FlowR: Flowing from Sparse to Dense 3D Reconstructions

Tobias Fischer[1,2] Samuel Rota Bulò[2] Yung-Hsu Yang[1] Nikhil Keetha[2,3]
Lorenzo Porzi[2] Norman Müller[2] Katja Schwarz[2] Jonathon Luiten[2]
Marc Pollefeys[1] Peter Kontschieder[2]
[1] ETH Zürich [2] Meta Reality Labs Zürich [3] CMU
https://tobiasfshr.github.io/pub/flowr

Abstract

3D Gaussian splatting enables high-quality novel view synthesis (NVS) at real-time frame rates. However, its quality drops sharply as we depart from the training views. Thus, dense captures are needed to match the high-quality expectations of applications like Virtual Reality (VR). However, such dense captures are very laborious and expensive to obtain. Existing works have explored using 2D generative models to alleviate this requirement by distillation or generating additional training views. These models typically rely on a noise-to-data generative process conditioned only on a handful of reference input views, leading to hallucinations, inconsistent generation results, and subsequent reconstruction artifacts. Instead, we propose a multi-view, flow matching model that learns a flow to directly connect novel view renderings from possibly sparse reconstructions to renderings that we expect from dense reconstructions. This enables augmenting scene captures with consistent, generated views to improve reconstruction quality. Our model is trained on a novel dataset of 3.6M image pairs and can process up to 45 views at 540×960 *resolution (91K tokens) on one H100 GPU in a single forward pass. Our pipeline consistently improves NVS in sparse- and dense-view scenarios, leading to higher-quality reconstructions than prior works across multiple, widely-used NVS benchmarks.*

1. Introduction

3D reconstruction is the process of estimating the 3D scene geometry and appearance from a set of 2D images. Given a large and dense enough set of images, modern 3D reconstruction methods such as neural radiance fields (NeRF) [37] and 3D Gaussian splatting (3DGS) [25] produce 3D representations that can be rendered into novel views that are almost indistinguishable from reality. This is the task of novel view synthesis (NVS), and it enables various applications such as immersive VR experiences inside captured real-world scenes. However, for these methods to achieve such a high level of photorealism, they need an extremely large number of captured images per scene. Obtaining these images is laborious and not always possible. Thus, one of the core challenges in 3D reconstruction and novel view synthesis is how to build an algorithm that can achieve equally good results with far fewer images, while also being able to take advantage of a large set of images when they are available.

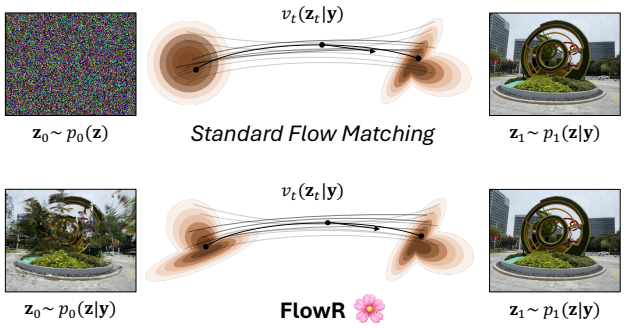

Figure 1. **Flowing from sparse to dense 3D reconstructions.** Contrary to previous diffusion and flow matching models that map a standard multivariate Gaussian distribution $p_0(\mathbf{z})$ to an, often conditional, target distribution $p_1(\mathbf{z} \mid \mathbf{y})$, we consider *source* distributions of the form $p_0(\mathbf{z} \mid \mathbf{y})$. We use novel view renderings of sparse reconstructions as source distribution samples, which we map to the target distribution $p_1(\mathbf{z} \mid \mathbf{y})$ that represents reconstructions obtained under optimal, dense conditions (*i.e.* ground truth).

In this paper, we tackle this problem with our FlowR method. FlowR consists of two parts: (a) a robust initial reconstruction pipeline based on 3DGS [25] but designed for both sparse- and dense-view settings; and (b) a data densification procedure, which uses flow matching [33] to generate high-quality extra views that can be used to improve reconstruction.

Several previous approaches [12, 19, 34, 45, 59, 61, 67] have used generative models to generate novel views to improve reconstruction. These works rely on diffusion [21] or flow matching [31] models that map samples between a data and a noise distribution. As such, these works use a noise-to-data generative process, usually conditioned on initial input images or renderings (Fig. 1, top). We hypothesize that a generative process that directly maps samples from a con-

ditional source distribution of rendered images to the target distribution of ground truth images is preferable to conditional noise-to-data generation. Additionally, we observe that flow matching is a paradigm for generative modeling where the model learns a velocity field that transports samples from *any* source to a target distribution. Therefore, in this work, we pose the problem as a flow matching problem where, instead of modeling the velocity field between noise and data, we model the velocity field between the distribution of incorrect novel view renderings and the respective real images of that viewpoint (Fig. 1, bottom).

In this way, if we already have enough dense input images that our initial reconstruction is good enough for a particular view, the flow matching model can simply learn not to change the input. This formulation ensures that the generative model does not hallucinate unnecessary new details that are inconsistent and conflict with existing scene content, and thus results in sharp scene details, avoiding blurred-out averages of inconsistent generations.

Our flow matching model generates N images simultaneously with a single multi-view diffusion transformer to ensure that all generated images are consistent with each other, while also conditioning the generation of these N images with M input images from the initial reconstruction to ensure that the novel generated views are consistent with the input views. To train our model, we create a dataset of 10.3k reconstructed scenes using our robust initial reconstruction approach, from which we obtain 3.6M pairs of novel view reconstructions with their corresponding ground truth images. In summary, we introduce FlowR, a pipeline that bridges the gap between sparse and dense 3D reconstructions. Our contributions are as follows.

- We develop a fast initial 3D reconstruction pipeline that produces 3DGS representations from arbitrary view distributions and use it to collect a large-scale dataset of 3.6M rendered and ground-truth image pairs.
- We propose a flow matching formulation that directly incorporates novel-view renderings from initial reconstructions as a surrogate for the initial "noise" distribution and use it to train a multi-view flow matching model that enhances novel-view renderings.
- We demonstrate that our trained model enhances the quality of 3D reconstructions in both sparse and dense view scenarios by simply incorporating novel-view renderings enhanced by our flow matching model into the reconstruction process.

Our approach outperforms prior methods in both sparse- and dense-view scenarios across three NVS benchmarks.

2. Preliminaries

We describe the core ideas that our approach is built upon, specifically 3D Gaussian splatting (Sec. 2.1) and flow matching (Sec. 2.2).

2.1. 3D Gaussian Splatting

3D Gaussian splatting [25] describes a method for NVS from a set of calibrated images.

