Unified Geometry and Color Compression Framework for Point Clouds via Generative Diffusion Priors

Tianxin Huang[1]* Gim Hee Lee[2]
The University of Hong Kong[1], National University of Singapore[2]
huangtx@hku.hk and gimhee.lee@nus.edu.sg

Abstract

With the growth of 3D applications and the rapid increase in sensor-collected 3D point cloud data, there is a rising demand for efficient compression algorithms. Most existing learning-based compression methods handle geometry and color attributes separately, treating them as distinct tasks, making these methods challenging to apply directly to point clouds with colors. Besides, the limited capacities of training datasets also limit their generalizability across points with different distributions. In this work, we introduce a test-time unified geometry and color compression framework for 3D point clouds. Instead of training a compression model based on specific datasets, we adapt a pre-trained generative diffusion model to compress original colored point clouds into sparse sets, termed 'seeds', using prompt tuning. Decompression is then achieved through multiple denoising steps with separate sampling processes. Experiments on objects and indoor scenes demonstrate that our method has superior performances compared to existing baselines for the compression of geometry and color. Our codes would be released at https://github.com/Tianxinhuang/DiffCom.git.

1. Introduction

3D point clouds with color attributes are widely used in applications such as AR [45, 46], VR [4, 12], and autonomous driving [2, 13, 37]. With advancements in sensor and display technologies, the scale of 3D point clouds is rapidly increasing, placing greater demands on effective compression algorithms. Since early works [18, 52, 53], learning-based methods have become a research focus, demonstrating strong performance in compressing the geometric structures of 3D point clouds. More recent studies [25, 44, 51] have also explored the compression of attributes such as color, assuming the original coordinates are fully preserved, and designing networks to recover attributes from encoded features and coordinates. However, most existing approaches still treat geometry [19, 54] and color [39, 44, 51] compression as separate tasks, handled through different frameworks. Geometry compression frameworks often disregard color information, while attribute compression frameworks assume uncompressed coordinates. Although some works [30, 40] have attempted to integrate geometry and attribute compression within a single framework, their generalizability remains limited to specific small-scale datasets. As a result, most existing approaches fail to provide a unified and generalizable compression method for both geom-

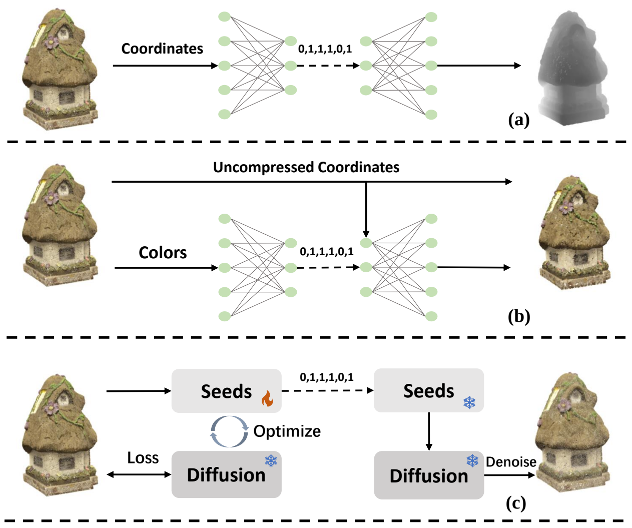

Figure 1. Differences between our framework and existing point cloud compression methods. Conventional geometry compression techniques (a) focus solely on compressing coordinates, leaving color information unhandled. Attribute compression methods (b) encode colors into binary codes but retain uncompressed coordinates to assist with color decompression. In contrast, our framework (c) simultaneously optimizes the compression of both coordinates and colors into sparse sets, or '**seeds**.' These seeds are encoded into binary codes using a non-learning-based method and, after decoding, are used in a denoising process with a diffusion model for decompression.

*Work done when the first author was a Research Fellow at the National University of Singapore.

etry structure and color attribute in 3D point clouds.

Point-E [32] is a 3D generative model capable of generating 3D point clouds and colors from text descriptions or images. Benefiting from training on a large collection of annotated point clouds, Point-E has great generalizability across point clouds with diverse categories and distributions. It comprises three diffusion models tailored to specific tasks: text-to-3D generation, image-to-3D generation, and point-to-point 3D up-sampling. The generative up-sampling model reconstructs high-resolution point clouds and colors from sparse inputs through a denoising process [16, 32], leveraging rich prior knowledge to achieve coarse-to-fine transformation.

By learning continuous transformations from noise to structured points, this diffusion model can generate point clouds at various resolutions through the denoising process guided by sparse point sets, with multiple separate sampling processes, making it potentially applicable for point cloud compression by retaining only the sparse data needed to reconstruct dense point clouds. Fine-tuning the pre-trained diffusion model of Point-E [32] is a straightforward approach to apply it for compression, but it requires substantial time and computational resources to adapt the entire model to specific datasets. To address this, we introduce prompt tuning for test-time optimization, adapting generative diffusion priors to specific point clouds for compression. Prompt tuning [10, 22, 41] is used to optimize prompts for frozen diffusion models, aligning generated output more closely with the ground truth. In our case, sparse point sets act as prompts and would be optimized accordingly to improve the precision of decompressed dense point clouds.

In this work, we propose the first test-time unified compression framework for both geometry and colors of 3D point clouds. As shown in Figs. 1-(a) and (b), most existing geometry and attribute compression methods handle coordinates and color attributes separately, treating them as distinct tasks. As shown in Fig. 1-(c), our method leverages the pre-trained up-sampling diffusion model from 3D generative model Point-E [32] to simultaneously compress coordinates and colors, reducing the point clouds into sparse point sets, or 'seeds,' via prompt tuning. These seeds could be further encoded into binary codes and decoded using a non-learning-based method such as G-PCC [14]. Then, these seeds are decompressed into full-resolution 3D point clouds by denoising steps in multiple separate sampling processes.

This approach enables compression and decompression purely through the strong 3D priors of a pre-trained generative diffusion model, eliminating the need for additional dataset-specific training. Prompt tuning during compression requires only short-term optimization, taking approximately 3 to 5 minutes on an RTX 4090 Ti for a 160k-point cloud. Additionally, we introduce a simple patch division strategy to better handle dense point clouds, effectively balancing compressed bit length and decompression accuracy.

Our contribution in this work can be summarized as:

- We present a unified geometry and color compression framework based on a pre-trained diffusion model;
- By introducing prompt tuning for the pre-trained diffusion model, we propose the first test-time compression and decompression pipeline;
- Experiments on diverse data confirm that our method effectively handles 3D point clouds with colors, achieving superior performances compared to existing methods.

