A. Appendix

A.1. Data Details

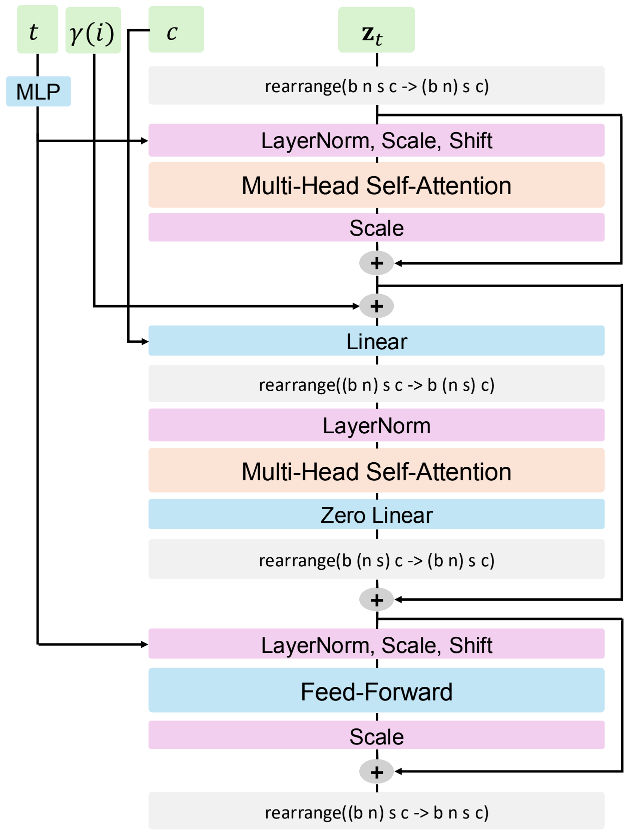
To collect our large-scale dataset of image pairs, we utilize DL3DV10K [30] and ScanNet++ [65] benchmarks. For DL3DV10K, we select $k \in [6, 36]$ equally spaced views as the initial sparse training set. We use the 960P resolution images and undistort them before reconstruction. We filter sequences based on scale factor plausibility, *i.e.* we remove sequences with very large or small scale factors. This makes our calibration pipeline robust to inaccuracies in initial camera parameters or depth estimates. We store the reconstructions in a compressed format following [38]. For ScanNet++, we first use farthest point sampling to determine a small number of keyframes and subsequently select 25% - 50% of the remaining training set closest to these keyframes. This ensures good spatial coverage while also promoting target views far from the initial training views, even in a dense view setting. Additionally, we use the out-of-distribution test views of the training sequences. We undistort the images and resize them to 640×960. The resulting dataset covers a wide variety of input rendering quality for training, with initial PSNR values ranging from 5 to more than 30.

A.2. Method Details

Camera selection. For unordered view sets, we exploit the fact that our reconstructions are metric scale, which allows us to choose a reasonable range for the radius of the sphere that the candidate poses lie on. In practice, we found a range of $[0.2, 0.5]$ to work well, while for the orientation we use a random perturbation within $[0, 30]$ degrees in yaw and pitch.

Initial reconstruction. Note that before running point triangulation, we check if there are sufficient reliable feature tracks inferred from the MASt3R matches, and if there are not enough feature tracks, we resort to global pointcloud alignment [28] to ensure sufficiently dense 3D geometry estimates. This is usually only the case for very sparse input view scenarios, *i.e.* 9 input views or less.

Training details. For training our flow model, we found it important to choose the right camera selection strategy to ensure both sufficient viewpoint variability and high covisibility between input views. Therefore, given a pool of target view candidates based on spatial similarity, we randomly sample the pool of candidates. Then, we select k reference views based on a k-means clustering of the target views in a 6D space consisting of position and look-at direction. Given the k clusters, we can choose the reference view closest to each cluster center. For timestep scheduling during training, we follow [15] and sample t from a logit normal distribution with constant shift. Furthermore, we sample t independently for each target frame [8], which we found to speed up convergence.



t
$\gamma(i)$
c
$\mathbf{z}_t$
MLP
rearrange(b n s c -> (b n) s c)
LayerNorm, Scale, Shift
Multi-Head Self-Attention
Scale
+
+
Linear
rearrange((b n) s c -> b (n s) c)
LayerNorm
Multi-Head Self-Attention
Zero Linear
rearrange(b (n s) c -> (b n) s c)
+
LayerNorm, Scale, Shift
Feed-Forward
Scale
+
rearrange((b n) s c -> b n s c)

Figure 5. **Illustration of our multi-view DiT block.** Here, we denote ray map embeddings as c.

