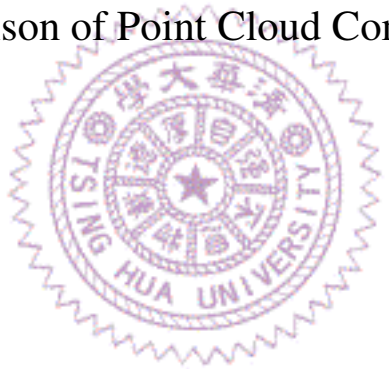
國立清華大學電機資訊學院資訊系統與應用研究所
碩士論文
Institute of Information Systems and Applications
College of Electrical Engineering and Computer Science
National Tsing Hua University
Master Thesis

點雲壓縮演算法之量化分析
Quantitative Comparison of Point Cloud Compression Algorithms



吳丞浩
Cheng-Hao Wu

學號：108065520
Student ID:108065520

指導教授：徐正炘 博士
Advisor: Cheng-Hsin Hsu, Ph.D.

中華民國 110 年 七 月
July, 2021

國立清華大學
資訊系統與應用研究所

碩士論文

點雲壓縮演算法之量化分析

吳丞浩

Abstract

With the growth of Extended Reality (XR) and capturing devices, point cloud representation has become attractive to academics and industry. Point Cloud Compression (PCC) algorithms further promote numerous XR applications that may change our daily life. However, in literature, PCC algorithms are often evaluated with heterogeneous datasets, metrics, and parameters, making the results hard to interpret. In this thesis, we propose an open-source benchmark platform called PCC Arena. Our platform is modularized in three aspects: PCC algorithms, point cloud datasets, and performance metrics. Users can easily extend PCC Arena in each aspect to fulfill the requirements of their experiments. To show the effectiveness of PCC Arena, we integrate seven PCC algorithms into PCC Arena along with six point cloud datasets. We then compare the algorithms on ten carefully selected metrics to evaluate the quality of the output point clouds. We further conduct a user study to quantify the user-perceived quality of rendered images that are produced by different PCC algorithms. Several novel insights are revealed in our comparison: (i) Signal Processing (SP)-based PCC algorithms are stable for different usage scenarios, but the trade-offs between coding efficiency and quality should be carefully addressed, (ii) Neural Network (NN)-based PCC algorithms have the potentials to consume lower bitrate yet perform similarly to SP-based algorithms, (iii) NN-based PCC algorithms may generate artifacts and suffer from long running time, and (iv) NN-based PCC algorithms are worth more in-depth studies as the recently proposed NN-based PCC algorithms improve on the quality and running time. We believe that PCC Arena can play an essential role in allowing engineers and researchers to better interpret and compare the performance of future PCC algorithms.

中文摘要

隨著擴增實境和攝像裝置的成長普及，點雲這一種資料表示型態越來越受到學術界和產業界的青睞。點雲壓縮演算法更進一步地促進了無數的擴增實境應用並改變我們的日常生活。然而，點雲壓縮演算法在文獻中通常都是以不同的資料集、量尺和執行參數進行評估，這讓評估出來的結果很難公平的解讀比較。在這篇論文中，我們提出一個開源的評量平台叫做「PCC Arena」。這個平台在三個方面做了模組化的設計：點雲壓縮演算法、點雲資料集和表現量尺。使用者可以很輕易的在任何一個方面擴展我們的平台以滿足他們實驗需求。為了展示PCC Arena的成效，我們將七個點雲壓縮演算法和六個點雲資料集整合到其中並透過十個特別挑選的表現量尺來比較不同演算法所產出的點雲之品質。我們進一步進行了使用者研究來量化不同點雲壓縮演算法產生的點雲渲染在二維平面上的使用者主觀品質感受。我們從實驗結果中得出幾點新穎的觀察：一、以訊號處理為基底的點雲壓縮演算法在不同的使用情境下表現得比較穩定，但是要小心注意編碼效率與品質之間的取捨。二、以神經網路為基底的點雲壓縮演算法有潛力用更低的位元速率來呈現相似於以訊號處理為基底的點雲壓縮演算法的品質。三、以神經網路為基底的點雲壓縮演算法可能產生缺損並受限於較長的執行時間。四、最近提出的以神經網路為基底的點雲壓縮演算法分別在品質及執行時間上都有進步，值得更深入的研究。我們相信PCC Arena會扮演一個重要的角色，讓工程師和研究人員更好地解釋和比較未來所提出的點雲壓縮演算法的表現。

Contents

List of Figures

List of Tables


TSING HUA UNIVERSITY

Chapter 1

Introduction

With the growing number of Extended Reality (XR)[1] capable devices, we are a step closer to many immersive applications in various industry verticals, such as entertainment, healthcare, educations, emergency management, engineering, and marketing. 3D meshes and point clouds are two popular data formats for *3D models* representing objects and scenes. A mesh comprises a set of polygons, where each polygon contains multiple vertices, edges, and a face. These polygons can be augmented with additional information like texture, reflectance, and orientation. Meshes can be efficiently rendered on hardware-accelerated GPUs. Meshes are widely used to represent 3D models created by computer graphics software, like AutoCAD, 3D Max, and Maya. The creators may choose to attach textures to the meshes created in this manner. Meshes can also represent natural objects and scenes that were captured by sensors. However, such meshes are harder to produce because *none* of the sensors, including RGB cameras, Time-of-Flight (ToF) cameras, and LiDAR scanners, *natively* output 3D meshes. That is, sensor data *must* be processed to generate mesh models, which is computationally expensive and less suitable for real-time applications.

A point cloud, in contrast, is a lighter-weight representation, which only consists of a set of points storing the coordinates and optional attributes, such as colors, normals, and reflectance. Although point clouds can either be directly captured (e.g., from LiDAR scanners) or efficiently generated (e.g., from RGB images, depth images, or both), the resulting point clouds may be too sparse to be visually appealing in XR applications. Therefore, dense point clouds are needed. For example, it is reported that at least half a million points are required to represent an acceptable-quality human avatar [15]. However, streaming *uncompressed* dynamic point clouds dictates more than 4 Gbps bandwidth [10], which exceeds the capacity of most commodity access links. Therefore, an

[1]We use XR as an umbrella term to refer to Virtual Reality (VR), Mixed Reality (MR), Augmented Reality (AR), and other spatial computing technologies.

effective and efficient Point Cloud Compression (PCC) algorithm is the *enabler* of many real-time XR applications.

The existing PCC algorithms can be classified into two categories: (i) Signal-Processing (SP) [19, 49, 53] and (ii) Neural Network (NN) [3, 20, 23, 45, 46, 47, 60, 61, 65] based algorithms. MPEG Geometry-based Point Cloud Compression (G-PCC) [38] and Video-based Point Cloud Compression (V-PCC) [39] are two representative SP-based PCC algorithms. They employ classical signal processing techniques to leverage the data redundancy for compressing point clouds. In contrast, NN-based PCC algorithms adopt neural networks to learn the key features from large point cloud datasets. While increasingly more NN-based PCC algorithms [45, 47, 60, 61] have been proposed, they are often evaluated by solely comparing against prior SP-based PCC algorithms. Furthermore, PCC algorithms are evaluated with different datasets, metrics, and parameters, making their evaluation results hard to interpret. *To the best of our knowledge, quantitative comparison among multiple SP- and NN-based PCC algorithms under a broad spectrum of conditions has not been performed in the literature.*

We believe that the absence of comprehensive comparisons among PCC algorithms is due to the lack of a general and extensible benchmark platform of PCC algorithms.

1.1 Contributions

In this thesis, we propose *PCC Arena*, which includes: (i) a diverse set of point cloud datasets, derived from popular mesh datasets, (ii) a rich set of performance metrics for quantifying perceived quality, bitrate, running time, etc., and (iii) a recommended procedure to fairly compare existing and future PCC algorithms. The PCC Arena project was launched in April 2020 on GitHub [2] and published in a preliminary workshop paper [63]. Since then, we have extensively extended PCC Arena. We report the extensions and completely new evaluation results in the current thesis. This thesis makes three main contributions:

- We design, implement, and release a modularized benchmark platform, PCC Arena, to facilitate performance comparison among PCC algorithms using diverse datasets and metrics.

- We carry out a quantitative comparison study among representative SP- and NN-based PCC algorithms using our proposed PCC Arena. Both objective and subjective quality metrics are employed in this study.

[2] See `https://github.com/xtorker/PCC_Arena` for our project page.

- We provide a detailed discussion on the performance, limitations, and potential of NN-based PCC algorithms, which sheds some light on the future development of PCC algorithms.

In particular, PCC Arena is an open-source project with a modularized design that offers extensibility in three dimensions: PCC algorithms, point cloud datasets, and performance metrics. Engineers and researchers can readily implement their proposed algorithms, collected datasets, and preferred metrics following the design of PCC Arena. After that, they can compare their work against the popular ones in the literature. If they choose to share their implementations with the open-source community, their work will become more visible to the authors of follow-up studies. Hence, we believe PCC Arena would accelerate the development of future PCC algorithms.

1.2 Organizations

The rest of this thesis is organized as follows. We first introduce the background, including 3D representations, point cloud analytics, and performance benchmark in Chapter 2. We survey the literature in Chapter 3. We introduce representative PCC algorithms in Chapter 4. This is followed by the architecture and implementations of PCC Arena in Chapter 5. Chapter 6 presents the point cloud datasets and Chapter 7 defines the performance metrics. We analyze the evaluation results in Chapter 8 and 9. Chapter 10 highlights the pros and cons of NN-based PCC algorithms, and Chapter 11 concludes the thesis.

1.3 Limitations

To demonstrate the practicality of PCC Arena, we use it to benchmark seven representative PCC algorithms: three SP-based [38, 39, 53], and four NN-based [45, 47, 60, 61]. Our evaluation results reveal that:

- The SP-based PCC algorithms offer different trade-off points between coding efficiency and time complexity when bandwidth is scarce. The gap among SP-based PCC algorithms diminishes when the bandwidth is abundant (above 1 bit-per-point, or bpp).

- The NN-based PCC algorithms show great potential at lower bitrates, e.g., saving as high as 50% of bandwidth compared to SP-based ones. They, however, run slower than the SP-based PCC algorithms (at least 10 times slower) and suffer from varying performance with different models or datasets.

- The NN-based PCC algorithms deliver similar point cloud quality to the SP-based ones at lower bitrates most of the time. However, some models seem to be more challenging for NN-based PCC algorithms and are vulnerable to catastrophic quality drops. More precisely, we observe that about 4.4% of the output point clouds from some NN-based PCC algorithms suffer at least two orders of magnitude drops in visual quality.

- Several objective metrics show correlations with subjective scores. However, such correlation is only significant in a relatively small dataset of avatars, showing opportunities for developing better Quality-of-Experience (QoE) models for 3D point clouds.

We emphasize that the seven PCC algorithms are not an exhaustive list of PCC algorithms in the literature. This is, however, not a concern. Given its modularized design, PCC Arena can be used to evaluate other PCC algorithms, including those developed in the future.

Chapter 2

Background

2.1 3D Representations

To represent a 3D model, there are several data types to store and render it, including RGBD images, light-field images, 3D meshes, and point clouds. We introduce the last two representations, 3D meshes and point clouds, in this section.

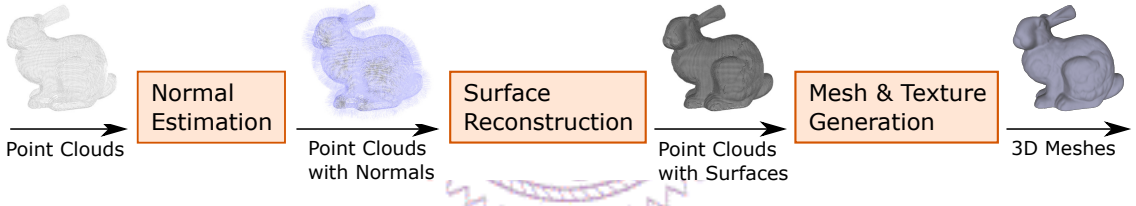


Figure 2.1: Simplified pipeline for generating 3D meshes from point clouds.

3D Meshes. 3D meshes consist of static or dynamic sets of: (i) points, (ii) polygons, and (iii) textures. The polygons are grouped to represent the surfaces in scenes. While different types of polygons can be used in 3D meshes, triangles are used the most often for their simplicity. The pipelines of rendering 3D meshes are quite mature, probably due to the popularity of computer games. In fact, modern GPUs support fast and high-quality rendering of 3D meshes in a scene, from arbitrary virtual camera position and orientation.

The key limitation of the 3D meshes is the *capturing* of nature scenes. 3D meshes are *not* output data types of any capturing sensors. In particular, the raw data from, e.g., RGB cameras, RGBD cameras, and LiDAR sensors, need to be processed by some non-trivial algorithms in order to generate 3D meshes. Fig. 2.1 shows a simplified pipeline of 3D meshes generation from point clouds, which consists of three stages. In *normal estimation*, the normal of each point is estimated by considering near-by points. In *surface reconstruction*, close-by points are clustered to form surfaces. In *mesh and texture generation*, the points, edges, and surfaces form the meshes, while the textures are also attached. We emphasize that each of these stages can be done by a wide spectrum of

algorithms, each with its pros and cons. Generating 3D meshes is no easy task for two reasons. First, the algorithms are quite heavy, and thus cannot be run in real-time. Second, the noise produced by the capturing sensors may be amplified in the pipeline, leading to degraded 3D mesh quality.

MPEG groups have defined quite a few 3D mesh codecs in the past. For example, the MPEG-4 group proposed a 3D mesh codec called FAMC (Frame-based Animated Mesh Compression) [33], which supported both spatial and temporal scalability. At least a 60% average gain in the compression ratio was reported when comparing FAMC with previous 3D mesh codecs.

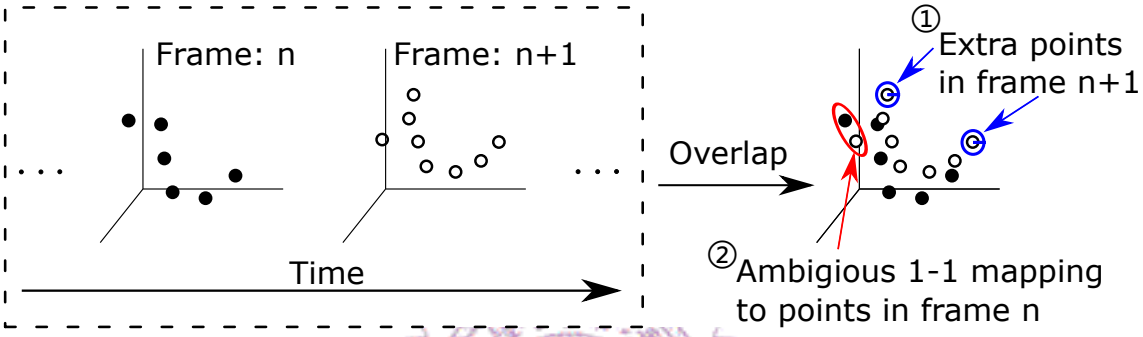


Figure 2.2: Example of varying number of points and non-existing 1-1 mapping over time.

Point Clouds. Point clouds are composed of points, where each point consists of (x, y, z) coordinates as well as other attributes, like RGB, reflection, normals, etc. Point clouds can be directly captured by LiDAR or other sensors. They can also be generated from multiple 2D images taken at different positions with diverse orientations. Compared to 3D meshes, point clouds have the following unique properties.

- **No connectivity information.** Point clouds do not contain edge connections among points. Therefore, to achieve a comparable rendered quality to 3D meshes, many more points are needed in corresponding point clouds. Otherwise, *holes* and *cracks* may become visible. Point clouds, therefore, may incur high storage, transmission, and processing complexities.

- **Unordered points.** The points of a static point cloud have no specific order among them. Unfortunately, some graphics algorithms may dictate the proper order of points before performing the actual tasks. These algorithms, therefore, must estimate the point order themselves. For dynamic point clouds, this problem is even worse, as point clouds contain no 1-1 mapping among points at different time instances. That is, the *future* coordinates of a point at the *current* moment are generally unknown. In fact, the number of points of a dynamic point cloud could be

varying over time. Fig. 2.2 illustrates these two challenges when processing dynamic point clouds.

- **Ease of manipulation.** Because of the above two properties, 3D point clouds are easy to manipulate. For example, stitching multiple point clouds is less complicated than doing the same on multiple 3D meshes.

Point clouds, compared to 3D meshes, have a much lower rendering complexity. Hence, they may be more suitable for interactive applications. The main limitation of point clouds is the tremendous data size for dense enough point clouds to guarantee visually-appealing rendered scenes.

