LaRa

LaRa:Efficient Large-Baseline Radiance Fields

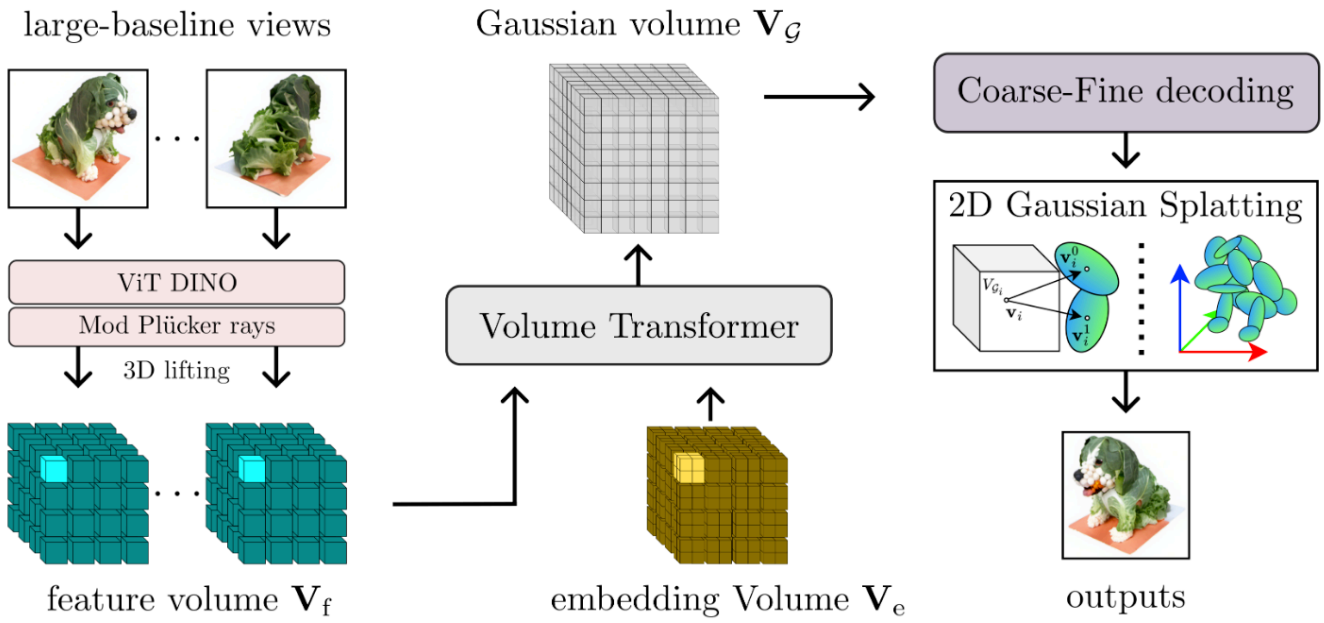

Fig. 2: Pipeline. Our method represents objects as dense voxels filled with 2D Gaussian primitives. We first construct 3D feature volumes $\mathbf{V}_\mathrm{f}$ by lifting 2D DINO features to a canonical volume, modulated by Plücker rays (Section 3.1). We then apply a volume transformer to reconstruct a Gaussian volume $\mathbf{V}_\mathcal{G}$ from the feature and embedding volumes (Section 3.2). We use a coarse-to-fine decoding process to regress 2D Gaussian primitive parameters (Section 3.3), followed by rasterization for efficient rendering.

In this work, we present *LaRa*, a feed-forward reconstruction model without the requirement of heavy training resources for the task of 360° bounded radiance fields reconstruction from unstructured few-views. The core idea of our work is to progressively and implicitly perform feature matching through a novel volume transformer. We propose a Gaussian volume as the 3D representation, in which each voxel comprises a set of learnable Gaussian primitives. To obtain the Gaussian volume from image conditions, we progressively update a learnable embedding volume by querying features in 3D. Specifically, we utilize a DINO image feature encoder to obtain image tokens and lift 2D tokens to 3D by unprojecting them into a shared canonical space. Next, we propose a novel *Group Attention Layer* architecture to enable local and global feature aggregation. Specifically, we divide dense volumes into local groups and only apply attention within each group, inspired by standard feature point matching. The grouped features and embeddings are fed to a cross-attention sub-layer to implicitly match features between feature groups of the feature volume and embedding volume, which is followed by a 3D CNN layer to efficiently share information across neighboring groups. After passing through all attention layers, the volume transformer outputs a Gaussian volume, and is then decoded as 2D Gaussian [27] parameters using a coarse-to-fine decoding process. By incorporating efficient rasterization, our method achieves high-resolution renderings.