Representation. Let $\mathcal{I} := \{\mathcal{I}_i := (\mathtt{I}_i, \mathbf{P}_i, \mathbf{K}_i)\}_{i=1}^N$ denote a set of N input images $\mathtt{I}_i$ with camera-to-world transformation $\mathbf{P}_i := [\mathbf{R}_i|\mathbf{t}_i]$ and intrinisc matrix $\mathbf{K}_i$. We are interested in representing the underlying 3D scene as a set of 3D Gaussian primitives $\mathcal{G} := \{\mathfrak{g}_k\}_{k=1}^K$. Each 3D Gaussian $\mathfrak{g}_k \in \mathcal{G}$ is parametrized by $\mathfrak{g}_k := \{\boldsymbol{\mu}_k, \mathbf{s}_k, \mathbf{q}_k, \alpha_k, \mathbf{c}_k\}$, *i.e.* position, scale, rotation, opacity and view-dependent color, respectively. The Gaussian kernel takes the following form

$$\mathfrak{g}_k(\mathbf{p}) := \exp\left(-\frac{1}{2}[\mathbf{p} - \boldsymbol{\mu}_k]^\top \Sigma_k^{-1}[\mathbf{p} - \boldsymbol{\mu}_k]\right) . \quad (1)$$

Here, the covariance matrix factorizes as $\Sigma_k := \mathbf{U}_k \operatorname{diag}(\mathbf{s}_k)^2 \mathbf{U}_k^\top$, where $\mathbf{U}_k$ is the rotation matrix corresponding to quaternion $\mathbf{q}_k$ and $\mathbf{s}_k \in \mathbb{R}_+^3$ entails positive scaling factors turned into a diagonal matrix with $\operatorname{diag}()$.

Rendering. To render the 3D scene from a camera c, we map the 3D Gaussians to the image plane. In particular, for each primitive $\mathfrak{g}_k$, let $\mathfrak{g}_k^c$ denote a 2D Gaussian kernel with its mean $\boldsymbol{\mu}_k^c$ defined as the primitive's position projected to the image plane, *i.e.* $\boldsymbol{\mu}_k^c := \Pi^c(\boldsymbol{\mu}_k)$, and its covariance defined as $\Sigma_k^c := \mathtt{J}_k^c \Sigma_k \mathtt{J}_k^{c\top}$, where $\mathtt{J}_k^c$ is the Jacobian of the 3D-to-2D projection function Π^c evaluated at $\boldsymbol{\mu}_k$. To render the color of pixels $\mathbf{p}'$ of camera c, we apply alpha compositing on the depth-sorted primitives $\mathfrak{g}_k$:

$$\mathbf{c}(\mathbf{p}') := \sum_{k=0}^{K} \mathbf{c}'_k w_k \prod_{j=0}^{k-1} (1 - w_j) , \quad (2)$$

where $w_k := \alpha_k \mathfrak{g}_k^c(\mathbf{p}')$ and $\mathbf{c}'_k$ is the evaluated spherical harmonics function of $\mathbf{c}_k$ at the view direction of c.

Optimization. Using the differentiable rasterizer in [25], we fit the set of 3D Gaussians $\mathcal{G}$ to the training images $\mathcal{I}$ by applying the following per-image loss function:

$$\mathcal{L}_{\text{GS}}(\mathtt{I}; \mathcal{G}) := (1 - \lambda_{\text{ssim}}) \|\hat{\mathtt{I}} - \mathtt{I}\|_1 + \lambda_{\text{ssim}} \operatorname{SSIM}(\hat{\mathtt{I}}, \mathtt{I}) , \quad (3)$$

where $\hat{\mathtt{I}}$ is a rendering as per Eq. (2) from the viewpoint of training image $\mathtt{I}$, $\lambda_{\text{ssim}} := 0.2$, and $\operatorname{SSIM}(\cdot)$ is the structural similarity loss [58]. The optimization is interleaved with an adaptive density control (ADC) mechanism [25] that prunes transparent primitives, and splits and clones primitives in over- and under-reconstructed areas, respectively.

2.2. Flow Matching

Flow matching is a paradigm for generative modeling that we briefly introduce in this section and refer to the original papers [31, 33] for further details.

Model. Let $\mathbf{z}_0$ denote a sample from a source distribution p_0 that we wish to map to a sample $\mathbf{z}_1$ of a target distribution p_1. We model this transformation as a continuous

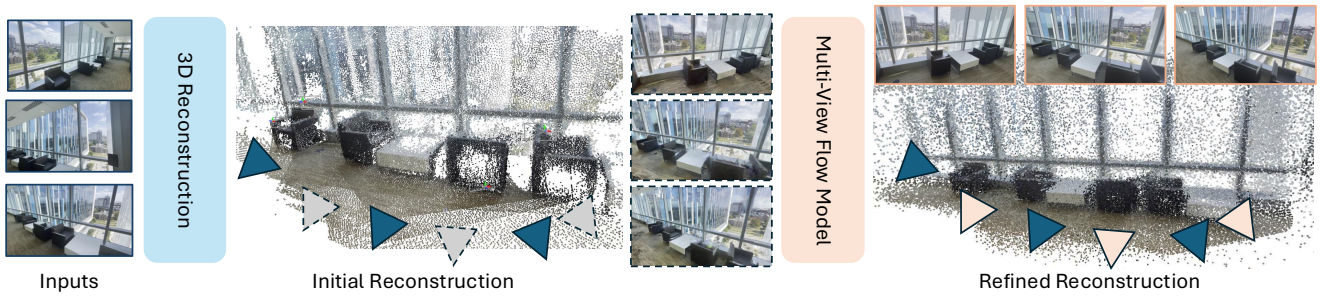

Figure 2. **Overview.** Given a set of source input images $\mathcal{I}_{\text{src}}$ (blue), we use our robust reconstruction method (Section 3.1) to create an initial reconstruction result that can be rendered at various perspectives (gray). We use these renderings as the source samples of our flow matching model (Section 3.2), which maps the rendered images to the target distribution, *i.e.* ground-truth images. We use the generated views (orange) to improve the quality of the reconstruction (Section 3.3).

probability flow with time $t \in [0, 1]$, governed by an ordinary differential equation (ODE):

$$d\mathbf{z}_t = \boldsymbol{v}(\mathbf{z}_t, t)dt \tag{4}$$

where $\boldsymbol{v}(\mathbf{z}_t, t)$ is a time-dependent velocity field that induces a probability path p_t interpolating p_0 and p_1. Assuming an optimal transport path, the state $\mathbf{z}_t$ should satisfy

$$\mathbf{z}_t \coloneqq t\mathbf{z}_1 + (1 - (1 - \sigma_{\text{min}})t)\mathbf{z}_0 \tag{5}$$

where $\sigma_{\text{min}} \coloneqq 10^{-5}$. It follows from differentiating the equation w.r.t. t that the true velocity field is given by

$$\boldsymbol{v}_t \coloneqq \frac{d\mathbf{z}_t}{dt} = \mathbf{z}_1 - (1 - \sigma_{\text{min}})\mathbf{z}_0 \,. \tag{6}$$