2.2 Novel 3D Applications

With the growing development of 3D representations, different novel 3D applications pop out and attract the attentions. There are many potential applications benefiting from the developing of the 3D representations, including cultural heritage conservation, teleconference, film making support, etc. Schwarz et al. [48] demonstrated the ability to decode and playback the AR content in real-time with MPEG V-PCC codec. Although it is a preliminary version of playing the AR contents on mobile devices, the authors properly showed the potential of V-PCC and the 3D content consuming on mobile devices. Lee et al. [29] proposed a real-time streaming system called GROOT. GROOT encodes the point clouds using conventional signal-processing tools and makes the decision based on the viewport information, including positions and orientations, sent by the client mobile devices. The mobile devices at the end user side will decode and render the point clouds. The main difference of GROOT and the other related work in the literature is that the authors avoided the 3D to 2D projection procedure during the encoding and decoding processes. This makes GROOT more suitable for fully immersive and interactive applications on large 3D content.

2.3 Neural Networks

Neural network has been adopted to several point cloud related applications in the literature for years. In the earlier works, researchers have proposed neural network approaches for voxelized point clouds. In other words, the whole pipeline required a pre-processing procedure to convert the raw point clouds into voxelized point clouds, which had uniform distance among each pair of neighboring points. This procedure significantly reduced the search space of the applied neural network model.

In 2017, Qi et al. [43] proposed PointNet, a neural network model that can directly take point clouds as input for classification and segmentation. This work is a milestone for the NN-based point cloud applications. Before meeting PointNet, research communities relied on heavy computational operation, 3D CNN. PointNet gave researchers a completed new direction to design a neural network for point cloud related applications with lower computational overhead and better performance. Tons of papers based on PointNet have been published in the literature to solve the classification and segmentation problem for point clouds [6, 28, 30, 31, 42, 50, 54, 68, 69, 71].

Besides classification and segmentation, there are other popular analytics of point clouds in the literature. Gilani et al. [18] proposed a deep neural network model for 3D facial landmark identification. After that, several neural network models have been proposed to solve the 3D face recognition problem [17, 25, 52]. For object detection on LiDAR point clouds, Cheng et al. [12] proposed a projection-based object detection neural network model. The authors projected the LiDAR point clouds into two planes, front view and bird eye biew. They combined the features extracted from two projected view and RGB channel with multi-layer deep fusion convolutional neural network. On the other hand, instead of projecting LiDAR point clouds into 2D planes, Yan et al. [66] attempted with 3D sparse convolutions to extract the features.

2.4 Performance Benchmarks

To get a better view of the performance comparison among different algorithms or codecs, a comprehensive benchmark will always be one of the solutions. A benchmark usually includes or defines the following components: (i) datasets, (ii) sample algorithms or codecs, and (iii) performance metrics. The purpose of designing a benchmark is to evaluate different algorithms or codecs with identical pipeline under exactly the same environment, so that the benchmarking results are meaningful and valuable.

Alves et al. [5] proposed a complete benchmark to evaluate the performance of the light field image codecs. The authors broke down the whole procedure of consuming a light field image into three stages and evaluated the coding results in two different aspects. Bernardo et al. [8] designed a benchmark for assessing the objective quality of different image codecs, which encode and decode on the object plane for digital holography. Doumanoglou et al. [16] adopted a comprehensive benchmarking on 3D meshes codecs for immersive interactive real-time streaming like teleconferencing. The authors parsed the whole pipeline and marked out several metrics that helped them to analyze the performance at each point during the pipeline.

Chapter 3

Related Work

Performance comparison of various PCC algorithms has been attempted in both standardization and research communities. In the former community, MPEG has published the Common Test Condition (CTC) document [36, 37] to define the test procedures for several usage scenarios of PCC algorithms. The test procedures specify: (i) predefined coding parameters for G-PCC [38] and V-PCC [39], (ii) object, avatar, and scene datasets, and (iii) several objective quality metrics. Although the CTC document focuses on MPEG PCC algorithms, it can serve as the starting point of benchmarking other PCC algorithms.

Some papers [4, 41, 58] evaluated the performance of MPEG G-PCC and V-PCC codecs under diverse settings. In particular, van der Hooft et al. [58] employed objective and subjective metrics to quantify the user-perceived quality of 6 Degree-of-Freedom (6DoF) point cloud streaming. Through a user study, they derived the implication of each network metric on the user-perceived quality of rendered 2D views. Perry et al. [41] also carried out a user study to analyze the performance of MPEG G-PCC and V-PCC codecs. In addition to subjective metrics, they adopted several objective ones. Using the user study results, they identified the objective metric having the highest correlation with the subjective scores. Similarly, Alexiou et al. [4] developed a Web-based point cloud renderer used to collect subjective quality scores. They conducted a user study to understand the impacts of diverse coding parameters on user-perceived quality. Unlike the above studies[4, 41, 58], the current thesis considers a broad spectrum of PCC algorithms spanning the SP- and NN-based ones. Moreover, we strive to include as many quality metrics as possible.

Torlig et al. [56] developed a renderer for point clouds and conducted a user study using objects and avatars. They then investigated the correlation between every pair of objective and subjective metrics. Similar to our work, they also included additional: (i) objective metrics and (ii) PCC algorithms beyond the ones in the CTC document [36, 37]. They, however, did not design a general and extensible benchmark platform, like our PCC

Arena, nor make the platform open-source.

Chapter 4

Point Cloud Compression Algorithms

We introduce representative PCC algorithms in this section.

4.1 General Architecture

Our discussion focuses on the PCC encoder structure, as the decoder structure can be derived from the encoder. Fig. 4.1 shows that the encoder comprises three components: *pre-process*, *redundancy elimination*, and *entropy code*. Each component contains multiple potential coding tools, which can be classified into SP-only, NN-only, and general ones. The pre-process component strives to turn point clouds into representations that are more suitable for redundancy elimination, such as the scale, normalization, and partitioning tools. Among them, the scale and normalization tools ensure the comparable point cloud scale within the same or among different datasets. The partition tool divides each input point cloud into smaller blocks, then sent to the redundancy elimination component. The purpose of partition is to reduce the searching space for the NN-based PCC algorithms. The pre-processed point clouds comprise coordinates and optional attributes. Coordinates are the geometry positions in the Cartesian coordinate system. Attributes carry additional information, such as colors, normals, and reflectance.

The redundancy elimination component is the crux of PCC algorithms, which applies different coding tools to the coordinates and attributes. These tools can further be classified into *transform*, *quantization*, and *integrated* tools. Among the transform tools for coordinates, the projection tool transforms each 3D point cloud to multiple 2D images, which can be readily compressed using existing 2D image codecs. Specialized data structures such as K-D trees and octrees can also be seen as transform tools. As a quantization tool, voxelization turns point clouds with diverse point densities into 3D grids, called voxels, trading lower output quality for less information to compress. More recently, NN architectures have become popular integrated tools that directly eliminate redundancy.

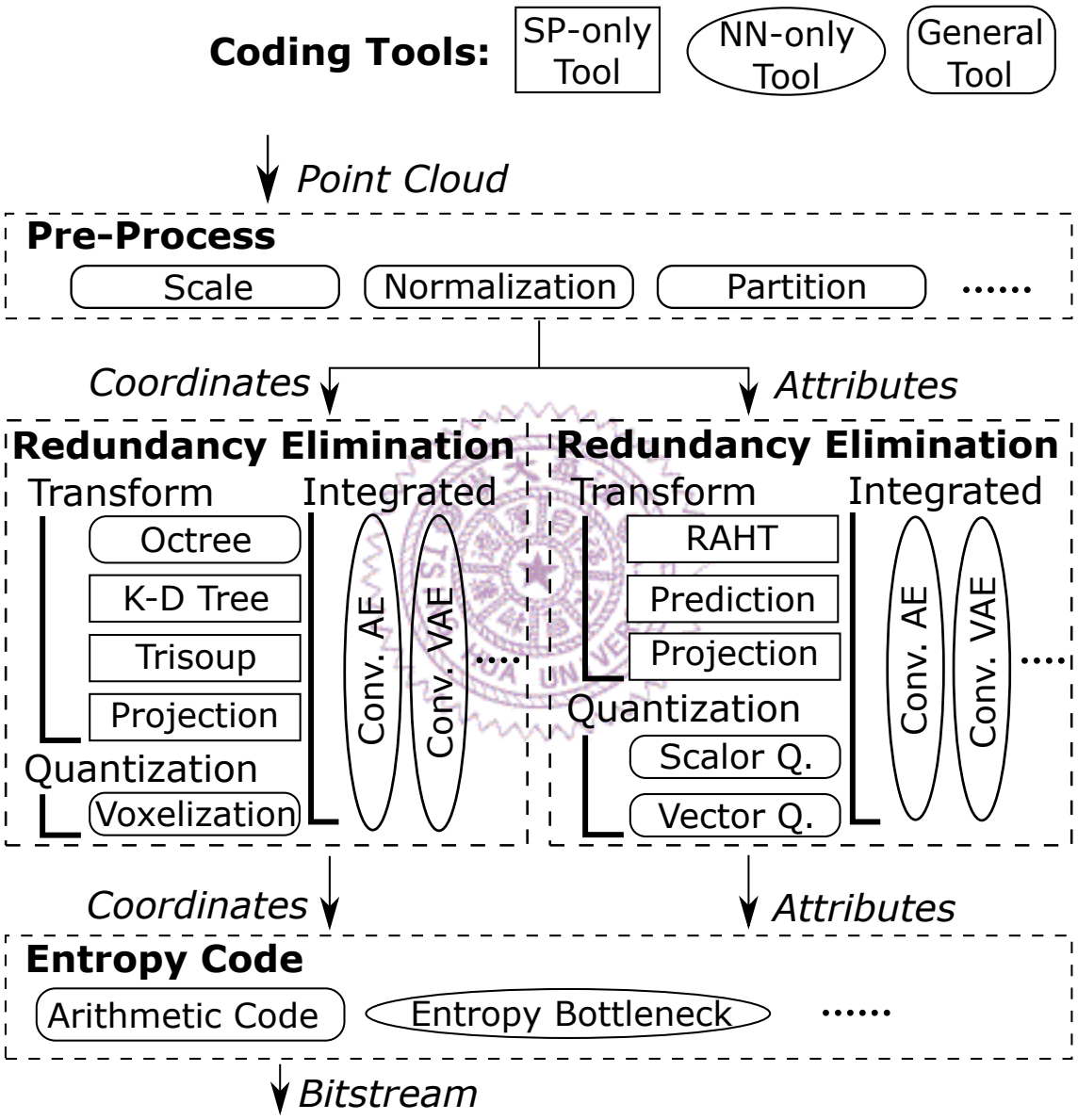


Figure 4.1: The general encoder architecture of PCC algorithms.

For instance, Convolutional AutoEncoder (Conv. AE) [34] and Convolutional Variational AutoEncoder (Conv. VAE) [27] have been applied to deal with the point cloud coordinates. Conv. AE uses multiple convolutional layers to extract the features automatically with unsupervised learning. Compared to AE, VAE searches for the mapping between the input dataset and a Gaussian distribution rather than the output dataset. Therefore, VAE is expected to better generalize to unseen datasets.

Some transform tools are more suitable for attributes like colors. For example, Region-Adaptive Hierarchical Transform (RAHT) [14] and prediction [38] are widely adopted for redundancy elimination of attributes. The RAHT and prediction tools also utilize specialized data structures, like the aforementioned K-D trees and octrees, to predict, e.g., the color of a given point (or block) from the neighboring ones. The transformed data are then quantized using scalar or vector quantization tools. NN architectures can also handle the attributes as integrated tools. However, to our best knowledge, only Conv. AE [34] has been adopted for removing the redundancy of attributes [3, 46] at the time of writing.

The entropy code component further removes the redundancy in the outputs of the redundancy elimination component. Arithmetic code is a popular lossless entropy coding tool, which can be used in both SP- and NN-based PCC algorithms. For NN-based PCC algorithms, the entropy bottleneck layer [7] is widely used to compress the output tensors from earlier NN layers into a bitstream. The entropy bottleneck layer estimates the entropy of the input tensor from the previous layer. It then uses the estimated entropy to transform the input tensor into a more compact string. The length of the transformed string can be readily controlled thanks to the design of the entropy bottleneck.

In the rest of this section, we introduce seven PCC algorithms, starting with the SP-based algorithms, followed by the NN-based ones. The selected PCC algorithms are summarized in Table 4.1.

Table 4.1: Key Coding Tools Adopted by Representative PCC Algorithms

Cate.	Algorithm	Coord.	Color	Partition	Voxelization	Conv. (V)AE	Entropy Bottleneck	Entropy Code
SP	**Draco [53]**	✓	✓		✓			✓
	G-PCC [38]	✓	✓		✓			✓
	V-PCC [39]	✓	✓					✓
NN	**GeoCNN [45]**	✓			✓	✓	✓	✓
	GeoCNNv2 [47]	✓		✓	✓	✓	✓	✓
	PCGCv1 [61]	✓		✓	✓	✓	✓	✓
	PCGCv2 [60]	✓				✓	✓	✓

4.2 SP-Based PCC Algorithms

Draco. Draco [53] is an open-source project from Google. It aims to improve the performance of 3D graphics applications through compression. Draco is initially designed for web page renderers in a browser, but it is later extended for native applications. Draco supports both lossy and lossless compression modes for coordinates and attributes. Its quantizer allows users to specify different bit depths for coordinates and individual attributes. After quantization, Draco encodes the coordinates with K-D trees and applies predictive coding tools to the attributes. This is followed by several entropy coding tools using both static and dynamic codebooks.

G-PCC and V-PCC. MPEG classifies the usage scenarios of PCC algorithms into three categories [49]: *static point clouds*, *dynamically acquired point clouds*, and *point cloud videos*. They propose two codecs accordingly: (i) G-PCC [38], which takes advantage of geometric data structures such as octrees for compressing static and dynamic point clouds, and (ii) V-PCC [39], which leverages mature 2D video compression tools for point cloud videos. G-PCC supports both lossy and lossless compression modes for coordinates and attributes. It contains three modules: voxelization, octree transformation, and entropy coding. The voxelization module quantizes the coordinates from floating-point numbers to integers with a scaling parameter, and it also optionally merges the duplicate points. The octree of the resulting voxels is then built and finally encoded using several entropy coding tools, such as the arithmetic code. V-PCC stores each point cloud frame into three 2D video frames: *occupancy map*, *geometry image*, and *attribute image*. The encoder consists of two steps: 3D projection and 2D video compression. The former step projects each point cloud on multiple 2D planes in different directions, which can be considered as a form of partitioning. The projected results are called *patches*, which are packed into a 2D canvas. A sequence of 2D canvases is then aggregated into a 2D video. V-PCC sends the 2D video to a conventional 2D video codec. H.265 is probably a good choice because it can prevent motion vector search across the boundaries of patches. More details on the V-PCC and G-PCC algorithms can be found in their standard documents [38, 39].

4.3 NN-Based PCC Algorithms

GeoCNNv1. GeoCNNv1 [45] is a pioneering NN-based algorithm taking lossy voxelization as a pre-processing tool for manageable complexity and better performance. GeoCNNv1 consists of two 3D convolutional layers and an entropy bottleneck layer. Its loss function is a weighted sum of the focal loss [32] and the extracted feature size. The focal loss aims to get a biased model toward more complicated point clouds for better overall

performance across the dataset. The extracted feature size is in the unit of bits per occupied voxel (bpov). GeoCNNv1 only supports lossy compression on coordinates. For rate control, users are encouraged to train NN models with different loss function parameters λ, which can be used to trade off the extracted feature size (rate) and the focal loss (quality).

GeoCNNv2. GeoCNNv2 [47] improves GeoCNNv1 in several aspects: a deeper neural network, optimized focal loss parameters, and a fine-tuned decoding process. The key enhancement is to adopt VAE [26] instead of AE [22]. It is reported that GeoCNNv2 can handle larger point clouds and achieve better rate-distortion performance with a shorter training time. GeoCNNv2 employs the same rate control mechanism as GeoCNNv1. While GeoCNNv2 only deals with geometry data, the same authors also proposed a Folding-Based Coding (FBC) algorithm [46] to compress attributes, like colors. To our best knowledge, FBC is the very first attempt to compress point cloud attributes using NNs. We, however, have not been able to integrate FBC with GeoCNNv2. Thus, we do not consider FBC in the rest of this thesis.

PCGCv1. Similar to GeoCNNv2, VAE is adopted in PCGCv1 [61], which incorporates multiple 3D convolutional blocks and Voxception-ResNet (VRN) [9] for better coding efficiency. VRN combines ResNet [21] and Inception [51] by stacking several ResNet blocks following the Inception-style. ResNet [21] aims to solve the vanishing gradient problem, while Inception [51] helps feature extraction in different scopes by increasing the width of the NNs. PCGCv1 employs several pre-processing tools, including scaling and partitioning. Particularly, PCGCv1 partitions each point cloud into blocks and compresses the blocks individually at the encoder. The encoder utilizes the entropy bottleneck before performing entropy coding. At the decoder, the decoded blocks are reassembled into the final point cloud. PCGCv1 only compresses coordinates. The rate control approach of PCGCv1 is similar to that of GeoCNNv1: models with different λ values are trained.

