TinySplat: Feedforward Approach for Generating Compact 3D Scene Representation

Zetian Song, Jiaye Fu, Jiaqi Zhang, Xiaohan Lu,
Chuanmin Jia, *Member, IEEE,* Siwei Ma, *Fellow, IEEE,* Wen Gao, *Fellow, IEEE,*

***Abstract*—The recent development of feedforward 3D Gaussian Splatting (3DGS) presents a new paradigm to reconstruct 3D scenes. Using neural networks trained on large-scale multi-view datasets, it can directly infer 3DGS representations from sparse input views. Although the feedforward approach achieves high reconstruction speed, it still suffers from the substantial storage cost of 3D Gaussians. Existing 3DGS compression methods relying on scene-wise optimization are not applicable due to architectural incompatibilities. To overcome this limitation, we propose TinySplat, a complete feedforward approach for generating compact 3D scene representations. Built upon standard feedforward 3DGS methods, TinySplat integrates a training-free compression framework that systematically eliminates key sources of redundancy. Specifically, we introduce View-Projection Transformation (VPT) to reduce geometric redundancy by projecting geometric parameters into a more compact space. We further present Visibility-Aware Basis Reduction (VABR), which mitigates perceptual redundancy by aligning feature energy along dominant viewing directions via basis transformation. Lastly, spatial redundancy is addressed through an off-the-shelf video codec. Comprehensive experimental results on multiple benchmark datasets demonstrate that TinySplat achieves over 100× compression for 3D Gaussian data generated by feedforward methods. Compared to the state-of-the-art compression approach, we achieve comparable quality with only 6% of the storage size. Meanwhile, our compression framework requires only 25% of the encoding time and 1% of the decoding time.**

***Index Terms*—3D Gaussian Splatting, Data Compression, Novel View Synthesis, Feedforward 3DGS.**

I. Introduction

THE 3D Gaussian Splatting (3DGS) has emerged as a powerful technique for reconstructing 3D scenes from multi-view images. By explicitly representing a scene as a collection of anisotropic Gaussian primitives, 3DGS enables photorealistic rendering and supports real-time free-viewpoint navigation. The vanilla 3DGS formulation relies on stochastic gradient descent (SGD) and adaptive density control to jointly optimize the geometric and appearance attributes of the Gaussian primitives. While this approach achieves impressive visual fidelity, its reliance on per-scene optimization and dense input views significantly limits its scalability and practicality.

Recent feedforward 3DGS utilizes Neural Network (NN) to directly infer the parameters of 3D Gaussian primitives from input images, thereby eliminating the need for iterative optimization. They offer substantial improvements in reconstruction speed and are capable of handling sparse view inputs, making them attractive for time-sensitive applications such as AR/VR and mobile 3D capture. However, the efficiency comes at the cost of significantly increased data volume. 3DGS requires numerous Gaussian primitives to model a specific scene, resulting in substantial memory and storage demands. This expansion poses major challenges for data transmission and interactive rendering, highlighting the need for effective compression techniques specialized for the unique characteristics of 3DGS data.

Prior research has explored optimization-based strategies for compressing 3DGS models, such as pruning [1]–[5], clustering [6], motion estimation [7], [8], and probabilistic modeling [9], [10]. Notably, HAC [9] employs multi-resolution hash grid priors for compact encoding, while ContextGS [10] enhances compression efficiency through hierarchical context modeling. Nevertheless, due to their reliance on iterative optimization, these methods inherit the same limitations as vanilla 3DGS and are thus unsuitable for feedforward methods.

To overcome these limitations, FCGS [11] introduced an optimization-free compression framework applicable to arbitrary 3D Gaussian models. By integrating hash grid-based hyperpriors and context priors through a multi-path entropy module, FCGS achieves notable compression ratios across diverse datasets. However, its effectiveness for feedforward-generated Gaussian models remains limited. First, FCGS does not fully exploit the inherent spatial redundancy derived from feedforward geometric generation. Additionally, the uniform treatment of feature channels hinders effective utilization of perceptual redundancy present in appearance features. Furthermore, reliance on multiple NN-based prior models introduces computational overhead unsuitable for real-time or edge-based scenarios.

In this paper, we propose **TinySplat**, a fully training-free approach for generating compact 3D Gaussian scene representations directly from multi-view images. TinySplat effectively addresses existing limitations by coupling a feedforward 3DGS generation stage with a rendering-aware compression stage. Specifically, in the Gaussian generation stage, we employ existing feedforward methods, such as DepthSplat [12], to produce Gaussian feature maps, where each element defines parameters of individual Gaussian primitives. Subsequently,

Zetian Song, Jiaqi Zhang, Xiaohan Lu, Siwei Ma and Wen Gao are with the State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University, Beijing, 100871, China (e-mail: songzt@pku.edu.cn, jqzhang@pku.edu.cn, luxiaohan@stu.pku.edu.cn, swma@pku.edu.cn, wgao@pku.edu.cn).

Jiaye Fu is with the School of Electronic and Computer Engineering, Peking University, Shenzhen, 518055, China, and also with the State Key Laboratory of Multimedia Information Processing, School of Computer Science, Peking University, Beijing, 100871, China (email: jyfu@stu.pku.edu.cn).

Chuanmin Jia is with the Wangxuan Institute of Computer Technology, State Key Laboratory of Multimedia Information Processing, Peking University, Beijing 100871, China (email: cmjia@pku.edu.cn).

we introduce a novel compression framework designed to leverage spatial and perceptual redundancies within these feature maps for efficient storage and transmission.

In the compression framework, we propose *View-Projection Transform* (**VPT**) to address structural redundancy from geometric generation. By exploiting the pixel-aligned characteristics of feedforward 3DGS, VPT reveals a more regular spatial layout, significantly enhancing local correlation and enabling more effective compression. Additionally, we propose the *Visibility-Aware Basis Reduction* (**VABR**) method to handle perceptual redundancy in color attributes. Leveraging the anisotropic properties of spherical harmonic (SH) functions, VABR selectively retains perceptually dominant components while suppressing negligible ones, thereby improving compression efficiency without compromising rendering fidelity. Lastly, given the resemblance between Gaussian feature maps and traditional 2D images, we further exploit spatial redundancies using an off-the-shelf video codec.

Our main contributions are summarized as follows:

- We introduce TinySplat, a fully feedforward pipeline consisting of Gaussian generation and a novel compression stage, enabling compact 3D Gaussian scene representations without scene-dependent optimization.
- We develop the VPT module, applying a coordinate-space transformation that exploits characteristics specific to feedforward geometric inference, thereby reducing structural redundancy and enhancing reconstruction quality.
- We propose the VABR module, which utilizes the visibility of SH bases to selectively preserve the most perceptually informative radiance components under constrained viewing directions, enabling highly compact color representations.
- Extensive experiments demonstrate that TinySplat can achieve comparable quality to DepthSplat with only 1% of storage. In comparison to the state-of-the-art (SoTA) compression method, TinySplat achieves a 90% reduction in storage and a 75% reduction in encoding time, simultaneously delivering superior visual fidelity..

The remainder of this paper is organized as follows. Section II provides an overview of related works on 3DGS generation and compression. Section III describes the proposed TinySplat in detail. Section IV presents experimental evaluations and demonstrates the advantages of TinySplat. Section V outlines limitations and future research directions. Finally, Section VI concludes the paper.

II. Related Works

A. Novel View Synthesis via Per-scene Optimization

Novel-view synthesis is a pivotal task in computer vision and graphics, aiming to generate photorealistic views from captured multi-view images. Recently, per-scene optimization methods have achieved remarkable progress by formulating the 3D representation as learnable parameters optimized using SGD.

As a seminal work in this field, the Neural Radiance Field (NeRF) [13] proposed by Mildenhall *et al.* uses a deep multi-layer perceptron to learn color and opacity mappings. Novel view rendering is then performed through the volumetric rendering pipeline. Muller *et al.* further propose INGP [14], which models the scene using multi-resolution hash grids. By querying the explicit 3D structure, INGP can significantly reduce the computational cost of both training and inference. Researchers have also explored various explicit data structures for scene representation, including voxel grids [15], [16], multiple tensor planes [17], [18], and octrees [19]. Based on NeRFs, Kerbl *et al.* propose 3DGS [20], which replaces volumetric rendering with a hardware-friendly rasterization technique. 3DGS enables real-time rendering on consumer devices and has inspired numerous follow-up works targeting different application domains [21]–[24].