Training. We train a network with parameters θ to approximate $\boldsymbol{v}_t$ with the conditional flow matching loss [1, 31]:

$$\mathcal{L}_{\text{CFM}}(\theta) \coloneqq \mathbb{E}_{t,\mathbf{z}_0,\mathbf{z}_1} \|\boldsymbol{v}_\theta(\mathbf{z}_t, t) - \boldsymbol{v}_t\|_2^2 \tag{7}$$

where $\mathbf{z}_t$ and $\boldsymbol{v}_t$ are given as per Eqs. (5) and (6) with $\mathbf{z}_0$ and $\mathbf{z}_1$ sampled from p_0 and p_1, respectively, and t sampled from a logit normal distribution [15]. Note that, although this formulation can transport samples between two arbitrary distributions, in practice p_0 is often a standard Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

Inference. To transport a sample $\mathbf{z}_0$ from p_0 to p_1, we numerically solve the ODE in Eq. (4) using the estimated velocity $\boldsymbol{v}_\theta(\mathbf{z}_t, t)$ and Euler's method, but other numerical solvers are viable.

3. Method

Our goal is to improve the quality of an initial reconstruction of a scene created from a possibly sparse set of posed images. We exploit flow matching to improve the quality of novel view renderings from the initial reconstruction, which serve as auxiliary training data to fit an improved 3D representation. Although our method could work with different scene representations, it is convenient to adopt Gaussian primitives because they can be trained fast. In fact, since our approach is data-driven, we need to construct a dataset of reconstructions under different sparsity levels, and Gaussian splatting ensures better scalability for this purpose. In Sec. 3.1, we describe our robust 3D reconstruction method that serves as a basis for generating our dataset of 3D reconstructions and provides the initial scene reconstructions. Sec. 3.2 describes how we employ flow matching to improve novel-view renderings from the initial reconstructions, which, in turn, are used to improve our reconstruction results in Sec. 3.3. See Fig. 2 for an illustration.

3.1. Robust 3D Reconstruction

We aim to devise a 3D reconstruction pipeline that produces semi-dense, metric-scale point clouds from arbitrary input view distributions, *i.e.* ranging from a few sparse to thousands of dense views. To this end, we combine recent advances in learning-based structure-from-motion (SfM) and monocular depth estimation with classical SfM tools.

First, we define a sparse co-visibility graph $G_{\text{vis}} \coloneqq (\mathcal{I}, \mathcal{E})$ on the set of posed input images $\mathcal{I}$ similar to [14]. We use keyframing to limit the number of view pairs for matching and thus enable a scalable 3D reconstruction process. In particular, we sample $\sqrt{N}$ keyframes using farthest point sampling. The edges $\mathcal{E} \subset \mathcal{I} \times \mathcal{I}$ are constructed by densely connecting all keyframes and connecting other nodes to their closest keyframe and their k nearest neighbors. For both keyframes selection and edge construction we consider the simple image co-visibility metric, $\mathcal{S}(\mathcal{I}_i, \mathcal{I}_j) \coloneqq 1 - \frac{\mathcal{D}(\mathcal{I}_i, \mathcal{I}_j)}{\max_{ij} \mathcal{D}(\mathcal{I}_i, \mathcal{I}_j)}$, where

$$\mathcal{D}(\mathcal{I}_i, \mathcal{I}_j) \coloneqq \|\mathbf{R}_i - \mathbf{R}_j\|_F + \eta\|\mathbf{t}_i - \mathbf{t}_j\|_2 \,. \tag{8}$$

Here, $\|\cdot\|_F$ is the Frobenius norm and $\eta \coloneqq \frac{1}{6}$.

Second, we run MASt3R [28] on all edges $\mathcal{E}$ to obtain 3D points and descriptors for each view pair. We extract matches following [28] with a coarse-to-fine nearest-neighbor search on the descriptors. Since extracting dense matches is useful for dense geometry reconstruction from a small number of observations, but not beneficial for scenes

with many images, we subsample matches using confidence and pixel tolerance thresholds based on the number of available input views.

Third, given the feature matches for each pair of views in $\mathcal{E}$, we can take advantage of G_{vis} to create feature tracks by exploiting the transitivity of the matches. Finally, we use COLMAP [50] to triangulate 3D points from the given feature tracks. To achieve a 3D reconstruction on a metric scale, we align the reconstructed depth $\mathtt{D}$ with the monocular depth estimates $\hat{\mathtt{D}}$ of [23] by computing a scalar scaling factor β solving $\min_\beta \sum_{i=1}^{N} \mathtt{W}_i \|\beta \mathtt{D}_i - \hat{\mathtt{D}}_i\|_1$, where $\mathtt{W}_i$ is the confidence of the monocular depth estimate $\hat{\mathtt{D}}_i$.

Fitting 3D Gaussians. Using this reconstruction as an initialization, we fit 3D Gaussians $\mathcal{G}$ from the input images with a short schedule. We train for 5k steps with a warm-up phase of 200 steps. After warm-up, we use ADC with AbsGrad [64] metric until step 2.5k. Compared to previous works [17] that use 3D points obtained from global point-cloud alignment [28] as initialization, our pipeline has two advantages. First, it refines the geometry by using track constraints. Second, it removes duplicated and inconsistent 3D points from the initialization, improving performance and scalability (see Sec. 4.2).

Dataset generation. We leverage our robust reconstruction pipeline and two recent large-scale NVS benchmarks [30, 65] to generate a dataset of rendered and ground-truth image pairs. For each scene, we first select an input view subset with a varying degree of view sparsity to achieve a large variety of view distributions (see Appendix A.1). Next, we utilize all views not in this subset as privileged information for our flow matching model. We reconstruct the scenes from the input view subset with our aforementioned pipeline and create rendering, ground-truth pairs from all other available views, producing a training dataset of 3.6M image pairs from 10.3k sequences. Processing all scenes took about 1.1k GPU hours or 6.5 minutes on average per scene on budget GPUs like RTX 2080 Ti.

