Hierarchy UGP: Hierarchy Unified Gaussian Primitive for Large-Scale Dynamic Scene Reconstruction

Supplementary Material

1. Overview

In Section 2, we provide a detailed introduction to the Dynamic City dataset we collected and compare it with Waymo and PandaSet datasets.

In Section 3, we elaborate on the implementation details of our method. This includes how large-scale scenes are partitioned into multiple sub-scenes, how element extraction is performed within each sub-scene, and our Block-wise Objects training strategy and Virtual Warping View supervision. We also conduct further ablation studies on these two strategies.

In Section 4, we introduce the baseline methods used for comparison in the paper. Section 5 presents the specific details of the evaluation process.

In Section 6, we show additional experimental results, including more interpolation and extrapolation comparison experiments, more pedestrian result images, and both qualitative visuals and quantitative evaluation of the projected depth renderings. We also provide further analysis of the experimental results, explaining how our method, Hierarchy UGP, outperforms the previous algorithm [4] in modeling pedestrians.

In Section 7, we demonstrate the ability of our method to perform scene decoupling and scene editing. Finally, in Section 8, we discuss the limitations of Hierarchy UGP.

2. Dynamic City Dataset

Currently, there is a lack of publicly available large-scale dynamic street scene datasets. To address this gap, we present the Dynamic City Dataset, which comprises image and radar data sequences captured at a frequency of 10 Hz, spanning street scenes from 600 meters to over one kilometer.

The data structure of the Dynamic City is comparable to that of existing datasets such as Waymo [12] and PandaSet [15], including radar data with a 10 Hz frame rate, RGB image data, and camera poses. However, a distinguishing feature of Waymo and PandaSet is their focus on small-scale street scenes. For instance, Waymo's single-camera image sequences typically consist of around 200 frames, with a collection duration of less than 20 seconds, while PandaSet contains fewer than 100 frames, captured over less than 10 seconds. In contrast, the Dynamic City contains sequences with thousands of frames, with each sequence lasting more than one minute, covering significantly broader street scenes.

We are committed to open-sourcing the Dynamic City Dataset. Given the current absence of large-scale dynamic street scene datasets in the public domain, we believe that Dynamic City can contribute to the advancement of research in the field of large-scale dynamic street scene reconstruction.

3. Implementation Details

3.1. Dynamic City Dataset

Dividing Large-scale Scenes into Sub-scenes Given the large-scale scene covered by numerous images from calibrated cameras, we first transform the poses of all cameras into the AABB coordinate system. Next, we define each sub-scene as 110 meters by 110 meters, allowing us to partition the scene into multiple sub-scenes. We then assign the cameras to their respective sub-scenes based on the spatial relationships between the camera poses and the sub-scenes. Inspired by [9, 13], we introduce a 50% overlap between adjacent sub-scenes. This overlap enhances the consistency between neighboring sub-scenes and minimizes boundary artifacts that may arise when merging adjacent sub-scenes.

Element Extraction We employ a multi-modal approach combining SAM2 [10] for visual instance tracking with LiDAR-based geometric clustering. For temporal consistency, SAM2 maintains object identity $id_t \in \mathbb{Z}^+$ across frames using

$$\min_{\theta} \sum_{t=1}^{T} \mathcal{L}_{\text{track}} \left(\mathbf{M}_t^{\text{SAM2}}, \mathbf{M}_{t-1}^{\text{SAM2}} \right) , \tag{1}$$

where $\mathbf{M}_t^{\text{SAM2}}$ denotes the instance mask at time t, and $\mathcal{L}_{\text{track}}$ combines appearance and motion costs through a weighted sum $\lambda_{\text{app}} = 0.7$ and $\lambda_{\text{mot}} = 0.3$. For LiDAR point cloud $\mathcal{P} = \{\mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$, we apply three-stage clustering starting with DBSCAN [6] using neighborhood radius $\epsilon = 0.3$m and minimum cluster size $N_{\text{min}} = 10$ presented in

$$\mathcal{C}_k^{\text{DBSCAN}} = \{\mathbf{p} \in \mathcal{P} \mid \|\mathbf{p} - \mathbf{c}_k\|_2 \leq \epsilon \wedge |\mathcal{N}_\epsilon(\mathbf{p})| \geq N_{\text{min}}\} . \tag{2}$$

