S3R-GS: Streamlining the Pipeline for Large-Scale Street Scene Reconstruction

Supplementary Material

In this appendix, we provide additional ablation studies, more comparative experiments, and further details of our S3R-GS. Additionally, we provide more quantitative results on nuScenes [5].

6. Additional Experiment Results

Effectiveness of Each Module on Reconstruction Speed. To evaluate the effectiveness of each module in our method, we compare the training speed and reconstruction quality. Here, we study how each design in our S3R-GS influences the reconstruction speed. Tab. 7 and Fig. 5 detail the performance and reconstruction speed across different ablated runs of S3R-GS on the KITTI 0009 scene. In particular, by incorporating the instance-specific projection, the reconstruction time decreases by 11.57 hours due to the mitigation of unnecessary transformations. As shown in Fig. 6, the conventional reconstruction pipeline adopts a local-to-global transformation strategy. During the forward pass, all 3D Gaussian primitives must be transformed from their local coordinate systems into a shared global space. This operation incurs a time complexity of $O(4 \times 4 \times M)$, where M denotes the total number of 3D Gaussians. After transformation, these Gaussians are projected onto the image plane via a parallel CUDA C++ rasterization module. In the backward pass, the loss gradients are first propagated to the global coordinates of all 3D Gaussians, which is relatively efficient due to explicit CUDA C++ implementation. However, the gradients must then be back-propagated from the global space to each individual Gaussian in its ego local coordinate frame by the autograd, again with a time complexity of $O(4 \times 4 \times M)$. This step introduces significant computational overhead, particularly as the number of Gaussians increases. In contrast, our instance-specific projection approach avoids the global transformation bottleneck. During the forward pass, we only compute the instance-specific extrinsic transformation from each object's local space to the camera space. This reduces the complexity to $O(4 \times 4 \times 4 \times k)$, where k is the number of object instances and $k \ll M$. More importantly, the primary source of computational redundancy in the conventional pipeline stems from the autograd operations during the backward pass. Our method circumvents this by explicitly implementing the backward projection within the CUDA C++ rasterizer, significantly reducing the training cost. As a result, incorporating instance-specific projection significantly reduces the reconstruction time. Next, we assign a temporal visibility attribute to each Gaussian to reduce excessive 3D Gaussian projection; the reconstruction time further reduces by 0.19 hours. Since the KITTI dataset primarily consists of single-view reconstructions, the viewpoint overlap is relatively high. As a result, the improvement of incorporating temporal visibility in single-view scenes is relatively less. Despite this, excluding temporal visibility can easily lead to memory overflow during training on large-scale scenes, underscoring its necessity. Lastly, incorporating the adaptive LOD strategy reduces reconstruction times by 0.91 hours.

Instance-specific proj.	Temporal visibility	Adaptive LOD	Rec. Time ↓↑
✗	✗	✗	15.69h
✓	✗	✗	4.12h
✓	✓	✗	3.93h
✓	✓	✓	3.02h

Table 7. **Ablation Study on the Modules of the Reconstruction Pipeline.** Each component of our streamlined pipeline contributes to its overall effectiveness.

Quantitative Evaluation under Ego-Vehicle Lane Change. We conduct a focused comparison among our proposed S3R-GS, NeuRAD [42], and StreetGaussian [53], following the evaluation metric of NeuRAD. As shown in Table 8, our method consistently outperforms both baselines across various ego-vehicle pose movements, including lateral lane shifts and vertical viewpoint deviations.

Visualization of the Influence of BEV-Semantic Initialization Augmentation. As illustrated in Fig. 7, after supplementing the initial points of buildings using the BEV-semantic initialization augmentation (left), the high-rise de-

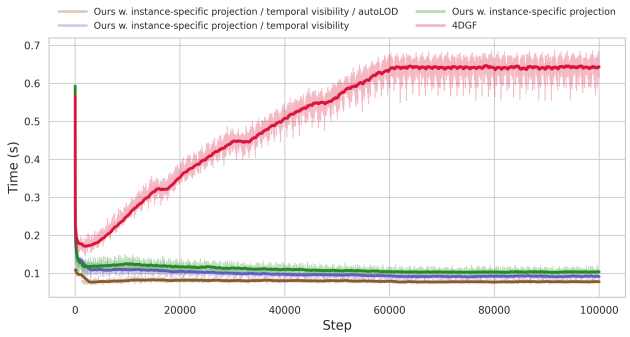


Figure 5. **Comparison of Reconstruction Times per iteration**. In large-scale scene reconstruction, our method maintains a stable per-iteration training time by incorporating instance-specific projection, temporal visibility, and adaptive LOD, preventing overhead from increasing as the number of Gaussians grows.

