ARBI3R: ARBITRARY-SCALE 3D SUPER-RESOLUTION IN FEED-FORWARD NOVEL VIEW SYNTHESIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Feed-forward novel view synthesis (NVS) is fast and generalizable, but renders at a fixed, low resolution. Recent feed-forward 3D super-resolution (SR) lifts this limit, yet predicts a fixed set of explicit 3D Gaussians, and trains a separate model for each integer scale. Arbitrary-scale SR otherwise exists only in 2D image space or in per-scene 3D NVS. None of the existing methods are at once generalizable and arbitrary-scale. We present Arbi3R, the *first* feed-forward (FF) framework to treat arbitrary-scale 3D-SR in FF NVS. For this, we replace the explicit Gaussians with an Implicit Function Decoder (IFD) that renders the novel views as a continuous function of pixel coordinates and scales. So, one trained model produces an output at any resolution from sparse, LR input views, including scales unseen at training. A Multi-View Geometry Encoder (MVGE) in our Arbi3R encodes the inputs into scene tokens, and the IFD decodes the target views through its scale-aware Coarse View Renderer (CVR) that produces an LR coarse view, an Epipolar Dual-Stream Detail Aggregator (EDDA) that recovers view-consistent high-frequency details from the source views along epipolar lines, and a Coordinate-Modulated Neural Operator (CMNO) for continuous upsampling. On RealEstate10K, ACID and DL3DV, our Arbi3R outperforms the latest FF 3D-SR specialists (SRSplat, SR3R) *by 0.8 to 2.2 dB in PSNR*, at their supported fixed scales, while uniquely spanning continuous and unseen resolutions. Arbi3R code will be released upon acceptance.

1 INTRODUCTION

Synthesizing novel views of a scene from a sparse set of posed images is a long-standing problem in vision and graphics. While per-scene optimization methods such as NeRF (Mildenhall et al., 2020; Barron et al., 2021) and 3D Gaussian Splatting (Kerbl et al., 2023) achieve photorealistic results, they require minutes to hours of optimizing per scene. Feed-forward (FF) methods instead perform reconstruction in a single network pass, predicting a scene representation directly from the input views (Charatan et al., 2024; Chen et al., 2024). With geometry-grounded backbones such as VGGT (Wang et al., 2025), DA2 (Yang et al., 2024), DUSt3R (Wang et al., 2024), and MASt3R (Leroy et al., 2024), these single-pass pipelines are becoming increasingly practical (Xu et al., 2025b; Ye et al., 2024). A recurring bottleneck is that the output resolution is tied to the input resolution. To keep the multi-view backbone and renderer tractable, FF NVS operates on sparse, LR input views, so the novel views are limited to the same low resolution. Decoding more output pixels does not recover finer details, because rendering is not SR without a learned high-frequency prior. Existing 3D super-resolution (3D-SR) methods are either per-scene or FF. Per-scene 3D-SR (Wang et al., 2022) can reach arbitrary scales (Zeng et al., 2026), but requires test-time optimization, and is therefore not generalizable. On the other hand, FF 3D-SR (Hu et al., 2026; Feng et al., 2026) is generalizable, but its models must be separately trained to fixed scale factors (e.g. $\times 4$). To the best of our knowledge, none of the existing FF 3D-SR methods can support arbitrary-scale SR due to their explicit 3D representations in a single pass.

Supporting arbitrary scales in FF 3D-SR would require an FF network to predict a different number of primitives for each scale in a single pass. Instead, the FF 3D-SR methods such as SRSplat and SR3R design their densification and primitive-refinement modules for one fixed target scale factor. We instead propose **Arbi3R**, the *first framework to treat arbitrary-scale 3D-SR in FF NVS*, using an Implicit Function Decoder (IFD) (Figure 1). Inspired by continuous image representations in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

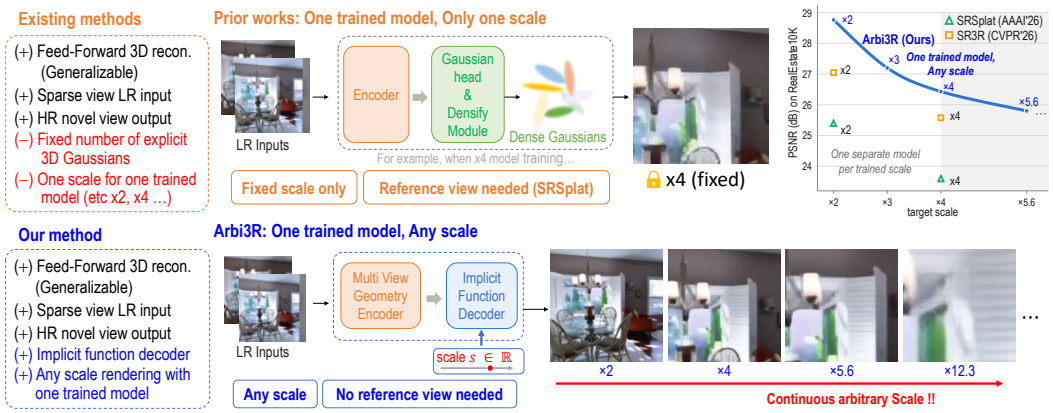

Figure 1: Comparison of our Arbi3R against previous models. The previous FF 3D-SR methods (SRSplat, SR3R) predict a fixed set of explicit 3D Gaussians and are trained as a separate model for each integer scale. Arbi3R is a single model for arbitrary scales that replaces this with an Implicit Function Decoder that renders the novel view as a continuous function of pixel coordinates and scales. The right plot reports PSNR on RealEstate10K across scales, where Arbi3R spans the continuous axis and outperforms the fixed-scale baselines at $\times 2$ (28.76 dB) and $\times 4$ (26.43 dB) scales.

2D SR (Chen et al., 2021; Wei & Zhang, 2023), the IFD bypasses primitive prediction entirely, and decodes the target views as a continuous function of pixel coordinate $\mathbf{x}$ and query cell c. The Multi-View Geometry Encoder (MVGE) produces scene tokens, and the Coarse View Renderer (CVR) renders a scale-aware LR base. The Epipolar Dual-Stream Detail Aggregator (EDDA) uses (i) high-frequency details and underlying 3D structure sampled from the source input views, and (ii) 3D-aware features from MVGE along epipolar lines. The Coordinate-Modulated Neural Operator (CMNO) decodes the continuous, arbitrary-scale output. Because the decoder is natively continuous and trained across continuous scales, one trained model can instantly reconstruct a novel view at any requested scale, including unseen scales at training. Our contributions are summarized as:

- **(C1) Continuous, arbitrary-scale FF 3D-SR.** Arbi3R is the *first 3D-SR framework* that is both *feed-forward* and *arbitrary-scale* (Table 1), realized by an Implicit Function Decoder that bypasses explicit primitive prediction and decodes the target views as a continuous function. So, one trained model serves 3D-SR at any continuous scale.
- **(C2) Epipolar Dual-Stream Detail Aggregation (EDDA).** This is a geometry-guided module that fuses (i) 3D-aware features from a geometry foundation model and (ii) fine-grained color texture from the input view images, which is essential to enhance high-frequency details and 3D geometric structures for target views. These two complementary source-view streams are obtained along deformable epipolar lines for a target view.
- **(C3) End-to-end scale conditioning.** Continuous target scales are conditioned into the Implicit Function Decoder with CVR, EDDA and CMNO. In the CVR, scale-aware Integrated Positional Encoding (IPE) (Barron et al., 2021) keeps coarse views alias-free at any resolution, and the CMNO conditions its continuous upsampling on the same scale.
- **(C4) State-of-the-art (SOTA) results.** Arbi3R sets a new SOTA in FF 3D-SR, *significantly* outperforming the latest baselines by 0.8 to 2.2 dB PSNR on RealEstate10K, ACID, and DL3DV, and uniquely supports continuous and unseen scales.

2 Related Work

FF 3D reconstruction and NVS. Generalizable FF methods predict a scene representation or rendered views directly from input images in a single pass, removing per-scene optimization (Yu et al., 2021; Wang et al., 2021; Charatan et al., 2024; Chen et al., 2024). Current pipelines rely on strong multi-view features from geometry-grounded backbones (Xu et al., 2025b; Ye et al., 2024; Zhang et al., 2025). We build our reconstructor on a recent geometry foundation model, VGGT (Wang et al., 2025), and focus on the continuous-resolution decoding.

Method family	Generalizable	SR	Arb.-scale
Per-scene 3D NVS (NeRF, 3DGS)	×	×	×
Feed-fwd 3D NVS (pixelSplat, MVSplat)	✓	×	×
Feed-fwd 3D-SR (SRSplat, SR3R)	✓	✓	×
Arbi3R (Ours)	✓	✓	✓

Table 1: Comparison of Arbi3R and other NVS methods. Arbi3R is the only single FF model that supports generalizable NVS and SR at arbitrary scales.

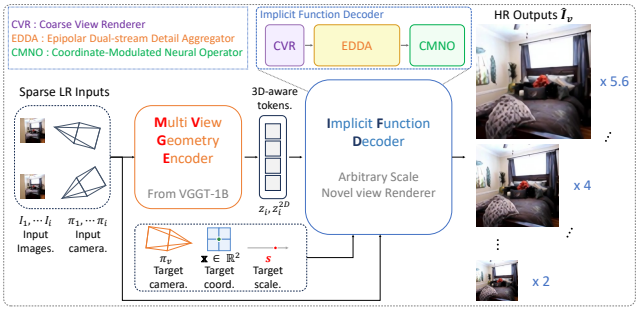

Figure 2: The Arbi3R pipeline. The MVGE encodes the sparse LR views into geometry-aware scene tokens, and the IFD decodes the target view through the CVR, EDDA, and CMNO as a continuous function of the target coordinate and scale, so a single model renders any output resolution.

Implicit FF NVS. These works decode target views with implicit decoders that render pixels from multi-view tokens without building explicit 3D primitives, e.g. SRT (Sajjadi et al., 2022), LVSM (Jin et al., 2024) and LagerNVS (Szymanowicz et al., 2026). These implicit decoders are efficient and use little explicit geometry, but render at a fixed target resolution.

Arbitrary-scale image SR. Arbitrary-scale SR decoders such as Meta-SR (Hu et al., 2019), LIIF (Chen et al., 2021), LTE (Lee & Jin, 2022), SRNO (Wei & Zhang, 2023), and CiaoSR (Cao et al., 2023) predict an output at any query coordinate from a feature grid. So, they are not tied to a fixed scale. These operate in 2D image space and are not multi-view consistent. We adopt the neural-operator formulation (Li et al., 2020) of SRNO, but condition it on 3D epipolar evidence.

