Technical Report: Quantitative Comparison of Point Cloud Compression Algorithms

Cheng-Hao Wu, Chih-Fan Hsu, Tzu-Kuan Hung, Carsten Griwodz, Wei Tsang Ooi, and Cheng-Hsin Hsu

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 article, 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 provide similar 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.

Index Terms—Benchmark, compression, signal processing, neural networks, quality assessment, subject metrics, objective metrics, Quality-of-Experience, Quality-of-Service

I. INTRODUCTION

With the growing number of Extended Reality (XR) 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 [1]. However, streaming uncompressed dynamic point clouds dictates more than 4 Gbps bandwidth [2], which exceeds the capacity of most commodity access links. Therefore, an 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) [3]–[5] and (ii) Neural Network (NN) [6]–[14] based algorithms. MPEG Geometry-based Point Cloud Compression (G-PCC) [15] and Video-based Point Cloud Compression (V-PCC) [16] 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 [6], [9], [13], [14] 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. In this article, 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 and published in a preliminary workshop paper [17]. Since then, we have extensively extended PCC Arena. We report the extensions and completely new evaluation results in the current article. This article 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.
- 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.

To demonstrate the practicality of PCC Arena, we use it to benchmark seven representative PCC algorithms: three SP-based [3], [15], [16], and four NN-based [6], [9], [13], [14]. 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.

The rest of this article is organized as follows. We survey the literature in Sec. II. We introduce representative PCC algorithms in Sec. III. This is followed by the architecture and implementations of PCC Arena in Sec. IV. Sec. V presents the point cloud datasets and Sec. VI defines the performance metrics. We analyze the evaluation results in Sec. VII. Sec. VIII highlights the pros and cons of NN-based PCC algorithms, and Sec. IX concludes the article.

II. 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 [18], [19] to define the test procedures for several usage scenarios of PCC algorithms. The test procedures specify: (i) predefined coding parameters for G-PCC [15] and V-PCC [16], (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 [20]–[22] evaluated the performance of MPEG G-PCC and V-PCC codecs under diverse settings. In particular, van der Hooft et al. [20] 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. [21] 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. [22] 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 [20]–[22], the current article 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.

Torgil et al. [23] 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 [18], [19]. They, however, did not design a general and extensible benchmark platform, like our PCC Arena, nor make the platform open-source.

2See https://github.com/xtorker/PCC_Arena for our project page.
III. POINT CLOUD COMPRESSION ALGORITHMS

We introduce representative PCC algorithms in this section.

Coding Tools: SP-only Tool NN-only Tool General Tool

Pre-Process

Point Cloud

Coordinates

Normalization

Partition

Attributes

Redundancy Elimination

Transform

Integrated

K-D Tree

Octree

Trisoup

Projection

Quantization

Voxelization

Entropy Code

Arithmetic Code

Entropy Bottleneck

Bitstream

Coordinates

Attributes

Fig. 1. The general encoder architecture of PCC algorithms.

A. General Architecture of PCC Algorithms

Our discussion focuses on the PCC encoder structure, as the decoder structure can be derived from the encoder. Fig. 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. For instance, Convolutional AutoEncoder (Conv. AE) [24] and Convolutional Variational AutoEncoder (Conv. VAE) [25] 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) [26] and prediction [15] 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 [24] has been adopted for removing the redundancy of attributes [10], [11] 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 [27] 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 I.

B. Representative SP-Based PCC Algorithms

Draco. Draco [3] 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 [4]: static point clouds, dynamically acquired point clouds, and point cloud videos. They propose two codecs accordingly: (i) G-PCC [15], which takes advantage of geometric data structures such as octrees for compressing static and dynamic point clouds, and (ii) V-PCC [16], 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 [15], [16].

**C. Representative NN-Based PCC Algorithms**

**GeoCNNv1.** GeoCNNv1 [6] 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 [28] 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 \( \lambda \), which can be used to trade off the extracted feature size (rate) and the focal loss (quality).

**GeoCNNv2.** GeoCNNv2 [9] 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 [29] instead of AE [30]. 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 [10] 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 article.

**PCGCv1.** Similar to GeoCNNv2, VAE is adopted in PCGCv1 [13], which incorporates multiple 3D convolutional blocks and Voxception-ResNet (VRN) [31] for better coding efficiency. VRN combines ResNet [32] and Inception [33] by stacking several ResNet blocks following the Inception-style. ResNet [32] aims to solve the vanishing gradient problem, while Inception [33] 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 \( \lambda \) values are trained.

