# Decoding VGGT Features in 3D with Sparse Convolution

Ran Gong Zhehong Ren Canyu Zhang Yuxuan Feng

December 2025

## Abstract

Feed-forward 3D Gaussian Splatting has became as an efficient approach for real-time novel view synthesis by directly predicting 3D Gaussian primitives from unconstrained images. In this work, we investigate whether explicitly decoding intermediate representations in 3D can further improve reconstruction quality. Building on the VGGT backbone, we introduce a sparse 3D convolutional refinement module that operates on voxelized features derived from predicted depth and image features. The proposed module aggregates local geometric context through residual sparse 3D convolutions while preserving the feed-forward efficiency of the original pipeline. To ensure stable learning under sparse activations, we apply a lightweight normalization during 3D feature aggregation. Experimental results demonstrate consistent improvements over per-point MLP baselines and naive 3D convolution variants, yielding gains in PSNR, SSIM, and perceptual metrics, particularly in scenes with complex geometry or sparse observations. The project repository link is: https://github.com/Aatroeee/cv_project

## 1 Introduction

In recent years, 3D Gaussian Splatting has emerged as a highly efficient and visually compelling representation for real-time novel view synthesis, which enables high-quality rendering through explicit Gaussian primitives [3]. Methods such as AnySplat extend this paradigm to unconstrained image collections via a transformer-based backbone that jointly estimates geometry, appearance, and camera parameters in a single forward pass [2]. However, existing feed-forward pipelines rely largely on per-point decoding with limited 3D spatial interaction, often resulting in incoherent predictions in regions with complex geometry or sparse observations. We therefore investigate whether explicitly decoding intermediate representations in 3D can improve reconstruction quality while preserving feed-forward efficiency.

Specifically, we preserve the original VGGT (Vision Geometry Grounded Transformer) backbone as the core feature extractor, utilizing pre-trained weights to retain the model's robust visual representation capabilities [5]. Our main contribution lies in replacing the original Gaussian head with a structurally similar but more expressive variant, and integrating a sparse 3D convolutional refinement network designed in a residual style [1]. This sparse 3D ResNet module operates on voxelized intermediate representations, with the goal of better aggregating local geometric and appearance cues across neighboring regions in 3D space.

We hypothesize that this architectural augmentation enables the model to make more coherent predictions in scenes with complex depth discontinuities or sparse observations, ultimately leading to improved fidelity in the reconstructed 3D Gaussians. Through this work, we aim to assess whether incorporating spatial priors via sparse 3D convolutions enhances the generalization and reconstruction quality of feed-forward Gaussian Splatting pipelines under minimal modification of the VGGT backbone.

## 2 Related Work

### 2.1 Feed-Forward Transformer Models

Recent work has shown that transformer-based architectures, when scaled and trained on diverse 3D-annotated datasets, can directly infer the full set of geometric attributes of a scene in a *pure feed-forward* manner [5]. A representative model is the Visual

Geometry Grounded Transformer (VGGT), which predicts camera intrinsics, camera extrinsics, depth maps, point maps, and long-range 3D point tracks from one or many input views [5]. Unlike classical Structure-from-Motion pipelines that rely on iterative optimization—feature matching, triangulation, and bundle adjustment—VGGT performs all computation in a single pass, dramatically reducing inference time.

This architectural shift demonstrates that large transformers can replace geometry-heavy optimization with learned inductive priors, achieving competitive or superior reconstruction accuracy while remaining significantly faster at test time [5]. Because of its efficiency, generality across sparse- and dense-view inputs, and strong geometric priors, VGGT forms an effective backbone for downstream tasks such as pose-free Gaussian Splatting [2].

### 2.2 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) represents a different line of work focused on explicit scene reconstruction and high-quality novel-view synthesis [3]. Traditional radiance field approaches, including NeRF-style volumetric rendering, achieve photorealistic results but require large neural networks, long training time, and slow rendering, making real-time performance impractical.