3D-SR. Recently, some per-scene methods support 3D-SR using a radiance field (Wang et al., 2022; Huang et al., 2023) or 3D GS at arbitrary scale (Zeng et al., 2026), but require minutes to hours of optimization per scene, and degrade under sparse input. Recent FF 3D-SR works, SRSplat (Hu et al., 2026) and SR3R (Feng et al., 2026), are generalizable and view-consistent, but are separately trained for different scale factors. SRSplat additionally requires an HR reference view. In contrast, Arbi3R is FF and supports arbitrary-scale 3D-SR in a single model.

3 Method

3.1 Overview

Figure 2 illustrates the overview of our Arbi3R. Given V sparse low-resolution (LR) posed source views $\mathcal{I}_s = \{I_i\}_{i=1}^{V}$ ($I_i \in \mathbb{R}^{3\times H_{lr}\times W_{lr}}$), Arbi3R renders a novel target view at an arbitrary resolution. We learn a continuous view function $\hat{I}_v[\mathbf{x}] = f(\mathbf{x}, c \mid \cdot)$ that maps a target pixel coordinate $\mathbf{x} \in \mathbb{R}^2$ and a query cell $c = (c_h, c_w) \in \mathbb{R}^2$ to an RGB value $\hat{I}_v[\mathbf{x}]$. The cell c indicates a normalized footprint (cell size) of one output pixel. A scale factor $s = (s_h, s_w)$ over the coarse grid $H_{lr} \times W_{lr}$ of LR input views gives an output of $s_h H_{lr} \times s_w W_{lr}$ pixels, so $c = \big(2/(s_h H_{lr}),\ 2/(s_w W_{lr})\big)$ shrinks as s grows and a single model renders any resolution. The model is an encoder–decoder. A Multi-View Geometry Encoder (MVGE, $\mathcal{E}$) produces scene tokens, and the Implicit Function Decoder (IFD, $\mathcal{D}$, which realizes f) learns the function through three stages: (i) a Coarse View Renderer (CVR, $\mathcal{R}$), (ii) an Epipolar Dual-Stream Detail Aggregator (EDDA, $\mathcal{A}$), and (iii) a Coordinate-Modulated Neural Operator (CMNO, Φ). Arbi3R is trained in two phases: $\mathcal{E}$ and $\mathcal{R}$ are first trained for LR NVS, then $\mathcal{R}$, $\mathcal{A}$ and Φ are trained while $\mathcal{E}$ is frozen for arbitrary-scale 3D-SR.

3.2 Multi-View Geometry Encoder (MVGE, $\mathcal{E}$)

A VGGT-1B backbone (Wang et al., 2025) with a camera-token MLP and a connector of channel dimension C encodes the sparse input views into 3D-aware scene tokens, conditioned on camera

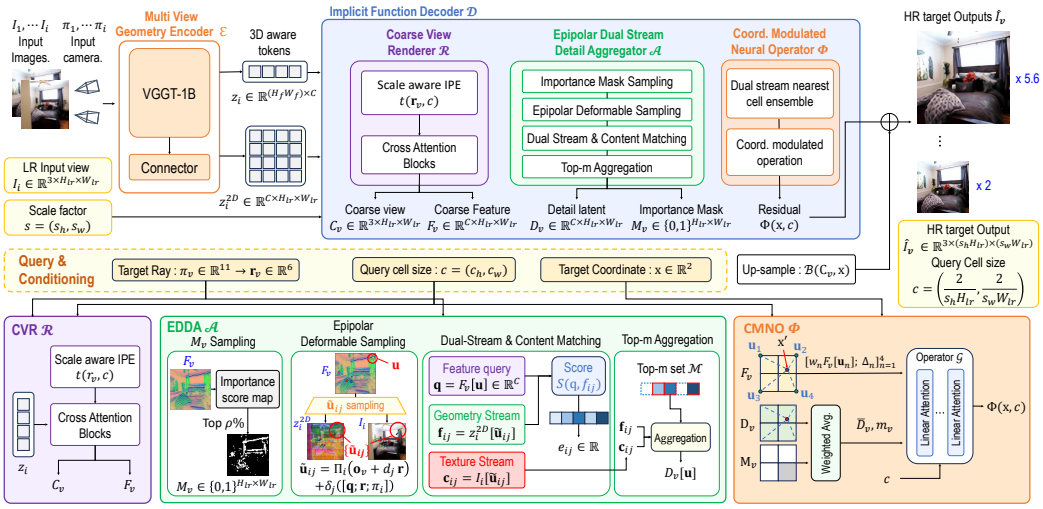

Figure 3: Architecture of Arbi3R. **Top** - MVGE ($\mathcal{E}$) encodes the sparse LR views into scene tokens, and IFD ($\mathcal{D}$) renders the target view through the CVR ($\mathcal{R}$), EDDA ($\mathcal{A}$), and CMNO (Φ). **Bottom** - three decoding stages are constructed in a row as $\mathcal{R} \to \mathcal{A} \to \Phi$. Details are in Section 3.

tokens $\pi_i \in \mathbb{R}^{11}$ as (Szymanowicz et al., 2026):

$$\left(z_i,\ z_i^{\mathrm{2D}}\right) = \mathcal{E}\left(\mathcal{I}_s, \{\pi_i\}\right). \tag{1}$$

The sequence $z_i \in \mathbb{R}^{(H_f W_f)\times C}$ is the cross-attention condition for the CVR, and the same tokens reshaped and resized into $z_i^{\mathrm{2D}} \in \mathbb{R}^{C\times H_{lr}\times W_{lr}}$ are the source geometry stream for the EDDA.

3.3 Implicit Function Decoder (IFD, $\mathcal{D}$)

$\mathcal{D}$ decodes $\hat{I}_v[\mathbf{x}] = f(\mathbf{x}, c \mid \{z_i\}, \mathbf{r}_v, \{\pi_i\}, \mathcal{I}_s)$ as a continuous function over target pixel coordinates $\mathbf{x}$. The target view is requested through its camera π_v, given the source cameras $\{\pi_i\}_{i=1}^{V}$, and each target pixel's Plücker ray $\mathbf{r}_v$ (Sitzmann et al., 2021) is computed from π_v. $\mathcal{D}$ learns f through three stages: $\mathcal{R} \to \mathcal{A} \to \Phi$ (Figure 3), which are defined as:

$$(C_v,\ F_v) = \mathcal{R}\left(\mathbf{t}(\mathbf{r}_v, c) \mid \{z_i\}\right), \tag{2}$$

$$(D_v,\ M_v) = \mathcal{A}\left(F_v,\ \mathbf{r}_v \mid \{z_i^{\mathrm{2D}}\},\ \mathcal{I}_s,\ \{\pi_i\}\right), \tag{3}$$

$$\hat{I}_v[\mathbf{x}] = \mathcal{B}(C_v, \mathbf{x}) + \Phi\left(\mathbf{x},\ c \mid F_v,\ D_v,\ M_v\right), \tag{4}$$

where each module takes a query (input) and a condition (scene context), as in $\mathcal{R}\left(\text{query} \mid \text{condition}\right)$. $\mathcal{B}(C_v, \mathbf{x})$ is the bilinear upsampling of coarse view C_v at $\mathbf{x}$, and F_v is the coarse feature from $\mathcal{R}$.

3.3.1 Coarse View Renderer (CVR, $\mathcal{R}$)

The renderer $\mathcal{R}$ is a stack of transformer blocks (Vaswani et al., 2017). It embeds the target ray $\mathbf{r}_v$ into a ray token and cross-attends it against the scene tokens $\{z_i\}_{i=1}^{V}$ to produce the coarse view $C_v \in \mathbb{R}^{3\times H_{lr}\times W_{lr}}$ and feature $F_v \in \mathbb{R}^{C\times H_{lr}\times W_{lr}}$. The embedding is made scale-aware by the IPE (Barron et al., 2021) which encodes $\mathbf{r}_v$ as attenuated Fourier features (Tancik et al., 2020). Then, the encoded ray γ_l is computed as:

$$\gamma_l(\mathbf{r}_v) = \left(\sin(\omega_l \mathbf{r}_v),\ \cos(\omega_l \mathbf{r}_v)\right) e^{-\frac{1}{2}\sigma^2 \omega_l^2}, \tag{5}$$

where $\{\omega_l = 2^l \pi\}_{l=0}^{n}$ are octave frequencies and σ^2 is a cell size c-adaptive variance (see details in Appendix A). So, a larger-sized cell gives a higher σ^2 and suppresses high-frequency components, which acts as a c-adaptive anti-aliasing prior. The ray token $\mathbf{t}$ is then obtained as:

$$\mathbf{t}(\mathbf{r}_v, c) = E_{\mathrm{ipe}}\left(\mathbf{r}_v,\ \{\gamma_l(\mathbf{r}_v)\}_l,\ c\right), \tag{6}$$

where E_{ipe} is a scale-aware embedding of the ray. Scale awareness enters at the $\mathcal{R}$ through the cell c. $\mathcal{R}$ outputs both the C_v and F_v that are then fed into the EDDA and CMNO.