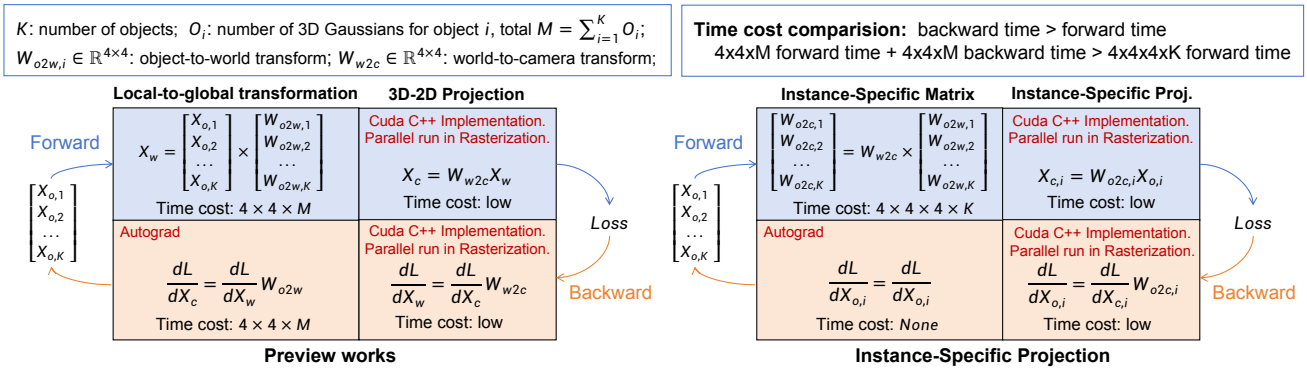


Figure 6. **Time Complexity Comparison** between the local-to-global transformation used in conventional reconstruction pipelines and our instance-specific projection approach.

FID ↓	KITTI [75%]				KITTI 0009 [75%]			
	Lane 0m	Lane 2m	Lane 3m	Vert. 1m	Lane 0m	Lane 2m	Lane 3m	Vert. 1m
StreetGS	79.54	246.17	285.39	108.53	80.62	445.75	424.49	92.02
NeuRAD	34.14	161.09	193.21	196.56	95.16	324.87	339.08	147.91
S3R-GS	**9.07**	**70.42**	**96.78**	**106.97**	**45.42**	**46.64**	**49.09**	**58.57**

Table 8. **FID Scores when Shifting Pose of Ego Vehicle.** Supplementing points in tall structure regions contributes to improved reconstruction quality.

tails of the buildings can be reconstructed more quickly (middle). In contrast, the setting without augmentation results in missing details in the upper parts of the buildings (right). As a result, the application of the BEV-semantic initialization augmentation method to supplement points in tall structure regions enhances the overall reconstruction quality. We also qualitatively compare the BEV-semantic initialization augmentation method (BEV-aug) and SfM initialization augmentation method (SfM-aug). As shown in Fig. 8, the BEV-aug initialization is capable of accurately initializing tall structures—such as buildings—that are often sparsely represented in LiDAR point clouds. In contrast, the SfM-aug initialization tends to produce inaccurate results in these regions, primarily due to misalignment and geometric inconsistencies between the point clouds estimated by Structure-from-Motion (SfM) and those acquired from LiDAR sensors. This misregistration leads to significant errors in the initialization of scenes.

Qualitative Comparison with Different Decomposition Methods. Fig. 9 shows the qualitative reconstruction results with different decomposition methods. Compared to 3D bounding boxes decomposition (bottom), our 2D decomposed approach effectively models the appearance of vehicles and accurately learns their positions (middle), achieving competitive results to precise 3D bounding boxes.

