	92
MPEG	3	6,274	58

has higher bitrates than G-PCC on point clouds which are not sufficiently dense. This is due to some basic hypotheses behind voxelization and convolutional neural networks, which require some substantial change of network architectures and PC representation. We are currently working towards an efficient learning-based lossless coding scheme for sparser point clouds to overcome these limitations.

The contents of this chapter have been published in D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Multiscale deep context modeling for lossless point cloud geometry compression," in *2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2021, pp. 1–6.

6 - Folding-based Point Cloud Attribute Compression

6.1 . Introduction

In the previous part, we have focused on geometry compression. The sparsity of the geometry was handled with different methods such as block partitioning, adaptive partitioning or the focal loss. Considering voxelized point clouds, geometry can be considered as a sparse binary signal on a regular 3D grid. However, attributes are defined on the geometry and thus on a domain that is both sparse and irregular. In this chapter, we focus on lossy compression of point cloud attributes and we make the observation that point clouds can be interpreted as 2D discrete manifolds in 3D space. Therefore, instead of compressing point cloud attributes using 3D structures such as octrees, we can fold this 2D manifold onto an image. This opens many avenues of research, as it provides, e.g., a way to apply existing image processing techniques straightforwardly on point cloud attributes. Thus, we propose a novel system for folding a point cloud and mapping its attributes to a 2D grid. Furthermore, we demonstrate that the proposed approach can be used to compress static point cloud attributes efficiently.

6.2 . Related Work

This chapter is at the crossroads of static point cloud attribute compression and deep representation learning of 3D data. Compressing static point cloud attributes has been explored using graph transforms [195], the Region-Adaptive Hierarchical Transform (RAHT) [54] and volumetric functions [104]. Graph transforms take advantage of the Graph Fourier Transform (GFT) and the neighborhood structure present in the 3D space to compress point cloud attributes. The RAHT is a hierarchical transform which extends the Haar wavelet transform to an octree representation. In this chapter, we propose a different perspective, and leverage the manifold interpretation of the point cloud by mapping its attributes onto a 2D grid, which can then be compressed as an image.

Deep learning methods have been used for representation learning and compression of point clouds [141]. In particular, the initial folding in our work is inspired by [192] where an autoencoder network is trained on a dataset to learn how to fold a 2D grid onto a 3D point cloud. In our work, we build on this folding idea; however, we employ it in a very different way. Specifically, we do not aim at learning a good representation that can generalize over a dataset; instead, we

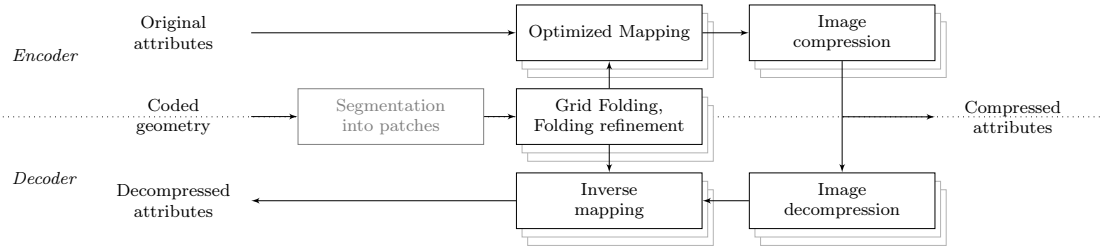


Figure 6.1: Proposed system for attribute compression. Segmentation is optional and can help to adapt to local geometry complexity.

employ the folding network as a parametric function that maps an input 2D grid to points in 3D space. The parameters of this function (i.e., the weights of the network) are obtained by overfitting the network to a specific point cloud. In addition, the original folding proposed in [192] is highly inefficient for PCC as it poorly adapts to complex geometries. In our work, we propose a number of solutions to improve folding.

6.3 . Proposed method

We propose a novel system for compressing point cloud attributes based on the idea that a point cloud can be seen as a discrete 2D manifold in 3D space. In this way, we can obtain a 2D parameterization of the point cloud and we can map attributes from a point cloud onto a grid, making it possible to employ 2D image processing algorithms and compression tools. The overall system is depicted in Figure 6.1. In a nutshell, our approach is based on the following two steps: a) we find a parametric function (specifically, a deep neural network) to fold a 2D grid onto a 3D point cloud; b) we map attributes (e.g., colors) of the original point cloud to this grid. The grid and the parametric function contain all the necessary information to recover the point cloud attributes. Assuming the point cloud geometry is coded separately and transmitted to the decoder, the folding function can be constructed at the decoder side, and the 2D grid is fully decodable without any need to transmit network parameters. In practice, the 3D-to-2D mapping is lossy, which entails a mapping distortion in the step b) above. In the following, we propose several strategies to reduce this mapping distortion.

Notation. We use lowercase bold letters such as $\mathbf{x}$ to indicate 3D vectors (point cloud spatial coordinates), and uppercase letters such as X to indicate sets of 3D points (vectors). We denote with a tilde (like $\tilde{\mathbf{x}}$ or $\tilde{X}$) compressed (distorted) vectors or sets of vectors. We use the notation $\langle S \rangle = \sum_{x \in S} x/|S|$ for the average over a set S.

6.3.1 . Grid folding

We propose a grid folding composed of two steps, namely, an initial folding step to get a rough reconstruction of X and a folding refinement step to improve the reconstruction quality, which is quintessential to map point cloud attributes with minimal mapping distortion.

We fold a grid onto a point cloud to obtain its 2D parameterization by solving the following optimization problem:

$$\min_{f} \mathcal{L}(X, \tilde{X}) \tag{6.1}$$

where X is the set of n points in the original point cloud, $\tilde{X} = f(X, G)$ is the set of n' points in the reconstructed point cloud obtained by folding G onto X where G the set of $n' = w \times h$ points of a 2D grid with 3D coordinates. In general, $n' \neq n$; however, we choose n' to be close to n. $\mathcal{L}$ is a loss function and f is a folding function.

We parameterize f using a neural network composed of an encoder f_e and a decoder f_d such that $\mathbf{y} = f_e(X)$ and $\tilde{X} = f_d(G, \mathbf{y})$. The encoder f_e is composed of four pointwise convolutions with filter sizes of 128 followed by a maxpooling layer. The decoder f_d is composed of two folding layers with $f_d(G, \mathbf{y}) = \mathrm{FL}(\mathrm{FL}(G, \mathbf{y}), \mathbf{y})$. Each folding layer has two pointwise convolutions with filter sizes of 64 and concatenates $\mathbf{y}$ to its input. The last pointwise convolution has a filter size of 3. We use the ReLU activation [131] for the encoder and LeakyReLU activation [118] for the decoder. A one-to-one mapping exists between each point $\tilde{\mathbf{x}}_i$ in the folded grid $\tilde{X}$ and their original position $\mathbf{g}_i$ in the grid G.

We propose the following loss function

$$\mathcal{L}(X, \tilde{X}) = \mathrm{d}_{\mathrm{ch}}(X, \tilde{X}) + \mathrm{d}_{\mathrm{rep}}(\tilde{X}) \tag{6.2}$$

where d_{ch} is the Chamfer distance:

$$\mathrm{d}_{\mathrm{ch}}(X, \tilde{X}) = \sum_{\mathbf{x} \in X} \min_{\tilde{\mathbf{x}} \in \tilde{X}} \|\mathbf{x} - \tilde{\mathbf{x}}\|_2^2 + \sum_{\tilde{\mathbf{x}} \in \tilde{X}} \min_{\mathbf{x} \in \tilde{X}} \|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2, \tag{6.3}$$

and $\mathrm{d}_{\mathrm{rep}}$ is a novel *repulsion loss* computed as the variance of the distance of each point in $\tilde{X}$ to its nearest neighbor:

$$\mathrm{d}_{\mathrm{rep}}(\tilde{X}) = \mathrm{Var}(\{ \min_{\tilde{\mathbf{x}}' \in \tilde{X} \setminus \tilde{\mathbf{x}}} \|\tilde{\mathbf{x}} - \tilde{\mathbf{x}}'\|_2^2 \mid \tilde{\mathbf{x}} \in \tilde{X} \}). \tag{6.4}$$

The Chamfer distance ensures that the reconstruction $\tilde{X}$ is similar to X and the repulsion loss penalizes variations in the reconstruction's density.

We obtain the parameterized folding function f by training a neural network using the Adam optimizer [102]. We use the point cloud X as the single input which is equivalent to overfitting the network on a single sample.

(a) Original (b) Folded (27.63 dB) (c) Refined (30.62 dB) (d) Opt. refined (33.39 dB)

Figure 6.2: Different steps of our proposed attribute mapping method for the first frame of phil9 [116]. Top row: phases of point cloud reconstruction; bottom row: the attributes mapped on a 2D grid, which is later compressed and transmitted. The initial folding (b) provides a rough reconstruction $\tilde{X}$ which is improved with folding refinement (c) and occupancy optimization (d) to reduce the density mismatch between X and $\tilde{X}$. We then map attributes from the point cloud onto a 2D grid. The holes in the grid are filled to facilitate compression with HEVC. We indicate Y PSNRs between original and colors distorted by mapping.

6.3.2 . Folding refinement

The initial folding has difficulties reconstructing complex shapes accurately as seen in Figure 6.2b. Specifically, the two main issues are mismatches in local density between X and $\tilde{X}$ and inaccurate reconstructions for complex shapes. As a result, this introduces significant mapping distortion when mapping attributes from the original PC to the folded one; additionally, this mapping distortion affects the reconstructed point cloud attributes. For compression applications, this is a serious issue as there are now two sources of distortion from both mapping and compression. This is why we propose a folding refinement method that alleviates mismatches in local density and inaccurate reconstructions.

First, we reduce local density variations by considering density-aware grid structure preservation forces inside $\tilde{X}$. Specifically, each point $\tilde{\mathbf{x}}$ is attracted towards the inverse density weighted average of its neighbors $\mathbf{p}_{\text{grid}}$. Since a one-to-one mapping exists between $\tilde{X}$ and G, each point $\tilde{\mathbf{x}}_i$ in the folded grid $\tilde{X}$ has a corresponding point $\mathbf{g}_i$ in the grid G. We then define the inverse density weight ω_i for $\tilde{\mathbf{x}}_i$ as

$$\omega_i = \langle \{ \, \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2 \mid \mathbf{g}_j \in \mathcal{N}_G(\mathbf{g}_i) \, \} \rangle \, . \tag{6.5}$$

with $\mathcal{N}_G(\mathbf{g}_i)$ the set of horizontal and vertical neighbors of $\mathbf{g}_i$ in the grid G. This

encourages the reconstruction to have a more uniform distribution by penalizing high density areas. Given the set Ω comprising all weights ω_i, we define the normalized weights

$$\hat{\omega}_i = (\omega_i - \min(\Omega))/(\max(\Omega) - \min(\Omega)). \tag{6.6}$$

This allows us to define the weighted average

$$\mathbf{p}_{\text{grid}_i} = \langle \{ \, \hat{\omega}_j \tilde{\mathbf{x}}_j \mid \mathbf{g}_j \in \mathcal{N}_G(\mathbf{g}_i) \, \} \rangle \, . \tag{6.7}$$

Second, we set up bidirectional attraction forces between X and $\tilde{X}$ to solve two issues: incomplete coverage, when $\tilde{X}$ does not cover parts of X, and inaccurate reconstructions, when $\tilde{X}$ fails to reproduce X accurately. As a solution, we attract each point $\tilde{\mathbf{x}}$ towards two points $\mathbf{p}_{\text{push}}$ and $\mathbf{p}_{\text{pull}}$. Specifically, $\mathbf{p}_{\text{push}}$ is the nearest neighbor of $\tilde{\mathbf{x}}$ in X and *pushes* $\tilde{X}$ towards X which allows for more accurate reconstructions. On the other hand, $\mathbf{p}_{\text{pull}}$ is the average of the points in X which have $\tilde{\mathbf{x}}$ as their nearest neighbor and allows X to *pull* $\tilde{X}$ closer which alleviates incomplete coverage issues.

Finally, we combine these components into an iterative refinement system to update the point cloud reconstruction:

$$\tilde{\mathbf{x}}_{t+1,i} = \alpha \mathbf{p}_{\text{grid}_{t,i}} + (1 - \alpha)(\mathbf{p}_{\text{push}_{t,i}} + \mathbf{p}_{\text{pull}_{t,i}})/2 \tag{6.8}$$

where $\tilde{\mathbf{x}}_{t,i}$ is the value of $\tilde{\mathbf{x}}_i$ after t iterations and $\tilde{\mathbf{x}}_0 = \tilde{\mathbf{x}}$. The inertia factor $\alpha \in [0, 1]$ balances the grid structure preservation forces in $\tilde{X}$ with the bidirectional attraction forces set up between X and $\tilde{X}$. Preserving the grid structure preserves the spatial correlation of the attributes mapped on the grid and the density-aware aspect of these forces results in more uniformly distributed points. In addition, the bidirectional forces improve the accuracy of the reconstruction significantly.

6.3.3 . Optimized Attribute Mapping

Once a sufficiently accurate 3D point cloud geometry is reconstructed (Figure 6.2c), we can map attributes from X to $\tilde{X}$. To this end, we first build a mapping $m_{X \to \tilde{X}}$ from each point in X to a corresponding point in $\tilde{X}$ (for example, the nearest neighbor). Hence, the inverse mapping $m_{\tilde{X} \to X}$ maps $\tilde{\mathbf{x}}$ back to X. As $m_{X \to \tilde{X}}$ is not one-to-one (due to local density mismatches and inaccuracy of the reconstruction), several points in X can map to the same $\tilde{\mathbf{x}}$. Thus, a given $\tilde{\mathbf{x}}$ can correspond to zero, one or many points in X; we define the number of these points as its occupancy $o(\tilde{\mathbf{x}})$. Attribute mapping from X to $\tilde{X}$ is obtained using $m_{\tilde{X} \to X}$ as the attribute value for a point $\tilde{\mathbf{x}}$ is the average of the attribute values of $m_{\tilde{X} \to X}(\tilde{\mathbf{x}})$. In case $m_{\tilde{X} \to X}(\tilde{\mathbf{x}}) = \varnothing$, we simply assign to $\tilde{\mathbf{x}}$ the attribute of its nearest neighbor in X. As a consequence of this approach, points with higher occupancy tend to have higher mapping distortion, as more attributes are averaged.

To overcome this problem, we integrate the occupancy as a regularizing factor when building the mapping. For each point $\mathbf{x}$ in X, we consider its k nearest neighbors set $\mathcal{N}_k(\mathbf{x}) \in \tilde{X}$ and select

$$m_{X \to \tilde{X}}(\mathbf{x}) = \underset{\tilde{\mathbf{x}} \in \mathcal{N}_k(\mathbf{x})}{\arg\min}\, o(\tilde{\mathbf{x}}) \|\tilde{\mathbf{x}} - \mathbf{x}\|_2. \tag{6.9}$$

Specifically, the mapping is built iteratively and the occupancies are updated progressively.

As noted above, when $o(\tilde{\mathbf{x}}) > 1$, the attributes are averaged which introduces distortion. We mitigate this problem by adding rows and columns in the 2D grid (see Fig. 6.2d) using the following procedure. Since $o(\tilde{\mathbf{x}})$ is defined on $\tilde{X}$ and there is a one-to-one mapping between $\tilde{X}$ and G, we can compute mean occupancies row-wise and column-wise. In particular, we compute mean occupancies with zeros excluded and we select the row/column with the maximum mean occupancy. Then, we reduce its occupancy by inserting additional rows/columns around it. We repeat this procedure until we obtain a lossless mapping or the relative change on the average of mean occupancies Δ_r is superior to a threshold $\Delta_{r,min}$.

