(Appendix) Splatography: Sparse multi-view dynamic Gaussian Splatting for filmmaking challenges

1. Overview

The appendix contains:

1. Additional details on the proposed model in Sec. 2
2. Additional details on the experimental set-up in Sec. 3, including information on the optical flow metric used in the ablations
3. Additional results in Sec. 4, including per-frame statistics for all scenes, frame-by-frame comparisons of various scenes and novel-view renders
4. Finally, in Sec. 5 we provide an in-depth discussion. We review related work on static SV3D - fixating on novel primitives, surface regularization and generative model intervention. We present issues relating to Gaussian point count and emphasize the need for approximating point count. We discuss greater issues with depth-based approaches for regularizing geometry, with example. Finally, we evaluate the relationship between video duration and plane-based feature spaces

2. Additional Model Details

In this section we provide additional information on the formulation of our dynamic SV3D reconstruction pipeline.

2.1. Canonical MipSplatting Representation

[19] relies on the Nyquist-Shanon sampling theorem [10, 14], demonstrating that Equation 2 is sensitive to the sampling rate (the ratio between Σ and the pixel width/sampling interval). To address aliasing artifacts, the method proposes a 3-D Gaussian filter that constrains the highest frequency component (as determined by σ and s) to lie below the Nyquist limit for at least one camera. This prevents high-frequency artifacts when rendering at low resolution (e.g. during zoom-out or distant views). Also, a 2-D Mip filter is proposed to handle under-sampling issues when rendering scenes at higher resolutions (e.g. zoom-in or close-up views). This requires tracking an additional parameter for modulating point opacity and scale during training. We select MipSplatting as the canonical representation for our method as it has shown to be capable on datasets with non-salient object scale across views, which is expected of sparse view datasets.

2.2. Reducing Computation.

Our method uses two sets of hex-planes and two sets of canonical Gaussians. This requires more computation than the original 4D-GS model, which only uses one set of each. To deal with computation we do the following. Firstly, instead of modeling P_ψ^f, P_ψ^b as multi-scale feature planes as in [5], we model them as a set of learnable 2-D wavelet coefficient parameters as in [1]. This reduces GPU utilization by modeling high resolution feature grids as lower resolution 2-D wavelet coefficients. Secondly, backgrounds are typically much larger in spatial extent than foregrounds and would traditionally require a higher density of points to accurately reconstruct. However, as backgrounds are often non-salient across most views and only required for edge cases, e.g. modeling transparent materials, they can be overfitted without corrupting the quality of the foreground reconstruction. Therefore, for SV3D problems, this makes it reasonable to model the background using a sparse set of points, which simplifies computation without significantly compromising visual quality.

Our method requires little GPU (< 3 GB) and CPU (< 8 GB) memory to handle 10 views at a 2K pixel resolution with a batch size of 2 images per training iteration.

2.3. Regularization

In addition to the proposed losses and regularizers we also use the PhysGaussian regularizer [16],

$$\mathcal{L}_{pg} = |\text{argmax}(\frac{\text{argmax}(s)}{\text{argmin}(s)}, k) - k|, \tag{1}$$

where k is the desired ration between the highest and lowest orthogonal scale axis values. This constrains the ratio of the largest and smallest scale axis' to be equivalent to $k = 10$, disincentivizing spiky Gaussians; a notable problem in SV3D. NeuSG [3] instead propose to minimize the smallest scale component to achieve the same objective, though for the datasets we tested this has negligible effect on the final results.

2.4. Sparse Mask Creation

For DyNeRF and ViVo datasets there are four and ten images to mask per scene, respectively. Due to the sparsity of

Hand Drawn SAM-web SAM-demo

Figure 1. Comparison of masks generated using hand-drawn (left), SAM-web (middle) and SAM-demo (right). Note that it was not possible to select the square performance region for automated segmentation

this data we could have either created masks with generative methods or hand-draw the masks. Considering the application of interest, filmmaking, we chose to draw masks by hand. This is a common skill in cinematographic applications, it requires minimal technical expertise and provides precision in selecting mask boundaries, especially in regions that are challenging to segment. We used Adobe Photoshop (a common tool for roto-scoping). For simple scenes (e.g. the Pony-ViVo scene) the drawing process took < 1 minute per image and for more complex scenes (e.g. the Flame-Steak-DyNeRF scene) this took < 3 minutes. Note that we are not professionals and have limited experience rotoscoping so we anticipate these times to be lower in practice.

Generative methods could also be used to accomplish the same goal, however it is worth considering: (1) rendering times can take up to 1 minute per image, and (2) in scenarios where RTD textures appear, or capture artifacts exist due to set lighting, the segmentation may suffer. We demonstrate this in Fig. 1, where we compare the hand drawn masks capturing the entire performance area, to two different API implementations of the SAM model [13]: SAM-web[1] and SAM-demo[2]. Note that it was not possible to select the square stage area (the perimeter is marked by red-take) with the SAM APIs as they failed to differentiate between the stage and surrounding floor.

