Volumetric Attribute Compression for 3D Point Clouds using Feedforward Network with Geometric Attention

Tam Thuc Do

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 1, 2023

Abstract

We study 3D point cloud attribute compression using a volumetric approach: given a target volumetric attribute function $f : \mathbb{R}^3 \to \mathbb{R}$, we quantize and encode parameter vector θ that characterizes f at the encoder, for reconstruction $f_{\hat{\theta}}(\mathbf{x})$ at known 3D points $\mathbf{x}$'s at the decoder, where $\hat{\theta}$ is a quantized version of θ. Extending a previous work Region Adaptive Hierarchical Transform (RAHT) that employs piecewise constant functions to span a nested sequence of function spaces, we propose a feedforward linear network that implements higher-order B-spline bases spanning function spaces without eigen-decomposition. Feedforward network architecture means that the system is amenable to end-to-end neural learning. The key to our network is space-varying convolution, similar to a graph operator, whose weights are computed from the known 3D geometry for normalization. We show that the number of layers in the normalization at the encoder is equivalent to the number of terms in a matrix inverse Taylor series. Experimental results on real-world 3D point clouds show up to 2-3 dB gain over RAHT in energy compaction and 20-30% in bitrate reduction.

Contents

List of Figures

Chapter 1

Introduction

A Point Cloud (PC) is a set of discrete points $\{(\mathbf{x}_i, \mathbf{y}_i)\}$ in 3D space, where each point i consists of a *point coordinate* ($\mathbf{x}_i \in \mathbb{R}^3$) in 3D Euclidean space and an *attribute* ($\mathbf{y}_i \in \mathbb{R}^r$), such as RGB values to describe color. PC is a basic representation of a 3D scene and/or objects so that a viewpoint image can be realistically rendered from an arbitrary perspective via 3D rendering (e.g., [1]).

Driven by advances in image sensing technologies, PCs can now be captured cheaply in real time for different applications, such as virtual and augmented reality, immersive communication, remote sensing, and autonomous vehicles [2, 3, 4]. Due to the sheer volume of the captured data, one key technical challenge in PC processing is compression for storage and/or transmission: how to efficiently reduce data representation size with minimum loss in signal fidelity upon data decoding.

The PC compression problem can be divided into two: *geometry compression* and *attribute compression*. Geometry compression is the problem of computing a compact representation of a list of 3D point coordinates. Attribute compression is the problem of compressing the corresponding attribute values of the 3D points, given that the point coordinates are available at the decoder.

Although geometry and attribute compression can be done jointly (e.g., [5]), the approach to perform *geometry compression* first and then perform *attribute compression* conditioned on the decoded geometry is dominant in recent research [6]–[11] as well as in the MPEG geometry-based point cloud compression (G-PCC) standard [12]–[14]. Thus, we focus on *attribute compression* and make the assumption that the decoded geometry is known at both the attribute encoder and at the attribute decoder.

Specifically, we pursue a *volumetric approach*: real vector-valued function $f : \mathbb{R}^d \mapsto \mathbb{R}^r$ is *volumetric* if $d = 3$ (or *hyper-volumetric* if $d > 3$). Hence, our point clouds are clearly samples of a volumetric function where $\mathbf{y}_i = f(\mathbf{x}_i)$. Suppose such a volumetric function can be represented by a parametric function f_θ, which belongs to a family of parametric functions $\mathcal{F}_\theta$, and perfectly produces point samples $\{(\mathbf{x}_i, \mathbf{y}_i)\}$. Then, compression of a point cloud can be done by quantizing and entropy coding the parameter vector θ, decoding it as $\hat{\theta}$, and reconstructing the compressed field as $f_{\hat{\theta}}$. To recover the point cloud, we just need to evaluate $\hat{\mathbf{y}}_i = f_{\hat{\theta}}(\mathbf{x}_i)$.

The volumetric approach to point cloud attribute compression is the core of MPEG G-PCC [10], which uses the *Region Adaptive Hierarchical Transform* (RAHT) [8, 15]. RAHT projects any volumetric function f that interpolates attributes $\mathbf{y}_i$ at positions $\mathbf{x}_i$ onto a nested sequence of function spaces $\mathcal{F}_0^{(1)} \subseteq \cdots \subseteq \mathcal{F}_L^{(1)}$ at different levels of detail $l = 0, \ldots, L$. These nested sequence of function spaces are sets of parametric functions defined by B-spline of order $p = 1$ with different scales corresponding to different levels of detail. Further, [10] showed how to improve RAHT by using the parametric function spaces of B-splines of order $p > 1$. However, projecting f onto $\mathcal{F}_l^{(p)}$ for $p > 1$ required large computation resources for expensive eigen-decomposition.

Meanwhile, following the success of learned neural networks for image compression [16]–[25], a number of works have successfully improved point cloud *geometry compression* in MPEG G-PCC using learned neural networks [26]–[32]. However, on the task of point cloud attribute compression, the learned neural methods have yet to outperform MPEG G-PCC [33]–[38].

As the focus of the thesis, we take a step towards learning networks for point cloud attribute compression by leveraging the well-developed framework of nested volumetric function representation. We list our contributions below:

- We showed that the multi-resolution approximation framework [10], which normally requires eigen-decomposition of large matrices, can be implemented as a feedforward neutral network, *without eigen-decompositions*. We consider this a new neural multi-resolution compression framework for 3D point cloud attributes.

- We proposed a geometric attention layer, which originated from the model-based approach so that it has a specific mathematical interpretation. This attention layer will incorporate the irregularity and sparsity of the underlying 3D geometry into the transformation to derive the representation of the 3D point clouds attributes.

- Experimental results showed that the high order B-spline basis functions are special cases of our feedforward neural network. On real-world 3D point clouds, our model shows up to 2-3 dB gain in PSNR over RAHT in reconstruction signal quality, or 20-30% in bitrate reduction. This means that we require less memory to store the 3D point cloud attributes at the same quality.

Chapter 2

Related Works

The essence of point cloud attribute compression is managing the irregularity of the underlying 3D geometry. Traditional approaches to handling the irregular patterns of the points' locations can be grouped into two categories.

The first category relies on the 3D Euclidean point locations to construct a representation structure for the PC. It can be further divided into two groups of approaches: hierarchical representations (e.g., octree) that also capture the multiscale resolution of the PC [39, 1, 40], and graphs of connected points on which attributes are defined as signals on the graph and later transformed using *Graph Fourier Transforms* (GFTs) [41, 42, 43, 44]. Note that octree representation, which will be discussed in section 3.2.2, is currently used for standard static PC compression [1].

However, building graphs and finding graph transforms for compression still require complex operations that are not easily mitigated. Interestingly, the connection between these two groups of approaches has only been partially explored; for instance, a hybrid of graph structure and hierarchical structure was explored in [44] and achieved better compression performance than the standalone octree representation. Recently, [45] generalized RAHT [39] with their framework using volumetric

functions defined on a B-spline wavelet basis; this framework made no assumption on the underneath structure of the PC and proved that the octree representation is one of its special cases.

The second category manages PC irregularity by assuming that the PC is a sampling of continuous surface (2D manifold) so that the PC can be represented by regular grids of 2D images—the attributes can be projected onto 2D grid images and then compressed using available image/video compression tools [46, 47]. Due to its efficiency and simplicity, this approach was standardized into MPEG codec for video-based point cloud compression [1]. However, this approach induces distortion during the projection step, and it also suffers in the case of sparse PC, where the projected images have many missing patches, limiting its coding efficiency.

Meanwhile, following the success of learned neural networks for image compression [16]–[25], a number of works have successfully improved on the point cloud geometry compression in MPEG G-PCC using learned neural networks [26]–[32]. A few works have tried using learned neural networks for point cloud attribute compression as well [5][33]–[35]. Other works have used neural networks for volumetric attribute compression more generally [36]–[38]. On the task of point cloud attribute compression, the learned neural methods have yet to outperform MPEG G-PCC standard in [1]. The reason may be traced to the fundamental difficulty with point clouds that they are sparse, irregular, and changing.

The main direction of these approaches is to express the 3D Euclidean point locations in 3D grids [48, 27] (or 2D grids by projecting PC onto 2D grid images [47, 49]) and stacked 3D convolutional layers to generate compact latent code. The models are then trained with a loss function similar to the classical Rate-Distortion

tradeoff. These methods are very inefficient since they require huge computation and memory because of the 3D convolution layers.

Another major approach is point-based [50, 49], which embraces the hierarchical representation of the point cloud geometry. At each level of resolution, the attributes of 3D points were grouped together and went through a multi-layer perceptron to extract features of that local region—this was called the point convolution operation. This is more efficient than earlier 3D convolutional approaches, but at the decoder, the upsampling layers change the geometric coordinates, resulting in incorrect reconstruction of the attributes in its 3D location, introducing unnecessary distortions.

Chapter 3

Preliminary

This chapter provides an introduction to important concepts for the volumetric approach in attribute compression for 3D point clouds. In Section 3.1, we provide some definitions and theories needed to define Hilbert space. We then provide a definition of projection onto a subspace. In Section 3.2, we define a Hilbert space of interest, then we provide an overview of the volumetric approach for point cloud compression. Finally, a review of finite Taylor series expansion for function approximation is provided in Section 3.3.

3.1 Hilbert Space

3.1.1 Vector Space

Let us begin with the definition of an important object in our study.

Definition. *Denote by* $\mathbb{F}$ *a field, whose elements* $s \in \mathbb{F}$ *are called* ***scalars****. A* ***vector space*** *over* $\mathbb{F}$ *is a nonempty set* $\mathcal{V}$*, with elements* $\mathbf{v}, \mathbf{u}, \mathbf{w} \in \mathcal{V}$*, and equipped with two operations. The first operation is* ***addition****, defined as a mapping* $+ : \mathcal{V} \times \mathcal{V} \longrightarrow \mathcal{V}$ *that maps two vectors* $\mathbf{v}, \mathbf{u} \in \mathcal{V}$ *to its sum* $\mathbf{u} + \mathbf{v} \in \mathcal{V}$*. The second operation is* ***scalar***

***multiplication**, defined as a mapping $\cdot : \mathbb{F} \times \mathcal{V} \longrightarrow \mathcal{V}$ that maps $(\mathbf{s}, \mathbf{v}) \in \mathbb{F} \times \mathcal{V}$ to a new vector $\mathbf{s} \cdot \mathbf{v} \in \mathcal{V}$. Furthermore, the following properties much also be satisfied, so that the definition of a vector space is established.*

- ***Commutativity of addition**: $\forall \mathbf{v}, \mathbf{u}, \mathbf{w} \in \mathcal{V}$ s.t*

$$\mathbf{u} + (\mathbf{v} + \mathbf{w}) = (\mathbf{u} + \mathbf{v}) + \mathbf{w}. \tag{3.1}$$

- ***Associativity of addition**: $\forall \mathbf{v}, \mathbf{u} \in \mathcal{V}$ s.t*

$$\mathbf{v} + \mathbf{u} = \mathbf{u} + \mathbf{v}. \tag{3.2}$$

- ***Associativity of multiplication**: $\forall \mathbf{v} \in \mathcal{V}$ and $\forall a, b \in \mathbb{F}$ s.t*

$$(\mathbf{a} \cdot \mathbf{b}) \cdot \mathbf{v} = \mathbf{a} \cdot (\mathbf{b} \cdot \mathbf{v}). \tag{3.3}$$

- ***Distributivity of multiplication**: $\forall \mathbf{v}, \mathbf{u} \in \mathcal{V}$ and $\forall a, b \in \mathbb{F}$,*

$$a \cdot (\mathbf{v} + \mathbf{u}) = a \cdot \mathbf{v} + a \cdot \mathbf{u} \tag{3.4}$$

$$(a + b) \cdot \mathbf{v} = a \cdot \mathbf{v} + b \cdot \mathbf{v}. \tag{3.5}$$

- ***Existence of a zero**: $\forall \mathbf{v} \in \mathcal{V}$ then $\exists \mathbf{0} \in \mathcal{V}$ s.t*

$$\mathbf{0} + \mathbf{v} = \mathbf{v} + \mathbf{0} = \mathbf{v}. \tag{3.6}$$

- ***Existence of additive inverses**: $\forall \mathbf{v} \in \mathcal{V}$ then $\exists (-\mathbf{v}) \in \mathcal{V}$ s.t*

$$(-\mathbf{v}) + \mathbf{v} = \mathbf{v} + (-\mathbf{v}) = \mathbf{0}. \tag{3.7}$$

- ***Existence of multiplicative inverses****: $\forall \mathbf{v} \in \mathcal{V}$ then $1 \cdot \mathbf{v} = \mathbf{v}$.*

Definition. *Let $\mathcal{V}$ be a vector space over $\mathbb{F}$, and let $\mathcal{U} \subset \mathcal{V}$ be a subset of $\mathcal{V}$. Then we call $\mathcal{U}$ a **subspace** of $\mathcal{V}$ if $\mathcal{U}$ is a vector space over $\mathbb{F}$ under the same operations that make $\mathcal{V}$ into a vector space over $\mathbb{F}$.*

Definition. *The **subspace spanned** by a nonempty set $\mathcal{S}$ of $\mathcal{V}$ is the set of all linear combinations of vectors from S:*

$$span(\mathcal{S}) = \{r_1\mathbf{v}_1 + ... + r_n\mathbf{v}_n | r_i \in \mathbb{F}, \mathbf{v}_i \in \mathcal{V}\}. \tag{3.8}$$

Definition. *Let $\mathcal{V}$ be a vector space. A non-empty set $\mathcal{S}$ of vectors in $\mathcal{S}$ is **linear independent** if for any distinct vectors $\mathbf{s}_1, .., \mathbf{s}_n \in S$:*

$$a_1 \cdot \mathbf{s}_1 + ... + a_n \cdot \mathbf{s}_n = 0 \Rightarrow a_i = 0 \; \forall i. \tag{3.9}$$

This essentially means that any nonzero vector $\mathbf{v} \in span(\mathcal{S})$ is a unique linear combination of the vectors in $\mathcal{S}$.

