DeSiRe-GS: 4D Street Gaussians for Static-Dynamic Decomposition and Surface Reconstruction for Urban Driving Scenes

Supplementary Material

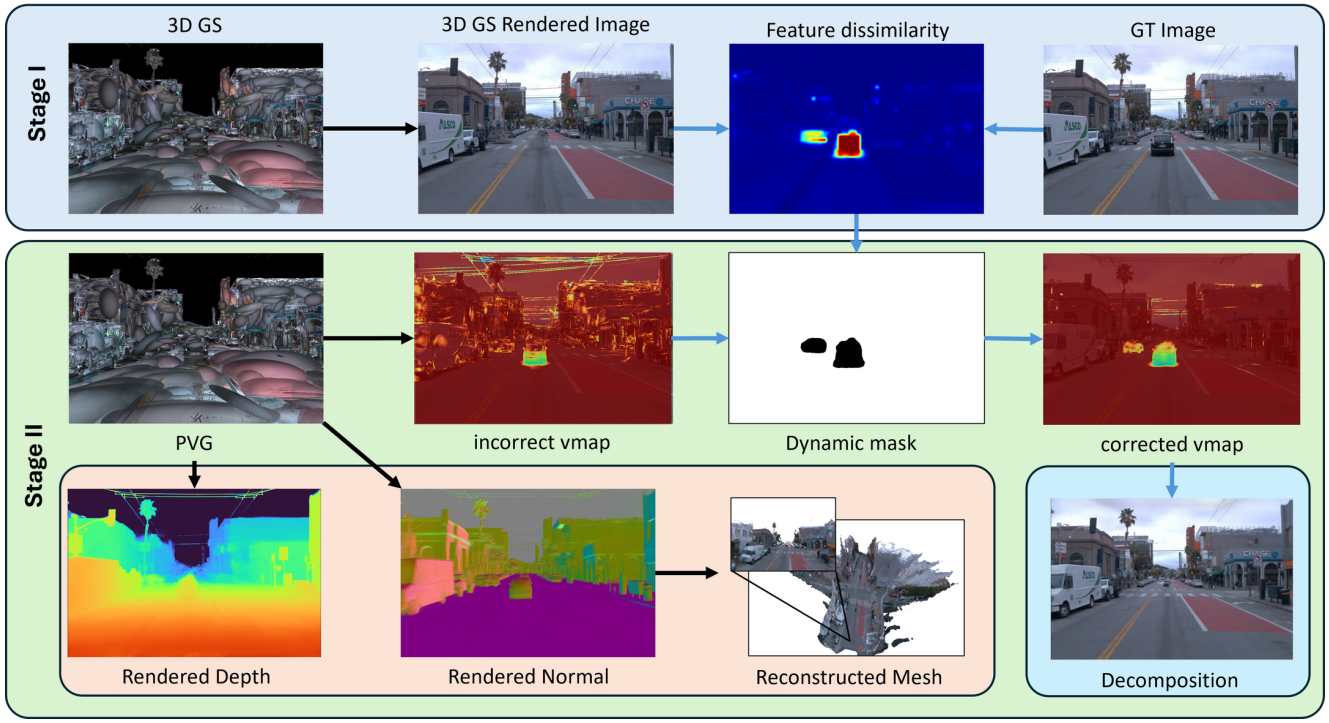

Figure 7. **DeSiRe-GS**. We present a 4D street gaussian splatting representation for self-supervised static-dynamic decomposition and high-fidelity surface reconstruction without the requirement for extra 3D annotations such as bounding boxes.

7. Implementation Details

All experiments are conducted on NVIDIA RTX A6000. We sample a total of 1×10^6 points for initialization, with 6×10^5 from LiDAR point cloud, 2×10^5 near points and 2×10^5 far points depending on their distance to LiDAR origin. In the first stage, we train for a total of 30,000 iterations. We do not train the uncertainty model during the initial 6,000 iterations. After that, the uncertainty model gradually increases its weight over a 1,800-iteration warm-up process. In the second stage, we train a total of 50000 iterations. Cross-view consistency regularization begins after 20,000 iterations, with 102400 pixels sampled each time. Motion masks, obtained from the trained dynamic model, are employed after 30000 iterations to supervise the training of velocity $\mathbf{v}$ and time scale β. We use Adam [17] as our optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ and maintain similar optimization settings of [5]. For the dynamic model we employ a learning rate of 0.001 and a dropout rate of 0.1.