**PCGCv2.** PCGCv2 [14] 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 sparseness levels of different point clouds and reduce inference time and memory usage. According to their evaluation results [14], PCGCv2 outperforms PCGCv1 in both quality and running time. PCGCv2 only compresses coordinates and employs the same rate control mechanism as PCGCv1.

**IV. 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.

**A. Architecture**

Fig. 2. High-level architecture of the PCC Arena.

Fig. 2 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.

B. 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 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.

\[
\text{Procedure 1 Top-level Evaluation Script}
\]

\begin{verbatim}
1: for alg in all considered algorithms do
2:   Initialize an instance of class alg
3: for rate defined in alg_cfg do
4:   for ds in all considered datasets do
5:     Load ds_cfg
6:     Apply parallel-execution cmd. in alg_cfg
7:     for pc in ds do
8:       Execute en/decoding commands built from alg_cfg, rate, and ds_cfg on pc
9:       Initialize an evaluator eval
10:  for met in all considered metrics do
11:    Calculate met with eval
12:    Save met results in log files
\end{verbatim}

C. 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 must 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. 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, 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.

V. POINT CLOUD DATASETS

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

A. Dataset Selection

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 article\textsuperscript{3}. We select six datasets, as summarized in Table II. Fig. 3 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 [34]. 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 [35]. ModelNet contains 151,128 models across 660 classes. We adopt one of its subsets: ModelNet40 (MN40), which contains 40 classes\textsuperscript{4}.

**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 [36]. 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 [19] 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.

### B. Dataset Generation

Most selected datasets (except 8iC/8i) are 3D mesh datasets. We employ CloudCompare [40] 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 $nx$, $ny$, $nz$, respectively. 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 [2].

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. IV-C.

### VI. 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.

--

\textsuperscript{3}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.

\textsuperscript{4}We skip three point clouds in MN40, because they are 2D instead of 3D models.
A. Non-visual Metrics

We first define the non-visual metrics, which are crucial to understanding the efficiency of the PCC algorithms. Let \( P_r \) and \( P_t \) 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 article. 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., \( |P_r| \)). The running time refers to the sum of the encoding and decoding time. The encoding and decoding times are also recorded individually.

B. 3D Visual Metrics: Coordinates

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

- **Asymmetric Chamfer Distance from \( P_1 \) to \( P_2 \) (ACD):**
  \[
  \text{ACD}(P_1, P_2) = \frac{1}{|P_1|} \sum_{p \in P_1} \min_{p' \in P_2} ||p - p'||^2_2, \tag{1}
  \]
  which is a directional metric from point cloud \( P_1 \) to point cloud \( P_2 \). Based on this, we define ACD\(_r\) = ACD\((P_r, P_t)\) and ACD\(_t\) = ACD\((P_t, P_r)\).

- **Chamfer Distance (CD):**
  \[
  \text{CD} = \frac{1}{2} (\text{ACD}_r + \text{ACD}_t), \tag{2}
  \]
  which is a bidirectional metric, defined based on ACD.

- **CD Peak Signal-to-Noise Ratio (CD-PSNR):**
  \[
  \text{CD-PSNR} = 10 \log_{10} \frac{M^2_{r,t}}{\text{CD}}, \tag{3}
  \]
  where \( M_r \) is the maximal distance between any two points in \( P_r \) [41].

- **Hausdorff Distance (HD):**
  \[
  \text{HD} = \max \left( \max \left( \min_{p \in P_r} ||p_r - p_t||^2_2 \right), \right. \\
  \left. \max \left( \min_{p_t \in P_t} ||p_t - p_r||^2_2 \right) \right), \tag{4}
  \]
  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\(_r\) and ACD\(_t\), and (ii) ACD\(_r\) in the literature [41], [42]. Each of these definitions comes with a different intuition. The maximum of ACD\(_r\) and ACD\(_t\) finds the largest distance between both directions. ACD\(_r\) 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 article. 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 (Sec. VII-B). CD-PSNR is in the dB scale to better capture the visual quality [42].

The above quality metrics are defined with the point-to-point distance, i.e., \( ||p - p'||_2 \). We also generalize the quality metrics with the point-to-plane distance [42]:

\[
(p - p') \cdot N_p, \tag{5}
\]
where \( N_p \) is the normal vector of the plane of \( P_t \) 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: ACD\(_r\), ACD\(_t\), CD\(_r\), CD-PSNR\(_r\), and HD\(_r\).

C. 3D Visual Metrics: With Colors

We also define several colored metrics.