PCGCv2. PCGCv2 [60] is an extension of PCGCv1 with two significant enhancements. First, the authors create a multiscale NN based on PCGCv1, which can extract and combine local features in a hierarchical way to get higher-quality output point clouds. Second, sparse convolution is adopted to cope with heterogeneous spareness levels of different point clouds and reduce inference time and memory usage. According to their evaluation results [60], PCGCv2 outperforms PCGCv1 in both quality and running time. PCGCv2 only compresses coordinates and employs the same rate control mechanism as PCGCv1.

Chapter 5

PCC Arena: Architecture and Implementations

After introducing the representative PCC algorithms we implemented into PCC Arena, we now give an overview and implementation details of PCC Arena in this section.

5.1 Architecture

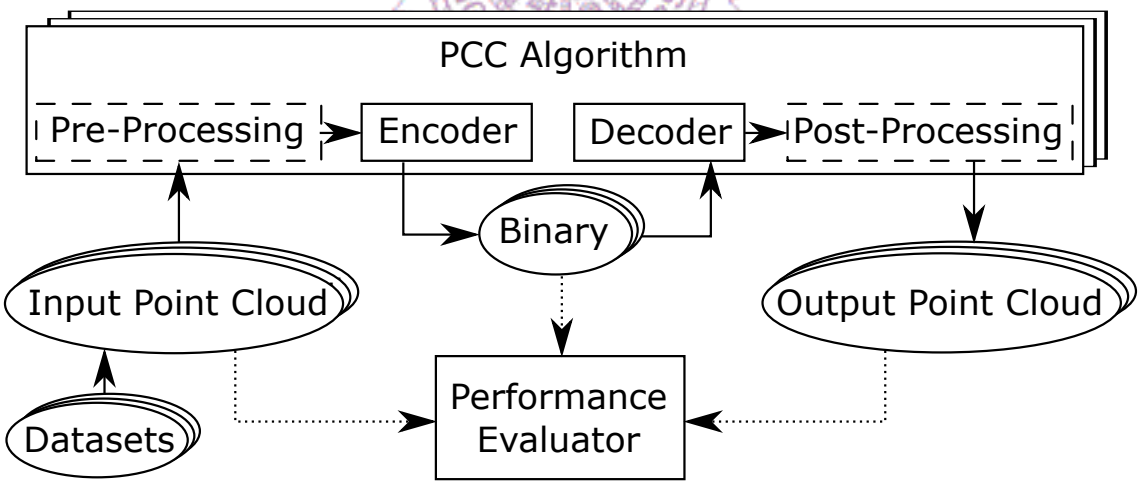


Figure 5.1: High-level architecture of the PCC Arena.

Fig. 5.1 reveals the high-level architecture of our proposed benchmark platform: PCC Arena. The *encoder* takes an *input* point cloud and generates a compressed binary representation, which may be saved as a file or streamed over a network. The *decoder* reconstructs an *output* point cloud from the compressed binary representation. The input point clouds are selected from one of the PCC Arena datasets. Individual PCC algorithms may require some optional *pre-* and *post-processing* steps. Examples of pre-processing

steps include point cloud scaling and partitioning, while the post-processing steps are essentially their inverse functions. The PCC algorithms allow users to make a trade-off between the bitrate and quality through different *coding parameters*. As mentioned above, each PCC algorithm has its own rate control method. For each: (i) input point cloud, (ii) PCC algorithm, and (iii) set of coding parameters, we get an output point cloud, which is analyzed by a *performance evaluator* along with the input point cloud and the compressed binary representation, in order to compute the results of various performance metrics. We also develop a suite of scripts to automate the quantitative comparison across various PCC algorithms under different bitrates. Moreover, the output point clouds are stored and optionally rendered into 2D images for subjective tests.

5.2 Implementations

PCC Arena is implemented mainly in Python 3 using several common Python packages for file I/O, logging, etc. We use Anaconda to provide portable Python environments and libraries in versions dictated by individual PCC algorithms. We strive to follow the settings reported by the authors of individual PCC algorithms. Scripts for getting the environments ready are provided to help engineers and researchers pick up the speed.

We define base classes for PCC algorithms and performance metrics as `AlgorithmBase` and `MetricBase`, respectively. Both default and virtual functions are defined in the base classes, and we have implemented all the subclasses for the considered PCC algorithms and performance metrics in PCC Arena. In addition to the Python classes, configurations related to PCC algorithms (referred to as `alg_cfg`) and datasets (referred to as `ds_cfg`) are given as YAML files. These YAML files can be easily augmented or created by engineers and researchers.

We provide a top-level script as the entry point for running all the pre-defined experiments, summarized in Procedure 1. The script loops over all PCC algorithms, coding parameters, and point cloud datasets at lines 1, 4, and 5, respectively. The script first creates a PCC algorithm instance in line 2 for each PCC algorithm to facilitate encoding point clouds and decoding coded bitstreams. Each algorithm comes with a sample config file (`alg_cfg`), which defines how this PCC algorithm controls the encoding bitrate. To speed up the benchmarking process, we configure parallel executions using, e.g., thread pools or GPU queues in line 7. A dataset-dependent config file (`ds_cfg`) specifies the path to the point clouds and settings, e.g., the point cloud resolutions and whether the point clouds contain attributes, like colors. For each input point cloud, it is encoded, decoded, and evaluated between lines 8–13.

Procedure 1 Top-level Evaluation Script

```
for alg in all considered algorithms do
    Initialize an instance of class alg
    Load the corresponding alg_cfg
    for rate defined in alg_cfg do
        for ds in all considered datasets do
            Load ds_cfg
            Apply parallel-execution cmd. in alg_cfg
            for pc in ds do
                Execute en/decoding commands built from alg_cfg, rate, and ds_cfg on pc
                Initialize an evaluator eval
                for met in all considered metrics do
                    Calculate met with eval
                    Save met results in log files
```

5.3 Extensions

PCC Arena is modularized and extensible in three dimensions: PCC algorithms, point cloud datasets, and performance metrics. To add a new PCC algorithm into PCC Arena, programmers have to prepare: (i) a new PCC algorithm class and (ii) a corresponding configuration file. For the PCC algorithm class, two virtual functions need to be implemented: `make_encode_cmd()` and `make_decode_cmd()`. These two functions put the parameters, such as input file path, into encode and decode commands. The configuration file stores the algorithm-specific parameters, including the coding parameters for rate control. Users can change those parameters based on their experiment requirements. To add a new dataset into PCC Arena, programmers must add a new dictionary into the dataset configuration file. This dictionary records the dataset-specific parameters, such as point cloud filenames and dataset metadata. To add a new performance metric, programmers need to define a subclass of `MetricBase`. For this new metric, a virtual function `evaluate` needs to be implemented, which returns the evaluation results as a human-readable string.

In our experience, most PCC algorithms can be readily integrated into PCC Arena via straightforward *wrapper* classes in Python. A few PCC algorithms require modification on the source code before being interfaced with PCC Arena. Typical modifications include: (i) separating encoding/decoding commands from the testing scripts and (ii) extracting the hard-coded values into parameters. Based on our experience, it took an engineer up to 1 hour to finish such modifications for a new PCC algorithm. Incorporating new datasets and performance metrics takes much less time compared to PCC algorithms.

Chapter 6

Point Cloud Datasets

We now present the default point cloud datasets of PCC Arena in this section.

6.1 Dataset Selection

Table 6.1: Point Cloud Datasets

	CAPOD [40]	**MN40** [64]	**SNC** [11]	**8i** [15]	**SNCC** [11]	**8iC** [15]
Dataset Source	3D Mesh	3D Mesh	3D Mesh	3D Point Cloud	3D Mesh	3D Point Cloud
Data Type	Coordinate	Coordinate	Coordinate	Coordinate	Coord. with Color	Coord. with Color
Content Type	Object + Avatar	Object	Object	Avatar	Object	Avatar
Classes	15	40	55	-	55	-
Models	180	12,308	51,094	1,200	51,094	1,200
Train/Val/Test	-	$0.8/0/0.2$	$0.7/0.1/0.2$	-	$0.7/0.1/0.2$	-
Format	obj	off	obj + mtl	ply	obj + mtl	ply
Adoption	-	[43],[44],[67]	[43],[67],[65]	[47],[61],[60]	-	-

Public 3D model datasets can be roughly classified along two dimensions: (i) data format: 3D meshes versus 3D point clouds and (ii) content type: objects, avatars, or scenes. We consider both meshes and point clouds as dataset sources, but we only focus on objects and avatars in this thesis[1]. We select six datasets, as summarized in Table 6.1. Fig. 6.1 gives sample point clouds from them. We introduce the selected datasets below.

CAPOD. The CAnonically Posed 3D Objects Dataset (CAPOD) contains canonically posed 3D models [40]. That is, these models were pre-processed with several tools, including normalization, rotation alignment, and semantic alignment. CAPOD contains 180 models uniformly distributed across 15 classes.

MN40. ModelNet is a large 3D Computer Aided Design (CAD) dataset [64]. ModelNet contains 151,128 models across 660 classes. We adopt one of its subsets: *ModelNet40*

[1]We do plan to maintain and grow PCC Arena. For example, collecting datasets for real 3D scenes is on our roadmap. We hope to attract more engineers and researchers to join us.

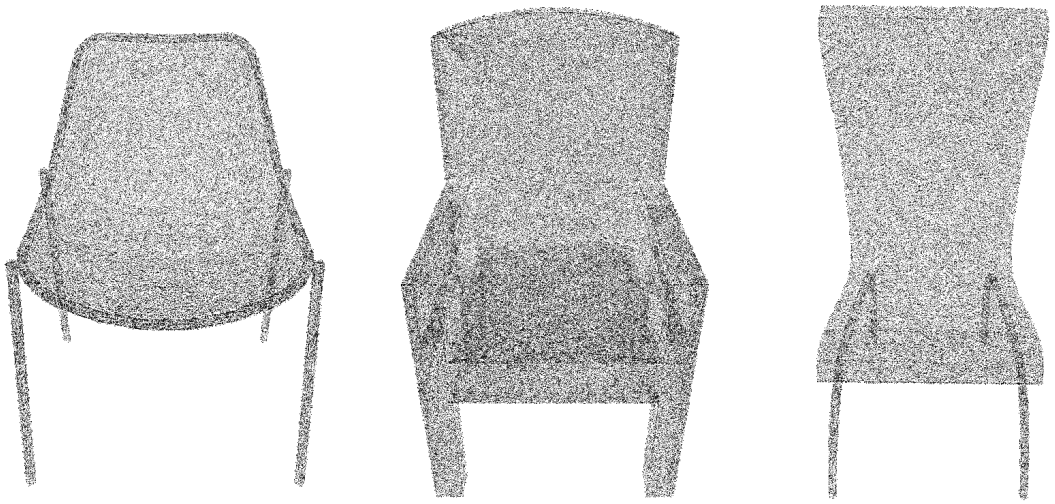


Figure 6.1: Sample point clouds generated from MN40 (left), SNC (middle), and CAPOD (right). Each point cloud consists of 500k points.

(MN40), which contains 40 classes[2].

SNC. ShapeNet is a large 3D model repository with more than 3,000,000 models, where some of the models are labeled and classified into 3,135 classes [11]. Models in ShapeNet come with several annotations, such as rigid hierarchical alignments, parts, and key points. We adopt a subset of ShapeNet, called *ShapeNetCore (SNC)*. SNC contains 51,094 models across 55 classes, such as table, airplane, and guitar.

SNCC. We also consider the colored version of SNC, which is referred to as *ShapeNetCoreColor (SNCC)*. The numbers of classes and models are identical to those of SNC.

8iC. The MPEG CTC document [37] provides multiple point clouds contributed by different research teams. 8iC dataset consists of 4 colored avatar point cloud sequences built from 14 RGBD video clips.

8i. We also consider the coordinate-only version of 8iC, which is referred to as 8i.

6.2 Dataset Generation

Most selected datasets (except 8iC/8i) are 3D mesh datasets. We employ CloudCompare [13] to convert these 3D meshes into 3D point clouds, where each point is composed of coordinates and attributes. Coordinates are represented as x, y, and z. Attributes include colors and normals, represented as red, $green$, $blue$ and nx, ny, nz, respec-

[2]We skip three point clouds in MN40, because they are 2D instead of 3D models.

tively. More precisely, for each 3D mesh object, we generate 500k random points using `SAMPLE_MESH()` in CloudCompare, which randomly samples points using interpolations on the faces of each triangle. We use 500k because it was reported to be the minimum number of points for acceptable rendered quality [10]. Engineers and researchers, however, may modify our scripts to vary the number of points. If the mesh contains colors, we sample colors using interpolation whenever necessary. We derive normals of individual points using `OCTREE_NORMALS()` in CloudCompare using an automatically chosen radius to establish a quadric plane from the neighbors found on the octree. The normals are needed when quantifying some performance metrics. Last, we normalize all the point clouds to [0,1024] along the three axes to fairly compare the PCC algorithms across different datasets. Note that PCC Arena does not ship the point cloud datasets directly; instead, Python scripts for downloading and converting datasets are included.

Our selected datasets are diverse: 8iC/8i was directly captured from the real world, while others were sampled from mesh datasets. These mesh datasets contain objects with flat surfaces and basic geometry structures, which may be easier to compress. We acknowledge that our selected datasets were created by third parties who have applied normalization, alignment, and rotations, which make them different from raw point clouds collected from LiDAR or RGBD cameras. These, however, are not serious concerns, as new datasets can be readily integrated with PCC Arena, as explained in Sec. 5.3.

Chapter 7

Performance Metrics

In this section, we define the default performance metrics included in PCC Arena. These metrics can be classified into: (i) *non-visual*, (ii) *3D visual*, and (iii) *2D visual* metrics.

7.1 Non-visual Metrics

We first define the non-visual metrics, which are crucial to understanding the efficiency of the PCC algorithms. Let $\mathbf{P}_\text{r}$ and $\mathbf{P}_\text{t}$ be the reference and target point clouds, which are two sets of input and output points. We use p to denote a point in a point cloud in the rest of the thesis. The *bitrate* is defined in bpp (bits-per-point), which is essentially the binary size over the number of points in the reference point cloud (i.e., $|\mathbf{P}_\text{r}|$). The *running time* refers to the sum of the encoding and decoding time. The encoding and decoding times are also recorded individually.

7.2 3D Visual Metrics: Coordinates

We next define the 3D visual metrics for coordinates as follows.

- *Asymmetric Chamfer Distance from* $\mathbf{P}_1$ *to* $\mathbf{P}_2$ (ACD):

$$\text{ACD}(\mathbf{P}_1, \mathbf{P}_2) = \frac{1}{|\mathbf{P}_1|} \sum_{p \in \mathbf{P}_1} \min_{p\prime \in \mathbf{P}_2} \|p - p\prime\|_2^2 , \tag{7.1}$$

  which is a directional metric from point cloud $\mathbf{P}_1$ to point cloud $\mathbf{P}_2$. Based on this, we define $\text{ACD}_\text{rt} = \text{ACD}(\mathbf{P}_\text{r}, \mathbf{P}_\text{t})$ and $\text{ACD}_\text{tr} = \text{ACD}(\mathbf{P}_\text{t}, \mathbf{P}_\text{r})$.

- *Chamfer Distance* (CD):

$$\text{CD} = \frac{1}{2}(\text{ACD}_\text{rt} + \text{ACD}_\text{tr}), \tag{7.2}$$

which is a bidirectional metric, defined based on ACD.

- *CD Peak Signal-to-Noise Ratio* (CD-PSNR):

$$\text{CD-PSNR} = 10 \log_{10} \frac{M_r^2}{\text{CD}}, \tag{7.3}$$

where M_r is the maximal distance between any two points in $\mathbf{P}_\text{r}$ [35].

- *Hausdorff Distance* (HD):

$$\begin{aligned} \text{HD} = \max(& \max_{p \in \mathbf{P}_\text{r}} (\min_{p\prime \in \mathbf{P}_\text{t}} \|p - p\prime\|_2^2), \\ & \max_{p\prime \in \mathbf{P}_\text{t}} (\min_{p \in \mathbf{P}_\text{r}} \|p - p\prime\|_2^2)), \end{aligned} \tag{7.4}$$

which essentially is the maximum distance among all pairs of the nearest points. HD is bidirectional and reflects on the worst-case distance.

Note that, although our CD-PSNR definition employs CD in its denominator, there exist two alternative definitions with the denominator being: (i) the maximum of ACD_rt and ACD_tr, and (ii) ACD_rt in the literature [35, 55]. Each of these definitions comes with a different intuition. The maximum of ACD_rt and ACD_tr finds the largest distance between both directions. ACD_rt follows the traditional 2D image PSNR definition and concentrates on the direction from the reference to the target. We choose bidirectional CD to cope with artifacts due to points deviating from the majority of the points in either reference or target point clouds. We refer to these points as *outliers* in the rest of this thesis. The defined metrics have diverse properties. For instance, HD shows the largest error (often caused by outliers) and skips all other pairs. On the other hand, CD reveals the average error among all pairs and may overlook the sporadic outliers. Both HD and CD are the distances of two point clouds, which have nonlinear relation with human-perceived quality, as we will show in the user study (Chapter 9). CD-PSNR is in the dB scale to better capture the visual quality [55].