Definition. *$\mathcal{S}$ is called a **basis** for $\mathcal{V}$ if $\mathcal{S}$ is linear independent and $span(\mathcal{S}) = \mathcal{V}$.*

In our work, the field $\mathbb{F}$ is basically the real numbers $\mathbb{R}$. Hence, from this point, we use $\mathbb{R}$ in our definitions that the vector space is defined on. Below, we show some examples of the vector space that is used in our works. We will give more details in later sections:

- **Example 1**: $\mathcal{F}$ is the vector space of all volumetric functions $f : \mathbb{R}^3 \longrightarrow \mathbb{R}$. Then $\forall f \in \mathcal{F}$ define an attribute field over $\mathbb{R}^3$. It is trivial that $\mathcal{F}$ satisfies all the requirements of a vector space.

- **Example 2**: $\mathcal{F}_0^{(1)}$ is the vector space of all *constant* functions $c : \mathbb{R}^3 \longrightarrow \mathbb{R}$. Then, it is quite clear that $\mathcal{F}_0^{(1)}$ is a proper subspace of $\mathcal{F}$, and can be denoted as $\mathcal{F}_0^{(1)} \subset \mathcal{F}$. Further, it is also clear that $\mathcal{F}_0^{(1)}$ is spanned by a simple function $f(\mathbf{x}) = 1,\ \forall \mathbf{x} \in \mathbb{R}^3$.

3.1.2 Inner Product, Norm, and Distance

Following our definition of vector space $\mathcal{V}$, we next equip it with an additional function defined on $\mathcal{V} \times \mathcal{V}$, called an *inner product*.

Definition. *Denote by $\mathcal{V}$ a vector space over real number $\mathbb{R}$.* ***Inner product*** *on $\mathcal{V}$ is a mapping $\langle , \rangle : \mathcal{V} \times \mathcal{V} \longrightarrow \mathbb{R}$ that satisfies the following properties:*

- ***Positive definiteness****: For all $\mathbf{v} \in \mathcal{V}$,*

$$\langle \mathbf{v}, \mathbf{v} \rangle \geq 0 \text{ and } \langle \mathbf{v}, \mathbf{v} \rangle = 0 \Leftrightarrow v = 0. \tag{3.10}$$

- ***Symmetry****: For all $\mathbf{v}, \mathbf{u} \in \mathcal{V}$,*

$$\langle \mathbf{v}, \mathbf{u} \rangle = \langle \mathbf{u}, \mathbf{v} \rangle. \tag{3.11}$$

- ***Linearity****: For all $\mathbf{v}, \mathbf{u}, \mathbf{w} \in \mathcal{V}$ and $a, b \in \mathbb{R}$,*

$$\langle a\mathbf{v} + b\mathbf{u}, \mathbf{w} \rangle = a\langle \mathbf{v}, \mathbf{w} \rangle + b\langle \mathbf{u}, \mathbf{w} \rangle. \tag{3.12}$$

A vector space equipped with an inner product is called an inner product space. If $\mathcal{V}$ is an inner product space, the **norm**—often called **length** also—of $\mathbf{v} \in \mathcal{V}$ is defined as below:

$$\|\mathbf{v}\| = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle}. \tag{3.13}$$

Then, based only on the definitions introduced earlier, the following properties are always true for an inner product space with the norm induced by the inner product, these properties are proved in chapter 9, [51]:

Theorem. *Let $\mathcal{V}$ be the norm-induced inner product space.*

- $\|\mathbf{v}\| \geq 0$, *and* $\|\mathbf{v}\| = 0$ *if and only if* $\mathbf{v} = \mathbf{0}$.
- ***Cauchy-Schwarz inequality****: For all $v, u \in \mathcal{V}$,*

$$\langle \mathbf{v}, \mathbf{u} \rangle \leq \|\mathbf{v}\| \, \|\mathbf{u}\| \, . \tag{3.14}$$

- ***Triangle inequality****: For all $\mathbf{v}, \mathbf{u} \in \mathcal{V}$,*

$$\|\mathbf{v} + \mathbf{u}\| \leq \|\mathbf{v}\| + \|\mathbf{u}\| \, . \tag{3.15}$$

It is then straightforward to define the **distance** between any vectors $\mathbf{u}, \mathbf{v} \in \mathcal{V}$:

$$d(\mathbf{v}, \mathbf{u}) = \|\mathbf{v} - \mathbf{u}\| \, . \tag{3.16}$$

Here are the basis properties of distance:

Theorem. *Let $\mathcal{V}$ be a norm-induced inner product space. Then,*

- $d(\mathbf{v}, \mathbf{u}) \geq 0$, *and* $d(\mathbf{v}, \mathbf{u}) = 0$ *if and only if* $\mathbf{u} = \mathbf{v}$.
- ***Symmetry****: If* $\forall \mathbf{u}, \mathbf{v} \in \mathcal{V}$, *then* $d(\mathbf{u}, \mathbf{v}) = d(\mathbf{v}, \mathbf{u})$.
- ***The triangle inequality****:* $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathcal{V}$ *then* $d(\mathbf{v}, \mathbf{u}) \leq d(\mathbf{v}, \mathbf{w}) + d(\mathbf{w}, \mathbf{u})$.

Hence, we can see that with the appropriate inner product equipped, our norm-induced inner product space is enriched with practical properties. We discuss in detail the inner product that we use for our work in Section 3.1.4.

3.1.3 Orthogonality

The vector space equipped with an inner product also allows us to define the concept of orthogonality.

Definition. *Given an inner product space* $\mathcal{V}$.

- $\mathbf{u}, \mathbf{v} \in \mathcal{V}$ *are* ***orthogonal****, written as* $\mathbf{u} \perp \mathbf{v}$*, if* $\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{0}$.
- *Two subsets* $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{V}$ *are orthogonal, written as* $\mathcal{X} \perp \mathcal{Y}$*, if* $\mathbf{x} \perp \mathbf{y}$ *for all* $\mathbf{x} \in \mathcal{X}$ *and* $\mathbf{y} \in \mathcal{Y}$.
- *The* ***orthogonal complement*** *of a subset* $\mathcal{X} \subseteq \mathcal{V}$ *is the set*

$$\mathcal{X}^{\perp} = \{\mathbf{x} \in \mathcal{V} | \mathbf{v} \perp \mathcal{X}\}. \tag{3.17}$$

Then the following result is proved in chapter 9, [51]:

Theorem. *Denote by* $\mathcal{V}$ *an inner product space. Then,*

- *The orthogonal complement* $\mathcal{X}^{\perp}$ *of any subset* $\mathcal{X} \subseteq \mathcal{V}$ *is also a subspace of* $\mathcal{V}$.

- *For any subspace $\mathcal{X} \subseteq \mathcal{V}$, $\mathcal{X} \cap \mathcal{X}^{\perp} = 0$.*

Definition. *A nonempty set $\mathcal{O} = \{\mathbf{u}_i \,|\, i \in \mathcal{K}\}$ of vectors in inner product space is said to be an **orthogonal set** if $\mathbf{u}_i \perp \mathbf{u}_j$ for all $i \neq j$. Additionally, if each vector $\mathbf{u}_i$ is a unit vector, then $\mathcal{O}$ is an **orthonormal set**, i.e., for all $i, j \in \mathcal{K}$,*

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle = \delta_{i,j} \tag{3.18}$$

where $\delta_{i,j}$ is the Kronecker delta function.

Then, the below theorem is showed in [51]

Theorem. *Any orthogonal set of nonzero vectors in $\mathcal{V}$ is linearly independent.*

3.1.4 Hilbert Space

Before we define a Hilbert Space, we first formalize the notion of completeness.

Definition. *A sequence of vectors $\{\mathbf{x}_i\}$ in a norm-induced inner product space $\mathcal{V}$ is a **Cauchy sequence** if for any $\epsilon > 0$, there exists an $N > 0$ such that*

$$n, m > N \Longrightarrow \|\mathbf{x}_n - \mathbf{x}_m\| < \epsilon \tag{3.19}$$

Intuitively, it is clear that any convergent sequence is a Cauchy sequence. We now provide the definition of completeness.

Definition. *Denote by $\mathcal{V}$ a norm-induced inner product space. Then,*

- *$\mathcal{V}$ is said to be **complete** if every Cauchy sequence in $\mathcal{V}$ converges to an element in $\mathcal{V}$.*

- *A subspace $\mathcal{S}$ in $\mathcal{V}$ is* ***complete*** *if every Cauchy sequence in $\mathcal{S}$ converges to an element in $\mathcal{S} \subseteq \mathcal{V}$.*

We can now provide the definition of a Hilbert space as follows.

Definition. *A Hilbert space $\mathcal{H}$ is an inner product space that is complete with respect to the norm defined by the inner product.*

Specifically, we consider one example of Hilbert space that is used in our work. $\mathcal{L}^2(\mathcal{X})$, also known as the **Lebesgue space** with $p = 2$, is a norm-induced inner product vector space of *functions* on a measurable space. In our work, this measurable space is the set of points locations in $\mathbb{R}^3$, $\mathcal{X} = \{\mathbf{x}_i\}_{i=0}^N$, and the measure, $\mu(\mathcal{S}) = |\mathcal{S}|$, is basically just counting the number of elements for any subset $\mathcal{S} \subseteq \mathcal{X}$. Then, the inner product of this particular vector space is defined as below: $\forall f, g \in \mathcal{L}^2(\mathcal{X})$,

$$\langle f, g \rangle = \int_{\mathcal{X}} f(\mathbf{x}) \cdot g(\mathbf{x}) d\mu(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{X}} f(\mathbf{x}_i) \cdot g(\mathbf{x}_i). \tag{3.20}$$

Then, the norm is

$$\|f\| = \left[\sum_{\mathbf{x}_i \in \mathcal{X}} f^2(\mathbf{x}_i) \right]^{\frac{1}{2}}. \tag{3.21}$$

The above norm is known as ℓ_2-norm in traditional $\mathbb{R}^d$ space and is extensively used in statistics. In our work, it defines how two functions f and g are considered equivalent, which only happens when $f(\mathbf{x}_i) = g(\mathbf{x}_i)\ \forall\ \mathbf{x}_i \in \mathcal{X}$. Hence, the notion of "equal" in our functional space is subject to the input point cloud geometry $\mathcal{X}$.

The proof that the above inner product space is a valid Hilbert space is outside

the scope of this thesis. We recommend readers to consult details in [51, 52]. Our work relies heavily on the derived properties of a Hilbert space; we devote the entire Section 3.2 to provide important details.