3.3.2 EPIPOLAR DUAL-STREAM DETAIL AGGREGATOR (EDDA, $\mathcal{A}$)

This is illustrated in the EDDA $\mathcal{A}$ block of Figure 3. $\mathcal{A}$ injects into F_v the details of two source streams: (i) the geometry stream of 3D-aware scene tokens $\{z_i^{\text{2D}}\}_{i=1}^{V}$ and (ii) the texture stream of LR input views $\mathcal{I}_s$. Let $\mathbf{q} = F_v[\mathbf{u}] \in \mathbb{R}^C$ be the query at pixel $\mathbf{u}$ of the LR grid. For efficiency, EDDA runs only on a sparse set of the top $\rho\%$ ($\rho = 12.5$) most important pixels selected by a parameter-free importance map and written into a binary mask $M_v \in \{0,1\}^{H_{lr} \times W_{lr}}$. The importance map is shown in the Sampling block of Figure 3 (details in Appendix B). On each selected pixel, the aggregator $\mathcal{A}$ runs in three stages: (i) epipolar deformable sampling, (ii) dual-stream content matching, and (iii) top-m aggregation. The sampling stage forms a world point on target ray $\mathbf{r}_v$, and projects $\mathbf{r}_v$ into each LR source stream pair $\{z_i^{\text{2D}}, I_i\}$ through the source-view projection Π_i (defined in Appendix C). We define $\mathbf{o}_v$ as the target camera center, $\mathbf{r}$ as the pixel ray direction of $\mathbf{r}_v$ and $\{d_j\}_{j=1}^{k}$ as k learned depth candidates spaced between the near and far planes. The three stages operate as:

$$\textbf{Stage 1} : \tilde{\mathbf{u}}_{ij} = \Pi_i\big(\mathbf{o}_v + d_j\,\mathbf{r}\big) + \delta_j\big([\mathbf{q}; \mathbf{r}_v; \pi_i]\big), \tag{7}$$

$$\textbf{Stage 2} : e_{ij} = \mathcal{S}(\mathbf{q}, \mathbf{f}_{ij}) = (\mathbf{W}_q\mathbf{q})^\top(\mathbf{W}_k\mathbf{f}_{ij})/\sqrt{d_h}, \tag{8}$$

$$\textbf{Stage 3} : D_v[\mathbf{u}] = M_v[\mathbf{u}]\mathbf{W}_o \sum_{(i,j)\in\mathcal{M}} \alpha_{ij} \cdot \big(\mathbf{W}_v[\mathbf{f}_{ij}; \mathbf{c}_{ij}] + \psi_{ij}\big),$$

$$\psi_{ij} = \mathbf{W}_p\big[\mathbf{f}_{ij};\, \mathbf{p}_{ij}\big], \tag{9}$$

where $[\,\cdot\,;\cdot\,]$ denotes a vector concatenation. $\tilde{\mathbf{u}}_{ij}$ and e_{ij} are the sampling location, matching score logit for i-th source view and j-th depth candidate index, respectively. δ_j is the learned offset for the j-th sample $\tilde{\mathbf{u}}_{ij}$. $\mathbf{f}_{ij}$ and $\mathbf{c}_{ij}$ are the geometry stream and texture stream from z_i^{2D} and I_i, respectively. $\mathcal{S}(\cdot,\cdot)$ is the geometry content matching function with query and key projections $\mathbf{W}_q, \mathbf{W}_k \in \mathbb{R}^{d_h \times C}$ where d_h is the head dim and C is the dim of $\mathbf{f}_{ij}$ and $\mathbf{q}$. $M_v[\mathbf{u}]$ and $D_v[\mathbf{u}]$ are the importance mask and the detail feature at location $\mathbf{u}$. $\mathcal{M}$ is a set of indices (i,j) for the top-m $\{e_{ij}\}$ and α_{ij} is the top-m weight at $(i,j) \in \mathcal{M}$. ψ_{ij} is a deformable geometry assembly and $\mathbf{p}_{ij}$ is the set of deformable offset features. $\mathbf{W}_v \in \mathbb{R}^{d_v \times (C+3)}$, $\mathbf{W}_o \in \mathbb{R}^{C \times d_v}$ and $\mathbf{W}_p \in \mathbb{R}^{d_v \times (N+1)C}$ are the value, output, and deformable projection, with value dim d_v and N deformable offsets.

Stage 1: Epipolar Deformable Sampling. This is shown in Eq. (7). For each source view i ($i = 1 \ldots V$), this stage finds a set of sampling locations $\{\tilde{\mathbf{u}}_{ij}\}_{j=1}^{k}$, from the camera token π_i containing the pose of I_i, each pixel ray $\mathbf{r}$ and query $\mathbf{q}$ at pixel $\mathbf{u}$ of F_v. At first, $\{\tilde{\mathbf{u}}_{ij}\}_{j=1}^{k}$ are initialized at the projected point by epipolar projection Π_i. δ_j is learned to shift each projected point. Since δ_j is zero-initialized, $\tilde{\mathbf{u}}_{ij}$ starts exactly on the geometric epipolar line (He et al., 2020).

Stage 2: Dual Stream & Content Matching. This is shown in Eq. (8). This stage scores all sampling locations $\{\tilde{\mathbf{u}}_{ij}\}_{i=1,j=1}^{V,k}$ by how well $\{\mathbf{f}_{ij}\}_{i=1,j=1}^{V,k}$ match with query $\mathbf{q}$ at pixel $\mathbf{u}$ of F_v. We sample two feature streams: (i) geometry stream $\mathbf{f}_{ij} = z_i^{\text{2D}}[\tilde{\mathbf{u}}_{ij}] \in \mathbb{R}^C$ and (ii) texture stream (RGB color) $\mathbf{c}_{ij} = I_i[\tilde{\mathbf{u}}_{ij}] \in \mathbb{R}^3$ at the same sampling location $\tilde{\mathbf{u}}_{ij}$. $\mathcal{S}(\cdot,\cdot)$ compares the query $\mathbf{q}$ with the key $\mathbf{f}_{ij}$, so $e_{ij} \in \mathbb{R}$ is their matching score.

Stage 3: Top-m aggregation. This is shown in Eq. (9). This stage selects the top-m matches $\{e_{ij}\}$ where $(i,j) \in \mathcal{M}$, and combines the corresponding $\mathbf{f}_{ij}$ and $\mathbf{c}_{ij}$ into the detail feature $D_v[\mathbf{u}]$ at pixel $\mathbf{u}$ of F_v. The top-m weights are $\alpha_{ij} = \text{softmax}_{(i,j)\in\mathcal{M}}\{e_{ij}\}$. With α_{ij}, the detail feature $D_v[\mathbf{u}]$ is aggregated from (i) the projections of $[\mathbf{f}_{ij}; \mathbf{c}_{ij}]$ by $\mathbf{W}_v$ and (ii) the deformable geometry assembly ψ_{ij}. ψ_{ij} is built from $\mathbf{f}_{ij}$ and local features $\mathbf{p}_{ij} = \{z_i^{\text{2D}}[\tilde{\mathbf{u}}_{ij} + \xi_{ij,n}]\}_{n=1}^{N}$ taken at N learnable local deformable offsets (Dai et al., 2017; Zhu et al., 2020) $\{\xi_{ij,n}\} \in \mathbb{R}^2$ around each top-m location $\tilde{\mathbf{u}}_{ij}$. With sampling $\mathbf{p}_{ij}$ for $(i,j) \in \mathcal{M}$, EDDA can locally aggregate additional geometric features at $\tilde{\mathbf{u}}_{ij}$ and its N neighboring locations $\{\tilde{\mathbf{u}}_{ij} + \xi_{ij,n}\}_{n=1}^{N}$. Then the coarse feature F_v, detail feature $D_v \in \mathbb{R}^{C \times H_{lr} \times W_{lr}}$ and importance mask M_v are passed to the CMNO Φ.

3.3.3 COORDINATE-MODULATED NEURAL OPERATOR (CMNO, Φ)

As shown in Figure 3, Φ is the last stage that decodes the continuous, arbitrary-scale output from the coarse feature F_v and the detail feature D_v into target scales s, whose output pixel coordinate $\mathbf{x} \in \mathbb{R}^2$ and cell size c are defined in Section 3.1. For output $\hat{I}_v[\mathbf{x}]$, similar to (Chen et al., 2021), Φ samples $F_v[\mathbf{u}_n]$ and $D_v[\mathbf{u}_n]$ from the four nearest locations $\{\mathbf{u}_n\}_{n=1}^{4}$ of $\mathbf{x}'$, the corresponding

location of $\mathbf{x}$ in the LR grid. Each $\mathbf{u}_n$ is tagged with its relative coordinate $\Delta_n = \mathbf{x}' - \mathbf{u}_n$. The ensemble of D_v at $\mathbf{x}'$ is:

$$\bar{D}_v = \frac{1}{m_v} \sum_{n=1}^{4} w_n \, M_v[\mathbf{u}_n] \, D_v[\mathbf{u}_n], \tag{10}$$

where $M_v[\mathbf{u}_n]$ is the importance mask from EDDA explained in Section 3.3.2 and the w_n is a bilinear weight. The total valid weight is $m_v = \sum_{n=1}^{4} w_n \, M_v[\mathbf{u}_n]$, and the collapsed detail of $\bar{D}_v$ is set to 0 when $m_v = 0$. As in Figure 3, the operator $\mathcal{G}$ consists of linear-attention (Cao, 2021; Katharopoulos et al., 2020) blocks, which maps F_v, Δ_n, $\bar{D}_v$ and m_v to the SR RGB residual $\Phi(\mathbf{x}, c)$, conditioned on the cell size c as:

$$\Phi(\mathbf{x}, c) = \mathcal{G}\big(\big[\, w_n F_v[\mathbf{u}_n];\, \Delta_n \,\big]_{n=1}^{4};\; \bar{D}_v;\; m_v;\; c\big), \tag{11}$$

where $[\,\cdot\,]_{n=1}^{4}$ concatenates over the four neighbors. We pass m_v as a condition which tells $\mathcal{G}$ how much valid detail a coordinate received. Finally, $\Phi(\mathbf{x}, c)$ is added to $\mathcal{B}(C_v, \mathbf{x})$ that is the bilinear upsample of the coarse view C_v to yield HR target novel view $\hat{I}_v[\mathbf{x}] \in \mathbb{R}^3$ at $\mathbf{x}$.

3.4 Training

Progressive two-phase training. Learning from scratch is unstable for continuous, high-frequency 3D details from sparse views. To address this, a two-phase training approach is adopted: (i) The first phase trains MVGE with pretrained VGGT and the coarse view renderer CVR together using $\mathcal{L}_{\text{p1}}$ at the same resolution as the LR input view. (ii) The second phase freezes the MVGE and trains the IFD with CVR, EDDA and CMNO using $\mathcal{L}_{\text{total}}$ in an end-to-end manner.