B. Sparse View 3D Reconstruction

Acquiring dozens of input views is usually impractical in real-world applications. As a result, researchers have investigated methods for 3D reconstruction from sparse input views. Several approaches [25]–[29] improve the 3DGS optimization process by introducing specialized regularization, such as view consistency and depth normalization. Meanwhile, some methods construct 3D scene representations using principles from multi-view stereo [30]–[35]. They utilize techniques like epipolar geometry and cost volume aggregation. Some methods also leverage Vision Transformer architectures to effectively fuse features across different views [36]. Further research also employs structural priors derived from large-scale diffusion models to maintain consistency between multiple viewpoints, enhancing the quality of novel view synthesis [37]–[42].

C. feedforward 3D Gaussian Splatting

Vanilla 3DGS [20] requires per-scene optimization, making it computationally intensive and time-consuming to obtain explicit scene representations. Recently, numerous feedforward 3DGS methods have been proposed, targeting 3D objects [43]–[47] or entire scenes [12], [30], [33], [36], [42], [48]–[51]. The feedforward methods focus on fast reconstruction of 3D scenes from sparse input views. Specifically, these methods typically comprise a depth estimation module and a Gaussian attribute synthesis module.

For single object reconstruction, GPS-Gaussian [46] and GPS-Gaussian+ [45] focus on 3D Gaussian reconstruction of the human body. Leveraging strong structural priors of human anatomy, these methods can generate a high-quality human body model from only a few input views. Szymanowicz *et al.* propose Splatter Image [43], which maps each input pixel to a 3D Gaussian using a simple yet efficient network, achieving real-time performance at 38 FPS for forward reconstruction. GRM [47] further introduces a transformer-based architecture that effectively fuses multi-view information to achieve better reconstruction quality.

In contrast, full-scene reconstruction poses greater challenges than single-object reconstruction due to increased scene complexity and lack of reliable structural priors. To address these challenges, Charatan *et al.* propose PixelSplat [30], which leverages deep learning to predict dense probability distributions in 3D space and samples Gaussian centers accordingly to enable fast scene reconstruction. Chen *et al.* introduce MVSplat [35], which generates multiple depth maps per view and computes cross-view confidence scores to achieve more accurate geometric reconstruction. Zhang et al. propose the Gaussian Graph Network (GGN) [52] to improve reconstruction quality with abundant input views. By constructing a graph structure, GGN enables each Gaussian primitive to aggregate features from multiple viewpoints, which significantly reduces the number of Gaussian primitives and effectively suppresses artifacts. Xu *et al.* further propose DepthSplat [12], leveraging pre-trained monocular depth features to improve both depth prediction and reconstruction quality, which achieves SoTA performance.

D. 3D Gaussian Splatting compression

The substantial computational cost of generating 3D Gaussian models highlights the need for efficient storage and transmission, making compression a key challenge in 3DGS research. Researchers have proposed numerous training-based methods to compress 3D Gaussian structures and reduce their storage and transmission costs.

Some works create sparser representations by pruning less significant Gaussian primitives during training [1]–[5]. Vector quantization is also widely employed in 3D Gaussian compression [1], [2], [4], [5], [53], where a learnable codebook is used to quantize high-dimensional feature representations, achieving efficient compression. Some other methods map 3D Gaussian primitives onto feature planes [54], [55] to compress with conventional video codecs. The feature planes require end-to-end optimization to obtain more compressible representations. Further research focuses on generating more compact Gaussian models by meticulously designing 3D representations. Lu *et al.* propose Scaffold-GS [6], clustering Gaussian primitives into anchor points with neural features. More recent approaches utilize structural priors such as hash grids and context models to exploit the redundancies among 3DGS primitives, achieving about 100× compression compared to the vanilla 3DGS [9], [10], [56].

However, the aforementioned methods require an optimization process for each specific scene. They also require dense multi-view images as input. Both characteristics limit their applicability. To address this, Chen *et al.* propose FCGS [11], a general-purpose feedforward compression framework. By incorporating multiple priors into the Gaussian mixture modeling, FCGS achieves efficient compression for arbitrary 3D Gaussian data, even outperforming many scene-specific optimization methods. Nevertheless, the generality of FCGS limits its ability to exploit the unique characteristics of different types of 3D Gaussian data, leaving room for further improvement in compression efficiency.

III. Method

A. Preliminary

The widely recognized 3D Gaussian splatting employs a collection of Gaussian primitives to model 3D scenes. Each primitive consists of geometric and SH parameters that define its shape and radiance. Specifically, the geometric properties of Gaussian primitives are characterized by the Gaussian probability density function,

$$\mathcal{G}(\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{3}{2}}|\boldsymbol{\Sigma}|^{\frac{1}{2}}} exp(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})), \quad (1)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{x}$ denote the Cartesian coordinates of the Gaussian center and the sample point, respectively. $\boldsymbol{\Sigma}$ is the 3D covariance matrix and $|\boldsymbol{\Sigma}|$ denotes its determinant. In the vanilla 3DGS, the covariance matrix is further decomposed into rotation matrix $\boldsymbol{R}$ and scaling matrix $\boldsymbol{S}$,

$$\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^T\boldsymbol{R}^T, \quad (2)$$

where the diagonal scaling matrix $\boldsymbol{S}$ is stored as a vector $\boldsymbol{s} \in \mathbb{R}^3$ and $\boldsymbol{R}$ is stored as quaternion $\boldsymbol{q} \in \mathbb{R}^4$ that represents rotation.

In the rendering process, each Gaussian primitive is projected into the camera space and integrated to produce a 2D density function, which is combined with the opacity parameter σ to compute pixel-wise opacity. Meanwhile, the color value is obtained by querying SH functions with the view direction.

To reconstruct a specific scene, the vanilla 3DGS approach first generates an initial set of Gaussian primitives via structure-from-motion techniques. Subsequently, this initial model undergoes optimization to match input ground-truth images, including adaptive densification guided by gradients to effectively capture fine geometric and textural details.

Despite delivering high-quality reconstructions, this optimization-based approach suffers from slow processing speeds. To mitigate this limitation, recent studies propose eliminating the computationally intensive gradient descent step by introducing deep NNs enhanced with cross-view attention mechanisms. These networks trained on extensive multi-view datasets can directly produce accurate 3D scene representations through feedforward inference. To facilitate NN processing, this approach typically generates Gaussian feature maps of the same resolution as the input images, where each location stores all geometric and color attributes of a single Gaussian. Formally, the Gaussian inference process can be expressed as,

$$\begin{aligned} \mathcal{F} : \{(\boldsymbol{I}^v, \boldsymbol{K}^v, \boldsymbol{E}^v)\}_{v=1}^{V} \Rightarrow \\ \{\cup(\sigma_i^v, \boldsymbol{\mu}_i^v, \boldsymbol{q}_i^v, \boldsymbol{s}_i^v, \boldsymbol{SH}_i^v)\}_{i=1,\dots,H\times W}^{v=1,\dots,V}, \end{aligned} \quad (3)$$

where $\boldsymbol{I}$ denotes the input image, $\boldsymbol{K}$, $\boldsymbol{E}$ denote the intrinsic and extrinsic parameters corresponding to the input image, respectively. V denotes the number of viewpoints.

B. Proposed Pipeline

The overall pipeline of the proposed approach is illustrated in Fig. 1. Initially, we employ pre-trained feedforward 3D Gaussian inference networks [12], [35] to generate a set of 3D Gaussian primitives representing the target scene. For each input view, the network produces Gaussian primitives corresponding directly to image pixels, with each primitive