3.1.5 Projection

Theorem. *Let $\mathcal{V}$ be an inner product space, and let $\mathcal{S}$ be a complete subspace of $\mathcal{V}$. Then for any $\mathbf{x} \in \mathcal{V}$, the best approximation to $\mathbf{x}$ in $\mathcal{S}$, denoted as $\hat{\mathbf{x}} \in \mathcal{S}$, is defined as*

$$\|\mathbf{x} - \hat{\mathbf{x}}\| = \inf_{\mathbf{s} \in \mathcal{S}} \|\mathbf{x} - \mathbf{s}\| \tag{3.22}$$

and it will be a unique vector for which $(\mathbf{x} - \hat{\mathbf{x}}) \perp \mathcal{S}$.

We restate the projection theorem [51] below.

Theorem. *(**The projection theorem**) If $\mathcal{S}$ is a complete subspace of a norm-induced inner product space $\mathcal{V}$, then*

$$\mathcal{V} = \mathcal{S} \oplus \mathcal{S}^{\perp}. \tag{3.23}$$

In particular, if $\mathcal{S}$ is a closed subspace of a Hilbert space $\mathcal{H}$, then

$$\mathcal{H} = \mathcal{S} \oplus \mathcal{S}^{\perp}. \tag{3.24}$$

We take a closer look at best approximations to any vector $\mathbf{x}$ from within a subspace $\mathcal{S}$ of a Hilbert space $\mathcal{H}$. Suppose that $\mathcal{O} = \{\mathbf{u}_i\}_{i=1}^{K}$ is an orthonormal set in

$\mathcal{H}$. Then, the **Fourier expansion** of any $\mathbf{x} \in \mathcal{H}$, with respect to $\mathcal{O}$, is given by

$$\hat{\mathbf{x}} = \sum_{i=1}^{K} \langle \mathbf{x}, \mathbf{u}_i \rangle \mathbf{u}_i \tag{3.25}$$

and $\langle \mathbf{x}, \mathbf{u}_i \rangle$ is called the **Fourier coefficient** of $\mathbf{x}$ with respect to $\mathbf{u}_i$. Then, we can easily see that

$$\langle \mathbf{x} - \hat{\mathbf{x}}, \mathbf{u}_i \rangle = 0, \qquad \forall \mathbf{u}_i \in \mathcal{O}. \tag{3.26}$$

Hence, $\hat{\mathbf{x}}$ is the best approximation to $\mathbf{x}$ in the subspace spanned by $\mathcal{O}$. This is the basic approximation scheme we will use in Section 3.2.

3.2 Volumetric Approach for Point Cloud Attribute Compression

This section introduces the volumetric approach for point cloud attribute compression, which was first formalized in [42, 45, 10].

3.2.1 3D Point Cloud Attribute Compression

In a point cloud (PC) representing a 3D object, each volumetric element, often called a *voxel*, is associated with its geometrical location in $\mathbb{R}^3$ and its visual attributes. A voxel exists only in 2 states: occupied or unoccupied. An unoccupied voxel is considered a void without any properties and is visually transparent. Occupied voxels are associated with geometry (their location in space) and color.

The PC compression problem is to find and compress a point cloud representation, which consists of a list of occupied voxels $\{\mathbf{v}_i = (\mathbf{x}_i, \mathbf{c}_i)\}_{i=1}^{N}$, each being described by the locations of the points, $\mathbf{x}_i = [x_i, y_i, z_i]$, and the attributes attached to each point,

$\mathbf{c}_i = [Y_i, U_i, V_i]$. The compression of point locations is known as *geometry compression*, and the compression of point attributes given that the point locations are available at the decoder is called *attribute compression*. We consider *attribute compression* to be the main focus of our work and assume point locations are given at the encoder and decoder.

PC compression problem is different from traditional image compression because the geometry of images is always available at the decoder, whereas it is not the case for PC compression. Therefore, we could consider image compression as a special case of an *attribute compression* problem where the point locations are the coordinates of the pixels, which are implicitly available at both the encoder and decoder.

Specifically, we pursue a *volumetric approach*: the real vector-valued function $f : \mathbb{R}^d \mapsto \mathbb{R}^r$ is *volumetric* if $d = 3$ (or *hyper-volumetric* if $d > 3$). A point cloud is clearly a collection of samples of a volumetric function where $\mathbf{c}_i = f(\mathbf{x}_i)$. For simplicity, we assume scalar attribute ($r = 1$), so $\mathbf{c}_i$ is just an array of size 1.

3.2.2 Octree Representation

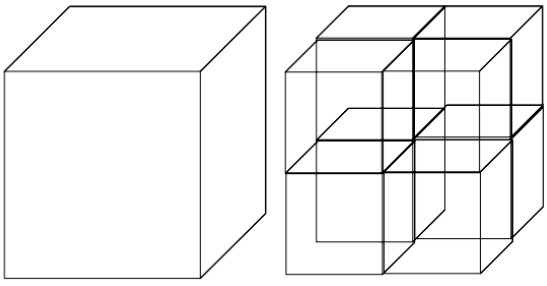


Figure 3.1: Example of an octree, where a 3D cube of size W is divided into 8 smaller sub-cubes. These sub-cubes are further divided by the same manner, and after L levels of subdivision, we reach unit cubes of size $2^{-L}W$, which we called voxels.

An octree can be seen as a 3D extension of a 1D binary tree. A visualization of an octree is shown in Figure 3.1, illustrating a cube of size $W \times W \times W$, big enough to include all points in a point cloud—this cube can be seen as the root of the tree at level 0. The cube is then partitioned into smaller sub-cubes of size $W/2 \times W/2 \times W/2$; this is the next level of the tree. This process can be repeated for L levels, resulting in 2^{3L} unit cubes of dimension $2^{-L}W \times 2^{-L}W \times 2^{-L}W$. Note that at each level of the octree, not all cubes are occupied.

This octree presentation leads to an efficient method to encode the geometry of the point cloud [2, 53, 54], where for level l, 8 bits are used only for an occupied cube in level l to signal the occupation of the sub-cubes in level $l+1$ (because if a cube is unoccupied, then all its children will also be unoccupied, hence requiring no further bits).

Assuming that it is required to have L levels to reach the desired resolution for a given point cloud, where each voxel only has one occupied point. Denote by $\mathcal{B}_l$ the set of all available cubes at level l,

$$\mathcal{B}_l = \{2^{-l} \cdot ([0,1)^3 + \mathbf{n}) | \mathbf{n} \in \mathbb{Z}^3\} \tag{3.27}$$

where $2^{-l}\mathbf{n}$ is the location of the root corner for each block at level l. Denote by $\hat{\mathcal{B}}_l \subseteq \mathcal{B}_l$ the set of occupied cubes in each level; these occupied blocks (or cubes) define the support regions where the volumetric function $f(\mathbf{x}_i) = \mathbf{c}_i$ is approximated.

3.2.3 Volumetric Function Space with B-Spline

A cardinal B-Spline function of order p is a function on the real line $f_{\phi^{(p)}} : \mathbb{R} \to \mathbb{R}$ that is generated by a set of B-Spline basis functions of order p,

$$f_{\phi^{(p)}}(x) = \sum_{n \in \mathbb{Z}} F_n^{(p)} \cdot \phi^{(p)}(x - n) \tag{3.28}$$

where $n \in \mathbb{Z}$ is the integer shift of the B-Spline, and $F_n^{(p)}$ is the coefficient corresponding to each B-Spline basis function. The B-spline basis function with order $p = 1$ is defined as

$$\phi^{(1)}(x) = \begin{cases} 1 & x \in [0, 1) \\ 0 & \text{otherwise} \end{cases}. \tag{3.29}$$

Then, the B-spline basis function with order $p > 1$ is derived as follow,

$$\phi^{(p)}(x) = \int \phi^{(1)}(x) \cdot \phi^{(p-1)}(x - t)dt. \tag{3.30}$$

Another expression for $p > 1$ can be derived by the Cox–de Boor recursion formula [55]:

$$\phi^{(p)}(x) = \frac{x}{p-1}\phi^{(p-1)}(x) + \frac{p-x}{p-1}\phi^{(p-1)}(x-1). \tag{3.31}$$

It is clear that $\phi^{(p)}(x)$ is p-fold convolution of $\phi^{(1)}(x)$ with itself. In addition, $\phi^{(p)}(x)$ has a bounded support region, which means that $\phi^{(p)}(x) = 0,\ \forall x \notin [0, p]$. Figure 3.2 shows two examples of the B-Spline basis function for $p = 1$ and $p = 2$.

Similarly, a volumetric cardinal B-spline of order p is a volumetric function $f_{\phi^{(p)}}$:

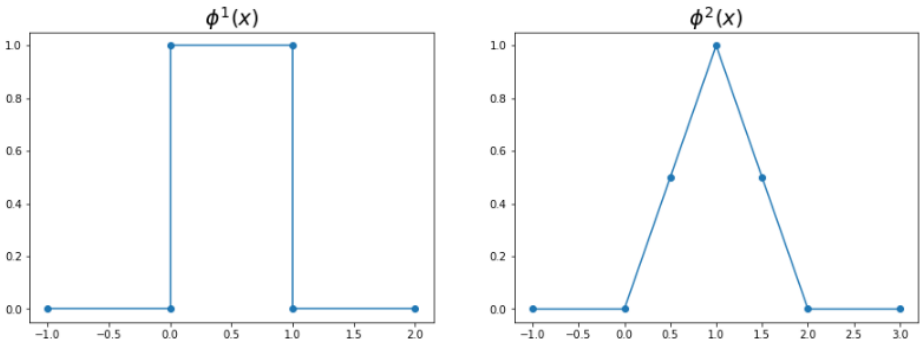


Figure 3.2: Example of B-spline basis functions p=1 and p=2

$\mathbb{R}^3 \to \mathbb{R}$ that is generated by a set of volumetric B-Spline basis functions of order p defined on $\mathbb{R}^3$, where $\phi^{(p)}(\mathbf{x}) = \phi^{(p)}(x, y, z) = \phi^{(p)}(x) \cdot \phi^{(p)}(y) \cdot \phi^{(p)}(z)$:

$$f_{\phi^{(p)}}(\mathbf{x}) = \sum_{\mathbf{n} \in \mathbb{Z}^3} F_n^p \cdot \phi^{(p)}(\mathbf{x} - \mathbf{n}). \tag{3.32}$$

We see that the volumetric cardinal B-spline function $f_{\phi^{(p)}}(\mathbf{x})$ consists of multiple segments of 3D polynomial of degree $p-1$ over each shifted unit cube $([0,1]^3 + \mathbf{n})$. This is the basic building block of the volumetric approach using the B-spline basis function. We will fit various volumetric cardinal B-spline functions at different scales to the given input point cloud attributes $\{\mathbf{x}_i, \mathbf{c}_i = f(\mathbf{x}_i)\}$, which are considered as samples of the volumetric function f. The process is discussed in detail in the next section.

3.2.4 Multiresolution Approximation

The input volumetric function f can be approximated at different scales depending on how the set of B-spline basis functions is constructed. First, we define the elementary

basis function as a centralized volumetric B-spline function,

$$\phi_{0,\mathbf{0}}^{(p)}(\mathbf{x}) = \phi^{(p)}(\mathbf{x} - \mathbf{n}_0) \tag{3.33}$$

which is the volumetric B-spline function shifted to the origin $\mathbf{0}$. To simplify notations, we omit the degree p and use $\phi_{0,\mathbf{0}}$ to indicate the elementary basis function. We note that

- For $p = 1$, $\phi_{0,\mathbf{0}}^{(1)}(\mathbf{x}) = 0,\ \forall \mathbf{x} \notin [0,1]^3$.
- For $p = 2$, $\phi_{0,\mathbf{0}}^{(2)}(\mathbf{x}) = 0,\ \forall \mathbf{x} \notin [-1,1]^3$.