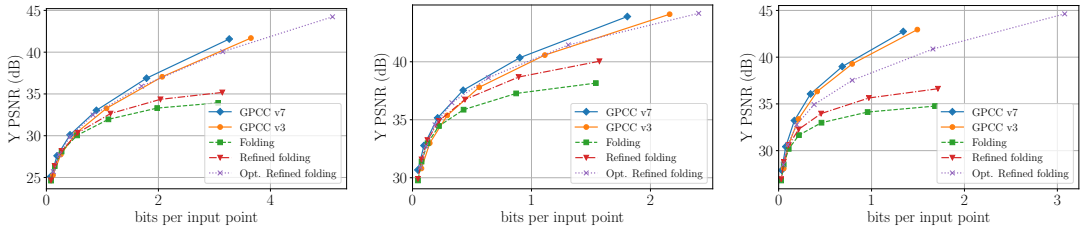
6.4 . Experimental results

We evaluate our system for static point cloud attribute compression and compare it against G-PCC v3 [119] and v7 [4]. We also study the impact of folding refinement and occupancy optimization on our method by presenting an ablation study. Since folding is less accurate on complex point clouds, we manually segment the point clouds into patches and apply our scheme on each patch. The patches are then reassembled in order to compute rate-distortion measures.

We use TensorFlow 1.15.0 [14]. For the folding refinement, we set α to $1/3$ and perform 100 iterations. When mapping attributes, we consider $k = 9$ neighbors for assignment. When optimizing occupancy, we set $\Delta_{r,min}$ to 10^{-6}. We then perform image compression using BPG [27], an image format based on HEVC intra [5], with QPs ranging from 20 to 50 with a step of 5. We note that any other image compression approach could be used here, including learning based approaches.

In Figure 6.3, we observe that our method performs comparably to G-PCC for “longdress" and “redandblack". The performance is slightly worse for “soldier" as its geometry is much more complex making a good reconstruction difficult and introducing mapping distortion. We obtain significant gains in terms of rate-distortion by improving the reconstruction quality using folding refinement and occupancy optimization. This shows the potential of our method and confirms the importance of reducing the mapping distortion.

In our experiments, we’ve found that this approach does not perform well when directly applied on sparser point clouds. Indeed, the mapping is signifi-



(a) "longdress" (b) "redandblack" (c) "soldier"

Figure 6.3: RD curves showing the performance of the different steps of our method on three point clouds [154].

cantly harder in sparser point clouds. Approaches to handle local sparsity such as downscaling may help solve this problem.

6.5 . Conclusion

Based on the interpretation of a point cloud as a 2D manifold living in a 3D space, we propose to fold a 2D grid onto it and map point cloud attributes into this grid. As the mapping introduces distortion, this calls for strategies to minimize this distortion. In order to minimize mapping distortion, we proposed a folding refinement procedure, an adaptive attribute mapping method and an occupancy optimization scheme. With the resulting image, we compress point cloud attributes leveraging conventional image codecs and obtain encouraging results. Our proposed method enables the use of 2D image processing techniques and tools on point cloud attributes.

The contents of this chapter have been published in M. Quach, G. Valenzise, and F. Dufaux, "Folding-Based Compression Of Point Cloud Attributes," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 3309–3313.

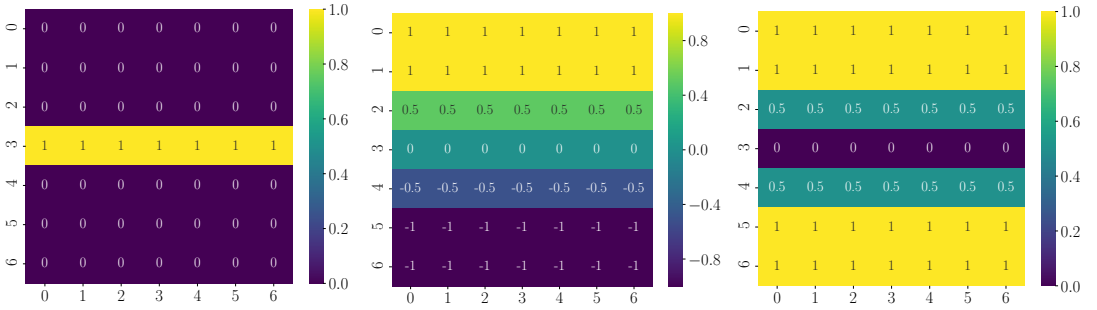
7 - Deep Perceptual Point Cloud Quality Metric

In the previous parts, we have discussed deep learning approaches to compress geometry [141, 142, 180, 182, 168, 126] and attributes [140, 22] of points clouds. Specific approaches have also been developed for sparse LIDAR point clouds [87, 29]. In existing approaches, different point cloud geometry representations are considered for compression: G-PCC adopts a point representation, V-PCC uses a projection or image-based representation and deep learning approaches commonly employ a voxel grid representation. Point clouds can be represented in different ways in the voxel grid. Indeed, voxel grid representations include binary and Truncated Signed Distance Fields (TSDF) representations [50]. TDSFs rely on the computation of normals; however, in the case of point clouds this computation can be noisy. We then ignore the normal signs and reformulate TDSFs to propose a new Truncated Distance Field (TDF) representation for point clouds

Deep learning approaches for lossy geometry compression typically jointly optimize rate and distortion. As a result, an objective quality metric, employed as a loss function, is necessary to define the distortion objective during training. Such metrics should be differentiable, defined on the voxel grid and well correlated with perceived visual quality. In this context, the Weighted Binary Cross Entropy (WBCE) and the focal loss [114] are commonly used loss functions based on a binary voxel grid representation. They aim to alleviate the class imbalance between empty and occupied voxels caused by point cloud sparsity. However, they are poorly correlated with human perception as they only compute a voxel-wise error.

A number of metrics have been proposed for Point Cloud Quality Assessment (PCQA): the point-to-plane (D2) metric [171], PC-MSDM [124], PCQM [125], angular similarity [21], point to distribution metric [94], point cloud similarity metric [20], improved PSNR metrics [93] and a color based metric [179]. These metrics operate directly on the point cloud. However, they are not defined on the voxel grid and hence cannot be used easily as loss functions. Recently, to improve upon existing loss functions such as the WBCE and the focal loss, a neighborhood adaptive loss function [77] was proposed. Still, these loss functions are based on the explicit binary voxel grid representation. We show in this chapter that loss functions based on the TDF representation are more correlated with human perception than those based on the binary representation.

The perceptual loss has previously been proposed as an objective quality metric for images [196]. Indeed, neural networks learn representations of images that are well correlated with perceived visual quality. This enables the definition of the perceptual loss as a distance between latent space representations. For the case of



(a) Binary.

(b) Truncated Signed Distance Field (TSDF).

(c) Truncated Distance Field (TDF).

Figure 7.1: Voxel grid representations of a point cloud. The upper bound distance value is 2 for the TDF and the TSDF. Normals are facing up in the TSDF.

images, the perceptual loss provides competitive performance or even outperforms traditional quality metrics. We hypothesize that a similar phenomenon can be observed for point clouds.

Therefore, we propose a differentiable perceptual loss for training deep neural networks aimed at compressing point cloud geometry. We investigate how to build and train such a perceptual loss to improve point cloud compression results. Specifically, we build a differentiable distortion metric suitable for training neural networks to improve PCC approaches based on deep learning. We then validate our approach experimentally on the ICIP2020 [139] subjective dataset. The main contributions of the chapter are as follows:

- A novel perceptual loss for 3D point clouds that outperforms existing metrics on the ICIP2020 subjective dataset
- A novel implicit TDF voxel grid representation
- An evaluation of binary (explicit) and TDF (implicit) representations in the context of deep learning approaches for point cloud geometry compression

7.1 . Voxel grid representations

In this study, we consider different voxel grid representations for point clouds. A commonly used voxel grid representation is the explicit binary occupancy representation where the occupancy of a voxel (occupied or empty) is represented with a binary value (Figure 7.1a). In the binary (Bin) representation (Figure 7.1a), each voxel has a binary occupancy value indicating whether it is occupied or empty. When the ith voxel is occupied, then $x_i = 1$ and otherwise $x_i = 0$.

Table 7.1: Objective quality metrics considered in this study.

Domain	Name	Signal type	Block aggregation	Learning based	Description
Points	D1 MSE	Coordinates	✗	✗	Point-to-point MSE
	D2 MSE	Coordinates	✗	✗	Point-to-plane MSE
	D1 PSNR	Coordinates	✗	✗	Point-to-point PSNR
	D2 PSNR	Coordinates	✗	✗	Point-to-plane PSNR
Voxel Grid	Bin BCE	Binary	L1	✗	Binary cross entropy
	Bin naBCE	Binary	L1	✗	Neighborhood adaptive binary cross entropy [77]
	Bin WBCE 0.75	Binary	L2	✗	Weighted binary cross entropy with $w = 0.75$
	Bin PL	Binary	L1	✓	Perceptual loss (explicit) on all feature maps
	Bin PL F1	Binary	L1	✓	Perceptual loss (explicit) on feature map 1
	TDF MSE	Distances	L1	✗	Truncated distance field (TDF) MSE
	TDF PL	Distances	L1	✓	Perceptual loss (implicit) over all feature maps
	TDF PL F9	Distances	L1	✓	Perceptual loss (implicit) on feature map 9

Another representation is the implicit TSDF representation which has been employed for volume compression [168]. Instead of an occupancy value, the value of a voxel is the distance from this voxel to the nearest point and the sign of this value is determined from the orientation of the normal (Figure 7.1b). However, this requires reliable normals which may not be available in sparse and/or heavily compressed point clouds.

Hence, we propose an implicit TDF representation which is a variant of the TSDF without signs and therefore does not require normals. In the implicit TDF representation (Figure 7.1c), the ith voxel value is $x_i = d$, where d is the distance to its nearest occupied voxel. Consequently, $x_i = 0$ when a voxel is occupied and $x_i = d$ with $d > 0$ otherwise. Additionally, we truncate and normalize the distance values into the $[0, 1]$ interval with

$$x_i = \min(d, u)/u, \tag{7.1}$$

where u is an upper bound value.

In this study, we focus on the explicit binary and implicit TDF representations.

7.2 . Objective quality metrics

In Table 7.1, we present the objective quality metrics considered in this study. Specifically, we evaluate metrics that are differentiable on the voxel grid to evaluate their suitability as loss functions for point cloud geometry compression. We include metrics defined on binary and TDF representations and we compare their performance against traditional point set metrics.

7.2.1 . Voxel grid metrics

We partition the point cloud into blocks and compute voxel grid metrics for each block. For each metric, we aggregate metric values over all blocks with either L1 or L2 norm. Specifically, we select the best aggregation experimentally for each metric.

Given two point clouds $\mathbf{A}$ and $\mathbf{B}$, we denote the ith voxel value for each point cloud as $x_i^{(\mathbf{A})}$ and $x_i^{(\mathbf{B})}$. Then, we define the WBCE as follows

$$-\frac{1}{N}\sum_i \Big(\alpha x_i^{(\mathbf{A})} \log(x_i^{(\mathbf{B})}) + (1-\alpha)(1-x_i^{(\mathbf{A})}) \log(1-x_i^{(\mathbf{B})})\Big), \tag{7.2}$$

where α is a balancing weight between 0 and 1. The binary cross entropy (BCE) refers to the case $\alpha = 0.5$.

Different from the WBCE, the Focal Loss (FL) amplifies ($\gamma > 1$) or reduces ($\gamma < 1$) errors and is defined as follows

$$-\sum_i \Big(\alpha x_i^{(\mathbf{A})} (1-x_i^{(\mathbf{B})})^\gamma \log(x_i^{(\mathbf{B})}) + (1-\alpha)(1-x_i^{(\mathbf{A})})(x_i^{(\mathbf{B})})^\gamma \log(1-x_i^{(\mathbf{B})})\Big), \tag{7.3}$$

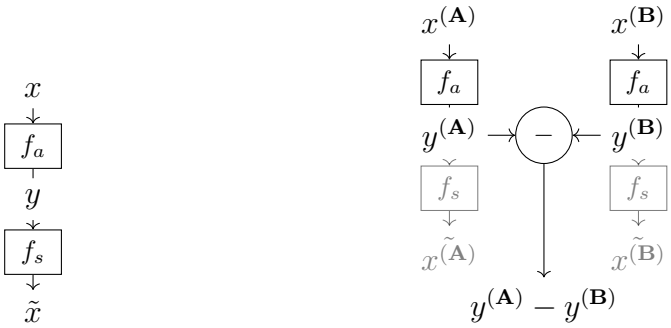
where α is a balancing weight and the log arguments are clipped between 0.001 and 0.999.

Compared to the WBCE, the FL adds two factors $(1-x_i^{(\mathbf{B})})^\gamma$ and $(x_i^{(\mathbf{B})})^\gamma$. However, while in the context of neural training, $x_i^{(\mathbf{B})}$ is an occupancy probability, in the context of quality assessment, $x_i^{(\mathbf{B})}$ is a binary value. As a result, the FL is equivalent to the WBCE since γ has no impact in the latter case. For this reason, we include the WBCE with $\alpha = 0.75$ in our experiments as an evaluation proxy for the FL used in [142].

The neighborhood adaptive BCE (naBCE) [77] was proposed as an alternative to the BCE and FL [114]. It is a variant of the WBCE in which the weight α adapts to the neighborhood of each voxel u resulting in a weight α_u. Given a voxel u, its neighboorhood is a window W of size $m \times m \times m$ centered on u. Then, the neighborhood resemblance r_u is the sum of the inverse euclidean distances of neighboring voxels with the same binary occupancy value as u. Finally, the weight α_u is defined as $\alpha_u = \max(1 - r_u/\max(r)), 0.001)$ where $\max(r)$ is the maximum of all neighborhood resemblances.

The Mean Squared Error (MSE) on the TDF is expressed as follows

$$\frac{1}{N}\sum_i (x_i^{(\mathbf{A})} - x_i^{(\mathbf{B})})^2. \tag{7.4}$$



(a) Autoencoder model. (b) Perceptual loss.

Figure 7.2: Perceptual loss based on an autoencoder. The grayed out parts do not need to be computed for the perceptual loss.

7.2.2 . Perceptual Loss

We propose a perceptual loss based on differences between latent space representations learned by a neural network. More precisely, we use an autoencoder as the underlying neural network. The model architecture of autoencoder used and its training procedure are presented in the following.

Model architecture

We adopt an autoencoder architecture based on 3D convolutions and transposed convolutions. Given an input voxel grid x, we perform an analysis transform $f_a(x) = y$ to obtain the latent space y and a synthesis transform $f_s(y) = \tilde{x}$ as seen in Figure 7.2a. The analysis transform is composed of three convolutions with kernel size 5, stride 2 while the synthesis transform is composed of three transposed convolutions with same kernel size and stride. We use ReLU [131] activations for all layers except for the last layer which uses a sigmoid activation.

Training

Using the previously defined architecture, we train two neural networks: one with explicit representation (binary) and another with implicit representation (TDF).

In the explicit case, we perform training of the perceptual loss with a focal loss function as defined in Eq. (7.3). In the implicit case, we first define the Kronecker delta δ_i such that $\delta_i = 1$ when $i = 0$ and otherwise $\delta_i = 0$. Then, we define an

adaptive MSE loss function for the training of the perceptual loss as follows

$$\frac{1}{N}\sum_i \Big(\delta_{1-x_i^{(\mathbf{A})}} w (x_i^{(\mathbf{A})} - x_i^{(\mathbf{B})})^2 + (1-\delta_{1-x_i^{(\mathbf{A})}})(1-w)(x_i^{(\mathbf{A})} - x_i^{(\mathbf{B})})^2\Big), \tag{7.5}$$

where w is a balancing weight. Specifically, we choose w as the proportion of distances strictly inferior to 1 with

$$w = \min\left(\max\left(\frac{\sum_i 1-\delta_{1-x_i^{(\mathbf{A})}}}{N}, \beta\right), 1-\beta\right). \tag{7.6}$$

where β is a bounding factor such that w is bounded by $[\beta, 1-\beta]$. This formulation compensates for class imbalance while avoiding extreme weight values.

