PointPCA2-RS: PointPCA2 with Resource Saving

Arthur Henrique Silva Carvalho and Pedro Garcia Freitas

***Abstract*— This paper introduces PointPCA2-RS, a lightweight version of PointPCA2 with emphasis on low computational resources usage. PointPCA2-RS optimizes the original PointPCA2 by re-implementing it using a different programming language, data structures, and specific algorithmic details. Designed to evaluate visual quality, both projects extract identical features from point cloud (PC) data. Their core functionalities are similar, including Principal Component Analysis (PCA) on local geometric data, computation of spatial and geometric descriptors, and feature pooling and aggregation. These attributes are achieved through a programming language change, which positively impacts architecture and performance. PointPCA2-RS adopts a modular approach optimized for performance and maintainability, being more suitable for large-scale or real-time applications. Experimental results demonstrate the high performance of PointPCA2-RS against PointPCA2 without sacrificing prediction accuracy. PointPCA2-RS outperforms state-of-the-art point cloud quality assessment (PCQA) metrics, offering significant improvements for PCQA field. The code of PointPCA2-RS metric is available at `https://github.com/arthurhscarvalho/pointpca2-rs`.**

***Keywords*— Point cloud, quality assessment, resource saving**

I. Introduction

Point clouds are 3D signals and the primary modality for representing captured real-world scenes. Applications range from autonomous driving to virtual and augmented reality. As a result, the need to obtain methods capable of efficiently assessing the quality of these PCs becomes increasingly critical. Nevertheless, their intrinsic complexity and data variability impose considerable challenges in ensuring PC visual quality.

There are two main approaches to assess the PC quality: subjective and objective. Subjective quality assessment involves evaluating the perceived visual quality based on human judgment. It requires tiresome experiments for the evaluation of the visual stimuli and produces a scalar score that enables validation of how well objective metrics correlate with human perception. Although subjective methods are the baseline for understanding how humans perceive the quality of point clouds based on the visual stimuli they receive, these methods are costly, time-consuming, and labor-intensive, making them unsuitable for many applications. Objective approaches, on the other hand, provide a more efficient and scalable way to evaluate PCs, complementing subjective assessments. These approaches use computational algorithms, known as 'quality metrics', that generate numerical scores. These scores must correlate well with subjective assessments, ensuring they reflect human perception.

Arthur Henrique Silva Carvalho and Pedro Garcia Freitas are with Department of Computer Science of University of Brasília (UnB), e-mail: henrique-arthur.ah@aluno.unb.br, pedro.freitas@unb.br; This work was supported by the Federal District Research Support Foundation (FAP-DF), the Coordination for the Improvement of Higher Education Personnel (CAPES), and by the National Council for Scientific and Technological Development (CNPq).

Due to the importance of objectively assessing the quality of PCs, a myriad of objective metrics have been proposed in recent years. Existing objective quality metrics are modeled using very assorted strategies. Metrics based on point-to-point (P2P) [1], point-to-plane (P2Pl) [2], plane-to-plane (Pl2Pl) [3], and point-to-distribution (P2D) [4] distances primarily focus on measuring geometric fidelity and occasionally ignore the perceptual nuances observed by humans, failing to capture local structures that are essential to determine the perceptual quality of PCs. Although recent research has explored perceptual-based metrics [5], [6], [7], many of these approaches remain limited in their ability to effectively capture the complex structural variations in PCs while maintaining low computational resource consumption [8].

While fidelity metrics (i.e., P2P, P2Pl, etc.) are easy to implement and require low computational resources, they generally show little correlation with subjective data. On the other hand, perceptual metrics exhibit high correlation with perceptual data at the cost of high computational resource consumption. In this scenario, descriptor-based methods offer the best trade-off between efficiently predicting subjective scores and maintaining computational efficiency [9], [10]. Using the intrinsic connectivity of PCs, descriptor-based methods can represent both local and global structures, as well as incorporate texture information and other visual attributes that may influence human perception of quality. For instance, local binary patterns (LBP) [11], local luminance patterns (LLP) [12], and their variants [13], [14], [15] have demonstrated the potential of these descriptors to extract distortion-aware features to model efficient quality metrics with low computational consumption. More recently, the introduction of PCA, as seen in PointPCA [16] and PointPCA2 [17], has shown promise in representing PC quality through learning-based techniques.