2. Related Works

2.1. 3D Generation via Diffusion Models

Following the success of 2D diffusion models in text-to-image generation [16, 38, 43], 3D generation has gained significant research interest. To create 3D representations such as point clouds, meshes, NeRF [28], or 3D Gaussian primitives [21] from text or image inputs, researchers have explored two primary approaches. The first approach involves adapting 2D priors for 3D generation [24, 26, 29, 33, 47, 50]. These methods typically optimize specific 3D representations using guidance from 2D diffusion models from different viewpoints, employing Score Distillation Sampling (SDS) [33] through rendered images. These approaches leverage robust 2D pre-trained priors by fine-tuning view-guided diffusion models based on Stable Diffusion [38]. The second approach [17, 20, 32, 49] involves direct 3D generation by training on large-scale 3D datasets [5, 7]. These models offer improved efficiency and perform well with 3D objects, though scene-level generation still presents challenges due to limited data. Among these methods, Point-E [32] stands out for its ability to directly generate colored 3D point clouds from texts, images, or sparse point clouds, presenting promising potential for application in 3D point cloud compression.

2.2. Point Cloud Compression

3D point clouds consist of discrete points in 3D space without inherent topology. Existing point cloud compression methods can be broadly classified into non-learning-based and learning-based approaches. Non-learning-based methods [3] typically use manually defined encoding rules and data structures, such as octrees [14, 42] and kd-trees [11]. Learning-based geometry compression methods, on the other hand, seek to enhance compression by learning memory-efficient representations. Some approaches [31, 35, 52, 53] convert point clouds into voxel representations and use 3D CNNs to encode geometric structures into binary formats, though this may introduce precision loss. To mitigate this, point-based compression frameworks [15, 18, 19, 55] directly operate on points, avoiding voxelization errors.

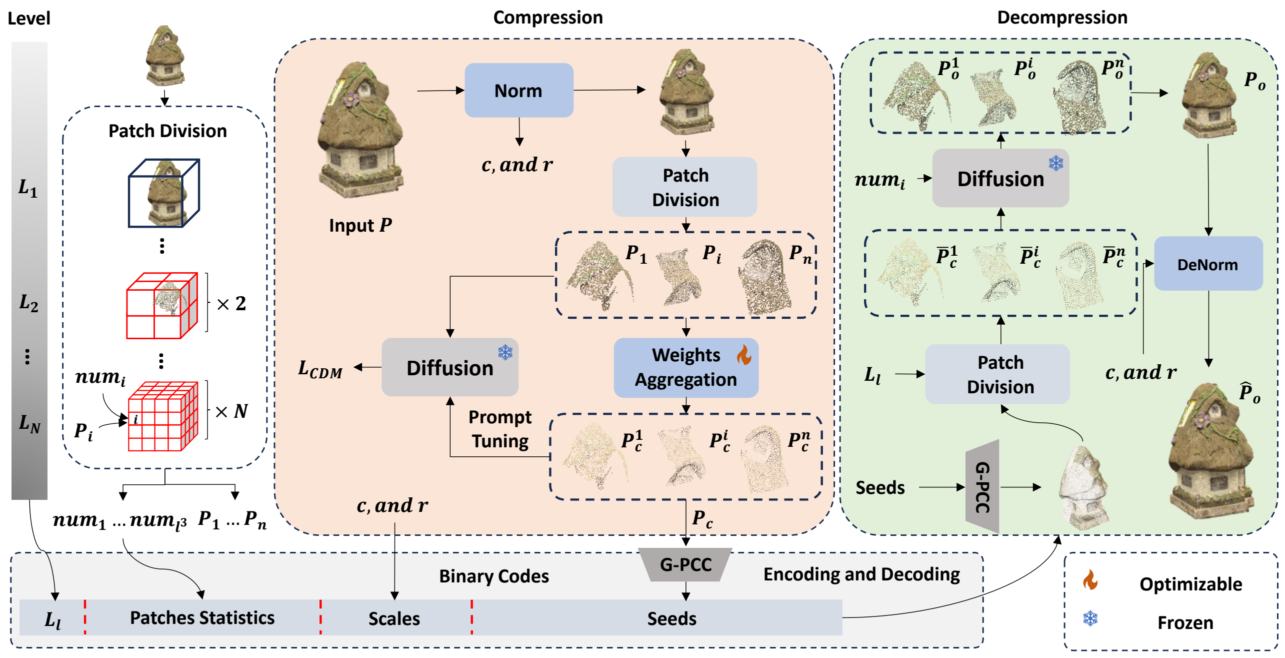

Figure 2. The overall pipeline of our method. During compression, the input point cloud P is first normalized using the calculated center c and radius r, then divided into n patches $P_1, \ldots, P_n$ according to the chosen compression level L_l using Patch Division. Seeds $P_c^1, \ldots, P_c^n$ for each patch are obtained through Weights Aggregation and optimized via prompt tuning using our proposed loss L_{CDM} computed on the pre-trained up-sampling diffusion model. Subsequently, L_l, patch statistics (including patch resolutions), scales (center c and radius r), and seeds $P_c = [P_c^1, \ldots, P_c^n]$ are encoded into binary codes, which are then decoded for decompression. During decompression, the decoded seeds $\bar{P}_c$ are split into patches $\bar{P}_c^1, \ldots, \bar{P}_c^n$ using patch division. Each patch $\bar{P}_c^i$ is up-sampled to its recorded resolution num_i through the diffusion model's denoising process, yielding P_o^i. The concatenated output $P_o = \{P_o^i | i = 1 \ldots n\}$ is then denormalized to produce the final output $\hat{P}_o$ using the recorded scales c and r.

However, these learning-based methods primarily focus on geometry compression and lack the capability to process color attributes. In contrast, non-learning-based approaches such as Draco [11], G-PCC [14], and V-PCC [14] support both geometry and color compression. Consequently, recent research [25, 39, 44, 51] has increasingly emphasized attribute compression for normals and colors. Methods like SparsePCAC [51] and ProgressivePCAC [39] leverage sparse convolutions to encode attributes based on 3D coordinates, assuming the geometry remains uncompressed while compressing only colors or normals. Although some works [30, 40] attempt to compress both geometry and attributes simultaneously, they still require extensive training on specific datasets to adapt to different point cloud distributions, limiting their generalizability.

