GenFusion: Closing the loop between Reconstruction and Generation via Videos

Supplementary Material

A. Overview

In the supplementary materials, we provide comprehensive experimental details and extensive ablation studies to evaluate the contributions of our framework designs. Additionally, we present qualitative comparisons between our approach and baseline methods.

B. Experimental Details

B.1. Video Diffusion Model Details

Our diffusion model is built upon a pre-trained image-to-video latent diffusion model [7] which operates on RGB latent space. However, we found that relying only on RGB inputs fails to produce consistent video frames, particularly in regions with severe artifacts. Therefore, we leverage depth maps to inject geometry information into the diffusion model. To process RGB-D inputs, we utilize a pre-trained VAE from LDM3D [6], which is designed to encode RGB-D image into the latent space. Therefore, given an RGB-D video of size $4\times T\times 512\times 320$ (T: video length), we flatten it along the first two dimensions, encode it into latent features of shape $4T\times 64\times 40$, and reshape to $4\times T\times 64\times 40$ for diffusion. For CLIP feature embedding, we randomly sample a reference frame from the input sequence. During reconstruction, the nearest input frame to the target trajectory serves as the reference for the CLIP guidance. In the training process, each training example comprises an artifact-prone RGB-D video, a reference image, and one target RGB-D video. To obtain the temporally consistent depth map for training, we leverage the SOTA monocular depth estimatior [1] to augment the training data. During inference, we employ DDIM sampling with classifier-free guidance to modulate condition adherence strength. To do so, we implement random dropout of conditioning images with 10% probability per sample during training.

In the video diffusion experiment section, we explore different designations of diffusion model to identify the optimal balance between model performance and computational efficiency. Therefore, four diffusion models are trained and analyzed in three aspects, input type, resolution, and video length. To this end, the base model that generates 16 frames of videos with a resolution of 512×320 is trained for 30k iterations using a learning rate of $1e-5$ and a batch size of 2 on each GPU. To assess the impact of depth information, we conduct a comparative analysis by training two base models: one utilizing RGB-D inputs and another with RGB inputs only. Both models are trained under identical hyperparameter settings to ensure a fair comparison. To enhance the quality of generated videos, we fine-tune the base RGBD model for higher resolution inputs (16 frames at 960×512) with an additional 34k iterations, maintaining the same learning rate and batch size configurations. To extend video generation capabilities, we fine-tune the temporal layers of our base model to produce 48-frame sequences for 30k iterations while maintaining the base model's batch size and learning rate.

B.2. Masked 3D Reconstruction

In the main paper, we introduce a masked 3D reconstruction scheme to mime the far-field rendering artifacts. The marked 3D reconstruction is used in both video diffusion data generation and novel view synthesis evaluation. In practice, we use a patch mask of size $H/2\times W/2$ to enable narrow field-of-view inputs in both settings. But differently, we randomly select one of the four corner locations for training dataset generation, since fixing the mask location introduces diverse artifacts and under-observed regions, enriching the dataset's complexity. However, the extremely limited observation setup often produces large black regions near the boundaries. Using such data directly for evaluation can lead to unrealistically low quantitative metrics in these regions due to content ambiguity. To enable a fair comparison, we generate a trajectory to move the mask over time, rather than fixing its location as in video data generation. This ensures that most scene content is included in the input. Notably, all baselines and our method use the same sampling trajectories for each scene. To further reduce sparsity along the camera trajectory, we downsample the viewpoints by factors of 2 and 4, using these masked frames as our training input while using the remaining full frames for evaluation.

B.3. Cyclic Fusion

We close the loop between reconstruction and generation through cyclic fusion that updates the 3D scene representation (i.e. 2D Gaussian primitives) using input captures and generated videos.

Warm-up: During the warm-up phase of the fusion process, the 3D representation is updated exclusively from input captures for the first 1000 iterations. Afterward, we apply our reconstruction-driven video diffusion every 1000 iterations to remove the artifacts and generate new content for the video renderings, which are then added to the training view set.