Evaluation Metrics. We adopt Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) [34] and Learned Perceptual Image Patch Similarity (LPIPS) [45] as metrics for the assessment of both image reconstruction and novel view synthesis. Following [15, 38, 39], we also include DPSNR and DSSIM to assess the rendering quality of dynamic objects. Specifically, these values are calculated by projecting the 3D bounding box of moving objects onto the camera plane and computing the PSNR and SSIM within the projected box. Additionally, we introduce depth L1, which measures the L1 error between the rendered depth map and the ground truth depth map, as an evaluation metric which is related to surface reconstruction.

Feature Extractor. DINOv2 [23] is widely used as a foundation model for feature extraction and have demonstrated potential in previous novel view synthesis works, such as Wild-Gaussians [18]. However, we found through experiments that the features extracted from DINOv2 are usually noisy, especially on the road and in the sky, as shown in Fig. 8. The DINOv2 features cannot produce accurate motion masks sometimes. On the other hand, FiT3D [44] fine-tunes

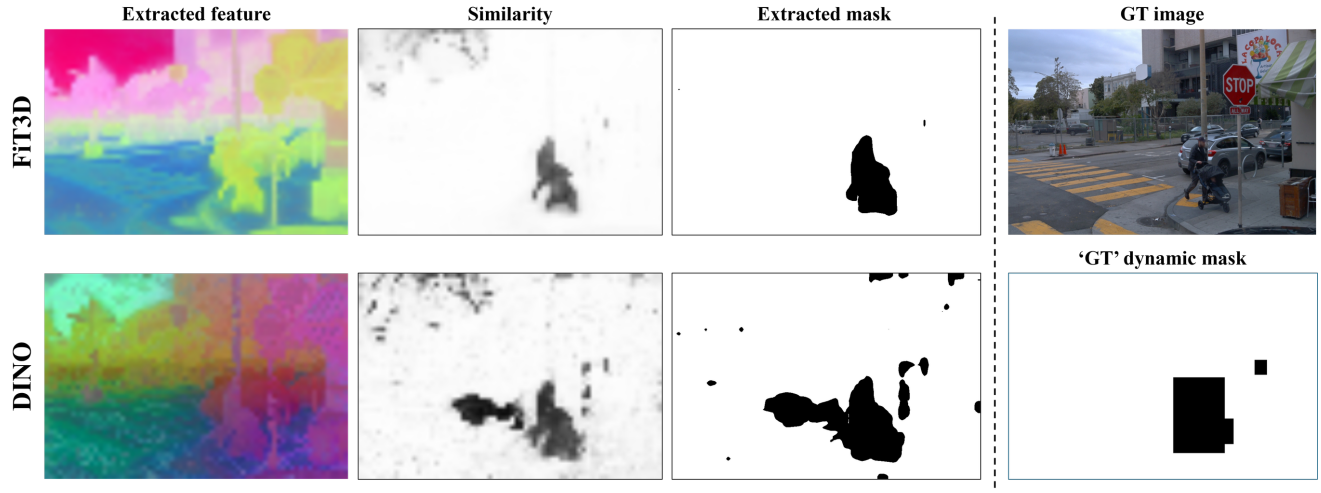

Figure 8. Comparison between motion masks and features extracted from DINOv2 [23] and FiT3D [44]. The 'GT' masks are obtained using the GT bounding boxes and associated trajectories following EmerNeRF [39].