Algorithm 1 BEV-Semantic Initialization Augmentation

Input: original_points P, cam_intrs, cam_extrs, image_paths, image_width w, image_height h
aug_points_list ←[ ]
aug_labels ← $['building','house','tree']$
masks ← SegmentModel(image_paths, aug_labels)
for each camera (i, intrinsics K, extrinsics W) pair **do**
  points_uv, visble_mask←ProjectToImage(P,K,W)
  mask ← masks[i]
  idxs ← mask[points_uv]
  bev_coords ← P[visible_mask][idxs, :2]
  grid_size ← 0.5
  bev_quats ← floor(bev_coords / grid_size)
  z_coords ← P[visible_mask][idxs, 2]
  grid_dict ← {lambda:'max_z'":,'avg_xy':,'count':}
  grid_dict.record(bev_quats, z_coords)
  z_interval ← 0.2
  max_h ← 40
  dense_thresh ← 5
  supple_points SP ← [ ]
  for each valid grid_cell in grid_dict **do**
    if grid_cell['count'] > dense_thresh **then**
      max_z ← grid_cell['max_z']
      z_vals ← Range(max_z, max_h, z_interval)
      x, y ← grid_cell['avg_xy']
      Add (x,y,z_vals) to supple_points SP
    end if
  end for
  SP_uv, SP_vis_mask←ProjectToImage(SP,K,W)
  Add SP[SP_vis_mask] to aug_points
end for
Return the augmentation points aug_points

7. The Details of S3R-GS

In this section, we provide a detailed description of each part of our method.

7.1. BEV-Semantic Initialization Augmentation

For the LiDAR points captured at each time step t, we apply the BEV-semantic initialization augmentation, as described in Algorithm 1, to supplement the point cloud. After performing this augmentation across all frames, we employ a voxel grid downsampling method to remove redundant points.

7.2. Neural field architectures and implementations

NeuralODE model. To maximize efficiency, we implement our NeuralODE model with simple multilayer perceptrons (MLPs) as detailed in Tab. 9. The NeuralODE models are employed to learn the velocity of objects. To enable the network to accommodate objects with varying speeds and trajectories, we introduce an instance embedding, where the embedding size corresponds to the number of objects. The input to our model consists of the timestamp t and the instance embedding. Initially, the timestamp t is encoded using sinusoidal encoding into a 4-dimensional representation. This encoded temporal representation is then concatenated with the instance embedding and passed through an MLP. Specifically, we utilize a 4-layer MLP with a hidden dimension of 64. Furthermore, we leverage the **torchdiffeq.odeint** library, which provides an efficient and computationally optimized solver for ordinary differential equations (ODEs). This enables us to predict an object's position at a given timestamp based on its initial position and the learned NeuralODE model, facilitating accurate trajectory estimation.

	Time encode		Instance Embedding		MLP	
Method	Input dim	Output dim	Embbed num	Output dim	Layers	Hidden dim
Sinusoidal	1	4	object nums	16	4	64

Table 9. **NeuralODE Model Architecture.** We provide the detailed parameter configurations of the NeuralODE Model which is used to model object poses.

Neural fields. To predict the color of 3D Gaussians, we take the view direction, 3D position, depth and a time-dependent embedding as the input to neural fields. By introducing the depth, the neural fields can learn the color of Gaussians across different LODs. Following 4DGF [11], for different types of inputs, we adopt different encoding methods: We encode view directions using Spherical Harmonics of degree 4, enabling the model to capture directional lighting effects efficiently; We employ a HashGrid encoding to represent spatial positions, which helps mitigate excessive sparsity in the feature space; In contrast to 4DGF, we further introduce depth as an additional input to the neural fields, and encode it using a Frequency encoding with four frequency bands. For different types of Gaussians, we adopt distinct neural architectures. We use a color MLP and an opacity MLP to model the appearance of static Gaussians; We employ a color MLP to predict the color of dynamic Gaussians of rigid objects; We introduce both a color MLP and a deformable MLP to capture the dynamic appearance of dynamic Gaussians of non-rigid objects. We adopt from 4DGF to handle transient objects and varying illumination conditions. The detailed parameter configurations are shown in Tab. 10.