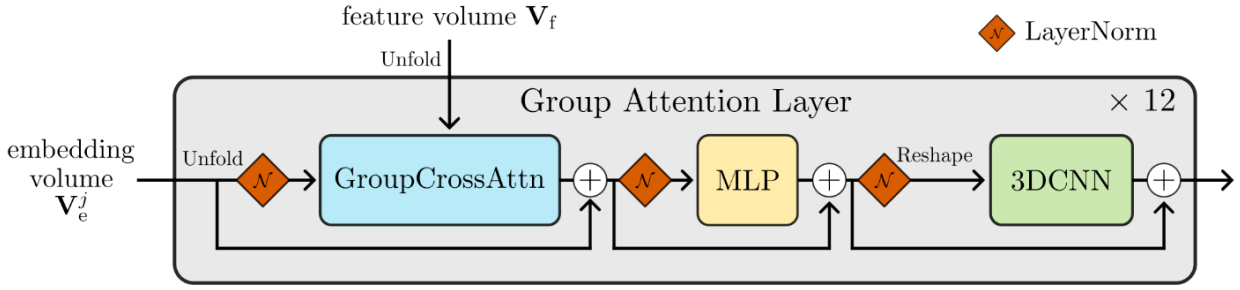

Fig. 3: Volume Transformer. We aggregate the embedding volume $\mathbf{V}_\mathrm{e}$ and feature volume $\mathbf{V}_\mathrm{f}$ through a series of Group Attention Layers that progressively match features. In each layer, the volumes are first unfolded into local groups. Subsequently, a layer normalization is applied, followed by a GroupCrossAttn sublayer. This is followed by another normalization and an MLP layer. The output is reshaped back to the original embedding volume shape, processed by a 3D convolution layer, and forwarded to the next layer. To connect the output of the sublayers, we use residual connections.

3.2 Volume Transformer

To predict the Gaussian volume, we propose a volume transformer architecture to perform attention between volumes. Self-attention and cross-attention modules, as commonly used in transformers [17], are inefficient for volumes, since the number of tokens grows cubically with the resolution of the 3D representation. Naïve applications thus result in long training times and large GPU memory requirements. In addition, geometry constraints and regional matching play crucial roles in the context of 3D reconstruction, which should be considered in the attention design.

We now present our novel volume transformer containing a set of *group attention layers* that progressively update the embedding volume. Our group attention layers contain three sublayers (see Figure 3): group cross-attention, a multi-layer perceptron (MLP), and 3D convolution. Given the image feature volume and embedding volume, we first unfold these 3D volumes (i.e., $\mathbf{V}_{\text{f}}$ and $\mathbf{V}_{\text{e}}$) into G groups along each axis. We then apply a cross-attention layer between the corresponding groups of embedding tokens $\mathbf{V}_{\text{e}}^{g,j}$ and image feature tokens $\mathbf{V}_{\text{f}}^{g}$, where j denotes the index of the layer starting from 1, and $\{\mathbf{V}_{\text{e}}^{g,1}\}_g = \mathbf{V}_{\text{e}}$. Figure 2 illustrates the unfolding of $G = 4$ and highlights the corresponding groups.

The next sublayer is an MLP, similar to the original transformer [29,48,62]. The updated embedding groups $\{\ddot{\mathbf{V}}_{\text{e}}^{g,j}\}_{g=1}^{G}$ are reassembled into the original volume shape, resulting in $\ddot{\mathbf{V}}_{\text{e}}^{j}$, which are subsequently processed by a 3D convolutional layer to share information between groups and enable the intra-model connections within the spatially organized voxels. In summary,