In this work, we propose a test-time unified compression framework for 3D point clouds with colors. Leveraging Point-E's generative capabilities [32], our method decompresses both coordinates and colors through its denoising process while compressing the original point clouds into sparse sets via prompt tuning. We focus on the lossy compression [25, 35, 53], primarily evaluating our framework's performance under constrained encoding bit budgets.

3. Methodology

To accelerate the compression and decompression process, we only use the first T steps from Point-E's original diffusion-based up-sampling model [32] in this work. As illustrated in Fig. 2, our pipeline consists of Patch Division, Compression, Encoding and Decoding, and Decompression.

3.1. Patch Division

Point-E's original diffusion-based upsampling model [32] generates only sparse 3,072-point outputs from 1,024-point input prompts. To extend its capability to higher-resolution point clouds, we introduce a patch division strategy that segments the original point cloud into N smaller patches.

As illustrated in Fig. 2, the patch division is performed via simple voxelization. The input point cloud is partitioned into patches $\{P_i | i = 1, \ldots, n\}$ using an l^3 resolution grid at level L_l. These patches are processed sequentially based on their xyz coordinates, and the number of points in each grid $\{num_i | i = 1, \ldots, l^3\}$ is recorded to preserve resolution information for decompression. Although the number of grids appears to grow cubically, our experiments show that setting $l <= 10$ is sufficient for most cases. Please note that we balance the encoded bits and geometric details by adjusting the voxelization resolution, where more patches allows for better preserved details but longer encoded bits.

3.2. Compression Process

Weights Aggregation. In this module, we first employ the boundary sampling (BDSam) method from [19] to initialize the seeds for each i-th patch P_i, ensuring that the boundary sizes of patches are preserved. During subsequent optimization, seeds positioned on the boundaries are detached to maintain the boundary integrity. To mitigate the significant loss of local geometry and color information that can occur during sampling, we propose re-encoding each point in the initialized seeds P_c^s through weighted interpolation using k neighboring points from the original dense point cloud P. This approach aggregates neighboring information to enhance the initialization quality. The k weights for the neighbor of each point in the seeds are initialized as an approximated one-hot array, where the weight for the original neighbor center is 1 and others are $1e^{-4}$.

Given the initialized sparse seeds P_c^s, the KNN operation is used to obtain their k neighbors $P_k = N_k(P_c^s, P)$ from the original point cloud P, with weights $W_c \in \mathbb{R}^{|P_c^s| \times k}$. The refined seeds P_c are then computed using a function $g(\cdot)$ defined as follows:

$$P_c = g(W_c, P_k) = \sum \frac{|W_c|}{\sum |W_c|} \cdot P_k \tag{1}$$

During subsequent prompt tuning, we further optimize W_c to more accurately capture the geometry and colors of the ground truth point cloud. By focusing on weight optimization rather than directly adjusting the seeds, we can maintain seed coordinates and colors within a reasonable range, promoting stability in the optimization process.

Prompt Tuning. After initialization, we apply prompt tuning as introduced by [22, 41] to refine the seeds. During prompt tuning, the parameters of Point-E's diffusion-based model remain frozen, and only weights W_c are optimized through a loss calculated with the original point cloud P.

Given the i-th patch out of n patches, P_i, with corresponding seeds P_c^i, and since the up-sampling model output is limited to 3072 points, we randomly sample 3072 points from P_i as the input $x_0 \sim \text{RandomSample}(P_i, 3072)$ for the diffusion model. Following [16, 38], the common diffusion model loss is defined as:

$$L_{DM} = \mathbb{E}_{x_0, \epsilon \sim \mathcal{N}(0,1), t}[\|\epsilon - \epsilon_\theta(x_t, P_c^i, t)\|_2^2], \tag{2}$$

where t is a time step uniformly sampled from $\{1, 2, \ldots, T\}$, and $\epsilon_\theta(x_t, P_c^i, t)$ is the noise prediction network conditioned on the seeds P_c^i and time step t. The noisy version x_t of the input x_0 at time step t is defined as:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon. \tag{3}$$

Algorithm 1 The Compression process.

```
1:  Input: Divided patches $\hat{P} = [P_1, ..., P_n]$, the number of
    iterations $N_{iter}$, KNN operation $N_k(\cdot, \cdot)$.
2:  Down-sample the patches with BDSam to initialized
    seeds $P_c$:
3:  Initialize $P_c = [], P_c^s = []$
4:  for $i = 1$ to $n$ do
5:      $P_c^s.insert(BDSam(P_i, 1024))$
6:  end for
7:  $P_k = N_k(P_c^s, P)$, Initialize $W_c \in \mathbb{R}^{|P_c^s| \times k}$
8:  for $i = 1$ to $N_{iter}$ do
9:      Randomly select idx $\sim$ Uniform$(1, n)$
10:     Updating $W_c$ by descending gradient:
        $\nabla_{W_c} L_{CDM}(P_{idx}, g(W_c, P_k)[idx], t)$
11: end for
12: Calculate seeds $P_c = g(W_c, P_k)$
13: Output: the optimized seeds $P_c$.
```

Here, α_t and $\bar{\alpha}_t$ denote constants selected as in [16].

As point clouds are permutation-invariant [34], altering the order of individual points does not affect the overall 3D shape. Chamfer Distance (CD) [9] is commonly used to measure shape differences between two point clouds through point-to-point matching.

From Eq. 2, we observe that the original diffusion model loss does not account for the permutation invariance of point clouds, implicitly assuming that the added noise ϵ during noising and the predicted noise $\epsilon_\theta(x_t, P_c^i, t)$ during denoising follow a consistent permutation. However, two similar point distributions do not necessarily share the same order. To address this, we introduce a matching-based CD loss to relax this problem. The Chamfer Distance is defined as:

$$\begin{aligned} \mathcal{L}_{CD}(S_g, S_o) = \frac{1}{2}\Big(&\frac{1}{|S_g|} \sum_{x \in S_g} \min_{y \in S_o} \|x - y\|_2 \\ &+ \frac{1}{|S_o|} \sum_{x \in S_o} \min_{y \in S_g} \|x - y\|_2\Big), \end{aligned} \tag{4}$$

where S_g and S_o are two point clouds.