Next, we scale and shift $\phi_{0,\mathbf{0}}$ to different scales l and centers $\mathbf{n} \in \mathbb{Z}^3$,

$$\phi_{l,\mathbf{n}}^{(p)}(\mathbf{x}) = \phi_{0,\mathbf{0}}^{(p)}(2^{-l}(\mathbf{x} - \mathbf{n})). \tag{3.34}$$

Then, given that we have $\mathcal{X} = \{\mathbf{x}_i\}_{i=0}^{N}$ as the input point locations, we define $\mathcal{N}_l$ as the set of centers at a particular scale l, where $\phi_{l,\mathbf{n}}(\mathbf{x}_i)$ is active for some $\mathbf{x}_i$,

$$\mathcal{N}_l = \left\{ \mathbf{n} \in \mathbb{Z}^3 \mid \exists \mathbf{x}_i \in \mathcal{X} \ \text{ s.t } \phi_{l,\mathbf{n}}(\mathbf{x}_i) \neq 0 \right\}. \tag{3.35}$$

The size of $\mathcal{N}_l$ is $|\mathcal{N}_l| = N_l$. Note that N_l depends greatly on degree p of the B-spline elementary basis function:

- For $p = 1$, set $\mathcal{N}_l$ exactly corresponds to the root corner $\mathbf{n}$ of each occupied block $([0,1]^3 + \mathbf{n}) \in \hat{\mathcal{B}}_l$, which we defined earlier in Section 3.2.2.
- For $p = 2$, set $\mathcal{N}_l$ exactly corresponds to the union of all corners for all occupied blocks in $\hat{\mathcal{B}}_l$, because $\phi_{l,\mathbf{n}}^{(2)}(\mathbf{x}) \neq 0$ for all $\mathbf{x} \in ([-1,1]^3 + \mathbf{n})$. Hence, with higher

order of B-spline functions, the set $\mathcal{N}_l$ contains more elements, because the size of the support region of each B-spline function is expanded.

Given that $\mathcal{N}_l$ is well defined for a given elementary basis function $\phi^{(p)}_{0,\mathbf{0}}(\mathbf{x})$, we next consider the relationship between $\mathcal{N}_l$ and $\mathcal{N}_{l+1}$. Denote by $\mathbf{n_l} \in \mathcal{N}_l$ a location where the B-spline function at scale l is active. Then, there must exist a finite set $\mathcal{K} \subset \mathbb{Z}^3$ such that $2\mathbf{n_l} + \mathbf{k} \in \mathcal{N}_{l+1}$ and $\mathbf{k} \in \mathcal{K}$. This leads to the following relationship between sets of basis functions at two consecutive scales l and $l+1$ (also described in [56]):

$$\phi^{(p)}_{l,\mathbf{n}_l} = \sum_{\mathbf{k}\in\mathbb{Z}^3} a_{\mathbf{k}} \phi^{(p)}_{l+1,2\mathbf{n_l}+\mathbf{k}} = \sum_{(\mathbf{n_{l+1}}-2\mathbf{n_l})\in\mathbb{Z}^3} a_{\mathbf{n_{l+1}}-2\mathbf{n_l}} \phi^{(p)}_{l+1,\mathbf{n_{l+1}}}. \tag{3.36}$$

Since both $\phi_{l,\mathbf{n}_l}$ and $\phi_{l+1,\mathbf{n_{l+1}}}$ have finite support regions, the set of coefficients $\{a_{\mathbf{k}}\}$ is also finite. Interestingly, $\{a_{\mathbf{k}}\}$ is also independent of l and $\mathbf{n}_l$, because our basis functions are the scaled (by a factor of 2^l) and shifted versions of the same elementary basis function at each scale. (3.36) is known as the *two-scale equation* for B-spline function [57].

Given that the basis functions $\phi^{(p)}_{l,\mathbf{n}_i}$ at each scale l and centered at $\mathbf{n}_i \in \mathcal{N}_l$ are defined, the definition of the approximation subspace is the set of all possible linear combinations of the basis functions $\{\phi^{(p)}_{l,\mathbf{n}_i}\}_{i=1}^{N_l}$,

$$\mathcal{F}_l = \left\{ f_l \mid f_l(\mathbf{x}) = \sum_{\mathbf{n}_i \in \mathcal{N}_l} F_{l,\mathbf{n}_i} \phi^{(p)}_{l,\mathbf{n}_i}(\mathbf{x}) \right\} \tag{3.37}$$

where $F_{l,\mathbf{n}_i}$ is the coefficient corresponding to the basis function $\phi^{(p)}_{l,\mathbf{n}_i}$.

As mentioned in Section 3.1.1, we define $\mathcal{F}$ as the vector space of all volumetric

function $f : \mathbb{R}^3 \to \mathbb{R}$. Because each basis function $\phi^{(p)}_{l,\mathbf{n}_i}$ at scale l is a linear combination of some basis functions in at scale $l+1$ due to (3.36), we have a nested sequence of subspaces that represent different approximations at different scales l, *i.e.*,

$$\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_l \subseteq \mathcal{F}_{l+1} \subseteq \cdots \subseteq \mathcal{F}_L \subseteq \mathcal{F}. \tag{3.38}$$

For notation convenience, denote the set of basis functions at each scale l by a row-vector of N_l basis functions and omit the degree p, $\mathbf{\Phi}_l = [\phi_{l,\mathbf{n}_i}]$, where $\mathbf{n}_i \in \mathcal{N}_l$. Then, for $f_l \in \mathcal{F}_l$, we express f_l as a linear combination of the basis functions by the column vector of coefficients $\mathbf{F}_l = [F_{l,\mathbf{n}_i}] \in \mathbb{R}^{N_l}$,

$$f_l = \sum_{\mathbf{n}_i \in \mathcal{N}_l} F_{l,\mathbf{n}_i} \cdot \phi_{l,\mathbf{n}_i}(\mathbf{x}) = \mathbf{\Phi}_l \mathbf{F}_l. \tag{3.39}$$

Then, (3.36) can be expressed as

$$\mathbf{\Phi}_l = \mathbf{\Phi}_{l+1} \mathbf{A}_l^T \tag{3.40}$$

where $\mathbf{A}_l$ is a $N_l \times N_{l+1}$ matrix of coefficients which expresses each basis function at scale l in term of basis functions at scale $l+1$.

Denote by f_l^* and f_{l+1}^* the best approximations of f in $\mathcal{F}_l$ and in $\mathcal{F}_{l+1}$, respectively. Then, by the approximation theorem in Section 3.1.4, we know that $(f - f_{l+1}^*) \perp \mathcal{F}_l \subseteq \mathcal{F}_{l+1}$, so that for all $f_l \in \mathcal{F}_l$, by Pythagorean theorem, we have

$$\|f - f_l\|^2 = \|f - f_{l+1}^*\|^2 + \|f_{l+1}^* - f_l\|^2. \tag{3.41}$$

Since $f_l^* = \arg\min_{f_l \in \mathcal{F}_l} \|f - f_l\|^2$ by definition, $f_l^* = \arg\min_{f_l \in \mathcal{F}_l} \|f_{l+1}^* - f_l\|^2$. This proves that f_l^* is also the best approximation of f_{l+1}^* in $\mathcal{F}_l$. This means that finding the best approximation of f in the nested sequence of subspaces can be done sequentially from the highest level L to the lowest level 0.

Finding the best approximation can be formulated as solving the least square problem:

$$\mathbf{F}_l^* = \arg\min_{\mathbf{F}_l} \|f - \mathbf{\Phi}_l \mathbf{F}_l\| \tag{3.42}$$

where $f_l^* = \mathbf{\Phi}_l \mathbf{F}_l^*$. The solution to the above problem is the solution to the following system of linear equations, also known as the *normal equation*:

$$(\mathbf{\Phi}_l^T \mathbf{\Phi}_l)\mathbf{F}_l^* = (\mathbf{\Phi}_l^T f) \Longleftrightarrow \mathbf{F}_l^* = (\mathbf{\Phi}_l^T \mathbf{\Phi}_l)^{-1}(\mathbf{\Phi}_l^T f) \tag{3.43}$$

where $(\mathbf{\Phi}^T \mathbf{\Phi})$, often known as *Gram matrix*, is a $N_l \times N_l$ matrix whose entries are the inner products (defined in section 3.1.4) between the basis functions $[\langle \phi_{l,\mathbf{n}_i}, \phi_{l,\mathbf{n}_j} \rangle]$, and $(\mathbf{\Phi}^T f)$ is a vector of size $N_l \times 1$ with entries $[\langle \phi_{l,\mathbf{n}_i}, f \rangle]$.

3.2.5 Projection residual representation

Denote by $\mathcal{G}_l$ the orthogonal complement subspace of $\mathcal{F}_l$ in $\mathcal{F}_{l+1}$, so that $\mathcal{G}_l \perp \mathcal{F}_l$ and $\mathcal{F}_{l+1} = \mathcal{F}_l \oplus \mathcal{G}_l$. Thus, we have the sequence of orthogonal complement subspaces as

$$\mathcal{F}_L = \mathcal{F}_0 \oplus \mathcal{G}_1 \oplus \mathcal{G}_2 \oplus \cdots \oplus \mathcal{G}_{L-1}. \tag{3.44}$$

Based on the approximation theorem, the residual of the approximation for each

level, $g_l^* = f_{l+1}^* - f_l^*$, belongs to the subspace $\mathcal{G}_l$. Denote by $\mathbf{\Psi}_l$ the basis functions for the subspace $\mathcal{G}_l$. Thus, any function $g_l \in \mathcal{G}_l$ can be represented as a linear combination $g_l = \mathbf{\Psi}_l G_l$. Because $\mathbf{\Psi}_l \perp \mathbf{\Phi}_l$ and $\text{span}(\mathbf{\Psi}_l) \in \mathcal{F}_{l+1}$, $\mathbf{\Psi}_l$ can be expressed as a linear combination of $\mathbf{\Phi}_{l+1}$ such that $\mathbf{\Psi}_l^T \mathbf{\Phi}_l = 0$. For convenience, denote by $\mathbf{Z}_l$ a $(N_{l+1} - N_l) \times N_{l+1}$ matrix of coefficients that satisfy the following,

$$\mathbf{\Psi}_l = \mathbf{\Phi}_{l+1} \mathbf{Z}_l^T \tag{3.45}$$

$$\text{s.t.} \quad \mathbf{\Psi}_l^T \mathbf{\Phi}_l = \mathbf{0}. \tag{3.46}$$

Given $\mathbf{\Psi}_l$, the coefficients G_l^* to represent the residual of the approximation can be calculated by solving the normal equation,

$$\mathbf{G}_l^* = (\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-1} \mathbf{\Psi}_l^\top g_l^*. \tag{3.47}$$

In Chapter 4, we discuss in detail a possible operation $\mathbf{Z}_l$ that satisfies (3.46), and methods to approximate both $\mathbf{F}_l^*$ and $\mathbf{G}_1^*$, so that given the finest resolution of the point cloud at level L, we can decompose it into multi-level of resolution as

$$f_L^* = f_0^* + g_0^* + \cdots + g_{L-1}^* \tag{3.48}$$

where each function is expressed by a set of coefficients $\mathbf{F}_0^*, \mathbf{G}_1^*, \cdots, \mathbf{G}_{L-1}^*$.