Training Losses. The $\mathcal{L}_{\text{p1}}$ loss is defined as:

$$\mathcal{L}_{\text{p1}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \Big(\|C_v - I_v^{LR}\|_1 + \lambda_{\text{l}} \, \mathcal{L}_{\text{lpips}}(C_v, I_v^{LR}) \Big), \tag{12}$$

where C_v is the coarse view from CVR, I_v^{LR} is the downsampled ground-truth (GT) target view I_v for each target view $v \in \mathcal{V}$. In the second phase, following prior works (Chen et al., 2021; Wei & Zhang, 2023), for each target view $v \in \mathcal{V}$, we sample a set $\mathcal{Q}_v$ of $Q = 128^2$ random query coordinates rather than the full image grid, which keeps the training cost fixed across scales. A pixel loss $\mathcal{L}_{\text{pixel}}$ over scattered coordinates constrains each pixel in isolation, gives no signal on local structures, and tends to over-smooth fine details. To address this, a patch loss $\mathcal{L}_{\text{patch}}$ is proposed which helps maintain local structures and fine details. For each target view $\hat{I}_v$, we sample one random $p \times p$ patch on a spatial window Ω_v and supervise it with perceptual loss (Johnson et al., 2016) $\mathcal{L}_{\text{lpips}}$ (Zhang et al., 2018) and ℓ_1 terms. The $\mathcal{L}_{\text{total}}$ used in the second phase is

$$\mathcal{L}_{\text{total}} = \lambda_{\text{pixel}} \, \mathcal{L}_{\text{pixel}} + \lambda_{\text{patch}} \, \mathcal{L}_{\text{patch}}, \tag{13}$$

$$\mathcal{L}_{\text{pixel}} = \frac{1}{|\mathcal{V}| \, Q} \sum_{v \in \mathcal{V}} \sum_{\mathbf{x} \in \mathcal{Q}_v} \|\hat{I}_v[\mathbf{x}] - I_v[\mathbf{x}]\|_2^2, \tag{14}$$

$$\mathcal{L}_{\text{patch}} = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \Big(\mathcal{L}_{\text{lpips}}(\hat{P}_v, P_v) + \lambda_{\text{c}} \, \|\hat{P}_v - P_v\|_1 \Big), \tag{15}$$

where the predicted patch $\hat{P}_v = \hat{I}_v|_{\Omega_v}$ and GT patch $P_v = I_v|_{\Omega_v}$ are the matching $p \times p$ crops on a random spatial window Ω_v, with $p = 64$. Both the pixel set and the patch are supervised pointwise and patchwise, so $\mathcal{L}_{\text{total}}$ does not require rendering the full view.

4 Experiments

4.1 Experimental Setup

Our main comparison trains and evaluates on RealEstate10K (RE10K) (Zhou et al., 2018) and ACID (Liu et al., 2021), with each model trained on the dataset it is evaluated on. On RE10K and ACID, we follow the official train–test splits used in prior FF NVS works (Charatan et al.,

Method	Venue	RealEstate10K ×2 PSNR↑	RealEstate10K ×2 SSIM↑	RealEstate10K ×2 LPIPS↓	RealEstate10K ×4 PSNR↑	RealEstate10K ×4 SSIM↑	RealEstate10K ×4 LPIPS↓	ACID ×2 PSNR↑	ACID ×2 SSIM↑	ACID ×2 LPIPS↓	ACID ×4 PSNR↑	ACID ×4 SSIM↑	ACID ×4 LPIPS↓
SRSplat (Hu et al., 2026)	AAAI'26	25.40	0.842	0.131	23.60	0.782	0.199	28.29	0.836	0.137	25.31	0.736	0.230
SR3R (Feng et al., 2026)	CVPR'26	27.08	0.878	0.107	25.59	0.813	0.191	28.87	0.862	0.124	26.63	0.753	0.231
Arbi3R (Ours)	**Ours**	**28.76**	**0.906**	**0.085**	**26.43**	**0.838**	**0.173**	**30.51**	**0.885**	**0.103**	**27.64**	**0.790**	**0.209**

Table 2: Main in-domain comparison on RE10K and ACID at $\times 2$ and $\times 4$. SRSplat and SR3R are reproduced and separately trained for each scale factor. More Arbi3R results are in Appendix F.

Method	Venue	DL3DV (In distribution scales) ×2 PSNR↑	×2 SSIM↑	×2 LPIPS↓	×3 PSNR↑	×3 SSIM↑	×3 LPIPS↓	×4 PSNR↑	×4 SSIM↑	×4 LPIPS↓	DL3DV (Out of distribution scales) ×5.6 PSNR↑	×5.6 SSIM↑	×5.6 LPIPS↓	×8 PSNR↑	×8 SSIM↑	×8 LPIPS↓
SRSplat (Hu et al., 2026)	AAAI'26	18.12	0.487	0.373	17.66†	0.472†	0.450†	17.49	0.475	0.444	17.28†	0.487†	0.485†	17.10†	0.511†	0.519†
SR3R (Feng et al., 2026)	CVPR'26	19.47	0.563	0.312	18.98†	0.537†	0.354†	18.86	0.521	0.376	18.63†	0.512†	0.418†	18.29†	0.524†	0.457†
Arbi3R (Ours)	**Ours**	**21.70**	**0.666**	**0.238**	**20.79**	**0.620**	**0.294**	**20.30**	**0.594**	**0.339**	**19.87**	**0.577**	**0.393**	**19.56**	**0.579**	**0.447**

Table 3: High-resolution test on 960P version of DL3DV. The fixed-scale baselines (SRSplat, SR3R) are trained at $\times 2$ and $\times 4$. To test them at $\times 3$, $\times 5.6$ and $\times 8$ (marked †), the $\times 2$ SR is upscaled to $\times 3$ and the $\times 4$ SR is upscaled to $\times 5.6$ and $\times 8$.

2024; Chen et al., 2024). We are also the *first* to stress-test on the 960P version of DL3DV (Ling et al., 2023), a high-resolution in-the-wild benchmark, where all models are trained on DL3DV, using 2 input views and 48 target views per scene as the test set. For zero-shot generalization, the RE10K-trained weights are evaluated on ACID and DL3DV without any fine-tuning, and the results on DTU (Jensen et al., 2014) are given in Appendix G. All experiments use two posed LR views as input. Our Arbi3R is trained at continuous scales randomly selected between $\times 1$ and $\times 4$. Each view is preprocessed by a center crop followed by a bilinear rescale, which gives 64×64 LR inputs and $\times s$ GT targets at the requested scale s. We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) at the $\times 2$ and $\times 4$ factors, at continuous intermediate factors $\times 3$ and at the out-of-distribution ($\times 5.6$, $\times 8$) factors. Our Arbi3R model trains for 100k iterations with a batch size of 8, which takes about three days on eight RTX A6000 GPUs.

4.2 Main Results

Comparison with baselines. We compare Arbi3R against the recent SOTA FF 3D-SR baselines SRSplat (Hu et al., 2026) and SR3R (Feng et al., 2026) on RE10K and ACID (Table 2) and on the DL3DV benchmark (Table 3). We reproduce both baselines and report their performance after retraining. Both baselines are limited to one fixed scale factor per model, so we train them separately at $\times 2$ and $\times 4$, while our Arbi3R is a single model that supports arbitrary scale factors. On RE10K and ACID, we report the performance of the baselines only at their trained factors. On DL3DV, they are tested at $\times 2$, $\times 3$, $\times 4$, $\times 5.6$ and $\times 8$ where $\times 2$ SR output is upsampled to $\times 3$ with the 2D SR model SRNO (Wei & Zhang, 2023), and $\times 4$ output is upsampled to $\times 5.6$ and $\times 8$ with the same model (marked † in Table 3). This upsampling is only for the fixed-scale baselines, which cannot render unseen scales. As shown in Tables 2 and 3, Arbi3R *significantly* outperforms the two baselines by at least $+1.6$ dB at $\times 2$ scales and $+0.8$ dB at $\times 4$ scales. It holds a lead at $\times 5.6$ scale and at $\times 8$ scale.

Zero-shot generalization. Table 4 shows the RE10K-trained models on the unseen ACID and DL3DV datasets without any fine-tuning. On both datasets, Arbi3R keeps the best score on every metric. Appendix G adds the object-centric DTU as a test under severe domain shift.

3D consistency. We measure multi-view consistency on DL3DV, providing 48 consecutive video frames between the two input views. Each model renders all 48 frames, and we score neighboring frames with two metrics. The warping error E_{warp} (Lai et al., 2018) warps each rendered frame to its neighbor with the optical flow of the GT video, estimated by RAFT (Teed & Deng, 2020), and measures the pixel error outside occluded regions. It shows whether the rendered content moves with the real camera motion. MEt3R (Asim et al., 2025) lifts two neighboring views to 3D with MASt3R (Leroy et al., 2024) and compares their DINO (Caron et al., 2021) features where the two views overlap. It shows whether the two views show the same scene at the feature level. As shown in Table 5, Arbi3R gives the lowest E_{warp} at both scales. Arbi3R also gives the lowest MEt3R, with

Method	×2 PSNR↑	×2 SSIM↑	×2 LPIPS↓	×4 PSNR↑	×4 SSIM↑	×4 LPIPS↓
RE10K (train) → ACID (test)						
SRSplat (Hu et al., 2026)	28.04	0.834	0.142	25.26	0.738	0.235
SR3R (Feng et al., 2026)	28.18	0.852	0.141	26.53	0.744	0.246
Arbi3R (Ours)	**30.25**	**0.879**	**0.110**	**27.47**	**0.786**	**0.225**
RE10K (train) → DL3DV (test)						
SRSplat (Hu et al., 2026)	16.01	0.393	0.457	14.81	0.355	0.539
SR3R (Feng et al., 2026)	17.87	0.512	0.378	17.11	0.468	0.451
Arbi3R (Ours)	**18.56**	**0.545**	**0.329**	**17.64**	**0.502**	**0.436**

Table 4: Zero-shot generalization on ACID and DL3DV. All methods use weights trained only on RE10K, with no fine-tuning. Arbi3R is best for all cases.

Method	×2 $E_{\text{warp}}\downarrow$	×2 MEt3R↓	×4 $E_{\text{warp}}\downarrow$	×4 MEt3R↓
SRSplat (Hu et al., 2026)	3.35	0.077	4.02	0.069
SR3R (Feng et al., 2026)	3.09	0.072	3.94	0.061
Arbi3R (Ours)	**2.54**	**0.065**	**2.85**	**0.060**

Table 5: 3D consistency comparison on DL3DV.

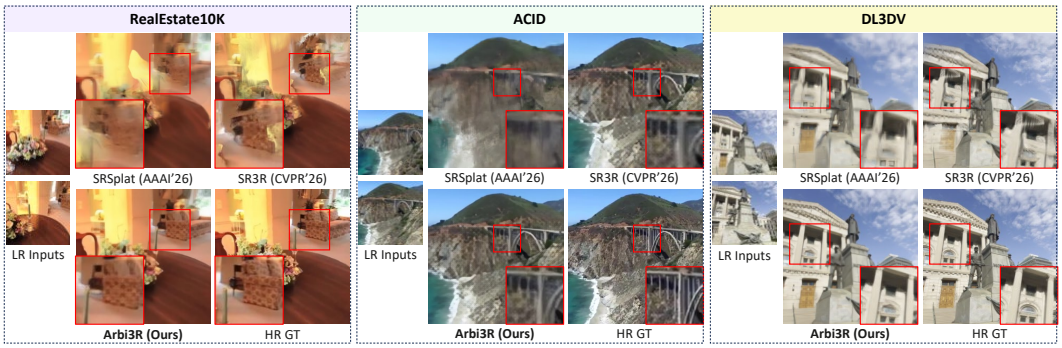

Figure 4: Qualitative comparison of SRSplat, SR3R and our Arbi3R on novel views at ×4.

a small margin over SR3R at ×4. These results show that Arbi3R keeps multi-view consistency, even though the IFD renders each target view without an explicit 3D representation.

