Dual-Hilbert Scan for Efficient Video-Based Gaussian Splatting Compression

Zhiwei Zhu, Yiyi Liao, and Lu Yu
Zhejiang University, Hangzhou, China
{zhuzhiwei21, yiyi.liao, yul}@zju.edu.cn

***Abstract*—3D Gaussian Splatting (3DGS) enables real-time, high-fidelity rendering, but its large data volume poses challenges for storage and transmission. Compressing 3DGS with video codecs is promising, yet its effectiveness critically depends on the 3D-to-2D mapping. Existing approaches face a trade-off: simple scans are fast but yield poor rate–distortion (RD) performance, while optimization-based methods improve RD at high computational cost. We propose a Dual-Hilbert scan algorithm that first sorts Gaussians along a 3D Hilbert curve to preserve spatial locality, then maps them onto 2D grids via a 2D Hilbert curve, producing spatially coherent, block-like layouts suitable for video compression. A parallel GPU implementation accelerates Hilbert code generation, making the approach practical for large-scale 3DGS data. Experiments demonstrate that our method achieves RD performance comparable to state-of-the-art PLAS while being over 500× faster, effectively breaking the performance–complexity trade-off and enabling efficient, high-quality video-based 3DGS compression.**

***Index Terms*—Gaussian Splatting, Compression, Video Coding, Space-Filling Curves, Hilbert Curve.**

I. Introduction

3D Gaussian Splatting (3DGS) [1] has emerged as a revolutionary technique, enabling photorealistic rendering in real time. However, this high fidelity comes at the cost of a substantial data footprint, with typical models reaching several gigabytes. This massive storage requirement presents a significant barrier to the widespread application and deployment of 3DGS, particularly in streaming and mobile contexts. As 3DGS gains traction as a potential universal representation format, Gaussian Splatting Compression (GSC) has become a focal point of research in both academia and industry [2]–[4].

Among the various strategies, leveraging mature 2D video codecs (e.g., H.265/HEVC [5]) is particularly promising for GSC [6]–[10]. This video-based paradigm is also under active exploration by standardization bodies, notably the Moving Picture Expert Group (MPEG) within its GSC activity [11]–[13]. The approach is compelling as it harnesses decades of innovation in 2D data compression. Its effectiveness, however, critically depends on the challenge of mapping unordered 3D Gaussian primitives onto a 2D grid that a video encoder can process efficiently.

Current 3D-to-2D mapping methods for GSC present a stark trade-off between computational complexity and compression performance. On the one hand, low-complexity approaches such as raster or Morton scan [8], [12] are computationally efficient. However, they produce disorganized 2D representations with sharp discontinuities, which fundamentally limit the efficacy of video codecs. On the other hand, advanced methods like Parallel Linear Assignment Sorting (PLAS) [6] employ iterative optimization to arrange Gaussians by similarity. While this approach yields superior spatial correlation and thus rate-distortion (RD) performance, the iterative process incurs a significant computational overhead. This high complexity makes it impractical for lightweight applications where encoding speed is a critical constraint [14].

To resolve this trade-off, our work introduces a solution that achieves high RD efficiency at a speed comparable to simple scans. Leveraging the locality-preserving Hilbert curve, well-suited to the strong spatial correlation of 3DGS attributes, our approach overcomes two key challenges: the high cost of Hilbert code generation and the limitations of a simple 3D-to-1D sort for constructing a spatially coherent 2D grid. We propose a Dual-Hilbert scan algorithm to preserve locality in 2D and a fast, parallel GPU implementation to make it efficient. Our main contributions are:

- We analyze the spatial correlation in 3DGS data, providing a theoretical foundation for using the locality-preserving Hilbert curve.
- We propose a Dual-Hilbert scan that sorts Gaussians via a 3D curve and maps them onto 2D grids using a 2D curve, yielding spatially coherent block-like layouts.
- We develop a fast, parallel GPU algorithm for Hilbert code generation, making our scan process highly efficient.

II. Related Work

A. Gaussian Splatting Compression

Recent research on GSC [2], [3] has evolved along two main directions. The first, *post-compression*, compresses pre-trained vanilla 3DGS models [6], [7], [15], [16] while maintaining compatibility with standard 3DGS renderers. The second, *joint optimization*, redesigns the representation together with its compression scheme [17]–[20], reaching state-of-the-art rate–distortion performance at the cost of more complex decoders or specialized rendering pipelines.