3.2. Flowing from Sparse to Dense Reconstructions

We use flow matching to model a joint distribution of novel-view (target) images $\mathbf{z} \coloneqq \{\mathtt{I}_i^{\text{tgt}}\}_i$ conditioned on $\mathbf{y} \coloneqq (\mathcal{I}_{\text{src}}, \mathcal{P}_{\text{tgt}})$, namely a set of (source) posed, reference images $\mathcal{I}_{\text{src}}$ and a set of target novel-view poses $\mathcal{P}_{\text{tgt}}$. Ideally, it is supposed to match the true data distribution $p_1(\mathbf{z}|\mathbf{y})$. A classical way to implicitly fit $p_1(\mathbf{z}|\mathbf{y})$ in the flow matching framework consists of learning a vector field $\boldsymbol{v}_\theta(\mathbf{z}, \mathbf{y}, t)$ that depends also on the conditioning variables $\mathbf{y}$, which generates a (conditional) probability density path connecting a standard multivariate Gaussian as the source distribution $p_0(\mathbf{z})$ to the target distribution $p_1(\mathbf{z}|\mathbf{y})$.

Inspired by [32], in our work we adopt a different strategy and condition also the source distribution p_0 on $\mathbf{y}$, *i.e.* we consider source distributions of the form $p_0(\mathbf{z}|\mathbf{y})$. Now, fitting a 3D Gaussian representation $\mathcal{G}_{\text{src}}$ to the set of posed reference images $\mathcal{I}_{\text{src}}$ can be interpreted as drawing a 3D representation from a stochastic process [26]. Hence, rendering novel views at the poses $\mathcal{P}_{\text{tgt}}$ using $\mathcal{G}_{\text{src}}$ can be interpreted as drawing a sample from a conditional distribution $\mathbf{z}|\mathbf{y}$. Consequently, this process can characterize our conditional source distribution $p_0(\mathbf{z}|\mathbf{y})$. By doing so, our model learns a flow connecting renderings $\mathbf{z}_0 \sim p_0(\mathbf{z}|\mathbf{y})$ obtained from potentially sparse reconstructions (*e.g.* if $\mathcal{I}_{\text{src}}$ has few images) to renderings $\mathbf{z}_1 \sim p_1(\mathbf{z}|\mathbf{y})$ that we would have under denser reconstructions (or even ground-truth) as enforced by our training procedure.

Below, we detail the implementation of our multi-view flow model and in particular the velocity field $\boldsymbol{v}_\theta(\mathbf{z}, \mathbf{y}, t)$.

Image encoding. In our model $\mathbf{z}$ does not entail images in pixel space, but we use a pre-trained VAE encoder [16] to encode each image to a latent tensor with dimensions $h \times w \times 16$. Similarly, conditioning images in $\mathbf{y}_{\text{src}}$ undergo the same encoding in latent space before entering our model. Finally, a pre-trained VAE decoder is used to map this latent representation back in pixel space.

Camera conditioning. The network implementing the velocity field described below is conditioned on both source and target camera poses. We encode camera information as Plücker coordinate ray maps [43, 69] expressed in the coordinate frame of the reference view that is closest to the center of mass of all involved camera positions.

Velocity field $\boldsymbol{v}_\theta(\mathbf{z}, \mathbf{y}, t)$. We input the target and source views as a single set of tokens to the flow matching model implemented as a diffusion transformer (DiT) [15, 42]. All image latents provided as input, namely $\mathbf{z}$ and the encoded source images from $\mathbf{y}_{\text{src}}$, are first split into 2×2 patches. We project each patch to the token dimension and add a 2D positional encoding. In addition, we encode the index i of the image from which each token originates. To this end, we adopt a 1D sinusoidal encoding $\gamma(i)$ [55]. During training, the view index is chosen randomly from $[0, 1000]$, whereas at inference the view indices are $[0, N]$ where N the number of images for a single forward pass. Originally from LLM literature [10], this technique enables inference with very long context lengths and has recently been applied to 3D reconstruction [62]. The sequence of tokens is then processed with a series of transformer blocks to predict the output velocity tensor. To share information across multiple reference and target views while leveraging large-scale text-to-image pre-training, we adapt the DiT block as follows. We first collapse the view dimension into the batch dimension, keeping the self-attention layer equivalent to the pre-trained image model. Then, we collapse the view dimension into the spatial dimension. We add the image index encoding $\gamma(i)$ to the state, concatenate it with the ray map embeddings, and linearly project the concatenated tokens

back to the original dimension. We then insert a second self-attention layer to enable full *multi-view attention*. Finally, before adding the output to the prior state of the block, we insert a zero linear layer [70] to keep the pre-trained initialization intact. See Appendix A.2 for an illustration.

3.3. Improving the 3D Reconstruction

We sketch how we employ the flow matching model presented in the previous section to improve the *initial* 3D reconstruction $\mathcal{G}_{\text{src}}$ that we obtain by running the procedure described in Sec. 3.1 on the initial source views $\mathcal{I}_{\text{src}}$.

Generation of target views $\mathcal{P}_{\text{tgt}}$. We first select a set of camera poses suitable for augmenting the initial input views. We distinguish between scenes where sparse input views lie on a continuous trajectory, *i.e.* input views are ordered, and unordered photo collections. If input views are ordered, we model a smooth trajectory using B-spline basis functions from a sparse set of control points (source camera poses) and sample the target poses from it. If not, we use the distance metric in Eq. (8) to select reference poses with farthest point sampling. For each reference pose, we generate candidate points on a sphere with a random radius using Fibonacci sphere sampling [51]. We select the point with the maximum distance to the source cameras and apply a small perturbation to the reference view orientation to obtain the target camera pose. Finally, we filter target camera poses with too many points close to the camera or too few points inside the view frustum using the initial 3D reconstruction results, yielding the final set of target poses $\mathcal{P}_{\text{tgt}}$.

Renderings of target views. Renderings obtained from our initial reconstruction $\mathcal{G}_{\text{src}}$ at the target poses $\mathcal{P}_{\text{tgt}}$ can be considered a sample $\mathbf{z}_0$ of the conditional prior distribution $p_0(\mathbf{z}|\mathbf{y})$ as argued in Sec. 3.2. Thus, the inference process of our flow matching model yields refined renderings $\mathbf{z}_1$ starting from $\mathbf{z}_0$. Once decoded, these renderings together with the respective poses $\mathcal{P}_{\text{tgt}}$ form additional training data denoted by $\mathcal{I}_{\text{tgt}}$ that complements the original set $\mathcal{I}_{\text{src}}$.