Num. views	GPU mem. (GB)	PSNR ↑	SSIM ↑	LPIPS ↓
72	42.3	24.15	0.808	0.180
12	17.0	24.29	0.801	0.175

Table 7. **Memory consumption vs. performance across input view counts.** We show our model is effective both at high and low number of views on DL3DV140 at 512px width (cf. Tab. 5).

Architecture details. The overall architecture of our model follows [15]. To compress the input images into latents, we use a VAE [48] with latent dimension of 16. The latents are patchified with a patch size of 2 and fed to our latent flow matching model, which consists of 24 DiT blocks [42]. Each block has 24 attention heads with dimension 64, and the feedforward network has the same hidden dimension of 1536. We show an illustration of our multi-view DiT block architecture Fig. 5. As mentioned in Sec. 3.3, we keep the first self-attention layer, the normalization layers, and the feed-foward layer equivalent to [15], and insert a multi-view attention layer after the first attention layer. This multi-view attention is concluded with a zero linear layer to keep the initialization intact. As mentioned in Sec. 3.2, for each source input view i, we condition the model on its camera

	$\mathcal{L}_{\text{tgt}}$	Re-init.	12-view PSNR ↑	12-view SSIM ↑	12-view LPIPS ↓	24-view PSNR ↑	24-view SSIM ↑	24-view LPIPS ↓
1	-	-	22.18	0.760	0.314	24.88	0.831	0.243
2	✓	-	22.35	0.763	0.285	25.10	0.835	**0.212**
3	✓	✓	**22.43**	**0.766**	**0.280**	**25.13**	**0.836**	**0.212**

Table 8. **Refined reconstruction $\mathcal{G}_{\text{tgt}}$ ablation breakdown**. We report the scores on both DL3DV [30] view splits. Incorporating generated views in pointcloud initialization (Re-init.) benefits view synthesis, while the improvement is more pronounced in the 12 view setting.

Method	3-view PSNR ↑	3-view SSIM ↑	3-view LPIPS ↓	6-view PSNR ↑	6-view SSIM ↑	6-view LPIPS ↓	9-view PSNR ↑	9-view SSIM ↑	9-view LPIPS ↓
ZipNeRF [3]	12.77	0.271	0.705	13.61	0.284	0.663	14.30	0.312	0.633
ZeroNVS [49]	14.44	0.316	0.680	15.51	0.337	0.663	15.99	0.350	0.655
ReconFusion [61]	15.50	0.358	0.585	16.93	0.401	0.544	18.19	0.432	0.511
CAT3D [19]	16.62	0.377	0.515	17.72	0.425	0.482	18.67	0.460	0.460
FlowR	14.46	0.347	0.587	16.18	0.409	0.520	17.53	0.456	0.467
FlowR (Initial)	12.77	0.243	0.592	14.40	0.320	0.532	15.67	0.379	0.491

Table 9. **Few-view 3D reconstruction on Mip-NeRF 360 [2]**. We follow the experimental setting of [61].

pose $\mathbf{P}_i$ and intrinsics $\mathbf{K}_i$, *i.e.* a pixel $\mathbf{p}'$ of image i is represented as a ray $\mathbf{r} = \langle \mathbf{o} \times \mathbf{d}, \mathbf{d} \rangle$ with $\mathbf{o} = \mathbf{R}_j^\top(\mathbf{t}_i - \mathbf{t}_j)$ and $\mathbf{d} = \mathbf{R}_j^\top \mathbf{R}_i \mathbf{K}_i^{-1} \mathbf{p}'$ where j is the reference view defining common coordinate frame.