Given a time step t, following Eq. 3, the noised result at step $t-1$ is calculated as $x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} x_0 + \sqrt{1 - \bar{\alpha}_{t-1}} \epsilon$. Our proposed Chamfer Diffusion Model Loss is defined as:

$$L_{CDM}(P_i, t) = L_{CD}(x_{t-1}, \bar{x}_{t-1}), \tag{5}$$

where $\bar{x}_{t-1}$ represents the denoised distribution at step $t-1$, calculated as:

$$\bar{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, P_c^i, t) \right). \tag{6}$$

Here, L_{CDM} employs the Chamfer Distance loss to measure the difference between the noised distribution x_{t-1} and the denoised distribution $\bar{x}_{t-1}$ at step $t-1$, mitigating potential issues caused by permutations. As the $\bar{x}_0$ can also be estimated by $\bar{x}_0 = \frac{x_t - \sigma_t \epsilon_\theta(x_t, t)}{\sqrt{\alpha_t}}$, we also try the loss:

$$L_{inver} = L_{CD}(x_0, \bar{x}_0) \tag{7}$$

Their performances are compared in Sec. 4.6. After optimization, W_c are transformed into seeds using Eq. 1 for encoding. The full compression pipeline is provided in Alg. 1.

3.3. Encoding and Decoding

The encoding and decoding process aims to achieve a lossless transformation between the decompression information and binary codes. As illustrated in Fig. 2, the features to be encoded consist of four components: compression level L_l, patch statistics, scales, and seeds. The compression level and patch statistics are stored as a 1×8 bits char and $l^3 \times 8$ bits chars, respectively. Scales, including the point cloud centers and radius, are saved as 4×32 bits floats. Seeds are concatenated, and compressed using G-PCC [14] in a lossless manner. In the decoding process, the compression level, patch statistics, scales, and seeds are sequentially decoded and prepared for usage in decompression.

3.4. Decompression Process

During decompression, the decoded seeds $\bar{P}_c$ are divided into patches again using patch division, based on the decoded compression level L_l. Each patch $\bar{P}_c^i$ from the i-th of n patches is then used to generate the output point cloud P_o^i at a specific resolution num_i, estimated from the patch statistics as shown in Fig. 2.

The decompression $f_D(\cdot)$ is carried out through a denoising process with multiple sampling steps, as described in Alg. 2. This process yields $P_o^i = f_D(\bar{P}_c^i, num_i)$, and the full decompressed output is $P_o = [P_o^1, \ldots, P_o^n]$.

Finally, the decompressed point cloud is de-normalized to its original scale using the decoded center c and radius r as follows:

$$\hat{P}_o = [P_o^g \cdot r + c, P_o^c]. \tag{8}$$

P_o^g and P_o^c denotes coordinates and colors, respectively.

Algorithm 2 Decompression Process $f_D(\bar{P}_c^i, num_i)$

```
1: Input: Patch centers $\bar{P}_c^i$, target resolution $num_i$, constant $\sigma_t$ (as defined in Point-E [32]), Gaussian noise $\epsilon$.
2: Calculate the number of iterations:
3: $N_s = \text{round}(num_i / 3072)$
4: Initialize $P_o^i = []$
5: for $i = 1$ to $N_s$ do
6:     Sample $x_T = \sqrt{\bar{\alpha}_T}\bar{P}_c^i + \sqrt{1-\bar{\alpha}_T}\epsilon$
7:     Denoise the sampled $x_T$:
8:     for $t = T$ to $1$ do
9:         Update $x_t = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x_t, \bar{P}_c^i, t)\right)$
10:        if $t > 1$ then
11:            Add noise $x_t = x_t + \sigma_t \epsilon$
12:        else
13:            Set $x_0 = x_t$
14:        end if
15:    end for
16:    Append $x_0$ to $P_o^i$
17: end for
18: Output: Decompressed results $P_o^i$
```

4. Experiments

4.1. Dataset and Implementation Details

To comprehensively evaluate our method, we conduct unified compression experiments on colored point clouds from point cloud object models [23] and indoor scenes [6], as well as geometry compression experiments [19]. Unlike the voxelized 8iVFB dataset [8], we construct evaluation data by randomly sampling 160k colored points from objects and indoor scenes following 3QNet [19], which we believe better represents real-world point cloud distributions.

Our implementation is based on PyTorch, with seeds optimized using the Adam optimizer at a learning rate of 1e-3 for approximately 1000 iterations during prompt tuning, without requiring re-training on specific datasets. Additional implementation details are available in the supplementary. To validate our method's effectiveness, we include comparisons with representative non-learning methods, Draco [11] and G-PCC [14], as baselines. Although methods like CNet [30] claim to compress both geometry and color attributes, they do not provide open-source repositories or pretrained models. Therefore, we construct learning-based baselines for unified compression by manually integrating geometry and attribute compression methods. Specifically, we use widely used open-source geometry compression methods, PCGCv2 [52] and SparsePCGC [54] to compress coordinates, and the latest open-source attribute compression framework ProgressivePCAC [39] to compress colors. The encoded bits from both components are then combined to form a unified representation.

4.2. Metrics

Following prior work [35, 52, 53], we evaluate compression bit rate using bits per point (bpp) and reconstruction quality using Peak Signal-to-Noise Ratio (PSNR). The bpp metric, defined as the average number of encoding bits per point, is calculated as:

$$\text{bpp} = \frac{L}{N}, \tag{9}$$

where L is the size of the compressed binary, and N is the number of points in the compressed point cloud. For geometry evaluation, inspired by [19, 48], PSNR is defined as:

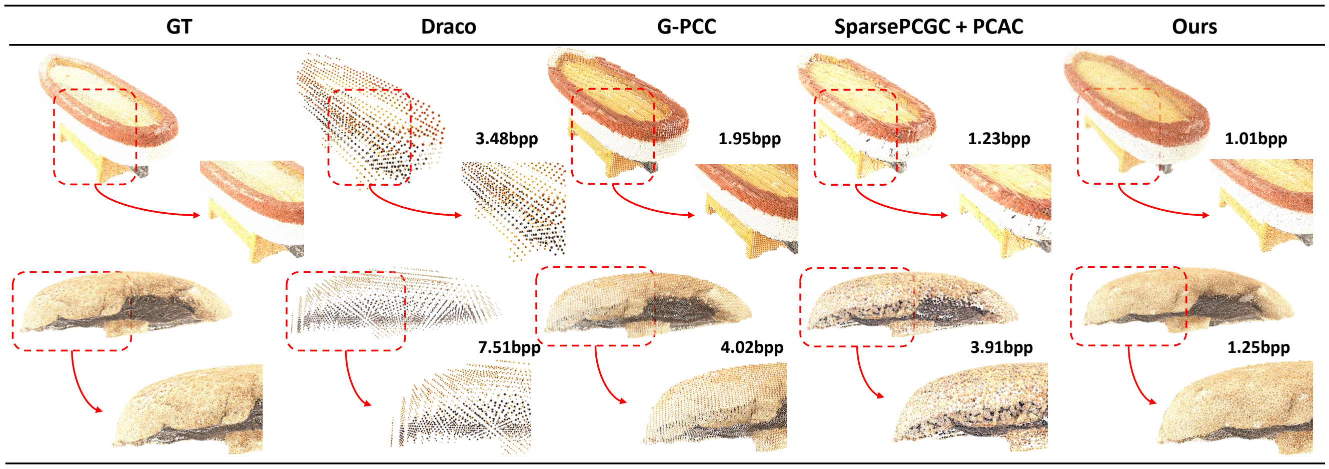

Figure 3. Qualitative comparisons on 3D objects with colors.