This division is reflected in the MPEG GSC standardization activity. The activity consists of two tracks: the Inria-track (I-track), which focuses on post-compression, and the Alternate-track (A-track), which allows for joint optimization [13]. Within the I-track, current studies investigate both geometry-based [21], [22] and video-based [11], [12] approaches. Our work belongs to the video-based I-track, where efficient 3D-to-2D mapping remains a key and actively studied challenge.

B. 3D-to-2D Mapping

Mapping high-dimensional data onto a 2D grid is common in 3D compression. Surface parameterization, such as mesh UV mapping [23], flattens 3D surfaces into 2D textures, while projection-based schemes like MPEG's V-PCC [24] project dense point clouds onto planes. These approaches, however, target bounded surfaces and are ill-suited for the sparse, unbounded, volumetric nature of 3DGS.

Consequently, GSC mapping focuses on scan ordering. Simple fixed patterns such as Morton curves [8], [12] are computationally efficient but fail to preserve 3D spatial locality, resulting in incoherent 2D layouts that limit video codec performance. PLAS [6] improves upon classic grid-sorting methods (SOM [25], LAS [26]) with GPU-parallelized assignment, scaling to millions of points. Yet its iterative optimization remains a bottleneck, making it too slow for scenarios that require rapid encoding. This highlights the need for a mapping method that achieves both high compression efficiency and fast execution.

III. Proposed Method

A. Overview

To map sparse, irregular 3DGS data into dense, regular 2D video frames, we propose an efficient mapping pipeline based on Dual-Hilbert scan. As illustrated in Fig. 2, our method first sorts Gaussians along a 3D Hilbert curve, then maps them onto 2D grids via a 2D Hilbert curve. The key idea is to maintain the spatial coherence throughout the mapping process and reorganize Gaussian attributes into spatially coherent block-like 2D patches that are well suited for the predictive and transform coding mechanisms of modern video codecs.

B. Motivation: Spatial Correlation in 3DGS Data

Our approach builds on the observation that 3DGS data exhibit strong spatial correlation. To verify this, we conducted an empirical analysis by sorting the Gaussians of the MPEG-GSC *Cinema* sequence [27] along a 3D Hilbert curve and computing the autocorrelation of their standardized attribute vectors at varying lags in the resulting 1D order. As shown in Fig. 1, the autocorrelation remains high at small lags and decays rapidly with increasing distance, confirming the strong spatial correlation inherent in 3DGS data. Consequently, the Hilbert curve scan effectively groups similar Gaussians, producing a structured data stream that is well-suited for efficient video compression.

C. The Dual-Hilbert scan

Prior work [28] has demonstrated that for 3D data, a Hilbert curve preserves spatial correlation more effectively than Morton or Gray-coded curves, leading to gains in point cloud attribute coding. This established the benefit of a 3D-to-1D Hilbert sort. However, for video-based GSC, this is only a partial solution, as a naive rasterization of the 1D sequence onto a 2D grid reintroduces discontinuities that harm video codec performance.

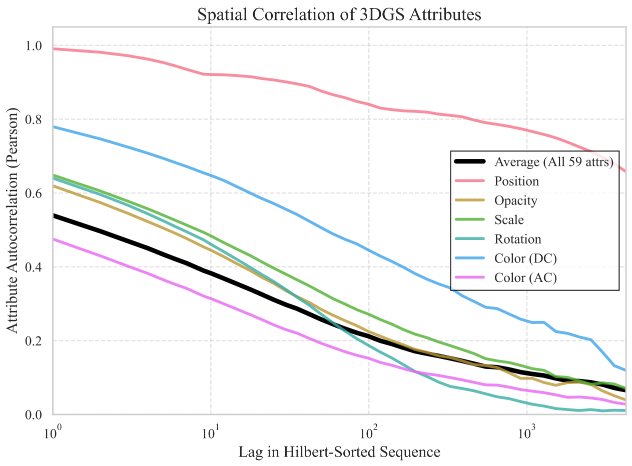

Fig. 1: Spatial correlation of 3DGS attributes. The strong correlation observed at small lags ($k < 100$) along the Hilbert-sorted sequence validates the high spatial locality of 3DGS data and motivates the use of locality-preserving scans for efficient compression.