PointPCA [16] and PointPCA2 [17] are currently the state-of-the-art (SOTA). PointPCA employs geometric and textural descriptors computed per point to extract quality-aware features. Inspired on it, PointPCA2 performs PCA only on the geometry data while enriching existing geometry and texture descriptors, that are computed more efficiently and with less computational resources. Similarly to PointPCA, PointPCA2 predicts the quality score through a learning-based fusion of individual predictions from geometry and texture descriptors that capture local shape and appearance properties, respectively. In this paper, PointPCA-RS leverages local structure analysis for PCQA. Our method focuses on extracting local descriptors using fewer system resources, achieving substantially faster performance with reduced memory consumption.

The rest of this paper is organized as follows. In Section II, we present the proposed method. Section III describes the experimental results. Finally, Section IV concludes the paper.

II. METHODOLOGY

A PC is a set of 3D points, each having spatial coordinates and color information. Formally, a PC of n points can be described as a set $\mathcal{P}$=$\{(p_i, c_i) \mid i = 1, 2, \cdots, n\}$, where each point p_i=(x_i, y_i, z_i) is associated with a color attribute c_i=(r_i, g_i, b_i). $\mathcal{A}$ is defined as the reference point cloud (RPC), whose K nearest points correspond to the distorted point cloud (DPC) $\mathcal{B}$. Given a query point p_i of a RPC $\mathcal{A}$, the coordinates of the k nearest neighbors (NNs) of p_i are indicated as $p_{\mathcal{K}}^{g,\mathcal{F}}$=$(x_{\mathcal{K}}, y_{\mathcal{K}}, z_{\mathcal{K}})^T$ with $1 \leq \mathcal{K} \leq k$ and $\mathcal{F} \in \{\mathcal{A}, \mathcal{B}\}$. The geometric representation of p_i is expressed as $p_i^{g,\mathcal{A}}$, while the geometry of its NN in $\mathcal{B}$ is represented as $p_{\mathcal{K}}^{g,\mathcal{B}}$. At first, the covariance matrix $\Sigma_i^{\mathcal{A}}$ is computed as

$$\Sigma_i \mathcal{A} = \frac{1}{k} \sum_{j=1}^{k} \left(p_j^{g,\mathcal{A}} - \bar{p}_i^{g,\mathcal{A}}\right) \cdot \left(p_j^{g,\mathcal{A}} - \bar{p}_i^{g,\mathcal{A}}\right)^{\top}, \quad (1)$$

where $\bar{p}_i^{g,\mathcal{A}}$=$\frac{1}{k} \sum_{j=1}^{k} p_j^{g,\mathcal{A}}$ is the centroid of the sampled nearest points. Next, eigen-decomposition is performed on $\Sigma_i^{\mathcal{A}}$ to derive the eigenvectors, which constitute an orthonormal basis $V^{\mathcal{A}}$ made up of eigenvectors $v_m^{\mathcal{A}}$ with corresponding eigenvalues $\lambda_m^{\mathcal{A}}$, where $\lambda_1^{\mathcal{A}}$>$\lambda_2^{\mathcal{A}}$>$\lambda_3^{\mathcal{A}}$. Subsequently, RPC and DPC neighborhoods are projected onto the new orthonormal basis, $\omega_j^{\mathcal{F}}$=$\left(p_j^{g,\mathcal{F}} - \bar{p}_i^{g,\mathcal{A}}\right) \cdot V^{\mathcal{A}}$. PCA is applied to the covariance matrix of $\omega_j^{\mathcal{B}}$ to derive the eigenvectors $v_m^{\mathcal{B}}$ and eigenvalues $\lambda_m^{\mathcal{B}}$. The mapped coordinates $\omega_j^{\mathcal{F}}$, eigenvectors $v_m^{\mathcal{F}}$, and unit vector u_m^T are then used to form the descriptors in Table I.

Similar to [18], multiple distances are computed based on the descriptors in Table I. The Euclidean distance d_1 is employed to quantify the pairwise distances between query points in the transformed space to generate the $r_\alpha = \sqrt{\Sigma_m d_1^2}$ predictor. The point-to-plane distance, denoted as $r_\beta = |d_2|$, is measured using the absolute value, where d_2 represents the distance from a point to the reference axes after projection. The relative difference for covariance features is defined as:

$$r_\gamma = \frac{|q^{\mathcal{A}} \odot q^{\mathcal{B}} - Q|}{q^{\mathcal{A}} \odot q^{\mathcal{B}}}, \quad (2)$$

where $\{q^{\mathcal{F}}$=$\lambda^{\mathcal{F}}, Q$=$\Sigma\}$ and $\{q^{\mathcal{F}}$=$\tilde{s}^{\mathcal{F}}, Q$=$\tilde{\Sigma}\}$ for geometry and texture attributes, respectively. For the remaining descriptors, the relative difference formula from [19] is applied:

$$r_\delta = \frac{|\phi^{\mathcal{A}} - \phi^{\mathcal{B}}|}{|\phi^{\mathcal{A}}| + |\phi^{\mathcal{B}}| + \varepsilon}, \quad (3)$$

where ε is a small constant to avoid division by zero. Notationally, distances r_ρ and r_θ correspond to P_m and θ_m, respectively. Features are aggregated predictors. The predictors $\psi_{i,j,k}$ re computed for each point p_i, each descriptor j, and each distance function r_k ($k \in \{\alpha, \beta, \gamma, \delta, \rho, \theta\}$). This process is applied to all descriptors j in Table I with their associated distance functions r_k, pooling them to derive one feature $f_{j,k}$:

$$f_{j,i} = \frac{1}{|\mathcal{A}|} \sum_{i=1}^{\mathcal{A}} \psi_{i,j,k}. \quad (4)$$

The PointPCA2 algorithm was fully re-implemented in Rust [20] to generate these descriptors. Rust's static typing and ownership-based memory management ensure robustness and predictable performance. These features eliminate garbage collection overhead and reduce memory requirements through sophisticated memory modeling strategies. Named PointPCA2-RS, our implementation reconceptualizes sequential processes, optimizing computational efficiency through parallelism and data structure modeling while preserving the fundamental algorithmic principles. PointPCA2-RS parallelizes PointPCA2's sequential processing, particularly for the computationally intensive feature extraction.

TABLE I: DEFINITION OF DESCRIPTORS.

	Descriptor	Definition	D
Geometric	Error vector	$e = (\omega_i^B - \omega_i^A)$	r_α
	Error along axes	$\epsilon_m = (\omega_i^B - \omega_i^A)^T \cdot u_m$	r_β
	Error from origin	$\epsilon = \omega_i^{\mathcal{F}}$	r_α, r_β
	Mean	$\mu^{\mathcal{B}} = \frac{1}{n} \sum_j \omega_j^{\mathcal{B}}$	r_α, r_β
	Variance	$\lambda_m^{\mathcal{F}} = \frac{1}{n} \sum_j (\omega_j^{\mathcal{F}} - \mu^{\mathcal{F}})^2$	r_δ
	Sum of variance	$\Sigma^{\mathcal{F}} = \sum_m \lambda_m^{\mathcal{F}}$	r_δ
	Covariance	$\Sigma = \frac{1}{n} \sum_j (\omega_j^A - \mu^A) \cdot (\omega_j^B - \mu^B)^T$	r_γ
	Omnivariance	$\mathcal{O}^{\mathcal{F}} = \sqrt[3]{\prod_m \lambda_m^{\mathcal{F}}}$	r_δ
	Eigenentropy	$\mathcal{E}^{\mathcal{F}} = -\sum_m \lambda_m^{\mathcal{F}} \cdot \log \lambda_m^{\mathcal{F}}$	r_δ
	Anisotropy	$\mathcal{A}^{\mathcal{F}} = (\lambda_1^{\mathcal{F}} - \lambda_3^{\mathcal{F}})/\lambda_1^{\mathcal{F}}$	r_δ
	Planarity	$\mathcal{P}^{\mathcal{F}} = (\lambda_2^{\mathcal{F}} - \lambda_3^{\mathcal{F}})/\lambda_1^{\mathcal{F}}$	r_δ
	Linearity	$\mathcal{L}^{\mathcal{F}} = (\lambda_1^{\mathcal{F}} - \lambda_2^{\mathcal{F}})/\lambda_1^{\mathcal{F}}$	r_δ
	Scattering	$\mathcal{S}^{\mathcal{F}} = \lambda_3^{\mathcal{F}}/\lambda_1^{\mathcal{F}}$	r_δ
	Curvature	$\mathcal{C}^{\mathcal{F}} = \lambda_3^{\mathcal{F}} / \sum_m \lambda_m^{\mathcal{F}}$	r_δ
	Parallelity	$p_m = 1 - u_m \cdot v_m^B$	–
	Angular similarity	$\theta_m = 1 - 2 \cdot \arccos(\cos(u_m, v_m^B))/\pi$	–
Textural	Mean	$\tilde{\mu}^{\mathcal{F}} = \frac{1}{n} \sum_j c_{n,t}^{\mathcal{F}}$	r_δ
	Variance	$\tilde{g}^{\mathcal{F}} = \frac{1}{n} \sum_j (c_{n,t}^{\mathcal{F}} - \tilde{\mu}^{\mathcal{F}})^2$	r_δ
	Sum of variance	$\tilde{\Sigma}^{\mathcal{F}} = \sum_m \tilde{g}_m^{\mathcal{F}}$	r_δ
	Covariance	$\tilde{\Sigma} = \frac{1}{n} \sum_j (c_{n,t}^A - \tilde{\mu}^A) \cdot (c_{n,t}^B - \tilde{\mu}^B)^T$	r_γ
	Omnivariance	$\tilde{\mathcal{O}}^{\mathcal{F}} = \sqrt[3]{\prod_m \tilde{g}_m^{\mathcal{F}}}$	r_δ
	Entropy	$\tilde{\mathcal{H}}^{\mathcal{F}} = -\sum_m \tilde{g}_m^{\mathcal{F}} \cdot \log \tilde{g}_m^{\mathcal{F}}$	r_δ