It is also worth noting that the masks used for testing foreground predictions for the ViVo and DyNeRF dataset are generated using SAM2 [13]. SAM2 operates on video so it takes longer to process and generate masks (> 3 minutes per 300 frame video at 2K). Though, we find the masks are more reliable than the masks in Fig. 1.

[1]Accessed: https://huggingface.co/spaces/Xenova/segment-anything-web on 15 August 2025

[2]Accessed: https://segment-anything.com/demo on 15 August 2025

3. Additional Experiment Details

In this section we provide additional details concerning the experimental results in the main paper.

3.1. ViVo Dataset

Information on Test Data. The test images from the ViVo dataset are masked using SAM2 [13] and dilated by a kernel of 5×5 pixels to ensure complex edges are captured during evaluation. In principle, this allows for a better approximation of foreground edge details, where background details present in dilated edge regions will minimally influence the final metrics.

Benchmark Settings. We trained and tested on the full 300 frames at 2K resolution. Each benchmark was tuned to produce the best results. All methods used the same initial point cloud.

Optical Flow Metric. To assess the temporal geometry on a finer level we employ the following metric. We define a pseudo ground truth optical flow estimation $\Omega^* \in \mathcal{R}^{N-1,H,W}$ for N frames, using off-the-shelf RAFT [11] implementations provided by PyTorch. The optical flow of each method is also predicted, as Ω. We then calculate the MSE between the two parameters on a pixel level, using $||\Omega^* - \Omega||$.

3.2. DyNeRF Dataset

Grid Plane Settings. For all grid-based dynamic GS methods, we set the resolutions of the grid axis to $\{x, y, z, t\} = \{128, 128, 128, T/2\}$, where $T/2$ indicates half the frame count - a rule of thumb proposed in [5, 15]. This allows the temporal axis to be smooth and continuous. The resolution of the space axis' are the same settings used in [5, 9, 15] for their benchmarks.

Selecting Training and Test Cameras. In principle, forward-facing (2.5-D) scenes are easier to reconstruct as the background view is mostly shared between training cameras. To make this more challenging, we select only four cameras at the most distance positions from the test camera. This minimizes positional biases between training and test camera placements.

Benchmark Settings. We train and evaluate images with a downsampling factor of 2, making the resolution 1080p. Each method was tuned to produce the best results. All methods used the same initial point cloud. However, for

our method we downsampled the number of initial background points to enhance the separation of foreground and background points; as per our canonical training strategy laid out in the main paper.

4. Additional Results

In this section we present additional visual and metric results, demonstrating the quality of our approach.

4.1. Per-Scene Metrics

In Tabs. 1 and 2 we present the full metric results for the ViVo and DyNeRF datasets. We use the following objective metrics, PSNR, SSIM, LPIPS-Alex and LPIPS-VGG.

4.2. Per-Frame Per-Camera Metrics

In Fig. 2 we present the per-frame per-camera results comparing our full and foreground renders with 4D-GS and STG, on PSNR, SSIM, LPIPS-Alex and LPIPS-VGG, for the Bassist and Pianist scenes.

4.3. More Frame-by-Frame Renders

In Figs. 3 to 5, we further compare visual results of our method and SotA. In Fig. 6 we provide additional test frames from our method on the ViVo scenes.

4.4. Novel View Synthesis

In Fig. 7 we present the novel view synthesis renders that are not included in the ViVo test set.

4.5. More on Limitations

Our method produces SotA metric and visual results on dynamic SV3D datasets and also provides the ability to segment important and unimportant features without the need for dense video object segmentation tools (MiVOS, SAM2, etc.). This is especially true in 3-D where it is significantly easier to disentangle features due to larger depth differences between the foreground and background. Inversely, the 2.5-D results show that segmenting a foreground that heavily interacts with the background (through shadows or spatial proximity) is not as straight forward. For the DyNeRF scenes, over-reconstruction occurs around the edges of foreground surfaces that occlude background features at $t = 0$. This indicates that while Eq. 4 in the main paper is suitable for 3-D scenes, it is less suited to the 2.5-D edge case where the foreground and background heavily interact.

Furthermore, while our results on the Vivo dataset demonstrate superiority to the SotA, there is still room for improvement. In XX we show that there are commonalities in the failure cases from 4D-GS and our method. This reveals greater challenges with the SV3D paradigm, primarily relating to the high degree of occlusion in sparsely viewed scenes.

5. Additional Discussion

In this section we present a general discussion on researching dynamic GS pipelines. This may be helpful for informing future work on SV3D. Note, this discussion is not specifically tied to our method but instead focuses on the greater challenge of conducting research.