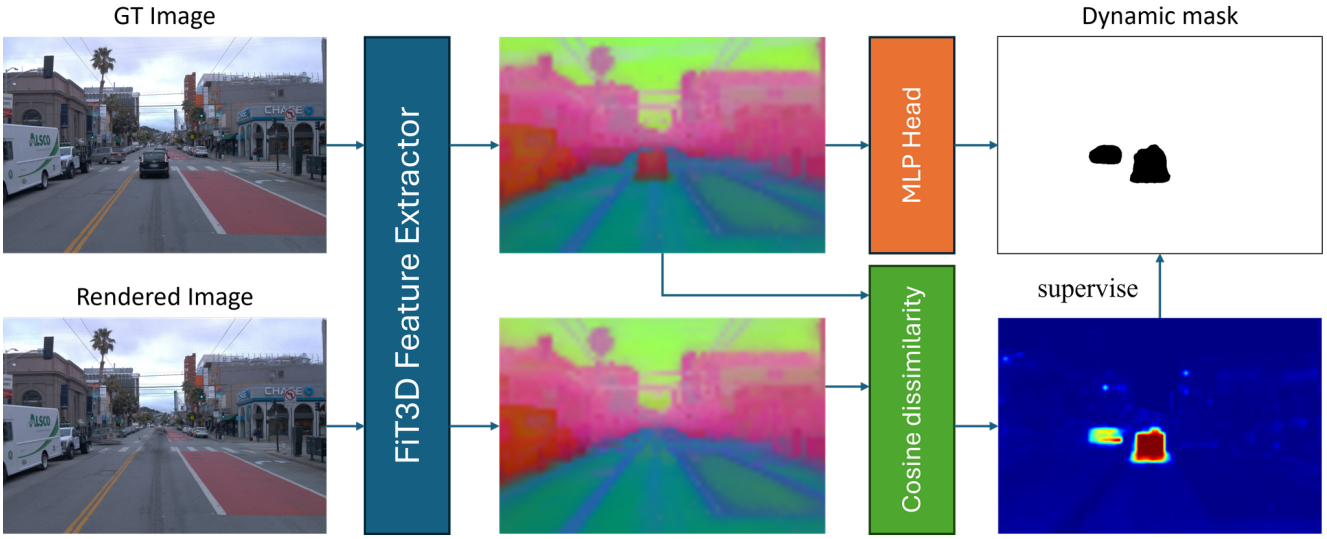

Figure 9. Segmentation Mask Extraction of DeSiRe-GS. We utilize the differences between the rendered image and ground truth to train a dynamic mask decoder.

DINOv2 with gaussian splatting to improve the 3D awareness of the extracted features, which is a perfect match for our setting in the driving world. Therefore, we turn to FiT3D as the feature extractor, producing more clean and robust features, to measure the similarity of two images.

Motion Mask Extractor.

By introducing the learnable decoder, we are not limited to the viewpoints where GT images are available. Instead, given a rendered image, we can first extract the FiT3D features, and then use our decoder to extract the motion mask, without the requirement for ground truth images.

During training, the joint optimization of image rendering and mask prediction will benefit from each other by using the obtained mask M to mask out the dynamic regions. The rendering loss is as follows:

$$\mathcal{L}_{masked-render} = M \odot \|\hat{I} - I\| \tag{23}$$

As we mask out the dynamic regions, the reconstruction at the regions will not be supervised. As a result, the difference between the rendered images and the ground truth images will be become more significant, which benefits the extraction of the desired motion masks.

We provide a few samples in Fig. 10. It can be observed that our model can handle the dynamic objects well, even for far-away pedestrians.

Figure 10. Extracted motion masks using FiT3D features

Temporal Geometric Constraints. Due to the sparsity nature of views in driving scenarios, it tends to overfit to the training views when optimizing gaussian splatting. Single-view image loss often suffers from texture-less area in far distance. As a result, relying on photometric consistency is not reliable. Instead, we propose to enhance the geometric consistency by aggregating temporal information.