$$\dot{\mathbf{V}}_{\mathrm{e}}^{g,j} = \mathrm{GroupCrossAttn}\left(\mathrm{LN}\left(\mathbf{V}_{\mathrm{e}}^{g,j}\right), \mathbf{V}_{\mathrm{f}}^{g}\right) + \mathbf{V}_{\mathrm{e}}^{g,j}, \tag{2}$$

$$\ddot{\mathbf{V}}_{\mathrm{e}}^{g,j} = \mathrm{MLP}\left(\mathrm{LN}\left(\dot{\mathbf{V}}_{\mathrm{e}}^{g,j}\right)\right) + \dot{\mathbf{V}}_{\mathrm{e}}^{g,j}, \tag{3}$$

$$\mathbf{V}_{\mathrm{e}}^{j+1} = \mathrm{3DCNN}\left(\mathrm{LN}\left(\ddot{\mathbf{V}}_{\mathrm{e}}^{j}\right)\right) + \ddot{\mathbf{V}}_{\mathrm{e}}^{j}. \tag{4}$$

To incorporate information from multiple views, we flatten and concatenate the image feature tokens from multi-view feature volumes. It is important to note that different groups are processed simultaneously by the group attention layer across the batch dimension. This parallel processing allows for a larger training batch size within the attention sublayer, reducing the number of training steps required. In addition, using a 3D convolution layer increases inference efficiency compared to the popular self-attention layer. Also, we also apply layer norms $\mathrm{LM}(\cdot)$ between the sub-layers. Finally, the output embedding volume $\mathbf{V}_{\mathrm{e}}^{j}$ serves as input for the subsequent $(j+1)^{\mathrm{th}}$ group attention layer.

After passing through all (12 in our experiments) group attention layers, we use a 3D transposed CNN to scale up the updated embedding volume $\dot{\mathbf{V}}_{\mathrm{e}}$, $\mathbf{V}_{\mathcal{G}} = \text{Transpose-3DCNN}\left(\dot{\mathbf{V}}_{\mathrm{e}}\right)$. Now we have a Gaussian volume $\mathbf{V}_{\mathcal{G}}$, each Gaussian voxel is a 1-D feature vector $\mathbf{V}_{\mathcal{G}}^{i} \in \mathbb{R}^{1\times B}$, representing the primitives associated with the voxel.

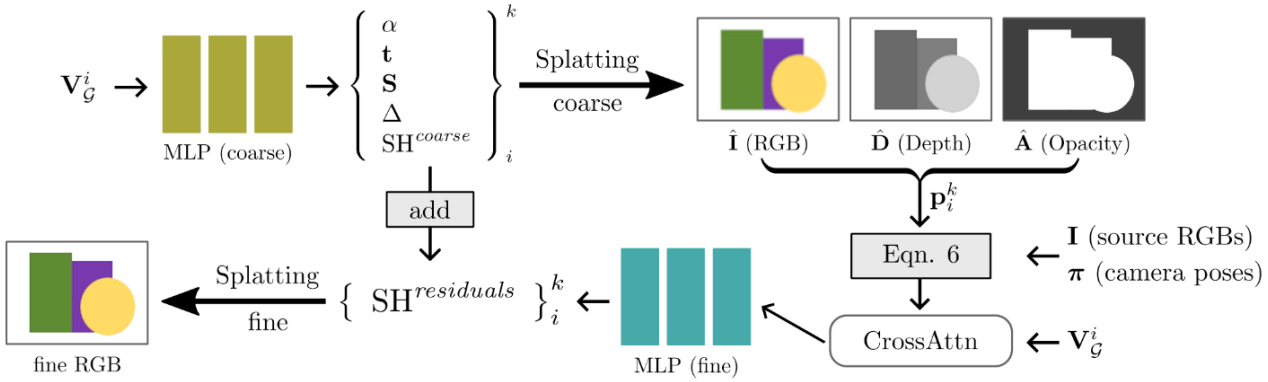

Fig. 4: Coarse-fine decoding. Top row: A "coarse" decoding module transforms the voxel features $\mathbf{V}_{\mathcal{G}}^{i}$ into K 2D Gaussian parameters, representing shape (specifically, $\alpha, \mathbf{t}, \mathbf{S}, \Delta$) and appearance (denoted as SH^{coarse}). This step is followed by a splatting procedure. On the bottom, a "fine" decoding module aggregates rendering buffers (i.e., RGB, depth, and alpha maps) from the coarse module, volume feature, and source images for appearance enhancement. It projects the centers of primitives onto these buffers, applies cross-attention with the voxel features $\mathbf{V}_{\mathcal{G}}^{i}$, and produces residual spherical harmonics $\mathrm{SH}^{residuals}$. These residuals are added to the coarse spherical harmonics for a refined splatting process.