Both PointPCA2 and PointPCA2-RS comprise four major stages, namely preprocessing, feature extraction, descriptor computation, and regression. The preprocessing stage is composed of duplicate merging and neighborhood identification steps. Duplicate merging, as the name suggests, merges duplicate points by averaging the multiple colors associated with each point. While PointPCA2 employs a naive brute force method where it checks every point against all others to identify duplicates, PointPCA2-RS utilizes a `BTreeMap` [21] to map each point to a vector that holds all of its associated colors (including duplicated).

Most of the computational overhead occurs during the feature extraction stage. Much of this overhead is due to the intensive use of NN searches that must be performed. For each point in the RPC, two neighborhood sets are identified: the k nearest neighbors (KNNs) within reference itself, and the KNNs within its distorted version under assessment. This procedure establishes critical correspondence between local geometric regions in the RPC and DPC.

PointPCA2 identifies local neighborhoods around each point in the RPC, accomplished via Matlab's `knnsearch` function. KNN search is inherently computationally intensive, but PointPCA2's strategy further intensifies these demands by precomputing all neighborhoods sequentially and storing two $n \times k$ tensors in memory (where n is the number of points). In PointPCA2, k is typically set to 81. Consequently, for a RPC

with 1M points, the neighborhood arrays would each contain 81M elements, leading to 162M elements allocated in memory prior to feature computation.

PointPCA2-RS improves computational efficiency by adopting Rust's `kd_tree`. This references original point arrays without explicit neighborhood storage, reducing additional memory overhead from $O(nk)$ to $O(n)$. For the same aforementioned example, the 1M points require only 2M memory elements. Moreover, the employed PointPCA2-RS construction process is parallelized, significantly reducing the computation time required for this step. Next, in parallel, 42 local features are computed per neighborhood by iterating through each RPC point to compute the covariance matrix described in Equation 1. Next, PCA is computed over the covariance matrix for both RPC and DPC to determine eigenvectors and establish a local orthonormal basis. From these bases, features such as the mean, variance, eigenvectors, and covariance of both geometric and texture data are then computed.

Since each iteration operates independently of the others, PointPCA2-RS executes iterations in parallel to compute the feature matrix of 42×n dimensions. Each iteration extracts neighborhoods from the pre-constructed `kd_trees` on demand, computes features, stores the results in the feature matrix, and then releases memory. As RPC typically contain millions or billions of points, this process is highly effective.

Finally, in the descriptor computation stage, the feature matrix is processed to produce the descriptors as depicted in Equation 4. In this stage, the pooling operations convert the 42 features into a set of 40 descriptors. To perform this conversion, PointPCA2 maintains in memory a features matrix of dimensions 42×1M and simultaneously a descriptors matrix of dimensions 40×1M for 1M points, incurring huge memory usage. To address this issue, PointPCA2-RS avoids storing the entire descriptors matrix by computing pooled predictors directly during spatial metric calculation. As each column of metrics is computed, it is immediately average-pooled into the 40×1 output vector, significantly reducing memory usage.

III. Experimental Results

The experiments were all performed on an AMD Ryzen™ Threadripper™ 2950X with 128GB RAM. We conducted experiments on the four PC quality benchmark datasets to investigate the performance of the proposed method. These datasets and their attributes are depicted in Table II. To validate the effectiveness and reliability of the proposed PointPCA2-RS, we compare it with the original in terms of feature similarity, spent time, allocated memory, and correlations with Mean Opinion Score (MOS).

The Kolmogorov–Smirnov test (KST) [22] was used to assess whether features from PointPCA2 and the proposed PointPCA2-RS come from the same distribution. For each of the 40 features generated by the compared methods, a p-value of 0.05 as the significance level was used to determine whether the distributions were similar enough. Table III depicts the KST results. From this table, we can observe that the feature produces by PointPCA2-RS is statistically comparable with PointPCA2 for the most of features in D1 (37 out of 40), D2 (27 out of 40), D3 (36 out of 40), and D4 (40 out of 40).