Our core contribution is a complete two-stage locality-preserving mapping that resolves this issue. Our algorithm first linearizes the 3DGS data into a 1D 3DGS sequence and subsequently folds this sequence onto 2D 3DGS grids, with each stage employing a Hilbert curve to maintain spatial coherence. Let the initial unordered set of N Gaussians be denoted by $\mathcal{G} = \{G_i\}_{i=1}^{N}$.

1) Stage 1: 3D Hilbert Sorting (3D $\rightarrow$ 1D): This stage transforms the unordered 3D set $\mathcal{G}$ into a 1D sequence $\mathcal{S}$ where neighboring elements correspond to Gaussians that were physically close in 3D space. The process involves three steps:

1) **Quantization:** The 3D position $\mathbf{p}_i \in \mathbb{R}^3$ of each Gaussian G_i is normalized and quantized to a fixed bit-depth p, yielding an integer coordinate vector $\mathbf{p}'_i \in \{0, \dots, 2^p - 1\}^3$.
2) **Hilbert Code Generation:** We compute a unique 1D Hilbert index, or code, h_i for each quantized position $\mathbf{p}'_i$ using our fast GPU-based function: $h_i = \mathcal{H}_{3D}(\mathbf{p}'_i)$.
3) **Sorting:** The entire set of Gaussians $\mathcal{G}$ is sorted based on the ascending order of their Hilbert codes $\{h_i\}$. This produces a globally sorted 1D sequence $\mathcal{S} = (G_{\pi(1)}, G_{\pi(2)}, \dots, G_{\pi(N)})$, where π is the sorting permutation derived from $\{h_i\}$.

This process ensures that Gaussians clustered in 3D space are now adjacent in the linearized sequence $\mathcal{S}$, establishing a strong 1D locality.

2) Stage 2: 2D Hilbert Mapping (1D $\rightarrow$ 2D): The critical second stage addresses the challenge of arranging the sorted 1D sequence $\mathcal{S}$ onto a 2D grid $\mathcal{M}$ without degrading the locality achieved in Stage 1. A naive raster scan would create artificial boundaries at the end of each row, separating adjacent elements from the 1D sequence (e.g., $G_{\pi(k)}$ and $G_{\pi(k+1)}$) and making them spatially distant on the 2D grid.

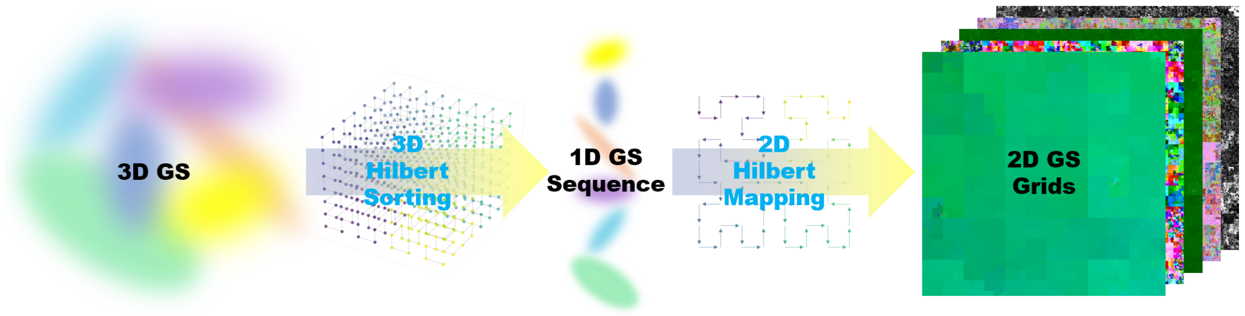

Fig. 2: Overview of our Dual-Hilbert scanning pipeline. (1) An unordered set of 3D Gaussians is sorted into a 1D sequence using a 3D Hilbert curve, preserving 3D spatial locality. (2) The attributes of this sequence are then mapped onto 2D grids using a 2D Hilbert curve, creating spatially coherent patches that are highly amenable to video compression.

To prevent this, our key innovation is to employ a second Hilbert curve to define the filling path on the 2D grid. Instead of a raster scan, we map the j-th element of the sequence $\mathcal{S}$ to the grid coordinate (u, v) that corresponds to the j-th position on a 2D Hilbert curve path. This mapping can be expressed as:

$$\mathcal{M}(\mathcal{H}_{2D}^{-1}(j)) = G_{\pi(j)}, \quad \text{for } j = 0, \dots, N-1 \tag{1}$$

where $\mathcal{H}_{2D}^{-1}(j)$ is the inverse Hilbert map that provides the 2D coordinate for the j-th point on the curve. This mapping process ensures the 1D locality is maximally preserved in the 2D domain, resulting in the formation of compact, spatially coherent blocks that are highly amenable to the predictive coding schemes used in modern video compression.