3DGS addresses these limitations by representing scenes using anisotropic 3D Gaussians derived from sparse points obtained during camera calibration [3]. Through interleaved optimization of positions, colors, opacities, and full covariance matrices, the method captures complex geometry and view-dependent radiance with high fidelity. A visibility-aware splatting algorithm enables efficient rendering and supports real-time frame rates at 1080p resolution [3]. Despite these strengths, 3DGS requires heavy optimization for each scene and does not operate in a feed-forward manner, limiting scalability to large datasets or interactive applications.

### 2.3 Hybrid Feed-Forward Gaussian Splatting

Bridging the two paradigms above, AnySplat introduces a hybrid framework that combines feed-forward transformer-based geometry prediction with the expressive 3D Gaussian scene representation [2]. Starting from uncalibrated image collections, a transformer encoder predicts depth maps, camera intrinsics, camera extrinsics, and Gaussian-related features in a single forward pass [2]. These predictions are fused into pixel-wise 3D Gaussians and further regularized through differentiable voxelization and geometry-aware losses.

This design allows AnySplat to scale to both sparse- and dense-view scenarios while avoiding the iterative optimization required by traditional 3DGS [3]. As a result, AnySplat achieves real-time novel-view synthesis with competitive geometric accuracy and substantially reduced inference latency [2]. Its hybrid nature—feed-forward geometry prediction coupled with Gaussian-based rendering—offers an efficient and flexible blueprint for next-generation 3D reconstruction pipelines.

## 3 Methodology

### 3.1 Overall Architecture

Our method follows the AnySplat pipeline while keeping the original design as intact as possible. We preserve both the pretrained VGGT backbone and the Gaussian prediction head, and introduce a lightweight 3D refinement module after the backbone features are converted into voxelized Gaussian features.

Given a set of input images, the VGGT backbone first produces multi-view geometric features and pseudo-geometry (camera parameters, depth, and related 3D attributes). These features are then processed by the original DPT-style Feature head to obtain per-point (or per-voxel) feature representations associated with the 3D Gaussians. Instead of directly using these features to parameterize the Gaussians, we insert an additional sparse 3D convolutional decoder, implemented as a ResNet-style network, which operates in the voxel space. The decoded features are finally used to produce the Gaussian parameters for differentiable splatting and rendering.