Qualitative results. Figure 4 compares ×4 novel-view renderings on RE10K, ACID, and DL3DV. Each panel shows the LR inputs, the results of SRSplat, SR3R, Arbi3R, and the GT. The baselines reconstruct the overall scene but blur fine structure. They (i) wash out the candle and weave of the sofa on RE10K, (ii) fail to preserve the bridge's thin columns on ACID, and (iii) lose the window frames of the facade on DL3DV. In contrast, Arbi3R restores these high-frequency patterns close to the GT while keeping the scene layout intact, which matches its consistent LPIPS advantage in Tables 2 and 3. Continuous-scale renderings are provided in Appendix K.

Comparison with per-scene 3D-SR. In Table 14 of Appendix H, we also compare against per-scene 3D-SR methods on sparse-view DTU. Without any per-scene optimization, Arbi3R is the best on every metric and renders about 1000× faster.

Computation costs and efficiency. Table 15 in Appendix H compares the computation costs on RE10K. Arbi3R is the fastest and uses the least memory at every scale, although it has about 4× as many parameters as SR3R.

4.3 Ablation Study

Module contributions. In Table 6, we ablate the two components of the IFD on RE10K with the total loss in Eq. 13: (i) the EDDA and (ii) the cell size c-conditioning. EDDA recovers texture details and 3D-structure consistency of HR target views. Adding the c-conditioning injected by the IPE and the CMNO makes the renderer CVR and the CMNO scale-aware, which suppresses aliasing for a further gain. Enabling both yields our full model, which is the best on all metrics.

EDDA	c-cond.	×2 PSNR↑	×2 SSIM↑	×2 LPIPS↓	×4 PSNR↑	×4 SSIM↑	×4 LPIPS↓
		27.53	0.884	0.103	25.36	0.811	0.198
✓		28.57	0.903	0.089	26.28	0.834	0.184
✓	✓	**28.76**	**0.906**	**0.085**	**26.43**	**0.838**	**0.173**

Table 6: Ablation on EDDA and cell c modulation for RE10K. The full model enables both and is best on all metrics.

Objective	×2 PSNR↑	×2 SSIM↑	×2 LPIPS↓	×4 PSNR↑	×4 SSIM↑	×4 LPIPS↓
$\mathcal{L}_{\text{pixel}}$	28.60	0.903	0.127	26.39	0.835	0.238
$+\,\mathcal{L}_{\text{patch}}$ **(total)**	**28.76**	**0.906**	**0.085**	**26.43**	**0.838**	**0.173**

Table 7: Loss ablation on RE10K. $\mathcal{L}_{\text{pixel}} + \mathcal{L}_{\text{patch}}$ leaves fidelity almost unchanged while greatly improving LPIPS.

Method	PSNR↑ ×2	×3	×4	×5.6
DepthSplat+SRNO	25.43	24.43	24.04	23.84
ReSplat+SRNO	26.10	24.96	24.53	24.30
LagerNVS+SRNO	27.21	25.79	25.12	24.54
Arbi3R (Ours)	**28.76**	**27.20**	**26.43**	**25.80**

Table 8: Comparison with arbitrary-scale 2D cascades on RE10K. Arbi3R is best at every scale.

Method	×2 PSNR↑	×2 E_{warp} ↓	×2 MEt3R↓	×4 PSNR↑	×4 E_{warp} ↓	×4 MEt3R↓
w/o EDDA	20.01	3.32	0.076	18.65	4.08	0.068
Arbi3R (Ours)	**21.70**	**2.54**	**0.065**	**20.30**	**2.85**	**0.060**

Table 9: 3D consistency ablation on DL3DV. Removing EDDA lowers PSNR and makes both metrics worse.

Patch loss. $\mathcal{L}_{\text{total}}$ adds the patch term $\mathcal{L}_{\text{patch}}$ to the pixel loss $\mathcal{L}_{\text{pixel}}$. Table 7 shows this barely changes PSNR and SSIM but sharply lowers LPIPS at both scales, which matches its role of preserving fine local details that $\mathcal{L}_{\text{pixel}}$ alone averages out, so $\mathcal{L}_{\text{total}}$ reaches the best perceptual quality.

A cascade of 3D NVS and 2D arbitrary-scale SR. In Table 8, we test it by pairing three FF 3D reconstruction models, DepthSplat (Xu et al., 2025b), ReSplat (Xu et al., 2025a), and LagerNVS (Szymanowicz et al., 2026), a VGGT based model similar to ours, with a 2D arbitrary-scale SR model such as SRNO (Wei & Zhang, 2023). In the cascade, SRNO only sees the blurry rendered 2D LR output and cannot super-resolve it in a geometrically consistent way across different 2D LR input views. Our Arbi3R instead samples evidence directly from the input source views along epipolar lines, anchors it to the shared 3D geometry, and feeds the scale into a decoder, thus yielding the SR output with high fidelity.

EDDA and 3D consistency. Table 9 shows the comparison on DL3DV with the 3D consistency metrics of Table 5. Removing EDDA raises E_{warp} from 2.54 to 3.32 at $\times 2$ and from 2.85 to 4.08 at $\times 4$, which is also higher than the 3.94 of SR3R at $\times 4$ of Table 5. MEt3R becomes worse at both scales as well. These results suggest that the 3D consistency of Arbi3R comes from EDDA, which gathers geometrical details from the input views with the epipolar aggregation.

5 Limitations

Arbi3R needs known camera poses for its input views, so unlike pose-free methods (Ye et al., 2024; Hong et al., 2024) it cannot run on unposed images. It also builds on a pretrained geometry backbone such as VGGT, so its underlying quality is limited by that backbone. Arbi3R also fails to recover regions that the input views do not observe well, such as occluded or unseen areas.

6 Conclusion

We presented Arbi3R, the *first* FF framework that treats generalizable NVS as continuous, arbitrary-scale 3D-SR within a single model through an Implicit Function Decoder (IFD). Input source-view detail is injected by an EDDA and detail features are decoded at any target scale by a CMNO, so one trained model can produce SR output at any resolution. On RE10K, ACID, DL3DV and DTU datasets, our Arbi3R outperforms recent FF 3D-SR methods at their fixed scales ($\times 2$, $\times 4$) and uniquely reaches continuous and unseen scales. In future work, we plan to extend Arbi3R to arbitrary-scale 3D-SR from unposed LR inputs, removing its reliance on known camera poses.

AI USE STATEMENT

We used generative AI tools to help with translation, English editing, and organizing the manuscript. We reviewed and revised the AI-assisted texts for technical accuracy and for consistency with the implemented method and the reported experiments. We take full responsibility of the final manuscript, including its methodology, results, claims, and references.

REFERENCES

Mohammad Asim, Christopher Wewer, Thomas Wimmer, Bernt Schiele, and Jan Eric Lenssen. Met3r: Measuring multi-view consistency in generated images. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6034–6044. IEEE, 2025.

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5855–5864, 2021.

Jiezhang Cao, Qin Wang, Yongqin Xian, Yawei Li, Bingbing Ni, Zhiming Pi, Kai Zhang, Yulun Zhang, Radu Timofte, and Luc Van Gool. Ciaosr: Continuous implicit attention-in-attention network for arbitrary-scale image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1796–1807, 2023.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9630–9640. IEEE, 2021.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. Pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19457–19467. IEEE, 2024.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8628–8638, 2021.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*, pp. 370–386. Springer, 2024.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Xiang Feng, Yongbo He, Linxi Chen, Yan Yang, Chengkai Wang, Yifei Chen, Yixuan Zhong, Zhenzhong Kuang, Xufei Yin, Yanming Zhu, et al. Srgs: Super-resolution 3d gaussian splatting. *arXiv preprint arXiv:2404.10318*, 2024.

Xiang Feng, Xiangbo Wang, Tieshi Zhong, Chengkai Wang, Yiting Zhao, Tianxiang Xu, Zhenzhong Kuang, Feiwei Qin, Xuefei Yin, and Yanming Zhu. Sr3r: Rethinking super-resolution 3d reconstruction with feed-forward gaussian splatting. *arXiv preprint arXiv:2602.24020*, 2026.

Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pp. 7779–7788, 2020.

Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jiaolong Yang, Seungryong Kim, and Chong Luo. Unifying correspondence pose and nerf for generalized pose-free novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20196–20206, 2024.

Xinyuan Hu, Changyue Shi, Chuxiao Yang, Minghao Chen, Jiajun Ding, Tao Wei, Chen Wei, Zhou Yu, and Min Tan. Srsplat: Feed-forward super-resolution gaussian splatting from sparse multi-view images. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 4950–4958, 2026.

Xuecai Hu, Haoyuan Mu, Xiangyu Zhang, Zilei Wang, Tieniu Tan, and Jian Sun. Meta-sr: A magnification-arbitrary network for super-resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1575–1584, 2019.

Xudong Huang, Wei Li, Jie Hu, Hanting Chen, and Yunhe Wang. Refsr-nerf: Towards high fidelity and super resolution view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8244–8253, 2023.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 406–413, 2014.

Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *European conference on computer vision*, pp. 179–195. Springer, 2018.

Jaewon Lee and Kyong Hwan Jin. Local texture estimator for implicit representation function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1929–1938, 2022.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 2024.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. *arXiv preprint arXiv:2312.16256*, 3, 2023.

Andrew Liu, Ameesh Makadia, Richard Tucker, Noah Snavely, Varun Jampani, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a single image. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14438–14447. IEEE, 2021.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European conference on computer vision*, pp. 405–421. Springer, 2020.

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

Mehdi SM Sajjadi, Henning Meyer, Etienne Pot, Urs Bergmann, Klaus Greff, Noha Radwan, Suhani Vora, Mario Lučić, Daniel Duckworth, Alexey Dosovitskiy, et al. Scene representation transformer: Geometry-free novel view synthesis through set-latent scene representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6229–6238, 2022.

Vincent Sitzmann, Semon Rezchikov, Bill Freeman, Josh Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. *Advances in Neural Information Processing Systems*, 34:19313–19325, 2021.

Stanislaw Szymanowicz, Minghao Chen, Jianyuan Wang, Christian Rupprecht, and Andrea Vedaldi. Lagernvs: Latent geometry for fully neural real-time novel view synthesis. *arXiv preprint arXiv:2603.20176*, 2026.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yecong Wan, Mingwen Shao, Yuanshuo Cheng, and Wangmeng Zuo. S2gaussian: Sparse-view super-resolution 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 711–721, 2025.