D. Fast GPU-Accelerated Hilbert Code Generation

The practical feasibility of our Dual-Hilbert scan hinges on efficient code generation. Traditional recursive Hilbert algorithms are inherently slow and ill-suited for parallel architectures. This has been a major barrier to their widespread adoption for large-scale data.

We address this critical bottleneck by developing a fully vectorized, iterative algorithm, inspired by [29]. The key insight is resolving inherent serial dependencies in the core logic by manually unrolling the inner dimensional loop (Algorithm 1). This dependency-aware vectorization enables massive parallelism on the GPU, allowing us to process hundreds of thousands of points concurrently. As a result, the entire Hilbert code generation process, for both 3D and 2D curves, is reduced to a few milliseconds, making our mapping approach fast enough for practical, near real-time applications.

Algorithm 1 Vectorized 3D Hilbert Code Generation

```
1:  procedure VectorizedHilbertCode($Coords, p$)
2:      Input: $Coords$: Tensor of shape $(N, 3)$, $p$: Order.
3:      Output: $D$: Tensor of shape $(N, )$ with Hilbert codes.
4:      $X, Y, Z \leftarrow Coords[:, 0], Coords[:, 1], Coords[:, 2]$
                                   ▷ Key: Dependency-Aware Vectorization
5:      $m \leftarrow 1 \ll (p-1)$
6:      $q \leftarrow m$
7:      while $q > 1$ do
8:          $p_{val} \leftarrow q - 1$
                                   ▷ Unroll inner loop to handle serial dependency
9:          $mask_X \leftarrow (X \ \& \ q) \neq 0$
10:         $X \leftarrow \text{where}(mask_X, X \oplus p_{val}, X)$
11:         $mask_Y \leftarrow (Y \ \& \ q) \neq 0$
12:         $t \leftarrow (X \oplus Y) \ \& \ p_{val}$
13:         $X \leftarrow \text{where}(mask_Y, X \oplus p_{val}, X \oplus t)$
14:         $Y \leftarrow \text{where}(mask_Y, Y, Y \oplus t)$
15:         $mask_Z \leftarrow (Z \ \& \ q) \neq 0$
16:         $t \leftarrow (X \oplus Z) \ \& \ p_{val}$
17:         $X \leftarrow \text{where}(mask_Z, X \oplus p_{val}, X \oplus t)$
18:         $Z \leftarrow \text{where}(mask_Z, Z, Z \oplus t)$
19:         $q \leftarrow q \gg 1$
                                   ▷ Standard Post-processing
20:     $(X, Y, Z) \leftarrow \text{GrayDecodeAndFinalize}(X, Y, Z, m)$
21:     $D \leftarrow \text{InterleaveBits}(X, Y, Z, p)$
22:     return $D$
```

IV. Experiments

A. Experimental Setup

1) Datasets and Metrics: We evaluate our method on the first frame of three sequences from the MPEG GSC test set [27]: *Bartender*, *Breakfast*, and *Cinema*. The rendered image quality is assessed against the corresponding ground-truth images. RD performance is reported using the Bjøntegaard Delta Rate (BD-Rate) [30], which quantifies the average bitrate savings relative to a reference anchor.

2) Baselines: We compare our method against representative 3D-to-2D mapping baselines:

- **Anchor**: No spatial sorting; 2D raster scan.
- **Morton**: 3D Morton sorting followed by a 2D raster scan.
- **PLAS** [6]: The default optimization-based configuration that balances performance and speed by excluding SH AC components from joint sorting.
- **PLAS++**: A variant of PLAS that includes all attributes in joint optimization, serving as an upper bound on performance at higher computational cost.
- **Dual-Morton**: An ablation of our method that replaces both the 3D and 2D Hilbert curves with Morton curves.