In summary, the overall architecture can be viewed

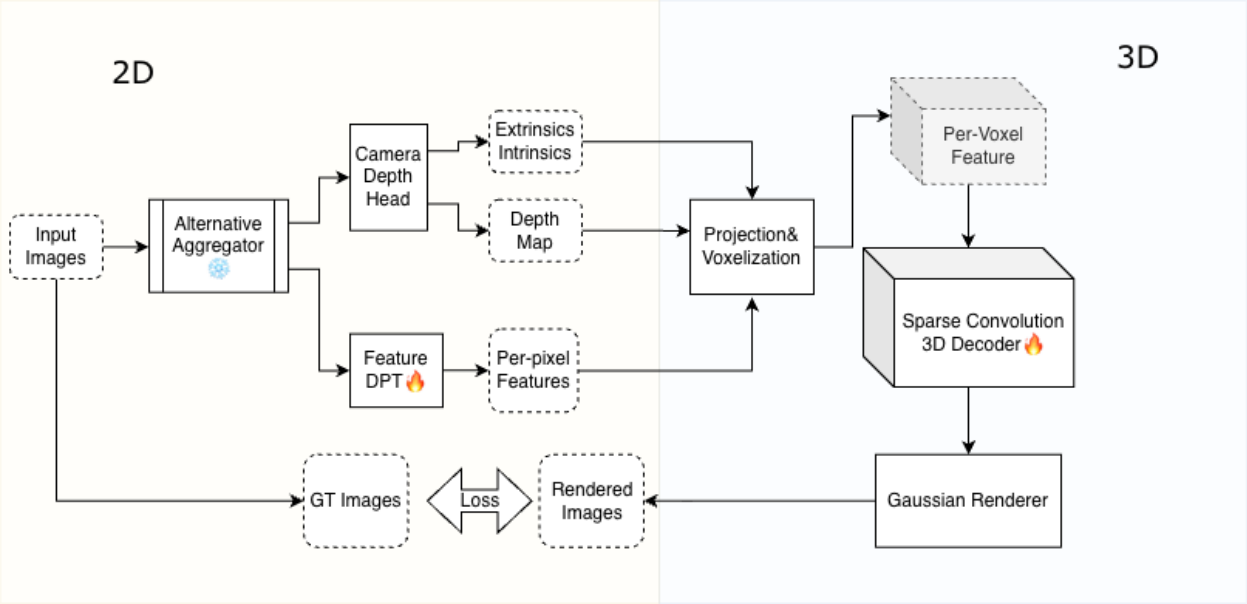


Figure 1: Overview of our pipeline.

as a three-stage pipeline:

1. VGGT backbone for multi-view feature and geometry prediction;
2. Gaussian Feature head for mapping backbone features to voxel-aligned features;
3. Sparse 3D convolutional ResNet for local 3D decoding of these features before final Gaussian parameterization.

Figure 1 illustrates the overall pipeline of our method.

### 3.2 Sparse 3D Convolutional Refiner

To decode voxelized Gaussian features, we introduce a sparse 3D convolutional module, termed `SparseConvRefiner` . The module is implemented using submanifold sparse convolutions to preserve the sparsity pattern of the voxel grid and to avoid unnecessary computation in empty regions.

The sparse decoder is parameterized by a configuration `SparseConvRefinerCfg`, specifying the input, hidden, and output channel dimensions, the number of layers, the convolutional kernel size, and the architectural variant. In our experiments, we adopt a ResNet-style architecture with kernel sizes $k \in \{1, 3, 5\}$.

**Sparse Tensor Construction.** Given voxel features $F \in \mathbb{R}^{N \times C}$ and their corresponding integer voxel coordinates in $(z, y, x)$ format, we construct a sparse 3D tensor by normalizing spatial indices such that the minimum occupied coordinate is zero and prepending batch indices. This yields a sparse voxel grid whose spatial extent is dynamically adapted to each scene.

**ResNet-style Sparse Blocks.** The SparseConvRefiner consists of three components: (i) an input submanifold convolution that maps input channels to hidden channels; (ii) a stack of sparse residual blocks(`SparseResBlock`); and (iii) an output submanifold convolution that projects features to the desired output dimension.

Each `SparseResBlock` contains two 3d convolution layers with kernel size $k$ and padding $k/2$, optional batch normalization in the feature space, and a ReLU nonlinearity. When the input and output channels differ, a $1 \times 1 \times 1$ sparse convolution is used as a shortcut to match dimensions; otherwise, the identity is used as the skip connection. The block outputs

$$\mathrm{ReLU}\big(\mathrm{Conv}_2(\mathrm{BN}_2(\cdot)) + \mathrm{shortcut}\big)$$

applied in the sparse feature space.

**Residual refinement.** The forward pass of `SparseConvRefiner` takes voxel features and indices, builds the sparse tensor, applies the chosen architecture (in our case, the ResNet stack), and then returns refined features at the same voxel locations. When the input and output channel dimensions match and `use_residual` is enabled, a global residual connection is added:

$$F_{\text{refined}} = F_{\text{input}} + F_{\text{conv}}.$$

These refined voxel features are then passed to the subsequent Gaussian parameterization stage of the pipeline, without altering the rest of the AnySplat–VGGT framework.

This design keeps the original VGGT and Gaussian head intact, while adding a structurally simple but spatially expressive 3D refinement layer that operates directly on the voxelized Gaussian features.