7.3. Pipeline Optimization.

Following [11], to optimize the scene reconstruction, we utilize the following loss function to optimize each training iteration:

$$\mathrm{L} = \lambda_{color}\mathcal{L}_{color}(I_r, I_{gt}) + \lambda_{ssim}\mathcal{L}_{SSIM}(I_r, I_{gt}) + \lambda_{dep}\mathcal{L}_{dep}(D_r, D_{gt}), \tag{8}$$

where I_r denotes the rendered image, I_{gt} denotes the ground-truth image, D_r denotes the rendered depth of the scene, and D_{gt} denotes the ground-truth depth map of image, which is obtained by projecting the LiDAR point captured by sensor at the given training timestep onto the image. $\mathcal{L}_{color}$ denotes the L1 norm, $\mathcal{L}_{ssim}$ denotes the structural index measure [45], and $\mathcal{L}_{dep}$ denotes the L2 norm.

7.4. Implementation details

During training, we set $\mathcal{L}_{color}$ as 0.8, $\mathcal{L}_{ssim}$ as 0.2, $\mathcal{L}_{dep}$ as 0.05. We follow [11] and train the scenes using Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$).In the adaptive Level of Detail (LOD) strategy, the LOD threshold r is typically set to 6 pixels. A larger r value accelerates the reconstruction process but may also degrade performance. The maximum culling probability p_{max} is generally set to 0.9, while the maximum distance D is usually defined based on the initial scale of the scene prior to training. The offset values $[\Delta x, \Delta y, \Delta z]$ are commonly set to $[0.5, 0.5, 0.2]$. In our experiments on Argoverse 2 [48] datasets, we train our model on 8 V100 32GB GPUs for 125,000 steps. In our experiments on large-scale scenes from KITTI [17] datasets, we train our model on 1 V100 32GB GPUs for 100,000 steps. In our experiments on the novel synthesis benchmark of KITTI datasets, we train our model on 1 V100 32GPUs for 30,000 steps. In our experiments on nuScenes [5] datasets, we train our model on 1 V100 32GB GPUs for 100,000 steps.

8. Additional Comparison Results

Comparison on Reconstruction Speed. To further evaluate the reconstruction speed and quality of our method compared to the baseline, we conducted experiments on challenging scenes selected from nuScenes datasets.

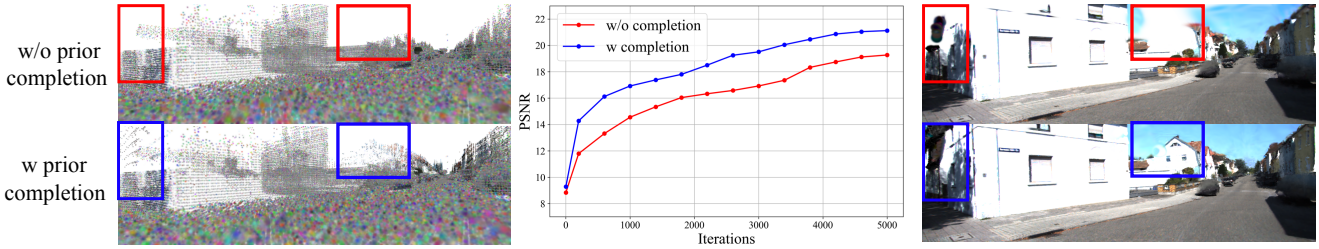


Figure 7. **Influence of Initial 3D Gaussian Prior Distribution.** Incorporating the BEV-semantic initialization augmentation to complete the initial 3D Gaussians is crucial for improving reconstruction quality.

Figure 8. **Visualization Results of BEV-aug Initialization and SfM-aug Initialization.** The BEV-aug initialization approach provides reliable initialization for tall structures that are inadequately captured by LiDAR sensors. In contrast, the SfM-aug initialization method often suffers from inaccurate initialization due to misalignment between the SfM-estimated point clouds and the LiDAR data.

Specifically, we chose two difficult scenes from the nuScenes dataset: scene 1 and scene 63. We used all frames and six camera views from each selected scene. We compared our method with StreetGaussian [53]. As shown in Fig. 10, our method demonstrates significant advantages in both reconstruction speed and quality. On the two nuScenes scenes, our method achieved better reconstruction quality while using only 25% of the time. This demonstrates that our method not only achieves faster reconstruction but also maintains high-quality results, particularly in large-scale and complex scenes.