$$\text{PSNR} = 10\log_{10}\left(\frac{\max_{x\in S_1}\|x-\hat{x}\|_2}{Dis(S_1,S_2)}\right), \qquad (10)$$

where S_1 and S_2 represent the ground truth and decompressed point clouds, respectively, and $\hat{x}$ is the nearest neighbor of x in S_1. $Dis(S_1, S_2)$ is the two-way average distance between S_1 and S_2, measured by either point-to-point or point-to-plane errors. We denote PSNR calculated from point-to-point or point-to-plane distances as $PSNR_{D_1}$ and $PSNR_{D_2}$, respectively.

For color evaluation, PSNR is defined as:

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{Dis(C_1,C_2)}\right), \qquad (11)$$

where C_1 and C_2 are the color attributes of S_1 and S_2. The color distance $Dis(C_1, C_2)$ is calculated as the average color difference between each point and its nearest neighbor in YUV space. The PSNR for colors is noted as $PSNR_{D_3}$.

Since PSNR varies with bpp as encoded information changes, we utilize BD-PSNR as a quantitative metric to assess our method's relative improvement over others, following [1, 35]. BD-PSNR based on $PSNR_{D_1}$ and $PSNR_{D_2}$ reflects geometry performances, while that based on $PSNR_{D_3}$ assesses color compression quality.

4.3. Comparisons on Colored Objects

In this section, we evaluate the performance of our method on 3D objects with color attributes. Quantitative results are shown in Table 1, where performance is measured using BD-PSNR to capture relative improvements over other colored point cloud compression baselines. **Please note that our method does not have a direct entry in Table 1 since BD-PSNR represents relative improvements, with '+' and '-' indicating increases and decreases compared to existing methods, respectively**.

The results show significant improvements in geometry metrics PSNR-D1 and PSNR-D2 over existing baselines, while maintaining slightly improved performance on color metrics. Qualitative comparisons are shown in Fig. 3. Non-learning-based methods like Draco and G-PCC often exhibit regular distortions due to direct octree-based quantization and coordinate encoding, while the PCGCv2+PCAC baseline constructed for comparison displays artifacts such as holes. In contrast, our method produces smooth decompressed results with lower bpp.

4.4. Comparisons on Indoor Scenes

To further assess our method's performance on real scanned point cloud data, we evaluate it on indoor scenes from ScanNet [6]. Qualitative and quantitative comparisons are shown in Fig. 4 and Table 2, respectively. As illustrated in Fig. 4, our method produces smoother results closer to the ground truth with lower bpp. The results in Table 2 demonstrate that our method still achieves obvious improvements over existing baseline models, revealing its generalizability from object-level to scene-level point clouds.

4.5. Comparison on Geometry Compression

Although our method is designed to simultaneously compress the geometry and colors of point clouds, it can also be applied to pure geometry compression by assigning constant colors, such as white or black, to the input point clouds. In this section, we conduct a simple comparison between our method and several representative geometry compression approaches by setting the point cloud colors to white. The quantitative results are shown in Table 3.

The results demonstrate that our method outperforms existing geometry compression baselines while requiring no dataset-specific training, unlike methods such as 3QNet [19] and PCGCv2 [52]. These improvements can be attributed to the strong generalization capability of the

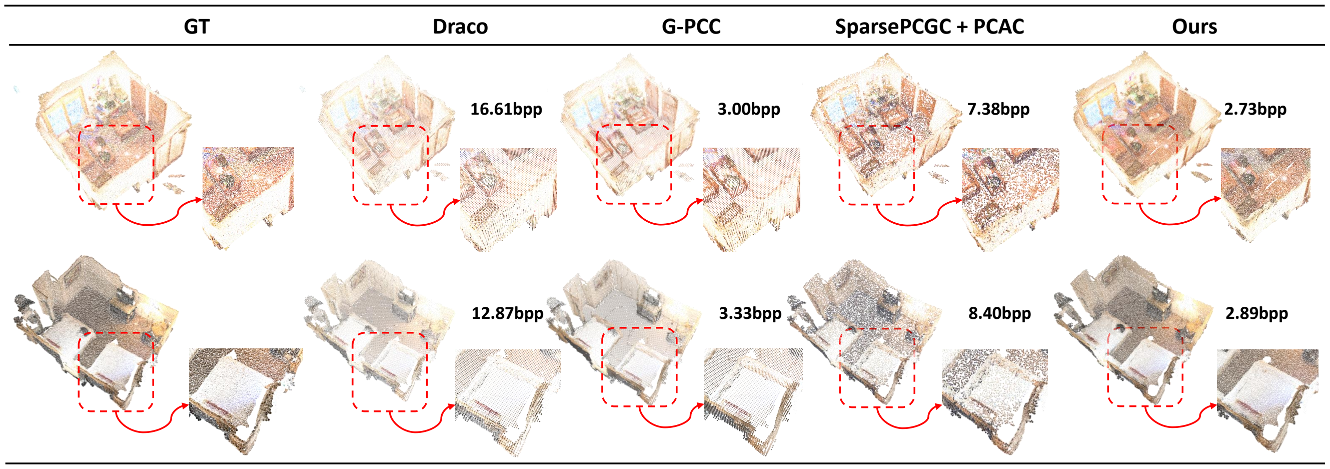

Figure 4. Qualitative comparisons on 3D indoor scenes with colors.