3.3 Taylor Series Expansion for Matrix Inverse Approximation

We first revisit finite Taylor series expansion [58] to approximate a real function $f : \mathbb{R} \mapsto \mathbb{R}$,

$$f(x) \approx f(a) + \sum_{k=1}^{K} \frac{f^{(k)}(a)}{k!}(x-a)^k. \tag{3.49}$$

We use the above to approximate the inverse function $f_{inv}(x) = x^{-1}$ and the inverse square root function $f_{sqrtinv}(x) = x^{-\frac{1}{2}}$,

$$\begin{aligned} \hat{f}_{inv}(x,a) &= a^{-1} + \sum_{k=1}^{K} a^{-k}(a-x)^k \\ \hat{f}_{sqrtinv}(x,a) &= a^{-\frac{1}{2}} + \sum_{k=1}^{K} \frac{1...(2k-1)}{2^k k!} a^{-\frac{1}{2}-k}(a-x)^k. \end{aligned} \tag{3.50}$$

By letting $\mu = \frac{1}{2a}$, we obtain the following instead,

$$\begin{aligned} \hat{f}_{inv}(x,\mu) &= 2\mu \left[1 + \sum_{k=1}^{K} (1-2\mu x)^k\right] \\ \hat{f}_{sqrtinv}(x,\mu) &= \sqrt{2\mu}\left[1 + \sum_{k=1}^{K} \frac{1...(2k-1)}{2^k k!}(1-2\mu x)^k\right]. \end{aligned} \tag{3.51}$$

Any invertible, positive definite and symmetric square matrix $\mathbf{X} \in \mathbb{R}^{d\times d}$ can be eigen-decomposed as $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, and the inverse and square root inverse of $\mathbf{X}$ can be exactly calculated as

$$\begin{aligned} \mathbf{X}^{-1} &= \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^\top \\ \mathbf{X}^{-\frac{1}{2}} &= \mathbf{U}\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^\top. \end{aligned} \tag{3.52}$$

Using this fact, we use the finite Taylor series as the approximation of $\hat{f}_{inv} \approx x^{-1}$

and $\hat{f}_{sqrtinv} \approx x^{-\frac{1}{2}}$ on $\mathbf{\Lambda}$ instead and re-derive an approximation procedure for inverse and square root inverse, which does not require any eigen-decomposition, *i.e.*,

$$\begin{aligned}
\mathbf{X}^{-1} &\approx \mathbf{U}\hat{f}_{inv}(\mathbf{\Lambda}, \mu)\mathbf{U}^T \\
&\approx 2\mu\mathbf{U}\left[1 + \sum_{k=1}^{K}(1 - 2\mu\mathbf{\Lambda})^k\right]\mathbf{U}^T \\
&\approx 2\mu\left[\mathbf{I} + \sum_{k=1}^{K}(\mathbf{I} - 2\mu\mathbf{X})^k\right].
\end{aligned} \tag{3.53}$$

Then, given any vector $\mathbf{v} \in \mathbb{R}^d$, $\mathbf{X}^{-1}\mathbf{v}$ can be approximated as

$$\mathbf{X}^{-1}\mathbf{v} \approx 2\mu\left[\mathbf{v} + \sum_{k=1}^{K}(\mathbf{I} - 2\mu\mathbf{X})^k\mathbf{v}\right]. \tag{3.54}$$

We use the same method to approximate the square root inverse of $\mathbf{X}$,

$$\mathbf{X}^{-\frac{1}{2}}\mathbf{v} \approx \sqrt{2\mu}\left[\mathbf{v} + \sum_{k=1}^{K}\frac{1...(2k-1)}{2^k k!}(\mathbf{I} - 2\mu\mathbf{X})^k\mathbf{v}.\right] \tag{3.55}$$

(3.54) and (3.55) can be expressed simply as re-applying the *same* matrix multiplication multiple times, where the coefficients for the terms are the coefficients of the Pth order Taylor expansion.

For (3.54) and (3.55) to converge, we need $(\mathbf{I} - 2\mu\mathbf{X})^k \to 0$ as $k \to \infty$. Hence, μ must be set to an appropriate value so that the spectral radius of $\mathbf{L} = \mathbf{I} - 2\mu\mathbf{X}$ is strictly less than 1. This can be accomplished by tuning μ as a hyper-parameter, so that it satisfies $\mu < \frac{1}{2*\lambda_{\max}(\mathbf{X})}$, where $\lambda_{\max}(\mathbf{X})$ is the spectral radius of $\mathbf{X}$.

Chapter 4

Proposed Compression Framework

This chapter provides details of our core coding framework, implemented as a feed-forward network. Section 4.1 introduces a nested sequence of subspaces $\mathcal{F}_0^{(2)} \subseteq \cdots \subseteq \mathcal{F}_L^{(2)}$ constructed using B-spline functions of order $p = 2$ as basis functions. Then, Section 4.2 shows how to project a volumetric function f onto the nested subspace $\mathcal{F}_l^{(2)}$ and how to represent the projected function, f_l^*, using coefficients $\mathbf{F}_l^*$ of the basis functions. In Section 4.3, we discuss how to calculate coefficients $\mathbf{G}_l^*$ to represent the residual of the projection, $g_l^* = f_{l+1}^* - f_l^*$. Section 4.4 introduces orthonormalization operators to transform all coefficients into independent coefficients via orthonormal basis functions. Finally, Section 4.5 combines all operations into one feedforward network implementation.

4.1 Nested sequence of subspace $\mathcal{F}_l$

We begin by re-introducing the definition of the vector space $\mathcal{F}$ of all volumetric functions $f : \mathbb{R}^3 \to \mathbb{R}$, assuming that our input point cloud has only one attribute. A point cloud can be viewed as a sampling of such volumetric function, $\{\mathbf{x}_i, \mathbf{c}_i = [f(\mathbf{x}_i)]\}$.

The inner product of our functional space can be defined as done in Section 3.1.4,

$$\langle f, g \rangle = \int_{\mathcal{X}} f(\mathbf{x}) \cdot g(\mathbf{x}) d\mu(\mathbf{x}) = \sum_{\mathbf{x}_i \in \mathcal{X}} f(\mathbf{x}_i) \cdot g(\mathbf{x}_i) \tag{4.1}$$

where $\mathcal{X}$ is the input geometric locations of 3D points in $\mathbb{R}^3$. Then the norm operation is defined as

$$\|f\| = \left[\sum_{\mathbf{x}_i \in \mathcal{X}} f^2(\mathbf{x}_i) \right]^{\frac{1}{2}} \tag{4.2}$$

which is also known as the ℓ_2-norm. This also defines the notion of equivalent functions:

$$f = g \Leftrightarrow f(\mathbf{x}_i) = g(\mathbf{x}_i) \; \forall \; \mathbf{x}_i \in \mathcal{X}. \tag{4.3}$$

Given that the functional space is defined, we next construct our subspace $\mathcal{F}_l \subseteq \mathcal{F}$. As done in Section 3.2, we use the centered volumetric B-spline function order $p = 2$ as the elementary basis function. We then shift it by $\mathbf{n} \in \mathbb{Z}^3$ and scale it by a factor of 2^{-l}:

$$\phi^{(2)}_{l,\mathbf{n}}(\mathbf{x}) = \phi^{(2)}_{0,\mathbf{0}}(2^{-l}(\mathbf{x} - \mathbf{n})). \tag{4.4}$$

Then, for each scale l (or level l of the octtree), we have a set of B-spline functions at each corner of the cubes in the octree representation. Because the B-spline functions have order $p = 2$, $\phi^{(2)}_{l,\mathbf{n}}(\mathbf{x}) > 0$ only in a finite region $([-1, 1]^3 + \mathbf{n})$ for each corner $\mathbf{n}$.

Hence, given that $\mathcal{N}_l$ is the set of corners such that

$$\mathcal{N}_l = \left\{ \mathbf{n} \in \mathbb{Z}^3 \mid \exists \mathbf{x}_i \in \mathcal{X} \text{ s.t } \phi^{(2)}_{l,\mathbf{n}}(\mathbf{x}_i) \neq 0 \right\}, \tag{4.5}$$

we are now able to construct the basis functions for our subspace $\mathcal{F}_l$ as a row vector $\mathbf{\Phi}_l = [\phi^{(2)}_{l,\mathbf{n}} \mid \mathbf{n} \in \mathcal{N}_l]$. In addition, based on the two-scale equation (3.36), we can express the basis functions at level l using basis functions at level $l+1$ as follow:

$$\mathbf{\Phi}_l = \mathbf{\Phi}_{l+1}\mathbf{A}_l^T \tag{4.6}$$

where the coefficients of matrix $\mathbf{A}_l = [a_{ij}]$ are calculated as

$$a_{ij} = \begin{cases} 2^{-|\mathbf{k}|} & \text{if } \mathbf{k} \in \{-1, 0, 1\}^3 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

with $\mathbf{k} = \mathbf{n}_{l+1,j} - 2\mathbf{n}_{l,i}$ and $|\mathbf{k}|$ is the conventional ℓ_1-norm. Matrix $\mathbf{A}_l$ can be seen as a moving window of size $3 \times 3 \times 3$ and can be implemented as a convolution layer well-known in deep learning methods.

Based on (4.6), we have

$$\mathbf{\Phi}_l^T\mathbf{\Phi}_l = \mathbf{A}_l\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1}\mathbf{A}_l^T \tag{4.8}$$

$$\mathbf{\Phi}_l^T\mathbf{\Phi}_{l+1} = \mathbf{A}_l\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1} \tag{4.9}$$

$$\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_l = \mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1}\mathbf{A}_l^T. \tag{4.10}$$

As we know, $\mathbf{\Phi}_l^T\mathbf{\Phi}_l$ is a sparse $N_l \times N_l$ matrix because each basis function only has a finite support region. Specifically, each row of $\mathbf{\Phi}_l^T\mathbf{\Phi}_l$ only has finite non-zeros

entries—fewer than 27 non-zeros entries when $p = 2$. Hence, equation (4.8) can be implemented as a space-invariant $27 \times 27 \times 27$ sparse hyper-convolution operation applied on a tensor of size $N_l \times 27$, where each row characterize the inner product of one basis functions with its local neigbors.

Assume that there is a sufficiently fine resolution L such that the ith basis function $\phi_{L,\mathbf{n}_i}$ is 1 on $\mathbf{x}_i$ and 0 on $\mathbf{x}_j$ for $j \neq i$, namely $\phi_{L,\mathbf{n}_i}(\mathbf{x}_j) = \delta_{ij}$ so that we have $\mathbf{\Phi}_L^T \mathbf{\Phi}_L = \mathbf{I}$. Then, based on (4.8), $\mathbf{\Phi}_l^T \mathbf{\Phi}_l$ at each level can be calculated efficiently and represented as a sparse matrix.

4.2 Low-pass coefficients $\mathbf{F}_l^*$

Beginning at the highest level of detail L, we have $\mathbf{\Phi}_L^T \mathbf{\Phi}_L = \mathbf{I}$, and hence by the normal equation (3.43), we have $\mathbf{F}_L^* = (\mathbf{\Phi}_L^T \mathbf{\Phi}_L)^{-1} \mathbf{\Phi}_L^T f = [y_i]$. Moving to $l < L$,

$$(\mathbf{\Phi}_l^T \mathbf{\Phi}_l)\mathbf{F}_l^* = \mathbf{\Phi}_l^T f_{l+1}^* \tag{4.11}$$

$$= \mathbf{\Phi}_l^T \mathbf{\Phi}_{l+1} \mathbf{F}_{l+1}^* \tag{4.12}$$

$$= \mathbf{A_l} (\mathbf{\Phi}_{l+1}^T \mathbf{\Phi}_{l+1}) \mathbf{F}_{l+1}^*. \tag{4.13}$$

By letting $\tilde{\mathbf{F}}_l^* = (\mathbf{\Phi}_l^T \mathbf{\Phi}_l)\mathbf{F}_l^*$, which we call un-normalized low-pass coefficients, we can easily calculate $\tilde{\mathbf{F}}_l^*$ for all levels l as

$$\tilde{\mathbf{F}}_l^* = \mathbf{A}_l \tilde{\mathbf{F}}_{l+1}^* \tag{4.14}$$

which, as defined earlier, $\mathbf{A}_l$ is an ordinary sparse convolution, then $\mathbf{F}_l^*$ can be computed by an normalization operation $(\mathbf{\Phi}_l^T \mathbf{\Phi}_l)^{-1}$:

$$\mathbf{F}_l^* = (\mathbf{\Phi}_l^T \mathbf{\Phi}_l)^{-1} \tilde{\mathbf{F}}_l^*. \tag{4.15}$$

We will see later how to compute this using a feedforward network in Section 4.5.

4.3 High-pass coefficients $\mathbf{G}_l^*$

To code the residual functions $g_0^*, \ldots, g_{L-1}^*$, they must be represented in a basis. Since $g_l^* \in \mathcal{G}_l \subset \mathcal{F}_{l+1}$, one possible basis in which to represent g_l^* is the basis $\mathbf{\Phi}_{l+1}$ for $\mathcal{F}_{l+1}$. In this basis, the coefficients for $g_l^* = f_{l+1}^* - f_l^* = \mathbf{\Phi}_{l+1}\mathbf{F}_{l+1}^* - \mathbf{\Phi}_l \mathbf{F}_l^*$ are given by the normal equation (3.43) and expression (4.9):

$$(\mathbf{\Phi}_{l+1}^T \mathbf{\Phi}_{l+1})^{-1} \mathbf{\Phi}_{l+1}^T g_l^* = \mathbf{F}_{l+1}^* - \mathbf{A}_l^T \mathbf{F}_l^* \triangleq \delta \mathbf{F}_{l+1}^*. \tag{4.16}$$

However, the number of coefficients in this representation is N_{l+1}, whereas the dimension of $\mathcal{G}_l$ much less than $N_{l+1} - N_l$. Coding g_l^* using this representation is thus called **overcomplete (or non-critical) residual coding**.