TABLE II: Used Point clouds quality benchmark databases

DB	REF	Mnemonic	Attributes	Distortion	SRC	PPC
D1	[23]	M-PCCD (APSIPA)	Color	G-PCC, V-PCC	8	254
D2	[24]	BASICS (ICIP2023)	Color, Normals	PCC, GPCC-RAHT, GPCC-Predlift and GEOCNN	75	1494
D3	[25]	SJTU-PCQA	Color	Octree, downsampling, color and geometry noise	9	378
D4	[26]	UnB_PC	Color, Normals	Octree, JPEG compression	6	60

The comparative temporal analysis was conducted by measuring the time required to extract features for each pair of RPC and DPC across all datasets. The top row of the Figure 1 shows these times plotted against the total number of processed points per pair (i.e., the sum of points in the reference and evaluated PCs). The blue points correspond to the times obtained using PointPCA2, while the orange points represent the times obtained when processing the same pairs with PointPCA2-RS. The curves in these plots represent the linear regression lines that best fit the data points. Based on these curves, it is clear that the proposed method is significantly more time-efficient than PointPCA2, since its curve consistently remains asymptotically under that of PointPCA2. To better quantify the performance gain from using PointPCA2-RS instead of PointPCA2, we computed the relative speedup as the following ratio:

$$Speedup = \frac{\text{Execution Time of PointPCA2}}{\text{Execution Time of PointPCA2-RS}}.$$

The speedup indicates how many times faster PointPCA2-RS is compared to the baseline PointPCA2. For instance, a speedup of 2 means the PointPCA2-RS takes half the time to extract the features of the same pair of PCs. Since we measured the speedup per dataset entry, we created a *regplot* to depict how PointPCA2-RS is faster than PointPCA2, as shown in the second row of Figure 1.

Similar to the time performance analysis, we calculated the memory efficiency ratio by dividing the memory used by PointPCA2 by the memory used by PointPCA2-RS. A higher ratio indicates that PointPCA2 consumes more memory than PointPCA2-RS for the same number of points, while ratio values below 1 means PointPCA2 is more memory-efficient. Figure 2 depicts the points and the curves of the compared methods. Similar to the analyses performed with the time data shown in Figure 1, Figure 2 presents the amount of memory consumed (in gigabytes) as a function of the total number of points in the PCs considered. Figure 2 also shows the memory saved by using PointPCA2-RS instead of PointPCA2.

Table IV consolidates the results presented in Figures 1 and 2, reporting the average time and memory consumption, along with the corresponding speedups and relative memory savings. From this table, the advantage of the proposed method over the state of the art is clear. In terms of processing time, the proposed approach achieves speedups of 26.69× on D1 (2,569% faster), 26.86× on D2 (2,586% faster), 19.72× on D3 (1,872.0% faster), and 29.05× on D4 (2,805.0% faster). As regards memory savings, the usage of PointPCA2-RS corresponded to a memory reduction of 8.03× on D1, 20.96× on D2, 33.45× on D3, and 6.88× on D4.

TABLE III: KOLMOGOROV–SMIRNOV TEST TO CHECK WHETHER POINTPCA2 AND POINTPCA2-RS COME FROM THE SAME DISTRIBUTION.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
D1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
D2	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗
D3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✗
D4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

TABLE IV: TIME AND MEMORY USAGE OF POINTPCA2 AND POINTPCA2-RS, INCLUDING SPEEDUP AND RELATIVE MEMORY SAVINGS

Attribute	Method	D1	D2	D3	D4
Time	PointPCA2	132.33 ± 35.67	236.70 ± 164.65	135.09 ± 59.23	99.32 ± 9.44
	PointPCA2-RS	4.97 ± 1.20	9.69 ± 11.09	7.33 ± 5.61	4.29 ± 6.61
	Speedup	26.69 ± 3.62	26.86 ± 4.32	19.72 ± 2.69	29.05 ± 5.49
Memory	PointPCA2	2.40 ± 0.66	4.59 ± 3.18	2.42 ± 0.99	1.90 ± 0.18
	PointPCA2-RS	0.36 ± 0.15	0.69 ± 0.66	0.28 ± 0.20	0.27 ± 0.05
	Memory Saving	8.03 ± 10.60	20.96 ± 43.63	33.45 ± 53.49	6.88 ± 0.84

TABLE V: SROCC PERFORMANCE ON M-PCCD [23], SJTU-PCQA [25] AND WPC [27] DATASETS.