Objects		PCGCv2+PCAC	G-PCC	Draco	SparsePCGC+PCAC
ship	D1	+12.22(51.12%)	+11.35(44.41%)	+32.84(61.03%)	+4.61(24.79%)
	D2	+11.52(42.15%)	+10.98(43.42%)	+30.51(59.71%)	+1.33(5.95%)
	D3	+9.08(25.43%)	+3.02(10.08%)	+15.06(47.98%)	+3.08(10.51%)
mushroom	D1	+11.32(39.43%)	+9.71(33.74%)	+21.74(48.51%)	+7.91(25.69%)
	D2	+7.83(24.50%)	+10.50(32.50%)	+20.03(44.89%)	+4.39(12.51%)
	D3	+4.20(14.64%)	+1.51(5.15%)	+7.77(28.93%)	+5.66(17.88%)
statue	D1	+10.70(59.91%)	+7.81(39.91%)	+21.12(47.80%)	+5.46(29.24%)
	D2	+6.80(32.26%)	+8.06(39.53%)	+18.98(43.69%)	+1.67(7.28%)
	D3	+5.44(18.20%)	+0.25(0.89%)	+7.42(28.68%)	+3.56(11.91%)
litchi	D1	+7.11(51.77%)	+8.63(40.24%)	+26.33(53.02%)	+5.54(31.48%)
	D2	+3.48(22.62%)	+7.62(36.50%)	+22.70(49.14%)	+2.41(11.38%)
	D3	+7.28(20.94%)	+3.02(9.30%)	+10.70(32.52%)	+10.86(27.85%)
pencontainer	D1	+15.32(44.24%)	+8.14(27.16%)	+24.90(54.82%)	+11.83(32.18%)
	D2	+12.76(33.01%)	+8.32(24.91%)	+23.38(52.55%)	+9.35(22.18%)
	D3	+8.50(28.86%)	+1.05(4.02%)	+10.41(38.50%)	+11.19(34.36%)
pineapple	D1	+10.38(55.24%)	+9.34(44.57%)	+25.90(53.93%)	+7.39(33.85%)
	D2	+4.79(21.84%)	+9.90(44.05%)	+24.51(50.05%)	+3.76(14.09%)
	D3	+3.43(12.52%)	+1.48(5.25%)	+9.26(32.24%)	+8.55(25.87%)
pear	D1	+13.25(65.58%)	+10.61(47.05%)	+27.39(54.37%)	+7.33(45.74%)
	D2	+7.42(35.06%)	+13.65(51.33%)	+28.44(53.00%)	+2.18(9.96%)
	D3	+8.35(23.18%)	+2.28(6.22%)	+10.51(28.44%)	+14.30(36.98%)
bumbameuboi	D1	+10.70(38.23%)	+5.99(23.24%)	+20.80(47.81%)	+8.22(29.27%)
	D2	+7.52(22.90%)	+7.45(25.02%)	+19.88(45.43%)	+1.44(4.61%)
	D3	+2.60(10.03%)	-1.21(4.97%)	+5.26(21.69%)	+3.00(11.34%)
house	D1	+8.25(69.17%)	+8.32(40.74%)	+26.83(55.54%)	+3.88(28.11%)
	D2	+4.12(28.52%)	+8.15(39.14%)	+24.64(51.81%)	+0.34(1.94%)
	D3	+3.37(12.68%)	+1.36(5.02%)	+10.94(38.96%)	+4.40(14.76%)
Aver	D1	+11.03(52.74%)	+8.88(37.90%)	+25.32(52.98%)	+6.91(31.15%)
	D2	+7.36(29.20%)	+9.40(37.38%)	+23.67(50.03%)	+2.99(9.99%)
	D3	+5.81(18.50%)	+1.42(4.55%)	+9.70(33.11%)	+7.18(21.27%)

Table 1. Quantitative comparisons on Colored Objects. **Items in the table denote the relative improvements of our method over other methods measured with BD-PSNR. '+' and '-' indicates performance increases and decreases, respectively.**

diffusion-based up-sampling model in Point-E [32], which reconstructs continuous surfaces from optimized sparse points, or 'seeds.' By encoding only the seeds for compression, our method eliminates the need for additional encoded features, as required by methods like 3QNet [19], resulting in shorter compressed binary codes.

Objects		PCGCv2+PCAC	G-PCC	Draco	SparsePCGC+PCAC
scene0	D1	+8.60(57.69%)	+7.84(34.92%)	+24.52(56.19%)	+0.33(2.19%)
	D2	+4.82(26.18%)	+6.68(31.64%)	+22.71(70.73%)	-2.58(13.72%)
	D3	+2.12(7.94%)	+2.23(8.06%)	+10.14(39.66%)	-0.89(3.25%)
scene1	D1	+7.76(36.16%)	+6.03(29.28%)	+24.15(54.79%)	+3.61(15.54%)
	D2	+5.32(21.68%)	+5.27(22.45%)	+21.81(55.72%)	+1.06(3.96%)
	D3	+4.66(16.42%)	+1.58(5.51%)	+10.65(37.19%)	+4.20(13.76%)
scene2	D1	+11.71(75.77%)	+7.41(36.35%)	+23.55(53.11%)	+4.72(28.63%)
	D2	+7.87(43.19%)	+6.67(33.99%)	+20.91(53.13%)	+0.09(0.49%)
	D3	+5.67(17.43%)	+2.46(7.52%)	+12.74(40.19%)	+3.01(9.02%)
scene3	D1	+9.84(62.57%)	+7.45(34.88%)	+23.37(53.36%)	+4.16(25.96%)
	D2	+6.07(32.28%)	+6.10(29.97%)	+21.58(56.08%)	+0.43(2.23%)
	D3	+3.99(14.53%)	+2.08(7.43%)	+9.91(37.98%)	+3.00(10.57%)
scene4	D1	+8.20(65.09%)	+5.50(30.00%)	+23.56(54.38%)	+1.00(7.58%)
	D2	+3.82(24.89%)	+4.68(26.90%)	+20.89(53.19%)	-5.17(31.60%)
	D3	+3.20(11.34%)	+0.11(0.39%)	+11.26(40.17%)	-2.38(7.98%)
scene5	D1	+10.88(71.09%)	+8.90(39.64%)	+27.36(56.41%)	+3.54(24.70%)
	D2	+6.49(34.38%)	+8.61(39.45%)	+24.29(56.74%)	-1.06(5.95%)
	D3	+8.66(26.39%)	+3.49(10.94%)	+13.93(43.82%)	+3.42(10.81%)
Aver	D1	+9.50(61.40%)	+7.19(34.18%)	+24.42(54.71%)	+2.89(17.43%)
	D2	+5.73(30.44%)	+6.33(30.73%)	+22.03(57.60%)	-1.20(7.43%)
	D3	+4.72(15.67%)	+1.99(6.64%)	+11.44(39.83%)	+1.73(5.49%)

Table 2. Quantitative comparisons on indoor scenes. Items in the table denote the relative improvements of our method over other methods measured with BD-PSNR.