We cover the following topics and discuss related works in the subsequent sub-sections:

- Static GS solutions do not consider dynamic features, like dynamic textures or non-rigid motions
- The number of Gausians is heavily tied the final quality
- Real depth priors are problematic
- Relationship between video duration and plane-based feature spaces

5.1. Static GS Problem

Static GS is specially relevant to deformable dynamic GS representations, as a static GS is required to model the canonical field, as in our approach. Unlike dynamic SV3D, for static scenes there exist a broad range of solutions. We organize the approaches into three groups:

1. Novel Gaussian-like primitives and rendering enhancements
2. Geometric enhancements via self-supervised regularization and/or adaptive density control (ADC)
3. Geometric enhancements with generative model supervision

Enhancements via primitives and rendering for SV3D mainly focus on correcting the shape of the 3-D gaussian. The original Gaussian is a 3-D ellipsoid with a diffuse surface, making rigid surface reconstruction challenging without forcing Gaussians to be small and dense - mimicking a hard surface at scale. Mip-Splatting resolves this by factoring in view-dependent scaling and opacity regulation to allow surfaces to appear denser/sparser under different visual contexts (e.g. zooming in/out). Instead, Gaussian Opacity Fields (GOF) [20] propose a surface-based representation that focuses on mesh reconstruction from GS representation to produce better surface geometry. While successful, it can not be elevated to dynamic use cases as realistic dynamic textures, like fire and smoke, do not have a "surface" thus can not be represented by meshes. This is a leading reason why we chose Mip-Splatting over other similar innovations in static GS research.

Enhancements via surface regularization and ADC focus on modeling surface-like behaviors and correcting gradient over-smoothing issues with the proposed ADC in 3DGS. Physgaussians [16] presents anisotropic scale minimization by evaluating the ratio of the largest and smallest scale axis' and minimizing it with respect to a predefined

Method	PSNR		SSIM		LPIPS-Alex		LPIPS-VGG	
	Full	Mask	Full	Mask	Full	Mask	Full	Mask
Bassist								
4D-GS	12.44	20.34	0.6421	0.9224	0.4889	0.1421	0.4755	0.151
STG	11.97	20.39	0.6526	0.931	0.5087	0.1455	0.4717	0.148
SC-GS	13.46	20.00	0.3848	0.8314	0.5227	0.1776	0.5383	0.1904
Ours	16.44	25.51	0.7487	0.9460	0.3464	0.0786	0.3757	0.1190
Pianist								
4D-GS	16.32	24.14	0.7263	0.9412	0.3290	0.094	0.3981	0.1330
STG	15.57	24.02	0.6804	0.936	0.3736	0.1110	0.3985	0.1356
SC-GS	15.92	23.00	0.4574	0.8604	0.4819	0.1651	0.5117	0.1787
Ours	17.99	27.83	0.7791	0.9563	0.2881	0.0596	0.3636	0.1145
Curling								
4D-GS	13.25	20.16	0.6544	0.9411	0.3801	0.1016	0.4284	0.1300
STG	13.09	19.83	0.6688	0.9440	0.3811	0.1046	0.4275	0.1305
SC-GS	12.21	19.30	0.3975	0.8749	0.5032	0.1511	0.5339	0.1731
Ours	13.63	20.55	0.6917	0.3456	0.0812	0.4059	0.1249	
Fruit								
4D-GS	16.40	24.04	0.6369	0.9264	0.3278	0.0836	0.3993	0.1155
STG	15.80	26.84	0.6806	0.9570	0.3375	0.0674	0.4016	0.1080
SC-GS	16.07	24.15	0.4867	0.8984	0.4731	0.1414	0.5131	0.1588
Ours	18.23	31.00	0.7818	0.9754	0.2792	0.0389	0.3572	0.0917
Pony								
4D-GS	12.70	17.42	0.7040	0.9328	0.3900	0.1585	0.4391	0.1950
STG	12.73	17.54	0.6969	0.9312	0.4016	0.1658	0.4387	0.1933
SC-GS	11.40	16.38	0.3954	0.8027	0.5428	0.2281	0.5631	0.2519
Ours	13.98	19.10	0.7500	0.9448	0.3630	0.1378	0.4170	0.1850