Sparsity-aware Densification: In the original Gaussian Splatting [3], scene primitives are cloned and split based

No.	Method	PSNR↑	SSIM↑	LPIPS↓
1	2DGS baseline	13.87	0.572	0.447
2	+train view monocular depth	13.89	0.575	0.442
3	+sample view rgb	15.33	0.602	0.442
4	+sample view depth	15.34	0.622	0.438
5	+sparsity aware densification	15.81	0.617	0.409

Table 1. Ablation studies using on Tanks and Temples dataset. ↑ indicates higher is better, while ↓ indicates lower is better.

on the average magnitude of view-space position gradients, and the gradient for each primitive is reset every K steps (i.e., 100 steps in 3DGS and 2DGS). We find this strategy performs well in scenarios where the scene is densely captured. In such cases, primitives are typically observed for more than half of the reset steps ($> \frac{K}{2}$), making the averaged gradient over K steps a reliable indicator for deciding whether to add the primitive to the densification list. However, this strategy becomes unreliable for masked 3D reconstruction, as the visibility counts of each Gaussian primitive are significantly lower, resulting in unstable gradient accumulation. To address this, we propose a sparsity-aware densification strategy that maintains the densification list by incorporating minimal visibility counts. Specifically, we disable gradient resets and add a primitive to the densification list only if its gradient exceeds the threshold and its visibility count surpasses the minimal visibility requirement. Accordingly, we perform the densification process every 100 iterations to progressively refine the point cloud representation. We found this strategy is more robust for handling diverse input scenarios.

C. Ablation Studies

In Table 1, we perform comprehensive ablation studies to validate the contributions of our model components using scenes from the Tanks and Temples dataset[4]. We begin with a vanilla 2D Gaussian Splatting (2DGS) model, following its original implementation, as the baseline. Building on this, we evaluate the effect of incorporating monocular depth supervision during training and view sampling using the ScaleAndShiftInvariant loss [5]. As shown in (2) of Table 1, this addition does not yield quantitative improvements. However, it encourages smoothness in the rendered depth, effectively reducing floating artifacts typically observed during initial reconstruction stages (visualized in Figure 1). Significant performance gains are observed in (3) and (5) of Table 1, attributed to our RGB regularization and sparsity-aware densification strategies, further confirming the effectiveness of our method.

D. More Evaluation

D.1. View Interpolation

Table 4 provides a per-scene break down for quantity metrics in Mip-NeRF360. These results showcase that our models consistently improve the baselines.

D.2. View Extrapolation and Scene Completion

Here we present extensive experimental results on masked 3D reconstruction. Figure 2 demonstrate that our performance also outperforms baselines in far-field viewpoint renderings. Table 3 and Table 2 provide per-scene quantitative results.

E. Conclusion

We have observed viewpoint saturation as a fundamental limitation in previous reconstruction and generation methods: high-quality reconstruction relies on dense captures, while generation methods are optimized for weak conditioning. To relax this constraint, we propose GenFusion, an efficient generative guidance framework that enables accurate 3D reconstruction and content generation for input conditions across varying densities. We achieve this by closing the loop between reconstruction and generation, creating a feedback loop where generation becomes aware of the reconstruction status through novel trajectory rendering, and reconstruction is further regularized using RGB-D videos generated by our video diffusion model. We evaluate the interpolation capability using a sparse view reconstruction setup and the extrapolation capability with a novel masked reconstruction mechanism. Both tasks demonstrate significant improvements over baseline methods. In addition, our approach achieves scene-level 3D completion, enabling 3D scene expansion. We hope our findings in bridging reconstruction and generation can inspire other novel view syntheses and 3D scene generation tasks.

Figure 1. From top to bottom: 2DGS baseline, with train view monocular depth added, with sample view RGB added, with sample view depth added, and finally with sparsity-aware densification.

Figure 2. Qualitative comparison of novel view synthesis using masked input on TnT scenes [4].