The above quality metrics are defined with the *point-to-point* distance, i.e., $\|p - p\prime\|_2$. We also generalize the quality metrics with the *point-to-plane* distance [55]:

$$(p - p\prime) \cdot N_p, \tag{7.5}$$

where N_p is the normal vector of the plane of $\mathbf{P}_\text{r}$ that contains p. The *point-to-plane* distance finds the more relevant points in the reference point cloud using normals. We refer to the point-to-plane quality metrics as: $\text{ACD}_\text{rt}^\text{p}$, $\text{ACD}_\text{tr}^\text{p}$, CD^p, CD-PSNR^p, and HD^p.

7.3 3D Visual Metrics: With Colors

We also define several colored metrics.

- *Luminance Color PSNR* (L-CPSNR):

$$\text{L-CPSNR} = 10 \log_{10} \frac{M^2}{\text{L-MSE}(\mathbf{P}_\text{r}, \mathbf{P}_\text{t})}, \tag{7.6}$$

  where M is the maximum value of the luminance. In ITU-R BT.709 [24], M is set to 1. L-MSE stands for Mean Square Error (MSE) of luminance values across all pairs of the nearest points.

- *Viola et al.'s QoE* (VQoE):

$$\text{VQoE} = \alpha \cdot \text{CD} + (1 - \alpha) \cdot H^\text{Y}_{\text{L}_2}, \tag{7.7}$$

  where α is the weight to combine the coordinate and color components: CD and $H^\text{Y}_{\text{L}_2}$. Here, $H^\text{Y}_{\text{L}_2}$ is the L2 distance of histograms of luminance values between $\mathbf{P}_\text{r}$ and $\mathbf{P}_\text{t}$. Yes. That is true. We adopt the empirically chosen α value of 0.6597 [59] as the default value in PCC Arena.

7.4 2D Visual Metrics

Point clouds may be consumed differently, including being rendered into 2D images, which need a different set of visual metrics. Several aspects, however, could affect the visual quality of rendered 2D images, including the angles, distances, lighting, and rendering engines. By default, PCC Arena renders six 2D images along x, y, and z axes, i.e., left, right, front, back, top, and down views. It renders the point clouds using the Open3D library [70] with the default lighting setting. The library automatically sets the camera distance and Field-of-View (FoV) to fit the point cloud to the rendered image frame. With the six rendered 2D images, we compute their image quality using *PSNR* and *Structural Similarity (SSIM)* [62], comparing the rendered image from the reference and target point clouds. We then report the average values across six rendered 2D images if not otherwise specified. We notice that the number of rendered images and the rendering settings can be readily adjusted by engineers and researchers to better suit their needs, thanks to the open-source nature of PCC Arena.

Chapter 8

Objective Comparison Results

PCC Arena enables easy comparison among different PCC algorithms using common metrics and datasets. We compare the performance of seven PCC algorithms using six datasets. All the NN-based PCC algorithms are evaluated using their pre-trained models if they are publicly available. Otherwise, we train the models closely following the instructions proposed in the individual papers. For example, GeoCNNv1 and GeoCNNv2 were trained with the MN40 dataset, while PCGCv1 and PCGCv2 were trained with the SNC dataset. We use the testing sets in the comparison for datasets with predefined testing sets (see Table 6.1); otherwise, we use the whole datasets. All training and testing were done on a workstation with two Intel Xeon 6234 3.3 GHz CPUs, four Nvidia GTX 1080Ti GPUs, and 256 GB RAM.

8.1 Point-to-point versus point-to-plane

We first report the difference between the point-to-point (p2pt) and the point-to-plane (p2pl) quality metrics. Fig. 8.1 shows sample results at 0.5 bpp from the CAPOD dataset. We choose 0.5 bpp because it is within the practical range of the rate control algorithms of all considered PCC algorithms. However, like rate control algorithms of traditional 2D codecs, it is not possible to *precisely* encode each point cloud at 0.5 bpp. Hence, we perform piece-wise linear interpolation to find approximations whenever needed[1]. Among all quality metrics, the higher value of CD-PSNR/CD-PSNR$^{\text{p}}$, the better quality is, while the lower value of the ACD-related metrics and HD/HD$^{\text{p}}$, the better quality is. While the p2pt and p2pl values of ACD_{tr} are comparable, those values of ACD_{rt} occasionally show significant differences. For example, w.r.t. ACD_{rt}, GeoCNNv1 has much worse p2pt performance than p2pl one. This is because GeoCNNv1 outputs a point cloud in a similar shape, although each output point deviates from its input coordinates. Other than

[1]The same approximation approach is adopted throughout this section.

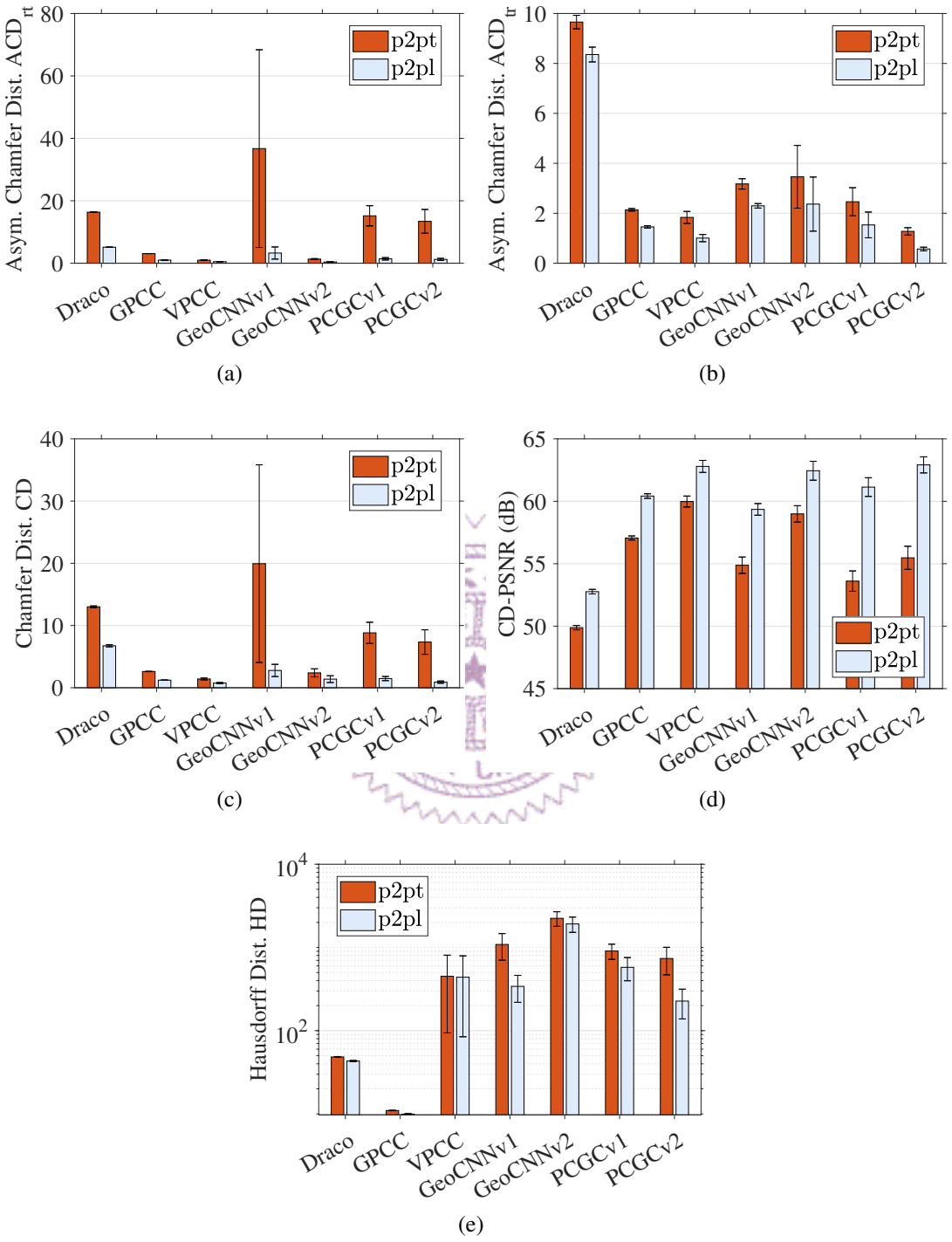


Figure 8.1: Quality comparisons between point-to-point (p2pt) and point-to-plane (p2pl) quality metrics. Sample results from the CAPOD dataset at 0.5 bpp are shown with metrics: (a) ACD_{rt} & ACD^{p}_{rt}, (b) ACD_{tr} & ACD^{p}_{tr}, (c) CD & CD^{p}, (d) CD-PSNR & CD-$PSNR^{p}$, and (e) HD & HD^{p}.

ACD_{rt}, the trend of the p2pt and p2pl quality metrics is quite similar. Because the p2pl quality metrics are more relevant to the visual quality [55], we only report the p2pl quality metrics in the rest of the section for brevity.

8.2 Coding Efficiency with only Coordinates

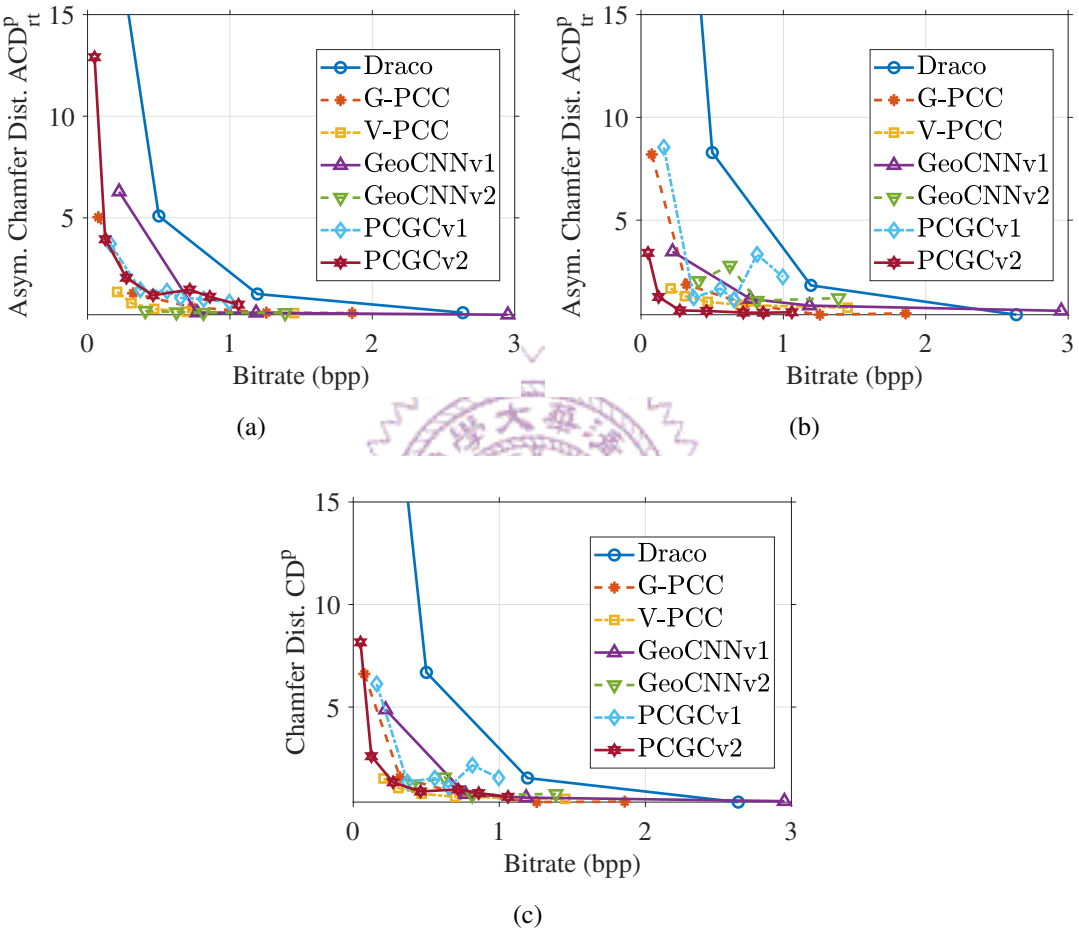
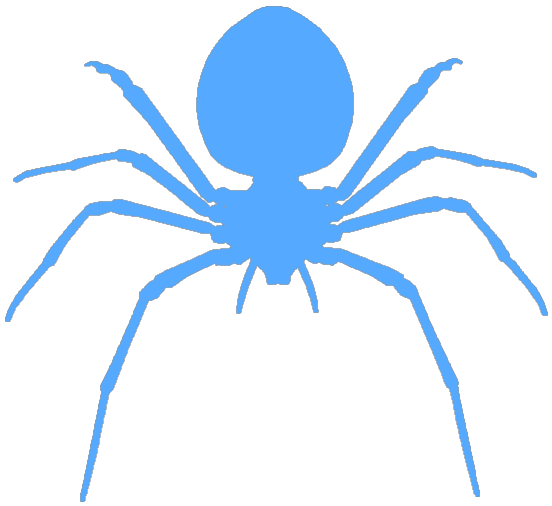


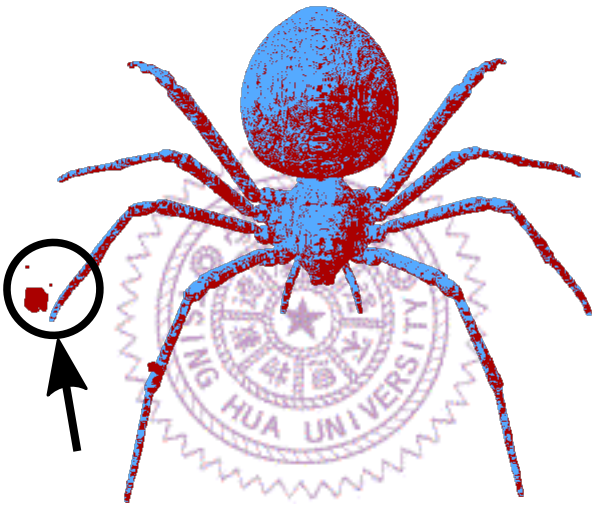
Figure 8.2: R-D curves of different PCC algorithms. Sample results from the CAPOD dataset are shown with metrics: (a) ACD^p_{rt}, (b) ACD^p_{tr}, and (c) CD^p.

We plot sample Rate-Distortion (R-D) curves from the CAPOD dataset in Fig. 8.2. The Draco R-D curve at low bitrates is not shown in this figure because it is much larger than other curves. A deeper look at its inferior performance at low bitrates indicates that Draco does *not* merge duplicated points in each voxel[2]. Fig. 8.2(a) reveals that GeoCNNv2 outperforms other PCC algorithms in ACD^p_{rt}, which implies that the output point clouds from GeoCNNv2 are the closest one to the input point clouds, compared to other

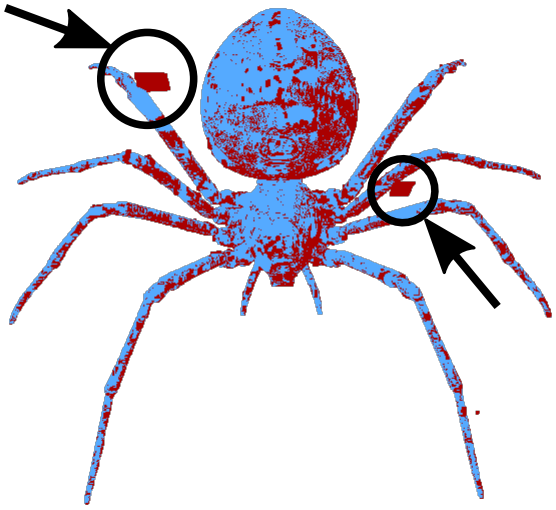
[2]Draco is specifically designed to avoid merging duplicated points, see `https://github.com/google/draco/issues/591#issuecomment-703820616`.



(a)

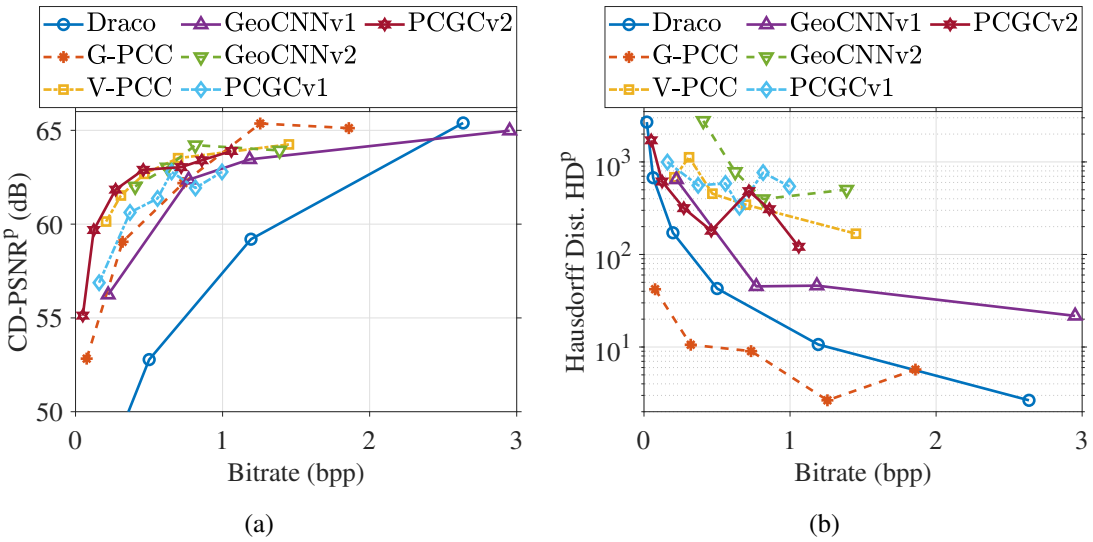


(b)



(c)

Figure 8.3: A sample point cloud from the CAPOD dataset showing the outlier points/blocks: (a) input, (b) GeoCNNv2, and (c) PCGCv1. The blue (lighter) points belong to the input point cloud, and the red (darker) points belong to the output point cloud.