In that way, the loss function adapts the contributions from the voxels that are far from occupied voxels ($x_i^{(\mathbf{A})} = 1$) and voxels that are near occupied voxels ($x_i^{(\mathbf{A})} < 1$). We train the network with the Adam [102] optimizer.

Metric

As seen in Figure 7.2b, in order to compare two point clouds $x^{(\mathbf{A})}$ and $x^{(\mathbf{B})}$, we compute their respective latent spaces $f_a(x^{(\mathbf{A})}) = y^{(\mathbf{A})}$ and $f_a(x^{(\mathbf{B})}) = y^{(\mathbf{B})}$ using the previously trained analysis transform. These latent spaces each have F feature maps of size $W \times D \times H$ (width, depth, height). Then, we define the MSE between latent spaces as follows

$$\frac{1}{N}\sum_i (y_i^{(\mathbf{A})} - y_i^{(\mathbf{B})})^2. \tag{7.7}$$

We compute this MSE either over all F feature maps or on single feature maps.

7.2.3 . Point set metrics

Point-to-point (D1) and point-to-plane (D2)

The point-to-point distance (D1) [1] measures the average error between each point in $\mathbf{A}$ and their nearest neighbor in $\mathbf{B}$:

$$e^{D1}_{\mathbf{A},\mathbf{B}} = \frac{1}{N_{\mathbf{A}}}\sum_{\forall a_i \in \mathbf{A}} \|a_i - b_j\|_2^2 \tag{7.8}$$

where b_j is the nearest neighbor of a_i in $\mathbf{B}$.

In contrast to D1, the point-to-plane distance (D2) [171] projects the error vector along the normal and is expressed as follows

$$e^{D2}_{\mathbf{A},\mathbf{B}} = \frac{1}{N_{\mathbf{A}}}\sum_{\forall a_i \in \mathbf{A}} ((a_i - b_j)\cdot n_i)^2 \tag{7.9}$$

where b_j is the nearest neighbor of a_i in $\mathbf{B}$ and n_i is the normal vector at a_i.

The normals for original and distorted point clouds are computed with local quadric fittings using 9 nearest neighbors.

The D1 and D2 MSEs are the maximum of $e_{\mathbf{A},\mathbf{B}}$ and $e_{\mathbf{B},\mathbf{A}}$ and their Peak Signal-to-Noise Ratio (PSNR) is then computed with a peak error corresponding to three times the point cloud resolution as defined in [1].

7.3 . Experiments

7.3.1 . Experimental setup

We evaluate the metrics defined above on the ICIP2020 [139] subjective dataset. It contains 6 point clouds [116, 56] compressed using G-PCC Octree, G-PCC Trisoup and V-PCC with 5 different rates yielding a total of 96 stimuli (with 6 references) and their associated subjective scores. The two G-PCC approaches use different geometry codecs (octree and trisoup) but the same Prediction-plus-Lifting (i.e. Lifting) attribute codec.

For each metric, we compute the Pearson Linear Correlation Coefficient (PLCC), the Spearman Rank Order Correlation Coefficient (SROCC), the Root Mean Square Error (RMSE) and the Outlier Ratio (OR). We evaluate the statistical significance of the differences between PLCCs using the method in [200]. These metrics are computed after logistic fittings with cross-validation splits. Each split contains stimuli for one point cloud (i.e. reference point cloud and its distorted versions) as a test set and stimuli of all other point clouds as a training set. The metrics are then computed after concatenating results for the test set of each split. They are summarized in Table 7.1 and the values before and after logistic fitting are shown in Figure 7.3.

We use an upper bound value $u = 5$ when computing the TDF in Eq. (7.1) and a block size of 64 when block partitioning point clouds. The naBCE window size is $m = 5$ as in the original paper. The perceptual loss is trained with a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ on the ModelNet dataset [156] after block partitioning using Python 3.6.9 and TensorFlow [14] 1.15.0.

7.3.2 . Comparison of perceptual loss feature maps

In our experiments, we first considered the perceptual loss computed over all feature maps. However, we observed that some feature maps are more perceptually relevant than others. Consequently, we include the best feature maps for each voxel grid representation in our results. This corresponds to feature map 9 (TDF PL F9) for TDF PL and 1 (Bin PL F1) for Bin PL. Here, the best feature map is the feature map with the lowest RMSE value after logistic fitting.

Moreover, we observe that some feature maps are unused by the neural network

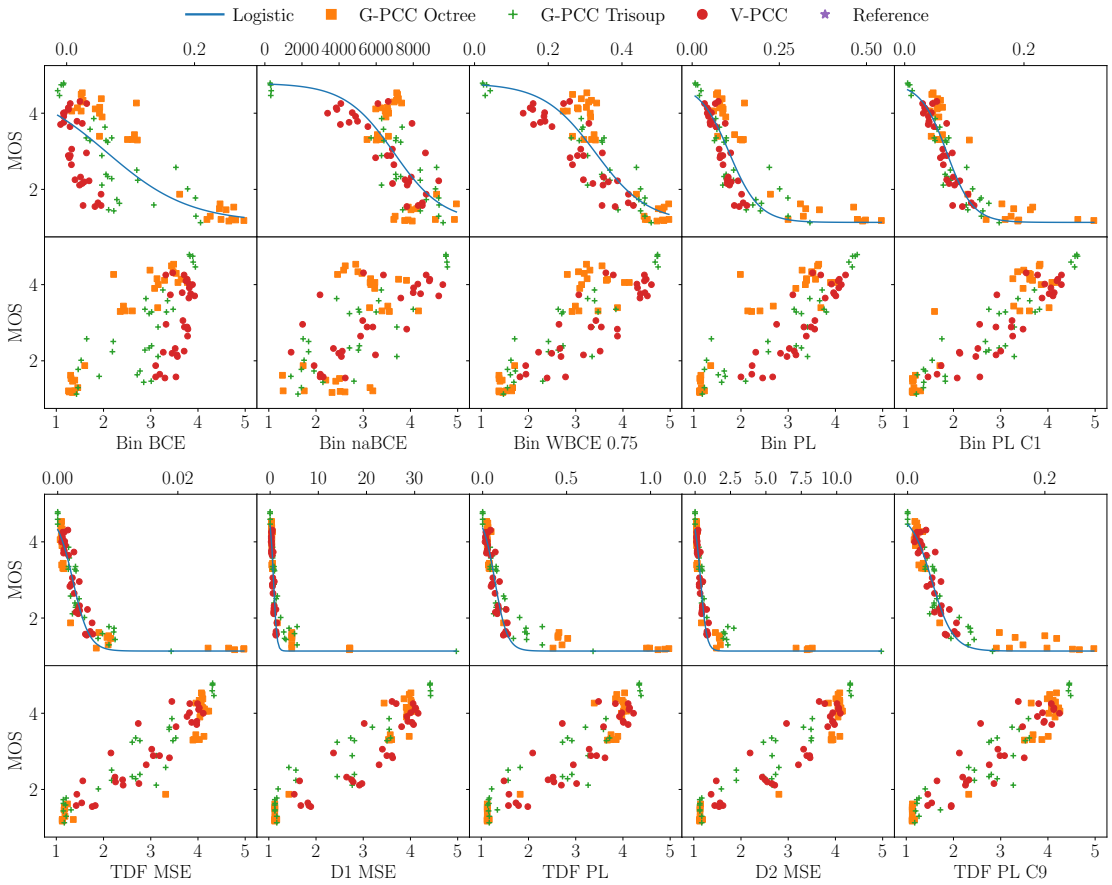


Figure 7.3: Scatter plots between the objective quality metrics and the MOS values. The plots before and after logistic fitting are shown.

Table 7.2: Statistical analysis of objective quality metrics.

Method	PLCC	SROCC	RMSE	OR
TDF PL F9	**0.951**	**0.947**	**0.094**	**0.375**
D2 MSE	0.946	0.943	0.100	0.469
TDF MSE	0.940	0.940	0.103	0.385
D1 MSE	0.938	0.933	0.109	0.479
TDF PL	0.935	0.933	0.110	0.490
Bin PL F1	0.922	0.916	0.115	0.406
D2 PSNR	0.900	0.898	0.129	0.500
Bin WBCE 0.75	0.875	0.859	0.144	0.531
Bin PL	0.863	0.867	0.151	0.552
D1 PSNR	0.850	0.867	0.158	0.448
Bin naBCE	0.740	0.719	0.201	0.573
Bin BCE	0.713	0.721	0.207	0.635

(constant). Therefore, they exhibit high RMSE values (all equal to 0.812) as their perceptual loss MSE are equal to 0. Specifically, we observe that TDF PL has 6 unused feature maps, while Bin PL has a single unused feature map. This suggests that the perceptual loss learns a sparser latent space representation when using TDF compared to binary. Thus, implicit representations may improve compression performance compared to explicit representations as fewer feature maps may be needed.

7.3.3 . Comparison of objective quality metrics

In Table 7.2, we observe that the TDF PL F9 is the best method overall. In particular, identifying the most perceptually relevant feature map and computing the MSE on this feature map provides a significant improvement. Specifically, the difference between the PLCCs of TDF PL F9 and TDF PL is statistically significant with a confidence of 95%.

For voxel grid metrics, we observe that TDF metrics perform better than binary metrics. In particular, the RMSEs of the former are noticeably lower for point clouds compressed with G-PCC Octree compared to the RMSE of the latter as can be seen in Table 7.3. This suggests that implicit representations may be better at dealing with density differences between point clouds in the context of point cloud quality assessment.

Table 7.3: Statistical analysis of objective quality metrics by compression method.

Method	G-PCC Octree			G-PCC Trisoup			V-PCC		
	PLCC	SROCC	RMSE	PLCC	SROCC	RMSE	PLCC	SROCC	RMSE
TDF PL F9	*0.975*	*0.859*	**0.078**	0.936	0.910	**0.101**	0.897	0.850	0.106
D2 MSE	0.962	0.829	0.094	**0.954**	**0.924**	*0.103*	*0.903*	0.860	*0.103*
TDF MSE	0.952	0.839	0.106	0.933	0.917	0.106	**0.912**	**0.867**	**0.098**
D1 MSE	**0.976**	0.851	*0.082*	*0.937*	*0.918*	0.126	0.876	0.844	0.119
TDF PL	0.970	0.840	0.087	0.918	0.900	0.127	0.876	0.837	0.115
Bin PL F1	0.941	0.786	0.138	0.927	0.907	0.107	0.898	*0.865*	0.109
D2 PSNR	0.943	**0.890**	0.110	0.926	0.895	0.108	0.738	0.723	0.166
Bin WBCE 0.75	0.923	0.747	0.163	0.918	0.886	0.112	0.850	0.786	0.164
Bin PL	0.931	0.852	0.186	0.892	0.886	0.130	0.880	0.852	0.142
D1 PSNR	0.903	0.859	0.156	0.910	0.895	0.117	0.599	0.689	0.202
Bin naBCE	0.552	0.357	0.277	0.846	0.786	0.154	0.748	0.692	0.170
Bin BCE	0.946	0.841	0.188	0.776	0.800	0.177	0.574	0.500	0.250

7.4 . Conclusion

We proposed a novel perceptual loss that outperforms existing objective quality metrics and is differentiable in the voxel grid. As a result, it can be used as a loss function in deep neural networks for point cloud compression and it is more correlated with perceived visual quality compared to traditional loss functions such as the BCE and the focal loss. Overall, metrics on the proposed implicit TDF representation performed better than explicit binary representation metrics. Additionally, we observed that the TDF representation yields sparser latent space representations compared to the binary representations. This suggests that switching from binary to the TDF representation may improve compression performance in addition to enabling the use of better loss functions.

The source code is available at https://github.com/mauriceqch/2021_pc_perceptual_loss.

The contents of this chapter have been published in M. Quach, A. Chetouani, G. Valenzise, and F. Dufaux, "A deep perceptual metric for 3D point clouds," *Electronic Imaging*, vol. 2021, no. 9, pp. 257–1–257–7, Jan. 2021.

8 - Convolutional Neural Network for Point Cloud Quality Assessment

8.1 . Introduction

In previous chapters, methods to handle point cloud sparsity and irregularity have been explored for geometry compression, attribute compression and quality assessment. In this chapter, we explore a convolution neural network approach with a patch extraction process for quality assessment.

As most multimedia contents, PCs may undergo different types of distortion introduced by several basic processing (acquisition, compression, transmission, visualization, etc.), usually applied to transmit or visualize such data [141, 142]. To estimate the perceptual impact of these distortions on the perceived quality, subjective and objective evaluations are usually conducted. Subjective evaluation gives scores that reflect the perception of human observers through psychovisual tests, while objective evaluation aims to automatically predict the subjective scores. As for 2D images and videos, objective methods can be classified according to the availability of the reference PC: Full Reference (FR) approaches that need the reference PC, Reduced Reference (RR) approaches that exploit only partial information from the reference PC and No Reference (NR) approaches that predict the quality from only the distorted version of the reference PC.

Different point cloud objective metrics have been proposed in the literature. Point-to-point metrics were among the first to be considered, and compute geometry distance between corresponding points in the original and distorted PC. On the other hand, the point-to-plane metric is an extension of the previous metric and consists in projecting the point-to-point error vector along the local normal [171]. Starting from these approaches, several point-based metrics have been then developed. In [21], the authors proposed to estimate the geometrical error between two PCs (i.e. reference PC and its distorted version) by measuring the angular similarity between tangent planes. In [124], the authors proposed a metric called PC-MSDM by extending the well-known SSIM metric [184], widely used for 2D images, to PC, by considering features including local mean curvature as they previously done for 3D meshes [109]. The authors proposed later a metric called PCQM that integrates color information [125]. In [94], the authors proposed a new approach that focuses more on the distribution of the data. They introduced a new type of correspondence from point to distribution characterized using the well-known Mahalanobis distance. In [179], the authors proposed a color-focused metric that integrates geometry information. In [20], the authors adapted also the

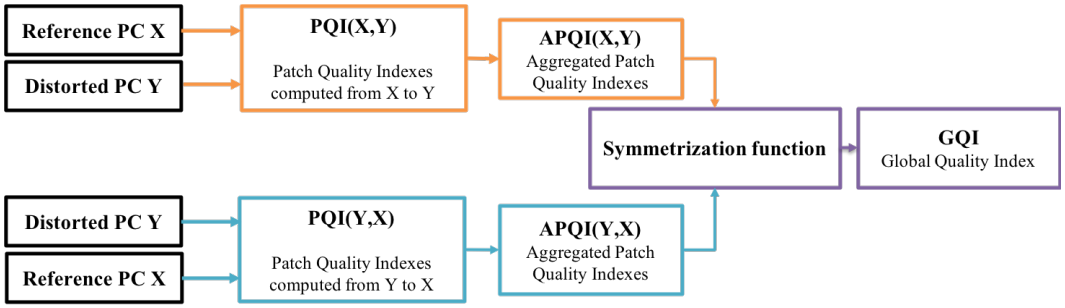


Figure 8.1: General framework of the proposed method.

SSIM metric for point clouds using a number of features, while in [93] the authors improved the point cloud PSNR metric. Interesting methods were also proposed for 3D meshes [15, 16, 40].

In this chapter, we present a *deep learning-based* method that efficiently predicts the quality of PCs. Our method extended our blind metric [44] and consists first of selecting a set of points from the reference PC and determining for each of them its nearest neighbor point in the distorted PC. We then define a patch of size 32×32 around each of the two points, each consisting of a point (i.e. selected point in the reference PC or its nearest neighbor point in the distorted) and its 1023 neighboring points. The structural information of each pair of patches is afterward compared by computing element-wise distances on three main attributes: geometry, curvature and color. The resulted patches (i.e. one per attribute) are stacked into a patch of size $32 \times 32 \times 3$ and fed as input to a Convolutional Neural Network (CNN) model to predict its quality. After applying the process in both directions (i.e. from the reference PC to its distorted PC and from the distorted PC to its reference PC) for each pair of corresponding patches, the global quality index is finally given by averaging the predicted patch quality scores and employing either a symmetrization function or a pooling strategy. The proposed method was evaluated using two datasets, including a large dataset more suited to deep models. The results obtained showed the relevance of using such deep learning-based approach for predicting the quality of PCs.