Chen Wang, Xian Wu, Yuan-Chen Guo, Song-Hai Zhang, Yu-Wing Tai, and Shi-Min Hu. Nerf-sr: High quality neural radiance fields using supersampling. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 6445–6454, 2022.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P Srinivasan, Howard Zhou, Jonathan T Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4690–4699, 2021.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20697–20709, 2024.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Min Wei and Xuesong Zhang. Super-resolution neural operator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18247–18256, 2023.

Haofei Xu, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Resplat: Learning recurrent gaussian splats. *arXiv preprint arXiv:2510.08575*, 2025a.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 16453–16463, 2025b.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

Botao Ye, Sifei Liu, Haofei Xu, Xueting Li, Marc Pollefeys, Ming-Hsuan Yang, and Songyou Peng. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4578–4587, 2021.

Huimin Zeng, Yue Bai, and Yun Fu. Arbitrary-scale 3d gaussian super-resolution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 12304–12312, 2026.

Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 9869–9877, 2025.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *European conference on computer vision*, pp. 145–163. Springer, 2024.

APPENDIX

A demo video, `Arbi3R_demo.mp4`, is included with this supplementary material. It compares the novel-view renderings of Arbi3R and the two baselines along a moving camera path on RE10K, ACID, and DL3DV, and it is described in Section J. This supplementary material provides the derivation of the cell-adaptive variance σ^2 used by the scale-aware IPE, the importance map that EDDA uses to select its sparse pixel set, the source-view projection Π_i used by the epipolar sampling stage, the evaluation metrics, the training hyperparameters and architecture dimensions, the data preprocessing and evaluation protocol for each benchmark, additional quantitative results at $\times 3$ and $\times 5.6$ on RealEstate10K (RE10K) and ACID, how the baselines are reproduced, and additional continuous-scale qualitative renderings.

A DERIVATION OF THE CELL-ADAPTIVE VARIANCE

This section expands the cell-adaptive variance σ^2 introduced in the Coarse View Renderer (CVR, $\mathcal{R}$) part of Section 3.3 of the main paper, where the attenuated Fourier feature $\gamma_l(\mathbf{r}_v)$ is defined. The scale-aware Integrated Positional Encoding (IPE) in the CVR attenuates the Fourier features of the target ray $\mathbf{r}_v$ with a variance σ^2 derived from the query cell $c = (c_h, c_w)$. We treat each output pixel as a uniform distribution over its cell rather than a point. A uniform distribution over an interval of length a has variance $a^2/12$. Applying this per axis gives the axis variances $c_h^2/12$ and $c_w^2/12$. The IPE uses a single isotropic variance, so we take the geometric mean of the two,

$$\sigma^2 = \sqrt{\frac{c_h^2}{12} \cdot \frac{c_w^2}{12}} = \frac{|c_h\, c_w|}{12}, \tag{16}$$

which reduces to $c^2/12$ for a square cell of side c. A larger cell therefore gives a larger σ^2, and the attenuation $e^{-\frac{1}{2}\sigma^2\omega_l^2}$ suppresses the high-frequency bands, which acts as a c-adaptive anti-aliasing prior.

B EDDA IMPORTANCE MAP

This section defines the parameter-free importance map that produces the binary mask M_v used by the Epipolar Dual-Stream Detail Aggregator (EDDA, $\mathcal{A}$) in Section 3.3 of the main paper. EDDA runs only on a sparse set of important pixels of the coarse feature $F_v \in \mathbb{R}^{C\times H_{lr}\times W_{lr}}$, selected by this map. For each pixel $\mathbf{u}$, we score its saliency by the deviation of its feature from a local average,

$$S_v[\mathbf{u}] = \frac{1}{C}\big\|F_v[\mathbf{u}] - \mathrm{avg}_{3\times 3}(F_v)[\mathbf{u}]\big\|_1, \tag{17}$$

where $\mathrm{avg}_{3\times 3}$ is a 3×3 average pooling with unit stride and the norm is the mean absolute deviation over the C channels. A pixel whose feature departs from its neighborhood receives a high score, so the map responds to high-frequency structural detail. The scores of all pixels form the importance map S_v. We keep the top $\rho\%$ of pixels ($\rho = 12.5$) as the important set and write it into the binary mask M_v. Only these pixels are passed to the epipolar deformable sampling, dual-stream matching, and top-m aggregation stages of EDDA. Figure 5 shows the gating on one RE10K target view. The map gives high scores on the window frame, the rails of the bed frame, and the pattern of the throw, and the mask keeps these pixels while excluding the flat wall and the ceiling.

C SOURCE-VIEW PROJECTION

The epipolar deformable sampling stage of EDDA, given by Eq. (7) of the main paper, projects a depth-swept world point on the target ray into each source view i through the source-view projection Π_i, which we define here. For a target pixel with camera center $\mathbf{o}_v$ and ray direction $\mathbf{r}$, and a depth candidate d_j, the world point is $\mathbf{P}_{ij} = \mathbf{o}_v + d_j\mathbf{r}$. The source camera token π_i provides the source camera center $\mathbf{c}_i$, the camera-to-world rotation $\mathbf{R}_i$, and the horizontal and vertical field-of-view angles ϕ_i^w and ϕ_i^h. We first bring the point into the source camera frame,

$$[X,\, Y,\, Z]^\top = \mathbf{R}_i^\top\big(\mathbf{P}_{ij} - \mathbf{c}_i\big), \tag{18}$$

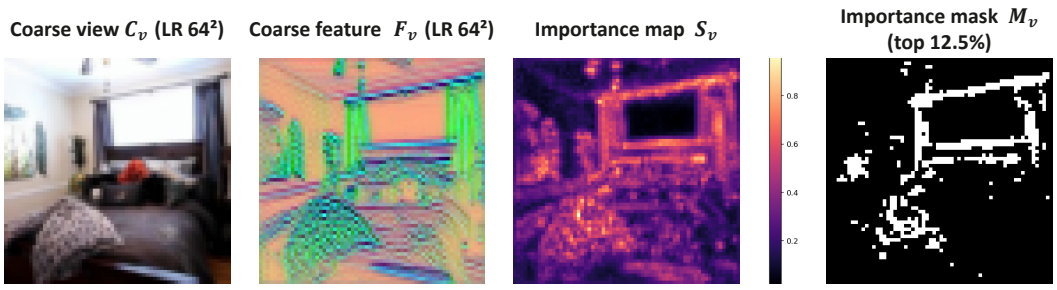

Figure 5: EDDA importance gating on one RE10K target view. From the coarse feature F_v on the LR grid, the importance map S_v scores each pixel and the top $\rho\%$ of scores become the binary mask M_v.

and then apply a perspective projection normalized by the field of view,

$$\Pi_i(\mathbf{P}_{ij}) = \Big(\frac{X/Z}{\tan(\phi_i^w/2)},\ \frac{Y/Z}{\tan(\phi_i^h/2)}\Big). \tag{19}$$

The k depth candidates $\{d_j\}_{j=1}^k$ are learned and spaced between the near and far planes, as stated in Section 3.3 of the main paper. They are held in log space and initialized uniformly in inverse depth, which places them densely near the camera and sparsely toward the far plane. The learned offset δ_j then shifts each projected location to refine the correspondence, as described in the EDDA sampling stage of Section 3.3 in the main paper.

D Visualizing EDDA Matching and CMNO Aggregation

Top-m matching in EDDA. Figure 6 visualizes the three EDDA stages of Section 3.3 of the main paper on one RE10K scene. The left column shows the importance mask M_v of Section B and the coarse feature F_v, with five selected query pixels marked in color. The middle column is Stage 1, the epipolar deformable sampling of Eq. (7). Each query is swept over its k depth candidates, and every candidate is projected into both source views through Π_i, which traces a set of sampling locations $\{\tilde{\mathbf{u}}_{ij}\}$ along the epipolar line. The right column visualizes the top-m sampling locations of Stage 2 and Stage 3, the dual-stream content matching and the top-m aggregation of Eqs. (8) and (9). Circles mark the top-m matches, drawn over the dimmed candidates that are dropped. The matches collapse from the full set of candidates onto the small region that the query pixel corresponds to in each source view. Both the geometry stream z_i^{2D} and the texture stream I_i are shown, so the selected points can be compared with the image content.

Four-neighbor aggregation in the CMNO. Figure 7 shows the neighbor geometry behind Eqs. (10) and (11) of the main paper. An output pixel coordinate $\mathbf{x}$ has a corresponding location $\mathbf{x}'$ in the LR grid, which in general does not coincide with a grid center. The four nearest locations $\{\mathbf{u}_n\}_{n=1}^4$ of $\mathbf{x}'$ are the neighbors sampled by the CMNO, and each one carries the bilinear weight w_n, the area of the cell diagonally opposite it, so a query close to one neighbor is weighted mostly toward that neighbor. The CMNO samples these four locations from the coarse feature F_v, the detail feature D_v, and the importance mask M_v. The masked weighted average over D_v gives the ensemble $\bar{D}_v$ and its total valid weight m_v of Eq. (10), and the operator $\mathcal{G}$ maps the weighted coarse features, the relative coordinates $\Delta_n = \mathbf{x}' - \mathbf{u}_n$, $\bar{D}_v$, m_v, and the cell size c to the SR RGB residual $\Phi(\mathbf{x}, c)$ of Eq. (11). Because the four neighbors and their weights are recomputed for every continuous coordinate, one LR grid supports any output resolution.

E Experimental Details

This section expands the experimental setup of Section 4.1 of the main paper, which produces Tables 2, 3, and 4 there. We describe the evaluation metrics, the training hyperparameters, the architecture dimensions, and the data preprocessing and evaluation protocol for each benchmark. All experiments take two posed LR views as input and render the remaining views as targets. All methods are evaluated on identical input views, novel-view targets, resolution crops, and test splits.