As mentioned in Section 3.2, g_l^* can be represented in a basis $\mathbf{\Psi}_l$ for $\mathcal{G}_l$. Denote by $\mathbf{G}_l^*$ a column vector of coefficients corresponding to the row vector of basis functions $\mathbf{\Psi}_l$, both of length $N_{l+1} - N_l$, such that $g_l^* = \mathbf{\Psi}_l \mathbf{G}_l^*$. The coefficients $\mathbf{G}_l^*$—called *high-pass* coefficients—can be calculated via the normal equation (3.47):

$$\mathbf{G}_l^* = (\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-1} \mathbf{\Psi}_l^\top g_l^*. \tag{4.17}$$

As expressed in (3.45), there are many choices of basis $\mathbf{\Psi}_l$ for $\mathcal{G}_l$. In our work, to satisfy (3.45), we select $\mathbf{Z}_l$ as follow.

We first define a selection operation called $\mathbf{I}_l^b$, which selects a subset of basis in level $l+1$:

$$\mathbf{\Phi}_{l+1}^b = \mathbf{\Phi}_{l+1}\mathbf{I}_l^{bT}. \tag{4.18}$$

Then, $\mathbf{Z}_l^T$ is defined as

$$\mathbf{Z}_l^T = \mathbf{I}_l^{bT} - \mathbf{A}^T(\mathbf{\Phi}_l^T\mathbf{\Phi}_l)^{-1}\mathbf{A}(\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1})\mathbf{I}_l^{bT} \tag{4.19}$$

so that

$$\begin{aligned}
\mathbf{\Phi}_l^T\mathbf{\Psi}_l &= \mathbf{\Phi}_l^T\mathbf{\Phi}_{l+1}\mathbf{Z}_l^T && (4.20)\\
&= (\mathbf{\Phi}_l^T\mathbf{\Phi}_{l+1}^b) - (\mathbf{\Phi}_l^T\mathbf{\Phi}_l)(\mathbf{\Phi}_l^T\mathbf{\Phi}_l)^{-1}(\mathbf{\Phi}_l^T\mathbf{\Phi}_{l+1}^b) && (4.21)\\
&= \mathbf{0}. && (4.22)
\end{aligned}$$

Hence, we can calculate high-pass coefficients as

$$\begin{aligned}
\mathbf{G}_l^* &= (\mathbf{\Psi}_l^T\mathbf{\Psi}_l)^{-1}\mathbf{\Psi}_l^T g_l^*\\
&= (\mathbf{Z}_l\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1}\mathbf{Z}_l^T)^{-1}\mathbf{Z}_l\mathbf{\Phi}_{l+1}^T g_l^* && (4.23)\\
&= (\mathbf{Z}_l\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1}\mathbf{Z}_l^T)^{-1}\mathbf{Z}_l(\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1})\delta\mathbf{F}_{l+1}^*.
\end{aligned}$$

We call $\tilde{\mathbf{G}}_l^* = \mathbf{Z}_l(\mathbf{\Phi}_{l+1}^T\mathbf{\Phi}_{l+1})\delta\mathbf{F}_{l+1}^*$ the *un-normalized high-pass* coefficients. Then,

normalized coefficients, $\mathbf{G}_l^*$, can be calculated as

$$\mathbf{G}_l^* = (\mathbf{Z}_l \mathbf{\Phi}_{l+1}^T \mathbf{\Phi}_{l+1} \mathbf{Z}_l^T)^{-1} \tilde{\mathbf{G}}_l^*. \tag{4.24}$$

4.4 Orthonormalization Operations

All sets of basis functions we described to this point are not mutually orthogonal, meaning that $\mathbf{\Phi}_l^\top \mathbf{\Phi}_l \neq \mathbf{I}$, and $\mathbf{\Psi}_l^\top \mathbf{\Psi}_l \neq \mathbf{I}$. Hence, we next create an orthogonalization operation for each set of basis and apply them to derive $\bar{\mathbf{\Phi}}_l$ and $\bar{\mathbf{\Psi}}_l$ as

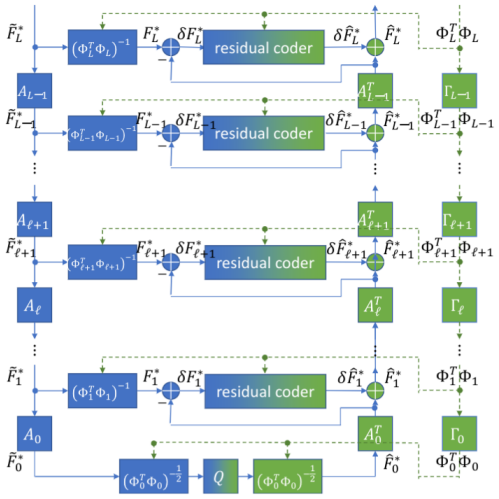
$$\begin{aligned} \bar{\mathbf{\Phi}}_l &= \mathbf{\Phi}_l \mathbf{R}_l = \mathbf{\Phi}_l (\mathbf{\Phi}_l^\top \mathbf{\Phi}_l)^{-\frac{1}{2}} \\ \bar{\mathbf{\Psi}}_l &= \mathbf{\Psi}_l \mathbf{S}_l = \mathbf{\Psi}_l (\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-\frac{1}{2}}. \end{aligned} \tag{4.25}$$

In this way, $\bar{\mathbf{\Phi}}_l^\top \bar{\mathbf{\Phi}}_l = \mathbf{I}$ and $\bar{\mathbf{\Psi}}_l^\top \bar{\mathbf{\Psi}}_l = \mathbf{I}$. The coefficients corresponding to the orthonormal basis functions $\bar{\mathbf{\Phi}}_l$ and $\bar{\mathbf{\Psi}}_l$ can now be calculated as,

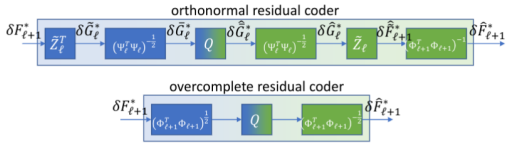
$$\begin{aligned} \bar{\mathbf{F}}_l^* &= \bar{\mathbf{\Phi}}_l^\top f = (\mathbf{\Phi}_l^\top \mathbf{\Phi}_l)^{-\frac{1}{2}} \mathbf{\Phi}_l^\top f = (\mathbf{\Phi}_l^\top \mathbf{\Phi}_l)^{\frac{1}{2}} \mathbf{F}_l^* = (\mathbf{\Phi}_l^\top \mathbf{\Phi}_l)^{-\frac{1}{2}} \tilde{\mathbf{F}}_l^* \\ \bar{\mathbf{G}}_l^* &= \bar{\mathbf{\Psi}}_l^\top g = (\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-\frac{1}{2}} \mathbf{\Psi}_l^\top g = (\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{\frac{1}{2}} \mathbf{G}_l^*. = (\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-\frac{1}{2}} \tilde{\mathbf{G}}_l^*. \end{aligned} \tag{4.26}$$

4.5 Feedforward Network Implementation

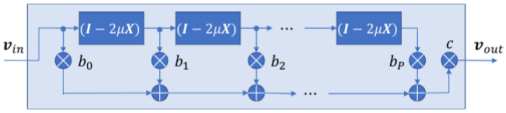
We implement the coding framework in the feedforward network shown in Fig. 4.1a. The encoding network consists of both blue and green components, while the decoding network consists of only green components. The encoding network takes as input, at the top left, the point cloud attributes $\tilde{\mathbf{F}}_L^* = [\mathbf{y}_i]$ represented as a $N \times r$ tensor, where N is the number of points in the point cloud, and r is the number of attribute channels. (We assume $r = 1$ for simplicity in earlier sections, but this is easily



(a) Implemetation of Encoder (blue and green) and decoder (green).



(b) Multilayer feedforward networks implementing the residual coder subnetwork in Fig.4.1a.



(c) Multilayer feedforward network implementing the subnetworks $h(\mathbf{X})$ in Figs.4.1a and 4.1b, where $h(x) = x^{-1}$, $x^{-1/2}$, or $x^{1/2}$ and $\mathbf{X} = (\mathbf{\Phi}_\ell^\top \mathbf{\Phi}_\ell)$, $(\mathbf{\Phi}_\ell^\top \mathbf{\Phi}_{\ell+1}^a)$, or $(\mathbf{\Psi}_\ell^\top \mathbf{\Psi}_\ell)$. Here, $b_0, \ldots, b_P, c$ are coefficients of the Pth order Taylor expansion of $h(x)$ around $1/(2\mu)$.

Figure 4.1: Multilayer feedforward network architecture for volumetric approach

generalized as coding each attribute channel independently.)

The attributes $[\mathbf{y}_i]$ may be considered features located at positions $[\mathbf{x}_i]$ in a sparse voxel grid, or they may be considered features located at vertices $[\mathbf{x}_i]$ in a sparse graph. The network takes $[\mathbf{x}_i]$ as input, at the top right, the matrix of inner products $\mathbf{\Phi}_L^\top \mathbf{\Phi}_L$ represented as a $N \times 27$ tensor, whose ijth entry is the inner product between the basis function $\phi_{L,\mathbf{n}_i}$ located at voxel or vertex $\mathbf{n}_i = \mathbf{x}_i$ and the basis function $\phi_{L,\mathbf{n}_j}$ located at voxel or vertex $\mathbf{n}_j$, the jth neighbor in the 27-neighborhood of $\mathbf{n}_i$. Recall that by construction, this entry is δ_{ij} at level L. In addition, $[\mathbf{x}_i]$ are voxelized, which means that they are the decoded geometric locations of points, and they are already quantized to be at the root corners of the finest cubes (at level L).

Then, from level $\ell = L-1$ to level $\ell = 0$, since we pick $p = 2$, the operation $\mathbf{A}_l$ can be considered a sparse convolution defined in (4.6). This operation can also be

considered a graph convolution layer in any well-known deep learning framework, in which the graph is a bipartite graph that connects nodes in $\mathcal{N}_l$ with $\mathcal{N}_{l+1}$ and edge weights are calculated in (4.7).

Similarly, the network computes the $N_l \times 27$ tensor form of $\mathbf{\Phi}_l^T \mathbf{\Phi}_l$ via a sparse convolution ($\mathbf{\Gamma}_l$) with a space-invariant $27 \times 27 \times 27$ kernel, which can be derived from (4.8). Following this, the network computes the $N_l \times r$ tensor $\mathbf{F}_l^* = (\mathbf{\Phi}_l^\top \mathbf{\Phi}_l)^{-1} \tilde{\mathbf{F}}_l^*$ as in (4.15), by a subnetwork shown in Fig. 4.1c. The subnetwork implements the operator $\mathbf{X}^{-1}\mathbf{v}$ for the matrix $\mathbf{X} = \mathbf{\Phi}_l^\top \mathbf{\Phi}_l$ using a truncated Taylor series approximation of $h(x) = x^{-1}$ around $x_0 = 1/(2\mu)$.

As introduced in Section 4.3, there are two methods to encode the residual function g_l^*: **overcomplete** and **critical**:

- The overcomplete method will first calculate the residual coefficients $\delta\mathbf{F}_{l+1}^* = \mathbf{F}_{l+1}^* - \mathbf{A}_l^T \mathbf{F}_l^*$ and then apply normalization operator $\mathbf{X}^{-\frac{1}{2}}\mathbf{v}$ for the matrix $\mathbf{X} = \mathbf{\Phi}_l^\top \mathbf{\Phi}_l$ using a truncated Taylor series approximation of $h(x) = x^{-\frac{1}{2}}$ around $x_0 = 1/(2\mu)$.

- The critical method will apply operation $\mathbf{Z}_l \delta\mathbf{F}_{l+1}^*$ (defined in (3.46)) to the calculated residual coefficients $\delta\mathbf{F}_{l+1}^*$ and then apply normalization operator $(\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-\frac{1}{2}}\mathbf{v}$. In our implementation, we represent $\mathbf{\Psi}_l^\top \mathbf{\Psi}_l$ as a sequence of operations $\mathbf{Z}_l^\top$, $\mathbf{\Phi}_{l+1}^\top \mathbf{\Phi}_{l+1}$ and $\mathbf{Z}_l$ that stack sequentially. Then, a truncated Taylor series approximation of $h(x) = x^{-\frac{1}{2}}$ is used to approximate $(\mathbf{\Psi}_l^\top \mathbf{\Psi}_l)^{-\frac{1}{2}}$.