	2DGS			3DGS			FSGS			Ours		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
14eb48a50e	16.44	0.690	0.384	17.40	0.730	0.360	17.64	0.690	0.434	19.92	0.767	0.351
0a1b7c20a9	15.74	0.733	0.278	16.38	0.760	0.263	18.17	0.752	0.310	19.27	0.811	0.230
06da796666	15.42	0.672	0.396	15.34	0.698	0.390	17.02	0.710	0.437	18.54	0.755	0.376
389a460ca1	18.04	0.810	0.309	18.24	0.824	0.311	18.48	0.799	0.367	21.11	0.861	0.273
2cbfe28643	16.09	0.782	0.257	16.79	0.799	0.254	19.50	0.790	0.321	22.03	0.850	0.227
374ffd0c5f	19.85	0.780	0.256	21.16	0.803	0.250	20.98	0.763	0.327	22.35	0.842	0.224
5c3af58102	15.66	0.692	0.273	15.95	0.709	0.260	16.22	0.661	0.325	20.10	0.794	0.214
66fd66cbed	21.42	0.855	0.235	22.15	0.873	0.224	22.29	0.867	0.246	23.27	0.897	0.191
3bb3bb4d3e	16.89	0.795	0.266	17.85	0.810	0.253	18.84	0.780	0.319	22.48	0.883	0.198
91afb9910b	19.18	0.765	0.274	19.91	0.773	0.278	20.86	0.776	0.304	22.76	0.820	0.240
7705a2edd0	16.74	0.698	0.398	16.78	0.712	0.396	18.89	0.715	0.440	21.71	0.792	0.350
71b2dc8a2a	15.67	0.796	0.264	15.94	0.814	0.252	20.42	0.857	0.252	21.64	0.887	0.199
a726c1112a	18.60	0.804	0.321	19.45	0.832	0.295	17.02	0.726	0.423	20.00	0.83	0.297
cbd44beb04	16.46	0.700	0.311	17.40	0.728	0.299	17.35	0.706	0.349	19.38	0.789	0.285
df4f9d9a0a	17.21	0.743	0.358	18.04	0.768	0.344	19.36	0.777	0.356	21.82	0.845	0.268
6d22162561	15.79	0.663	0.398	16.62	0.681	0.401	18.07	0.671	0.441	20.58	0.737	0.372
6d81c5ab0d	13.19	0.540	0.448	14.22	0.601	0.425	14.47	0.573	0.492	16.42	0.634	0.448
ec305787b7	16.75	0.751	0.286	16.98	0.763	0.278	16.78	0.688	0.381	22.122	0.846	0.211
85cd0e9211	18.17	0.758	0.285	18.45	0.767	0.289	18.90	0.688	0.372	22.73	0.814	0.269
95e4b24092	13.97	0.581	0.353	13.66	0.596	0.351	14.99	0.598	0.373	15.99	0.609	0.345
7da3db9905	16.51	0.737	0.309	18.69	0.778	0.285	19.98	0.765	0.314	22.09	0.831	0.231
d3812aad53	15.09	0.607	0.454	16.16	0.654	0.438	16.80	0.662	0.451	17.43	0.684	0.428
b0c4613d6c	15.10	0.612	0.332	15.54	0.623	0.336	17.71	0.637	0.368	19.00	0.668	0.324
b4f53094fd	13.48	0.634	0.306	14.28	0.653	0.299	17.17	0.677	0.311	18.59	0.698	0.271
average	16.56	0.717	0.323	17.22	0.740	0.314	18.25	0.722	0.363	20.47	0.788	0.284

Table 2. Quantitative comparison on DL3DV datasets. Each method is trained on 7000 steps.

	2DGS			3DGS			FSGS			Ours		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
barn	16.80	0.675	0.371	17.64	0.685	0.377	18.47	0.677	0.405	17.84	0.672	0.402
ignatius	15.75	0.588	0.329	15.88	0.591	0.359	16.14	0.521	0.458	17.51	0.614	0.363
meetingroom	17.63	0.672	0.364	17.80	0.694	0.356	17.71	0.667	0.421	19.37	0.733	0.348
truck	14.66	0.646	0.357	15.39	0.663	0.361	16.69	0.654	0.407	16.80	0.673	0.383
courthouse	14.80	0.630	0.411	15.15	0.640	0.419	15.80	0.632	0.454	15.68	0.622	0.461
caterpillar	13.79	0.532	0.403	14.33	0.542	0.431	15.35	0.530	0.490	16.58	0.580	0.432
train	13.77	0.561	0.423	14.31	0.587	0.424	14.47	0.528	0.516	15.34	0.580	0.458
average	15.31	0.615	0.380	15.79	0.629	0.390	16.38	0.601	0.450	17.01	0.639	0.406

Table 3. Quantitative comparison on TnT datasets. Each method is trained on 7000 steps with 1/2 frames