### 3.3 Kernel Normalization for Sparse 3D Convolution

Sparse 3D convolutions operate on irregularly occupied voxel grids, where the number of active neighbors varies significantly across spatial locations. In such settings, directly applying standard convolution kernels can lead to unstable feature aggregation: voxels with fewer active neighbors receive attenuated responses, while densely populated regions dominate the convolution output. This issue is particularly pronounced in feed-forward Gaussian Splatting pipelines, where voxel occupancy is highly sparse and uneven due to depth uncertainty and occlusions.

Formally, a sparse 3D convolution computes the output feature at voxel $u$ as

$$\mathbf{x}_u = \sum_{u+i \in \mathcal{N}(u)} W_i \, \mathbf{x}_{u+i}, \tag{1}$$

where $\mathcal{N}(u)$ denotes the set of active neighboring voxels within the convolutional kernel. When $|\mathcal{N}(u)|$ is small, the effective magnitude of $\mathbf{x}_u$ is reduced, leading to degraded signal propagation and unstable training.

To address this issue, we introduce a lightweight *kernel normalization* scheme that normalizes convolution weights over active neighbors only. Specifically, we rescale the convolution response by the $\ell_2$ norm of the participating kernel weights:

$$\mathbf{x}_u = \left( \sum_{u+i \in \mathcal{N}(u)} \|W_i\|^2 \right)^{-\frac{1}{2}} \sum_{u+i \in \mathcal{N}(u)} W_i \, \mathbf{x}_{u+i}. \tag{2}$$

This normalization stabilizes the output magnitude across varying sparsity patterns while preserving the relative contributions of neighboring features. Kernel normalization is applied within each sparse residual block of the 3D refinement module and incurs negligible computational overhead. As demonstrated in our experiments, this simple modification significantly improves training stability and reconstruction quality, particularly in scenes with sparse observations or complex geometric structures.

## 4 Experiments

### 4.1 Dataset

We conduct experiments on a subset of the DL3DV dataset [4], which contains large-scale multi-view image collections with diverse indoor and outdoor scenes. Due to computational constraints, we use only the first subset of DL3DV, consisting of a representative set of scenes with varying geometry complexity and camera trajectories.

Each scene includes multiple uncalibrated RGB images captured from different viewpoints. This subset provides sufficient diversity to evaluate the ability of feed-forward Gaussian Splatting methods to reconstruct fine-grained spatial structures under sparse and unconstrained observations.

### 4.2 Implementation Details

Our method is built upon the AnySplat framework and preserves the pretrained VGGT backbone without modification. We insert a lightweight sparse 3D convolutional refiner after voxelization of intermediate Gaussian features. Unless otherwise specified, we use submanifold sparse convolutions with kernel

Figure 2: Qualitative comparison of different methods. From left to right: ground truth, without normalization, with normalization, and baseline.

size $k = 3$ and a ResNet-style architecture. All models are trained end-to-end using the same optimization settings as the baseline.

### 4.3 Baselines

We compare our method against the following baselines:

- **Per-point MLP**: the original AnySplat Gaussian head with kernel size $k = 1$.
- **Naive 3D Convolution**: sparse 3D convolution with kernel size $k = 3$ without normalization.

These baselines allow us to isolate the impact of explicit 3D decoding and kernel normalization.

### 4.4 Evaluation Metrics

We report standard reconstruction metrics including PSNR, SSIM, and LPIPS. Higher PSNR and SSIM indicate better reconstruction quality, while lower LPIPS corresponds to improved perceptual similarity.

### 4.5 Results and Analysis

We provide qualitative visual comparisons in Fig. 2 to illustrate the behavior of our method under both indoor and outdoor scenes. Compared to the baseline, normalization leads to sharper and more coherent structures. This suggests that the model is able to more effectively aggregate information from neighboring regions, rather than relying on only 2D predictions. Without normalization, the sparse convolution suffer from unstable neighboring weighting, resulting in blurry reconstructions. In contrast, normalization stabilizes the optimization process and enables the recovery of finer details.