4.6. Ablation Study

Ablation study for different components To assess the impact of our proposed components, we conduct an ablation study in Table 4. The terms *Decom*, *Weights*, and *Tuning* refer to the decompress, weights aggregation, and prompt tuning described in Sec. 3, respectively. *Seeds* denotes using seeds directly as the results without decompression. Performance is evaluated using BD-PSNR of our method against G-PCC, calculated with $PSNR_{D1}$, $PSNR_{D2}$, and $PSNR_{D3}$. The results indicate that removing any component decreases overall performance, confirming that each component contributes to the final outcome.

Objects		3QNet	PCGCv2	G-PCC	SparsePCGC
eros	D1	+5.44(31.74%)	+25.43(69.36%)	+5.18(27.34%)	+9.26(43.57%)
	D2	+1.87(8.09%)	+13.93(59.85%)	+4.17(16.01%)	+5.87(22.52%)
kitten	D1	+7.80(38.40%)	+37.48(76.12%)	+3.90(21.40%)	+6.51(33.67%)
	D2	+8.67(31.40%)	+29.01(74.08%)	+5.64(19.85%)	+1.72(5.86%)
boy01	D1	+2.15(14.98%)	+20.03(63.13%)	+2.30(16.52%)	+8.34(50.15%)
	D2	+2.18(10.09%)	+10.22(51.89%)	+2.32(10.88%)	+10.05(38.71%)
julius	D1	+0.64(4.29%)	+21.08(60.57%)	+5.43(32.15%)	+6.56(34.86%)
	D2	-2.12(10.00%)	+7.61(40.55%)	+5.46(22.87%)	+0.41(1.58%)
pig	D1	+3.80(23.07%)	+24.91(71.26%)	+4.84(29.04%)	+9.18(49.36%)
	D2	+5.42(21.24%)	+15.99(66.24%)	+7.29(27.64%)	+7.95(28.62%)
biggirl	D1	+2.62(16.77%)	+21.34(67.34%)	+2.18(13.98%)	+8.71(52.02%)
	D2	+3.50(15.73%)	+14.09(54.67%)	+2.58(10.85%)	+10.20(39.42%)
pierrot	D1	+0.04(0.28%)	+8.25(37.97%)	+4.20(27.76%)	+7.34(37.82%)
	D2	+0.43(1.76%)	-0.42(2.49%)	+4.45(22.45%)	+1.43(4.89%)
nicolo	D1	+0.42(2.90%)	+19.50(57.31%)	+1.45(9.72%)	+4.17(23.81%)
	D2	-1.90(9.42%)	+7.14(34.26%)	+0.16(0.75%)	-4.04(15.85%)
duck	D1	-1.22(7.86%)	+8.62(34.96%)	+4.09(27.29%)	+7.53(38.86%)
	D2	-0.79(3.28%)	+1.28(6.51%)	+4.77(23.26%)	+1.40(4.84%)
star	D1	+3.64(21.85%)	+23.59(60.82%)	+5.50(30.13%)	+6.04(31.66%)
	D2	+3.61(13.78%)	+15.78(60.60%)	+9.18(31.51%)	+2.83(9.38%)
Aver	D1	+2.53(14.64%)	+21.02(59.88%)	+3.91(23.53%)	+7.36(39.58%)
	D2	+2.09(7.94%)	+11.46(44.61%)	+4.60(18.61%)	+3.78(14.00%)

Table 3. Quantitative comparisons on geometry compression. Items denote the relative improvements of our method over other methods measured with BD-PSNR.

Seeds	Decom	Weights	Tuning	D1	D2	D3
✓				-0.99	+5.49	-2.97
✓	✓			+8.27	+8.67	+0.91
✓	✓	✓		+8.37	+8.67	+0.97
✓	✓	✓	✓	**+8.88**	**+9.40**	**+1.42**

Table 4. Ablation Study for proposed components. Performances are evaluated with BD-PSNR of our method against G-PCC.

Ablation study for tuning losses In this work, we introduce the Chamfer Diffusion Model Loss L_{CDM} to replace the original MSE-based diffusion model loss L_{DM}. To assess its effectiveness, we compare various losses for prompt tuning, as shown in Table 5. The results, measured using BD-PSNR, evaluate our method against G-PCC. Here, L_{inver}, L_{DM}, and L_{CDM} refer to the inverse loss, diffusion model loss, and Chamfer diffusion model loss, respectively, as described in Sec. 3. As shown in Table 5, our proposed L_{CDM} achieves the best performance in prompt tuning, while L_{inver} and L_{DM} even perform worse than the framework without prompt tuning on certain metrics. This confirms the permutation-invariance of point clouds limits L_{DM}'s effectiveness for point generation, whereas our proposed loss effectively addresses this limitation.

Metrics	No tuning	L_{inver}	L_{DM}	L_{CDM}(ours)
D1	+5.90	+8.42	+6.19	**+8.88**
D2	+5.92	+8.70	+6.36	**+9.40**
D3	-2.81	+0.62	-2.49	**+1.42**

Table 5. Ablation Study for losses of prompt tuning. Performances are evaluated with BD-PSNR of our method against G-PCC.

4.7. Discussion about the Efficiency

In this section, we present a brief comparison of the efficiency of our method with other baselines, as shown in Table 6. While our method requires more time for compression due to prompt tuning, the preprocessing and decompression times remain affordable, and our approach requires no training on specific datasets while offering strong robustness. These characteristics make our method well-suited for offline applications such as cultural heritage digitization. Additionally, the efficiency of test-time prompt tuning could be further enhanced using techniques like Mixed Precision Training [27] or tensor/model parallelism [36], potentially reducing its computational overhead.

Methods		Pre-process	Compress	Decompress	Training
Non-Learning	Draco	0	0.30s	0.10s	0
	G-PCC	0	0.50s	0.06s	0
Learning	PCGCv2+PCAC	3.36s	1.55s	3.37s	days
	Ours	0.76s	4 min	6.05s	0

Table 6. Efficiency Comparison.