Refined reconstruction $\mathcal{G}_{\text{ref}}$. We fit an improved 3D scene representation $\mathcal{G}_{\text{ref}}$ on the union $\mathcal{I}_{\text{src}} \cup \mathcal{I}_{\text{tgt}}$ of source and target novel-view posed images. However, the two sets of images undergo different loss functions because views in $\mathcal{I}_{\text{tgt}}$ could exhibit small imperfections, *e.g.*, due to VAE compression artifacts. Namely, we apply Eq. (3) to images in $\mathcal{I}_{\text{src}}$ and the following loss function to images in $\mathcal{I}_{\text{tgt}}$:

$$\begin{aligned}\mathcal{L}_{\text{tgt}}(\mathtt{I};\mathcal{G}) :=& (1-\lambda'_{\text{ssim}})\|\hat{\mathtt{I}}-\mathtt{I}\|_2 \\ &+ \lambda'_{\text{ssim}}\,\text{SSIM}(\hat{\mathtt{I}},\mathtt{I}) + \lambda_{\text{lpips}}\,\text{LPIPS}(\hat{\mathtt{I}},\mathtt{I})\,,\end{aligned}$$

where $\lambda_{\text{lpips}} = \lambda'_{\text{ssim}} := 0.02$ and $\text{LPIPS}(\cdot)$ denotes perceptual similarity [71]. Since color values are bound to $[0,1]$, the L2 loss is less strict in terms of color variations. For the reconstruction, we employ the method provided in Sec. 3.1, but restrict the selection of keyframes to $\mathcal{I}_{\text{src}}$.

4. Experiments

Implementation details. We initialize our flow matching model from an image generation model trained on a large collection of image-text pairs similar to [15]. The base model has 2.7B parameters, and we drop the text conditioning layers, leaving us with 1.6B pre-initialized parameters. We train all multi-view layers from scratch; the total parameters are 1.75B. We use FlashAttention2 [13] for more efficient attention computation. We train our model on 64 H100 GPUs for 125k steps (48 hours) at batch size 64 with 12 views per batch element and 512px width, keeping the original aspect ratio of each image. We use a cosine learning rate (LR) schedule with a warmup phase of 1k steps and a maximum LR of 10^{-5} scaled by $\sqrt{n}$ where n is the number of batch elements. We fine-tune our model at 960px width for 55k steps with 6 views per batch element for another 31 hours. We lower the maximum LR to $5 \cdot 10^{-6}$, keeping the same schedule. For 3D Gaussian color $\mathbf{c}_k$, we use spherical harmonics degree of three.

Experimental setup. We compare to prior art across multiple benchmarks, *i.e.* DL3DV140 [30], ScanNet++ [65], and Nerfbusters [60]. For DL3DV140, we first split each sequence into training and evaluation views and create two training splits by sampling $k \in \{12, 24\}$ equally spaced views from the training trajectory. We evaluate each sequence at the original 540×960 resolution. For ScanNet++, we use the validation set, follow the official training and testing splits, and evaluate at 640×960 resolution. For Nerfbusters, we follow the original evaluation protocol in [60]. We measure view quality in terms of both low-level similarity via PSNR and SSIM [58], and high-level, perceptual similarity using LPIPS [71].

Baselines. We compare to state-of-the-art approaches that are open-source, *i.e.* ViewCrafter [67] and InstantSplat [17] in sparse-view settings, and GANeRF [47] in the dense-view setting. We also compare to Nerfbusters [60] on their proposed benchmark.

4.1. Comparison to State-of-the-Art

In Tab. 1, we compare to state-of-the-art sparse-view methods on the DL3DV [30] benchmark. In particular, we compare to InstantSplat [17] and ViewCrafter [67]. For a fair comparison, we use the same prior information (*i.e.* posed images) and downstream pipelines (*i.e.* our refined reconstruction) if applicable. While InstantSplat performs competitively to our *initial* reconstructions in the 12-view setting, its performance stagnates in the 24-view setting, highlighting its limitation to sparse view scenarios. The generated views from ViewCrafter do not improve reconstructions meaningfully in our experiments, as its results with our downstream pipeline lag behind our initial reconstructions. Our full method on the other hand improves 3D re-

Method	12-view			24-view		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Splatfacto [52]	16.71	0.528	0.478	22.17	0.738	0.309
InstantSplat† [17]	20.47	0.698	0.297	19.57	0.710	0.326
ViewCrafter* [67]	19.19	0.638	0.375	21.95	0.734	0.298
FlowR (Initial)	20.86	0.715	0.333	24.30	0.818	0.252
FlowR	22.43	0.766	0.280	25.13	0.836	0.212
FlowR++	22.60	0.793	0.261	25.33	0.863	0.193

Table 1. **Sparse-view 3D reconstruction on DL3DV140 [30].** † official code with GT poses, * official code with GT poses and our refined reconstruction method. Additionally, we report results of our generative model refining the test views (FlowR++).

Method	PSNR ↑	SSIM ↑	LPIPS ↓
Splatfacto [52]	22.41	0.843	0.352
GANeRF [47]	23.95	0.856	0.306
FlowR (Initial)	23.84	0.860	0.331
FlowR	24.11	0.870	0.303
GANeRF w/ GAN	24.01	0.860	0.291
FlowR++	**24.90**	**0.922**	**0.250**

Table 2. **Dense-view 3D reconstruction on ScanNet++ validation [65].** We use the official training and testing splits. In addition, we report results of our flow-matching model refining the test views (FlowR++), similar to [47] (w/ GAN).

construction results by a significant margin across all metrics, both in the 12-view and 24-view settings. The improvement is particularly pronounced in the 12-view setup, and furthermore, the relative gain compared to our initial reconstruction is the highest in perceptual quality (LPIPS). Additionally, we report the results of our flow model applied to the test view poses (FlowR++) to highlight its effectiveness for refining renderings of unseen views.

In Tab. 2, we compare to state-of-the-art dense-view methods on the ScanNet++ benchmark. This setting is very challenging, as the initial reconstruction benefits from a large set of source images, and the test view poses are specifically chosen to represent out-of-distribution viewpoints [65]. Despite these challenges, our method improves significantly over our initial reconstruction results, outperforming GANeRF [47]. When applying post-hoc refinement of rendered views (bottom two rows), we also outperform GANeRF by a large margin, improving our initial results dramatically with a most significant improvement in LPIPS. In Fig. 3 and Fig. 4, we show qualitative comparisons on the aforementioned benchmarks. Our method exhibits strikingly fewer artifacts from floaters and poorly reconstructed geometry than the baseline methods.

Finally, in Tab. 3, we report results on Nerfbusters [60]. The benchmark consists of completely disjoint training and testing trajectories that cover distinct view angles and possibly scene content. Thus, the evaluation excludes areas unseen during training for NVS quality assessment and also reports coverage, *i.e.* the percentage of 3D points

Method	PSNR ↑	SSIM ↑	LPIPS ↓	Coverage ↑
Splatfacto [52]	16.17	0.529	0.375	0.924
Nerfacto [52]	17.00	0.527	0.380	0.896
Nerfbusters [60]	17.99	0.606	0.250	0.630
FlowR (Initial)	17.02	0.567	0.365	0.932
FlowR	18.31	0.607	0.337	0.932
FlowR*	18.94	0.780	0.181	0.680

Table 3. **Dense-view 3D reconstruction on Nerfbusters [60].** We report view quality and coverage, *i.e.* how many pixels of the 3D points seen during training the method reconstructed, on the test trajectories. *using opacity thresholding.

seen in the training images reconstructed by a method, on the test trajectories. Our initial reconstruction method outperforms other reconstruction baselines like Nerfacto and Splatfacto [52] but lags behind Nerfbusters. Our full method outperforms Nerfbusters in PSNR and SSIM even without any postprocessing w.r.t. coverage, *i.e.* maintaining the same degree of coverage as our initial reconstruction. When applying simple opacity thresholding to our renderings (see Appendix A.3), we outperform Nerfbusters by a large margin on both NVS metrics and coverage.