Dataset	PointPCA2/RS	PointPCA1	PCQM	pSSIM	BitDance	Plane2plane	P2Plane MSE	P2P MSE	PSNR Y
M-PCCD	**0.960**	0.941	0.940	0.925	0.859	0.847	0.901	0.896	0.798
SJTU-PCQA	**0.980**	0.890	0.862	0.708	0.748	0.761	0.578	0.612	0.743
WPC	**0.910**	0.866	0.749	0.465	0.451	0.454	0.452	0.563	0.614

Finally, Table V compares the PointPCA2-RS/PointPCA2 with current state-of-the-art Full-reference (FR) PCQA methods using the M-PCCD, SJTU-PCQA and WPC datasets. The performance achieving the highest score for these metrics is shown in boldface. In other words, based on Table V and the previous results depicted in Table IV, it is patent that the proposed PointPCA2-RS achieves the best results in terms of correlation while maintaining a lightweight algorithm.

IV. CONCLUSIONS

This paper uses a descriptor-based approach to extract features for supervised-learning-based PCQA metrics. The proposed method produces a feature set comparable to SOTA PointPCA2, achieving equivalent correlation performance with significantly fewer computational resources. Future work will incorporate additional saliency-based features to enhance algorithm performance and accuracy.

REFERENCES

[1] Alireza Javaheri, Catarina Brites, Fernando Pereira, and João Ascenso. Improving psnr-based quality metrics performance for point cloud geometry. In *2020 IEEE international conference on image processing (ICIP)*, pages 3438–3442. IEEE, 2020.

[2] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE international conference on image processing (ICIP)*, pages 3460–3464. IEEE, 2017.

[3] Evangelos Alexiou, Touradj Ebrahimi, Marco V Bernardo, Manuela Pereira, Antonio Pinheiro, Luis A Da Silva Cruz, Carlos Duarte, Lovorka Gotal Dmitrovic, Emil Dumic, Dragan Matkovics, et al. Point cloud subjective evaluation methodology based on 2d rendering. In *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2018.

[4] Alireza Javaheri, Catarina Brites, Fernando Pereira, and João Ascenso. Mahalanobis based point to distribution metric for point cloud geometry quality evaluation. *Signal Processing Letters*, 27:1350–1354, 2020.

[5] Pedro Garcia Freitas, Mateus Gonçalves, Johann Homonnai, Rafael Diniz, and Mylène CQ Farias. On the performance of temporal pooling methods for quality assessment of dynamic point clouds. In *2022 14th international conference on quality of Multimedia experience (QoMEX)*, pages 1–6. IEEE, 2022.

[6] Pedro Garcia Freitas, Giovani Decido Lucafo, Mateus Gonçalves, Johann Homonnai, Rafael Diniz, and Mylène CQ Farias. Comparative evaluation of temporal pooling methods for no-reference quality assessment of dynamic point clouds. In *Proceedings of the 1st Workshop on Photo-realistic Image and Environment Synthesis for Multimedia Experiments*, pages 35–41, 2022.

[7] Pedro Garcia Freitas, Rafael Diniz, and Mylene CQ Farias. Point cloud quality assessment: unifying projection, geometry, and texture similarity. *The Visual Computer*, 39(5):1907–1914, 2023.

[8] Arthur HS Carvalho, Pedro G Freitas, Mateus Gonçalves, Johann Homonnai, and Mylène CQ Farias. Perception-driven point cloud quality assessment through projections and deep structure similarity. In *2024 IEEE 26th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2024.

[9] Pedro Garcia Freitas, Welington Y.L. Akamine, and Mylène C.Q. Farias. No-reference image quality assessment based on statistics of local ternary pattern. In *2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2016.

[10] Pedro Garcia Freitas, Luísa Peixoto da Eira, Samuel Soares Santos, and Mylène Christine Queiroz de Farias. A referenceless image quality assessment based on bsif, clbp, lpq, and LCP texture descriptors. In *Image Quality and System Performance XVI, Electronic Imaging 2019, IQSP, Burlingame, CA, USA, 13-17 January 2019*. Society for Imaging Science and Technology, 2019.

[11] Rafael Diniz, Pedro Garcia Freitas, and Mylène CQ Farias. Towards a point cloud quality assessment model using local binary patterns. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2020.

[12] Rafael Diniz, Pedro Garcia Freitas, and Mylène CQ Farias. Local luminance patterns for point cloud quality assessment. In *2020 IEEE 22nd international workshop on Multimedia signal processing (MMSP)*, pages 1–6. IEEE, 2020.

[13] Rafael Diniz, Pedro Garcia Freitas, and Mylène Farias. A novel point cloud quality assessment metric based on perceptual color distance patterns. *Electronic Imaging*, 33:1–11, 2021.