Table 1. Full metric results on the ViVo [2] data set

Method	PSNR		SSIM		LPIPS-Alex		LPIPS-VGG	
	Full	Mask	Full	Mask	Full	Mask	Full	Mask
Cook Spinach								
STG	22.00	21.15	0.8589	0.9853	0.1423	0.0131	0.2438	0.0173
ITGS	22.31	25.79	0.8483	0.9880	0.1896	0.0109	0.2816	0.0157
4DGS	25.81	26.35	0.8784	0.9874	0.1058	0.0121	0.2092	0.0166
W4DGS	25.78	26.20	0.8789	0.9874	0.1046	0.0125	0.2082	0.0173
Ours	25.84	25.78	0.8805	0.9867	0.1172	0.0117	0.2328	0.0164
Flame Steak								
STG	21.83	21.23	0.8793	0.9871	0.1193	0.0109	0.2258	0.0140
ITGS	24.03	25.26	0.8618	0.9888	0.1395	0.0087	0.2456	0.0126
4DGS	26.56	28.02	0.8862	0.9896	0.1001	0.0092	0.1984	0.0127
W4DGS	26.53	28.73	0.8905	0.9903	0.1023	0.0082	0.2008	0.0123
Ours	26.98	28.42	0.8923	0.9908	0.1120	0.0080	0.2196	0.0117
Flame Salmon								
STG	18.59	22.75	0.7988	0.9831	0.1952	0.0140	0.2801	0.0158
ITGS	19.51	23.74	0.7836	0.9832	0.2432	0.0146	0.3200	0.0178
4DGS	21.17	24.98	0.7999	0.9844	0.1866	0.0141	0.2717	0.0173
W4DGS	20.65	24.77	0.7962	0.9841	0.1887	0.0145	0.2791	0.0177
Ours	24.41	26.28	0.8606	0.9872	0.1444	0.0110	0.2493	0.0149