- **Luminance Color PSNR (L-CPSNR):**
  \[
  \text{L-CPSNR} = 10 \log_{10} \frac{M^2}{\text{L-MSE}(P_r, P_t)}, \tag{6}
  \]
  where \( M \) is the maximum value of the luminance. In ITU-R BT.709 [43], \( M \) 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):**
  \[
  \text{VQoE} = \alpha \cdot \text{CD} + (1 - \alpha) \cdot H^Y_{L_2}, \tag{7}
  \]
  where \( \alpha \) is the weight to combine the coordinate and color components: CD and \( H^Y_{L_2} \). Here, \( H^Y_{L_2} \) is the L2 distance of histograms of luminance values between \( P_r \) and \( P_t \). Yes. That is true. We adopt the empirically chosen \( \alpha \) value of 0.6597 [44] as the default value in PCC Arena.

D. 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 [45] 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) [46], 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.
VII. 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 II); 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.

A. Objective Results

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. 4 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. Among all quality metrics, the higher value of CD-PSNR/CD-PSNRp, the better quality is, while the lower value of the ACD-related metrics and HD/HDp, the better quality is. While the p2pt and p2pl values of ACDrt are comparable, those values of ACDrt occasionally show significant differences. For example, w.r.t. ACDrt, 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 ACDrt, the trend of the p2pt and p2pl quality metrics is quite similar. Because the p2pl quality metrics are more relevant to the visual quality [42], we only report the p2pl quality metrics in the rest of the section for brevity.

2) Coding Efficiency with only Coordinates: We plot sample Rate-Distortion (R-D) curves from the CAPOD dataset in Fig. 5. 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. Fig. 5(a) reveals that GeoCNNv2 outperforms other PCC algorithms in ACDrtp, which implies that the output point clouds from GeoCNNv2 are the closest one to the input point clouds, compared to other PCC algorithms. Fig. 5(b) shows that GeoCNNv2 and PCGCv1 suffer from unstable coding efficiency w.r.t. ACDtrp 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. 6. 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 ACDtrp. In all the results from the CAPOD dataset, we found that 24 point clouds (≈ 0.56%) are affected.

The same approximation approach is adopted throughout this section.
Fig. 5. R-D curves of different PCC algorithms. Sample results from the CAPOD dataset are shown with metrics: (a) ACD<sub>rtp</sub>, (b) ACD<sub>trp</sub>, and (c) CD<sub>p</sub>.

Fig. 6. 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.

Fig. 7. R-D curves of different PCC algorithms. Sample results from the CAPOD dataset are shown with metrics: (a) CD-PSNR<sub>p</sub> and (b) HD<sub>p</sub>.

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. 5(c) plots the R-D curves in CD<sub>p</sub>, which demonstrates consistent trends as ACD<sub>rtp</sub> in Fig. 5(a).

We also plot the R-D curves of CD-PSNR<sub>p</sub> and HD<sub>p</sub> in Fig. 7. Fig. 7(a) reports CD-PSNR<sub>p</sub>, which represents the average case, while Fig. 7(b) shows HD<sub>p</sub>, 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<sub>p</sub>, as shown in Fig. 7(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<sub>p</sub> values. The SP-based V-PCC also leads to higher HD<sub>p</sub> 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 similar SP-based PCC algorithms. Last, G-PCC outperforms other PCC algorithms in HD<sub>p</sub> 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. 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.

3) Coding Efficiency with Colors: We report sample R-D curves from the SNCC dataset in Fig. 9. We note that only three PCC algorithms can handle colored point clouds. As shown in Fig. 9(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, Y-CPNSNR improvement on the bitrate. Such outcome is
because we only adopt the rate control on the coordinates in G-PCC and the rate control mechanism in V-PCC, 2D image QP, is not notable. Fig. 9(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 Y-CPNSR and VQoE.

4) Running Time of PCC Algorithms: 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. 10. 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 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. 11 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. 10 and 11.

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. 12. We make the following observations. First, V-PCC and GeoCNNv2 outperform other PCC algorithms in terms of Y-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. Sample results at 0.5 bpp are shown with 2D visual quality metrics: (a) Y-PSNR and (b) SSIM.

B. Subjective Results

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 [45]. 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. 14 shows a screenshot of our Web page built on Qualtrics [47]. 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
We found that the average deviation is 0.203 bpp, and the worst-case deviation is 0.496 bpp. For PCC algorithms that cannot precisely achieve 0.5 bpp, we used the output point cloud closest to 0.5 bpp. We encoded each point cloud at 0.5 bpp; for PCC algorithms that cannot precisely achieve 0.5 bpp, we refer to the former algorithm as the winner and the latter as the loser. Fig. 15(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 [48]. However, such method requires additional effort to integrate all pairwise results and thus is rather indirect. Therefore, we employ the Plackett-Luce model [49] 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. 15(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 their rate control logic. The trend of the scores in this figure is consistent with pairwise results in Fig. 15(a). Last, we present the overall image quality scores across all considered objects in Fig. 15(c). This figure reveals that the

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. 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 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 was filtered out.