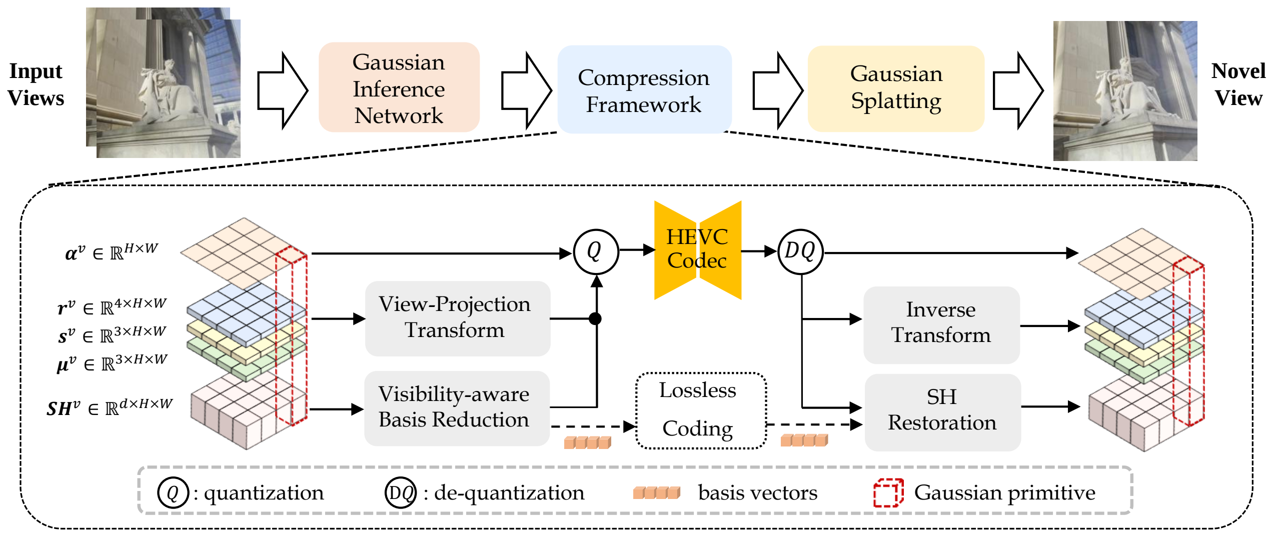
Input Views
Gaussian Inference Network
Compression Framework
Gaussian Splatting
Novel View
$\alpha^v \in \mathbb{R}^{H\times W}$
$r^v \in \mathbb{R}^{4\times H\times W}$
$s^v \in \mathbb{R}^{3\times H\times W}$
$\mu^v \in \mathbb{R}^{3\times H\times W}$
$SH^v \in \mathbb{R}^{d\times H\times W}$
View-Projection Transform
Visibility-aware Basis Reduction
Q
HEVC Codec
DQ
Lossless Coding
Inverse Transform
SH Restoration
Q : quantization
DQ : de-quantization
basis vectors
Gaussian primitive

Fig. 1. The overall framework of TinySplat. We generate Gaussian feature maps from existing Gaussian inference networks in a feedforward manner and compress them with a meticulously designed compression framework. The symbols $\boldsymbol{\alpha}^v, \boldsymbol{r}^v, \boldsymbol{s}^v, \boldsymbol{\mu}^v, \boldsymbol{SH}^v$ denote the opacity, rotation, scale, mean position, and SH maps associated with view v, respectively. These parameters jointly define the geometry and appearance of the Gaussian primitives. In our compression framework, we first apply the VPT and VABR to reduce cross-channel redundancy. Subsequently, all features are quantized into 14-bit integers, and each feature plane is independently encoded as a grayscale image using the HEVC codec. On the decoder side, we perform dequantization and apply the inverse transforms to reconstruct the original Gaussian feature maps. Finally, we merge Gaussian maps from different views to form the complete 3D scene representation and render novel views.

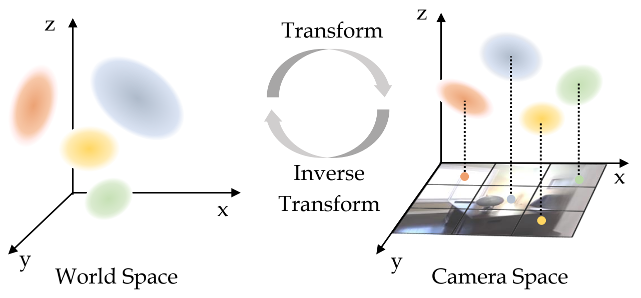

Fig. 2. Illustration of the proposed VPT. We project the feedforward-generated Gaussian primitives from world space into the corresponding input camera space. We perform compression in camera space, reduce inter-channel correlation of geometric parameters for better efficiency. During decoding, the geometry is transformed back into world space to reconstruct the 3D scene.

consisting of geometric and color features as described in Eq. 3. These features collectively form per-view feature maps.

Compared to the original images, the size of this 3D Gaussian representation increases by more than two orders of magnitude. However, according to information theory principles, applying deterministic transformations does not inherently amplify the information content of the source data. Therefore, the generated 3D Gaussian model contains substantial redundancy, while existing compression methods for 3DGS relying on optimization are incompatible with the feedforward inference pipeline. To address this, we introduce a training-free compression framework designed to reduce both spatial and perceptual redundancies, producing a significantly more compact representation of the 3D scene.

Our compression framework comprises two lightweight yet highly effective modules, VPT and VABR. These modules specifically address the challenges posed by the absence of end-to-end optimization. First, high-dimensional Gaussian primitives inherently contain extensive redundancies. Second, without supervision from explicit distortion metrics, quantifying how deviations in parameters affect rendering quality becomes challenging.

The VPT targets geometric compression based on the following key observations. First, rendering occurs in camera space, making distortion measurements in this domain more directly representative of rendering quality impacts. Second, Gaussian positions derived via back-projection exhibit stronger spatial correlations in camera space. VPT leverages these properties by performing a reversible transformation of geometric features from world space into camera space, facilitating more efficient compression.

The VABR module addresses perceptual redundancy in appearance parameters by leveraging the anisotropic properties of SH functions. By assigning visibility-aware importance weights to SH coefficients and analyzing SH distributions across the whole scene, VABR derives perceptually consistent basis functions. This adaptive approach selectively retains color features that significantly contribute to dominant viewing directions, enabling efficient yet visually faithful compression.

Subsequently, we pre-quantize the transformed features into a 14-bit integer format based on their standard deviations. These quantized feature maps are then encoded as grayscale images using an off-the-shelf HEVC codec to further reduce spatial redundancy. Additionally, metadata required for reconstruction, including camera parameters for the inverse VPT, basis vectors for inverse VABR, and other dequantization-related information, is losslessly encoded and transmitted to

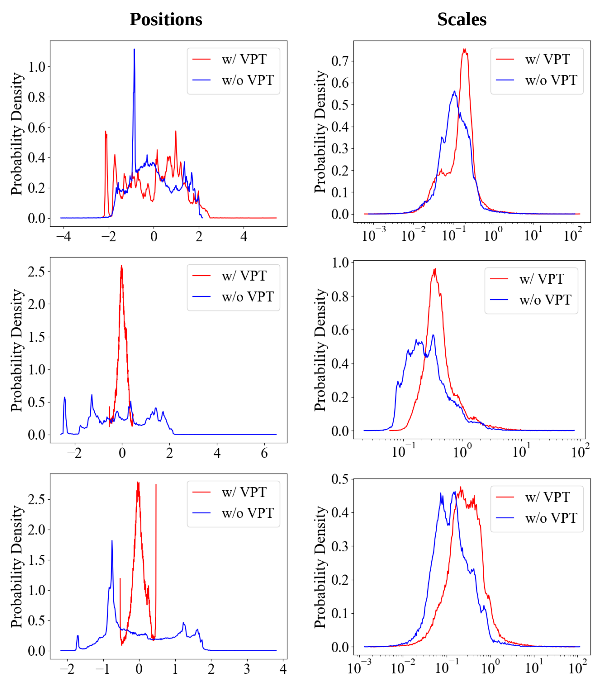

Fig. 3. Statistical distributions of positions and scaling factors before and after VPT. After applying VPT, most of the positional energy is concentrated in the depth channel (top row), and the distribution of scaling factors becomes more compact.

the decoder.

On the decoder side, we first decode the feature maps using an HEVC decoder, followed by dequantization to recover the floating-point features. The reconstructed geometric parameters are then transformed back to the original world coordinate system through inverse VPT. Concurrently, the color feature coefficients are restored to their original SH representation using the received basis vectors. Finally, by integrating these decoded attributes, we reconstruct the complete 3D Gaussian representation, enabling the rendering of novel views using a standard 3DGS renderer.

C. View-projection Transform

The Gaussian models generated by the inference network typically exhibit pixel-wise alignment with the input views, distinguishing them from conventional 3D Gaussian representations. This characteristic introduces a highly structured spatial arrangement into the generated models. Specifically, each Gaussian center is computed by back-projection, resulting in an ordered distribution of 3D positions. Moreover, the projected areas of these Gaussian ellipsoids remain relatively uniform across the input view, causing the scale parameters to correlate strongly with their distances to the camera. This reflects a systematic depth-dependent scaling pattern.