3.3 Coarse-Fine Decoding

We obtain 2D Gaussian primitive shape and appearance parameters from the Gaussian volume, so we introduce a coarse-fine decoding process to better recover texture details. Instead of using a single network and sampling scheme to reason about the scene, we simultaneously optimize two decoding modules: one "coarse" and one "fine". Intuitively, the "fine" decoding module attempts to learn a geometry-aware texture blending process based on multi-view images, primitive features, and rendering buffers from the coarse module.

For the "coarse" decoding module, we feed Gaussian volume features to a lightweight MLP and output a set of K Gaussian parameters per voxel. We employ the efficient 2D splatting technique [27] to form high-resolution renderings, including RGB, depth, opacity, and normal maps. During training, we render M input views and M novel views for supervision.

Despite the fact that the coarse renderings can already provide faithful depths/geometries, the appearance tends to be blurred, as shown in (e) of Figure 7. This is because the image texture can easily be lost after the DINO encoder and the Group Attention layers. To address this problem, we propose a "fine" decoding module to guide the fine texture prediction.

Specifically, we project the primitive centers $\mathbf{p}_i^k$ onto the coarse renderings (i.e., RGB image $\hat{\mathbf{I}}$, depth image $\hat{\mathbf{D}}$, and accumulation alpha map $\hat{\mathbf{A}}$) to contain the coarse renderings for each primitive using the camera poses $\boldsymbol{\pi}$,

$$\mathcal{X}_{\mathbf{p}_i^k} = \left(\mathbf{I}_{\mathbf{p}_i^k}, \hat{\mathbf{I}}_{\mathbf{p}_i^k}, \hat{\mathbf{D}}_{\mathbf{p}_i^k}, \hat{\mathbf{A}}_{\mathbf{p}_i^k}\right) = \boldsymbol{\Phi}\left(\mathcal{P}\left(\mathbf{p}_i^k, \boldsymbol{\pi}\right), \oplus\left[\mathbf{I}, \hat{\mathbf{I}}, \hat{\mathbf{D}}, \hat{\mathbf{A}}\right]\right), \tag{5}$$

where $\mathcal{P}$ denotes the point projection, $\oplus$ is a concatenation operation along the channel dimension, and $\boldsymbol{\Phi}$ is a bilinear interpolation in 2D space.

In practice, the depth features can change significantly in different scenes. To mitigate scaling discrepancies, we replace the rendering depth $\hat{\mathbf{D}}_{\mathbf{p}_i^k}$ with a displacement feature $\left|\hat{\mathbf{D}}_{\mathbf{p}_i^k} - z_{\mathbf{p}_i^k}\right|$ that compares the rendered depth for input views and the depth $z_{\mathbf{p}_i^k}$ of a primitive, allowing for occlusion-aware reasoning.

We then apply a point-based cross-attention layer to establish relationships between the features of a point $\mathcal{X}_{\mathbf{p}_i^k}$ and the primitive voxel. The results of this cross-attention process are then fed into an MLP, which is tasked with predicting the residual spherical harmonics

$$\mathrm{SH}_{i,k}^{residuals} = \mathrm{MLP}\left(\mathrm{CrossAttn}\left(\mathcal{X}_{\mathbf{p}_i^k}, \mathbf{V}_{\mathcal{G}}^i\right)\right), \tag{6}$$

$$\mathrm{SH}_{i,k}^{fine} = \mathrm{SH}_{i,k}^{coarse} + \mathrm{SH}_{i,k}^{residuals}. \tag{7}$$

Furthermore, both coarse and fine modules are differentiable and updated simultaneously. Thus, the fine renderings can further regularize the coarse predictions.

Splatting. Our work takes advantage of Gaussian splatting [27, 32] to facilitate efficient high-resolution image rendering. We follow the original rasterization process and further output depth and normal maps by integrating the z value and the normal of the primitives.