Table 2. Full metric results on the DyNeRF [8] data set

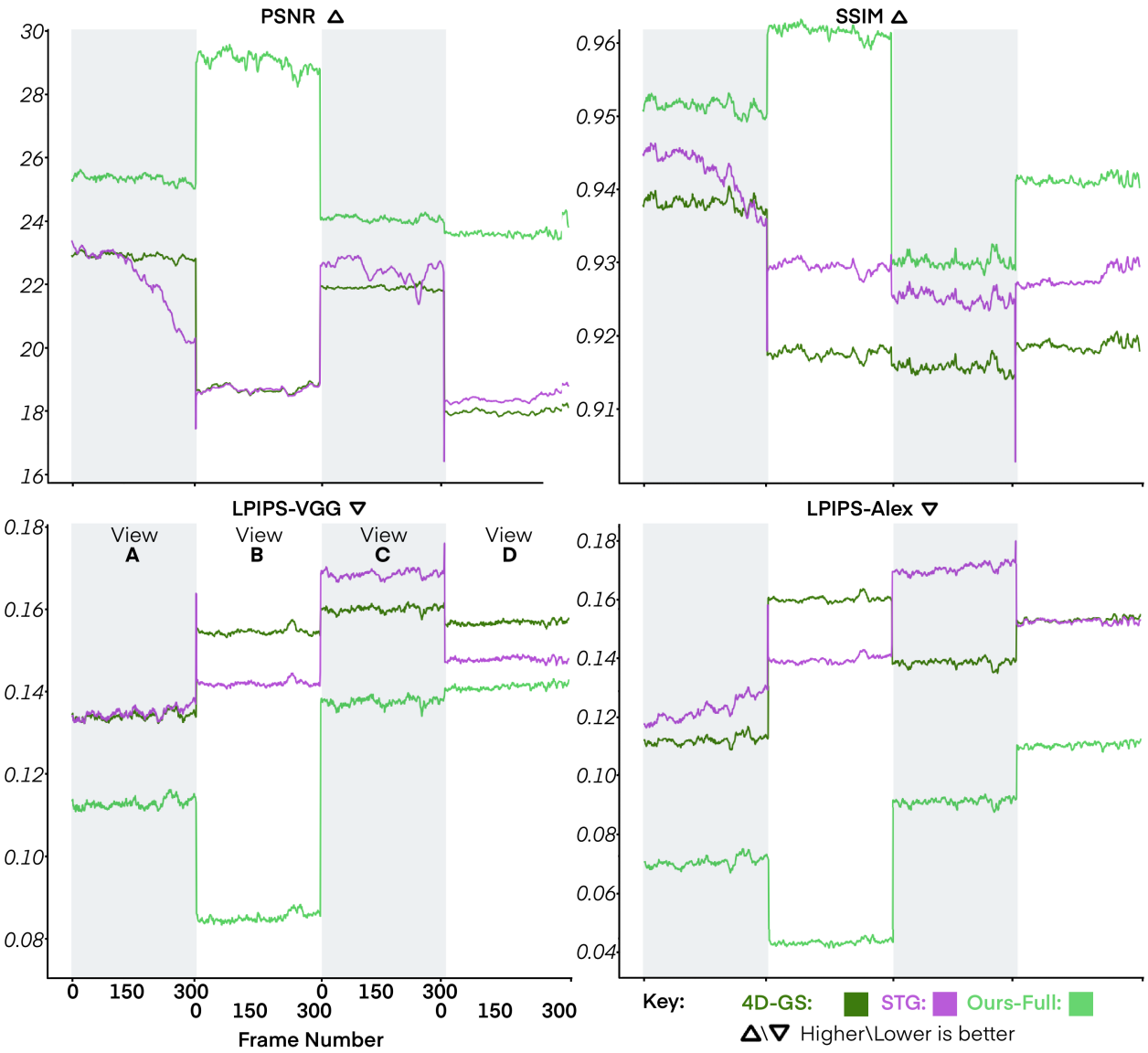

Figure 2. Full per-frame per-camera metric results for the Bassist-Vivo scene [2]. For each metric, Δ up-arrow indicates higher is better and down-arrow indicates lower is better

target ratio. This reduces the likeliness of "spikey" Gaussians. Differently, SuGaR [6] and NeuSG [3] introduce minimizing the smallest axis w.r.t to each gaussian point. This encourages flat surface-like appearances, though Surfels [4] highlights the lack of fine-details using this approach and instead proposes setting the minimum scale axis to 0 to improve stability and surface alignment during training. While developing our method we played around with the concept of surface-like scale regularizers. Though it's important to consider that to regularize surfaces the representation needs to have uniform local point normals. This is trivial to accomplish via regularizing nearest neighbors w.r.t the direction of the smallest scale axis - assumed to be the point-surface normal. Still, this adds complexity as it requires tracking and regularizing large matrices and also requires a fast and accurate nearest neighbor algorithm. We also re-iterate that this approach would not work on fire and smoke, meaning that a deeper investigation into quickly and accurately separating dynamic GS textures from other points is also required.

Enhancement using generative models also targets regularization. The obvious way to resolve geometric artifacts is to supervise depth, D, with monocular depth estimations, D_e using the ground truth RGB images. However, directly evaluating $\ell_n(|D_e - D|)$ or $\ell_n(|\hat{D}_e - \hat{D}|)$ for any

Figure 3. Frame-by-frame visual results of benchmarks for the Pony-Vivo scene [2]

degree n is problematic as D is anchored to the local intrinsics while D_e is not. This is shown in Fig. 9 where there is a significant amount of noise in D, generated using 3D-GS, that prevents any linear depth alignment. This means depth-based approaches can be considered a non-trivial problem. Instead, patch-based approaches have been proposed to localize depth errors [7, 12, 17]. DNGaussians [7] proposes soft and hard depth regularization, which combines shape-freezing[3] and global and local depth normal-