2) Coordinate-only Objects: 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 outperforms another PCC algorithm. We refer to the former algorithm as the winner and the latter as the loser. Fig. 15(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 [48]. However, such method requires additional effort to integrate all pairwise results and thus is rather indirect. Therefore, we employ the Plackett-Luce model [49] 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. 15(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 their rate control logic. The trend of the scores in this figure is consistent with pairwise results in Fig. 15(a). Last, we present the overall image quality scores across all considered objects in Fig. 15(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. 16. We notice that the ranks and scores are very similar to those in image quality reported in Fig. 15. 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.

3) Coordinate-only Avatars: Applying the same analysis methodology, we report the subjective image quality results in Fig. 17 and the subjective point cloud similarity results in Fig. 18, respectively. Compared 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. 17(a) and 18(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. 17(b) and 18(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.

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. 19. 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.

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 III. 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 on these artifacts, human eyes are very good at detecting 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.

VIII. FUTURE OF THE NN-BASED PCC ALGORITHMS

In this article, 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

![Fig. 19. Pairwise comparison matrices for colored objects (a), (b) and avatars (c), (d) in image quality (a), (c) and point cloud similarity (b), (d).](image)

**TABLE III**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V-PSNR (dB)</td>
<td>0.19</td>
<td>0.21</td>
<td>2.9</td>
<td>1.9</td>
<td>1.6</td>
<td>3.8</td>
<td>3.8</td>
<td>2.0</td>
<td>1.3</td>
<td>1.4</td>
</tr>
<tr>
<td>SSIM</td>
<td>0.07</td>
<td>0.07</td>
<td>6.0</td>
<td>5.9</td>
<td>6.7</td>
<td>8.5</td>
<td>8.5</td>
<td>5.6</td>
<td>4.5</td>
<td>4.7</td>
</tr>
<tr>
<td>AC$^d$</td>
<td>-0.16</td>
<td>-0.02</td>
<td>5.1</td>
<td>4.3</td>
<td>4.7</td>
<td>8.6</td>
<td>8.6</td>
<td>5.6</td>
<td>4.2</td>
<td>4.8</td>
</tr>
<tr>
<td>AC$^d$</td>
<td>0.03</td>
<td>0.13</td>
<td>5.1</td>
<td>4.3</td>
<td>4.7</td>
<td>8.6</td>
<td>8.6</td>
<td>5.6</td>
<td>4.2</td>
<td>4.8</td>
</tr>
<tr>
<td>CD$^p$</td>
<td>-0.11</td>
<td>0.23</td>
<td>5.1</td>
<td>4.3</td>
<td>4.7</td>
<td>8.6</td>
<td>8.6</td>
<td>5.6</td>
<td>4.2</td>
<td>4.8</td>
</tr>
<tr>
<td>CD-PSNR (dB)</td>
<td>0.38</td>
<td>0.38</td>
<td>5.1</td>
<td>4.3</td>
<td>4.7</td>
<td>8.6</td>
<td>8.6</td>
<td>5.6</td>
<td>4.2</td>
<td>4.8</td>
</tr>
<tr>
<td>HDP</td>
<td></td>
<td>0.19</td>
<td>2.9</td>
<td>1.9</td>
<td>1.6</td>
<td>3.8</td>
<td>3.8</td>
<td>2.0</td>
<td>1.3</td>
<td>1.4</td>
</tr>
<tr>
<td>L-CPSNR (dB)</td>
<td>0.03</td>
<td>0.13</td>
<td>5.1</td>
<td>4.3</td>
<td>4.7</td>
<td>8.6</td>
<td>8.6</td>
<td>5.6</td>
<td>4.2</td>
<td>4.8</td>
</tr>
<tr>
<td>VQoE</td>
<td></td>
<td>0.09</td>
<td>0.07</td>
<td>6.0</td>
<td>5.9</td>
<td>6.7</td>
<td>8.5</td>
<td>8.5</td>
<td>5.6</td>
<td>4.7</td>
</tr>
</tbody>
</table>

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

![Fig. 19. Pairwise comparison matrices for colored objects (a), (b) and avatars (c), (d) in image quality (a), (c) and point cloud similarity (b), (d).](image)
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 [10], [11] on this topic with some preliminary results. We believe there exists a large room to explore in this direction.

**IX. Conclusion**

In this article, 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.

**References**