Based on the above analysis, we propose to apply the VPT to the geometry parameters of Gaussian primitives, conditioned upon input camera parameters. A conceptual illustration of the VPT is shown in Fig. 2, where we transform the Gaussian geometric parameters into the corresponding camera space. Specifically, we first apply the view transformation to the position of Gaussian centers,

$$z_{i,j}^v(x_{i,j}^v, y_{i,j}^v, 1)^T = \boldsymbol{K}^v \cdot (\boldsymbol{R}^v \cdot \boldsymbol{\mu}_{i,j}^v + \boldsymbol{T}^v), \tag{4}$$

where $\boldsymbol{K}^v$ denotes the intrinsic matrix of the input view v, while $\boldsymbol{R}^v$ and $\boldsymbol{T}^v$ are the rotation matrix and translation vector derived from the extrinsic parameters. The vector $\boldsymbol{\mu}_{i,j}^v$ represents the Gaussian center in world coordinates. $z_{i,j}^v$, $x_{i,j}^v$, and $y_{i,j}^v$ represent the transformed position. In particular, $z_{i,j}^v$ corresponds to the depth map, while $x_{i,j}^v$ and $y_{i,j}^v$ represent normalized 2D coordinates in the image plane. Given the pixel-aligned nature of feedforward-generated Gaussians, coordinates $(x_{i,j}^v, y_{i,j}^v)$ closely cluster around their corresponding pixel centers $(\frac{i}{H}, \frac{j}{W})$, with H and W being the height and width of the input images, respectively. Using this regularity, we encode only the offsets relative to pixel centers to enhance compression efficiency.

After transforming center positions into camera space, we approximate transformations for the Gaussian shape parameters. Since ellipsoids do not maintain exact geometric forms under perspective projection, we perform the following approximations,

$$\hat{\boldsymbol{q}}_{i,j}^v = Quat(\boldsymbol{R}^v) \cdot \boldsymbol{q}_{i,j}^v, \tag{5}$$

$$\hat{\boldsymbol{s}}_{i,j}^v = f\frac{\boldsymbol{s}_{i,j}^v}{z_{i,j}^v}, \tag{6}$$

where $\hat{\boldsymbol{q}}_{i,j}^v$ and $\boldsymbol{q}_{i,j}^v$ denotes the rotation quaternions before and after the transformation, respectively, and $\hat{\boldsymbol{s}}_{i,j}^v$ and $\boldsymbol{s}_{i,j}^v$ denote the corresponding scaling factors. $Quat(\boldsymbol{R}^v)$ denotes the quaternion form of $\boldsymbol{R}^v$, f represents the focal length. The transformation of the rotation and scale parameters can be interpreted as rotating the Gaussian ellipsoids into the camera coordinate system, followed by a perspective scaling toward the focal plane. Since all of these transformations are invertible, the approximations involved do not compromise rendering performance.

To illustrate the effectiveness of VPT, we analyze the distribution of Gaussian geometric parameters before and after the transformation, as depicted in Fig. 3. Compared to the initial distribution, the transformed Gaussian geometric parameters exhibit enhanced regularity and spatial coherence.

D. Visibility-aware Basis Reduction

3DGS employs SH feature coefficients to model the anisotropic radiance distribution of each Gaussian ellipsoid. Although SH functions can effectively model the radiance with strong physical interpretability, from a compression perspective, the representation is notably redundant. Representing RGB colors using SH functions of order N_l requires a total of $3 \cdot (N_l + 1)^2$ coefficients, leading to substantial storage overhead. However, Many SH basis functions correspond to spherical regions rarely or never observed from the available viewpoints, thus minimally contributing to rendering quality.

The redundancy is particularly pronounced in sparse view reconstruction tasks, where the range of novel views is limited due to the lack of visual information from distant viewpoints. As illustrated in Fig. 4, the response of SH functions varies

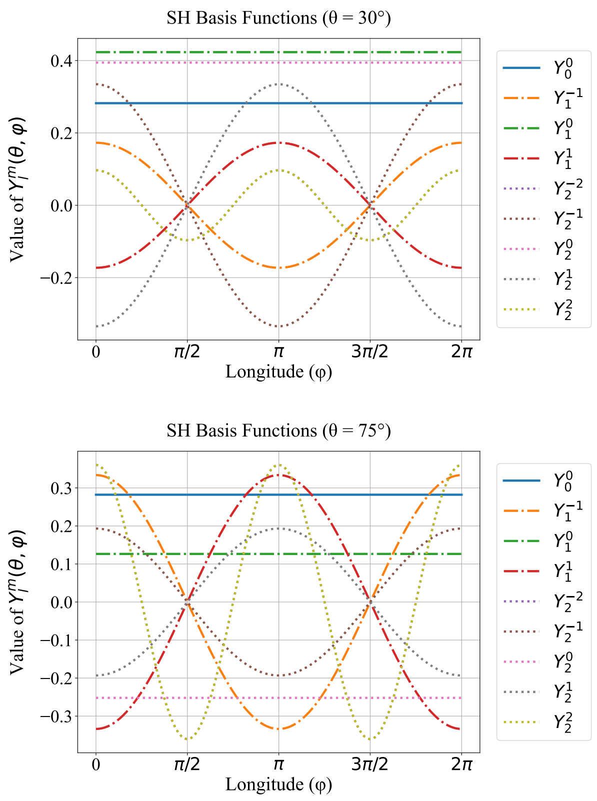

Fig. 4. The variation of SH basis function values with respect to viewing direction. At a specific direction, a larger absolute value indicates that the corresponding SH coefficient has a greater influence on the final rendering result. Y_l^m in the figure denotes the SH basis function of degree l, where m ranges from $-l$ to l, representing $2l+1$ distinct spatial orientations.

significantly across viewing angles. Therefore, treating all SH coefficients equally without considering their visibility may reduce the compression efficiency.

To address this issue, we introduce the VABR, which linearly adapts standard SH bases into a compact, visibility-aware basis set. We first compute the visibility-based weights to capture the directional importance of each SH component. Each weight factor λ_l^m is defined as the average absolute value of the corresponding SH basis function Y_l^m over a region of valid directions,

$$\lambda_l^m = \mathbb{E}_{\boldsymbol{D}}|Y_l^m(\boldsymbol{\omega})| = \frac{1}{|\boldsymbol{D}|}\int_{\boldsymbol{\omega}\in\boldsymbol{D}} |Y_l^m(\boldsymbol{\omega})| d\boldsymbol{\omega}, \tag{7}$$

where l and m represent the order and orientation factors of the SH function. $\boldsymbol{D}$ denotes the area of valid directions. In practice, it is challenging to rigorously define the area $\boldsymbol{D}$, so we approximate it via Monte Carlo integration,

$$\lambda_l^m = \frac{1}{N_s}\sum_{i=1}^{N_s} |Y_l^m(\boldsymbol{\omega}_i)|, \tag{8}$$

where N_s is the number of sample directions. To obtain representative viewing directions, we cast camera rays through a 3×3 grid uniformly covering the entire image plane for each input view and aggregate them.

Subsequently, these weights inform the construction of a compact set of new spherical basis functions, which also reflect the statistical distribution of SH coefficients within the scene. In particular, to ensure balanced perceptual contributions, we scale the original SH bases by their visibility-aware weights and linearly combine them into new optimized bases, resulting in the following transformation,

$$\hat{\boldsymbol{Y}}(\boldsymbol{\omega}) = \boldsymbol{W}^T \cdot \boldsymbol{\Delta}^{-1} \cdot \boldsymbol{Y}(\boldsymbol{\omega}), \tag{9}$$

where $\boldsymbol{Y}(\boldsymbol{\omega})$ and $\hat{\boldsymbol{Y}}(\boldsymbol{\omega})$ denote the original SH basis functions and the transformed basis functions, respectively. The weight matrix $\boldsymbol{\Delta}$ denotes the diagonal matrix consisting of $(\lambda_0^0, ..., \lambda_l^m, ...)$. $\boldsymbol{W} \in \mathbb{R}^{d\times k}$ represents the transform matrix to be solved, which contains a set of d-dimensional unit orthogonal vectors, with k representing the dimensionality of the transformed function space. Subsequently, the forward and inverse transformations of the color coefficients are as follows,

$$\boldsymbol{Z} = \boldsymbol{W}^T \cdot \boldsymbol{\Delta} \cdot \boldsymbol{X}, \quad \hat{\boldsymbol{X}} = \boldsymbol{\Delta}^{-1} \cdot \boldsymbol{W} \cdot \boldsymbol{Z}, \tag{10}$$

where $\boldsymbol{X}, \hat{\boldsymbol{X}} \in \mathbb{R}^{d\times M}, \boldsymbol{Z} \in \mathbb{R}^{k\times M}$ represent the input, reconstructed, and transformed data matrices, respectively. Here, M refers to the total number of Gaussian ellipsoids in the current scene.

To determine the transformation matrix $\boldsymbol{W}$ that can optimally preserve the perceptual information from higher-dimensional SH coefficients, we perform a statistical analysis on all SH coefficients in the current scene. Motivated by the formulation of principal component analysis, we compute the covariance matrix of $\boldsymbol{\Delta} \cdot \boldsymbol{X}$ and extract its principal components via eigen decomposition. In particular, the eigenvectors corresponding to the k largest eigenvalues are selected as the principal basis to form the transform matrix $\boldsymbol{W}$.