Then, given our coefficients $\bar{\mathbf{F}}_0^*, \bar{\mathbf{G}}_1^*, \cdots, \bar{\mathbf{G}}_{L-1}^*$, we next quantize the coefficients

by a stepsize Δ,

$$Q(\mathbf{x}) = \text{round}(\mathbf{x}/\Delta) * \Delta \tag{4.27}$$

$$\hat{\bar{\mathbf{F}}}_0^* = Q(\bar{\mathbf{F}}_0^*) \tag{4.28}$$

$$\hat{\bar{\mathbf{G}}}_l^* = Q(\bar{\mathbf{G}}_l^*) \tag{4.29}$$

which later on, we can use an entropy coder to code the index of the quantization bins — $bin(\mathbf{x}) = \text{round}(\mathbf{x}/\Delta)$.

Then we use the quantized coefficients to synthesize back the input attributes of the point cloud. We rely on two operations to do that, which are showed as the green components in Figure 4.1a:

- The transpose graph convolution layer $\mathbf{A}_l^\top$: this operation uses the same bipartite graph that connects nodes in $\mathcal{N}_l$ with $\mathcal{N}_{l+1}$, and edge weights are calculated as in (4.7). However, the direction is from nodes in $\mathcal{N}_l$ to $\mathcal{N}_{l+1}$, which is opposite to operation $\mathbf{A}_l$ in the analysis.

- The element-wise residual addition, $\mathbf{F}_{l+1}^* = \mathbf{A}_l^T \mathbf{F}_l^* + \delta\mathbf{F}_{l+1}^*$, where $\delta\mathbf{F}_{l+1}^*$ are calculated as shown in Figure 4.1b.

Chapter 5

Experimental Results

In this chapter, the results of conducted experiments to test the proposed coding framework are presented. Section 5.1 describes how we conduct our experiments. Section 5.3 shows the energy compaction of our proposed low-pass subspace. Section 5.4 presents the coding performance of our framework using a given entropy coder.

5.1 Experiments Setups

To demonstrate the effectiveness of our normalization approximation, we set the weights of our feedforward network directly without training, such that it corresponds to volumetric B-splines of order $p = 2$ (as described in Chapter 4). We then compare our pre-defined neutral network, called RAHT($p = 2$), with RAHT [8, 15], which is the current core coding framework of MPEG G-PCC. Evaluations are performed on *Longdress* and *Redandblack* datasets that have 10-bit resolution [59]. Information related to these two datasets is presented in Table 5.1.

We also compare the performance of two different methods for residual coding: **critical** and **overcomplete**. With critical residual coding, we pick the set b for the

Dataset name	Resolution	Number of points
Longdress	$1024 \times 1024 \times 1024$	857966
Redandblack	$1024 \times 1024 \times 1024$	757691

Table 5.1: Dataset information

selection operation $\mathbf{I}_l^b$ as the set of "odd" locations in $\mathcal{N}_{l+1}$, which is $\mathcal{N}_{l+1} \backslash 2 * \mathcal{N}_l$. Since we approximate the square root inverse $\mathbf{X}^{-\frac{1}{2}}$ and the inverse $\mathbf{X}^{-1}$ using a truncated Taylor series expansion of degree D, we also compare our models with different degrees $D \in \{20, 50, 200\}$. This will show how computation demanding our neutral net is.

In Section 5.2, we present the point cloud visually after compression / decompression. This will show the benefits of using higher-order B-spline to represent the volumetric functions. In Section 5.3, we plot the number of required coefficients along with the Peak signal-to-noise ratio (PSNR) to show that our method can express the volumetric functions more efficiently with a given number of coefficients. In the last Section 5.4, we apply Run-Length Golomb-Rice (RLGR) coder to code the coefficients in all level (each channel Y-U-V is coded separately), then we plot the required bit per point with PSNR to show the improvement of our model in term of traditional coding gain.

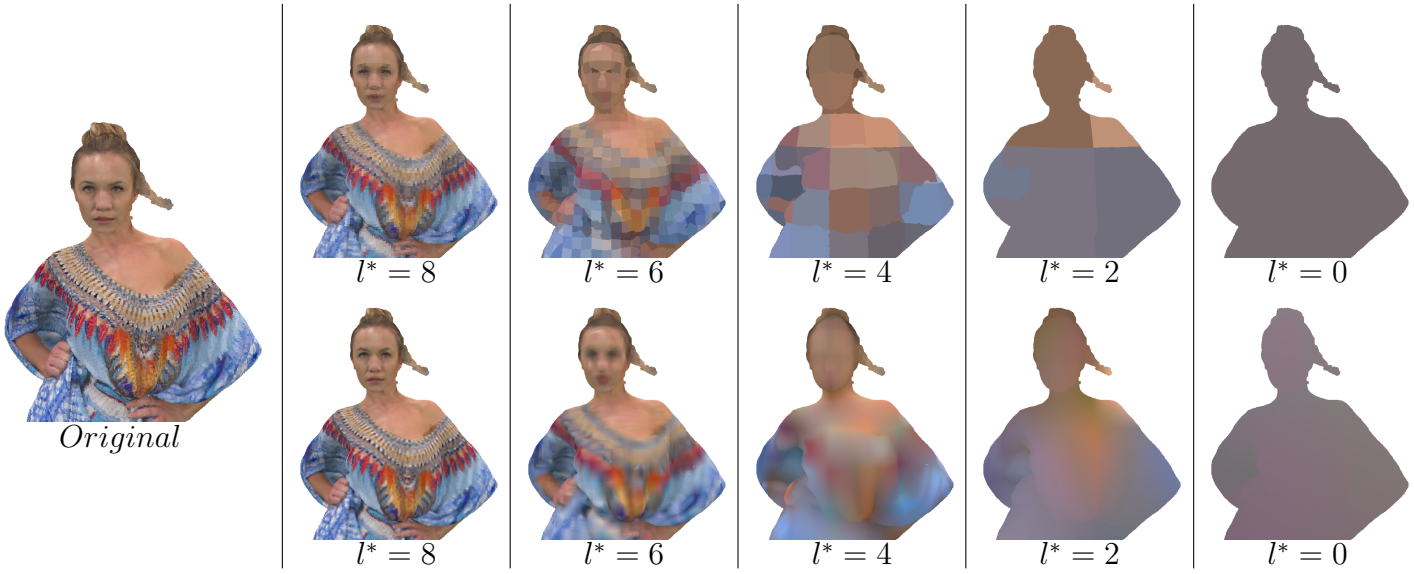


Figure 5.1: Longdress dataset: the first row is RAHT and the second row is RAHT(p=2)

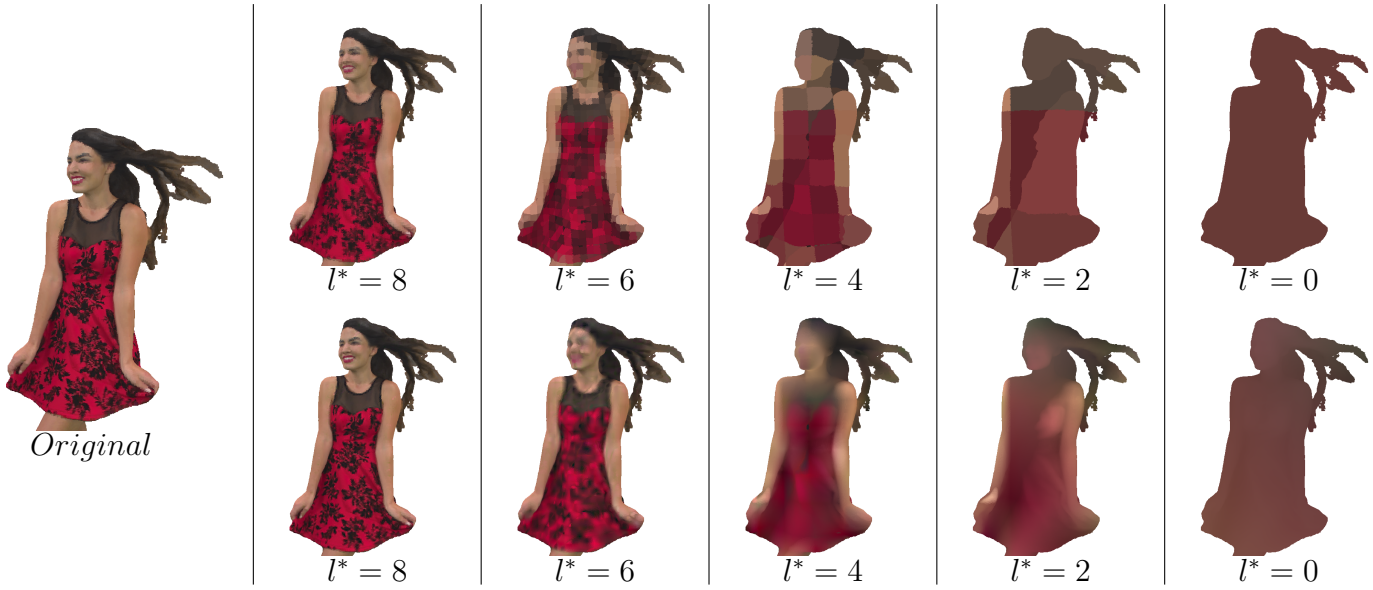


Figure 5.2: Redandblack dataset: the first row is RAHT and the second row is RAHT(p=2)

5.2 Visual Performance Results

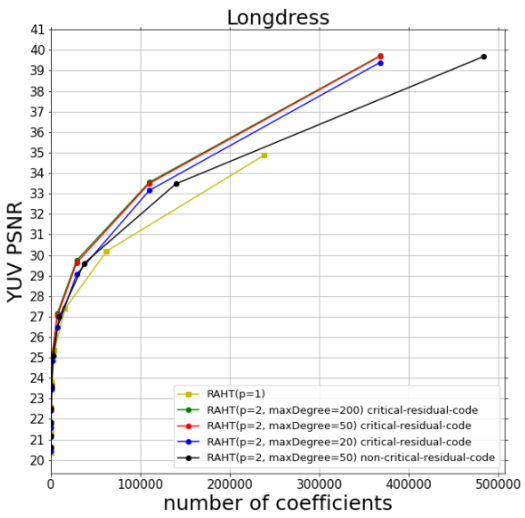
In Fig. 5.2 and Fig. 5.1, we reconstruct the point cloud at different levels of resolution to demonstrate the effectiveness of higher-order B-splines. This can be done by setting all the highpass coefficients $\mathbf{G}_l^* = 0$ for all $l > l^*$.

From the observable visual difference, it is clear that *piecewise trilinear* basis functions can visually fit the point cloud much better. As illustrated in [10], RAHT can be understood as a special case when $p = 1$, corresponding to the subspace $\mathcal{F}_l^{(1)}$ of *piecewise constant*.

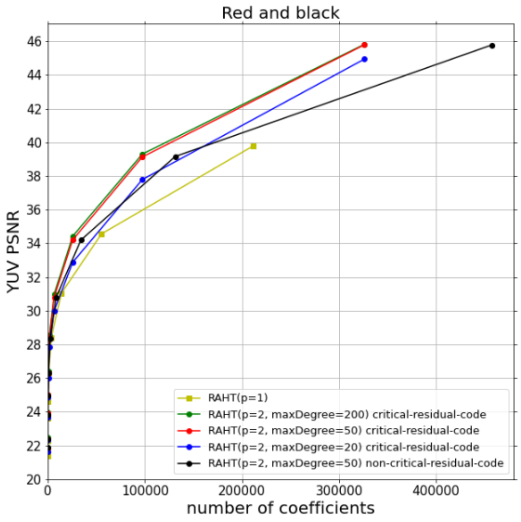
Hence, we can observe that the blocking artifacts appear in all levels of RAHT and vary in size depending on the scale l of the functional subspace. With RAHT(2), blocking artifacts are much less visible, since the higher-order basis functions guarantee a continuous transition of colors between blocks.

