SC-GS: Sparse-Controlled Gaussian Splatting for Editable Dynamic Scenes
– *Supplemental Material* –

Outline

In the supplementary file, we provide more implementation details and more results not elaborated in our paper due to the paper length limit:

- Sec. S1: more ablation results including the number of control points, the effect of control points, and the impact of ARAP loss.
- Sec. S2: qualitative results demonstrated in videos.
- Sec. S3: rendering speed analysis.
- Sec. S4: more implementation details.

S1. More Ablation Results

S1.1. Ablation on Initial Number of Control Points

We examine the influence of the initial number of control points on D-NeRF [5] datasets, considering both its effect on the final number of control points and its impact on performance. The results for varying initial control point quantities are reported in Table S1.

Table S1. We conducted an ablation analysis on the initial number of control points on D-NeRF [5] datasets. Evaluation metrics include average PSNR, SSIM, and LPIPS values. The second line shows the average final count of control points after training.

Initial	64	128	256	512	1024	2048	4096
Final	67	132	264	519	1018	1973	2083
PSNR(↑)	42.65	42.87	43.17	43.31	**43.51**	42.63	39.88
SSIM(↑)	.9967	.9968	.9969	**.9971**	.9969	.9961	.9953
LPIPS(↓)	.0073	.0067	**.0061**	.0063	.0064	.0079	.0090

Typically, the final number of control points varies from the initial number and tends to converge to a more appropriate quantity for representing motion through the adaptive densifying and pruning strategy. As demonstrated in the second row of Table S1, when the number of initial control points is too large (4096), exceeding the requirements for representing the motion space, redundant points will be pruned (2083). Moreover, increasing the number of control points does not necessarily lead to better performance due to optimization challenges. In contrast to using an excessive number of control points, the model performance is less sensitive to a smaller number of initial control points, with relatively stable performance observed for both 64 and 128 control points. Consequently, they will not grow significantly. This further validates our sparse motion assumption, which posits that motion can be represented by a small number of control points. Upon evaluating various control point configurations, we discovered that the best PSNR, SSIM, and LPIPS values were achieved with 1024, 512, and 256 control points, respectively. However, the overall performance was best with 512 control points.

S1.2. Ablation on Control Points and ARPR Loss

In the video from 03:32 – 04:19, we show the qualitative results to demonstrate the effects of control points and the ARPR loss for motion reconstruction and dynamic view synthesis.

Effects of control points: The reconstructed motion trajectories of Gaussians obtained from our baseline method and our formulation with control points are showcased in the video clip 03:32 – 04:04, emphasizing the advantage of our method in achieving more precise motion reconstruction. It is noticeable that the reconstruction from the baseline exhibits high-frequency, noisy shaking, whereas our approach generates a more stable motion. This underscores the effectiveness of our compact motion representation with control points in reconstructing dynamic scenes, which promotes a smoother and more accurate trajectory.

Effects of APRP loss: The ARAP loss, formulated in Eq. (10) in the paper, constrains the local rigidity of control point motion. As demonstrated in the video clip 04:04 - 04:19, without ARAP loss, some control points on the arm move towards the torso as the arm falls down, resulting in inconsistency with the actual motion and negatively affecting the subsequent Gaussian rendering to some extent, as shown in Table 3 in the paper. By incorporating ARAP loss on control points, the local rigidity of motion can be ensured which also improves the rendering qualities.

S2. More Qualitative Results with Video Demonstrations

We provide a video in the attached documents to showcase the quality of our results. The **dynamic scene reconstruction** results can be seen in the video clip 00:10 – 01:37, where we compare our method with TiNeuVox [1] and the concurrent work 4D GS [7] on the D-NeRF [5] dataset. We also make comparisons with NeRF-DS [8] and HyperNeRF [4] on the NeRF-DS [8] dataset and showcase additional reconstruction results on the HyperNeRF [4] dataset. Furthermore, the video clip 01:37 – 03:32 features our **motion editing** interface, designed for efficient interactions, and presents more results of reconstructed motion and edited motion, including both D-NeRF [5] data and self-captured data.

S3. Rendering Speed Analysis

Our method predicts transformations for a small number of control points, which in turn drive the motion of 3D Gaussians, leading to faster rendering speeds and a reduced network query load compared to our baseline method. We compare our rendering speed, measured in frames per second (FPS), with the baseline approach, 4D Gaussians [7] (4D-GS), and 3D Gaussians [2] (3D-GS) on the D-NeRF [5] datasets using an NVIDIA RTX 3090 GPU, where the average number of Gaussians was 48.98K. As illustrated in Table S2, our approach outperforms the baseline method, which directly predicts per-Gaussian deformation by 2.6 times. It is worth noting that our sparse-controlled approach has a higher speed compared to 4D-GS [7], which introduces multi-resolution hex-planes to store latent motion features of Gaussians and predicts Gaussian deformation with a small MLP to reduce the network query burden. Our method is approximately twice as fast as 4D-GS. Additionally, our method's rendering speed is comparable to that of the original 3D Gaussians [2]. Our approach adapts 3D Gaussians to dynamic scenes while maintaining its rendering quality and speed.

Table S2. Comparison of rendering speed on D-NeRF [5] datasets measured in FPS, at a resolution of 800×800 resolution.