[14] Rafael Diniz, Pedro Garcia Freitas, and Mylene CQ Farias. Color and geometry texture descriptors for point-cloud quality assessment. *IEEE Signal Processing Letters*, 28:1150–1154, 2021.

[15] Rafael Diniz, Pedro Garcia Freitas, and Mylene CQ Farias. Point cloud quality assessment based on geometry-aware texture descriptors. *Computers & Graphics*, 103:31–44, 2022.

[16] Evangelos Alexiou, Xuemei Zhou, Irene Viola, and Pablo Cesar. Pointpca: Point cloud objective quality assessment using pca-based descriptors. *Journal on Image and Video Processing*, 2024(1):20, 2024.

[17] Xuemei Zhou, Evangelos Alexiou, Irene Viola, and Pablo Cesar. Pointpca+: A full-reference point cloud quality assessment metric with pca-based features. *Signal Processing: Image Communication*, 2025.

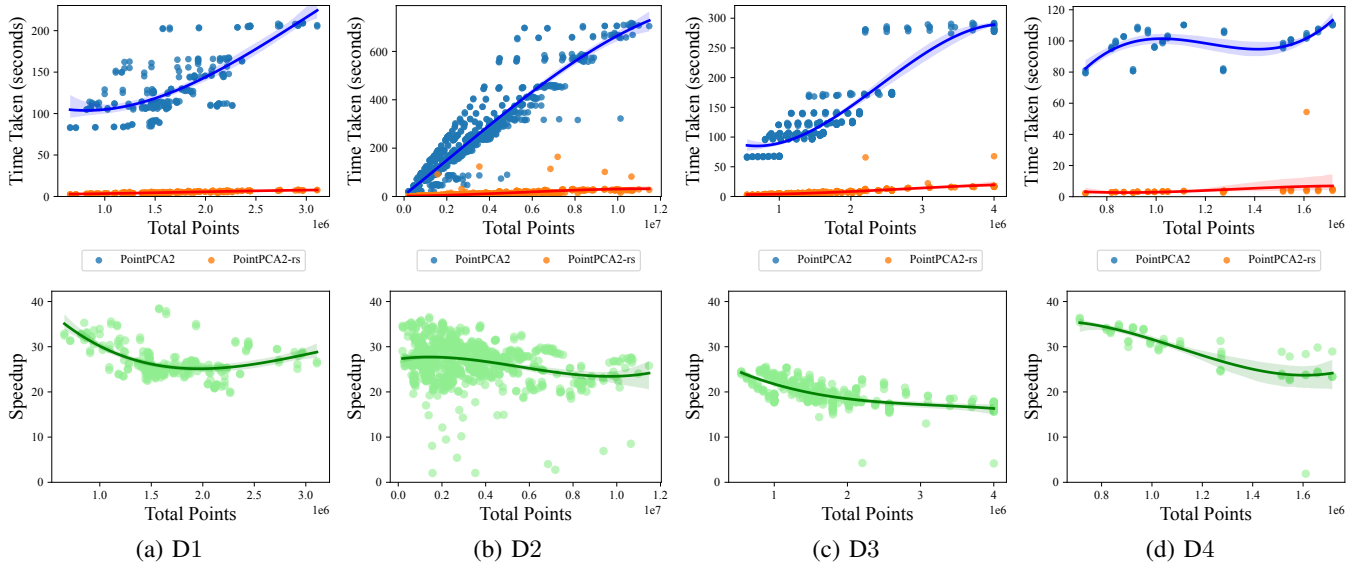

Fig. 1. Processing time as a function of the number of points (first row), and the corresponding relative speedup as a function of the number of points (second row). The time taken for extracting the features using PointPCA2 (baseline) and its enhanced algorithm PointPCA-RS (proposed). Speedup is computed as the time taken by PointPCA2 divided by the processing time of PointPCA2-RS, for each point cloud, as a function of its number of points.