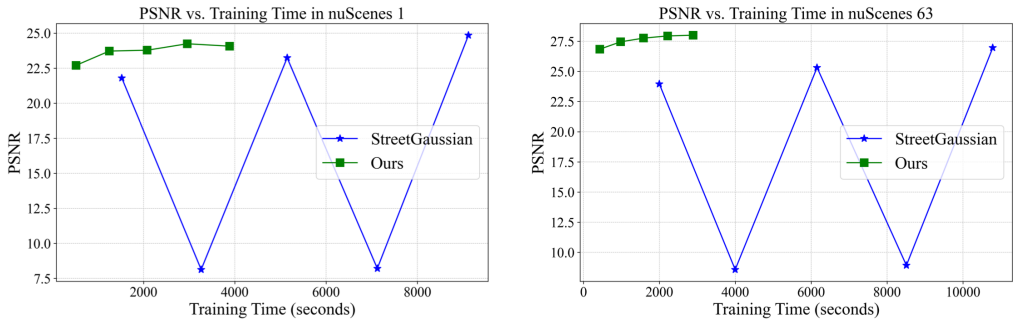
Additional Qualitative Comparison. We also provide additional qualitative comparison results. As shown in Fig. 11, the qualitative results on the nuScenes dataset highlight the remarkable reconstruction speed and quality of our method. For 10,000 rendering iterations, our method required only 428 seconds, compared to 1,999 seconds for StreetGaussian [53]. Furthermore, our method achieved a PSNR of 26.84 in 428 seconds, which is competitive with the PSNR of 26.98 achieved by StreetGaussian [53] after 10,485 seconds. When comparing reconstruction quality for the same number of rendering iterations, our method not only takes significantly less time but also delivers much better reconstruction quality.

Fig. 12 further illustrates the reconstruction results of our method compared to StreetGaussian [53]. Our approach achieved superior performance in modeling objects, particularly for fast-moving objects. In addition, our method demonstrated greater robustness. For example, as shown in the bottom row of Fig. 12, StreetGaussian [53] relies heavily on accurate 3D bounding boxes. If these 3D bounding boxes fail to accurately detect vehicles, StreetGaussian [53] will often fail to reconstruct. However, due to the challenging nature of the nuScenes scenes, our method occasionally showed artifacts during the reconstruction process.

Figure 9. **Qualitative Comparison with Different Decomposition Methods.** Decomposing the object and static elements with 2D boxes and modeling object motion trajectories with NeuralODE produce competitive results to precise 3D bounding boxes.

Model	Direction encode		Depth encode		Position encode				Color MLP		Opacity MLP		Deform. MLP	
	method	degree	method	n_freqs.	method	size	#levels	max. res.	#layers	hidden size	#layers	hidden size	#layers	hidden size
$NeurF_{static}$	SHs	4	Frequency	4	HashGrid	2^{19}	16	2048	3	64	2	64	-	-
$NeurF_{dynamic,rigid}$	SHs	4	Frequency	4	HashGrid	2^{17}	8	1024	2	64	-	-	-	-
$NeurF_{dynamic,non-rigid}$	SHs	4	Frequency	4	HashGrid	2^{17}	8	1024	2	64	-	-	2	64

Table 10. **Neural Field Architectures.** We provide the detailed parameter configurations of two neural fields for static Gaussians and dynamic Gaussians, which are used to model background and object appearance.



(a) Comparison on the scene nuScenes 01 (b) Comparison on the scene nuScenes 63

Figure 10. **Reconstruction speed comparison.** We evaluated reconstruction speed and quality on scenes from nuScenes datasets, where our method showed significant advantages. Specifically, our approach achieved higher reconstruction quality while requiring less reconstruction time. Notably, in the figure, the PSNR of StreetGaussian [53] drops below 10 during certain iterations due to the reset of Gaussian opacity.