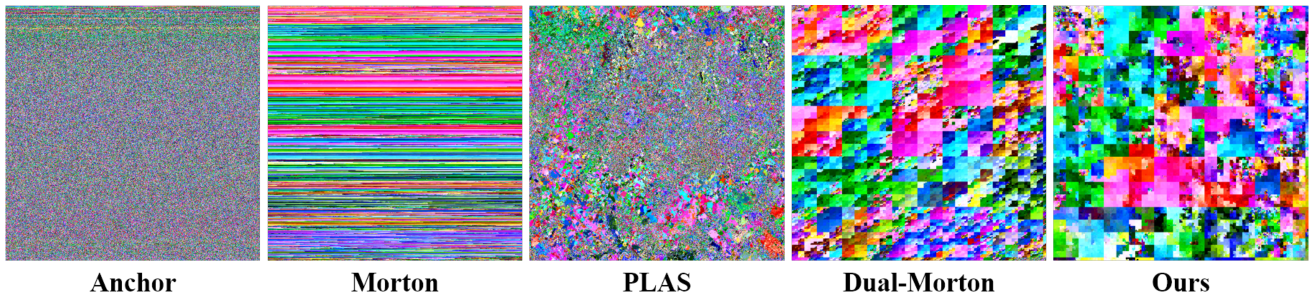

Fig. 3: Visual comparison of scan orders for mapped positions (low 8-bit precision).

TABLE I: BD-Rate (%) and Sorting Time (s) Comparison Across Datasets. Results are highlighted: 1st , 2nd , 3rd .

Method	**Bartender**				**Breakfast**				**Cinema**			
	PSNR	SSIM	LPIPS	Time(s)	PSNR	SSIM	LPIPS	Time(s)	PSNR	SSIM	LPIPS	Time(s)
Morton	-18.55	-20.88	-19.34	0.015	-10.89	-15.70	-16.89	0.014	-21.58	-20.10	-19.26	0.013
PLAS	-13.93	-16.66	-16.30	12.457	-14.49	-18.48	-18.30	10.576	-16.35	-16.75	-16.04	8.093
PLAS++	-24.89	-26.70	-24.48	30.541	-25.48	-29.33	-27.69	26.079	-28.63	-27.89	-26.62	18.643
Dual-Morton	-21.10	-23.28	-20.52	0.039	-8.87	-16.44	-19.63	0.037	-25.73	-24.17	-22.61	0.036
Ours	-21.28	-24.59	-22.80	0.062	-11.60	-19.72	-21.50	0.061	-26.66	-25.56	-23.76	0.059

3) Implementation Details: All experiments used standardized settings to isolate scan-order effects. Positions were quantized to 16 bits and coded losslessly, while SH, opacity, scale, and rotation were quantized to 10 bits and coded lossily. The resulting 2D grids were compressed with HEVC (HM v18.0 [31]) at multiple rate points.

B. Results and Analysis

1) Visual Analysis of Scan Orders: Fig. 3 visualizes the 2D mappings of Gaussian positions under different scan orders. The Anchor method appears noisy, while Morton introduces line-like artifacts. PLAS yields coherent but irregular patches, and Dual-Morton improves over Morton yet exhibits discontinuous Z-shaped patterns. In contrast, our Dual-Hilbert method produces spatially coherent, block-like structures that align well with block-based video codec processing, providing an ideal input for compression.

2) Rate-Distortion Performance: Table I summarizes the RD results. PLAS++ achieves the highest compression but at a prohibitive computational cost. Our Dual-Hilbert method consistently ranks second, offering the best performance among fast, non-iterative approaches. Notably, it also outperforms its direct counterpart, Dual-Morton, highlighting the superior locality-preserving properties of the Hilbert curve. Fig. 4 further illustrates its advantage over all except the most computationally intensive baseline.

3) Complexity Analysis: While PLAS++ achieves strong RD performance, it requires significant computation (19–31s per frame). In contrast, our Dual-Hilbert completes the entire 3D–to-2D mapping in only 0.06s, over 500× faster than PLAS++, with only minor overhead versus Dual-Morton (0.04s). This efficiency comes from our optimized GPU-based Hilbert code generation, which outperforms standard implementations [32] by orders of magnitude (Table II).

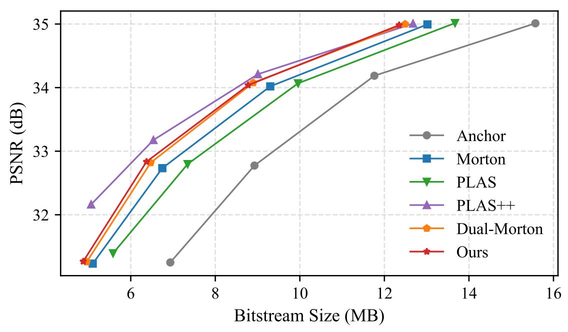

Fig. 4: RD performance on PSNR for the *Cinema* sequence.