[3]Freezing all Gaussian parameters except position or opacity

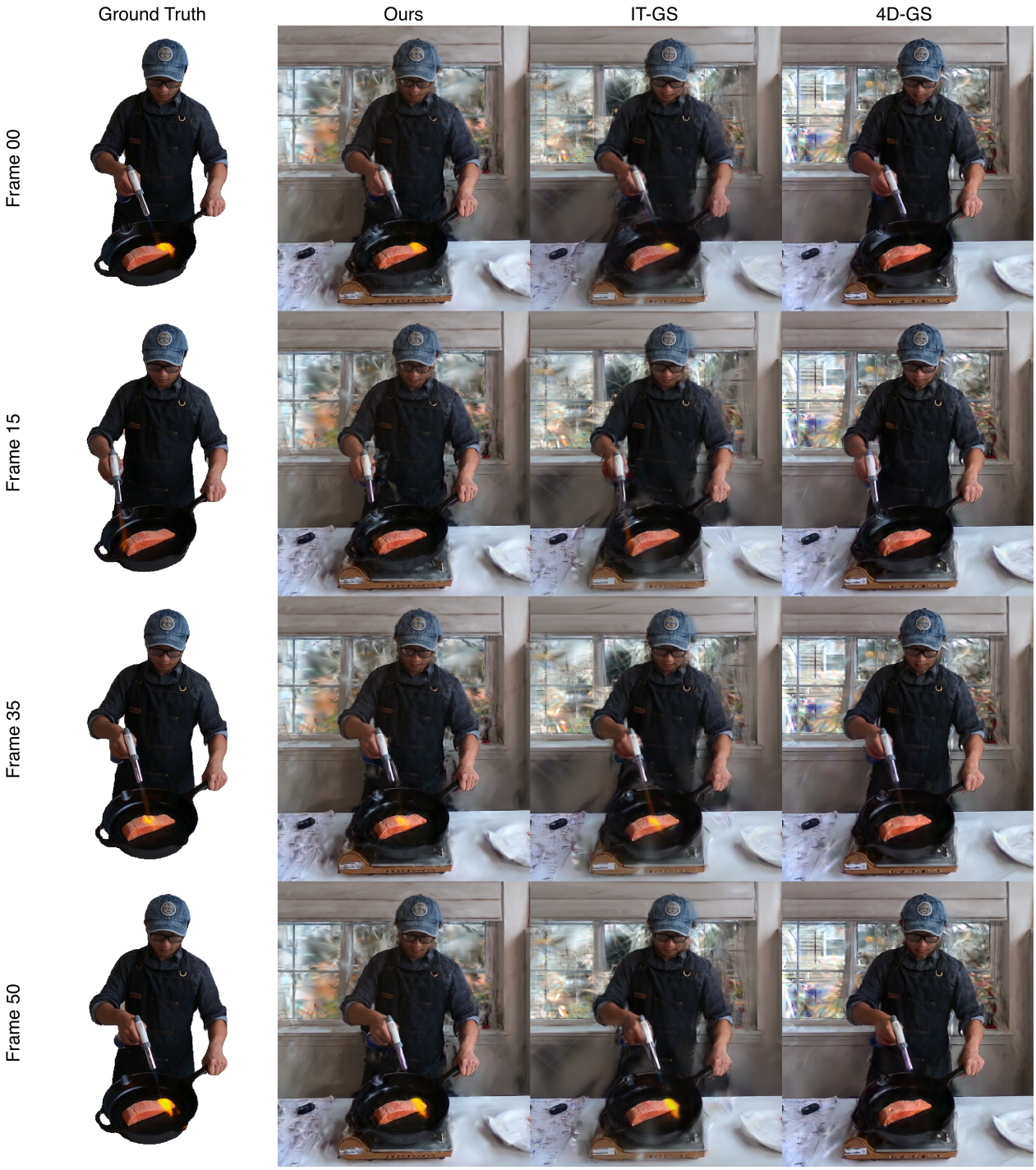

Figure 4. Frame-by-frame visual results of benchmarks for the Flame Salmon DyNeRF scene [8]

ization. SparseGS [17] proposes first rendering depth via a softmax function to maximize large alphas and minimize small alphas and uses this to evaluate the Pearson correlation coefficient $1-PCC(D^i, D^i_E)$ per patch i. CoherentGS [12] extends the use of generative priors with optical flow predictions. Alongside a set of depth atlases, generated with D_e, features are matched between views and encouraged to minimize the distance between the associated Gaussians.

Ultimately the use of generative priors would be beneficial in dynamic GS pipelines, and unlike other static SV3D

Figure 5. Frame-by-frame visual results of our method for the Cook Spinach (top) and Flame Steak (bottom) DyNeRF scene [8]. For the former we render only foreground points and fore the later we render the whole scene

solutions it is less vulnerable to issues with dynamic textures. The reason we did not explore this in our work was primarily due to challenges denoising and aligning the depth priors to the 3D-GS scene space. As DepthAnything also operates on single images, there are consistency issues between each frame that produce different shapes perf frame. We instead tried using VideoDepthAnything, however what it gained in consistency it lost in quality as surface edges returned noisier depth estimates.

5.2. Counting Gaussians

While testing various model designs we found that hyperparameter tuning the number of initial Gaussians as well as the number of instances for cloning led to differences in visual quality. For sparse vire scenes, we found that more Gaussians may improve fine details but it also increase the chances of view dependent foreground Gaussian point positioning, which leads to inconsistent novel views synthesis.

5.3. Real Depth Priors

Real depth priors can be obtained using video-depths sensors and it is not uncommon to find reconstruction datasets with real depth measurements, like with ViVo [2]. For sparse view problems, depth could be used to regularize surface distances, as discussed in the prior subsections. However, for most set-ups the depth video resolution is much lower than the RGB frame resolution, so applying depth losses would require a canonical Gaussian representation that can render alis-free reconstructions. Our method aloows for this by using MipSplatting as our canonical representation, though to our knowledge all other SotA are are based on 3D-GS so are not alias free.

Despite the applicability of our approach, depth-based losses still require investigation as well known issues persist with predicting surface depth using the Mhalanobis function for approximating depth.

5.4. Relationship with Video Duration.

We tested the SotA on varying lengths of video and found differing results. In the main paper we present the best results for each model and test all methods using the same number of frames. 4D-GS, W4D-GS. Conversely, IT-GS and STG performed best on 50 frames. The latter result is expected considering that low rank polynomials are limited by motion complexity. However, the former is less expected. Additionally, our approach performs best on 300 frames for the Flame Salmon scene whereas 50 frames is ideal for the Flame Steak and Cook Spinach scenes.