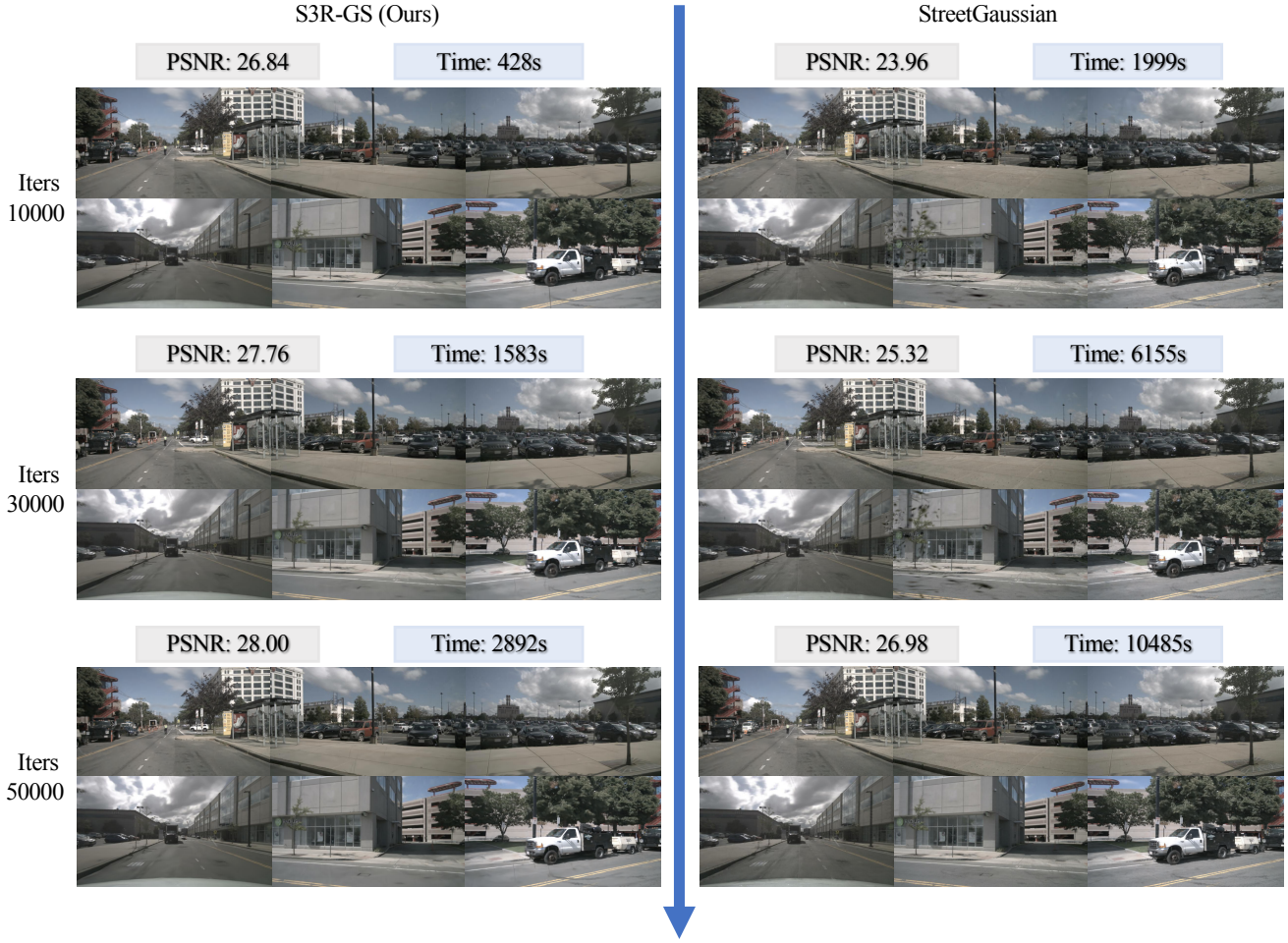


Figure 11. **Qualitative Comparison on the nuScenes Dataset.** We compare the reconstruction quality and reconstruction time of our proposed method against those of the primary competitor, StreetGaussian [53], on various scenes from the nuScenes dataset.

Figure 12. **Qualitative Comparison on the nuScenes Dataset.** We present a qualitative comparison of our method against the primary competitor, StreetGaussian [53], on the scenes from the nuScenes dataset.