TABLE II: Execution time (s) of 3D Hilbert code generation (bit-depth=16) for varying point counts. `np1/np4/np8` denote CPU threads. Results are highlighted: 1st , 2nd .

Method	0.5M	1M	1.5M	2M	2.5M	3M
Hilbert_lib_np1	12.930	29.240	43.224	56.778	69.762	79.309
Hilbert_lib_np4	5.285	11.963	17.488	22.791	26.197	31.069
Hilbert_lib_np8	4.011	8.678	13.105	17.051	19.879	24.406
Ours_cpu	0.057	0.081	0.200	0.265	0.348	0.397
Ours_gpu	0.008	0.018	0.025	0.033	0.048	0.056

V. Conclusion

This paper introduced a Dual-Hilbert scan algorithm for video-based Gaussian Splatting Compression (GSC). By preserving spatial locality during both 3D-to-1D sorting and 1D-to-2D mapping, the method produces spatially coherent, block-like layouts well-suited for video codecs. A parallel GPU implementation enables practical, large-scale deployment. Experiments show that it achieves RD performance comparable to PLAS while being over 500× faster, effectively overcoming the performance–complexity trade-off.

References

[1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023.

[2] M. T. Bagdasarian, P. Knoll, Y.-H. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern, "3dgs. zip: A survey on 3d gaussian splatting compression methods," *arXiv preprint arXiv:2407.09510*, 2024.

[3] M. S. Ali, C. Zhang, M. Cagnazzo, G. Valenzise, E. Tartaglione, and S.-H. Bae, "Compression in 3d gaussian splatting: A survey of methods, trends, and future directions," *arXiv preprint arXiv:2502.19457*, 2025.

[4] WG 02 MPEG Technical Requirements, "Draft gaussian splat coding requirements," ISO/IEC JTC1/SC29/WG02 MPEG Technical Requirements, Daejeon, Working Draft MDS25533_WG02_N00473, Jul. 2025, meeting 151, Meeting July 5–July 12, 2025.

[5] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on circuits and systems for video technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[6] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, "Compact 3d scene representation via self-organizing gaussian grids," in *European Conference on Computer Vision*. Springer, 2024, pp. 18–34.

[7] V. Ye, R. Li, J. Kerr, M. Turkulainen, B. Yi, Z. Pan, O. Seiskari, J. Ye, J. Hu, M. Tancik, and A. Kanazawa, "gsplat: An open-source library for Gaussian splatting," *arXiv preprint arXiv:2409.06765*, 2024. [Online]. Available: https://arxiv.org/abs/2409.06765

[8] P. Wang, Z. Zhang, L. Wang, K. Yao, S. Xie, J. Yu, M. Wu, and L. Xu, "V^ 3: Viewing volumetric videos on mobiles via streamable 2d dynamic gaussians," *ACM Transactions on Graphics (TOG)*, vol. 43, no. 6, pp. 1–13, 2024.

[9] S. Lee, F. Shu, Y. Sanchez, T. Schierl, and C. Hellge, "Compression of 3d gaussian splatting with optimized feature planes and standard video codecs," *arXiv preprint arXiv:2501.03399*, 2025.

[10] N. Labs, "SPZ: A File Format for 3D Gaussian Splats," open-source library and file format. GitHub repository. [Online]. Available: https://github.com/nianticlabs/spz

[11] S. Li, Y. Liao, and L. Yu, "[gsc][jee6.2-related] a potential video-based anchor for gaussian splats coding," ISO/IEC JTC1/SC29/WG04 MPEG Video Coding, online, Explorations Document m72063, Mar. 2025, meeting 150, Meeting March 20–March 26, 2025.

[12] W. Wang, Y. Xu, K. Zhang, and L. Zhang, "[gsc][jee6.7-related] exploration of v3c support for gsc," ISO/IEC JTC1/SC29/WG07 MPEG 3D Graphics Coding and Haptics Coding, Daejeon, Explorations Document m73323, Jun. 2025, meeting 151, Meeting June 23–July 1, 2025, submitted by Bytedance Inc.

[13] WG 07 MPEG 3D Graphics Coding and Haptics Coding, "Draft ctc for gaussian splat coding," ISO/IEC JTC1/SC29/WG07 MPEG 3D Graphics Coding and Haptics Coding, Daejeon, Explorations Document MDS25512_WG07_N01292, Jul. 2025, meeting 151, Meeting July 5–July 22, 2025.