IV. Experimental Results

Datasets. We conduct comprehensive experiments on several widely used benchmark datasets in the field. Among them, RealEstate10K [57] and ACID [58] represent two large-scale multi-view 3D scene datasets that have been extensively adopted in related research. RealEstate10K dataset primarily comprises indoor scene multi-view images collected from online sources, containing 7,289 test scenes. The ACID dataset consists of numerous outdoor scenes captured by aerial drones, with 1,972 scenes for testing. In addition, we evaluate our method on a subset of the DL3DV dataset, containing 140 test scenes. Our test condition strictly adheres to the common configurations adopted in previous studies [12], [30], [35].

Baseline. In the Gaussian generation stage, we adopt representative feedforward inference models, including MVSplat [35] and DepthSplat [12], to generate 3D Gaussian feature maps. To demonstrate the generality and efficiency of the proposed pipeline, we compare our method with the optimization-free compression approach FCGS [11] in terms of compression efficiency.

Implementation details. In our experiments, the reserved color feature dimension k for VABR is fixed at 6. For 14-bit quantization, the quantization step for each channel is configured as σ/α, where σ represents the standard deviation of the current channel and α is an empirically chosen scaling factor, as detailed in Table II. We initially set α to 256 for each

TABLE I
QUANTITATIVE COMPARISON WITH 2 INPUT VIEWS. TINYSPLAT IS COMPARED WITH UNCOMPRESSED MODELS AND FCGS ON 3D GAUSSIANS GENERATED BY DIFFERENT FEEDFORWARD METHODS. "RAW" INDICATES THE UNCOMPRESSED BASELINE.

Inference	Compression	Re10k				ACID			
		PSNR(dB)↑	SSIM↑	LPIPS↓	Size(MB)↓	PSNR(dB)↑	SSIM↑	LPIPS↓	Size(MB)↓
MVSplat	Raw	26.36	0.8679	0.1291	43	28.21	0.8419	0.1448	43
	FCGS	25.90	0.8586	0.1533	2.650	27.61	0.8298	0.1800	2.596
(CVPR 2024)	Ours	26.32	0.8671	0.1349	0.216	28.13	0.8401	0.1528	0.229
DepthSplat	Raw	27.50	0.8900	0.1125	19	28.37	0.8481	0.1417	19
	FCGS	27.35	0.8863	0.1201	2.674	28.27	0.8454	0.1466	2.666
(CVPR 2025)	Ours	27.43	0.8886	0.1192	0.166	28.29	0.8470	0.1501	0.181

TABLE II
α AND Q_c CONFIGURATIONS FOR ALL DATA CHANNELS.

	depth	offset xy	scale	rotation	color	opacity
N_{dim}	1	2	3	4	6	1
α	2048	256	256	256	1024	256
Q_c	-4	12	0	9	3	0

channel, and selectively increase it for a few error-sensitive channels to improve reconstruction quality. Due to the varied distributions across different color basis components, we adopt a shared quantization step for all color channels to ensure consistent quantization. Here, σ represents the standard deviation of the most dominant component. After quantization, an offset is applied so that the minimum quantized value becomes 0. Extreme outliers in each channel are truncated to ensure that the maximum quantized values do not exceed the valid 14-bit range.

In the video coding module, each channel is treated as an independent grayscale image and encoded using the HEVC reference software HTM15.0-RExt8.1, with all channels encoded in parallel via separate processes. To achieve different compression rates, we adjust a global Quantization Parameter (QP), denoted as Q_g. Then, a set of per-channel QP offsets Q_c is experimentally configured, as summarized in Table II. Subsequently, the QP values for the HEVC codec are configured as $Q_c + Q_g$. All experiments are conducted on a workstation equipped with an Intel Core i7-13700K CPU and a single NVIDIA RTX 4090 graphics card, running in a WSL2.0 environment with Ubuntu 20.04.

The overall syntax elements include Gaussian geometric parameters (position, scale, and rotation), color feature coefficients, and opacity. In addition, a small amount of metadata is explicitly signaled to the decoder, including camera parameters, the basis vectors used for dimensionality reduction, as well as per-channel quantization steps and offset values.

A. Objective Results

Overall Compression Performance. We first generate the 3D Gaussian models using the official implementations and

TABLE III
COMPRESSION PERFORMANCE UNDER DIFFERENT NUMBERS OF INPUT VIEWS. "RAW" INDICATES THE UNCOMPRESSED BASELINE GENERATED VIA DEPTHSPLAT.

Method	Views	PSNR (dB)↑	SSIM↑	LPIPS↓	Size (MB)↓
Raw	2	19.62	0.6265	0.2986	33
FCGS		19.26	0.6183	0.3061	4.518
Ours		19.52	0.6196	0.3147	0.615
Raw	4	23.16	0.7792	0.1745	66
FCGS		22.23	0.7619	0.1897	9.054
Ours		22.99	0.7694	0.1923	1.275
Raw	6	24.16	0.8191	0.1456	100
FCGS		22.74	0.7943	0.1672	13.561
Ours		23.95	0.8085	0.1633	1.892

pre-trained weights MVSplat[1] (SHA: 1f5e5486) and DepthSplat[2] (SHA: 175b17a6). Then we compress the generated models using our compression framework to evaluate its effectiveness.

We evaluate the rendering PSNR, SSIM, LPIPS, and compressed model size on two standard datasets with 2 input views and 3 target views, as summarized in Table I. We employ a near-lossless compression configuration in our TinySplat, where Q_g is set to 0. For the SoTA method FCGS, we faithfully reproduce its performance using the official implementation[3] to ensure a fair comparison. The results show that TinySplat achieves over 100× compression on the 3D Gaussian data produced by the baseline, with negligible loss in rendering quality. Compared to the SoTA training-free method, our approach still achieves a 15× higher compression ratio, while providing comparable rendering quality. The rendering quality suffers a noticeable drop when using the FCGS pipeline to compress Gaussian models from MVSplat. This performance degradation could be attributed to the distinctive distribution of 3D Gaussian data generated by MVSplat, as illustrated in the FCGS paper [11]. We highlight that we evaluate the performance on the complete datasets following the general configuration and report the average results to ensure a comprehensive and reliable comparison.

[1]https://github.com/donydchen/mvsplat
[2]https://github.com/cvg/depthsplat
[3]https://github.com/YihangChen-ee/FCGS

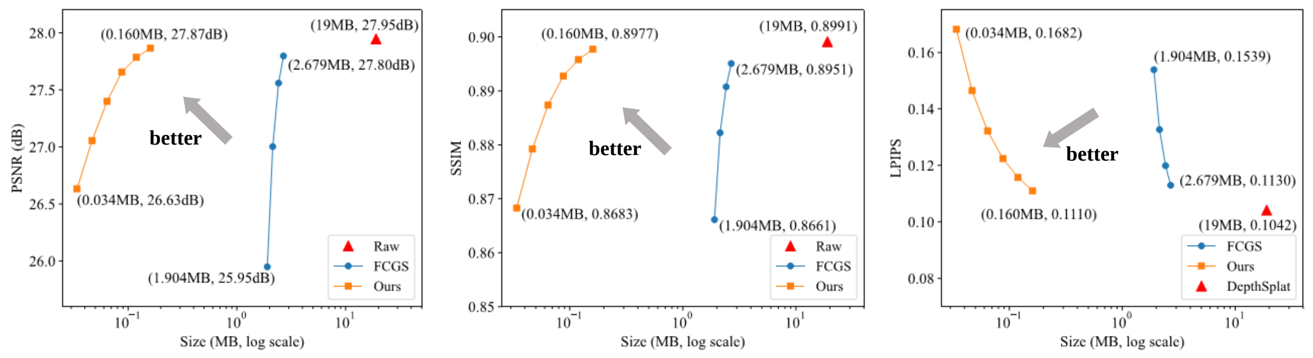

Fig. 5. Rate-distortion performance on the Re10K dataset, where distortion is measured in terms of novel view synthesis quality. "Raw" denotes the uncompressed 3D Gaussian model from DepthSplat. Compared to the SoTA method, our method achieves comparable quality (PSNR, SSIM, and LPIPS) with only 6% of the storage size.

TABLE IV
RUNTIME ANALYSIS. OUR TINYSPLAT IS DECOMPOSED INTO KEY COMPONENTS.

Method	Encoding Time (s)				Decoding
	VPT	VABR	HEVC	Total	Time (s)
FCGS	-	-	-	4.074	4.095
TinySplat	0.003	0.004	0.971	**1.016**	**0.042**

TABLE V
BIT ALLOCATION UNDER DIFFERENT COMPRESSION RATIOS.

