EMD: Explicit Motion Modeling for High-Quality Street Gaussian Splatting

Supplementary Material

A. Overview

The supplementary material includes the subsequent components. For additional visual comparisons, we highly encourage you to visit our anonymous website at https://emdgaussian.github.io/, where we showcase more side-by-side visualization results.

- Implementation Details
  - Training Schemes
  - Training Details
- Parameter Sensitivity
- Additional Visualization on Website

B. Implementation Details

B.1. Training Schemes

LiDAR Prior Initialization. To initialize the positions of the 3D Gaussians, we leverage the LiDAR point cloud captured by the vehicle instead of using the original SFM [35] point cloud to provide a better geometric structure. To reduce model size, we also downsample the entire point cloud by voxelizing it and filtering out points outside the image. For colors, we initialize them randomly.

Optimization Objective. Following Street Gaussian, we introduce the sky supervision loss L_{sky} into the original loss function proposed by S^3Gaussian. Subsequently, we get a composed training loss function which can impose various constraints to our model.

$$\mathcal{L} = \mathcal{L}_{color} + \lambda_{depth}\mathcal{L}_{depth} + \lambda_{feat}\mathcal{L}_{feat} + \lambda_{tv}\mathcal{L}_{tv} + \lambda_{sky}\mathcal{L}_{sky} + \lambda_{reg}\mathcal{L}_{reg} \quad (15)$$

Here, $\mathcal{L}_{depth}$ is the mean square error (MSE) loss between the rendered depth map and the estimated depth map from the LiDAR point cloud, which aids in supervising the expected position of 3D Gaussians. $\mathcal{L}_{feat}$ is also the L2 loss of semantic features to reduce the gap between both planes. $\mathcal{L}_{tv}$ is a total-variational loss based on grids to make rendered objects smoother. $\mathcal{L}_{color}$ is the main loss to give constraints to the reconstruction process formulated by:

$$\mathcal{L}_{color} = \mathcal{L}_{rgb} + \lambda_{ssim}\mathcal{L}_{sim} \quad (16)$$

Furthermore, $\mathcal{L}_{reg}$ is organized as:

$$\mathcal{L}_{color} = \mathcal{L}_{rgb} + \lambda_{depth}\mathcal{L}_{depth} \quad (17)$$

Table 1. Loss function coefficients

λ_{depth}	λ_{feat}	λ_{feat}	λ_{tv}	λ_{sky}	λ_{reg}
0.5	0.1	0.1	0.1	0.1	0.01

Table 2. Parameter sensitivity analysis on the D32 dataset, highlighting the effect of varying the dimensions of Gaussian embeddings $\mathbf{z}_k$ and temporal embeddings $\mathbf{z}_w$. All experiments are conducted in the self-supervised setting. Best performances are highlighted in **bold**. ↑ indicates higher is better, while ↓ indicates lower is better. We also include the changes in model parameters relative to the adopted setting.

$\mathbf{z}_k/\mathbf{z}_w$	Parameters	Full Image			Vehicle
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑
32/4	/	**32.50**	**0.933**	**0.082**	29.04
128/4	+14400	32.22	0.925	0.086	**29.05**
8/4	-3600	31.25	0.910	0.128	27.75
32/4	/	**32.50**	**0.933**	0.082	**29.04**
32/16	+14.42M	32.38	0.930	**0.081**	29.01
32/1	-3.60M	30.55	0.898	0.136	27.04

B.2. Training Details

For S^3Gaussian, we train the entire pipeline for 50,000 iterations using the Adam optimizer. Following the original S^3Gaussian setup, we perform a warm-up phase for each scene, employing 5,000 iterations to train a coarse representation using vanilla 3D Gaussians. After this warm-up phase, we integrate the proposed dual-scale deformation network, which is jointly optimized with the HexPlane. To implement a coarse-to-fine training strategy, temporal embeddings $N(i)$ are progressively increased from N_{min} to N_{max} in 20,000 iterations, allowing for the gradual motion modeling of objects. Since S^3Gaussian is evaluated on 50 frames per clip for each scene, we ensure a fair comparison by conducting all self-supervised validation experiments on the first clip of 32 dynamic scenes. Other configurations, including the detailed setup of the HexPlane and learning rates, are kept consistent with the S^3Gaussian. For StreetGaussian, the entire method is trained for 30,000 iterations on a subset of eight selected scenes from the StreetGaussian dataset. Unlike the self-supervised method, we bind the proposed EMD to the vehicle Gaussians in each scene. Temporal embeddings are applied based on the time each vehicle appears within the scene. All other settings, including detailed configurations, remain consistent with those described in StreetGaussian. All experiments are conducted on a single NVIDIA A800 GPU.

C. Parameter Sensitivity

To analyze the sensitivity of model performance to the dimensions of Gaussian embeddings $\mathbf{z}_k$ and temporal embeddings $\mathbf{z}_w$ (derived from the learnable embedding matrix $\mathbf{W}$), we conduct experiments by varying these dimensions. In the original setup, $\mathbf{z}_k$ is set to 32 and $\mathbf{z}_w$ to 4. Tab. 2 summarizes the results, demonstrating how these changes influence performance under the self-supervised setting.

The results reveal that reducing the embedding dimensions leads to significant performance degradation. This is primarily due to the reduced capacity to effectively model the motion of dynamic objects, which is crucial for high-quality reconstruction. On the other hand, increasing the embedding dimensions offers only marginal performance improvements. However, due to the large number of Gaussians in the driving scenes, the higher embedding dimensions result in a substantial increase in the total number of model parameters, leading to a higher computational cost. These findings highlight the trade-off between embedding dimension size and overall model efficiency. While lower dimensions compromise the ability to capture dynamic motion, higher dimensions introduce considerable overhead without proportional gains in performance. The adopted embedding configuration achieves a good balance, maintaining strong performance while keeping the parameter count manageable.