Based on the assumption that depth of static regions remains consistent across time from varying views, we designate a cross-view temporal spatial consistency module. For a static pixel (u_r, v_r) in the reference frame, with depth value of d_r, we can project it to the nearest neighboring view, which has the largest amount of overlap. Given the camera intrinsics K and extrinsics T_r, T_n, we can obtain the corresponding pixel location in the neighboring view:

$$[u_n, v_n, 1]^T = KT_nT_r^{-1}\left(d_r \cdot K^{-1}[u_r, v_r, 1]^T\right) \quad (24)$$

Again, we can query the depth value d_n at the position (u_n, v_n). When we project it back to the 3D space, the position should be consistent with the one obtained from back-projecting (u_r, v_r, d_r) to the reference frame.

$$[u_{nr}, v_{nr}, 1]^T = KT_rT_n^{-1}\left(d_n \cdot K^{-1}[u_n, v_n, 1]^T\right) \quad (25)$$

We apply geometric loss to optimize the Gaussians to produce cross-view consistent depth as follows:

$$\mathcal{L}_{uv} = \|(u_r, v_r) - (u_{nr}, v_{nr})\|_2 \quad (26)$$

8. Baselines

- **StreetSurf** [13] is an implicit neural rendering method for both geometry and appearance reconstruction in street views. The whole scene is divided in to close-range, distant-view and sky parts according to the distance of objects. A cuboid close-range hash grid and a hyper-cuboid distant-view model are employed to tackle long and narrow observation space in most street scenes, showcasing good performance in unbounded scenes captured by long camera trajectories.
- **NSG** [24] enables efficient rendering of novel arrangements and views by encoding object transformations and radiance within a learnable scene graph representation. It contains a background node approximating all static parts, several dynamic nodes representing rigidly moving individuals, and edges representing transformations. NSG[24] also combines implicitly encoded scenes with a jointly learned latent representation to describe objects in a single implicit function.
- **SUDS** [31] is a NeRF-based method for dynamic large urban scene reconstruction. It proposed using 2D optical flow to model scene dynamics, avoiding additional bounding box annotations. SUDS develops a three-branch hash table representation for 4D scene representation, enabling a variety of downstream tasks.
- **StreetGS** [38] models dynamic driving scenes using 3D Gaussian splatting. It represents the components in the scene separately, with a background model for static part and an object model for foreground moving objects. To capture dynamic features, the position and rotation of gaussians are defined in an object local coordinate system, which relies on bounding boxes predicted by an off-the-shelf model.
- **HUGS** [46] is a 3DGS-based method addressing the problem of urban scene reconstruction and understanding. It as-