5. Conclusion

In this work, we proposed a unified test-time compression framework for both geometry and color in 3D point clouds, leveraging priors from a pre-trained generative diffusion model. Unlike conventional methods that treat geometry and color compression as separate tasks, our approach integrates both using Point-E [32]'s upsampling model. By generating sparse 'seeds' through prompt tuning, our method enables high-fidelity decompression without dataset-specific training. Experimental results demonstrate that our framework achieves superior compression performance across diverse datasets.

Limitation

As discussed in Sec. 4.7, while our method eliminates the need for training, prompt tuning results in a longer compression time compared to existing learning-based approaches. However, it remains applicable for offline storage of 3D assets, while its efficiency may also be further improved through parallelism or mix-precision techniques. Moreover, to the best of our knowledge, as the first test-time compression framework for colored point clouds, our method offers significant potential for further exploration.

Acknowledgment. This work was supported in part by the Agency for Science, Technology and Research (A*STAR) under its MTC Programmatic Funds (Grant No. M23L7b0021), and by the Singapore Ministry of Education Tier 2 Grant (MOE-T2EP20124-0015).

References

[1] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001. 6

[2] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016. 1

[3] Chao Cao, Marius Preda, and Titus Zaharia. 3d point cloud compression: A survey. In *The 24th International Conference on 3D Web Technology*, pages 1–9, 2019. 2

[4] Alvaro Casado-Coscolla, Carlos Sanchez-Belenguer, Erik Wolfart, and Vitor Sequeira. Rendering massive indoor point clouds in virtual reality. *Virtual Reality*, 27(3):1859–1874, 2023. 1

[5] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2

[6] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 5, 6

[7] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 2

[8] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 7(8):11, 2017. 5

[9] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 4

[10] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 2

[11] Frank Galligan, Michael Hemmer, Ondrej Stava, Fan Zhang, and Jamieson Brettle. Google/draco: a library for compressing and decompressing 3d geometric meshes and point clouds, 2018. 2, 3, 5

[12] Daniel Garrido, Rui Rodrigues, A Augusto Sousa, Joao Jacob, and Daniel Castro Silva. Point cloud interaction and manipulation in virtual reality. In *2021 5th International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pages 15–20, 2021. 1

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 1

[14] Danillo Graziosi, Ohji Nakagami, Satoru Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. An overview of ongoing point cloud compression standardization activities: Video-based (v-pcc) and geometry-based (g-pcc). *APSIPA Transactions on Signal and Information Processing*, 9: e13, 2020. 2, 3, 5

[15] Yun He, Xinlin Ren, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Density-preserving deep point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4

[17] Tao Hu, Wenhang Ge, Yuyang Zhao, and Gim Hee Lee. X-ray: A sequential 3d representation for generation. *arXiv preprint arXiv:2404.14329*, 2024. 2

[18] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*, pages 890–898, 2019. 1, 2

[19] Tianxin Huang, Jiangning Zhang, Jun Chen, Zhonggan Ding, Ying Tai, Zhenyu Zhang, Chengjie Wang, and Yong Liu. 3qnet: 3d point cloud geometry quantization compression network. *ACM Transactions on Graphics (TOG)*, 41(6):1–13, 2022. 1, 2, 4, 5, 6, 7

[20] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2

[21] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 2

[22] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 2, 4

[23] 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*, 29(8):3642–3655, 2022. 5

[24] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023. 2

[25] Xiaolong Mao, Hui Yuan, Xin Lu, Raouf Hamzaoui, and Wei Gao. Pcac-gan: Asparse-tensor-based generative adversarial network for 3d point cloud attribute compression. *arXiv preprint arXiv:2407.05677*, 2024. 1, 3

[26] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 2

[27] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael

Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017. 8

[28] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*, pages 405–421. Springer, 2020. 2

[29] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia 2022 conference papers*, pages 1–8, 2022. 2

[30] Dat Thanh Nguyen and André Kaup. Lossless point cloud geometry and attribute compression using a learned conditional probability model. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(8):4337–4348, 2023. 1, 3, 5

[31] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Learning-based lossless compression of 3d point cloud geometry. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4220–4224. IEEE, 2021. 2

[32] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2, 3, 5, 7, 8

[33] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2

[34] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 4

[35] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Improved deep point cloud geometry compression. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2020. 2, 3, 5, 6

[36] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020. 8

[37] N Dinesh Reddy, Minh Vo, and Srinivasa G Narasimhan. Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1906–1915, 2018. 1

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 2, 4

[39] Michael Rudolph, Aron Riemenschneider, and Amr Rizk. Progressive coding for deep learning based point cloud attribute compression. In *Proceedings of the 16th International Workshop on Immersive Mixed and Virtual Environment Systems*, page 78–84, New York, NY, USA, 2024. Association for Computing Machinery. 1, 3, 5

[40] Michael Rudolph, Aron Riemenschneider, and Amr Rizk. Learned compression of point cloud geometry and attributes in a single model through multimodal rate-control. *arXiv preprint arXiv:2408.00599*, 2024. 1, 3

[41] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22500–22510, 2023. 2, 4

[42] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011. 2

[43] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022. 2

[44] Xihua Sheng, Li Li, Dong Liu, Zhiwei Xiong, Zhu Li, and Feng Wu. Deep-pcac: An end-to-end deep lossy compression framework for point cloud attributes. *IEEE Transactions on Multimedia*, 24:2617–2632, 2021. 1, 3

[45] Jaspreet Singh, Gurpreet Singh, Shikha Maheshwari, et al. Augmented reality technology: Current applications, challenges and its future. In *2022 4th International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 1722–1726. IEEE, 2022. 1

[46] Ryo Suzuki, Adnan Karim, Tian Xia, Hooman Hedayati, and Nicolai Marquardt. Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, pages 1–33, 2022. 1

[47] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023. 2

[48] 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. 5

[49] Dmitry Tochilkin, David Pankratz, Zexiang Liu, Zixuan Huang, Adam Letts, Yangguang Li, Ding Liang, Christian Laforte, Varun Jampani, and Yan-Pei Cao. Triposr: Fast 3d object reconstruction from a single image. *arXiv preprint arXiv:2403.02151*, 2024. 2

[50] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12619–12629, 2023. 2

[51] Jianqiang Wang and Zhan Ma. Sparse tensor-based point cloud attribute compression. In *2022 IEEE 5th International*

Conference on Multimedia Information Processing and Retrieval (MIPR), pages 59–64. IEEE, 2022. 1, 3

[52] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2021. 1, 2, 5, 6

[53] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4909–4923, 2021. 1, 2, 3, 5

[54] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multiscale representation for point cloud geometry compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):9055–9071, 2022. 1, 5

[55] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021. 2