The remainder of this chapter is structured as follows: the proposed method is described in Section 8.2. Experiment results are discussed in Section 8.3, followed by the conclusion in Section 8.4.

8.2 . Proposed method

The method proposed in this chapter is based on the prediction of Patch Quality Indexes (PQIs) through CNN models and their symmetrization or pooling. Fig. 8.1 summarizes the general framework of our method. From the two PCs, we

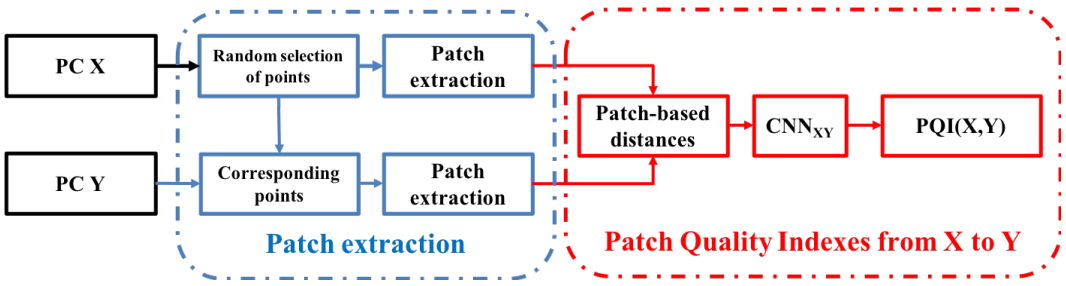


Figure 8.2: Patch Quality Index (PQI) prediction.

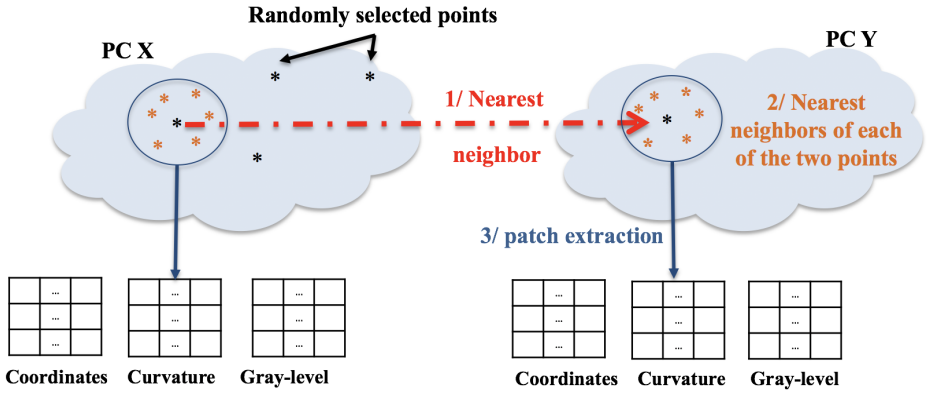


Figure 8.3: Patch extraction process.

first compute PQIs in both directions (i.e. from the reference PC to its distorted version: $\boldsymbol{PQI(X,Y)}$ as well as from the distorted PC to its reference version: $\boldsymbol{PQI(Y,X)}$) using a CNN model per direction. The Global Quality Index (GQI) is finally given by averaging the PQIs obtained and then either symmetrizing or pooling the resulted scores.

8.2.1 . Patch Quality Indexes

As illustrated by Fig. 8.2, PQIs are computed through several steps: patch extraction, patch-based distances computation and patch quality prediction using a CNN model. Each of these steps is described in this section. It is worth noting that here we limit our description to one direction (i.e. from the reference PC to its distorted version), since it remains similar in the second direction.

Patch extraction

In order to extract patches from the two PCs, we first select a set of points N from the reference PC $\boldsymbol{X}$. More precisely, 1000 points are randomly selected over

all points of X and are here considered as key points from which the patch quality indexes are predicted. For each selected point $\boldsymbol{N_i}$, we determine its corresponding point in the distorted PC $\boldsymbol{Y}$ by finding its nearest neighbor and define a region around each of the two points, delimited by the area covered by their 1023 neighboring points. The two regions are then reshaped into patches of size 32×32 based on the distance of each neighbor to the point, forming thus a pair of patches (see Fig. 8.3).

Patch-based distances

Element-wise patch-based distances are then computed to measure geometry, curvature and color distortions between patches of corresponding points. Let us first define $\boldsymbol{A_k}$ as the $\boldsymbol{k^{th}}$ considered attribute with $\boldsymbol{\{k = 1:}$ **Geometry;** $\boldsymbol{k = 2:}$ **Curvature;** $\boldsymbol{k = 3:}$ **Color**$\boldsymbol{\}}$.

For each attribute $\boldsymbol{A_k}$, the element-wise distance $\boldsymbol{D^n_{A_k}(i,j)}$ between the $\boldsymbol{n^{th}}$ pair of patches of the reference $\boldsymbol{X}$ ($\boldsymbol{P^n_X}$) and the distorted PCs $\boldsymbol{Y}$ ($\boldsymbol{P^n_Y}$) is computed as follows:

$$\mathbf{D^n_{A_k}(i,j)} = \sqrt{\left(\mathbf{P^n_{X_k}(i,j)} - \mathbf{P^n_{Y_k}(i,j)}\right)^2}, \tag{8.1}$$

where $\boldsymbol{P^n_{X_k}(i,j)}$ and $\boldsymbol{P^n_{Y_k}(i,j)}$ represent the value of the $\boldsymbol{k^{th}}$ attribute at position $\boldsymbol{(i,j)}$ of the pair of patches $\boldsymbol{n}$ in the reference and distorted PCs, respectively.

The resulting patches are then stacked into a patch of size $32 \times 32 \times 3$ where 32×32 represents the initial size of the extracted pair of patches and 3 is the number of considered attributes (i.e. 1: geometry distances, 2: curvature distances and 3: color distances). It is worth noting that the size of the patches was fixed according to several studies dedicated for 2D images where the impact of the patch size was analyzed [97, 31]. The authors concluded that a size of 32×32 constituted a good trade-off between performance and computation time. We opted here for the same size. However, more in-depth analysis has to be carried-out in order to better analyze its impact on the performance.

Architecture of the CNN models

Finally, the PQIs are predicted using a CNN model which has as input the normalized stacked patches. A plethora of models have been proposed in the literature, starting from one of the first models such as AlexNet [107] to the more complex like ResNet [82] that introduced a residual module or inception [167] that employed parallel layers. In this study, we adopted a pre-trained VGG model [161] that was widely used for classification tasks and was successfully employed in several studies related to quality assessment as well [57].

This model was developed by the Oxford Visual Geometry Group in 2014. It uses small filters of size 3×3 in contrast to AlexNet [107] where 11×11 and 5×5 filters are used in the first two layers. It is also characterized by applying a max pooling layer after a succession of convolution layers. Various versions with different depths have been proposed. Here, we used $VGG16$ which is composed of 13 convolutional layers, followed by 3 fully connected layers. It initially takes an image of size $224 \times 224 \times 3$ as input and highlights one class among 1000 classes (i.e. output of size 1000).

To adapt the model to our context, several modifications were realized. More precisely, we replaced its input layer by another of size $32 \times 32 \times 3$ and substituted its fully connected layers with 3 other layers of size 64, 64 and 1, respectively. The first fully connected layer is followed by a ReLu layer, while the second one is followed by a ReLu layer and a dropout layer with a probability fixed to 0.5. The third and last layer is a regression layer that aims to predict the PQIs of the normalized stacked patches. The model thus modified was finally fine-tuned to adapt its weights to our context. It is worth noting that two similar CNN models with the architecture described above are used, each dedicated to a specific direction: $\boldsymbol{CNN_{XY}}$ (from the reference PC to the distorted one) and $\boldsymbol{CNN_{YX}}$ (from the distorted PC to the reference one).

To train our models, we used the stochastic gradient descent optimization algorithm and the Mean Squared Error (MSE) as loss function. The learning rate and the momentum were set to 0.001 and 0.9, respectively. The batch size was fixed to 64 and the training data was shuffled every epoch. The number of epochs was set to 100 with a validation frequency fixed to 5000 iterations. At each epoch, the model was saved and the one that provides the best results was retained. It is worth noting that the target of each stacked patch was the subjective score of the whole distorted PC as commonly used to estimate the quality of several multimedia content [97, 123].

8.2.2 . Global Quality Index

The Global Quality Index (GQI) that reflects the quality of the whole distorted PC is finally given by aggregating the PQIs achieved for each direction and applying either a symmetrization function or a pooling method.

Aggregated Patch Quality Indexes

As mentioned above, each CNN model predicts the quality score of each stacked patch. In order to derive a single quality index for each direction, we aggregate the obtained PQIs by computing the average scores as follows:

$$\mathbf{APQI}(\mathbf{X},\mathbf{Y}) = \frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{N}}\mathbf{PQI_i}(\mathbf{X},\mathbf{Y}), \tag{8.2}$$

and

$$\mathbf{APQI}(\mathbf{Y},\mathbf{X}) = \frac{\mathbf{1}}{\mathbf{N}}\sum_{\mathbf{i=1}}^{\mathbf{N}}\mathbf{PQI_i}(\mathbf{Y},\mathbf{X}), \tag{8.3}$$

where $\boldsymbol{APQI(X,Y)}$ and $\boldsymbol{APQI(Y,X)}$ are the Aggregated Patch Quality Indexes (APQIs) that represent the predicted quality score in each direction. $\boldsymbol{N}$ is the number of stacked patches and it was fixed to 1000.

It is worth noting that this strategy is commonly applied in image quality assessment domain for both handcrafted and deep learning-based methods. Indeed, handcrafted metrics like SSIM [184] or VDP [51] derive a single quality score from the predicted visibility map by aggregating the local scores. Whereas, the initial deep learning-based method [97] and most of those developed later [31, 41, 39] predict the quality of patches and derive a single quality score by averaging the predicted scores as well. For 3D contents, the same strategy was applied for 3D PC [124] and 3D meshes [109, 17] as well.

Symmetrization or Pooling

As most of the existing full reference PC quality metrics [171, 93, 94], the final quality index is computed in two directions (i.e. from the reference PC to the distorted one and from the distorted PC to the reference one) where each reflects a specific aspect. The two indexes are then usually symmetrized by applying a function f as follows:

$$\mathbf{GQI}(\mathbf{X},\mathbf{Y}) = \mathbf{f}(\mathbf{APQI}(\mathbf{X},\mathbf{Y}),\mathbf{APQI}(\mathbf{Y},\mathbf{X})), \tag{8.4}$$

where $\boldsymbol{GQI(X,Y)}$ is the Global Quality Index computed between the reference PC $\boldsymbol{X}$ and its distorted version $\boldsymbol{Y}$ in both directions.

In this study, we considered three classical symmetrization functions: $\boldsymbol{min}$, $\boldsymbol{mean}$ and $\boldsymbol{weighted\ mean}$ ($\boldsymbol{Wmean}$). It is worth noting that the $\boldsymbol{max}$ was not applied since the considered datasets provide the MOS (Mean Opinion Score) as subjective score. The results are compared to those obtained by pooling the indexes through a Support Vector Machine (SVR) and a Multi Layer Perceptron (MLP). More precisely, the SVR model had a Gaussian function as kernel and the MLP model was composed of 2 fully connected layers of size 30, followed by a regression layer of size 1. Both models were trained and tested by applying the protocol described in Section 8.3.2 and the results are shown in Section 8.3.3.

8.3 . Experimental results

Our method is evaluated in terms of correlation with subjective judgments. To do so, we first present the datasets used and describe the protocol applied to train and test our CNN models. After defining the performance evaluation criteria employed, we analyze the results obtained by either applying symmetrization functions or pooling strategies. Finally, the performance of our method is compared with state-of-the-art PC metrics and a cross-dataset evaluation is carried-out to show the generalization ability of our method to predict the quality of unknown PCs.

8.3.1 . Datasets

The proposed method is evaluated using two recent 3D point cloud datasets: sjtu [191] and ICIP20 [139].

- **sjtu** is composed of 9 point clouds from which 378 degraded versions were derived (i.e. 42 distorted PC per reference PC) through 6 degradation types (OT: Octree-based compression, CN: Color Noise, DS: Downscaling, D+C: Downscaling and Color noise, D+G: Downscaling and Geometry Gaussian noise, GGN: Geometry Gaussian noise and, C+G: Color noise and Geometry Gaussian noise).

- **ICIP20** is composed of 6 common used point clouds from which 90 degraded versions were derived (i.e. 15 distorted PC per reference PC) through 3 types of compression: V-PCC, G-PCC with triangle soup coding and G-PCC with octree coding. Each reference point cloud was compressed using five different levels.

These datasets give a subjective score (i.e. MOS) for each distorted PC that are used to train the CNN models and evaluate the performance of our method to predict the global quality score.

8.3.2 . Protocol and performance evaluation

To assess the quality prediction effectiveness of our method, each dataset is split into training and test sets F times (i.e F folds). Each fold is composed of a reference PC and its distorted versions. Therefore, the training and test sets do not contain same PC (i.e. no overlap between the two). At each time, $F-1$ folds are used to train the models and the rest to test it. In this study, F is equal to 7 and 6 for sjtu and ICIP20, respectively.

Two evaluation criteria commonly used to evaluate the performance of quality metrics are adopted here: 1) Pearson Linear Correlation Coefficient (PLCC) and 2) Spearman Rank-Order Coefficient Correlation (SROCC). Both vary between

0 and 1 (i.e. absolute value) with 1 the best performance. These correlations are computed for each dataset over each fold and the mean correlations are then reported as results. It is worth noting that the same procedure was applied to the compared state-of-the-art metrics.

8.3.3 . Performance analysis

In this section, we analyze the correlations obtained by each CNN model as well as their combination by either a symmetrization function (i.e. *min*, *mean* and $Wmean$) or using an SVR and a MLP. Table 8.1 shows the results obtained for each model and those obtained after symmetrization and pooling on sjtu dataset. Best results are highlighted in bold.

Method	**PLCC**	**SROCC**
Aggregated Patch Quality Index		
APQI(X,Y)	0.902	0.901
APQI(Y,X)	0.908	0.900
Symmetrization function		
GQI(X,Y)=min(APQI(X,Y),APQI(X,Y))	0.908	0.900
GQI(X,Y)=mean(APQI(AB),APQI(X,Y))	0.908	0.902
GQI(X,Y)=Wmean(APQI(X,Y),APQI(X,Y))	0.909	0.904
Pooling (machine learning)		
GQI(X,Y)=SVR(APQI(X,Y),APQI(X,Y))	0.918	**0.907**
GQI(X,Y)=MLP(APQI(X,Y),APQI(X,Y))	**0.927**	0.906

Table 8.1: Mean correlations obtained on sjtu dataset before and after symmetrization and pooling. Best results are highlighted in bold.

As can be seen, the aggregated indexes (i.e. APQI) given by computing the quality scores in both directions obtain close correlations. Their symmetrization through the three functions does not increase the global performance. Whereas, the pooling strategies applied improve the performance, particularly the linear correlations (i.e. PLCC) without a significant increase in terms of mean SROCC. Both SVR and MLP models achieve close mean SROCC with a higher mean PLCC for MLP. The improvement gains in terms mean PLCC are about 1.8% and 2.8% for SVR and MLP, respectively. Based on these results, it seems that the symmetrization function, at least, does not always provide the optimal results and thus should be carefully employed.