Euclidean Clustering [2] subsequently merges neighboring clusters within $d_{\text{thresh}} = 0.5$m,

$$\mathcal{C}_m^{\text{Eucl}} = \left\{ \mathbf{p} \in \mathcal{P} \mid \min_{\mathbf{q} \in \mathcal{C}_m} \|\mathbf{p} - \mathbf{q}\|_2 \leq d_{\text{thresh}} \right\} , \tag{3}$$

followed by Gaussian Mixture Refinement [1] with $K = 20$ components and convergence tolerance $\tau = 10^{-3}$ as indicated in

$$p(\mathbf{p}_i) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{p}_i \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \,. \tag{4}$$

Object dimensions are estimated via oriented bounding boxes using PCA-derived principal axes with eigenvalue ratio threshold $\gamma = 0.8$:

$$\mathbf{B}_k = \left[\boldsymbol{\mu}_k, \mathbf{v}_k^{\max} - \mathbf{v}_k^{\min}, \theta_k\right] \tag{5}$$

where $\mathbf{v}_k^{\min/\max}$ are the principal component extremes along the dominant eigenvector $\mathbf{e}_1$ satisfying $\lambda_1 / \sum_i \lambda_i > \gamma$.

Finally, we match the objects detected by both the camera and LiDAR, and smooth the resulting trajectories to improve the accuracy and stability of the tracking. In this way, we obtain the time-dependent motion prior of the elements in the scene, $M(t) = (R_t \mid T_t)$.

Initialization We use the LiDAR point cloud collected in the dataset as the initialization for the 3D point cloud. We downsample these 3D point clouds, limiting the maximum number of points in the initialized background point cloud to 6×10^5 and 3×10^4 for each dynamic object. For non-rigid objects, spatiotemporal interpolation is applied to their initialized point clouds to improve consistency. We construct a Global UGP scaffold through 30,000 iterations of coarse training, which serves as the foundation for optimizing each sub-scene at the sub-scene level [9] .

Block-wise Objects training strategy In large-scale dynamic scenes, the motion of objects is highly complex, which poses significant challenges for the reconstruction of dynamic objects. For objects that remain confined to the same sub-scene throughout their motion, whether rigid or non-rigid, a single UGP model can be assigned to each object to achieve high-quality reconstruction. However, for objects that span multiple sub-scenes, sharing a single UGP model across these sub-scenes frequently results in noticeable artifacts caused by viewpoint inconsistencies between the sub-scenes.

To address this issue, inspired by [13] , we adopt a block-wise training strategy for objects spanning across sub-scenes. Specifically, for such objects, we assign a separate UGP model to each sub-scene. Each UGP model is independently optimized according to the object's behavior within the corresponding sub-scene. This strategy ensures high-quality representation in each sub-scenes, and during rendering, the corresponding UGP model is selected based on the spatial information of the objects. Experimental results demonstrate that our block-wise training strategy effectively enables high-quality reconstruction of complex dynamic objects in large-scale dynamic scenes. The results in Figure 1 illustrates the effectiveness of the block-wise objects training strategy.

w/ Block-wise Objects w/o Block-wise Objects

Figure 1. **More ablation on Block-wise Objects training strategy.** We emphasized the impact of our Block-wise Objects training strategy on vehicles to showcase its effectiveness.

Training Process We divide the large-scale dynamic scene spatially into multiple sub-scenes, each measuring 110 meters by 110 meters, and then perform 30,000 iterations of training for each sub-scene in parallel. We use the Adam optimizer, setting the learning rate for the UGP parameter t to 1.6×10^{-4} and the learning rate for the parameter scaling_t to 5×10^{-3}.