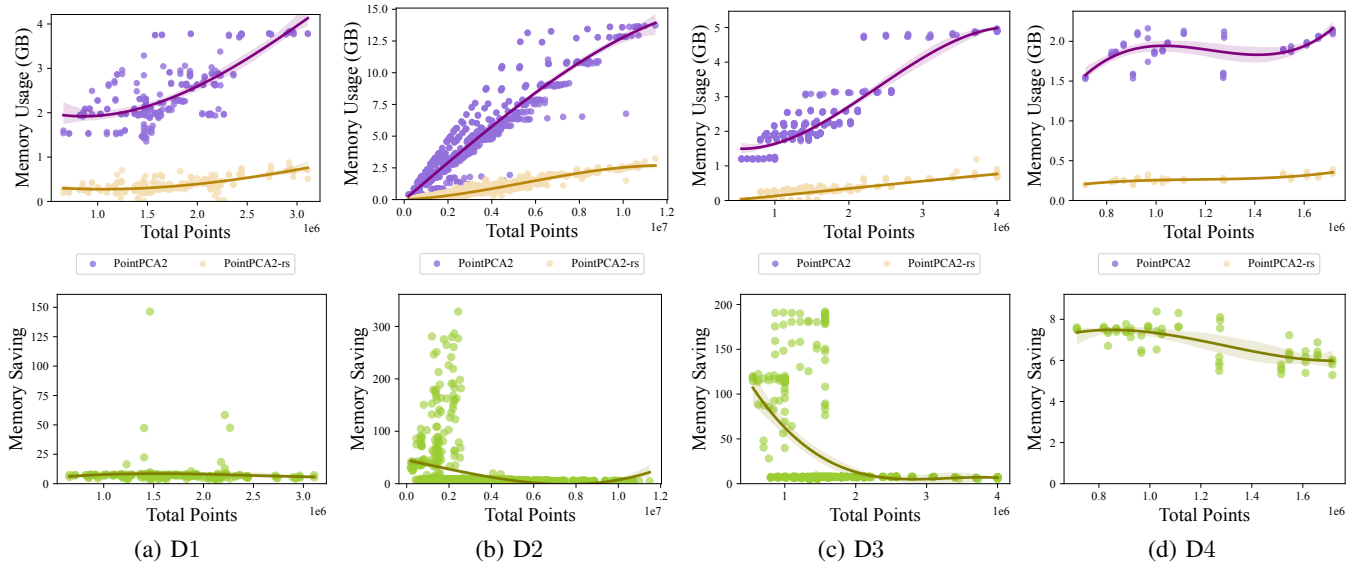

Fig. 2. Memory consumption depending on the number of points (first row) and relative decrease in RAM usage (second row). The peak of memory usage is seized as well as the number of points of the processed point clouds using both PointPCA2 and PointPCA-RS. The memory saving is measured by calculating the ratio of memory used by PointPCA2 to that used by PointPCA2-RS, with a higher ratio indicating more memory consumption.

[18] Rafael Diniz, Pedro Garcia Freitas, and Mylene CQ Farias. Multi-distance point cloud quality assessment. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3443–3447. IEEE, 2020.

[19] Evangelos Alexiou and Touradj Ebrahimi. Towards a point cloud structural similarity metric. In *2020 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pages 1–6. IEEE, 2020.

[20] Nicholas D Matsakis and Felix S Klock. The rust language. In *Proceedings of the 2014 ACM SIGAda annual conference on High integrity language technology*, pages 103–104, 2014.

[21] Marcus Müller, Lawrence Benson, and Viktor Leis. B-trees are back: Engineering fast and pageable node layouts. *Proceedings of the ACM on Management of Data*, 3(1):1–26, 2025.

[22] Leo A Goodman. Kolmogorov-smirnov tests for psychological research. *Psychological bulletin*, 51(2):160, 1954.

[23] Evangelos Alexiou, Irene Viola, Tomás M. Borges, Tiago A. Fonseca, Ricardo L. de Queiroz, and Touradj Ebrahimi. A comprehensive study of the rate-distortion performance in mpeg point cloud compression. *Transactions on Signal and Information Processing*, 8:e27, 2019.

[24] Ali Ak, Emin Zerman, Maurice Quach, Aladine Chetouani, Aljosa Smolic, Giuseppe Valenzise, and Patrick Le Callet. Basics: Broad quality assessment of static point clouds in a compression scenario. *IEEE Transactions on Multimedia*, 26:6730–6742, 2024.

[25] Qi Yang, Hao Chen, Zhan Ma, Yiling Xu, Rongjun Tang, and Jun Sun. Predicting the perceptual quality of point cloud: A 3d-to-2d projection-based exploration. *Transactions on Multimedia*, 23:3877–3891, 2020.

[26] Eric M. Torlig, Evangelos Alexiou, Tiago A. Fonseca, Ricardo L. de Queiroz, and Touradj Ebrahimi. A novel methodology for quality assessment of voxelized point clouds. In Andrew G. Tescher, editor, *Applications of Digital Image Processing XLI*, volume 10752, page 107520I. International Society for Optics and Photonics, SPIE, 2018.

[27] Qi Liu, Honglei Su, Zhengfang Duanmu, Wentao Liu, and Zhou Wang. Perceptual quality assessment of colored 3d point clouds. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2022.