8.3.4 . Comparison with the state-of-the-art

Our method is here compared with point-based state-of-the-art metrics. More precisely, we consider po2pointMSE and po2planeMSE metrics that are pooled with MSE, PSNRpo2pointMSE and PSNRpo2planeMSE metrics that are pooled with PSNR as well as recent metrics po2dist [94] (i.e. point to distribution) pooled with MSE and PSNR. We also compare our method with a recent metric, called PCQM, which is based on both geometry and color features [125]. Tables 8.2 and 8.3 show the results obtained for sjtu and ICIP20 datasets, respectively. The best results are highlighted in bold.

Method	PLCC	SROCC
po2pointMSE	0.686	0.801
PSNRpo2pointMSE	0.799	0.844
po2pointHausdorff	0.517	0.686
PSNRpo2pointHausdorff	0.638	0.682
po2planeMSE	0.642	0.717
PSNRpo2planeMSE	0.744	0.722
po2planeHausdorff	0.539	0.682
PSNRpo2planeHausdorff	0.755	0.825
po2distMSE (mmd)	0.710	0.603
PSNRpo2distMSE (mmd)	0.621	0.603
po2distMSE (msmd)	0.706	0.603
PSNRpo2distMSE (msmd)	0.642	0.715
PCQM	0.879	0.888
GQI (our method)	**0.927**	**0.906**

Table 8.2: Comparison with state-of-the-art methods on sjtu dataset. Bests results are highlighted in bold.

On sjtu (see Table 8.2), the proposed method performs the best with a performance gain in terms of mean PLCC between 5% (compared to PCQM) and 79% (compared to po2pointHausdorff). Among the state-of-the-art metrics, PCQM achieves the best correlations with 0.879 and 0.888 as mean PLCC and SROCC, respectively, far followed by PSNRpo2pointMSE and PSNRpo2planeMSE. All the compared metrics obtained a mean PLCC and SROCC lower than 0.88 and 0.89,

respectively. The worst result are obtained by po2pointHausdorff and po2planeHausdorff. The low correlations achieved by these metrics, except PCQM, can be explained by the fact that those metrics focus only on geometric aspects and therefore totally fail to capture other distortions like color noise.

Method	PLCC	SROCC
po2pointMSE	0.945	0.950
PSNRpo2pointMSE	0.880	0.934
po2pointHausdorff	0.717	0.690
PSNRpo2pointHausdorff	0.597	0.763
po2planeMSE	0.945	0.959
PSNRpo2planeMSE	0.916	0.953
po2planeHausdorff	0.753	0.763
PSNRpo2planeHausdorff	0.939	**0.970**
po2distMSE (mmd)	**0.965**	0.963
PSNRpo2distMSE (mmd)	0.865	0.965
po2distMSE (msmd)	**0.967**	0.965
PSNRpo2distMSE (msmd)	0.902	**0.972**
PCQM	0.796	0.832
GQI (our method)	**0.961**	**0.966**

Table 8.3: Comparison with state-of-the-art methods on ICIP20 dataset. Best results are highlighted in bold.

On ICIP20 (see Table 8.3), po2distMSE (msmd) obtains the best mean PLCC (0.967), closely followed by po2distMSE (mmd) (0.965). Whereas the two best mean SROCC is reached by PSNRpo2distMSE (0.972) and PSNRpo2planeHausdorff (0.970), respectively. Our metric obtains competitive results and surpasses most of the compared methods. PCQM is outperformed by most of the compared metrics, except po2point-based metrics pooled with Hausdorff (i.e. po2pointHausdorff and PSNRpo2pointHausdorff) and po2planeHausdorff. Similarly to the results obtained on sjtu, po2point-based and po2plane-based metrics pooled with MSE obtain higher correlations than those pooled with PSNR, while the po2dist-based metrics pooled with PSNR give better results than those pooled with MSE. The

high correlations obtain by the point-based metrics, especially po2pointMSE and po2planeMSE, can be explained by the fact that contrary to sjtu, this dataset contains only compressed PCs with joint distortion of geometry and attributes.

We also evaluate the impact of the random selection step on the performance by repeating the selection process 5 times on sjtu dataset. To this end, we computed the p-values and F-values between the correlations obtained over the 5 iterations using the One-way analysis of variance (ANOVA) test. A p-value close to 1 indicates that the difference is not high, while a F-value close to 0 indicates that the means and the standard deviations are similar. Table 8.4 shows the obtained values. As can be seen, all the p-value are higher than the significance level (i.e. 0.05), indicating that there is no statistically significant difference between them. Whereas the F-values are close to 0, indicating that the correlation distributions are essentially identical.

Iteration	**1**	**2**	**3**	**4**	**5**
1	**1.00/0.00**	0.83/0.05	0.99/0.00	0.88/0.02	0.89/0.02
2	0.83/0.05	**1.00/0.00**	0.83/0.05	0.94/0.01	0.93/0.01
3	0.99/0.00	0.83/0.05	**1.00/0.00**	0.88/0.02	0.89/0.02
4	0.88/0.02	0.94/0.01	0.88/0.02	**1.00/0.00**	0.99/0.00
5	0.89/0.02	0.93/0.01	0.89/0.02	0.99/0.00	**1.00/0.00**

Table 8.4: p-values and F-values between the correlations obtained over the 5 random selection (p-value/F-score).

8.3.5 . Cross Dataset Evaluation

In this section, we evaluate the generalization ability of our method to predict the quality of unknown PCs using sjtu as training/validation set and ICIP20 as test set without overlap between both. More precisely, sjtu was split into two sets: 80% for the training and 20% for the validation. We obtain high correlations with 0.889 as mean PLCC and 0.930 as mean SROCC. These results can be explained by the fact that ICIP20 is composed of only compressed PCs using G-PCC (octree and trisoup) and V-PCC, while sjtu contains compressed PCs using octree as well as PCs with several other distortions, including color noise. In other words, there is a data overlap in terms of distortion types between both datasets, which enables the trained models to well predict the quality. In addition, these results show that using such deep learning-based method allows to predict well the quality of unknown PCs with close unknown distortions. Indeed, training the considered CNN models on sjtu that contains PCs compressed with octree-based compression,

even allows to well predict the quality of trisoup-based compressed PCs as well as those compressed with V-PCC. Nevertheless, due to the relatively wide variety of distortions contained in sjtu, training those CNN models on ICIP20 does not provide high correlations (~ 0.5). The latter result is related to the generalization ability of those deep learning models which is one of the challenging issues that remains open.

8.4 . Conclusion and perspectives

In this chapter, we proposed a *deep learning-based* method that efficiently predicts the quality of distorted PCs with reference. We first randomly selecting a set of points from the reference PC and its distorted version and, determined the nearest neighbor of each of them. We then defined a region of size 32×32 for each of the two points and the structural distortions are computed through element-wise patch-based distances by considering three attributes: geometry, curvature and color. The resulted patches are then stacked into a patch of size $32 \times 32 \times 3$ and fed as input to CNN models to predict its quality (i.e. one CNN model per direction). The global quality index is finally given by aggregating the predicted patch quality indexes and applying either a symmetrization function or a pooling strategy using machine learning tools (SVR or MLP). The best result was provided by MLP and the use of symmetrization functions did not improve the performance. The results are compared with state-of-the-art methods and a cross-dataset evaluation was carried-out to show the generalization ability of our method to predict the quality of unknown PCs. The proposed method obtained promising results on two datasets and showed a good ability to predict unknown PCs with close unknown distortions. Despite the effectiveness of our method, some points should be deeper analyzed, including the use of more efficient deep learning models, the impact on the performance of the number of selected points as well as the patch size.

The contents of this chapter have been published in A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Convolutional Neural Network for 3D Point Cloud Quality Assessment with Reference," in *IEEE International Workshop on Multimedia Signal Processing (MMSP'2021)*, Oct. 2021.

9 - Conclusion and perspectives

9.1 . Conclusion

Point clouds are increasingly common, with numerous applications and the large amounts of data are a challenge for transmission and storage. Compression is thus crucial but also complex as point cloud compression lies at the intersection of signal processing, data compression, computer graphics and even deep learning for state of the art approaches. The signal characteristics make point cloud compression significantly different from traditionally compressed media such as images and video. Indeed, the geometry can be seen as a sparse binary signal on a voxel grid or as a set of points and the attributes then lie on this sparse, irregular geometry posing challenges for point cloud compression.

In recent works, deep learning based approaches have exhibited outstanding performance on geometry compression and have become comparable to state of the art approaches on attribute compression. However, many challenges remain and exchange of ideas between traditional and deep learning based methods have led to fast progress in the field. In this work, we put a focus on deep learning approaches and highlight their relation to traditional methods. Specifically, we highlight different categories of geometry and attribute compression approaches with relations between deep learning and traditional approaches. We discuss the importance of LoD decomposition for compression, the limitation of separating geometry and attribute compression and the importance of rendering in the context of compression. Then, we discuss the specific challenges posed by sparsity for geometry compression, the importance of point cloud quality assessment and how its challenges mirror those of compression. Finally, we discuss how point cloud compression relates to mesh compression and point out their intersection.

We note that approaches for compressing point cloud geometry have improved significantly by exploiting the duality between octree and voxel grid and the LoD decomposition resulting from octrees. Currently, deep learning approaches using sparse convolutions offer state of the art performance and competitive complexity compared to G-PCC. However, the characteristics of point cloud geometry can vary significantly resulting in numerous parameters that need to be tuned for a codec to work uniformly well across various point clouds. We believe that geometry compression approaches need to combine both hand-crafted and learning based approaches in order to achieve compression performance, reasonable complexity and adaptability to different point cloud characteristics (e.g. density).

In comparison, learning based attribute compression approaches has been the subject of less research compared to geometry compression. Current learning based

approaches struggle to compete the state of the art and often come at high complexity cost. We believe that research into the different methods to construct regular or almost regular neighborhoods for the attributes is essential for learning based attribute compression. For example, using an octree provides a hierarchical regular neighborhood (one parent for each octree node) and mapping the attributes to 2D grid can provide an almost regular neighborhood. The attributes are spatially correlated and this can be verified visually. As such, a key question is how can learning based approaches best exploit this spatial correlation for compression? Using which representation?

Overall, deep learning based approaches need to account for point cloud sparsity and irregularity for both geometry and attribute compression tasks. And learning based point cloud quality assessment is also met with the same challenge. To this end, our contributions are as follows:

- A lossy point cloud geometry compression method that casts decoding as a classification problem along with an extensive study on the key performance factors of such methods.

- A lossless point cloud geometry compression method using a generative model.

- A multiscale lossless point cloud geometry compression method which resolves the complexity issues of the previous approach.

- A lossy attribute compression method that folds the geometry in order to map the attributes onto a 2D grid. The problem is then reduced to an image compression problem.

- A deep perceptual point cloud quality metric which is differentiable and suitable as a loss function to train lossy point cloud geometry compression methods.

- A Convolutional Neural Network for point cloud quality assessment that makes use of projections to a 2D grid in order to learn an objective quality metric that correlates well with perceived visual quality.

- An extensive state of the art on Point Cloud Compression that compares both traditional and learning based methods. We emphasize the different and common methods that are used to deal with point cloud sparsity and irregularity and highlight the common patterns.

9.2 . Perspectives

9.2.1 . Joint compression of geometry and attributes in relation with rendering

We provide a general overview of the surroundings of point cloud compression in Figure 9.1. We define the notion of 3D scene in a general manner as including real world 3D scenes, meshes, voxels etc. A 3D scene can be modelled (or acquired, sampled...) into a point cloud and prior to rendering, point clouds must be interpreted as they are only sets of coordinates. Common interpretations of a point include screen-aligned squares with a given window-space size ("point" primitive), cubes, spheres etc. In addition, surface reconstruction can also be performed to build a mesh from a point cloud. Note that point clouds are only one way to compress 3D scenes and their visualizations.

As shown in Figure 2.1, separating geometry and attributes compression simplifies point cloud compression by dividing it into two independent components. However, jointly compressing both geometry and attributes may be important when considering the problem more globally as illustrated in Figure 9.1. Indeed, for numerous applications (visualization, machine vision) geometry and attributes may impact performance in an interdependent manner. In Computer Graphics, the well-known bump mapping technique simulates wrinkled surfaces with a simplified geometry by adding perturbations to the attributes [30]. In the case of meshes, perturbations are added to the surface normals which results in an apparently wrinkled surface due to lighting calculations. For point clouds, the same could be achieved by perturbing the point colors. This is closely related to the problem of finding an ideal rate balance between geometry and attributes (in the case they are considered separately).

As a result, an open research question is how to design compression techniques that compress point cloud geometry and attributes while maximizing perceived visual quality subject to a rate constraint? This differs from existing approaches, which take geometry and attributes into account separately and evaluate their quality separately.

9.2.2 . LoD decomposition and rendering: points and voxels

We have previously treated the topic of rendering and its importance to point cloud compression and quality assessment. Also, we have analyzed the topic of point cloud compression with approaches based on LoD decompositions. Decompositions can be interpreted in two manners: point based or volumetric.

In a point based view, the decomposition can be interpreted as as decomposing the point coordinates. With an octree decomposition, we decompose the coordinates in base 2; then, the octree can be interpreted as a prefix tree [32] on the Morton codes [127] of the coordinates.

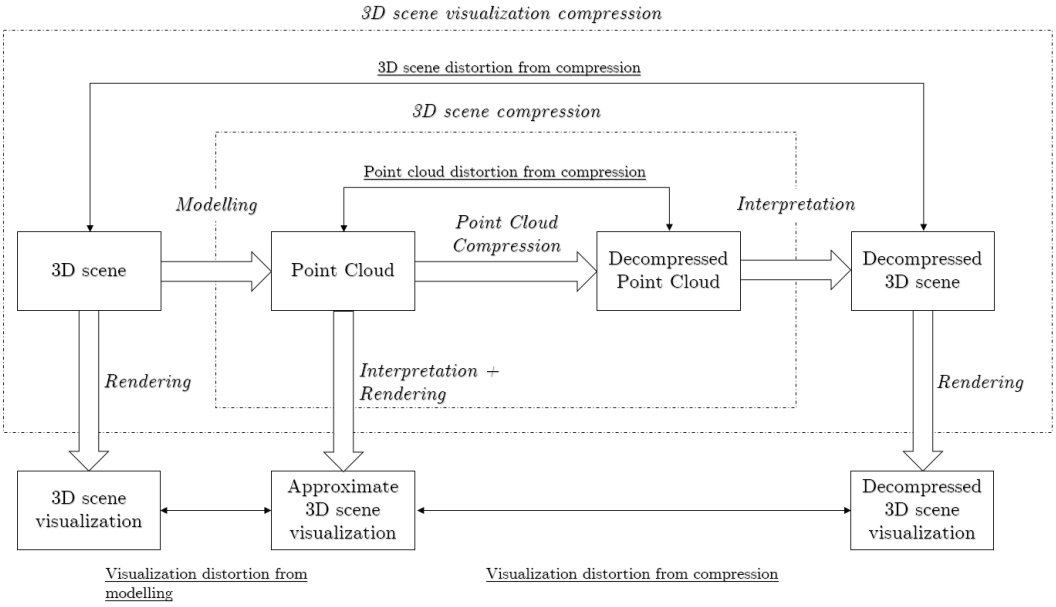


Figure 9.1: Different types of distortions when compressing 3D scenes (real world, mesh, point clouds, voxels) as point clouds. Depending on the context, different types of distortion may be evaluated: point cloud distortion, 3D scene distortion (e.g. mesh distortion) or visualization distortion. For visualization distortion, rendering is especially important and may be optimized as aprt of the point cloud codec.