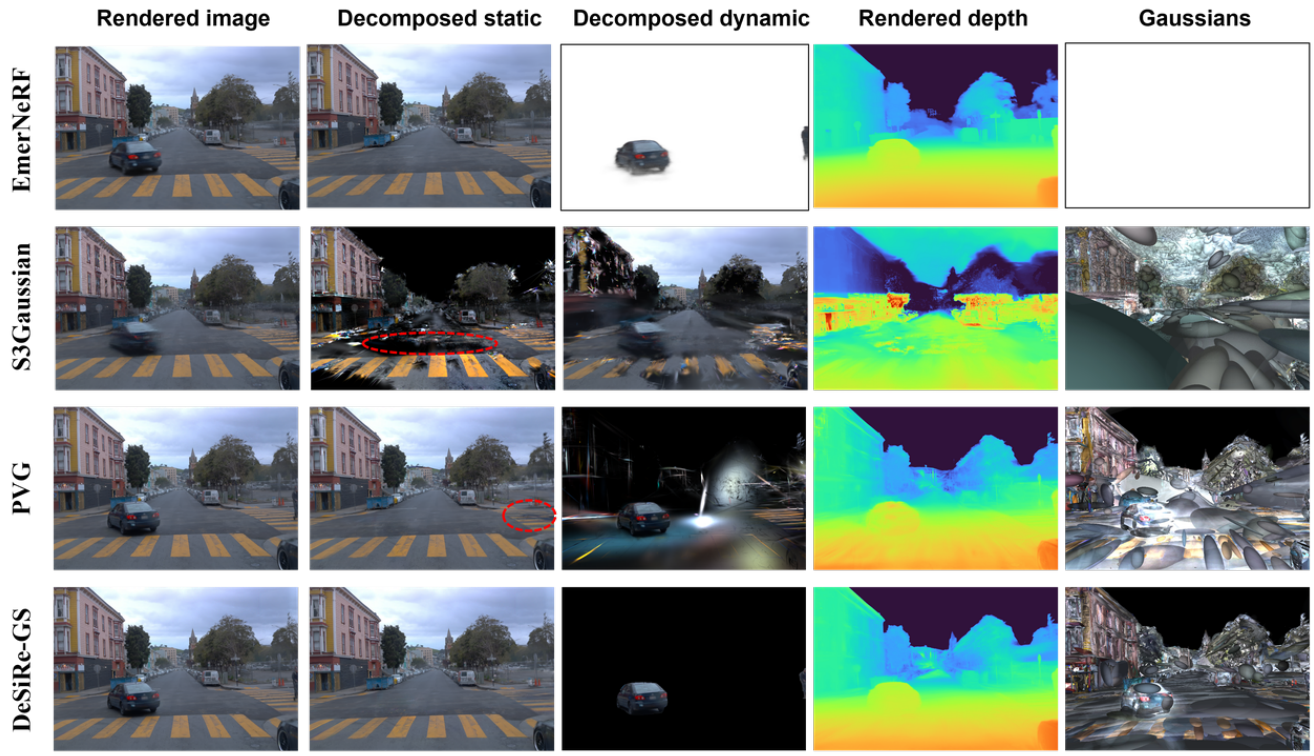

Figure 11. Qualitative Comparison.

sumes that the scene is composed of static rigions and moving vehicles with rigid motions, using a unicycle model to model vehicles' states. HUGS also extends original 3DGS to model additional modalities, including optical flow and semantic information, achieving good performance in both scene reconstruction and semantic reconstruction. Bounding boxes are also required in this process.

- **EmerNeRF** [39] is a NeRF-based method for constructing 4D neural scene representations in urban driving scenes. It decomposes dynamic scenes into a static field and a dynamic field, both parameterized by hash grids. Then an emergent scene flow field is introduced to represent explicit correspondences between moving objects and aggregate temporally-displaced features. Remarkably, EmerNeRF finishes these tasks all through self-supervision.
- **S3Gaussian** [15] is a self-supervised approach that decomposes static and dynamic 3D gaussians in driving scenes. It aggregates 4D gaussian representations in a spatial-temporal field network with a multi-resolution hexplane encoder, where the dynamic objects are visible only within spatial-temporal plane while static objects within spatial-only plane. Then S3Gaussian utilizes a multi-head decoder to capture the deformation of 3D Gaussians in a canonical space for decomposition.
- **Omnire** [6] successfully models urban dynamic scenes using Gaussian Scene Graphs, with different types of nodes tackling sky, background, rigidly moving objects and non-rigidly moving objects. It introduces rigid nodes for vehicles, where the Gaussians will not change over time, and non-rigid nodes for human-ralated dynamics, where local deformations will be taken into consideration. OmniRe additionally employs a Skinned Multi-Person Linear (SMPL) model to parameterize human body model, showcasing good results in reconstructing in-the-wild humans. Notably, Omnire also requires accurate instance bounding boxes for dynamic modeling.
- **PVG** [5] is a self-supervised gaussian splatting approach that reconstructs dynamic urban scenes and isolates dynamic parts from static background. Refer to Sec. 2 for more details about PVG.

In the approaches mentioned above, StreetSurf[13], Mars[36],SUDS[31] and EmerNeRF[39] are based upon NeRF, while others are based upon 3DGS. Notably, among the 3DGS-based approaches, HUGS[46], StreetGS[38] and OmniRe[6] all rely on instance-level bounding boxes for moving objects, which are sometimes difficult to obtain. PVG[5] and S3Gaussian[15] are most closely related to our work, both of which are self-supervised Gaussian Splatting method without reliance on extra annotations.