Method	3D-GS [2]	4D-GS [7]	Baseline	Ours
Speed (FPS) ↑	298.5	145.8	113.3	295.4

S4. Implementation Details

In this section, we introduce more details on model implementation and training.

Network To predict the 6 DoF transformations of control points, we utilize an MLP Ψ (described in Section 4.1 of the main paper) comprising 8 layers and a skip connection at the 4th layer. The network takes as input the positional embedding [3] of the control points' time and canonical position, with respective frequencies of 6 and 10.

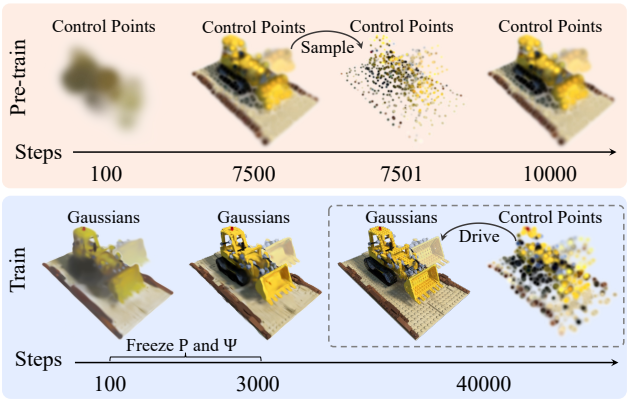

Figure S1. Our approach involves pre-training the sparse control points and MLP to capture the rough motion of the scene, before jointly training them with the Gaussians to refine the scene dynamics and capture finer details.

Pre-training We show the pre-training process in the first row of Fig. S1. To capture the scene's rough motion, we pre-train the control points $\mathcal{P} = \{(p_i \in \mathbb{R}^3, o_i \in \mathbb{R}^+)\}, i \in \{1, 2, \cdots, N_p\}$ and the MLP Ψ. In this stage, we initially sample N_p control points from the SfM point clouds or randomly if unavailable. Subsequently, we render these control points following Gaussian Splatting [2], treating them as trainable Gaussians with time-varying transformations predicted by the MLP Ψ. To ensure uniform distribution and prevent distortion, the Gaussians of all control points are regularized with the same size o_i, i.e., the sphere shape with the same radius, which is further employed as the sphere-shaped Gaussian kernel in the Radial Basis Function (RBF) for calculating neighboring weights, as demonstrated in Eq. (5) of the main paper.

The control points and MLP weights are pre-trained with both the $\mathcal{L}_{\text{render}}$ loss of control point Gaussians and $\mathcal{L}_{\text{arap}}$. The former encompasses $\mathcal{L}_1$ loss and D-SSIM loss, while the latter is defined in Eq. (10) of the main paper. During optimization, we allow the control points to be densified or pruned for 7500 iterations. Afterward, we sample N_p control points using farthest point sampling [6] and train the Gaussians of control points and MLP for another 2500 iterations without further densification or pruning to maintain a fixed number of control points. For scenes with relatively small motions (e.g., D-NeRF [5] and NeRF-DS [8] datasets), we pre-train control points and the MLP on all frames. For real-captured scenes where moving objects have entirely different positions at different times, such as the hand in the teaser image, we progressively pre-train the control points and MLP, beginning with the first 20% of

frames and adding the subsequent 20% to the training set every 1000 steps.

Training With the pre-trained control points $\mathcal{P}$ and MLP Ψ, our objective is to obtain well-trained Gaussians $\mathcal{G}$ that, in conjunction with the former two, can accurately capture the dynamic scene. Initially, we freeze the parameters of the control points and MLP, focusing on optimizing the Gaussians for the first 3000 iterations. This is because the Gaussians are relatively new and unstable at the beginning. Subsequently, we jointly train the Gaussians $\mathcal{G}$, control points $\mathcal{P}$, and MLP Ψ with the loss and adaptive learning strategy defined in Section 4.3 of the main paper. The training process is illustrated in the second row of Fig. S1.

During the training process, the weight of the ARAP loss, denoted as λ_{arap}, undergoes exponential decay from 1e-4 to 1e-7 over 10000 iterations. After 10000 iterations, the ARAP loss is no longer applied, as the rigidity of the control points has been effectively constrained. The following training process is primarily dominated by the rendering loss.

Analysis of training time While our training strategy includes a pre-training stage, the overall training speed is still fast, as the pre-training stage typically takes around 3-4 minutes. The main training process takes approximately 25 minutes. Note that the KNN interpolation process of Gaussian motion during train time differs from that during inference time: the index and weight used in the KNN search are not fixed and should be updated at each training iteration, which slightly increases the training time. Nevertheless, our total training time (approximately 28 minutes) remains comparable to the concurrent work 4D-GS [7] (approximately 20 minutes), and it does not increase the computation too much compared to the original 3D Gaussian (approximately 10 minutes) for static scenes.

References

[1] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2

[2] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4): 1–14, 2023. 2

[3] 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. 2

[4] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. ACM, 2021. 2

[5] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021. 1, 2

[6] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 2

[7] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 2, 3

[8] Zhiwen Yan, Chen Li, and Gim Hee Lee. Nerf-ds: Neural radiance fields for dynamic specular objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8285–8295, 2023. 2