Inference details. When the number of reference views we can fit into a single forward pass is limited, we apply the same reference view selection strategy as in training. For target view selection, we use the method described in Sec. 3.3 and use B-spline basis functions of degree 2. When inferring our model, we use 20 timesteps in the procedure described in Sec. 2.2, specifically:

$$\mathbf{z}_{t+\Delta t} = \mathbf{z}_t + \Delta t \boldsymbol{v}_\theta(\mathbf{z}_t, \mathbf{y}, t), \tag{9}$$

where the step size Δt is chosen empirically as a monotonically decreasing function of t [15].

Runtime and memory analysis. As mentioned in Sec. 3.1, our initial reconstruction takes an average of 6.5 minutes for pointcloud initialization (6 minutes) and initial 3DGS training (30 seconds). Our multi-view flow model takes approximately 1.5 minutes to generate 200 additional images, processing 45 views at 540 × 960 resolution (91K tokens) on one H100 GPU at each forward pass. The final reconstruction training takes on average 42.4 minutes, as we use a longer, 30k step schedule and an additional LPIPS loss for the target views. In Tab. 7, we report the memory usage of our model and show that it can run both on high-end and memory-constrained consumer grade GPUs by adjusting the number of input views that are processed in a single forward pass. Our model reaches comparable performance on our DL3DV140 benchmark with 72 and 12 views, respectively.

Limitations. While FlowR makes a significant step towards high-quality, photo-realistic 3D reconstructions from non-exhaustive captures, there remain meaningful directions for future work. For instance, our method relies on heuristics to select the camera views that are used to refine the 3D reconstruction results. In this regard, incorporating uncertainty quantification [20, 24] and active view selection [40] could improve results. Additionally, because our method aims to map incorrect renderings to ground-truth images, its performance depends on the initial 3D reconstruction. In particular, if there are large areas entirely unseen in the source views, our model will not hallucinate new content. Incorporating a suitable prior for such cases opens up a promising avenue for future research. Finally, we focus on static scenes, as 3DGS exhibits major artifacts with dynamic objects. Hence, another meaningful direction for future work is to extend our method to dynamic scenes using a dynamic 3DGS method, which requires adapting the training data to reflect the presence of dynamic objects in both the source and target distributions.

A.3. Additional Experiments

Evaluation details. We use LPIPS with VGG-16 features unless otherwise specified in the benchmark, *i.e.* we use VGG-16 for all experiments except for the Nerfbusters benchmark which uses AlexNet. Note that for Nerfbusters we resort to the test trajectory for target view selection since the test views are entirely disconnected from the initial reconstruction and as such it is not possible to effectively refine the reconstructed scene by interpolating along the training trajectory or by sampling poses around it. Finally, we optionally apply naive opacity thresholding, *i.e.* we define a single minimum opacity value applied to all rendered views to achieve a comparable coverage to our baselines. The intuition behind this is that high opacity along a pixel ray usually correlates with well-defined scene geometry.

Comparison to closed-source methods. In Tab. 9, we compare with closed-source methods such as ReconFusion [61] and CAT3D [19] using their provided data splits in MipNeRF360 [2]. For a fair comparison, we choose a similar camera selection strategy as CAT3D, where we generate an elliptic trajectory on a hemisphere around the common look-at point of the initial cameras. We note that the evaluation setting of [61] is distinct from ours since the training and evaluation splits are chosen so that there is a large fraction of the scene in the test views that was *not observed in the training views*. As such, an evaluation with view synthesis metrics is only approximate, as there are many plausible 3D scenes for a set of partial observations. We show that our method, despite not being tailored for scene extrapolation as mentioned in Appendix A.2, performs competitively to prior works. We further observe that the gap between our method and the state-of-the-art approach narrows when increasing the number of input views, where our method is almost on-par with CAT3D [19] in terms of SSIM and LPIPS in the 9-view setting.

Refined reconstruction ablation breakdown. As mentioned in Sec. 4.2, we observed that the benefit of incorporating generated views into the pointcloud reconstruction is more pronounced in the 12-view setting, as can be seen in Tab. 8 where we provide a breakdown of the two splits. We attribute this to the fact that with an increasing number of co-visible views, there are enough reliable matches to triangulate a good initialization from the source input images and adding more views therefore ceases to improve results.