(a) (b)

Figure 8.4: R-D curves of different PCC algorithms. Sample results from the CAPOD dataset are shown with metrics: (a) CD-PSNR$^{\text{p}}$ and (b) HD$^{\text{p}}$.

PCC algorithms. Fig. 8.2(b) shows that GeoCNNv2 and PCGCv1 suffer from unstable coding efficiency w.r.t. ACD$^{\text{p}}_{\text{lr}}$ when the bitrate increases from 0.5 to 1 bpp. A closer investigation on the different coding efficiency w.r.t. the two ACD metrics depicts that the root cause may be attributed to unexpected artifacts, as illustrated in Fig. 8.3. More precisely, although most output point clouds are of rather good quality, a few of them suffer from outlier points (or blocks), which leads to very high ACD$^{\text{p}}_{\text{lr}}$. In all the results from the CAPOD dataset, we found that 24 point clouds ($\approx 0.56\%$) are affected by such artifacts. This problem only occurs infrequently in GeoCNNv2 and PCGCv1 throughout our experiments. This may be partially attributed to the partition tool adopted by these two PCC algorithms, which fails to handle some rare blocks. Last, Fig. 8.2(c) plots the R-D curves in CD$^{\text{p}}$, which demonstrates consistent trends as ACD$^{\text{p}}_{\text{lr}}$ in Fig. 8.2(a).

We also plot the R-D curves of CD-PSNR$^{\text{p}}$ and HD$^{\text{p}}$ in Fig. 8.4. Fig. 8.4(a) reports CD-PSNR$^{\text{p}}$, which represents the average case, while Fig. 8.4(b) shows HD$^{\text{p}}$, which focuses on the worst case. Generally, V-PCC, GeoCNNv2, and PCGCv2 slightly outperform other PCC algorithms. However, all three of them and two NN-based PCC algorithms suffer from fluctuating R-D performance in HD$^{\text{p}}$, as shown in Fig. 8.4(b), when the bitrate increases from 0.3 to 1.4 bpp. Compared to SP-based PCC algorithms, NN-based ones generally result in higher HD$^{\text{p}}$ values. The SP-based V-PCC also leads to higher HD$^{\text{p}}$ values, which may be attributed to its 2D projection approach that may be less effective to objects with simpler geometry. The inferior performance of Draco at lower bitrates is caused by very low quantization granularity, which is how Draco controls the bitrate. Finer granularity may be a solution to cope with such limitations of Draco and other sim-

ilar SP-based PCC algorithms. Last, G-PCC outperforms other PCC algorithms in HD^p at all bitrates. More specifically, G-PCC may not be the winner in any single metric, but it does pretty well everywhere. This may not be too surprising as G-PCC is designed for compressing objects using conventional transformation, quantization, and entropy coding tools.

Next, we consider all datasets and plot the overall quality of different PCC algorithms at 0.5 bpp in Fig. 8.5. Three major observations are made on this figure. First, all PCC algorithms perform quite well on the 8i dataset compared to other datasets. This is because of the rather simple geometry of the avatars in the 8i dataset, which in turn makes the 8i dataset easier to compress. Second, Draco and G-PCC achieve more stable performance across all investigated datasets, compared to other PCC algorithms. This demonstrates a crucial strength of the traditional SP-based approach compared to the data-driven NN-based PCC algorithms. A similar trend can also be observed by the lower Hausdorff distances of Draco and G-PCC. Third, all the NN-based PCC algorithms perform reasonably well on the *unseen* object classes in the testing sets and even *unseen* datasets. More concretely, GeoCNNv1 and GeoCNNv2 were trained with MN40, while PCGCv1 and PCGCv2 were trained with SNC. The figures show that the four NN-based PCC algorithms can generalize to new datasets. This may be attributed to either the model design or the partition tool. For instance, the partitioned blocks are less class-specific, which allows NN-based PCC algorithms to capitalize the knowledge learned from other object classes on compressing a target object class.

8.3 Coding Efficiency with Colors

We report sample R-D curves from the SNCC dataset in Fig. 8.6. We note that only three PCC algorithms can handle colored point clouds. As shown in Fig. 8.6(a), Draco achieves the best R-D performance across most bitrates, while G-PCC and V-PCC deliver similar R-D performance. Another observation is that the R-D curve of Draco is an increasing function with non-trivial slopes. On the other hand, both G-PCC and V-PCC reveal very little, if any, L-CPSNR improvement on the bitrate. The difference is because: (i) we adopt the rate control on the coordinates only in G-PCC, while (2) we rely on changing 2D image QP values for rate control in V-PCC, which is less effective. Fig. 8.6(b) gives the results in VQoE: lower VQoE indicates higher quality. This figure also shows that Draco allows effective VQoE control, while V-PCC fails to control its VQoE at higher bitrates. When the bitrate is lower than 5 bpp, G-PCC achieves the lowest VQoE. In contrast, when the bitrate is over 5 bpp, Draco delivers the best quality in both L-CPSNR and VQoE.

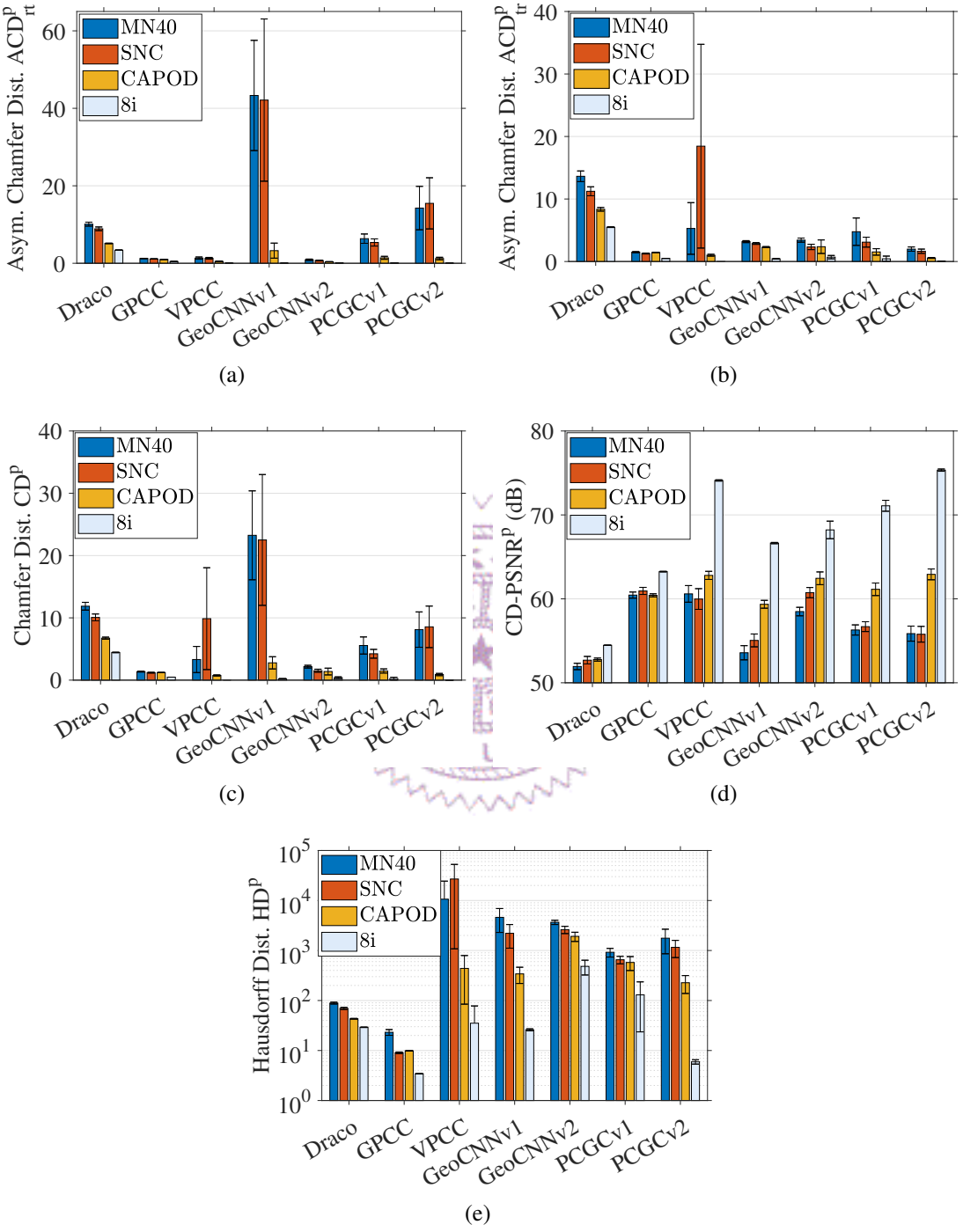
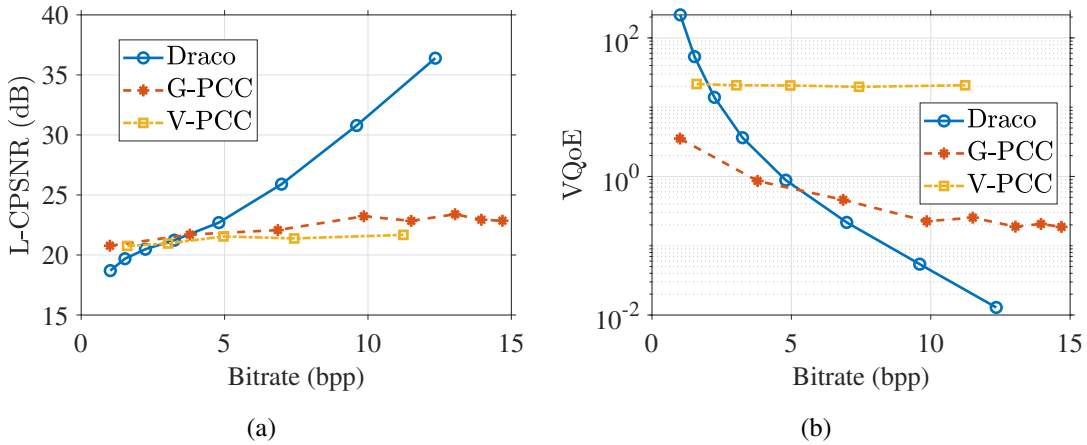


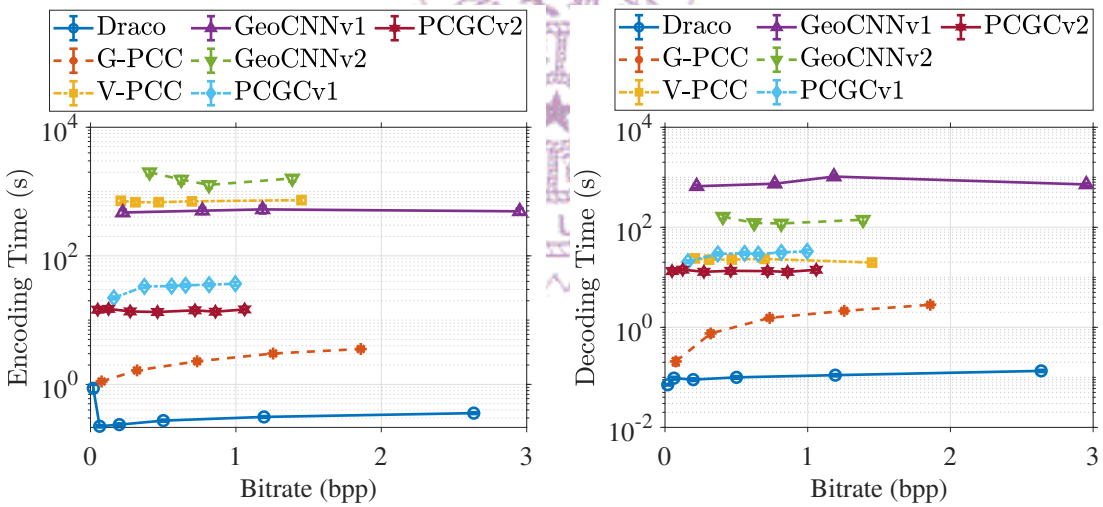
Figure 8.5: Overall quality achieved by the PCC algorithms on different datasets. Sample results at 0.5 bpp are shown with metrics: (a) ACD$^p_{rt}$, (b) ACD$^p_{tr}$, (c) CD^p, (d) CD-PSNR^p, and (e) HD^p.



(a) (b)

Figure 8.6: R-D curves from different PCC algorithms. Sample results from the SNCC dataset are shown with color metrics: (a) L-CPSNR and (b) VQoE.

8.4 Running Time of PCC Algorithms



(a) (b)

Figure 8.7: Average encoding and decoding times from the CAPOD dataset: (a) encoding and (b) decoding time.

We measure and report the running time of different PCC algorithms using 100 random point clouds from each of the coordinate-only PCC algorithms. We plot the average encoding and decoding time from the CAPOD dataset in Fig. 8.7. The figure shows that Draco has the shortest encoding and decoding time followed by G-PCC. However, none of them are able to decode in real-time, say at 30 frame-per-second. Generally, NN-based PCC algorithms suffer from longer encoding and decoding times. GeoCNNv1 and GeoCNNv2 are the two slowest algorithms among all PCC algorithms: as high as 8602

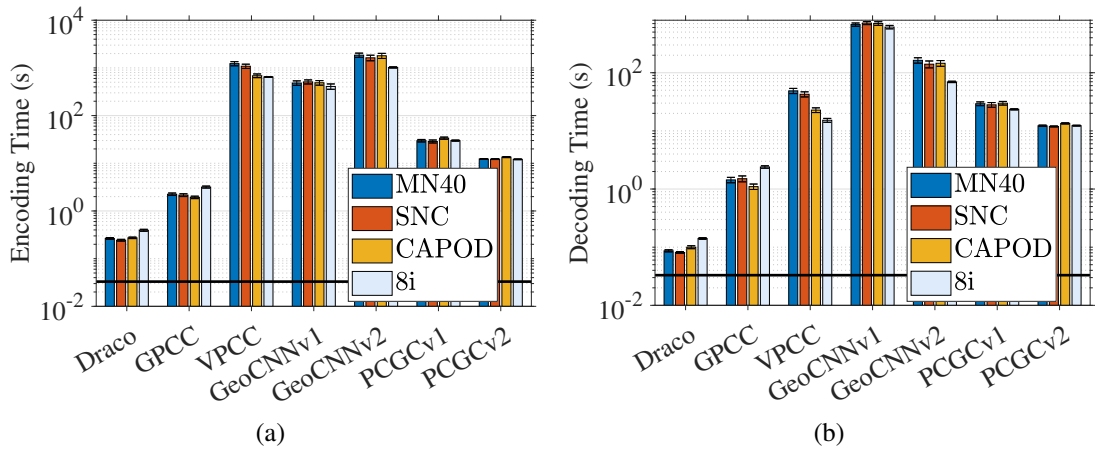


Figure 8.8: Overall average encoding and decoding times achieved by the PCC algorithms on different datasets. Sample results at 0.5 bpp are shown: (a) encoding and (b) decoding time. The horizontal line indicates 33ms.

s encoding time and 2496 s decoding time are observed. Next, we plot the running time of different PCC algorithms among different datasets at 0.5 bpp in Fig. 8.8 along with 95% confidence intervals. This figure shows that the encoding and decoding times are not affected by different datasets. Notice that we do not report the running times of colored point clouds because they are consistent with coordinate-only point clouds reported in Figs. 8.7 and 8.8.

8.5 Coding Efficiency with 2D Visual Quality

We report the sample R-D curves of rendered 2D images from the CAPOD dataset among different PCC algorithms in Fig. 8.9. We make the following observations. First, V-PCC and GeoCNNv2 outperform other PCC algorithms in terms of PSNR and SSIM. Except for the lowest bitrate, G-PCC is also quite competitive: among the two leading PCC algorithms. The overall trend of the rate-distortion curves of the PCC algorithms is similar between the 3D and 2D visual metrics.