Table 1: Ablation study for kernel weight normalization. Higher is better for PSNR/SSIM, lower is better for LPIPS.

| Method | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| Baseline: ks=1, MLP | 22.05 | 0.692 | 0.327 |
| ks=3, w/o Norm | 21.26 | 0.683 | 0.402 |
| ks=3, Norm | **23.22** | **0.744** | **0.277** |

Table 1 and Figure 3 summarize our main results.

Introducing sparse 3D convolution with kernel normalization consistently improves reconstruction quality over both the per-point MLP baseline and naive 3D convolution. In particular, kernel normalization stabilizes optimization under sparse activations and yields substantial gains in LPIPS. These results suggest that explicit 3D spatial decoding provides complementary geometric cues beyond purely 2D processing, especially in scenes with complex structure or sparse observations.

## 5 Conclusion

In this work, we investigated whether explicitly decoding intermediate representations in 3D can improve the reconstruction quality of feed-forward

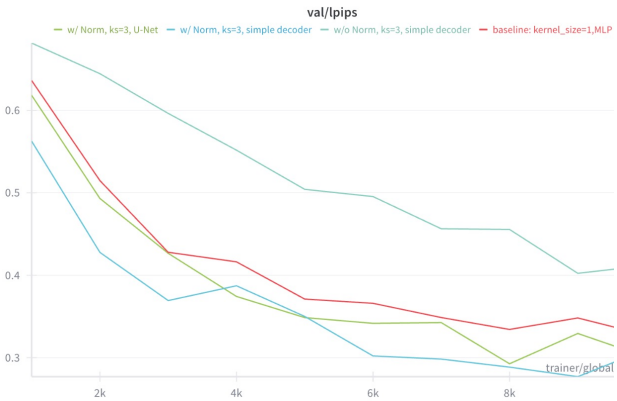


Figure 3: Validation LPIPS over training iterations for different decoder variants. From top to bottom: decoder without normalization, baseline decoder, U-Net based decoder with normalization, and simple decoder with normalization.

Gaussian Splatting pipelines. Building on the AnySplat framework and the pretrained VGGT backbone, we introduced a lightweight sparse 3D convolutional refinement module that operates on voxelized Gaussian features while preserving the feed-forward efficiency and architectural simplicity of the original pipeline.

The proposed sparse 3D ResNet-style refiner enables local spatial interaction in 3D, addressing a key limitation of per-point decoding approaches that lack explicit geometric context. To ensure stable feature aggregation under highly sparse voxel occupancy, we further introduced a lightweight kernel normalization scheme tailored for sparse 3D convolutions. Experimental results demonstrate that this design consistently outperforms both per-point MLP decoders and naive sparse 3D convolution baselines, yielding improvements in PSNR, SSIM, and perceptual quality as measured by LPIPS, particularly in scenes with complex geometry or sparse observations.

Overall, our findings suggest that incorporating explicit 3D spatial priors through sparse convolutional refinement is an effective and practical way to enhance feed-forward Gaussian Splatting methods. Importantly, these improvements are achieved with minimal architectural modification and negligible computational overhead, making the proposed approach well-suited for real-time and large-scale applications. Future work may explore deeper or adaptive sparse 3D architectures, as well as tighter integration between voxel-based refinement and Gaussian parameterization, to further improve robustness and generalization in unconstrained multi-view reconstruction scenarios.

## References

[1] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[2] Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)*, 2025. SIGGRAPH Asia 2025.

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

[4] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedřich Beneš, and Aniket Bera. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. *arXiv preprint arXiv:2403.XXXXX*, 2024.

[5] Xinyu Wang, Yifan Zhang, Zhicheng Liu, Deqing Sun, and Kun Zhou. Vggt: Visual geometry grounded transformer for feed-forward 3d reconstruction. In *CVPR*, 2025.