5.3 Energy compaction Results

In Fig. 5.3, we show distortion as a function of the number of coefficients corresponding to different resolutions l^* from 0 to 9. The gap between lines is the improvement in energy compaction. We also compare different degrees of the Taylor expansion in our approximation; the plots show that the truncated Taylor expansion can approximate quite well with $D \sim 20$. It is clear that RAHT(2) has better energy compaction, achieving up to 2-3 dB over RAHT(1). We also show that overcomplete residual coding has competitive energy compaction compared to RAHT(1), even though it requires more coefficients to express the residual at each level, especially at high levels of detail.

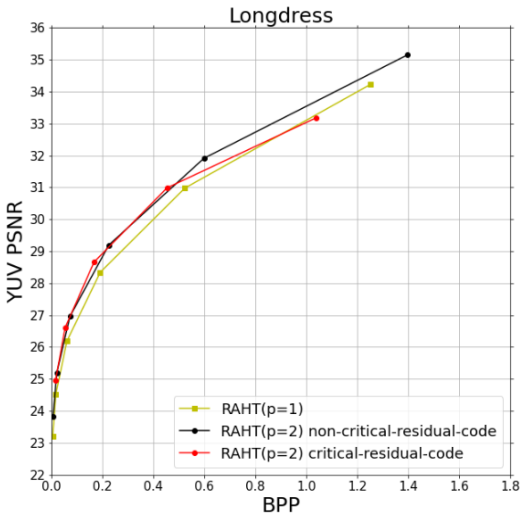


(a) Longdress

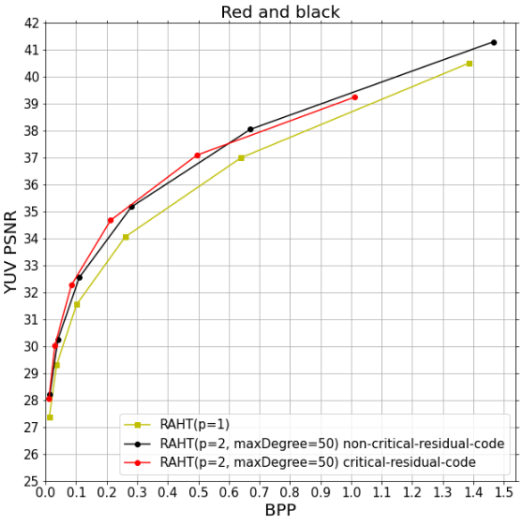


(b) RedAndBlack

Figure 5.3: Energy compaction performance



(a) Longdress



(b) RedAndBlack

Figure 5.4: Coding gain performance

5.4 Coding Gain Results

In Fig. 5.4, we show YUV PSNR as a function of bits per input voxel. Here, we use uniform scalar quantization followed by adaptive Run-Length Golomb-Rice (RLGR)

entropy coding [60] to code the coefficients of each color component (Y, U, V) separately. The plot shows that the improvement of RAHT(2) in distortion for a certain bit rate range can reach 1 dB, or alternatively 20-30% reduction in bit rate, for both overcomplete and orthonormal residual coding.

At higher bitrates, the performance of orthonormal (critical) residual coding falls off. We think the reason may have to do with the orthonormalization matrices, which can be poorly conditioned when the point cloud geometry has certain local configurations. Further investigation is left for future work.

Surprisingly, overcomplete (non-critical) residual coding performs better at these bitrates. However, at even higher bitrates (not shown), its performance also falls off, tracking its energy compaction performance. We think the reason is that most of the overcomplete highpass coefficients at high levels of detail, which tend to be zero at a low rate, now become non-zero; this puts a heavy penalty on coding performance.

Chapter 6

Conclusion

We investigated 3D point cloud attribute coding via a volumetric approach: successive projections to a nested sequence of parametric function spaces towards compact signal representations. These function spaces are spanned by higher-order B-spline basis functions; the attribute function f is then approximated by orthogonal projections into these parametric function spaces. The best approximation of f is parametrized by parameters θ, which are quantized and encoded. $\hat{\theta}$ are reconstructed at the decoder, and the point cloud attributes are evaluated as $f_{\hat{\theta}}(\mathbf{x})$ at locations $\mathbf{x} \in \mathbb{R}^3$.

The important implication of our work is that the process of encoding and decoding can be fully represented and implemented as a feedforward neural network. This is clearly demonstrated in our experiments, where no training process was executed. Unlike generic deep learning networks like conventional convolutional neural nets (CNN), the model-based network architecture developed here is compact, application-specific and interpretable and amenable to future optimizations.

For future work, we plan to further generalize our developed neural network by parameterizing B-spline basis functions and Taylor series expansion and learning these parameters end-to-end using a large dataset of point clouds. Unlike existing

generic neural networks, by initiating our network with parameters derived analytically from models, we know our stochastic optimization starts from a known good operation point, from which the rate-distortion performance can be further improved via stochastic gradient descent. This is currently under investigation.

In addition, since our coding framework is developed based on a mathematical model to manage the irregularity and sparsity of the underlying 3D geometry, we conjecture that our framework can also be adopted in others problems involving non-uniform samples of functions (fields) defined on 3D, $f : \mathbb{R}^3 \mapsto \mathbb{R}$. For example, graph signal denoising is one possible challenging problem, and in some applications, the graph's nodes are assumed to be samples of 2D manifold (or surfaces in 3D). Then, our framework can be applied here to project the noisy signal onto piecewise planar signal for denoising.

Bibliography

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. César, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. J. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, pp. 133–148, 2019.

[2] C. Loop, C. Zhang, and Z. Zhang, "Real-time high-resolution sparse voxelization with application to image-based modeling," in *Proceedings of the 5th High-performance Graphics Conference*, pp. 73–79, 2013.

[3] C. Tulvan, R. Mekuria, Z. Li, and S. Laserre, "Use cases for point cloud compression (pcc)," 2016.

[4] K. Sugimoto, R. A. Cohen, D. Tian, and A. Vetro, "Trends in efficient representation of 3d point clouds," in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 364–369, IEEE, 2017.

[5] E. Alexiou, K. Tung, and T. Ebrahimi, "Towards neural network approaches for point cloud compression," in *Applications of Digital Image Processing XLIII*

(A. G. Tescher and T. Ebrahimi, eds.), vol. 11510, p. 1151008, International Society for Optics and Photonics, SPIE, 2020.

[6] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE Int'l Conf. Image Processing (ICIP)*, 10 2014.

[7] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Trans. Image Processing*, vol. 25, 4 2016.

[8] 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, 2016.

[9] R. L. de Queiroz and P. A. Chou, "Motion-compensated compression of dynamic voxelized point clouds," *IEEE Transactions on Image Processing*, vol. 26, pp. 3886–3895, Aug. 2017.

[10] 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.

[11] 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.

[12] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. Tourapis, and V. Zakharchenko,

"Emerging MPEG standards for point cloud compression," *IEEE J. Emerging Topics in Circuits and Systems*, vol. 9, pp. 133–148, Mar. 2019.

[13] 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, p. e13, 2020.

[14] 3DG, "G-PCC Codec Description v12," Approved WG 11 document N18891, ISO/IEC MPEG JTC1/SC29/WG11, Geneva, CH, 10 2020.

[15] G. P. Sandri, P. A. Chou, M. Krivokuća, and R. L. de Queiroz, "Integer alternative for the region-adaptive hierarchical transform," *IEEE Signal Processing Letters*, vol. 26, no. 9, pp. 1369–1372, 2019.

[16] G. Toderici, S. M. O'Malley, S. J. Hwang, D. Vincent, D. Minnen, S. Baluja, M. Covell, and R. Sukthankar, "Variable rate image compression with recurrent neural networks," in *4th Int. Conf. on Learning Representations (ICLR)*, 2016.

[17] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimization of nonlinear transform codes for perceptual quality," in *2016 Picture Coding Symp. (PCS)*, 2016.

[18] G. Toderici, D. Vincent, N. Johnston, S. J. Hwang, D. Minnen, J. Shor, and M. Covell, "Full resolution image compression with recurrent neural networks," in *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," in *5th Int. Conf. on Learning Representations (ICLR)*, 2017.

[20] J. Ballé, "Efficient nonlinear transforms for lossy image compression," in *2018 Picture Coding Symp. (PCS)*, 2018.

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

[22] D. Minnen, J. Ballé, and G. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems 31*, 2018.

[23] J. Balle, P. A. Chou, D. Minnen, S. Singh, N. Johnston, E. Agustsson, S. J. Hwang, and G. Toderici, "Nonlinear transform coding," *IEEE Journal of Selected Topics in Signal Processing*, pp. 1–1, 2020.

[24] F. Mentzer, G. D. Toderici, M. Tschannen, and E. Agustsson, "High-fidelity generative image compression," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[25] Y. Hu, W. Yang, Z. Ma, and J. Liu, "Learning end-to-end lossy image compression: A benchmark," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[26] W. Yan, Y. Shao, S. Liu, T. H. Li, Z. Li, and G. Li, "Deep autoencoder-based lossy geometry compression for point clouds," *CoRR*, vol. abs/1905.03691, 2019.

[27] 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)*, pp. 4320–4324, IEEE, 2019.

[28] 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)*, pp. 34–39, 2019.

[29] 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)*, pp. 1–5, 2019.

[30] 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 Int. Conf. on Multimedia & Expo Wksps. (ICMEW)*, 2020.

[31] 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 Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[32] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pp. 1–6, 2020.

[33] 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.

[34] 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 preprint arXiv:2111.08988*, 2021.

[35] J. Wang and Z. Ma, "Sparse tensor-based point cloud attribute compression," in *2022 IEEE Int'l Conf. Multimedia Information Processing and Retrieval (MIPR)*, 2022.

[36] T. Bird, J. Ballé, S. Singh, and P. A. Chou, "3d scene compression through entropy penalized neural representation functions," in *Picture Coding Symposium (PCS)*, June 2021.

[37] B. Isik, "Neural 3d scene compression via model compression," *2021 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) WiCV Workshop. arXiv:2105.03120*, 2021.

[38] T. Takikawa, A. Evans, J. Tremblay, T. Müller, M. McGuire, A. Jacobson, and S. Fidler, "Variable bitrate neural fields," in *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings*, SIGGRAPH22 Conference Proceeding, (New York, NY, USA), Association for Computing Machinery, 2022.

[39] 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, 2016.

[40] A. L. Souto, V. F. Figueiredo, P. A. Chou, and R. L. de Queiroz, "Set partitioning in hierarchical trees for point cloud attribute compression," *IEEE Signal Processing Letters*, vol. 28, pp. 1903–1907, 2021.

[41] C. Zhang, D. Florêncio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2066–2070, 2014.

[42] 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)*, pp. 1374–1378, IEEE, 2016.

[43] 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, 2017.

[44] E. Pavez, B. Girault, A. Ortega, and P. A. Chou, "Region adaptive graph fourier transform for 3d point clouds," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 2726–2730, IEEE, 2020.

[45] 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, 2019.

[46] 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, 2016.

[47] 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, 2017.

[48] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, 2021.

[49] M. Quach, G. Valenzise, and F. Dufaux, "Folding-based compression of point cloud attributes," in *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 3309–3313, IEEE, 2020.

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

[51] S. Roman, S. Axler, and F. Gehring, *Advanced linear algebra*, vol. 3. Springer, 2005.

[52] P. R. Halmos, *Measure theory*, vol. 18. Springer, 2013.

[53] R. Schnabel and R. Klein, "Octree-based point-cloud compression.," *PBG@ SIGGRAPH*, vol. 3, 2006.

[54] 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, 2008.

[55] E. Lee, "A simplified b-spline computation routine," 1982.

[56] C. K. Chui, *An introduction to wavelets*, vol. 1. Academic press, 1992.

[57] C. K. Chui, *Wavelets: a tutorial in theory and applications*. Academic Press Professional, Inc., 1993.

[58] G. Thomas and R. Finney, *Calculus and Analytic Geometry.* Addison-Wesley world student series, Addison-Wesley Publishing Company, 1995.

[59] E. d'Eon, B. Harrison, T. Meyers, and P. A. Chou, "8i voxelized full bodies — a voxelized point cloud dataset," input document M74006 & m42914, ISO/IEC JTC1/SC29 WG1 & WG11 (JPEG & MPEG), Ljubljana, Slovenia, Jan. 2017.

[60] H. Malvar, "Adaptive run-length/golomb-rice encoding of quantized generalized gaussian sources with unknown statistics," in *Data Compression Conference (DCC'06)*, pp. 23–32, 2006.