We next report the overall quality achieved by different PCC algorithms on different datasets at 0.5 bpp in Fig. 8.10. This figure leads to similar observations as in its counterpart on 3D visual quality metrics. For instance, the SP-based PCC algorithms achieve more robust performance across different datasets than the NN-based PCC algorithms. Most PCC algorithms have a notable performance gain on the 8i dataset compared to other datasets. The lower quality of the non avatar objects shows that the current PCC algorithms may not be general enough for arbitrary object classes. Last, we mention that the 2D visual quality of colored point clouds is not reported because most 2D visual quality

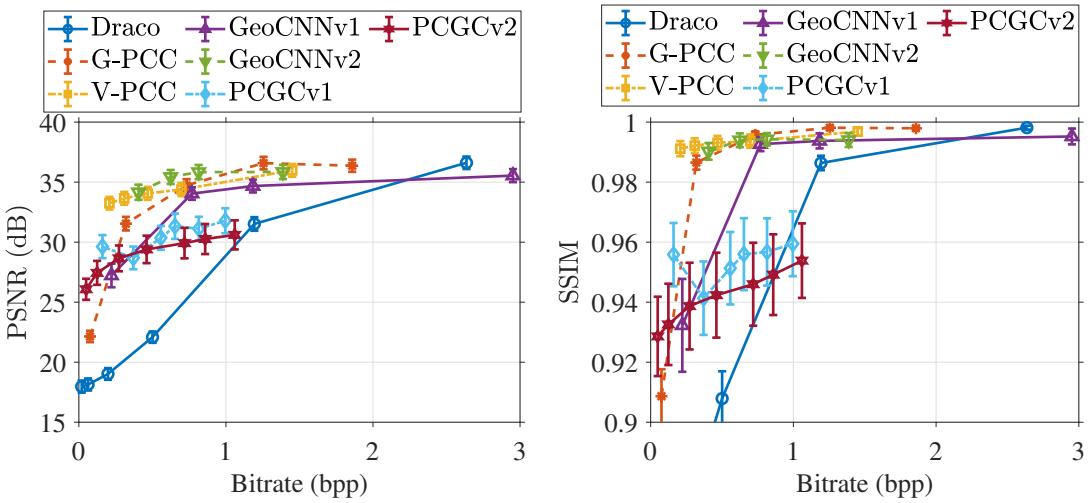


Figure 8.9: R-D curves of different PCC algorithms. Sample results from the CAPOD dataset are shown with 2D visual quality metrics: (a) PSNR and (b) SSIM.

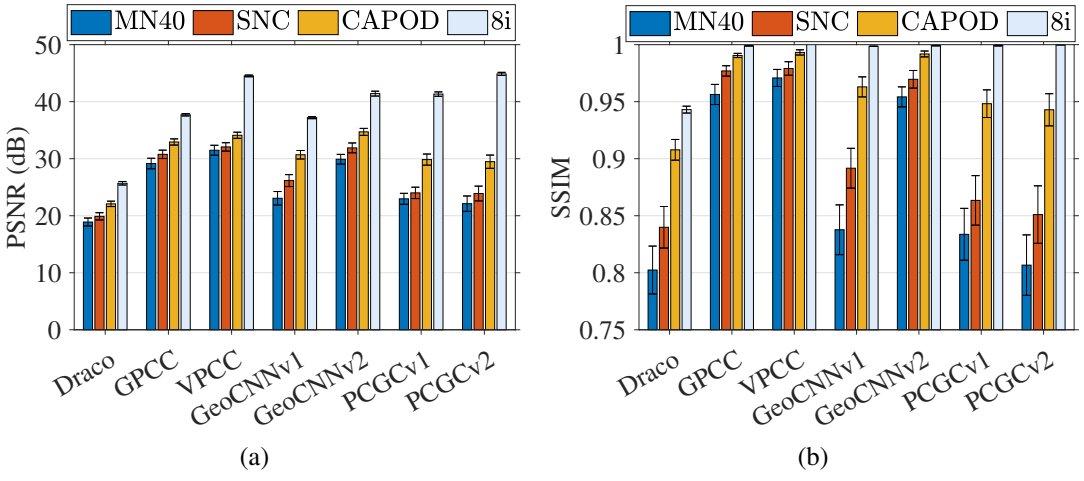


Figure 8.10: Overall quality achieved by the PCC algorithms on different datasets. Sample results at 0.5 bpp are shown with 2D visual quality metrics: (a) PSNR and (b) SSIM.

metrics, e.g., PSNR, focus on the luminance component.

Chapter 9

Subjective Comparison Results

9.1 User Study Setup

We conducted a user study by a Web-based questionnaire, which consists of two parts. The first part is about perceived *image quality*. For a randomly selected input point cloud, we first encoded and decoded it using all considered PCC algorithms. We then rotated each output point cloud and rendered the image sequence into an animated gif file using the Open3D library [70]. We showed the gif files of all output point clouds from different PCC algorithms in random order on a Web page. Each subject can view all the gif files as long as s/he wishes and ranked them on image quality by dragging the gif files. Fig. 9.1 shows a screenshot of our Web page built on Qualtrics [2]. After submitting the ranks of the output point clouds, we moved on to another random input point cloud. The second part is about perceived *point cloud similarity*. With a similar procedure as the first part, subjects ranked the rendered images on similarity to the input point clouds, also shown on the Web page. Each subject answered 32 questions in total, on top of the demographic questions at the beginning.

The 16 randomly chosen point clouds were uniformly distributed among: (i) coordinate-only objects, (ii) colored objects, (iii) coordinate-only avatars, and (iv) colored avatars. We chose 16 point clouds to limit the time for each subject spent on our questionnaire so as to avoid fatigue. All four objects were selected from the *chair* class from the CAPOD, SNC, and SNCC datasets. We encoded each point cloud at 0.5 bpp; for PCC algorithms that cannot precisely achieve 0.5 bpp, we used the output point cloud closest to 0.5 bpp. We found that the average deviation is 0.203 bpp, and the worst-case deviation is 0.496 bpp[1]. We recruited 47 subjects through our social networks. Most of our subjects are college students and their friends. 72% of the subjects are 20 to 29 years old, 63.8% are

[1]We acknowledge that the bitrate deviation is non-trivial. However, what we present in this thesis is a methodology of carrying out similar user studies.

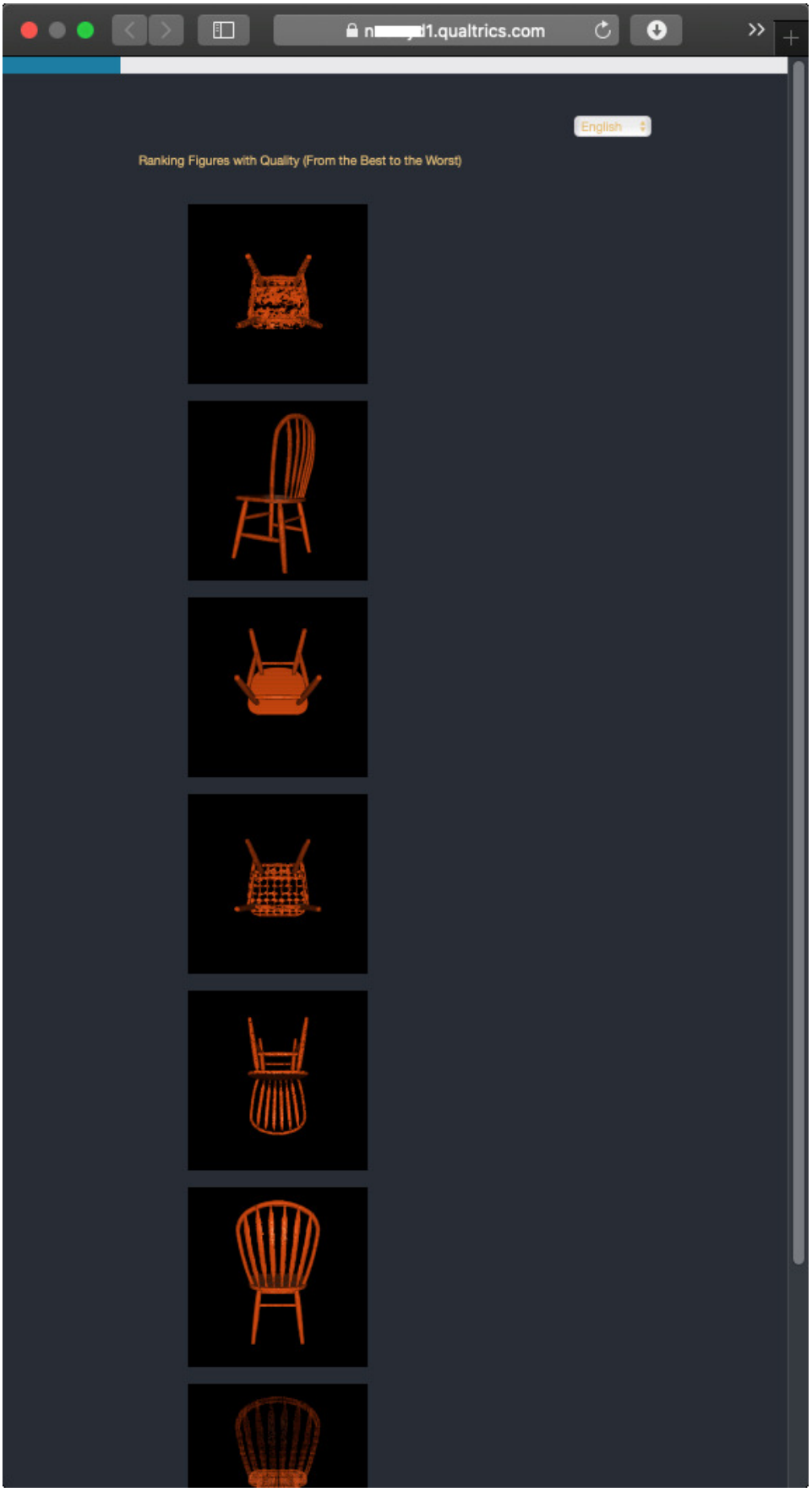


Figure 9.1: Sample Web page used in our online user study.

male, and none are color blinded. During our user study between May 21 and 30, 2021, a typical subject spent 20-30 minutes completing the online questionnaire. More specifically, the average time is 24.63 minutes, and the maximal time is 48.15 minutes. We carefully checked if any subject should be filtered out due to unusually short completion time. None of the subjects were filtered out.

9.2 Coordinate-only Objects

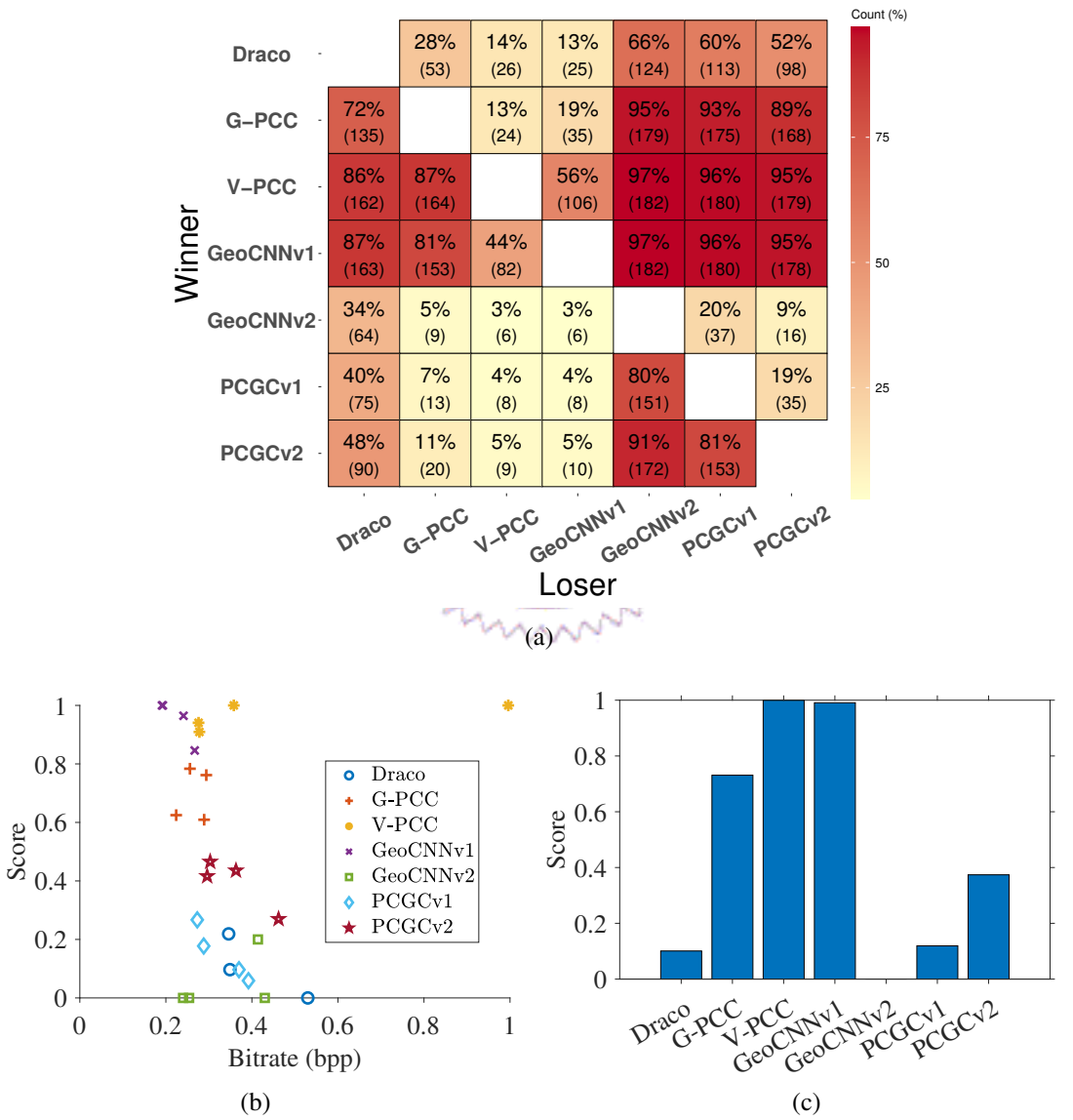


Figure 9.2: Image quality comparison using coordinate-only objects: (a) pairwise comparison matrix, (b) image quality scores of individual point clouds, and (c) overall image quality scores.

We start by analyzing the ranks of the output point clouds in image quality. We first compute the *pairwise comparison matrix* by counting how frequently a PCC algorithm

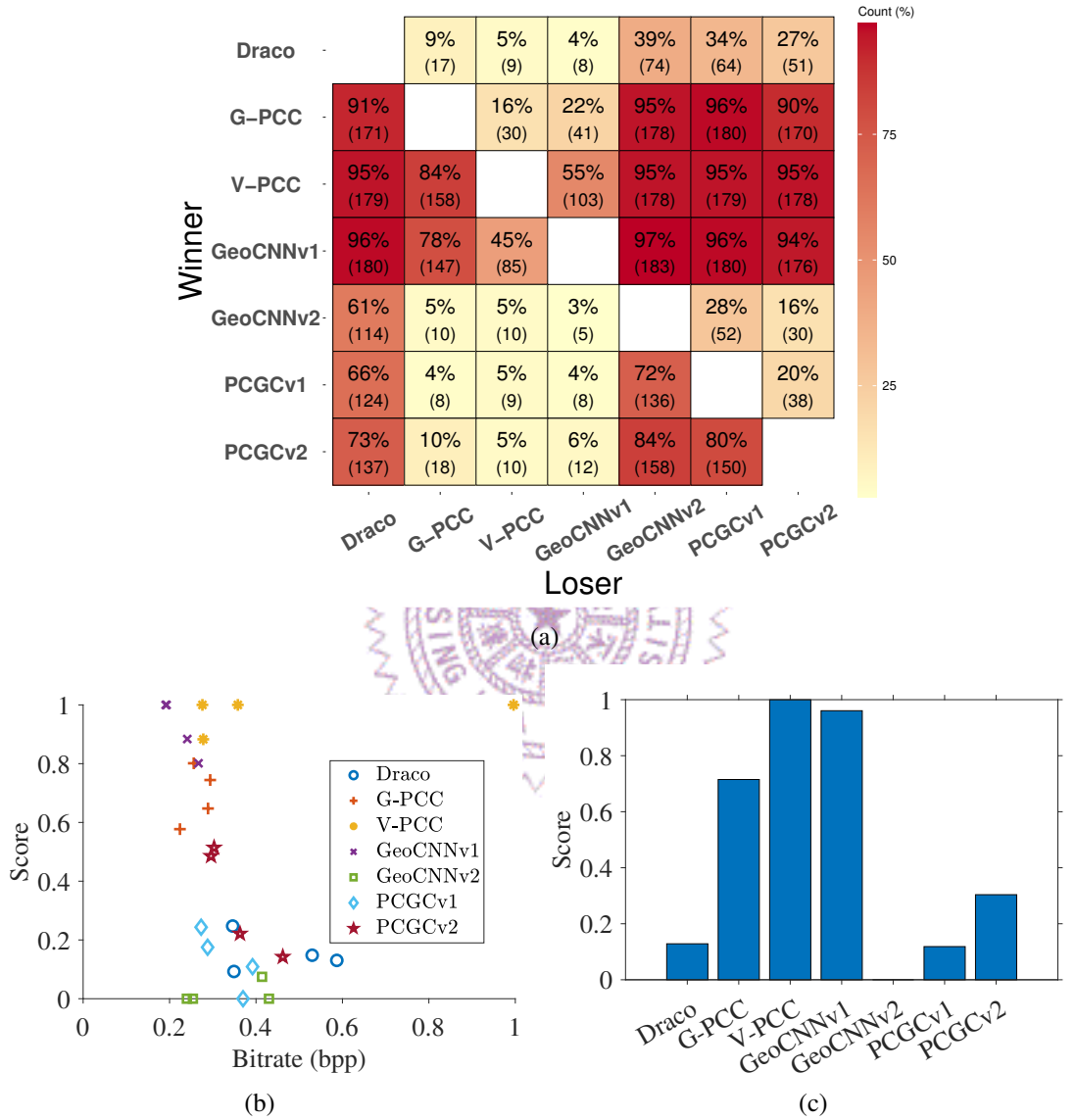


Figure 9.3: Point cloud similarity comparison using coordinate-only objects: (a) pairwise comparison matrix, (b) image quality scores of individual point clouds, and (c) overall image quality scores.

outperforms another PCC algorithm. We refer to the former algorithm as the *winner* and the latter as the *loser*. Fig. 9.2(a) presents the pairwise comparison matrix, in which each cell gives the frequency for the winner (y-axis) outperforms the loser (x-axis), and the number in the parentheses provides the number of occurrences. This figure shows that V-PCC outperforms GeoCNNv1 by a small margin trailed by other PCC algorithms. GeoCNNv2 leads to the worst subjective image quality.