Q_g	Size (KB)	position	scale	rotation	color	opacity
0	164.63	21.4%	17.5%	13.0%	34.6%	13.5%
3	122.29	21.0%	18.0%	11.5%	35.7%	13.7%
6	89.91	20.5%	18.8%	10.2%	36.7%	13.8%
12	47.90	20.0%	20.1%	8.4%	39.0%	12.6%

Beyond the two-view testing configuration, we also evaluate the compression performance under varying numbers of input views on the DL3DV dataset, as shown in Table III. We follow the testing configuration in DepthSplat [12] and reproduce the reported performance. The results demonstrate that our TinySplat generalizes well, as no significant degradation in compression quality is observed with more input views.

Rate-Distortion Performance. To evaluate the impact of compression distortion on the rendering quality of 3D Gaussian models, we compare the rate-distortion performance of our proposed method against FCGS, as shown in Fig. 5. To reduce the evaluation cost, we only conduct experiments on the first 200 test scenes from the Re10K and ACID datasets. The results show that our method achieves comparable or even better PSNR, SSIM, and LPIPS performance compared to FCGS with only 6% of its storage space.

Runing time Analysis. To assess the computational cost of our TinySplat, we measure the average encoding and decoding times, as reported in Table IV. TinySplat encodes a scene in approximately 1 second and supports real-time decoding. In contrast, the SoTA approach, which heavily relies on NN–based probabilistic models, requires about 4$\times$ more time for encoding and over 100$\times$ more for decoding.

The primary computational overhead in our pipeline arises from the HEVC video encoder. However, we note that the HEVC codec in our implementation is the reference software, which prioritizes coding efficiency and correctness over running speed. In practice, significantly faster encoding and decoding can be achieved through engineering optimizations [59] or dedicated hardware implementations [60], [61].

Bit Allocation. To better understand the bit allocation characteristics of our framework and guide future optimizations, we encode 200 scenes under various QP settings and analyze the average storage distribution across different components, as summarized in Table V. The results show that the relative bit allocation remains largely consistent across compression ratios, with color parameters consistently occupying the largest share of the bitrate.

B. Subjective Results

We adopt DepthSplat as the representative Gaussian inference method and compare subjective compression artifacts using FCGS and our proposed framework, as shown in Fig. 6. In particular, we adopt the highest compression ratio setting offered by FCGS, which still results in a storage size several times larger than that of TinySplat. The rendering results show that FCGS easily introduces prominent watermark-like artifacts, whereas TinySplat maintains visually negligible degradation, even at higher compression ratios.

We also evaluate the subjective performance under 6 input views, as illustrated in Fig. 8. The results demonstrate that increasing the number of input views does not noticeably degrade the perceptual quality of our method, highlighting its strong generalization capability. In contrast, FCGS produces severe artifacts even at relatively low compression rates.

C. Ablation Studies

Component-wise ablation analysis. We conduct an ablation study by separately disabling the proposed techniques to assess their impact on the compression performance, as shown in Fig. 7. We utilize 200 scenes from the Re10k dataset for

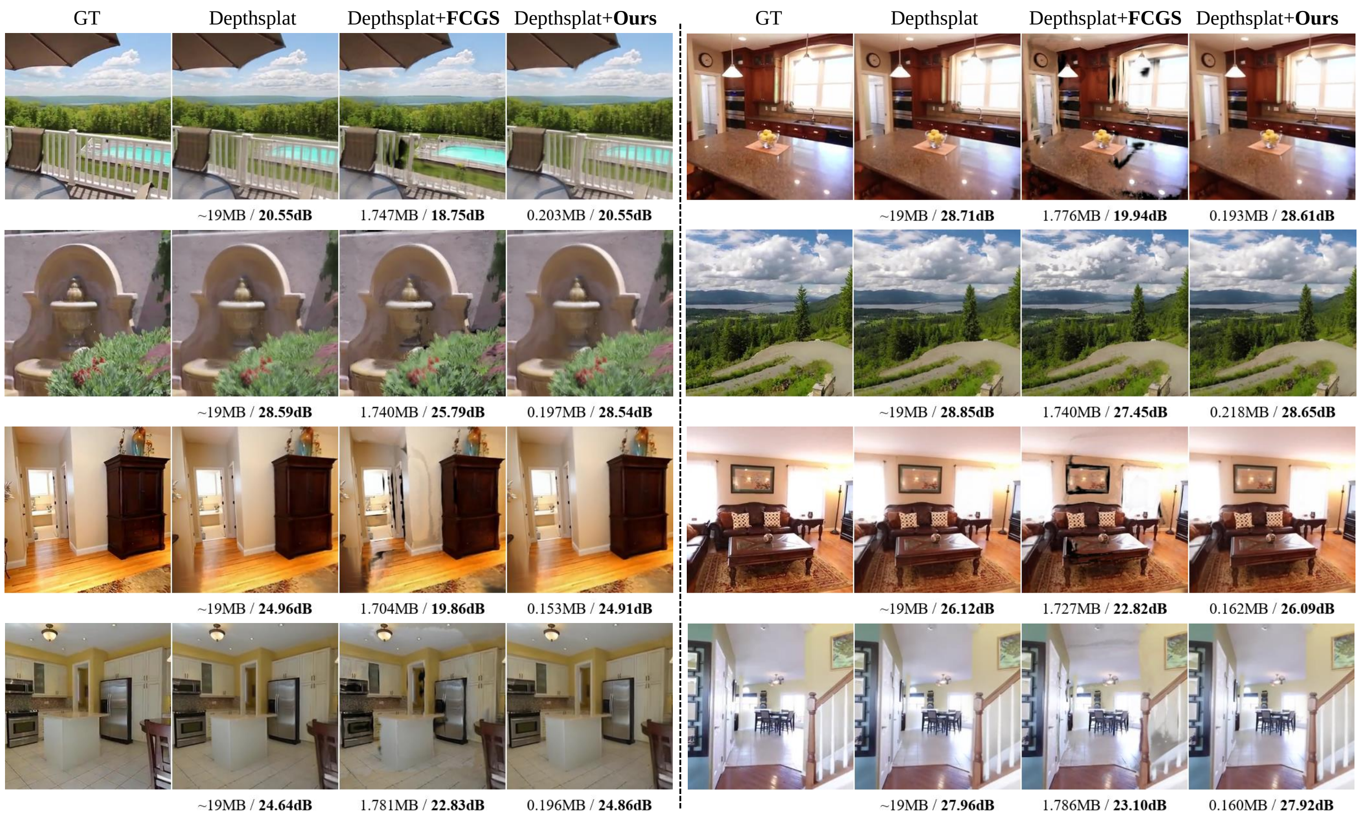

Fig. 6. Subjective results on Re10k dataset. We compress the 3D Gaussian models from DepthSplat with both FCGS and the proposed TinySplat. FCGS exhibits severe watermark-like artifacts at 11× compression ratio. In contrast, our method achieves up to 100× compression while maintaining visually indistinguishable rendering quality from the uncompressed reference.

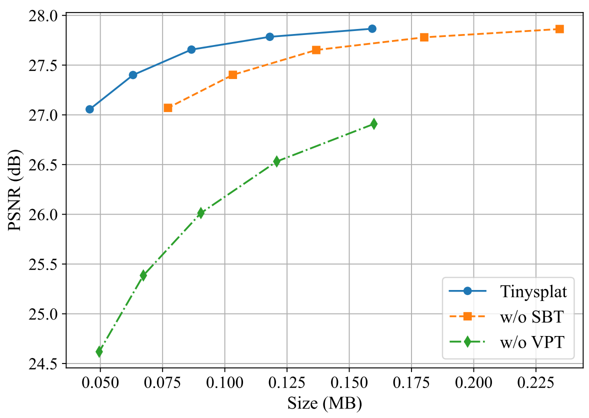

Fig. 7. Component-wise ablation analysis. We separately disable the VPT module and the VABR module to test their impact on coding efficiency. The PSNR is measured between the rendered novel view and the ground truth.

testing. Results demonstrate that VPT can effectively reduce the loss of geometric information under the same bitrate, thereby improving rendering quality. Meanwhile, VABR enhances compression efficiency by concentrating the energy of SH coefficients, enabling significantly higher compression ratios.

Ablation on VABR. We conducted an experimental analysis to investigate the impact of the retained feature dimensions on compression efficiency. Starting from the most dominant component, we progressively added additional components to evaluate how encoding performance evolves. In addition, we computed the storage cost associated with each individual dimension. As shown in Fig. 9, the results indicate that after applying the VABR, the first feature dimension concentrates the majority of the information, accounting for the largest portion of the total bitrate. The information content in subsequent dimensions decreases gradually. The overall rendering performance remains nearly unchanged as the feature dimensionality exceeds 6. These results validate the effectiveness of our VABR, which concentrates most of the meaningful information in the first few dimensions.