4.2. Ablation Studies

Initial reconstruction $\mathcal{G}_{\text{src}}$. In Tab. 4 we ablate the components of our reconstruction pipeline. Compared to the Splatfacto [52] baseline (1), using the dense 3D pointcloud from MASt3R [28] yields sizable gains in performance across all metrics with LPIPS being improved particularly. However, there is a steep increase w.r.t. the number of Gaussians $|\mathcal{G}|$. In addition, several hundreds of view pairs are needed to produce the initialization. The co-visiblity graph G_{vis} (3) decreases the number of required view pairs dramatically, even on the relatively small scale 12 and 24 view splits of DL3DV. Re-triangulating the matches (4) extracted from the MASt3R predictions, instead, dramatically reduces the number of Gaussians while increasing view quality in terms of PSNR and SSIM. However, (4) exhibits a significantly higher LPIPS than variants (2) and (3). This is because while the triangulated 3D points are more precise, they exhibit holes in areas where geometry is uncertain. If we do not use ADC to grow Gaussians in these under-reconstructed areas, the LPIPS score will significantly increase. Therefore, we enable ADC and achieve the best result in LPIPS, while performing similarly to (4) in terms of PSNR and SSIM. Finally, we show that our short schedule with 5k training steps is indeed sufficient to obtain a good reconstruction from an initial set of sparse views, since longer training (6) does not improve performance.

Flow matching model. To show the importance of our formulation for view generation quality, we first implement a baseline model that retains a standard Gaussian distribution as p_0. In particular, we concatenate the input latents with the

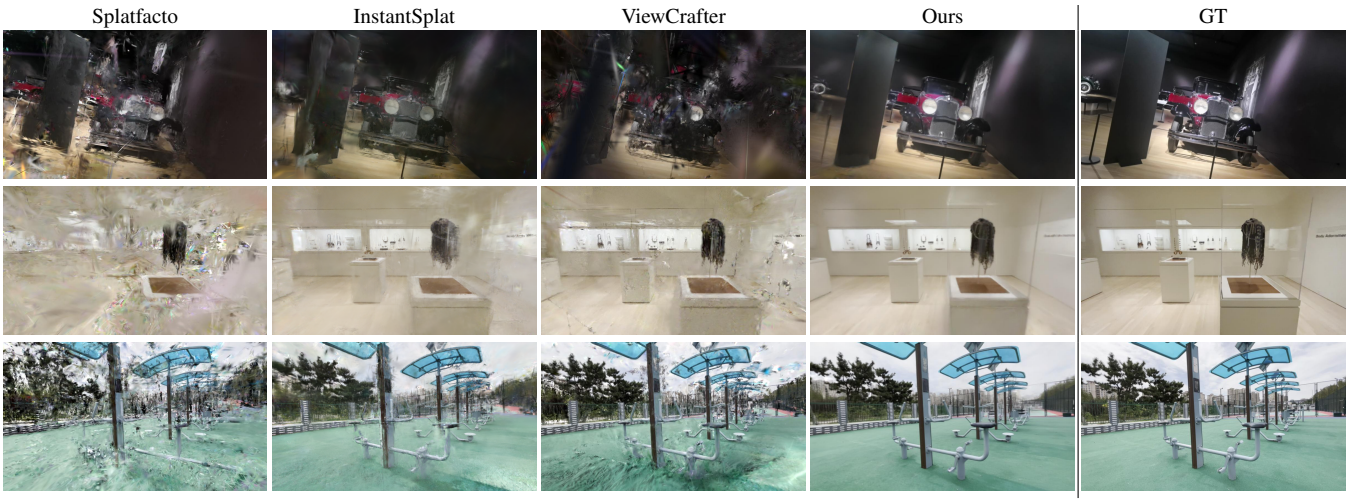

Figure 3. **Qualitative reconstruction results on DL3DV140 [30].** We show several examples of test view renderings from reconstructions obtained from our baselines and our method *on the more challenging 12-view split*.

	MASt3R	Graph G_{vis}	Re-triang.	ADC	Train steps	PSNR ↑	SSIM ↑	LPIPS ↓	$\lvert\mathcal{G}\rvert$	Pairs
1	-	-	-	✓	30K	19.44	0.633	0.394	317K	-
2	✓	-	-	-	5K	21.72	0.739	0.294	2.01M	342
3	✓	✓	-	-	5K	21.76	0.74	0.293	2.01M	129
4	✓	✓	✓	-	5K	22.23	0.768	0.322	186K	129
5	✓	✓	✓	✓	5K	22.58	0.766	0.292	244K	129
6	✓	✓	✓	✓	30K	22.21	0.752	0.295	286K	129

Table 4. **Initial reconstruction $\mathcal{G}_{\text{src}}$.** We compare 5 different variants (2-6) of our reconstruction pipeline to a naive Splatfacto [52] baseline, reporting the average scores across our DL3DV [30] view splits. We observe that dense initialization (2) improves results dramatically, but incurs a large computational burden. We show that our introduced co-visiblity graph G_{vis} (3), the triangulation of 3D points (4), and the short schedule with ADC (4 and 5) consistently improve results while dramatically decreasing computational complexity.

	p_0	DL3DV140			ScanNet++		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
-	-	*23.35*	*0.782*	*0.220*	*24.18*	*0.861*	*0.278*
1	$\mathcal{N}(\mathbf{0},\mathbf{I})$	23.50	0.769	0.200	24.09	0.873	0.208
2	$p_0(\mathbf{z}\lvert\mathbf{y})$	**24.15**	**0.808**	**0.180**	**25.53**	**0.921**	**0.188**

Table 5. **Different source distributions.** We report results on ScanNet++ [65] validation and the average of our DL3DV140 [30] view splits at 512px resolution. In the first row, we show the scores of our initial reconstruction $\mathcal{G}_{\text{src}}$ which serves as input to our model as reference. Our conditional formulation improves significantly over a model with a standard Gaussian as source distribution.

latents of the conditioning views, *i.e.* target renderings and reference images, and the raymaps, following [4]. We compare the performance of this baseline to our model in Tab. 5. We show the NVS metrics of the initial renderings as reference in gray. While the baseline (1) improves significantly over the renderings in terms of LPIPS, the improvement is weaker in terms of PSNR and especially SSIM, where on DL3DV the model deteriorates the performance compared to the renderings. Instead, our formulation (2) improves the renderings consistently across all metrics and outperforms the baseline by a significant margin.