[14] WG 02 MPEG Technical Requirements, "Preliminary draft requirements for lightweight gsc," ISO/IEC JTC1/SC29/WG02 MPEG Technical Requirements, Daejeon, Explorations Document MDS25531_WG02_N00471, Jul. 2025, meeting 151, Meeting July 5–July 10, 2025.

[15] S. Niedermayr, J. Stumpfegger, and R. Westermann, "Compressed 3d gaussian splatting for accelerated novel view synthesis," *arXiv preprint arXiv:2401.02436*, 2024.

[16] Q. Yang, L. Yang, G. Van Der Auwera, and Z. Li, "Hybridgs: High-efficiency gaussian splatting data compression using dual-channel sparse representation and point cloud encoder," *arXiv preprint arXiv:2505.01938*, 2025.

[17] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, "Scaffold-gs: Structured 3d gaussians for view-adaptive rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 654–20 664.

[18] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "Hac: Hash-grid assisted context for 3d gaussian splatting compression," in *European Conference on Computer Vision*. Springer, 2024.

[19] Y. Wang, Z. Li, L. Guo, W. Yang, A. C. Kot, and B. Wen, "Contextgs: Compact 3d gaussian splatting with anchor level context model," *arXiv preprint arXiv:2405.20721*, 2024.

[20] Y.-T. Zhan, C.-Y. Ho, H. Yang, Y.-H. Chen, J. C. Chiang, Y.-L. Liu, and W.-H. Peng, "Cat-3dgs: A context-adaptive triplane approach to rate-distortion-optimized 3dgs compression," *arXiv preprint arXiv:2503.00357*, 2025.

[21] G. Sandri, F. Thudor, N. Sabater, and B. Chupeau, "[jee6] ges-tm as anchor for 3d gaussian coding," ISO/IEC JTC1/SC29/WG07 MPEG 3D Graphics Coding and Haptics Coding, Kemer, Explorations Document m69429, Sep. 2024, meeting 148, September 16–September 19, 2024, submitted by InterDigital.

[22] D. Fujii, H. Yoshida, and K. Nonaka, "[gsc][jee6.2] report on g-pcc anchor generation," ISO/IEC JTC1/SC29/WG07 MPEG 3D Graphics Coding and Haptics Coding, Daejeon, Explorations Document m72849, Apr. 2025, meeting 151, April 30–June 6, 2025.

[23] E. E. Catmull, *A subdivision algorithm for computer display of curved surfaces*. The University of Utah, 1974.

[24] D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai, "An overview of ongoing point cloud compression standardization activities: Video-based (v-pcc) and geometry-based (g-pcc)," *APSIPA Transactions on Signal and Information Processing*, vol. 9, p. e13, 2020.

[25] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[26] K. U. Barthel, N. Hezel, K. Jung, and K. Schall, "Improved evaluation and generation of grid layouts using distance preservation quality and linear assignment sorting," in *Computer graphics forum*, vol. 42, no. 1. Wiley Online Library, 2023, pp. 261–276.

[27] J. Y. Jeong, R. Koo, K.-J. Oh, H.-C. Shin, and G. Lee, "[gsc][jee6.1-related] training method for generating temporally consistent per-frame i-3dgs dataset," ISO/IEC JTC1/SC29/WG04 MPEG Video Coding, online, Explorations Document m71763, Feb. 2025, meeting 150, Meeting February 17–February 24, 2025.

[28] J. Chen, L. Yu, and W. Wang, "Hilbert space filling curve based scan-order for point cloud attribute compression," *IEEE Transactions on Image Processing*, vol. 31, pp. 4609–4621, 2022.

[29] A. R. Butz, "Alternative algorithm for hilbert's space-filling curve," *IEEE Transactions on Computers*, vol. 100, no. 4, pp. 424–426, 2006.

[30] G. Bjontegaard, "Calculation of Average PSNR Differences between RD-Curves," *ITU-T SG16, Doc. VCEG-M33*, 2001.

[31] J. V. E. T. (JVET), "Hm-18.0: Hevc reference software," 2023. [Online]. Available: https://vcgit.hhi.fraunhofer.de/jvet/HM/-/tree/HM-18.0

[32] G. Tay, "hilbertcurve: Convert between 1-D distance and N-D points along a Hilbert curve," python package on PyPI. [Online]. Available: https://pypi.org/project/hilbertcurve/