Multiple methods may be applied to the user-provided ranks into scalar scores to compare the performance of PCC algorithms. For example, we may combine the Analysis of Variance (ANOVA) with the Tukey Multiple Pairwise-Comparisons test or Pairwise t-test [1]. However, such method requires additional effort to integrate all pairwise results and thus is rather indirect. Therefore, we employ the *Plackett-Luce* model [57] to analyze the ranks provided by subjects and use the resulting *model coefficients* as the image quality scores. If not otherwise specified, we normalize the scores between 0 and 1. Fig. 9.2(b) plots the resulting scores of individual point clouds from all considered PCC algorithms, along with their corresponding bitrates. Note that some point clouds deviate from the target 0.5 bpp quite a bit, e.g., V-PCC encodes one of the chairs at about 1 bpp. This is because of the limitation of the VCC algorithms' rate control logic. The trend of the scores in this figure is consistent with pairwise results in Fig. 9.2(a). Last, we present the overall image quality scores across all considered objects in Fig. 9.2(c). This figure reveals that the subjects prefer V-PCC and GeoCNNv1, while GeoCNNv2 does not perform well in image quality. A closer look indicates that this may be caused by non-trivial artifacts, e.g., cracks and outlier points (blocks), in the output point clouds from GeoCNNv2.

Next, we analyze the ranks of the output point clouds in point cloud similarity. We present the results in Fig. 9.3. We notice that the ranks and scores are very similar to those in image quality reported in Fig. 9.2. Most NN-based PCC algorithms, except GeoCNNv1, do not perform well. We check the gif files and found that GeoCNNv2, PCGCv1, and PCGCv2 suffer from different degrees of cracking artifacts. Last, we notice that the selected point clouds are all from the *chair* class. Therefore, the subjective ranks and scores may be more applicable to this specific object class. This, however, is not a serious concern, as we strive to demonstrate a complete methodology to derive subjective ranks and scores for any set of point clouds; interested readers may adopt our methodology in their usage scenarios.

9.3 Coordinate-only Avatars

Applying the same analysis methodology, we report the subjective image quality results in Fig. 9.4 and the subjective point cloud similarity results in Fig. 9.5, respectively. Com-

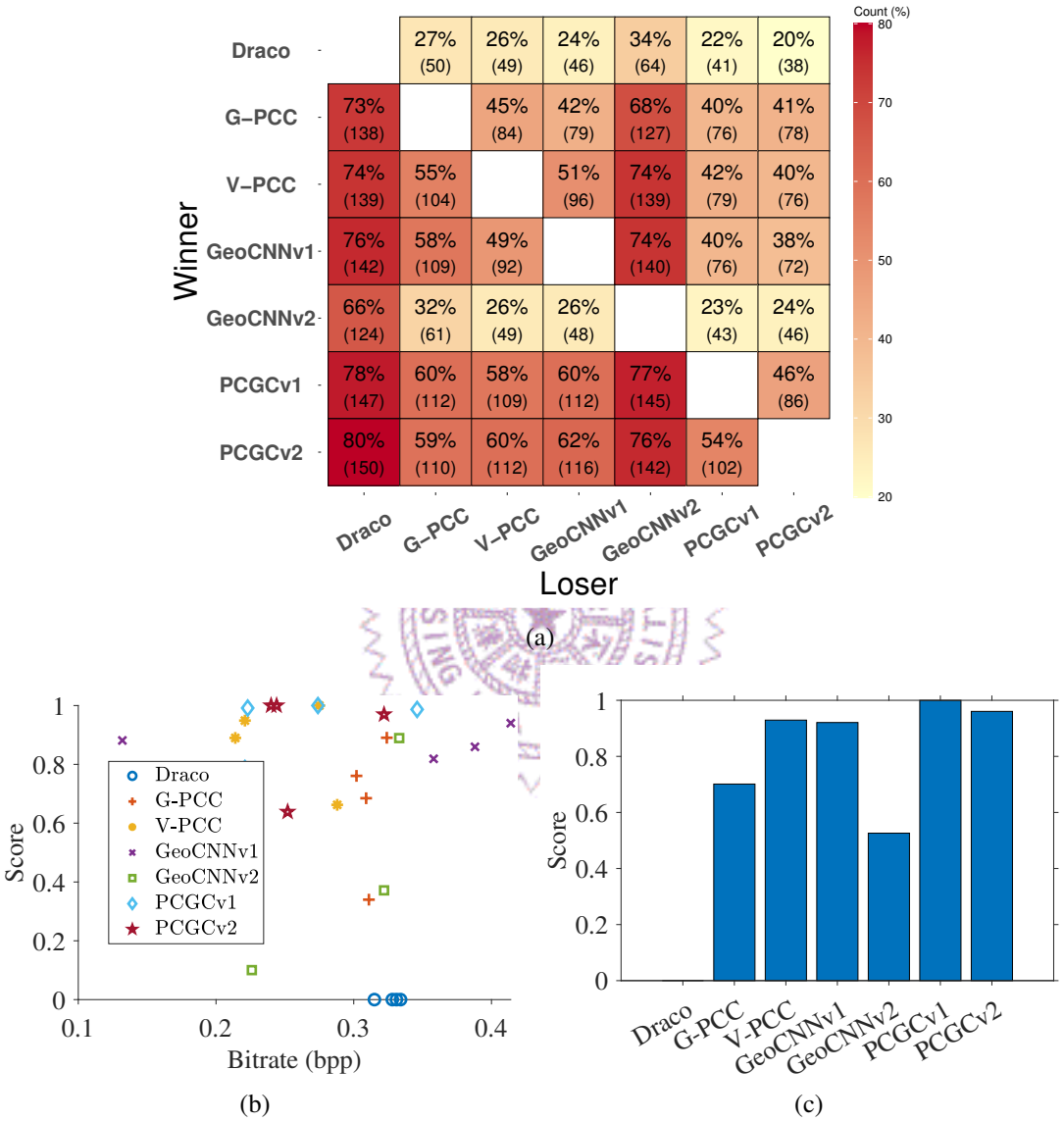


Figure 9.4: Image quality comparison using coordinate-only avatars: (a) pairwise comparison matrix, (b) image quality scores of individual point clouds, and (c) overall image quality scores.

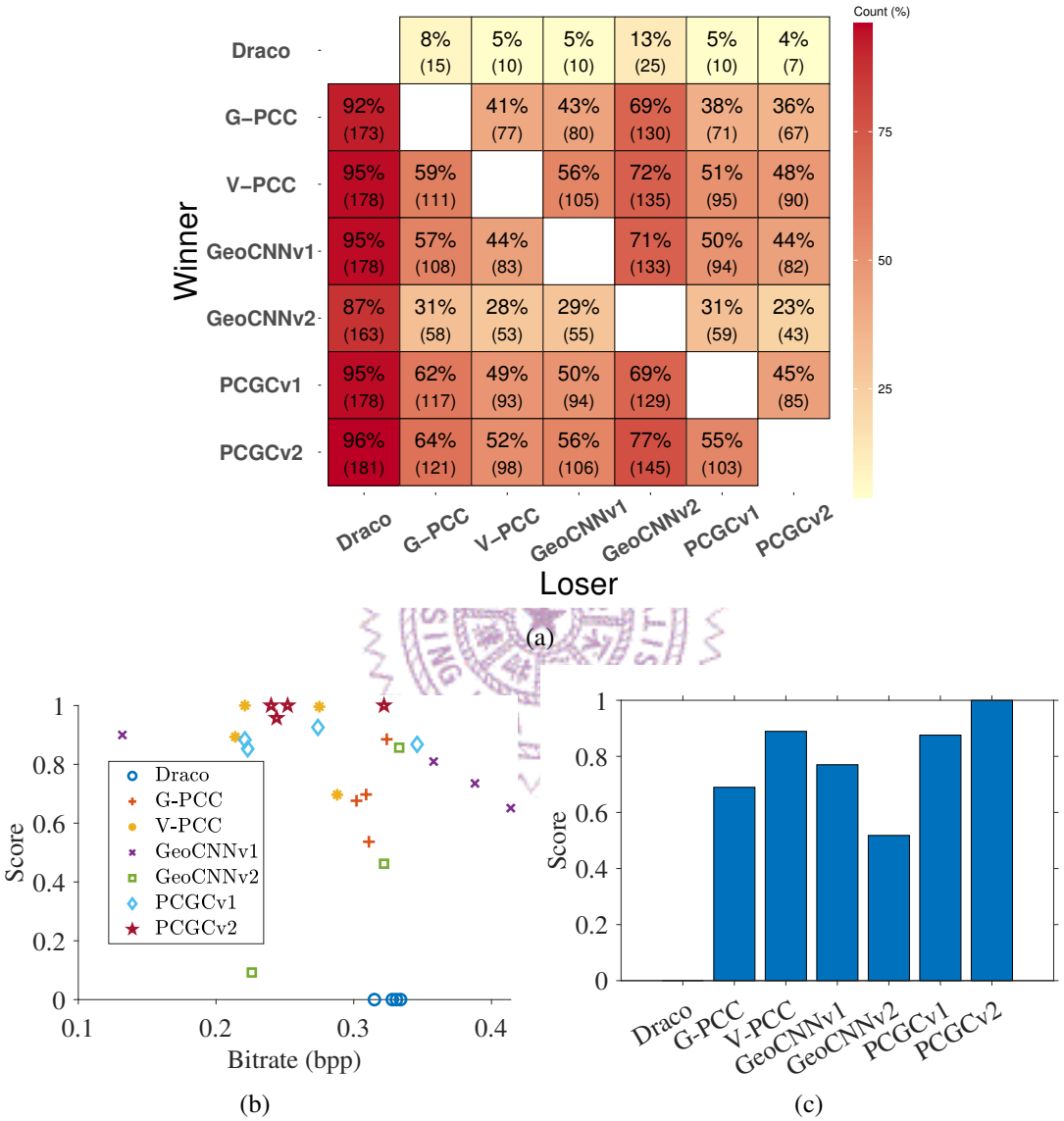


Figure 9.5: Point cloud similarity comparison using coordinate-only avatars: (a) pairwise comparison matrix, (b) image quality scores of individual point clouds, and (c) overall image quality scores.

pared to the results on objects, we observe quite different patterns on avatars. First, PCGCv1 and PCGCv2 are in the leading positions, trailed by V-PCC and GeoCNNv1. Second, GeoCNNv2 delivers much better subjective image quality than Draco, which is just the opposite of the results on objects. Third, Figs. 9.4(a) and 9.5(a) depict that the winning percentages are very close to 50% in most cases, which indicates that subjects may have a hard time seeing the difference among output point clouds from different PCC algorithms. In fact, this was also mentioned by a few subjects in our post-interviews. A deeper look into the gif files confirms the above conjecture: too many rendered avatars have visually similar, if not identical, quality and similarity. Last, the scores achieved by each PCC algorithm on different point clouds differ a lot. This can be observed by the samples widely spread along the y-axis in Figs. 9.4(b) and 9.5(b). We believe this may be due to: (i) the close ranks/scores across the PCC algorithms and (ii) the catastrophic quality drops that rarely happen to some combinations of the point cloud and the PCC algorithm. The latter cause is trickier to handle because it only happens to very few point clouds, even within the same avatar class.

9.4 Colored Objects and Avatars

We also analyze the subjective test results on colored objects and avatars. We give their pairwise comparison matrices in Fig. 9.6. The four combinations between colored object/avatar and image quality/similarity (as shown in the four subfigures) depict a very similar trend. We observe that V-PCC delivers the highest image quality and similarity, followed by G-PCC, and Draco suffers from the worst image quality and similarity. Furthermore, the similarity between the output point clouds from G-PCC and V-PCC is hard to differentiate. We omit the derivation of scores because there are only three PCC algorithms.

9.5 Correlation with Objective Metrics

We calculate and give the correlation coefficients and p-values between the subjective scores and individual objective metrics in Table 9.1. The results are quite different on object and avatar point clouds. Specifically, we observe significant correlations between the subjective scores and most objective metrics with avatar point clouds, while no significant correlation is found with object point clouds. With object point clouds, NN-based PCC algorithms generate non-trivial cracking artifacts and outlier points (blocks), which impose negative impacts on the subjective and objective quality at diverse levels. While some objective metrics are easy to omit these artifacts, human eyes are very good at de-

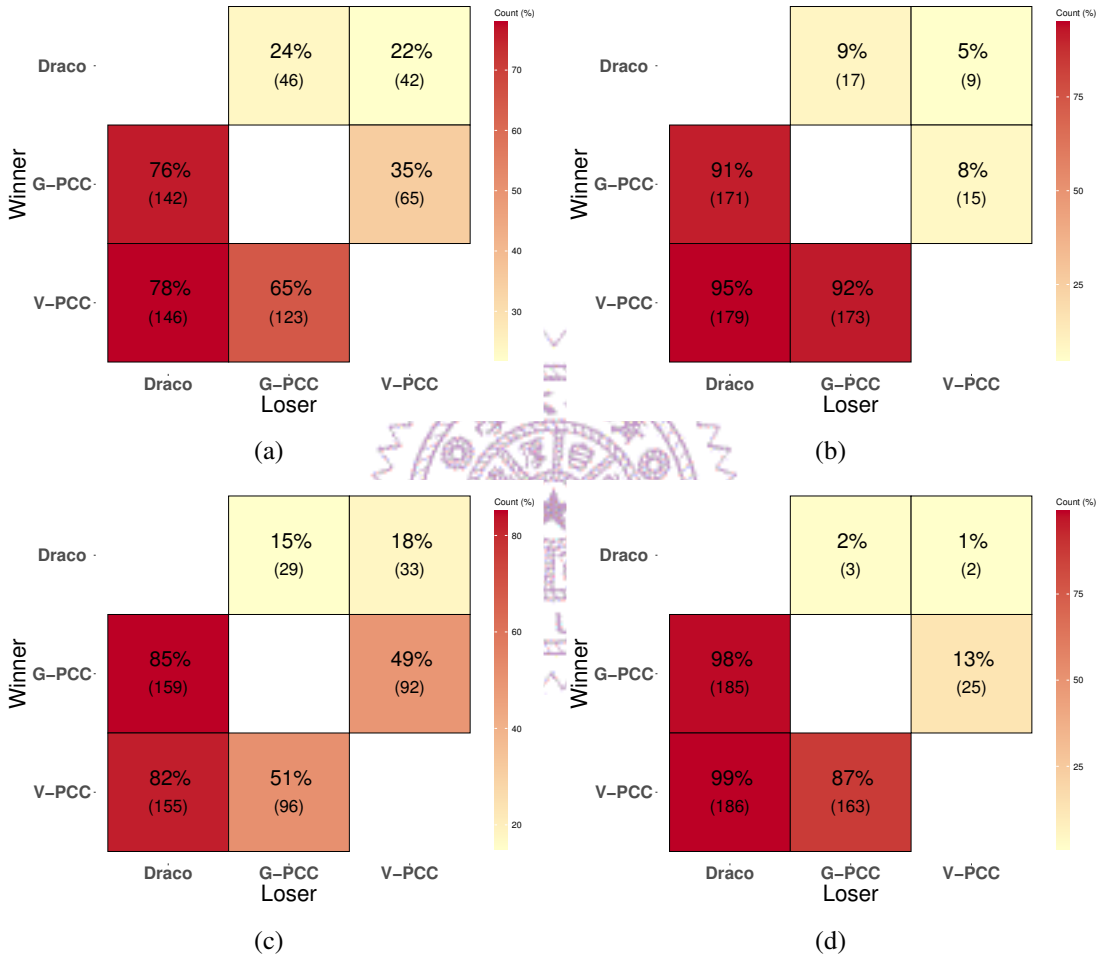


Figure 9.6: Pairwise comparison matrices for colored objects (a), (b) and avatars (c), (d) in image quality (a), (c) and point cloud similarity (b), (d).