	2DGS			3DGS			FSGS			Ours		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3 Views												
bicycle	12.70	0.124	0.622	14.33	0.300	0.556	14.30	0.234	0.624	15.46	0.275	0.647
bonsai	11.60	0.300	0.568	10.92	0.301	0.736	13.75	0.376	0.524	14.12	0.418	0.534
counter	13.17	0.311	0.539	12.62	0.305	0.597	13.99	0.392	0.527	15.20	0.470	0.520
garden	13.06	0.184	0.575	12.08	0.145	0.649	14.33	0.274	0.586	16.65	0.305	0.580
room	13.79	0.410	0.490	13.04	0.342	0600	14.26	0.483	0.484	16.40	0.570	0.438
stump	14.63	0.171	0.593	14.10	0.196	0.626	15.93	0.276	0.607	17.13	0.317	0.640
kitchen	14.07	0.307	0.542	13.35	0.257	0.621	14.76	0.361	0.538	16.02	0.427	0.542
flowers	10.57	0.104	0.657	10.08	0.129	0.794	12.17	0.177	0.664	12.89	0.210	0.715
treehill	11.95	0.186	0.627	11.22	0.200	0.793	14.10	0.290	0.647	12.89	0.326	0.652
average	13.06	0.318	0.576	13.07	0.243	0.580	14.17	0.318	0.578	15.29	0.367	0.585
6 Views												
bicycle	14.35	0.188	0.576	12.92	0.181	0.663	15.76	0.294	0.597	16.52	0.311	0.604
bonsai	14.77	0.471	0.457	13.07	0.373	0.602	16.67	0.546	0.436	16.55	0.557	0.441
counter	15.09	0.428	0.467	13.77	0.352	0.535	16.02	0.495	0.449	16.99	0.545	0.428
garden	16.06	0.308	0.465	14.03	0.201	0.569	17.57	0.401	0.504	18.74	0.406	0.490
room	14.80	0.481	0.446	13.98	0.426	0.564	15.22	0.542	0.443	17.54	0.623	0.410
stump	16.13	0.229	0.556	14.62	0.201	0.609	17.58	0.323	0.582	18.36	0.343	0.585
kitchen	17.12	0.494	0.397	15.11	0.321	0.530	17.64	0.577	0.374	18.54	0.560	0.390
flowers	11.89	0.145	0.607	10.89	0.147	0.757	13.21	0.211	0.649	14.01	0.237	0.658
treehill	13.33	0.240	0.584	12.10	0.222	0.741	15.46	0.347	0.613	15.36	0.363	0.605
average	14.96	0.355	0.505	15.02	0.338	0.506	16.12	0.415	0.517	17.16	0.447	0.500
9 Views												
bicycle	15.30	0.237	0.536	13.53	0.213	0.648	17.15	0.343	0.577	17.10	0.332	0.578
bonsai	17.43	0.609	0.373	15.51	0.460	0.482	19.30	0.669	0.356	19.31	0.662	0.354
counter	16.42	0.516	0.406	14.54	0.391	0.493	17.63	0.572	0.391	18.23	0.607	0.379
garden	18.10	0.412	0.397	15.06	0.250	0.522	19.22	0.477	0.455	19.97	0.470	0.446
room	17.36	0.600	0.370	15.49	0.492	0.499	18.16	0.662	0.359	19.75	0.700	0.366
stump	17.45	0.300	0.514	15.69	0.237	0.548	18.72	0.386	0.555	19.40	0.392	0.553
kitchen	19.17	0.611	0.324	16.21	0.393	0.473	20.30	0.682	0.305	20.59	0.640	0.322
flowers	13.01	0.191	0.564	12.01	0.163	0.695	14.33	0.247	0.629	14.95	0.267	0.629
treehill	14.34	0.300	0.555	13.23	0.265	0.733	15.46	0.347	0.613	15.98	0.390	0.595
average	16.79	0.447	0.446	16.67	0.423	0.449	17.94	0.492	0.471	18.36	0.496	0.465

Table 4. Per-scene Quantitative comparison on sparse view reconstruction

References

[1] Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. Depthcrafter: Generating consistent long depth sequences for open-world videos. *arXiv preprint arXiv:2409.02095*, 2024. 1

[2] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *SIGGRAPH Asia*, 2024. 3

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. on Graphics*, 2023. 1, 3

[4] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. 2, 3

[5] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. 2020. 2

[6] Gabriela Ben Melech Stan, Diana Wofk, Scottie Fox, Alex Redden, Will Saxton, Jean Yu, Estelle Aflalo, Shao-Yen Tseng, Fabio Nonato, Matthias Müller, and Vasudev Lal. LDM3D: latent diffusion model for 3d. *arXiv preprint arXiv:2305.10853*, 2023. 1

[7] Jinbo Xing, Menghan Xia, Yong Zhang, Haoxin Chen, Wangbo Yu, Hanyuan Liu, Xintao Wang, Tien-Tsin Wong, and Ying Shan. Dynamicrafter: Animating open-domain images with video diffusion priors. *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 1

[8] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. FSGS: real-time few-shot view synthesis using gaussian splatting. In *Proc. of the European Conf. on Computer Vision (ECCV)*, 2024. 3