	$\mathcal{L}_{\text{tgt}}$	Re-init.	PSNR ↑	SSIM ↑	LPIPS ↓
1	-	-	23.53	0.795	0.279
2	✓	-	23.72	0.799	0.248
3	✓	✓	**23.78**	**0.801**	**0.246**

Table 6. **Refined reconstruction $\mathcal{G}_{\text{ref}}$.** We report the average across our DL3DV [30] view splits. Both using $\mathcal{L}_{\text{tgt}}$ and incorporating generated views in the pointcloud initialization (Re-init.) improve view synthesis quality.

Refined reconstruction $\mathcal{G}_{\text{ref}}$. In Tab. 6, we test if differentiating the loss function between source and target images and using generated views for pointcloud initialization indeed improves reconstruction results. To this end, we first train a model (1) on $\mathcal{I}_{\text{tgt}}$ with the vanilla objective $\mathcal{L}_{\text{GS}}$ and the pointcloud obtained from the source images. We then train a model (2) with our alternative loss function $\mathcal{L}_{\text{tgt}}$ using the same initialization. Finally, we re-initialize the pointcloud (3) and train the model with it and $\mathcal{L}_{\text{tgt}}$. The alternative loss applied to the generated target images significantly improves all NVS metrics with a particularly-pronounced improvement in LPIPS. Using the generated views for initialization leads to further improvement, however more pronounced in the 12 view setting (see Appendix A.3).

Figure 4. **Qualitative reconstruction results on ScanNet++ [65].** We show *out-of-distribution* test view renderings of the baselines and our method in the dense-view setting. This is a particularly challenging setting as the test views are far from the initial camera poses.

5. Related Work

We review works that approach 3D reconstruction and novel view rendering based on non-exhaustive scene captures.

Geometric priors. To overcome the requirement of dense input views in [25, 37], a line of works relied on geometry regularizations such as depth smoothness [27, 39], ray entropy [2, 27], and re-projection losses [36, 54] for a more well-behaved scene-level optimization. Other works use monocular depth and normals from off-the-shelf networks as geometry priors [36, 46, 56, 68, 72]. Recently, several works initialize 3DGS from either monocular [18] or multi-view networks [17]. While our reconstruction pipeline aligns with these recent efforts, we propose a more scalable approach handling arbitrary view distributions and refine our reconstructions with generative priors.

Feed-forward methods. Another line of work aims to directly predict novel views [7, 22, 57, 66] from input images, skipping scene-level optimization entirely. Along a similar axis, more recent works directly predict explicit 3D representations from input images [6, 11, 12, 53]. Instead of abandoning per-scene optimization, we propose a pipeline that integrates data priors both in the initialization and the training process of 3D Gaussian splatting.

Generative priors. Generative priors enable improvements in both scene geometry and appearance. While earlier work used normalizing flows [39] and GANs [47], many recent works focus on diffusion models [12, 19, 34, 45, 59, 61, 67]. In particular, many works incorporate diffusion models in the training of radiance fields [45, 59, 61], *e.g.* via distillation [45, 59]. However, since this requires continuously evaluating the model during training, many recent approaches focus on directly generating additional training views [19, 34, 41, 67]. Among these, many works focus on multi-frame architectures since such models have shown great promise in video generation [4, 5, 9, 35, 44, 63], generating videos with high geometric consistency [29]. However, due to architectural limitations, these works only generate a handful of views at a time. Additionally, they keep the generative process as is and inject prior information into the model only as conditioning. We instead propose a flow matching process that directly refines initial reconstruction results and design the model architecture such that it still benefits from large-scale text-to-image pre-training.

6. Conclusion

We introduced FlowR, a novel pipeline that bridges the gap between sparse and dense 3D reconstructions. Our method enhances NVS by learning to map incorrect renderings to their corresponding ground-truth images. By training on a large-scale dataset of 3.6M image pairs, FlowR significantly improves 3DGS performance in both sparse- and dense-view scenarios, outperforming previous generative approaches that rely solely on 2D-conditioned synthesis.

Acknowledgements. Yung-Hsu Yang was supported by the Swiss AI Initiative and a grant from the Swiss National Supercomputing Centre (CSCS) under project ID a03 on Alps. The authors thank Andrew Brown, Corinne Stucker, and Duncan Zauss for helpful discussions and technical support.

References

[1] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. In *ICLR*, 2023. 3

[2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 8, 13, 14

[3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-nerf: Anti-aliased grid-based neural radiance fields. In *ICCV*, 2023. 13

[4] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 7, 8

[5] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your latents: High-resolution video synthesis with latent diffusion models. In *CVPR*, 2023. 8

[6] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *CVPR*, 2024. 8

[7] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021. 8

[8] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *NeurIPS*, 2025. 12

[9] Haoxin Chen, Yong Zhang, Xiaodong Cun, Menghan Xia, Xintao Wang, Chao Weng, and Ying Shan. Videocrafter2: Overcoming data limitations for high-quality video diffusion models. In *CVPR*, 2024. 8

[10] Shouyuan Chen, Sherman Wong, Liangjian Chen, and Yuandong Tian. Extending context window of large language models via positional interpolation. *arXiv preprint arXiv:2306.15595*, 2023. 4

[11] 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 *ECCV*, 2024. 8

[12] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. In *NeurIPS*, 2024. 1, 8

[13] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. 5

[14] Bardienus Duisterhof, Lojze Zust, Philippe Weinzaepfel, Vincent Leroy, Yohann Cabon, and Jerome Revaud. Mast3r-sfm: a fully-integrated solution for unconstrained structure-from-motion. *arXiv preprint arXiv:2409.19152*, 2024. 3

[15] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024. 3, 4, 5, 12, 13

[16] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 4

[17] Zhiwen Fan, Wenyan Cong, Kairun Wen, Kevin Wang, Jian Zhang, Xinghao Ding, Danfei Xu, Boris Ivanovic, Marco Pavone, Georgios Pavlakos, et al. Instantsplat: Unbounded sparse-view pose-free gaussian splatting in 40 seconds. *arXiv preprint arXiv:2403.20309*, 2(3):4, 2024. 4, 5, 6, 8

[18] Yang Fu, Sifei Liu, Amey Kulkarni, Jan Kautz, Alexei A Efros, and Xiaolong Wang. Colmap-free 3d gaussian splatting. In *CVPR*, 2024. 8