On the other hand, we can interpret this decomposition in a volumetric manner. A voxel is occupied if it contains at least one point within its volume; otherwise, it is empty. When the point cloud is represented at a lower LoD, the voxels become larger. With a point based interpretation, a problem of octree based compression is that the point cloud becomes sparser at lower LoDs. The volumetric interpretation offers an alternative in that the geometry becomes coarser at lower LoDs but not sparser. As a result, octree based compression with a volumetric interpretation does not result in sparse renderings. A possible implementation is to render each voxel as a cube of corresponding volume.

Such coarse but dense geometry for point cloud compression is interesting as the bounded geometric error allows preservation of watertightness which is beneficial for rendering quality. How to best exploit such characteristics of compression algorithms for rendering remains an open question and is currently unexplored in the literature.

9.2.3 . Sparsity for geometry compression

One of the key problems with point cloud compression is sparsity. Sparsity can be defined as the ratio between the number of captured points and the number of voxels which is directly related to the geometry precision, also defined as geometry bitdepth. The capturing process is the main driver behind sparsity: point clouds captured with camera arrays are typically dense because the sampling density is spatially uniform. On the other hand, LiDAR point clouds have non-uniform density: denser near the sensor and sparser far from the sensor. However, they exhibit near uniform density in the spherical coordinate system which explains non-uniformity in the 3D space.

In such point clouds, the number of points can be low compared to the precision of the point coordinates. To obtain a dense point cloud, it is necessary to use a geometry precision that is suitable to the number of captured points. This is an important consideration as some compression methods may perform better at some density/sparsity levels. For example, deep learning based convolutional geometry compression approaches tend to perform better on dense point clouds.

Decomposing point clouds into LoDs is also useful to analyze point cloud sparsity. In particular, the location of sparse LoDs is important to differentiate global from local sparsity. Global sparsity is typically handled by performing block partitioning which removes global sparsity by focusing compression algorithms onto occupied blocks. Handling local sparsity is currently an open problem. One approach is to remove local sparsity by removing the sparse local LoDs resulting in a dense point cloud which is easier to compress. Another approach is to compress these sparse LoDs separately by first using a lossless compression methods on the first LoDs and another compression method suitable for sparse point clouds on the later LoDs.

Also, we have observed that on local sparse LoDs, existing methods are not able to exceed 2 dB per bit per input point per LoD (D1 PSNR) which is equivalent to the RD performance of a scalar quantizer. This suggests that sparse local LoDs may not be compressible beyond this limit in some cases. Whether these local sparse LoDs are akin to acquisition noise and are not compressible beyond this limit remains an open question. If this is true, compressing dense LoDs and these sparse LoDs separately may be essential to sparse point cloud compression.

9.2.4 . Point cloud quality assessment

Commonly used distortion measures include the point-to-point (D1), the point-to-plane metric (D2) and other metrics based on nearest neighbors [1]. These distortion metrics may not be suitable for all applications such as LiDAR point clouds for autonomous driving which may require application-specific distortion metrics. For instance, Feng et al. [61] evaluate the performance of their LiDAR point cloud compression method with registration translation error, object detection accuracy and segmentation error. Such metrics may be more relevant than generic metrics as these are common operations on such point clouds.

In addition, interpreting LiDAR point clouds in a volumetric manner could potentially improve safety. Indeed, with volumetric octree compression point clouds could be compressed with only false positives. This could be useful for collision avoidance in the context of autonomous driving. As a result, a distorted point cloud with distance-bounded distortion and only false positives could actually improve collision avoidance. However, if such a distortion is too severe, an autonomous vehicle could fail to find a valid route to its destination by overestimating the size of obstacles.

Another related factor is the accuracy and the precision of the LiDAR sensor. When considering LiDAR point clouds in a voxelized manner, we may want to consider a certain space around each occupied voxel as occupied to account for sensor uncertainty. This may be interesting for collision avoidance as marking a safety margin as occupied would also make the point cloud denser and thus easier to compress. Overall, how to design an objective metric that correlates best with autonomous driving performance remains an open question.

Furthermore, deep learning based point cloud quality assessment approaches are being explored [43, 44, 115]. We note that it presents the same challenges as geometry and attribute compression simultaneously. We consider the following open question: how to design an objective metric that correlates best with perceived visual quality? We notice that rendering must be considered as the point cloud cannot be viewed otherwise. Generally, it is important to consider what is the distortion of interest: point cloud distortion, 3D scene distortion, or distortion of the 3D scene visualization (Figure 9.1)?

9.2.5 . Relation to mesh compression

Meshes can be constructed from point clouds using Poisson surface reconstruction [100] and point clouds can be sampled from meshes. Surfaces can be represented explicitly with polygons but also implicitly as Truncated Signed Distance Functions (TSDFs). As such, meshes can be compressed under a TSDF representation and deep learning based convolutional compression techniques for point clouds [141] and for meshes [168] are strongly related. The key difference lies in the signal representation which can be binary (point clouds) or a distance field (point cloud or surface). Surface reconstruction from TSDFs can be performed with methods such as Marching Cubes [117]. It remains an open question how exchange of ideas between techniques of point cloud and mesh compression could further progress in both fields.

9.3 . Publications

Over the course of the thesis, the following conference papers

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

- M. Quach, G. Valenzise, and F. Dufaux, "Folding-Based Compression Of Point Cloud Attributes," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 3309–3313

- M. Quach, G. Valenzise, and F. Dufaux, "Improved Deep Point Cloud Geometry Compression," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2020, pp. 1–6

- M. Quach, A. Chetouani, G. Valenzise, and F. Dufaux, "A deep perceptual metric for 3D point clouds," *Electronic Imaging*, vol. 2021, no. 9, pp. 257–1–257–7, Jan. 2021

- M. Quach, G. Valenzise, and F. Dufaux, "A Deep Point Cloud Geometry Coding Toolbox," in *2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2021, pp. 1–2

- D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Learning-Based Lossless Compression of 3D Point Cloud Geometry," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 4220–4224

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

- A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Deep Learning-Based Quality Assessment Of 3d Point Clouds Without Reference," in *2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2021, pp. 1–6

- A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Convolutional Neural Network for 3D Point Cloud Quality Assessment with Reference," in *IEEE International Workshop on Multimedia Signal Processing (MMSP'2021)*, Oct. 2021

- A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Combination of Deep Learning-based and Handcrafted Features for Blind Image Quality Assessment," in *9th European Workshop on Visual Information Processing (EUVIP 2021)*, Paris (virtual), France, Jun. 2021

- G. Valenzise, M. Quach, D.-T. Nguyen, and F. Dufaux, "Voxel-based Deep Point Cloud Geometry Compression," in *CORESA (COmpression et REprésentation Des Signaux Audiovisuels)*, Sophia-Antipolis, France, Nov. 2021

- A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "LEARNING-BASED 3D POINT CLOUD QUALITY ASSESSMENT USING A SUPPORT VECTOR REGRESSOR," in *Image Quality and System Performance, IS&T International Symposium on Electronic Imaging (EI 2022)*, San Francisco, United States, Jan. 2022

and the following journal papers

- 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, Dec. 2021

- M. Quach, J. Pang, D. Tian, G. Valenzise, and F. Dufaux, "Survey on Deep Learning-Based Point Cloud Compression," *Frontiers in Signal Processing*, vol. 2, 2022

have been published. In addition, the following patent disclosures have been filed

- M. Quach (CentraleSupelec), J. Pang, M.A. Lodhi, D. Tian (InterDigital, US), G. Valenzise, F. Dufaux (CentraleSupelec), "State Summarization for Binary Voxel Grid Coding", filed on Nov. 4, 2021

- J. Pang, D. Tian (InterDigital, US), M. Quach, G. Valenzise, F. Dufaux (CentraleSupelec), "Outlier Grouping based Point Cloud Compression", filed on Nov. 22, 2021

- J. Pang, D. Tian (InterDigital, US), M. Quach, G. Valenzise, F. Dufaux (CentraleSupelec), "Learning-based PCC via Unfolding of 3D Point Clouds", filed on Jun. 22, 2021

- D. Tian, J. Pang (InterDigital, US), M. Quach, G. Valenzise, F. Dufaux (CentraleSupelec), "Learning-based Point Cloud Compression via Tearing-Transform", filed on Apr. 29, 2021

A - Synthèse

A.1 . Contexte

Les avancées récentes ont augmenté la précision et la disponibilité des technologies de capture 3D, faisant des nuages de points une structure de données essentielle pour la transmission et le stockage des données 3D. Les nuages de points 3D sont cruciaux pour un large éventail d'applications telles que la réalité virtuelle [33], la réalité mixte [63], la conduite autonome [194], la construction [183], le patrimoine culturel [172], etc. Dans de telles applications, les nuages de points à grande échelle peuvent avoir un grand nombre de points. La compression est donc essentielle pour le stockage et la transmission des nuages de points.

Les nuages de points (Figure A.1) sont des ensembles de points avec des coordonnées x, y, z et des attributs associés tels que les couleurs, les normales et la réflectance. Les nuages de points peuvent être divisés en deux composants : la géométrie, la position de chaque point individuel, et les attributs, des informations supplémentaires attachées à chacun de ces points. Un nuage de points avec dimension temporelle est appelé nuage de points dynamique, et un nuage de points sans dimension temporelle est statique.

Le Moving Picture Experts Group (MPEG) a promu deux normes [155] : MPEG Geometry-based Point Cloud Compression (G-PCC) et MPEG Video-based Point Cloud Compression (V-PCC). Ces deux normes ont atteint le stade de projet final de norme internationale (FDIS). G-PCC utilise des structures de données 3D natives telles que l'octree pour compresser les nuages de points tandis que V-PCC adopte une approche basée sur la projection basée sur les technologies de compression vidéo existantes. De plus, le Joint Photographic Experts Group

Figure A.1: Visualisations des nuages de points. De gauche à droite, "Dourado Site" [201], "soldier" [56], "phil" [116] et "Arco Valentino" [9]

(JPEG) a lancé un appel à témoignages [2] sur la compression de nuages de points (PCC).

La recherche sur la PCC peut être classée selon deux dimensions. D'une part, on peut soit compresser la géométrie du nuage de points, c'est-à-dire la position spatiale des points, soit leurs attributs associés. D'autre part, on peut également séparer les travaux portant sur la compression des nuages de points dynamiques, qui contiennent des informations temporelles, et les nuages de points statiques.

En raison des propriétés des techniques de capture de nuages de points, les nuages de points sont généralement parcimonieux. Cela rend les nuages de points différents des autres modalités telles que les images qui ont une géométrie régulière et dense sous la forme d'une grille régulière. Par conséquent, la compression des nuages de points est un problème difficile.

A.2 . Objectifs et contributions

L'objectif principal de cette thèse est d'étudier l'utilisation des techniques d'apprentissage profond pour la PCC. Plus précisément, nous étudions l'application de techniques d'apprentissage en profondeur pour les problèmes suivants : compression géométrique avec perte, compression géométrique sans perte, compression d'attribut avec perte et évaluation de la qualité des nuages de points. En effet, l'apprentissage profond pour ces problèmes était auparavant inexploré dans la littérature. Dans cette thèse, nous proposons des méthodes basées sur l'apprentissage pour résoudre ces problèmes. De plus, en utilisant cette large perspective, nous mettons en évidence les défis sous-jacents que ces problèmes étroitement liés ont en commun, à savoir la rareté et l'irrégularité des nuages de points. Nous étendons ensuite cette perspective avec un état de l'art étendu sur la PCC et montrons comment la parcomonie et l'irrégularité des nuages de points peuvent être résolus ou atténués pour proposer de nouvelles approches basées sur l'apprentissage en profondeur.

Nous montrons d'abord comment les réseaux de neurones convolutifs peuvent être utilisés pour la compression géométrique avec et sans perte. Le décodage en tant que classification est une interprétation flexible qui convient à la fois à la compression géométrique avec et sans perte. Par conséquent, nous proposons une nouvelle approche de compression sans perte basée sur la convolution masquée, puis nous l'étendons avec une formulation multi-échelle efficiente. Nous étudions également la compression d'attributs basée sur l'apprentissage profond et traitons la parcomonie et l'irrégularité de la géométrie en réduisant le problème à un problème de compression d'image. En effet, nous proposons une approche basée sur l'apprentissage pour mapper les attributs des nuages de points sur une grille régulière 2D. Cela permet l'utilisation de n'importe quelle technique de traitement

d'image pour compresser les attributs des nuages et nous montrons comment les méthodes de compression d'image peuvent être utilisées dans un tel contexte.

De plus, nous explorons une métrique de qualité de nuage de points perceptuelle profonde différentiable pour estimer la qualité visuelle perçue d'un nuage de points à partir de sa géométrie. Ceci est fortement lié à la compression géométrique car cette métrique convient également comme fonction de perte pour la formation de réseaux de compression géométrique avec perte. En effet, les fonctions de perte utilisées dans la littérature tendent à être faiblement corrélées à la qualité visuelle perçue. Les fonctions de perte qui sont plus corrélées avec la qualité visuelle perçue pourraient améliorer les performances des méthodes de compression avec perte basées sur l'apprentissage. De plus, nous proposons également une approche convolutive pour l'évaluation de la qualité des nuages de points qui gère la parcimonie et l'irrégularité via une approche de projection. En effet, les attributs reposent sur une géométrie clairsemée et irrégulière ce qui est un défi pour la conception de métriques de qualité objectives. La parcimonie et l'irrégularité peuvent être traitées via une approche de projection 2D. Cela permet ensuite l'utilisation de réseaux de neurones convolutifs 2D pour l'évaluation de la qualité qui sont bien étudiés pour les images.

A.3 . Structure

Tout d'abord, nous présentons l'état de l'art du PCC au chapitre 2. Ensuite, nous abordons la compression géométrique basée sur l'apprentissage avec une approche convolutive avec perte au chapitre 3. L'interprétation du décodage comme classification ouvre la voie à une compression sans perte avec un modèle génératif auto-régressif au chapitre 4. Cependant, de tels modèles présentent une grande complexité en raison de la génération séquentielle de probabilités de symboles. Nous abordons ce problème avec une approche de prédiction multi-échelle dans le chapitre 5. Dans le chapitre 6, nous proposons une méthode de compression d'attribut basée sur l'apprentissage qui est basée sur la compression d'image et un mappage basé sur l'apprentissage des attributs de nuage de points sur une grille 2D. Dans les méthodes de compression géométrique avec perte telles que celle présentée au chapitre 3, le réseau de neurones est optimisé conjointement en débit et en distorsion via une fonction de perte. Ainsi, cette formulation de cette fonction de perte est d'une grande importance et pour la visualisation humaine, la mesure de la distorsion devrait idéalement être fortement liée à la qualité visuelle perçue. Dans le chapitre 7, nous proposons une métrique différentiable de qualité perceptuelle bien corrélée avec la qualité visuelle perçue. Ensuite, nous abordons l'évaluation de la qualité des nuages de points basée sur l'apprentissage avec une approche basée sur la projection dans le chapitre 8. Dans l'ensemble, ces différentes

approches ont en commun que la parcimonie et l'irrégularité des nuages de points constituent un problème clé pour la compression de la géométrie, la compression des attributs et l'évaluation de la qualité des nuages de points. Dans les différents chapitres, nous montrons comment ce problème peut être résolu ou atténué pour les différents problèmes. Enfin, nous présentons les conclusions, les perspectives sur les problèmes ouverts et la liste des publications dans le chapitre 9.

Bibliography

[1] "Common test conditions for PCC," in *ISO/IEC JTC1/SC29/WG11 MPEG Output Document N19324*, Jun. 2020.

[2] "Final Call for Evidence on JPEG Pleno Point Cloud Coding," in *ISO/IEC JTC1/SC29/WG1 JPEG Output Document N88014*, Jul. 2020.

[3] "G-PCC codec description v12," in *ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document N00151*, Sep. 2021.

[4] "G-PCC Test Model v7 User Manual," in *ISO/IEC JTC1/SC29/WG11 MPEG Output Document N18664*, Jul. 2019.