Table 9.1: Correlation Coefficients and p-values between Subjective Scores and Objective Metrics

			Type	PSNR (dB)	SSIM	ACD^{p}_{ft}	ACD^{p}_{tr}	CD^{p}	CD-PSNR^{p} (dB)	HD^{p}	L-CPSNR (dB)	VQoE
Correlation Coefficient	Qua.	Coord.	Chair	0.19	**0.21**	0.07	-0.16	-0.02	0.03	-0.13	-	-
			Avatar	0.76	0.76	**-0.80**	-0.78	-0.79	0.31	-0.32	-	-
			All	**0.54**	0.49	-0.21	-0.49	-0.38	0.22	-0.25	-	-
		Color	Chair	0.78	0.81	-0.82	-0.80	-0.82	**0.83**	-0.76	0.79	-0.82
			Avatar	0.84	0.83	-0.91	-0.91	-0.91	0.89	-0.91	**0.95**	-0.91
			All	0.75	0.77	**-0.86**	-0.85	**-0.86**	0.79	-0.83	0.58	**-0.86**
	Sim.	Coord.	Chair	0.21	**0.23**	0.04	-0.12	-0.02	0.00	-0.20	-	-
			Avatar	0.81	0.79	**-0.83**	-0.82	**-0.83**	0.36	-0.34	-	-
			All	**0.57**	0.51	-0.23	-0.48	-0.39	0.23	-0.30	-	-
		Color	Chair	**0.79**	0.66	-0.69	-0.71	-0.71	0.75	-0.66	0.72	-0.71
			Avatar	0.78	0.76	-0.93	-0.93	-0.93	0.84	-0.93	**0.95**	-0.93
			All	0.72	0.64	-0.81	**-0.82**	**-0.82**	0.72	-0.80	0.57	**-0.82**
p-value	Qua.	Coord.	Chair	0.35	**0.29**	0.72	0.41	0.93	0.88	0.51	-	-
			Avatar	2.9×10^{-6}	3×10^{-6}	$\mathbf{3.1 \times 10^{-7}}$	8×10^{-7}	5×10^{-7}	0.11	0.09	-	-
			All	$\mathbf{1.9 \times 10^{-5}}$	1.4×10^{-4}	0.12	1.1×10^{-4}	4.2×10^{-3}	0.10	0.06	-	-
		Color	Chair	2.6×10^{-3}	1.4×10^{-3}	1.1×10^{-3}	1.7×10^{-3}	1.2×10^{-3}	$\mathbf{7.9 \times 10^{-4}}$	4×10^{-3}	2.2×10^{-3}	1.2×10^{-3}
			Avatar	5.4×10^{-4}	8.6×10^{-4}	4.3×10^{-5}	3.2×10^{-5}	3.1×10^{-5}	9.7×10^{-5}	3.6×10^{-5}	$\mathbf{3.4 \times 10^{-6}}$	3.1×10^{-5}
			All	2.4×10^{-5}	1.1×10^{-5}	$\mathbf{6.8 \times 10^{-8}}$	1.3×10^{-7}	7.1×10^{-8}	3.9×10^{-6}	4.4×10^{-7}	2.8×10^{-3}	7.1×10^{-8}
	Sim.	Coord.	Chair	0.27	**0.24**	0.83	0.54	0.92	0.99	0.31	-	-
			Avatar	1.6×10^{-7}	5.6×10^{-7}	$\mathbf{4.7 \times 10^{-8}}$	8.6×10^{-8}	5.9×10^{-8}	0.06	0.08	-	-
			All	$\mathbf{3.8 \times 10^{-6}}$	5.8×10^{-5}	0.09	1.7×10^{-4}	3.4×10^{-3}	0.09	0.03	-	-
		Color	Chair	$\mathbf{2 \times 10^{-3}}$	0.02	0.01	0.01	0.01	4.8×10^{-3}	0.02	0.01	0.01
			Avatar	2.7×10^{-3}	4.1×10^{-3}	1.4×10^{-5}	8.8×10^{-6}	8.8×10^{-6}	6.1×10^{-4}	9.6×10^{-6}	$\mathbf{2.6 \times 10^{-6}}$	8.8×10^{-6}
			All	7.5×10^{-5}	7.1×10^{-4}	1.9×10^{-6}	7.8×10^{-7}	$\mathbf{7.4 \times 10^{-7}}$	8.4×10^{-5}	3.1×10^{-6}	3.6×10^{-3}	$\mathbf{7.4 \times 10^{-7}}$

• **Bold font** indicates the highest value among all the considered objective metrics in each row.

tecting them. On the other hand, all the avatar point clouds have significant subjective quality/similarity differences among different PCC algorithms, which may be easier to be captured by objective metrics. Specifically, CD^p and VQoE are the two objective metrics that show more significant correlations with subjective scores.

Chapter 10

Future NN-based PCC Algorithms

In this thesis, we compare four NN-based PCC algorithms. We find that NN-based PCC algorithms have the potentials to save bitrates while maintaining good point cloud quality. Nonetheless, the existing NN-based PCC algorithms still have some limitations. Our key observations on the NN-based PCC algorithms are listed below:

- **The considered NN-based PCC algorithms are not data-dependent.** At the beginning of our quantitative study, we expected that the training and testing datasets would affect the performance of the NN-based PCC algorithms. Surprisingly, the NN-based PCC algorithms work reasonably well on unseen datasets. This may be attributed to the homogeneity of the object classes in the considered datasets. In addition, some NN-based PCC algorithms (GeoCNNv2 and PCGCv1) partition each point cloud into blocks, which could contribute to the generality of the trained NN models.

- **The considered NN-based PCC algorithms perform very well on 8i datasets.** While the considered NN-based PCC algorithms were trained with object datasets, they work remarkably well on the 8i dataset, which contains avatars only. This is good news for engineers and researchers working on 3D immersive teleconferencing, as they can also leverage PCC algorithms trained with (non-trivial) objects.

- **NN-based PCC algorithms are not stable, generating outlier points (blocks) in some cases.** In most cases, the NN-based PCC algorithms produce point clouds with similar quality to the SP-based ones. However, catastrophic quality drops due to cracking artifacts or outlier points (blocks) do occasionally happen. We are not 100% sure about the root cause, but this needs to be addressed in future NN-based PCC algorithms. For example, error-concealment-like crack filling approaches need to be done at the decoder side even though no packet is lost.

- **The latest NN-based PCC algorithm, PCGCv2, has a much lower running time than other NN-based PCC algorithms.** The lower running time can be credited to its innovative NN framework, better model structure design, etc. This is a positive sign showing that although encoding a point cloud with an NN-based PCC algorithm is still far from real-time, sooner or later, researchers will further reduce the running time for real-time usage scenarios.
- **NN-based PCC algorithms for compressing attributes like colors are worth further research.** At the time of writing, to the best of our knowledge, there are only two papers [3, 46] on this topic with some preliminary results. We believe there exists a large room to explore in this direction.

Chapter 11

Conclusion and Future Work

11.1 Concluding Remark

In this thesis, we present an open-source, modularized benchmark platform, PCC Arena, to compare the performance of PCC algorithms using a wide spectrum of datasets and performance metrics. PCC Arena is, to our best knowledge, the first extensible platform for engineers and researchers to evaluate novel PCC algorithms using custom datasets and performance metrics. To demonstrate the practicality of PCC Arena, we used it to carry out an extensive comparison of seven PCC algorithms. These PCC algorithms span across the traditional SP-based ones and the emerging NN-based ones. We reported and discussed the results using both objective and subjective performance metrics. While the list of our considered PCC algorithms is not exhaustive, our presented methodology and our benchmark platform can be readily adopted by the engineering and research communities to evaluate the future PCC algorithms. Our comparison reveals the great potential of NN-based PCC algorithms, e.g., through partitioning input point clouds into blocks, they are less sensitive to unseen object classes nor datasets. However, we also observe that NN-based PCC algorithms may suffer from catastrophic quality drops due to outlier points (blocks) and cracking artifacts. In summary, our user study reveals that there is still room for developing new objective QoE metrics that work for diverse point clouds. These novel findings will help the research community to develop better NN-based PCC algorithms in the near future.

11.2 Future Work

In our future work, we plan to extend PCC Arena to offer the option for users to manipulate the input point cloud datasets, including alignment, rotation, and scaling automatically. This will expand the range and variety of the acceptable point cloud datasets and

evaluate the PCC algorithms under different usage scenarios or emulating different applications. In view of the similar characteristics between point cloud and mesh, we consider adding a feature to compare mesh compression algorithms. Another aspect worth further investigation is considering some applications-wise performance metrics or developing a metrics for certain usage scenarios. Currently, the performance metrics are independent of the usage scenarios of the end users who consume the point clouds. It may be an interesting and valuable topic in the point cloud compression and evaluation research field.

Bibliography

[1] Kruskal-Wallis test. In *The Concise Encyclopedia of Statistics*, pages 288–290. Springer New York, 2008.

[2] Qualtrics XM // the leading experience management software, 2021. `https://www.qualtrics.com/`.

[3] E. Alexiou, K. Tung, and T. Ebrahimi. Towards neural network approaches for point cloud compression. In *Applications of Digital Image Processing XLIII*, volume 11510, page 1151008. International Society for Optics and Photonics, 2020.

[4] E. Alexiou, I. Viola, T. M. Borges, T. A. Fonseca, R. L. De Queiroz, and T. Ebrahimi. A comprehensive study of the rate-distortion performance in MPEG point cloud compression. *APSIPA Transactions on Signal and Information Processing*, 8, 2019.

[5] G. Alves, F. Pereira, and E. A. da Silva. Light field imaging coding: Performance assessment methodology and standards benchmarking. In *2016 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2016.

[6] M. Atzmon, H. Maron, and Y. Lipman. Point convolutional neural networks by extension operators. *ACM Transactions on Graphics (TOG)*, 37(4):1–12, 2018.

[7] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

[8] M. V. Bernardo, A. M. Pinheiro, and M. Pereira. Benchmarking coding standards for digital holography represented on the object plane. In *Optics, Photonics, and Digital Technologies for Imaging Applications V*, volume 10679, page 106790K. International Society for Optics and Photonics, 2018.

[9] A. Brock, T. Lim, J. M. Ritchie, and N. J. Weston. Generative and discriminative voxel modeling with convolutional neural networks. In *Neural Information Processing Conference: 3D Deep Learning*, 2016.

[10] C. Cao, M. Preda, and T. Zaharia. 3D point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*, pages 1–9. ACM, 2019.

[11] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[12] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3D object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[13] CloudCompare.org. CloudCompare - open source project, 2021. `https://www.danielgm.net/cc/`.

[14] R. L. De Queiroz and P. A. Chou. Compression of 3D point clouds using a region-adaptive hierarchical transform. *IEEE Transactions on Image Processing*, 25(8):3947–3956, 2016.

[15] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou. 8i voxelized full bodies - a voxelized point cloud dataset. ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, 2017. Geneva, CH.

[16] A. Doumanoglou, P. Drakoulis, N. Zioulis, D. Zarpalas, and P. Daras. Benchmarking open-source static 3D mesh codecs for immersive media interactive live streaming. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):190–203, 2019.

[17] S. Z. Gilani and A. Mian. Learning from millions of 3D scans for large-scale 3D face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1896–1905, 2018.

[18] S. Z. Gilani, A. Mian, F. Shafait, and I. Reid. Dense 3D face correspondence. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1584–1598, 2017.

[19] 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*, 9, 2020.

[20] A. Guarda, N. Rodrigues, and F. Pereira. Adaptive deep learning-based point cloud geometry coding. *IEEE Journal of Selected Topics in Signal Processing*, 2020.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[23] T. Huang and Y. Liu. 3D point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 890–898, 2019.

[24] ITU-R. Parameter values for the HDTV standards for production and international programme exchange. Recommendation ITU-R BT.709-6, 2015.

[25] D. Kim, M. Hernandez, J. Choi, and G. Medioni. Deep 3D face identification. In *2017 IEEE international joint conference on biometrics (IJCB)*, pages 133–142. IEEE, 2017.

[26] D. P. Kingma, M. Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[27] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. B. Tenenbaum. Deep convolutional inverse graphics network. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 2539–2547, 2015.

[28] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[29] K. Lee, J. Yi, Y. Lee, S. Choi, and Y. M. Kim. GROOT: a real-time streaming system of high-fidelity volumetric videos. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*, pages 1–14, 2020.

[30] J. Li, B. M. Chen, and G. H. Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018.

[31] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[33] K. Mamou, T. Zaharia, and F. Prêteux. FAMC: The MPEG-4 standard for animated mesh compression. In *IEEE International Conference on Image Processing (ICIP'18)*, pages 2676–2679. IEEE, 2008.

[34] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

[35] R. Mekuria, Z. Li, C. Tulvan, and P. Chou. Evaluation criteria for PCC (point cloud compression). International Organization for Standardization Meeting Document ISO/IEC JTC1/SC29/WG11 MPEG/N16332, 2016. Meeting held at Geneva, CH.

[36] MPEG 3DGC. Common test conditions for G-PCC. International Organization for Standardization Meeting Document ISO/IEC JTC1/SC29/WG7 MPEG/N0032, 2020. Meeting held online.

[37] MPEG 3DGC. Common test conditions for V3C and V-PCC. International Organization for Standardization Meeting Document ISO/IEC JTC1/SC29/WG7 MPEG/N0038, 2020. Meeting held online.

[38] MPEG 3DGC. G-PCC codec description v9. International Organization for Standardization Meeting Document ISO/IEC JTC1/SC29/WG7 MPEG/N0011, 2020. Meeting held online.

[39] MPEG 3DGC. V-PCC codec description v12. International Organization for Standardization Meeting Document ISO/IEC JTC1/SC29/WG7 MPEG/N0012, 2020. Meeting held online.

[40] P. Papadakis. The canonically posed 3D objects dataset. In *Eurographics Workshop on 3D Object Retrieval*, pages 33–36, 2014.

[41] 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)*, pages 3428–3432. IEEE, 2020.

[42] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3D object detection from RGB-D data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.

[43] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[44] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017.

[45] 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)*, pages 4320–4324. IEEE, 2019.

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

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

[48] S. Schwarz and M. Personen. Real-time decoding and AR playback of the emerging MPEG video-based point cloud compression standard. *Nokia Technologies; IBC: Helsinki, Finland*, 2019.

[49] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, et al. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018.

[50] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2530–2539, 2018.

[51] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-ResNet and the impact of residual connections on learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[52] Y. Tan, H. Lin, Z. Xiao, S. Ding, and H. Chao. Face recognition from sequential sparse 3D data via deep registration. In *2019 International Conference on Biometrics (ICB)*, pages 1–8. IEEE, 2019.

[53] The Draco authors. Draco 3D Graphics Compression, 2020. https://google.github.io/draco/.

[54] 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*, pages 6411–6420, 2019.

[55] 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)*, pages 3460–3464. IEEE, 2017.

[56] E. M. Torlig, E. Alexiou, T. A. Fonseca, R. L. de Queiroz, and T. Ebrahimi. A novel methodology for quality assessment of voxelized point clouds. In *Applications of Digital Image Processing XLI*, volume 10752, page 107520I. International Society for Optics and Photonics, 2018.

[57] H. L. Turner, J. van Etten, D. Firth, and I. Kosmidis. Modelling rankings in R: The PlackettLuce package. *Computational Statistics*, pages 1–31, 2020.

[58] J. van der Hooft, M. T. Vega, C. Timmerer, A. C. Begen, F. De Turck, and R. Schatz. Objective and subjective QoE evaluation for adaptive point cloud streaming. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2020.

[59] 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)*, pages 1–6. IEEE, 2020.

[60] J. Wang, D. Ding, Z. Li, and Z. Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2021.

[61] 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*, 2021.

[62] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[63] C.-H. Wu, C.-F. Hsu, T.-C. Kuo, C. Griwodz, M. Riegler, G. Morin, and C.-H. Hsu. PCC Arena: A benchmark platform for point cloud compression algorithms.

In *Proceedings of the 12th ACM International Workshop on Immersive Mixed and Virtual Environment Systems*, pages 1–6, 2020.

[64] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[65] W. Yan, Y. Shao, S. Liu, T. H. Li, Z. Li, and G. Li. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691*, 2019.

[66] Y. Yan, Y. Mao, and B. Li. SECOND: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[67] Y. Yang, C. Feng, Y. Shen, and D. Tian. FoldingNet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 206–215, 2018.

[68] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu. Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features. *arXiv preprint arXiv:1904.10014*, 2019.

[69] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019.

[70] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[71] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3D object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.