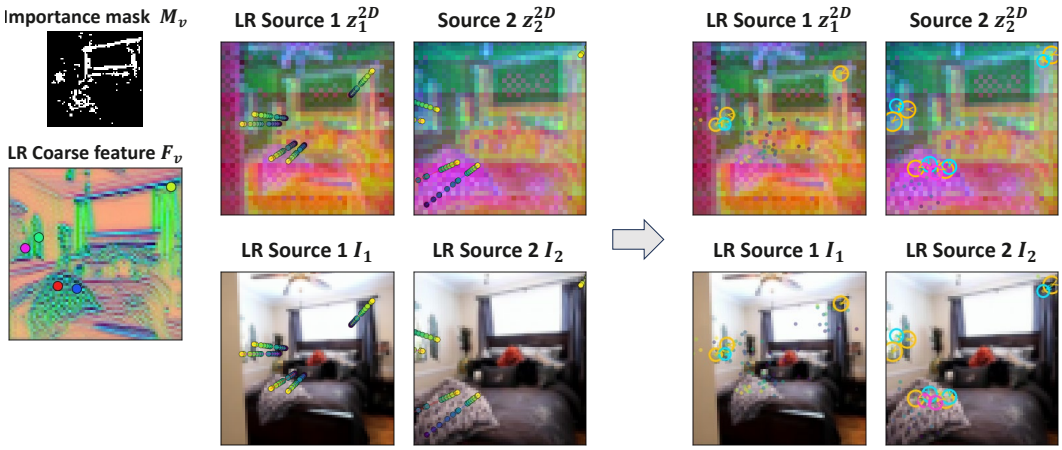

Figure 6: Epipolar deformable sampling and top-m matching in EDDA. Left, the importance mask M_v and the coarse feature F_v with query pixels. Middle, the starting locations of the sampling points $\tilde{\mathbf{u}}_{ij}$, projected into both source views. Right, the top-m matches selected by the content matching, which lie on the surface observed by the query.

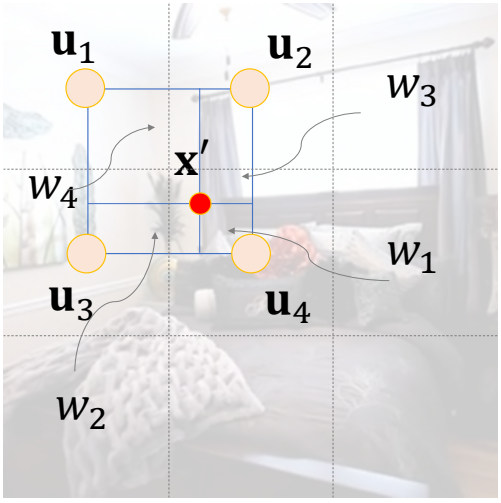

Figure 7: Four-neighbor aggregation in the CMNO. The LR location $\mathbf{x}'$ of an output coordinate falls between its four nearest locations $\{\mathbf{u}_n\}_{n=1}^4$, and each neighbor carries the bilinear weight w_n, the area of the cell diagonally opposite it. The CMNO samples F_v, D_v, and M_v at these four locations for every output coordinate.

Hyperparameter	Notation	Value
Input views	V	2 posed LR views
LR input resolution	$H_{lr} \times W_{lr}$	64×64
Training scale range	s	$[1.0,\ 4.0]$, continuous
Query coordinates per view	Q	128^2
Patch size	p	64
Importance ratio	ρ	12.5%
Optimizer		AdamW (β_1=0.9, β_2=0.95)
Weight decay		0.05
Learning rate		4×10^{-4}
LR schedule		cosine, 3000 warmup steps
Iterations		100k
Batch size		8
Pixel loss weight	λ_{pixel}	1.0
Patch loss weight	λ_{patch}	0.5
Patch ℓ_1 weight	λ_{c}	2.0

Table 10: Training hyperparameters for Arbi3R.

E.1 Evaluation Metrics

Section 4.1 of the main paper reports PSNR, SSIM, and LPIPS, which we define here. We compare each rendered novel view with its GT using these three standard metrics. PSNR measures pixel-level fidelity as $10 \log_{10}(\text{MAX}^2/\text{MSE})$, where MSE is the mean squared error between the two images and MAX is the maximum pixel value. SSIM (Wang et al., 2004) measures structural similarity from local luminance, contrast, and structure statistics. LPIPS (Zhang et al., 2018) measures perceptual similarity as a distance between deep features of the two images. We report all three because PSNR and SSIM measure reconstruction fidelity and coarse structure, while LPIPS is sensitive to the fine high-frequency texture that they often miss and that a 3D-SR model should recover.

Module	Component	Notation	Value
MVGE ($\mathcal{E}$)	Backbone		VGGT-1B
	Camera token dim	π_i	11
	Scene token dim	z_i	768
CVR ($\mathcal{R}$)	Transformer blocks		12
	Width, heads		768, 12
	Patch size at 64×64		4
	IPE octaves	n	6
EDDA ($\mathcal{A}$)	Feature dim	C	64
	Depth candidates	k	16
	Aggregated matches	m	3
	Deformable offsets	N	4
	Head dim	d_h	16
	Value dim	d_v	32
	Importance ratio	ρ	12.5%
CMNO (Φ)	Linear-attention blocks	$\mathcal{G}$	2
	Width, heads		256, 16

Table 11: Architecture dimensions of Arbi3R.

E.2 Implementation Details

This subsection expands the training setup summarized in Section 4.1 of the main paper. Our implementation refers to the architecture of LagerNVS (Szymanowicz et al., 2026) for the encoder and the coarse view renderer, and to the SR operator formulation of SRNO (Wei & Zhang, 2023) for the continuous decoder. Arbi3R is trained in two phases. Phase 1 trains the MVGE and the CVR at the LR input resolution. Phase 2 freezes the MVGE and trains the full IFD end-to-end with $\mathcal{L}_{\text{total}}$. Both phases use the same optimizer and schedule, and all models are trained on eight RTX A6000 GPUs. Table 10 lists the final hyperparameters.

E.3 Architecture Dimensions

Table 11 lists the dimensions of each module of the architecture described in Section 3 and Figure 3 of the main paper. The MVGE is a VGGT-1B (Wang et al., 2025) backbone whose per-view tokens are projected to the renderer width. The CVR is a stack of full-attention transformer blocks that reads these tokens and writes the coarse view and the coarse feature on the LR grid. The EDDA and the CMNO operate on the narrower feature width of the implicit decoder. The full model has 1003.9M parameters, of which 911.7M are the frozen VGGT-based MVGE backbone and 92.2M are the IFD that includes the CVR, EDDA and CMNO together.

E.4 Common Preprocessing

Section 4.1 of the main paper states that each view is preprocessed by a center crop followed by a bilinear rescale, and this subsection gives the full procedure. For RE10K and ACID we use the same evaluation index that prior FF NVS works release (Charatan et al., 2024; Chen et al., 2024), and for DL3DV we use the view-selection index released with LagerNVS (Szymanowicz et al., 2026). These index files fix the input and target frames of every test scene, so the evaluation is deterministic and identical for every method. All three benchmarks then share the same preprocessing steps. Each view is first center-cropped to a square that keeps the largest possible field of view, and the crop is then resized with bilinear interpolation. The LR input and the HR target are produced independently from the same crop, so the target at scale s is a $64s \times 64s$ resize of the crop while the input is a 64×64 resize of the same crop. Since the training scales s are sampled in $[1, 4]$, the $\times 5.6$ and $\times 8$ settings are out-of-distribution beyond the training range.

	$\times 3$			$\times 5.6$		
Dataset	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
RE10K	27.20	0.864	0.135	25.80	0.816	0.216
ACID	28.58	0.828	0.164	26.87	0.759	0.263

Table 12: Results of a single Arbi3R model on RE10K and ACID at the continuous intermediate factor $\times 3$ and the out-of-distribution factor $\times 5.6$. The fixed-scale baselines cannot render these factors.

E.5 RE10K

Data preprocessing. RE10K (Zhou et al., 2018) frames are stored at 640×360. A center crop gives 360×360, which is resized to the 64×64 LR input and to the HR target at the requested scale.

Evaluation protocol. The released index covers 7194 scenes. Scenes for which it defines no valid two-view pair are left out, which leaves 6474 of them as test scenes, each with 2 input views and 3 target views. The two input views are separated by 45 to 136 frames and the target views lie between them.

E.6 ACID

Data preprocessing. ACID (Liu et al., 2021) uses the same 640×360 source resolution and the same 360×360 center crop as RE10K.

Evaluation protocol. We follow the same protocol as RE10K. The released index covers 1848 scenes, which leaves 1595 of them as test scenes, each with 2 input views and 3 target views, with input-view gaps of 45 to 136 frames.

E.7 DL3DV

Data preprocessing. We use the 960P version of DL3DV (Ling et al., 2023) (DL3DV-ALL-960P), whose frames are stored at 960×540. We prepare this 960P split for arbitrary-scale evaluation. A center crop gives 540×540, which is resized to the 64×64 LR input and to the HR target at the requested scale. Because the crop is 540×540, even the $\times 8$ target at 512×512 is still a downsample of the source, so DL3DV is the only benchmark on which we report $\times 8$.

Evaluation protocol. We use the view-selection index released with LagerNVS (Szymanowicz et al., 2026), which covers 140 scenes for evaluation. Each scene uses 2 input views and 48 target views, where the input views are the first and the fiftieth frame and the targets are the 48 frames between them. This differs from the three-target protocol of RE10K and ACID, so the DL3DV results are considered separately.

F Continuous-Scale Quantitative Results

The caption of Table 2 of the main paper states that additional Arbi3R results are reported here. Table 12 gives them at the continuous intermediate factor $\times 3$ and the out-of-distribution factor $\times 5.6$ on RE10K and ACID, which complements the $\times 2$ and $\times 4$ comparison of Section 4.2 of the main paper. The fixed-scale baselines SRSplat (Hu et al., 2026) and SR3R (Feng et al., 2026) support only the factors they are trained for, so they cannot produce these scales and the table reports Arbi3R alone. All numbers come from the same single model as the main comparison, without any per-factor retraining.

G Zero-Shot Generalization on DTU

Alongside the scene-centric ACID and DL3DV in Table 4 of the main paper, we also test the RE10K-trained models on DTU, an object-centric dataset with a very different distribution of camera param-

eters, as a stress test under severe domain shift. All methods use the RE10K weights without any fine-tuning. As shown in Table 13, Arbi3R keeps the best score on every metric.

		×2			×4		
Method	Venue	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
SRSplat (Hu et al., 2026)	AAAI'26	13.12	0.397	0.406	12.84	0.328	0.494
SR3R (Feng et al., 2026)	CVPR'26	15.22	0.501	0.334	14.58	0.437	0.442
Arbi3R (Ours)	**Ours**	**17.95**	**0.619**	**0.249**	**17.41**	**0.581**	**0.396**

Table 13: Zero-shot generalization from RE10K to DTU. All methods use weights trained only on RE10K, with no fine-tuning. Arbi3R is best on every metric.