Method	Dynamic-32 Split						Static-32 Split			
	Image reconstruction			Novel view synthesis			Image reconstruction		Novel view synthesis	
	PSNR ↑	DPSNR ↑	L1↓	PSNR ↑	DPSNR ↑	L1↓	PSNR ↑	L1↓	PSNR ↑	L1↓
3DGS [16]	28.47	23.26	-	25.14	20.48	-	29.42	-	26.82	-
Mars [36]	28.24	23.37	-	26.61	22.21	-	28.31	-	27.63	-
EmerNeRF [39]	28.16	24.32	3.12	25.14	23.49	4.33	30.00	2.84	28.89	3.89
S3Gaussian [15]	31.35	26.02	5.31	27.44	22.92	6.18	30.73	5.84	27.05	6.53
PVG [5]	33.14	31.79	3.33	29.77	27.19	4.84	32.84	3.75	29.12	5.07
Ours	34.56	32.63	2.96	30.45	28.66	4.17	34.57	2.89	31.78	3.93

Table 4. Comparison of methods on the Waymo NOTR Dataset from EmerNeRF.

Segment Name	seg104554	seg125050	seg169514	seg584622	seg776165	seg138251	seg448767	seg965324
Scene Index	23	114	327	621	703	172	552	788

Table 5. Segment Names and Scene IDs of 8 scenes used in OmniRe[6].

Segment Name	seg102319	seg103913	seg109636	seg117188
Scene Index	17	22	50	81

Table 6. Segment Names and Scene IDs of 4 scenes used in PVG[5].

9. Data

We conduct our experiments on the Waymo Open Dataset [28] and the KITTI Dataset [11], both consisting of real-world autonomous driving scenarios.

9.1. Waymo Open Dataset

NOTR from EmerNeRF. NOTR is a subset consisting of diverse and balance sequences derived from Waymo Open Dataset introduced by [39]. It includes 120 distinct driving sequences, categorized into 32 static, 32 dynamic, and 56 diverse scenes covering various challenging driving conditions. Following [15], we incorporate the 32 dynamic scenes and 32 static scenes from NOTR into our testing set. Refer to EmerNeRF[39] for NOTR dataset details.

OmniRe subset. OmniRe[6] selects eight highly complex dynamic driving sequences from Waymo Open Dataset, each including dynamic classes such as vehicles and pedestrains. The Segment IDs of selected scenes are shown in Tab.5.

PVG subset. PVG [5] provides four sequences randomly selected from Waymo Open Dataset, which are also included in our experiments. The Segment IDs are shown in Tab.6.

For the sequences in the Waymo dataset, we follow the same setup as [39]. Camera images are captured from three frontal cameras—FRONT LEFT, FRONT, and FRONT RIGHT—and then resized to a resolution of 640×960. Only the first return of the LiDAR point cloud data is considered. We select the first 50 frames from each dataset for our experiments, and then scale the time range to [0,1].

9.2. KITTI Dataset

For KITTI Dataset, we only test DeSiRe-GS on the subset provided by PVG[5]. Different from the Waymo dataset, we only use the left and right cameras for evaluation on KITTI dataset. The resolution of images from each camera is 375×1242. Similar to Waymo Dataset preprocessing, we randomly choose 50 frames from the whole sequence from each KITTI dataset and rescale time duration to [0,1] with a frame interval of 0.02 seconds.

9.3. Data Source

For Tab. 1, the results of the baselines are taken from PVG [5], since we are using the same dataset and evaluated on the devices. The results of Tab. 2 are sourced from OmniRe [6].

10. Additional Results

10.1. Additional quantitative results

We conducted experiments on the NOTR dataset, and the results are listed in Tab. 4. Following PVG [5], we scaled the camera pose and point clouds during pre-processing. For a fair comparison of depth errors with other methods, we re-map the depth back to the original scale and calculate the depth L1 error. The results in Tab. 3 for ablation studies are without the rescaling.

10.2. Analysis

We compare the rendered depth of various methods in Fig. 11. S3Gaussian [15] fails to predict accurate depth map, because they use only LiDAR point clouds for initialization,

where there are no points at the upper part. Other than the LiDAR points, PVG [5] and DeSiRe-GS randomly sample points, enabling us to render much better depth map.

GS-based methods, such as PVG [5] and S3Gaussian [15] generally outperform NeRF-based methods like EmerNeRF [39] in terms of image rendering. However, the explicit GS methods tend to overfit to the images, thereby performing poorly on depth rendering. With the proposed cross-view consistency, our model can successfully solve the over-fitting problem, achieving satisfactory rendering quality both in image and depth.