V. Limitations

In this work, we do not modify existing 3D Gaussian inference networks; instead, we focus on developing a general compression framework applicable to the Gaussian data they generated. While our method effectively produces compact 3D Gaussian representations, there remains a notable gap between our achieved compression ratios and theoretical limits. Since 3D Gaussians are derived entirely from a few input images, information theory suggests that their entropy should not surpass that of the inputs. However, even after compression, the size of the 3D Gaussian models still exceeds that of the original images compressed with conventional codecs. We attribute this

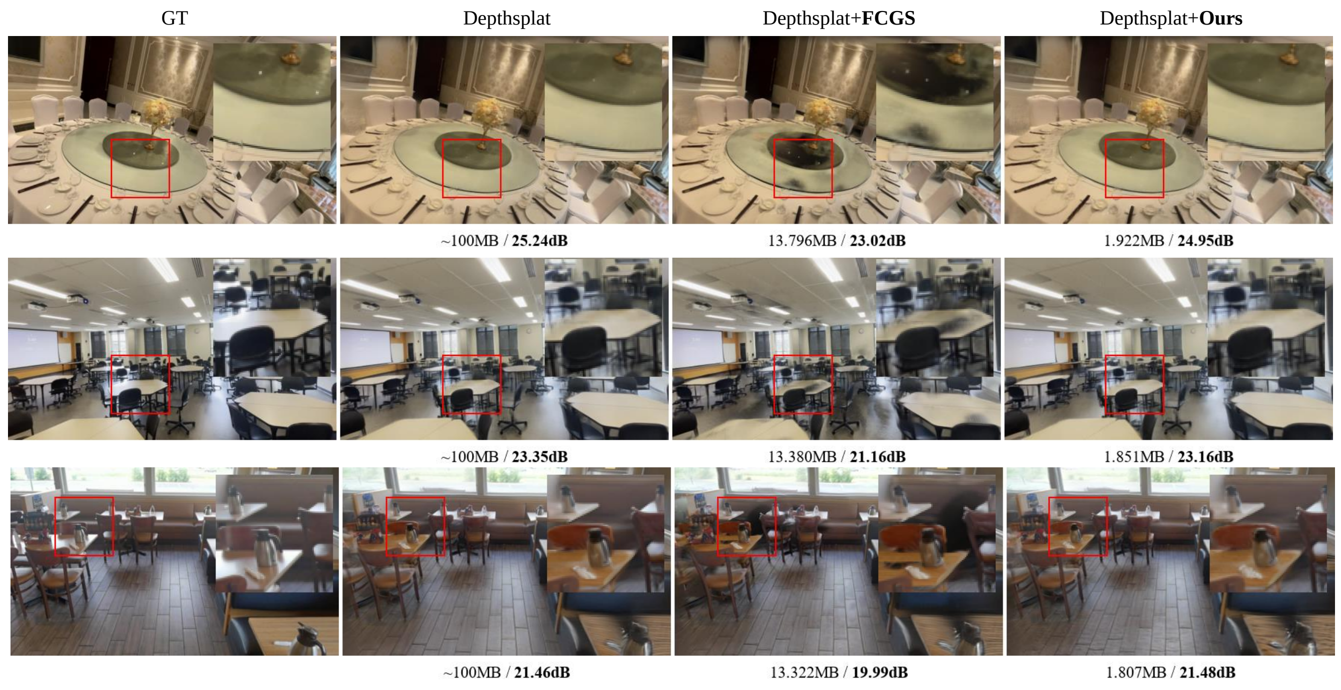

Fig. 8. Subjective results under 6 input views on DL3DV dataset. Our TinySplat achieves superior objective and perceptual quality using only 15% of the storage compared to the SoTA compression method. FCGS exhibits noticeable compression artifacts even at relatively high bitrates.

discrepancy primarily to the inference process itself, where complex NN models introduce significant prior knowledge into the generated 3D Gaussian representations. Developing lightweight methods to explicitly extract and leverage these priors presents a promising direction for future research. In addition, redesigning or fine-tuning inference networks to generate inherently more compressible representations represents another important avenue for exploration.

VI. Conclusion

In this paper, we presented TinySplat, a fully feedforward approach for generating compact 3D Gaussian scene representations directly from multi-view images without per-scene optimization. Bridging the gap between fast 3D Gaussian generation and practical deployment, TinySplat integrates a feedforward inference stage with a novel rendering-aware compression stage. For the compression stage, we proposed a dedicated framework with VPT and VABR modules to reduce structural and perceptual redundancy, respectively. Furthermore, we employed a standard video codec to eliminate spatial redundancy. Extensive experiments demonstrated that TinySplat significantly reduces storage requirements while maintaining high rendering quality.

These findings offer an intuitive understanding of the redundancy characteristics in feedforward-generated 3D Gaussian data, providing practical guidance for future exploration and system design. We believe TinySplat paves the way for scalable and deployable 3D scene representation, and sets a solid foundation for future research in real-time neural graphics systems.

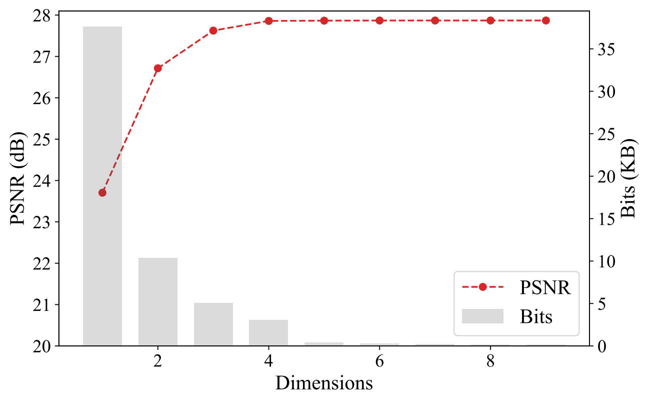

Fig. 9. Ablation study of the VABR. We present the variation of rendering PSNR with feature dimensionality increasing, along with the storage cost of each individual feature channel. The results show that our method more effectively concentrates energy into the leading dimensions, enabling a more compact and efficient feature representation.

References

[1] P. Papantonakis, G. Kopanas, B. Kerbl, A. Lanvin, and G. Drettakis, "Reducing the memory footprint of 3d gaussian splatting," in *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 7, no. 1, 2024, pp. 1–17.
[2] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3d gaussian representation for radiance field," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 719–21 728.
[3] Y. Shi, G. Morin, S. Gasparini, and W. T. Ooi, "Lapisgs: Layered progressive 3d gaussian splatting for adaptive streaming," *arXiv preprint arXiv:2408.14823*, 2024.
[4] H. Wang, H. Zhu, T. He, R. Feng, J. Deng, J. Bian, and Z. Chen, "End-to-end rate-distortion optimized 3d gaussian representation," in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 76–92.

[5] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, and Z. Wang, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," in *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 140 138–140 158.

[6] 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.

[7] Q. Gao, J. Meng, C. Wen, J. Chen, and J. Zhang, "Hicom: Hierarchical coherent motion for dynamic streamable scenes with 3d gaussian splatting," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 80 609–80 633.

[8] L. Tang, J. Yang, R. Peng, Y. Zhai, S. Shen, and R. Wang, "Compressing streamable free-viewpoint videos to 0.1 mb per frame," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 7, 2025, pp. 7257–7265.

[9] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "HAC: Hash-grid assisted context for 3d gaussian splatting compression," in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 422–438.

[10] Y. Wang, Z. Li, L. Guo, W. Yang, A. Kot, and B. Wen, "Contextgs: Compact 3d gaussian splatting with anchor level context model," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 51 532–51 551.

[11] Y. Chen, Q. Wu, M. Li, W. Lin, M. Harandi, and J. Cai, "Fast feedforward 3d gaussian splatting compression," *arXiv preprint arXiv:2410.08017*, 2024.

[12] H. Xu, S. Peng, F. Wang, H. Blum, D. Barath, A. Geiger, and M. Pollefeys, "Depthsplat: Connecting gaussian splatting and depth," *arXiv preprint arXiv:2410.13862*, 2024.

[13] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[14] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Transactions on Graphics*, vol. 41, no. 4, pp. 1–15, 2022.

[15] C. Sun, M. Sun, and H.-T. Chen, "Direct voxel grid optimization: Superfast convergence for radiance fields reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5459–5469.

[16] S. Fridovich-Keil, A. Yu, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5501–5510.