H Per-Scene 3D-SR Comparison and Computation Costs

Comparison with per-scene 3D-SR. Table 14 compares Arbi3R against per-scene 3D-SR methods on sparse-view DTU. Sparse inputs are a hard setting for per-scene optimization. We therefore pair SRGS (Feng et al., 2024) with the few-shot FSGS (Zhu et al., 2024) base, which allows the per-scene model to do a sparse-view-capable reconstruction, and we also test Arbi-3DGSR (Zeng et al., 2026) and S2Gaussian (Wan et al., 2025). Our Arbi3R uses the weights trained on RE10K, while all baselines optimize directly on each test scene. Despite not using per-scene optimization, our Arbi3R is the best on every metric, and with a single FF pass it renders about $1000\times$ faster than the baselines, measured at $\times 4$ scale on a single RTX A6000.

Computation costs and efficiency. Table 15 compares the costs of the baselines and our Arbi3R on RE10K. SRSplat additionally runs a ref-image generation step with a diffusion model (Rombach et al., 2022). This stage dominates its cost, making it more than $100\times$ slower than our Arbi3R. Overall, Arbi3R is the fastest and most memory-efficient at every scale. Although our Arbi3R has about $4\times$ as many parameters as SR3R, its runtime is faster and its memory footprint is smaller than that of SR3R, because our Arbi3R produces no explicit 3D Gaussians and avoids their buffers, while SR3R must store and rasterize many Gaussians.

	×2			×4			
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	Time (s)↓
SRGS+FSGS (ECCV'24)	12.72	0.348	0.568	12.34	0.360	0.629	157
S2Gaussian (CVPR'25)	12.94	0.344	0.569	12.53	0.342	0.632	424
Arbi-3DGSR (AAAI'26)	13.10	0.406	0.557	12.56	0.384	0.627	254
Arbi3R (Ours)	**17.95**	**0.619**	**0.249**	**17.41**	**0.581**	**0.396**	**0.13**

Table 14: Comparison against per-scene 3D-SR methods on sparse-view DTU dataset.

		×2		×4	
Method	Params (M)	VRAM (GB)↓	Time (s)↓	VRAM (GB)↓	Time (s)↓
SRSplat (Hu et al., 2026)	15820	32.27	16.67	32.27	16.67
SR3R (Feng et al., 2026)	245.6	7.29	0.42	7.43	0.42
Arbi3R (Ours)	1003.9	4.69	0.13	4.69	0.14

Table 15: Computation costs and efficiency on RE10K, measured on an RTX A6000 with two input views. Arbi3R shows the fastest per-view run time and lowest memory footprint across scales.

I Baseline Reproduction

Section 4.2 of the main paper states that both baselines are reproduced and reported after retraining. The SRSplat and SR3R numbers in Tables 2 and 3 of the main paper are therefore produced by our own reimplementation of each baseline, retrained under our protocol.

Reason for the reproduction. Each baseline paper reports results only on RE10K and ACID, and only at the fixed factors its models are trained for. Our comparison also covers the 960P version of DL3DV, a high-resolution in-the-wild benchmark built from hand-held captures with large camera gaps, which is the most common way scenes are filmed in everyday life, and on which neither SRSplat nor SR3R reports any result. Reproducing both baselines and retraining them under our protocol is therefore required to obtain a baseline result on that benchmark. A second reason is the dataset preprocessing. The LR input and the HR target are produced from the source frames, and the preprocessing pipeline sets the difficulty of the task. The two papers do not describe the preprocessing step in the same way, and neither describes it in full. Results measured under different preprocessing cannot be placed in one table, so we apply the same preprocessing pipeline of Section E.4 to Arbi3R and to both baselines, which requires retraining them.

Released code of each baseline. Neither baseline is fully released at the time of writing. The official repository of SRSplat provides part of the method. The official repository of SR3R contains no code, so we implement that method in full from its paper. In both cases we follow every architectural and training detail stated in the papers. For SR3R we adopt the DepthSplat base, which performs best in its paper.

Training of the baselines. Both baselines are fixed-scale, so we train a separate model for each of the scale factors $\times 2$ and $\times 4$ and report each model at the scale factor it was trained for. For each model we search the learning rate, the loss weights, and the training length, and select the final weights by evaluating the best checkpoint.

Every method in this paper is measured under identical conditions with our Arbi3R, so no comparison relies on a published result. Arbi3R and both baselines receive the same input views, render the same novel-view targets, use the same crops and resolutions, and are scored by the same metric code on the same test split.

J Demo Video of Qualitative Comparison

The demo video `Arbi3R_demo.mp4` compares Arbi3R with the two baselines along a moving camera path. It extends the single-frame $\times 4$ comparison of Figure 4 and Section 4.2 of the main paper to a full camera path. The video is organized into one part per dataset, with three test scenes for RE10K, three for ACID, and five for DL3DV. Every clip starts from the same two posed 64×64 input views and renders a novel-view path at $\times 4$, so each output frame is 256×256. The four panels show the two input views, SRSplat, SR3R, and our Arbi3R. All three methods follow the same camera path frame for frame.

On RE10K and ACID the methods differ in texture. In the kitchen scene, all three place the cabinets, the ceiling fan, and the counter correctly, but only Arbi3R keeps the shine and the wood grain of the polished counter in front. Both baselines wipe that surface flat. In the sea stack scene on ACID, Arbi3R again keeps the moss on the front rock, the cracks in the cliff on the right, and a clean outline where the far rock meets the sky, while both baselines smooth these surfaces out.

On DL3DV the difference is in structure, not only in texture. In the museum scene with a red tractor, Arbi3R keeps the tractor in one piece, with a round rear tire, a separate front wheel behind it, and the thin curved handrail in front unbroken on its posts. SRSplat breaks that handrail into see-through pieces and blurs the tractor body into red patches. SR3R spreads the same rail into a thick band and turns the rear tire into a dark blur. The other DL3DV scenes behave the same way. These scenes are hand-held captures with large camera gaps, so any error in the recovered structure shows up as double edges and blur that shift as the camera moves, while Arbi3R stays steady along the whole path.

K Continuous-Scale Qualitative Results

Section 4.2 of the main paper states that continuous-scale renderings are provided here. Figures 8 through 13 show novel-view renderings of a single Arbi3R model over a continuous range of output scales, with two scenes per benchmark. Each figure starts from the two 64×64 context views of one scene and decodes the same target view at $\times 1.3$, $\times 2$, $\times 4$, $\times 5.6$, $\times 8$, $\times 12.3$, and $\times 16$, which correspond to output resolutions from 83×83 to 1024×1024. Only the first three scale factors are in

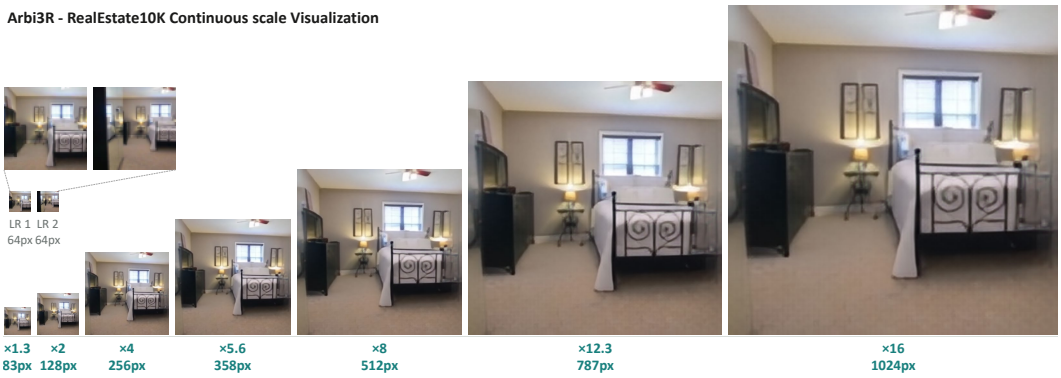

Figure 8: Continuous-scale renderings on RE10K, bedroom with a wrought-iron bed frame. From the two 64×64 context views on the left, one Arbi3R model decodes the same target view at every scale from $\times 1.3$ to $\times 16$.



Figure 9: Continuous-scale renderings on RE10K, living room with a fireplace. The mantel, the framed picture above it, and the rug weave remain resolved beyond the $\times 4$ upper end of the training range.

the training range, since scale factor s is sampled continuously from $[1.0,\ 4.0]$ during training. The four larger factors are therefore extrapolation beyond the scales seen at training. Structure continues to resolve across that boundary rather than converging to a smooth upsample of the $\times 4$ result. The bridge deck and its piers in Figure 10, the shelf edges and the labels on the boxes in Figure 12, and the scroll pattern of the bed frame in Figure 8 remain sharp at $\times 16$.



Figure 10: Continuous-scale renderings on ACID, aerial view of a causeway bridge. The deck of the bridge is one to two pixels wide in the input views and remains continuous over the whole range.



Figure 11: Continuous-scale renderings on ACID, aerial view of a cliff-top resort. The repeated gables and window openings of the building become separated as the query resolution increases.



Figure 12: Continuous-scale renderings on DL3DV, library shelving. The shelf edges and the colored boxes are dense high-frequency content, and both remain separated at $\times 16$.

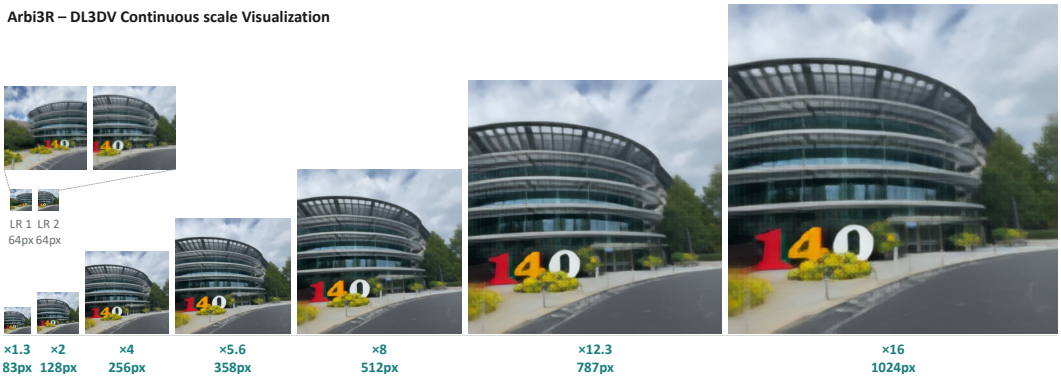

Figure 13: Continuous-scale renderings on DL3DV, curved glass office facade. The horizontal floor bands of the facade and the numerals in front of the entrance remain legible at the largest factors.