Feature grids (i.e. hex-planes) preserve local space, so are less prone to catastrophic forgetting than polynomial deformation fields that use MLPs. This is only limited by the space-only planes acting as a canonical dynamic feature

Figure 6. Frame-by-frame visual results of our method on the Fruit and Curling ViVo dataset [2]

grid. Whereby, the space-only planes will only fail to learn a canonical feature space when the range of features (i.e. possible combination of dynamic components over time) exceeds some (unknown) threshold; at which point a new canonical space could be generated for the subsequent video sequence. However, as long as a decent canonical approximation can be found the dynamic features from the space-time planes should preserve space locally in time, meaning the quality of frame N is only weakly tied to the quality of frame 1. This explains why video duration is not inversely proportional to reconstruction quality, as with polynomial deformation fields. Still, it does not answer the question of why they perform better. This is explained by under-training the feature space due to limited dynamic information with shorter videos. So long as a decent canonical space can be learned, longer videos will provide more visual examples of how dynamic GS points should behave, increasing the chances that the correct dynamic motions are learned.

For our method, this becomes more challenging as some parameters are polynomial (i.e. μ, h, ω) and others are deformed with grid features. As the polynomial opacity function is a single-peak function, GS points with large ω (i.e. highly dynamic textures) will only be visible for an instant throughout the entire video. This means the GS model requires a lot more points to reconstruct long lasting dynamic texture events. This is shown comparing the Flame Salmon and Flame Steak scenes, where the motions are similar but the amount of fire in the Flame Steak is more. Thus, for longer scenes where dynamic texture events are temporally dense, longer videos will be more challenging to reconstruct.

References

[1] Adrian Azzarelli, Nantheera Anantrasirichai, and David R Bull. Waveplanes: A compact wavelet representation for dynamic neural radiance fields. *arXiv preprint arXiv:2312.02218*, 2023. 1

[2] Adrian Azzarelli, Ge Gao, Ho Man Kwan, Fan Zhang, Nantheera Anantrasirichai, Ollie Moolan-Feroze, and David Bull. Vivo: A dataset for volumetric videoreconstruction and compression. *arXiv preprint arXiv:2506.00558*, 2025. 4, 5, 6, 8, 9, 10