[17] S. Fridovich-Keil, G. Meanti, F. R. Warburg, B. Recht, and A. Kanazawa, "K-planes: Explicit radiance fields in space, time, and appearance," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 12 479–12 488.

[18] A. Chen, Z. Xu, A. Geiger, J. Yu, and H. Su, "Tensorf: Tensorial radiance fields," in *Computer Vision–ECCV 2022*, 2022, pp. 333–350.

[19] A. Yu, R. Li, M. Tancik, H. Li, R. Ng, and A. Kanazawa, "Plenoctrees for real-time rendering of neural radiance fields," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 5752–5761.

[20] 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, pp. 139–1, 2023.

[21] Y. Fu, S. Liu, A. Kulkarni, J. Kautz, A. A. Efros, and X. Wang, "Colmap-free 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 796–20 805.

[22] Z. Qian, S. Wang, M. Mihajlovic, A. Geiger, and S. Tang, "3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 5020–5030.

[23] J. Meng, H. Li, Y. Wu, Q. Gao, S. Yang, J. Zhang, and S. Ma, "Mirror-3dgs: Incorporating mirror reflections into 3d gaussian splatting," in *Proceedings of the IEEE International Conference on Visual Communications and Image Processing*. IEEE, 2024, pp. 1–5.

[24] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-splatting: Alias-free 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 447–19 456.

[25] M. Niemeyer, J. T. Barron, B. Mildenhall, M. S. Sajjadi, A. Geiger, and N. Radwan, "Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 5480–5490.

[26] P. Truong, M.-J. Rakotosaona, F. Manhardt, and F. Tombari, "Sparf: Neural radiance fields from sparse and noisy poses," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 4190–4200.

[27] J. Zhang, J. Li, X. Yu, L. Huang, L. Gu, J. Zheng, and X. Bai, "Cor-gs: sparse-view 3d gaussian splatting via co-regularization," in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 335–352.

[28] J. Li, J. Zhang, X. Bai, J. Zheng, X. Ning, J. Zhou, and L. Gu, "Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 775–20 785.

[29] L. Han, J. Zhou, Y.-S. Liu, and Z. Han, "Binocular-guided 3d gaussian splatting with view consistency for sparse view synthesis," *arXiv preprint arXiv:2410.18822*, 2024.

[30] D. Charatan, S. L. Li, A. Tagliasacchi, and V. Sitzmann, "pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 457–19 467.

[31] A. Chen, Z. Xu, F. Zhao, X. Zhang, F. Xiang, J. Yu, and H. Su, "Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 124–14 133.

[32] Y. Chen, H. Xu, Q. Wu, C. Zheng, T.-J. Cham, and J. Cai, "Explicit correspondence matching for generalizable neural radiance fields," *arXiv preprint arXiv:2304.12294*, 2023.

[33] S. Szymanowicz, E. Insafutdinov, C. Zheng, D. Campbell, J. F. Henriques, C. Rupprecht, and A. Vedaldi, "Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image," *arXiv preprint arXiv:2406.04343*, 2024.

[34] H. Xu, A. Chen, Y. Chen, C. Sakaridis, Y. Zhang, M. Pollefeys, A. Geiger, and F. Yu, "Murf: multi-baseline radiance fields," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 041–20 050.

[35] Y. Chen, H. Xu, C. Zheng, B. Zhuang, M. Pollefeys, A. Geiger, T.-J. Cham, and J. Cai, "Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images," in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 370–386.

[36] B. Ye, S. Liu, H. Xu, X. Li, M. Pollefeys, M.-H. Yang, and S. Peng, "No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images," *arXiv preprint arXiv:2410.24207*, 2024.

[37] R. Gao, A. Holynski, P. Henzler, A. Brussee, R. Martin-Brualla, P. Srinivasan, J. T. Barron, and B. Poole, "Cat3d: Create anything in 3d with multi-view diffusion models," *arXiv preprint arXiv:2405.10314*, 2024.

[38] R. Wu, B. Mildenhall, P. Henzler, K. Park, R. Gao, D. Watson, P. P. Srinivasan, D. Verbin, J. T. Barron, B. Poole *et al.*, "Reconfusion: 3d reconstruction with diffusion priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 21 551–21 561.

[39] K. Zhang, S. Bi, H. Tan, Y. Xiangli, N. Zhao, K. Sunkavalli, and Z. Xu, "Gs-lrm: Large reconstruction model for 3d gaussian splatting," in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 1–19.

[40] X. Liu, C. Zhou, and S. Huang, "3dgs-enhancer: Enhancing unbounded 3d gaussian splatting with view-consistent 2d diffusion priors," in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 133 305–133 327.

[41] M. Liu, R. Shi, L. Chen, Z. Zhang, C. Xu, X. Wei, H. Chen, C. Zeng, J. Gu, and H. Su, "One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 072–10 083.

[42] Y. Chen, C. Zheng, H. Xu, B. Zhuang, A. Vedaldi, T.-J. Cham, and J. Cai, "Mvsplat360: Feed-forward 360 scene synthesis from sparse views," *arXiv preprint arXiv:2411.04924*, 2024.

[43] S. Szymanowicz, C. Rupprecht, and A. Vedaldi, "Splatter image: Ultra-fast single-view 3d reconstruction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 10 208–10 217.

[44] J. Tang, Z. Chen, X. Chen, T. Wang, G. Zeng, and Z. Liu, "Lgm: Large multi-view gaussian model for high-resolution 3d content creation," in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 1–18.

[45] B. Zhou, S. Zheng, H. Tu, R. Shao, B. Liu, S. Zhang, L. Nie, and Y. Liu, "Gps-gaussian+: Generalizable pixel-wise 3d gaussian splatting for real-time human-scene rendering from sparse views," *arXiv preprint arXiv:2411.11363*, 2024.

[46] S. Zheng, B. Zhou, R. Shao, B. Liu, S. Zhang, L. Nie, and Y. Liu, “Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 19 680–19 690.

[47] Y. Xu, Z. Shi, W. Yifan, H. Chen, C. Yang, S. Peng, Y. Shen, and G. Wetzstein, “Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation,” in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 1–20.

[48] A. Chen, H. Xu, S. Esposito, S. Tang, and A. Geiger, “Lara: Efficient large-baseline radiance fields,” in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 338–355.

[49] C. Wewer, K. Raj, E. Ilg, B. Schiele, and J. E. Lenssen, “Latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction,” in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 456–473.

[50] Z. Min, Y. Luo, J. Sun, and Y. Yang, “Epipolar-free 3d gaussian splatting for generalizable novel view synthesis,” *arXiv preprint arXiv:2410.22817*, 2024.

[51] Z. Chen, J. Yang, and H. Yang, “Pref3r: Pose-free feed-forward 3d gaussian splatting from variable-length image sequence,” *arXiv preprint arXiv:2411.16877*, 2024.

[52] S. Zhang, X. Fei, F. Liu, H. Song, and Y. Duan, “Gaussian graph network: Learning efficient and generalizable gaussian representations from multi-view images,” in *Advances in Neural Information Processing Systems*, vol. 37, 2024, pp. 50 361–50 380.

[53] K. Navaneet, K. Pourahmadi Meibodi, S. Abbasi Koohpayegani, and H. Pirsiavash, “Compgs: Smaller and faster gaussian splatting with vector quantization,” in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 330–349.

[54] W. Morgenstern, F. Barthel, A. Hilsmann, and P. Eisert, “Compact 3d scene representation via self-organizing gaussian grids,” in *Computer Vision–ECCV 2024*. Springer, 2024, pp. 18–34.

[55] 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.

[56] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, “HAC++: Towards 100X Compression of 3D Gaussian Splatting,” *arXiv preprint arXiv:2501.12255*, 2025.

[57] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely, “Stereo magnification: Learning view synthesis using multiplane images,” *arXiv preprint arXiv:1805.09817*, 2018.

[58] A. Liu, R. Tucker, V. Jampani, A. Makadia, N. Snavely, and A. Kanazawa, “Infinite nature: Perpetual view generation of natural scenes from a single image,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 14 458–14 467.

[59] T. K. Heng, W. Asano, T. Itoh, A. Tanizawa, J. Yamaguchi, T. Matsuo, and T. Kodama, “A highly parallelized H.265/HEVC real-time UHD software encoder,” in *Proceedings of the IEEE International Conference on Image Processing*. IEEE, 2014, pp. 1213–1217.

[60] G. Pastuszak and A. Abramowski, “Algorithm and architecture design of the h.265/hevc intra encoder,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 1, pp. 210–222, 2016.

[61] F. Bossen, B. Bross, K. Suhring, and D. Flynn, “HEVC Complexity and Implementation Analysis,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1685–1696, 2012.