[19] Ruiqi Gao, Aleksander Holynski, Philipp Henzler, Arthur Brussee, Ricardo Martin-Brualla, Pratul Srinivasan, Jonathan T Barron, and Ben Poole. Cat3d: Create anything in 3d with multi-view diffusion models. In *NeurIPS*, 2024. 1, 8, 13, 14

[20] Lily Goli, Cody Reading, Silvia Sellán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields. In *CVPR*, 2024. 13

[21] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1

[22] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3d. In *ICLR*, 2024. 8

[23] Mu Hu, Wei Yin, Chi Zhang, Zhipeng Cai, Xiaoxiao Long, Hao Chen, Kaixuan Wang, Gang Yu, Chunhua Shen, and Shaojie Shen. Metric3d v2: A versatile monocular geometric foundation model for zero-shot metric depth and surface normal estimation. *arXiv preprint arXiv:2404.15506*, 2024. 4

[24] Wen Jiang, Boshu Lei, and Kostas Daniilidis. Fisherrf: Active view selection and mapping with radiance fields using fisher information. In *ECCV*, 2024. 13

[25] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 1, 2, 8

[26] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. In *NeurIPS*, 2024. 4

[27] Mijeong Kim, Seonguk Seo, and Bohyung Han. Infonerf: Ray entropy minimization for few-shot neural volume rendering. In *CVPR*, 2022. 8

[28] Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *ECCV*, 2024. 3, 4, 6, 12

[29] Xuanyi Li, Daquan Zhou, Chenxu Zhang, Shaodong Wei, Qibin Hou, and Ming-Ming Cheng. Sora generates videos with stunning geometrical consistency. *arXiv preprint*

arXiv:2402.17403, 2024. 8

[30] 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. In *CVPR*, 2024. 4, 5, 6, 7, 12, 13

[31] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023. 1, 2, 3

[32] Qihao Liu, Xi Yin, Alan Yuille, Andrew Brown, and Mannat Singh. Flowing from words to pixels: A framework for cross-modality evolution. *arXiv preprint arXiv:2412.15213*, 2024. 4

[33] Xingchao Liu, Chengyue Gong, and qiang liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *ICLR*, 2023. 1, 2

[34] Xi Liu, Chaoyi Zhou, and Siyu Huang. 3dgs-enhancer: Enhancing unbounded 3d gaussian splatting with view-consistent 2d diffusion priors. In *NeurIPS*, 2024. 1, 8

[35] Willi Menapace, Aliaksandr Siarohin, Ivan Skorokhodov, Ekaterina Deyneka, Tsai-Shien Chen, Anil Kag, Yuwei Fang, Aleksei Stoliar, Elisa Ricci, Jian Ren, et al. Snap video: Scaled spatiotemporal transformers for text-to-video synthesis. In *CVPR*, 2024. 8

[36] Andreas Meuleman, Yu-Lun Liu, Chen Gao, Jia-Bin Huang, Changil Kim, Min H Kim, and Johannes Kopf. Progressively optimized local radiance fields for robust view synthesis. In *CVPR*, 2023. 8

[37] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 8

[38] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. In *ECCV*, 2024. 12

[39] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 8

[40] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. Activenerf: Learning where to see with uncertainty estimation. In *ECCV*, 2022. 13

[41] Soumava Paul, Christopher Wewer, Bernt Schiele, and Jan Eric Lenssen. Sp2360: Sparse-view 360 scene reconstruction using cascaded 2d diffusion priors. In *ECCV 2024 Workshop on Wild 3D*. OpenReview. net, 2024. 8

[42] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 4, 12

[43] Julius Plücker. *Analytisch-geometrische Entwicklungen*, volume 2. GD Baedeker, 1828. 4

[44] Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, and et al. Movie gen: A cast of media foundation models. *arXiv preprint arxiv:2410.13720*, 2025. 8

[45] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023. 1, 8

[46] Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022. 8

[47] Barbara Roessle, Norman Müller, Lorenzo Porzi, Samuel Rota Bulò, Peter Kontschieder, and Matthias Nießner. Ganerf: Leveraging discriminators to optimize neural radiance fields. *ACM Transactions on Graphics (TOG)*, 42(6):1–14, 2023. 5, 6, 8

[48] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 12

[49] Kyle Sargent, Zizhang Li, Tanmay Shah, Charles Herrmann, Hong-Xing Yu, Yunzhi Zhang, Eric Ryan Chan, Dmitry Lagun, Li Fei-Fei, Deqing Sun, et al. Zeronvs: Zero-shot 360-degree view synthesis from a single image. In *CVPR*, 2024. 13

[50] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4

[51] Eric J Stollnitz, Tony D DeRose, and David H Salesin. *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann, 1996. 5

[52] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–12, 2023. 6, 7

[53] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *ECCV*, 2024. 8

[54] Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *CVPR*, 2023. 8

[55] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[56] Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, 2023. 8

[57] 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 *CVPR*, 2021. 8

[58] 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. 2, 5

[59] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *NeurIPS*, 2023. 1, 8

[60] Frederik Warburg, Ethan Weber, Matthew Tancik, Aleksander Holynski, and Angjoo Kanazawa. Nerfbusters: Removing ghostly artifacts from casually captured nerfs. In *ICCV*, 2023. 5, 6

[61] Rundi Wu, Ben Mildenhall, Philipp Henzler, Keunhong

Park, Ruiqi Gao, Daniel Watson, Pratul P Srinivasan, Dor Verbin, Jonathan T Barron, Ben Poole, et al. Reconfusion: 3d reconstruction with diffusion priors. In *CVPR*, 2024. 1, 8, 13, 14

[62] Jianing Yang, Alexander Sax, Kevin J Liang, Mikael Henaff, Hao Tang, Ang Cao, Joyce Chai, Franziska Meier, and Matt Feiszli. Fast3r: Towards 3d reconstruction of 1000+ images in one forward pass. *arXiv preprint arXiv:2501.13928*, 2025. 4

[63] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 8

[64] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 1053–1061, 2024. 4

[65] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *ICCV*, 2023. 4, 5, 6, 7, 8, 12

[66] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 8

[67] Wangbo Yu, Jinbo Xing, Li Yuan, Wenbo Hu, Xiaoyu Li, Zhipeng Huang, Xiangjun Gao, Tien-Tsin Wong, Ying Shan, and Yonghong Tian. Viewcrafter: Taming video diffusion models for high-fidelity novel view synthesis. *arXiv preprint arXiv:2409.02048*, 2024. 1, 5, 6, 8

[68] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In *NeurIPS*, 2022. 8

[69] Jason Y. Zhang, Amy Lin, Moneish Kumar, Tzu-Hsuan Yang, Deva Ramanan, and Shubham Tulsiani. Cameras as rays: Pose estimation via ray diffusion. In *ICLR*, 2024. 4

[70] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *CVPR*, 2023. 5

[71] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[72] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024. 8