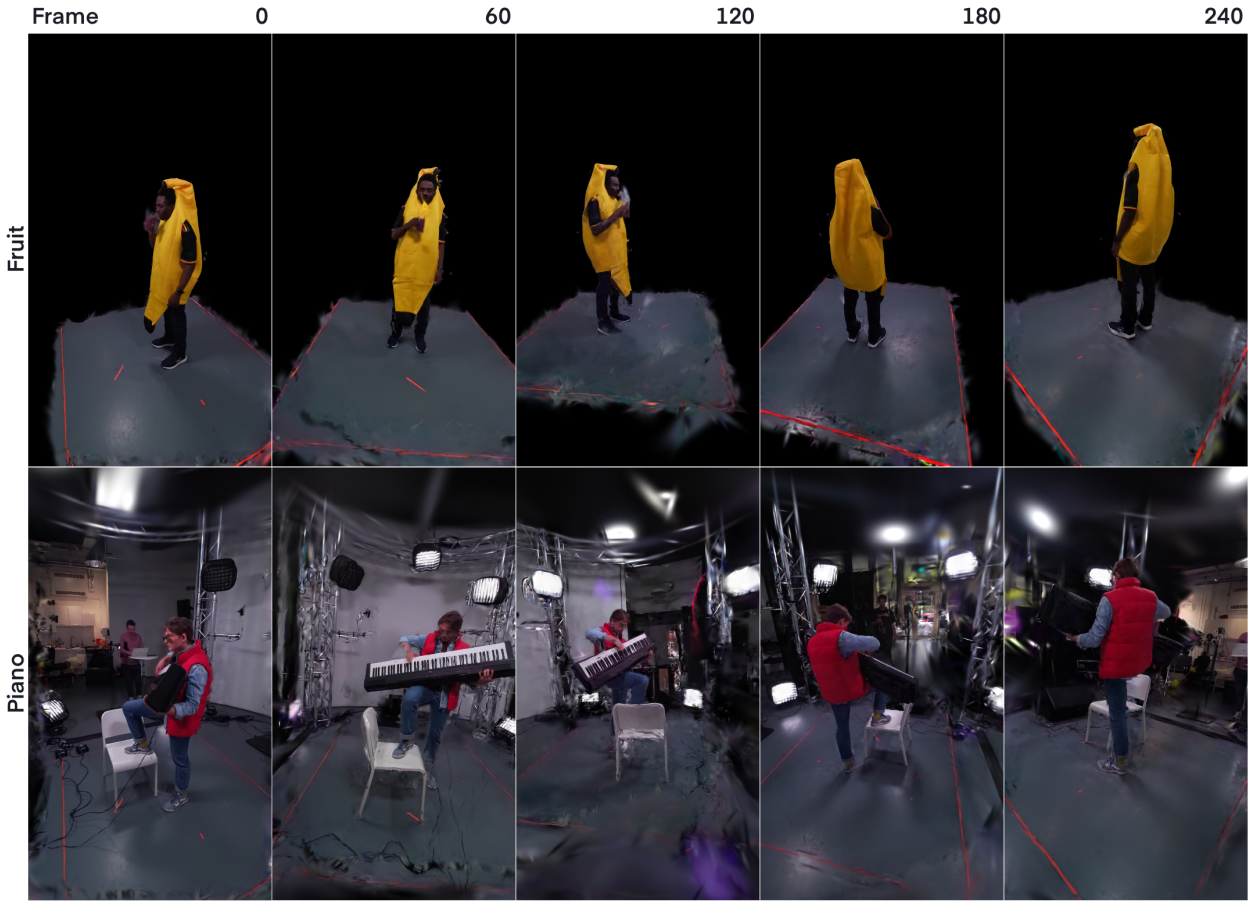

Figure 7. Frame-by-frame novel view synthesis results for the Fruit and Piano Vivo scene [2]

4DGS Ours

Figure 8. Failure case with 4DGS (left) and Ours (right) demonstrating greater problems with reconstructing large and fast motions

[3] Hanlin Chen, Chen Li, and Gim Hee Lee. Neusg: Neural implicit surface reconstruction with 3d gaussian splatting guidance. *arXiv preprint arXiv:2312.00846*, 2023. 1, 5

[4] Pinxuan Dai, Jiamin Xu, Wenxiang Xie, Xinguo Liu, Huamin Wang, and Weiwei Xu. High-quality surface reconstruction using gaussian surfels. In *ACM SIGGRAPH 2024 Conference Papers*, pages 1–11, 2024. 5

[5] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023. 1, 2

[6] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5354–5363, 2024. 5

[7] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *Proceedings of the IEEE/CVF conference on com-*

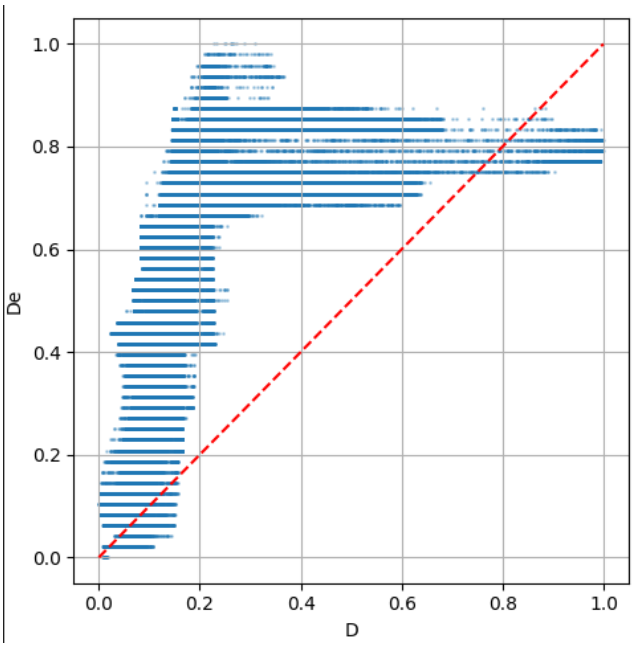

Figure 9. Normalized comparison of depth estimation, D_e from DepthAnything v2 [18] and depth estimation from 3D-GS, D

puter vision and pattern recognition, pages 20775–20785, 2024. 6

[8] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5521–5531, 2022. 4, 7, 8

[9] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8508–8520, 2024. 2

[10] Harry Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 2009. 1

[11] Diego Ongaro and John Ousterhout. In search of an understandable consensus algorithm. In *2014 USENIX annual technical conference (USENIX ATC 14)*, pages 305–319, 2014. 2

[12] Avinash Paliwal, Wei Ye, Jinhui Xiong, Dmytro Kotovenko, Rakesh Ranjan, Vikas Chandra, and Nima Khademi Kalantari. Coherentgs: Sparse novel view synthesis with coherent 3d gaussians. In *European Conference on Computer Vision*, pages 19–37. Springer, 2024. 6, 7

[13] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 2

[14] Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 2006. 1

[15] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20310–20320, 2024. 2

[16] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 1, 3

[17] Haolin Xiong, Sairisheek Muttukuru, Hanyuan Xiao, Rishi Upadhyay, Pradyumna Chari, Yajie Zhao, and Achuta Kadambi. Sparsegs: Sparse view synthesis using 3d gaussian splatting. In *International Conference on 3D Vision 2025*, 2025. 6, 7

[18] 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. 11

[19] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024. 1

[20] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (ToG)*, 43(6):1–13, 2024. 3