[5] "ITU-T Recommendation H.265: High efficiency video coding (HEVC)," Nov. 2019.

[6] "An improvement of the planar coding mode," in *ISO/IEC JTC1/SC29/WG11 MPEG Input Document M50642*, Oct. 2019.

[7] "Inference of a mode using point location direct coding," in *ISO/IEC JTC1/SC29/WG11 MPEG Input Document M42239*, Jan. 2018.

[8] "Intra mode for geometry coding," in *ISO/IEC JTC1/SC29/WG11 MPEG Input Document M43600*, Jul. 2018.

[9] "JPEG Pleno Database: GTI-UPM Point-cloud data set," http://plenodb.jpeg.org/pc/upm, 2016.

[10] "Neighbour-dependent entropy coding of occupancy patterns," in *ISO/IEC JTC1/SC29/WG11 MPEG Input Document M42238*, Jan. 2018.

[11] "Planar mode in octree-based geometry coding," in *ISO/IEC JTC1/SC29/WG11 MPEG Input Document M48906*, Jul. 2019.

[12] "LAS specification, version 1.4 – R15," Jul. 2019.

[13] "V-PCC Codec Description," in *ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document N00100*, Jun. 2020.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp,

G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," *arXiv:1603.04467 [cs]*, Mar. 2016.

[15] I. Abouelaziz, A. Chetouani, M. E. Hassouni, L. J. Latecki, and H. Cherifi, "Convolutional Neural Network for Blind Mesh Visual Quality Assessment Using 3D Visual Saliency," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct. 2018, pp. 3533–3537.

[16] I. Abouelaziz, A. Chetouani, M. El Hassouni, L. Latecki, and H. Cherifi, "3D visual saliency and convolutional neural network for blind mesh quality assessment," *Neural Computing and Applications*, 2019.

[17] I. Abouelaziz, A. Chetouani, M. El Hassouni, L. J. Latecki, and H. Cherifi, "No-reference mesh visual quality assessment via ensemble of convolutional neural networks and compact multi-linear pooling," *Pattern Recognition*, vol. 100, p. 107174, Apr. 2020.

[18] J. Ahn, K. Lee, J. Sim, and C. Kim, "Large-Scale 3D Point Cloud Compression Using Adaptive Radial Distance Prediction in Hybrid Coordinate Domains," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 3, pp. 422–434, Apr. 2015.

[19] A. Akhtar, W. Gao, X. Zhang, L. Li, Z. Li, and S. Liu, "Point Cloud Geometry Prediction Across Spatial Scale using Deep Learning," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, Dec. 2020, pp. 70–73.

[20] E. Alexiou and T. Ebrahimi, "Towards a Point Cloud Structural Similarity Metric," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2020, pp. 1–6.

[21] E. Alexiou and T. Ebrahimi, "Point Cloud Quality Assessment Metric Based on Angular Similarity," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2018, pp. 1–6.

[22] E. Alexiou, K. Tung, and T. Ebrahimi, "Towards neural network approaches for point cloud compression," in *Applications of Digital Image Processing XLIII*, vol. 11510. International Society for Optics and Photonics, Aug. 2020, p. 1151008.

[23] I. Alonso, L. Riazuelo, L. Montesano, and A. C. Murillo, "3D-MiniNet: Learning a 2D Representation From Point Clouds for Fast and Efficient 3D LIDAR Semantic Segmentation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5432–5439, Oct. 2020.

[24] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end Optimized Image Compression," in *2017 5th International Conference on Learning Representations (ICLR)*, 2017.

[25] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *2018 6th International Conference on Learning Representations (ICLR)*, Jan. 2018.

[26] F. O. Bartell, E. L. Dereniak, and W. L. Wolfe, "The Theory And Measurement Of Bidirectional Reflectance Distribution Function (Brdf) And Bidirectional Transmittance Distribution Function (BTDF)," in *Radiation Scattering in Optical Systems*, vol. 0257. SPIE, Mar. 1981, pp. 154–160.

[27] F. Bellard, "BPG Image format," https://bellard.org/bpg/.

[28] P. J. Besl, "Active Optical Range Imaging Sensors," in *Advances in Machine Vision*, ser. Springer Series in Perception Engineering, J. L. C. Sanz, Ed. New York, NY: Springer, 1989, pp. 1–63.

[29] S. Biswas, J. Liu, K. Wong, S. Wang, and R. Urtasun, "MuSCLE: Multi Sweep Compression of LiDAR using Deep Entropy Models," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[30] J. F. Blinn, "Simulation of wrinkled surfaces," in *Proceedings of the 5th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '78. New York, NY, USA: Association for Computing Machinery, Aug. 1978, pp. 286–292.

[31] S. Bosse, D. Maniry, K.-R. Müller, T. Wiegand, and W. Samek, "Deep Neural Networks for No-Reference and Full-Reference Image Quality Assessment," *IEEE Transactions on Image Processing*, vol. 27, no. 1, pp. 206–219, Jan. 2018.

[32] R. D. L. Briandais, "File searching using variable length keys," in *IRE-AIEE-ACM Computer Conference*, 1959.

[33] G. Bruder, F. Steinicke, and A. Nüchter, "Poster: Immersive point cloud virtual environments," in *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, Mar. 2014, pp. 161–162.

[34] C. Cao, M. Preda, and T. Zaharia, "3D Point Cloud Compression: A Survey," in *The 24th International Conference on 3D Web Technology*. LA CA USA: ACM, Jul. 2019, pp. 1–9.

[35] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of Sparse and Dense Dynamic Point Clouds–Methods and Standards," *Proceedings of the IEEE*, pp. 1–22, 2021.

[36] E. E. Catmull, "A subdivision algorithm for computer display of curved surfaces," Ph.D. dissertation, The University of Utah, 1974.

[37] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 77–85.

[38] J. Chen, J. Chen, H. Chao, and M. Yang, "Image Blind Denoising with Generative Adversarial Network Based Noise Modeling," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 3155–3164.

[39] A. Chetouani, "Image Quality Assessment Without Reference By Mixing Deep Learning-Based Features," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2020, pp. 1–6.

[40] A. Chetouani, "Three-dimensional mesh quality metric with reference based on a support vector regression model," *Journal of Electronic Imaging*, vol. 27, no. 4, p. 043048, Aug. 2018.

[41] A. Chetouani and L. Li, "On the use of a scanpath predictor and convolutional neural network for blind image quality assessment," *Signal Processing: Image Communication*, vol. 89, p. 115963, Nov. 2020.

[42] A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Combination of Deep Learning-based and Handcrafted Features for Blind Image Quality Assessment," in *9th European Workshop on Visual Information Processing (EUVIP 2021)*, Paris (virtual), France, Jun. 2021.

[43] A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Convolutional Neural Network for 3D Point Cloud Quality Assessment with Reference," in *IEEE International Workshop on Multimedia Signal Processing (MMSP'2021)*, Oct. 2021.

[44] A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "Deep Learning-Based Quality Assessment Of 3d Point Clouds Without Reference," in *2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2021, pp. 1–6.

[45] A. Chetouani, M. Quach, G. Valenzise, and F. Dufaux, "LEARNING-BASED 3D POINT CLOUD QUALITY ASSESSMENT USING A SUPPORT VECTOR REGRESSOR," in *Image Quality and System Performance, IS&T International Symposium on Electronic Imaging (EI 2022)*, San Francisco, United States, Jan. 2022.

[46] P. A. Chou, M. Koroteev, and M. Krivokuća, "A Volumetric Approach to Point Cloud Compression—Part I: Attribute Compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 2203–2216, 2020.

[47] C. Choy, J. Gwak, and S. Savarese, "4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 3070–3079.

[48] R. A. Cohen, D. Tian, and A. Vetro, "Attribute compression for sparse point clouds using graph transforms," in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 1374–1378.

[49] R. A. Cohen, M. Krivokuća, C. Feng, Y. Taguchi, H. Ochimizu, D. Tian, and A. Vetro, "Compression of 3-D point clouds using hierarchical patch fitting," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 4033–4037.

[50] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques - SIGGRAPH '96*. Not Known: ACM Press, 1996, pp. 303–312.

[51] S. J. Daly, "Visible differences predictor: An algorithm for the assessment of image fidelity," in *SPIE/IS&T 1992 Symposium on Electronic Imaging: Science and Technology*, B. E. Rogowitz, Ed., San Jose, CA, Aug. 1992, p. 2.

[52] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-Based Static 3D Point Clouds Geometry Coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 284–299, Feb. 2019.

[53] R. L. de Queiroz and P. A. Chou, "Transform Coding for Point Clouds Using a Gaussian Process Model," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3507–3517, Jul. 2017.

[54] R. L. de Queiroz and P. A. Chou, "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, Aug. 2016.

[55] R. L. de Queiroz, C. Dorea, D. R. Freitas, M. Krivokuca, and G. P. Sandri, "Model-Centric Volumetric Point Cloud Attributes," *arXiv:2106.15539 [cs, eess]*, Jun. 2021.

[56] 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, Jan. 2017.

[57] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2016, pp. 1–6.

[58] A. Dricot and J. Ascenso, "Adaptive Multi-level Triangle Soup for Geometry-based Point Cloud Coding," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2019, pp. 1–6.

[59] A. Dricot and J. Ascenso, "Hybrid Octree-Plane Point Cloud Geometry Coding," in *2019 27th European Signal Processing Conference (EUSIPCO)*, Sep. 2019, pp. 1–5.

[60] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv:1603.07285 [cs, stat]*, Mar. 2016.

[61] Y. Feng, S. Liu, and Y. Zhu, "Real-Time Spatio-Temporal LiDAR Point Cloud Compression," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 10 766–10 773.

[62] D. R. Freitas, E. Peixoto, R. L. de Queiroz, and E. Medeiros, "Lossy Point Cloud Geometry Compression Via Dyadic Decomposition," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 2731–2735.

[63] T. Fukuda, "Point Cloud Stream on Spatial Mixed Reality - Toward Telepresence in Architectural Field," in *Kepczynska-Walczak, A, Bialkowski, S (Eds.), Computing for a Better Tomorrow - Proceedings of the 36th eCAADe*

Conference - Volume 2, Lodz University of Technology, Lodz, Poland, 19-21 September 2018, Pp. 727-734. CUMINCAD, 2018.

[64] L. Gao, T. Fan, J. Wan, Y. Xu, J. Sun, and Z. Ma, "Point Cloud Geometry Compression Via Neural Graph Sampling," in *2021 IEEE International Conference on Image Processing (ICIP)*, Sep. 2021, pp. 3373–3377.

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

[66] D. C. Garcia and R. L. de Queiroz, "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct. 2018, pp. 1807–1811.

[67] D. C. Garcia, T. A. Fonseca, R. U. Ferreira, and R. L. de Queiroz, "Geometry Coding for Dynamic Voxelized Point Clouds Using Octrees and Multiple Contexts," *IEEE Transactions on Image Processing*, vol. 29, pp. 313–322, 2020.

[68] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.

[69] G. G. Goyer and R. Watson, "The Laser and its Application to Meteorology," *Bulletin of the American Meteorological Society*, vol. 44, no. 9, pp. 564–570, Sep. 1963.

[70] B. Graham and L. van der Maaten, "Submanifold Sparse Convolutional Networks," *arXiv:1706.01307 [cs]*, Jun. 2017.

[71] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, 2020/ed.

[72] D. P. Greenberg, "A framework for realistic image synthesis," *Communications of the ACM*, vol. 42, no. 8, pp. 44–53, Aug. 1999.

[73] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," in *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, Jun. 2015, pp. 1462–1471.

[74] S. Gu, J. Hou, H. Zeng, J. Chen, J. Zhu, and K. Ma, "Compression of 3D point clouds using 1D discrete cosine transform," in *2017 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, Nov. 2017, pp. 196–200.

[75] S. Gu, J. Hou, H. Zeng, H. Yuan, and K.-K. Ma, "3D Point Cloud Attribute Compression Using Geometry-Guided Sparse Representation," *IEEE Transactions on Image Processing*, vol. 29, pp. 796–808, 2020.

[76] A. Guarda, N. Rodrigues, and F. Pereira, "Adaptive Deep Learning-based Point Cloud Geometry Coding," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–1, 2020.

[77] A. Guarda, N. Rodrigues, and F. Pereira, "Neighborhood Adaptive Loss Function for Deep Learning-based Point Cloud Coding with Implicit and Explicit Quantization," *IEEE MultiMedia*, pp. 1–1, 2020.

[78] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Deep Learning-Based Point Cloud Coding: A Behavior and Performance Study," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*, Oct. 2019, pp. 34–39.

[79] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Point Cloud Coding: Adopting a Deep Learning-based Approach," in *2019 Picture Coding Symposium (PCS)*, Nov. 2019, pp. 1–5.

[80] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Deep Learning-Based Point Cloud Geometry Coding: RD Control Through Implicit and Explicit Quantization," in *2020 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2020, pp. 1–6.

[81] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Point Cloud Geometry Scalable Coding With a Single End-to-End Deep Learning Model," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 3354–3358.

[82] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[83] Y. He, G. Li, Y. Shao, J. Wang, Y. Chen, and S. Liu, "A point cloud compression framework via spherical projection," in *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, Dec. 2020, pp. 62–65.

[84] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[85] J. Hou, L. Chau, Y. He, and P. A. Chou, "Sparse representation for colors of 3D point cloud via virtual adaptive sampling," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar. 2017, pp. 2926–2930.

[86] H. Houshiar and A. Nüchter, "3D point cloud compression using conventional image compression for efficient data transmission," in *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT)*, Oct. 2015, pp. 1–8.

[87] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 1310–1320.

[88] T. Huang and Y. Liu, "3D Point Cloud Geometry Compression on Deep Learning," in *Proceedings of the 27th ACM International Conference on Multimedia*. Nice France: ACM, Oct. 2019, pp. 890–898.

[89] Y. Huang, J. Peng, C.-C. J. Kuo, and M. Gopi, "A Generic Scheme for Progressive Point Cloud Coding," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 2, pp. 440–453, Mar. 2008.

[90] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, Sep. 1952.

[91] M. Isenburg, "LASzip: Lossless compression of LiDAR data," *Photogrammetric engineering and remote sensing*, vol. 79, no. 2, pp. 209–217, 2013.

[92] B. Isik, P. A. Chou, S. J. Hwang, N. Johnston, and G. Toderici, "LVAC: Learned Volumetric Attribute Compression for Point Clouds using Coordinate Based Networks," *arXiv:2111.08988 [cs, eess]*, Nov. 2021.

[93] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, "Improving PSNR-based Quality Metrics Performance For Point Cloud Geometry," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 3438–3442.

[94] A. Javaheri, C. Brites, F. Pereira, and J. Ascenso, "Mahalanobis Based Point to Distribution Metric for Point Cloud Geometry Quality Evaluation," *IEEE Signal Processing Letters*, vol. 27, pp. 1350–1354, 2020.

[95] W. Jiang, J. Tian, K. Cai, F. Zhang, and T. Luo, "Tangent-plane-continuity maximization based 3D point compression," in *2012 19th IEEE International Conference on Image Processing*, Sep. 2012, pp. 1277–1280.

[96] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*, May 2012, pp. 778–785.

[97] L. Kang, P. Ye, Y. Li, and D. Doermann, "Convolutional Neural Networks for No-Reference Image Quality Assessment," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, OH, USA: IEEE, Jun. 2014, pp. 1733–1740.

[98] E. C. Kaya and I. Tabus, "Neural Network Modeling of Probabilities for Coding the Octree Representation of Point Clouds," *arXiv:2106.06482 [cs, eess]*, Jun. 2021.

[99] E. C. Kaya, S. Schwarz, and I. Tabus, "Refining The Bounding Volumes for Lossless Compression of Voxelized Point Clouds Geometry," in *2021 IEEE International Conference on Image Processing (ICIP)*, Sep. 2021, pp. 3408–3412.

[100] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, ser. SGP '06. Goslar, DEU: Eurographics Association, Jun. 2006, pp. 61–70.

[101] R. Killea, Y. Li, S. Bastani, and P. McLachlan, "DeepCompress: Efficient Point Cloud Geometry Compression," *arXiv:2106.01504 [cs, eess]*, Jun. 2021.

[102] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *2015 3rd International Conference on Learning Representations*, Dec. 2014.

[103] M. Krivokuća and C. Guillemot, "Colour compression of plenoptic point clouds using raht-klt with prior colour clustering and specular/diffuse component separation," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2020.

[104] M. Krivokuća, M. Koroteev, and P. A. Chou, "A Volumetric Approach to Point Cloud Compression," *arXiv:1810.00484 [eess]*, Sep. 2018.

[105] M. Krivokuća, P. A. Chou, and M. Koroteev, "A Volumetric Approach to Point Cloud Compression–Part II: Geometry Compression," *IEEE Transactions on Image Processing*, vol. 29, pp. 2217–2229, 2020.

[106] M. Krivokuća, E. Miandji, C. Guillemot, and P. Chou, "Compression of Plenoptic Point Cloud Attributes Using 6-D Point Clouds and 6-D Transforms," *IEEE Transactions on Multimedia*, p. 1, 2021.

[107] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *arXiv:1404.5997 [cs]*, Apr. 2014.

[108] M. Landy and J. A. Movshon, "The Plenoptic Function and the Elements of Early Vision," in *Computational Models of Visual Processing*. MIT Press, 1991, pp. 3–20.

[109] G. Lavoué, "A Multiscale Metric for 3D Mesh Visual Quality Assessment," *Computer Graphics Forum*, vol. 30, no. 5, pp. 1427–1437, 2011.

[110] D. Lazzarotto and T. Ebrahimi, "Learning residual coding for point clouds," in *Applications of Digital Image Processing XLIV*, A. G. Tescher and T. Ebrahimi, Eds. San Diego, United States: SPIE, Aug. 2021, p. 31.

[111] D. Lazzarotto, E. Alexiou, and T. Ebrahimi, "On Block Prediction For Learning-Based Point Cloud Compression," in *2021 IEEE International Conference on Image Processing (ICIP)*, Sep. 2021, pp. 3378–3382.

[112] L. Li, Z. Li, S. Liu, and H. Li, "Occupancy-Map-Based Rate Distortion Optimization and Partition for Video-Based Point Cloud Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 1, pp. 326–338, Jan. 2021.

[113] Y. Li and J. Ibanez-Guzman, "Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, Jul. 2020.

[114] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2999–3007.

[115] Y. Liu, Q. Yang, Y. Xu, and L. Yang, "Point Cloud Quality Assessment: Large-scale Dataset Construction and Learning-based No-Reference Approach," *arXiv:2012.11895 [eess]*, Dec. 2020.

[116] 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.

[117] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM siggraph computer graphics*, vol. 21, no. 4, pp. 163–169, 1987.

[118] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *In ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[119] K. Mammou, P. A. Chou, D. Flynn, and M. Krivokuća, "PCC Test Model Category 13 v3," in *ISO/IEC JTC1/SC29/WG11 MPEG Output Document N17762*, Jul. 2018.

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

[121] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. V. Gool, "Conditional Probability Models for Deep Image Compression," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 4394–4402.

[122] F. Mentzer, E. Agustsson, M. Tschannen, R. Timofte, and L. Van Gool, "Practical Full Resolution Learned Lossless Image Compression," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, CA, USA: IEEE, Jun. 2019, pp. 10 621–10 630.

[123] O. Messai, A. Chetouani, F. Hachouf, and Z. A. Seghir, "No-reference Stereoscopic Image Quality Predictor using Deep Features from Cyclopean Image," *Electronic Imaging*, vol. 2021, no. 9, pp. 297–1–297–9, Jan. 2021.

[124] G. Meynet, J. Digne, and G. Lavoué, "PC-MSDM: A quality metric for 3D point clouds," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2019, pp. 1–3.

[125] G. Meynet, Y. Nehmé, J. Digne, and G. Lavoué, "PCQM: A Full-Reference Quality Metric for Colored 3D Point Clouds," in *12th International Conference on Quality of Multimedia Experience (QoMEX 2020)*, Athlone, Ireland, May 2020.

[126] S. Milani, "A Syndrome-Based Autoencoder For Point Cloud Geometry Compression," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 2686–2690.

[127] G. M. Morton, "A computer oriented geodetic data base and a new technique in file sequencing," 1966.

[128] MPEG, "Description of Exploration Experiment 13.2 on inter prediction," in *ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document N0155*, Aug. 2021.

[129] MPEG, "EE 13.53 Low Latency Low Complexity LIDAR Codec (LL-LC^2)," in *ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document N0165*, Jul. 2021.

[130] MPEG, "Performance analysis of currently AI-based available solutions for PCC," in *ISO/IEC JTC 1/SC 29/WG 7 MPEG Output Document N00233*, Oct. 2021.

[131] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, J. Fürnkranz and T. Joachims, Eds. Omnipress, 2010, pp. 807–814.

[132] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Learning-Based Lossless Compression of 3D Point Cloud Geometry," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jun. 2021, pp. 4220–4224.

[133] 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, Dec. 2021.

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

[135] T. Ochotta and D. Saupe, *Compression of Point-Based 3D Models by Shape-Adaptive Wavelet Coding of Multi-Height Fields*. The Eurographics Association, 2004.

[136] A. V. Oord, N. Kalchbrenner, and K. Kavukcuoglu, "Pixel Recurrent Neural Networks," in *Proceedings of The 33rd International Conference on Machine Learning*. PMLR, Jun. 2016, pp. 1747–1756.

[137] A. Pacala, "Lidar as a camera - digital lidar's implications for computer vision," https://ouster.com/blog/the-camera-is-in-the-lidar, Aug. 2018.

[138] F. Pereira, A. Dricot, J. Ascenso, and C. Brites, "Point cloud coding: A privileged view driven by a classification taxonomy," *Signal Processing: Image Communication*, vol. 85, p. 115862, Jul. 2020.

[139] S. Perry, H. P. Cong, L. A. da Silva Cruz, J. Prazeres, M. Pereira, A. Pinheiro, E. Dumic, E. Alexiou, and T. Ebrahimi, "Quality Evaluation Of Static Point Clouds Encoded Using MPEG Codecs," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 3428–3432.

[140] M. Quach, G. Valenzise, and F. Dufaux, "Folding-Based Compression Of Point Cloud Attributes," in *2020 IEEE International Conference on Image Processing (ICIP)*, Oct. 2020, pp. 3309–3313.

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

[142] M. Quach, G. Valenzise, and F. Dufaux, "Improved Deep Point Cloud Geometry Compression," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2020, pp. 1–6.

[143] M. Quach, A. Chetouani, G. Valenzise, and F. Dufaux, "A deep perceptual metric for 3D point clouds," *Electronic Imaging*, vol. 2021, no. 9, pp. 257–1–257–7, Jan. 2021.

[144] M. Quach, G. Valenzise, and F. Dufaux, "A Deep Point Cloud Geometry Coding Toolbox," in *2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, Jul. 2021, pp. 1–2.

[145] M. Quach, J. Pang, D. Tian, G. Valenzise, and F. Dufaux, "Survey on Deep Learning-Based Point Cloud Compression," *Frontiers in Signal Processing*, vol. 2, 2022.

[146] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An Octree Based Framework for Point Cloud Compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6042–6051.

[147] S. Reed, A. Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, Y. Chen, D. Belov, and N. Freitas, "Parallel Multiscale Autoregressive Density Estimation," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 2912–2921.

[148] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing*

and Computer-Assisted Intervention – MICCAI 2015, ser. Lecture Notes in Computer Science, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[149] A. Said and W. Pearlman, "A new, fast, and efficient image codec based on set partitioning in hierarchical trees," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 243–250, Jun. 1996.

[150] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma, "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[151] G. Sandri, R. L. de Queiroz, and P. A. Chou, "Compression of Plenoptic Point Clouds," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1419–1427, Mar. 2019.

[152] G. Sandri, R. L. de Queiroz, and P. A. Chou, "Comments on "Compression of 3D Point Clouds Using a Region-Adaptive Hierarchical Transform"," *arXiv:1805.09146 [eess]*, May 2018.

[153] R. Schnabel and R. Klein, "Octree-based Point-cloud Compression," in *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, ser. SPBG'06. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 111–121.

[154] S. Schwarz, G. Martin-Cocher, D. Flynn, and M. Budagavi, "Common test conditions for point cloud compression," in *ISO/IEC JTC1/SC29/WG11 MPEG output document N17766*, Jul. 2018.

[155] 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*, pp. 1–1, 2018.

[156] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox, "Orientation-boosted Voxel Nets for 3D Object Recognition," in *Procedings of the British Machine Vision Conference 2017*. London, UK: British Machine Vision Association, 2017, p. 97.

[157] Y. Shen, W. Dai, C. Li, J. Zou, and H. Xiong, "Multi-Scale Structured Dictionary Learning for 3-D Point Cloud Attribute Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2020.

[158] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu, "Deep-PCAC: An End-to-End Deep Lossy Compression Framework for Point Cloud Attributes," *IEEE Transactions on Multimedia*, pp. 1–1, 2021.

[159] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015.

[160] J.-Y. Sim, C.-S. Kim, and S.-U. Lee, "Lossless compression of 3-D point data in QSplat representation," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1191–1195, Dec. 2005.

[161] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[162] F. Song, Y. Shao, W. Gao, H. Wang, and T. Li, "Layer-Wise Geometry Aggregation Framework for Lossless LiDAR Point Cloud Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[163] A. Souto, V. Figueiredo, P. Chou, and R. de Queiroz, "Set Partitioning in Hierarchical Trees for Point Cloud Attribute Compression," Jul. 2021.

[164] V. Sterzentsenko, A. Karakottas, A. Papachristou, N. Zioulis, A. Doumanoglou, D. Zarpalas, and P. Daras, "A Low-Cost, Flexible and Portable Volumetric Capturing System," *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, pp. 200–207, Nov. 2018.

[165] X. Sun, H. Ma, Y. Sun, and M. Liu, "A Novel Point Cloud Compression Algorithm Based on Clustering," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2132–2139, Apr. 2019.

[166] X. Sun, S. Wang, M. Wang, Z. Wang, and M. Liu, "A Novel Coding Architecture for LiDAR Point Cloud Sequence," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5637–5644, Oct. 2020.

[167] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1–9.

[168] D. Tang, S. Singh, P. A. Chou, C. Häne, M. Dou, S. Fanello, J. Taylor, P. Davidson, O. G. Guleryuz, Y. Zhang, S. Izadi, A. Tagliasacchi, S. Bouaziz, and C. Keskin, "Deep Implicit Volume Compression," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 1290–1300.

[169] L. Theis and M. Bethge, "Generative Image Modeling Using Spatial LSTMs," in *Advances in Neural Information Processing Systems*, vol. 28. Curran Associates, Inc., 2015.

[170] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[171] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*. Beijing: IEEE, Sep. 2017, pp. 3460–3464.

[172] C. Tommasi, C. Achille, and F. Fassi, "FROM POINT CLOUD TO BIM: A MODELLING CHALLENGE IN THE CULTURAL HERITAGE FIELD," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLI-B5, pp. 429–436, Jun. 2016.

[173] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Continuous point cloud data compression using SLAM based prediction," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1744–1751.

[174] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Point Cloud Compression for 3D LiDAR Sensor using Recurrent Neural Network with Residual Blocks," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 3274–3280.

[175] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Real-Time Streaming Point Cloud Compression for 3D LiDAR Sensor Using U-Net," *IEEE Access*, vol. 7, pp. 113 616–113 625, 2019.

[176] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing continuous point cloud data using image compression methods," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 1712–1719.

[177] D. E. Tzamarias, K. Chow, I. Blanes, and J. Serra-Sagristà, "Compression of point cloud geometry through a single projection," in *2021 Data Compression Conference (DCC)*, Mar. 2021, pp. 63–72.

[178] G. Valenzise, M. Quach, D.-T. Nguyen, and F. Dufaux, "Voxel-based Deep Point Cloud Geometry Compression," in *CORESA (COmpression et REprésentation Des Signaux Audiovisuels)*, Sophia-Antipolis, France, Nov. 2021.

[179] I. Viola, S. Subramanyam, and P. Cesar, "A Color-Based Objective Quality Metric for Point Cloud Contents," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2020, pp. 1–6.

[180] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned Point Cloud Geometry Compression," *arXiv:1909.12037 [cs, eess]*, Sep. 2019.

[181] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse Tensor-based Multiscale Representation for Point Cloud Geometry Compression," *arXiv:2111.10633 [cs, eess]*, Nov. 2021.

[182] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale Point Cloud Geometry Compression," in *2021 Data Compression Conference (DCC)*, Mar. 2021, pp. 73–82.

[183] Q. Wang and M.-K. Kim, "Applications of 3D point cloud data in the construction industry: A fifteen-year review from 2004 to 2018," *Advanced Engineering Informatics*, vol. 39, pp. 306–319, Jan. 2019.

[184] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[185] M. Weinberger, J. Rissanen, and R. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Transactions on Image Processing*, vol. 5, no. 4, pp. 575–586, Apr. 1996.

[186] X. Wen, X. Wang, J. Hou, L. Ma, Y. Zhou, and J. Jiang, "Lossy Geometry Compression Of 3d Point Cloud Data Via An Adaptive Octree-Guided Network," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, Jul. 2020, pp. 1–6.

[187] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley, "Deep Compression for Dense Point Cloud Maps," *IEEE Robotics and Automation Letters*, pp. 1–1, 2021.

[188] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1912–1920.

[189] J. Xiong, H. Gao, M. Wang, H. Li, and W. Lin, "Occupancy Map Guided Fast Video-based Dynamic Point Cloud Coding," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[190] W. Yan, Y. shao, S. Liu, T. H. Li, Z. Li, and G. Li, "Deep AutoEncoder-based Lossy Geometry Compression for Point Clouds," *arXiv:1905.03691 [cs, eess]*, Apr. 2019.

[191] Q. Yang, H. Chen, Z. Ma, Y. Xu, R. Tang, and J. Sun, "Predicting the Perceptual Quality of Point Cloud: A 3D-to-2D Projection-Based Exploration," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.

[192] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point Cloud Auto-encoder via Deep Grid Deformation," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Dec. 2017.

[193] Q. Yin, Q. Ren, L. Zhao, W. Wang, and J. Chen, "Lossless Point Cloud Attribute Compression with Normal-based Intra Prediction," in *2021 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, Aug. 2021, pp. 1–5.

[194] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, "A LiDAR Point Cloud Generator: From a Virtual World to Autonomous Driving," in *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 458–464.

[195] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, Oct. 2014, pp. 2066–2070.

[196] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 586–595.

[197] X. Zhang, W. Wan, and X. An, "Clustering and DCT Based Color Point Cloud Compression," *Journal of Signal Processing Systems*, vol. 86, no. 1, pp. 41–49, Jan. 2017.

[198] W. Zhu, Y. Xu, L. Li, and Z. Li, "Lossless point cloud geometry compression via binary tree partition and intra prediction," in *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2017, pp. 1–6.

[199] W. Zhu, Z. Ma, Y. Xu, L. Li, and Z. Li, "View-Dependent Dynamic Point Cloud Compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 2, pp. 765–781, Feb. 2021.

[200] G. Y. Zou, "Toward using confidence intervals to compare correlations." *Psychological Methods*, vol. 12, no. 4, pp. 399–413, Dec. 2007.

[201] M. Zuffo, "EmergIMG | Downloads - USPAULOPC," http://uspaulopc.di.ubi.pt/, 2018.