The temporal scale s_t of the UGP controls its temporal influence range, for non-rigid objects, we initialize s_t based on the frame rate of the data collection:

$$s_t = -\frac{(\lambda v \mathrm{d}t)^2}{2log(o_{th})}, \tag{6}$$

where, λ is a hyperparameter used to fine-tune the initial s_t, v is the image acquisition frequency, $\mathrm{d}t$ is the time duration corresponding to each frame, and o_{th} is the opacity threshold for pruning. we set λ to 0.15, v to the image acquisition frequency of the Dynamic City, which is 10. And o_{th} is set to 0.005.

The configurations for other parameters follow [8] . Inspired by [9] , we construct a hierarchical structure for each sub-scene after the training phase and conduct 15,000 iterations of post-training for the intermediate nodes within each sub-scene. Finally, all sub-scenes are merged together. Notably, since dynamic objects generally occupy limited space within the scene, we empirically designate them as leaf nodes to achieve a better trade-off between rendering quality and speed.

Objective Function We use the following objective function to optimize the large-scale dynamic scene:

$$\mathcal{L} = \lambda_{\text{rgb}}\mathcal{L}_{\text{rgb}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{warp}}\mathcal{L}_{\text{warp}} \tag{11}$$

where $\mathcal{L}_{\text{rgb}}$ combines the L1 and SSIM losses. $\mathcal{L}_{\text{depth}}$ is the L1 loss between rendered depth and the depth generated by projecting sparse LiDAR points onto the camera plane. $\mathcal{L}_{\text{warp}}$ is the L1 loss between the rendered image and the virtual warping view. The virtual warping view supervision significantly improves the performance of novel view synthesis. During training, we set λ_{rgb} to 0.8, and both λ_{depth} and λ_{warp} to 0.1. Next, we will discuss the virtual warping view in detail.

Virtual Warping View To improve the quality of extrapolated novel views, inspired by [5] , we introduce virtual warping view. The specific implementation is as follows:

Virtual View and Depth Map Generation. We generate multiple virtual views by translating the position of the source view(the camera views from the dataset) by 1 meter, 2 meters, and 3 meters, and applying small-angle rotations (e.g., 5 degrees, 10 degrees, and 15 degrees). Additionally, we generate depth maps for both the source views and the virtual views using the LiDAR point cloud data.

Warp RGB Generation. First, we use the RGB image and depth map of the source view to back-project each pixel into the camera coordinate system of the source view. Then, using the depth map of the virtual view, we back-project the pixels into its corresponding camera coordinate system. By applying the coordinate mapping from the virtual view to the source view, the 3D points of the virtual view are transformed into the camera coordinate system of the source view and subsequently projected onto the 2D image plane of the source view. Reliable points are selected by comparing the projected depth with the depth map of the source view. Points are deemed reliable if the relative error is below 0.05. Ultimately, we project the source view RGB values that meet the conditions onto the virtual viewpoint, thereby generating the warp RGB for the virtual camera.

Training with Virtual Warping View. During the training process, we utilize both the source view and the virtual warping view as the training set. The virtual warping view provides additional perspective information, enabling our Hierarchy UGP to perform well on novel views. Experimental results demonstrate that virtual warping view supervision significantly improving the quality of extrapolated novel views. The Figure 2 illustrates the effectiveness of the virtual warping view supervision.

3.2. Waymo Dataset

Initialization We filter out dynamic objects from the radar point cloud and sample 8×10^5 points. Following the OmniRe method, we apply inverse distance sampling to obtain 3×10^5 randomly distributed spherical points. These points are then aggregated to form the background points. The point cloud of dynamic objects extracted from the radar serves as the initial point cloud for dynamic objects. For non-rigid objects, the corresponding timestamp from the radar point cloud is used as the initialization of μ_t. In addition, since the point clouds of non-rigid objects scanned by the radar at each moment are relatively sparse, we weight each point according to the point cloud density commonly used in the field. After weighting, we perform random sampling followed by k-nearest neighbor interpolation, during which temporal information is also interpolated. In total, 1×10^4 points are added.

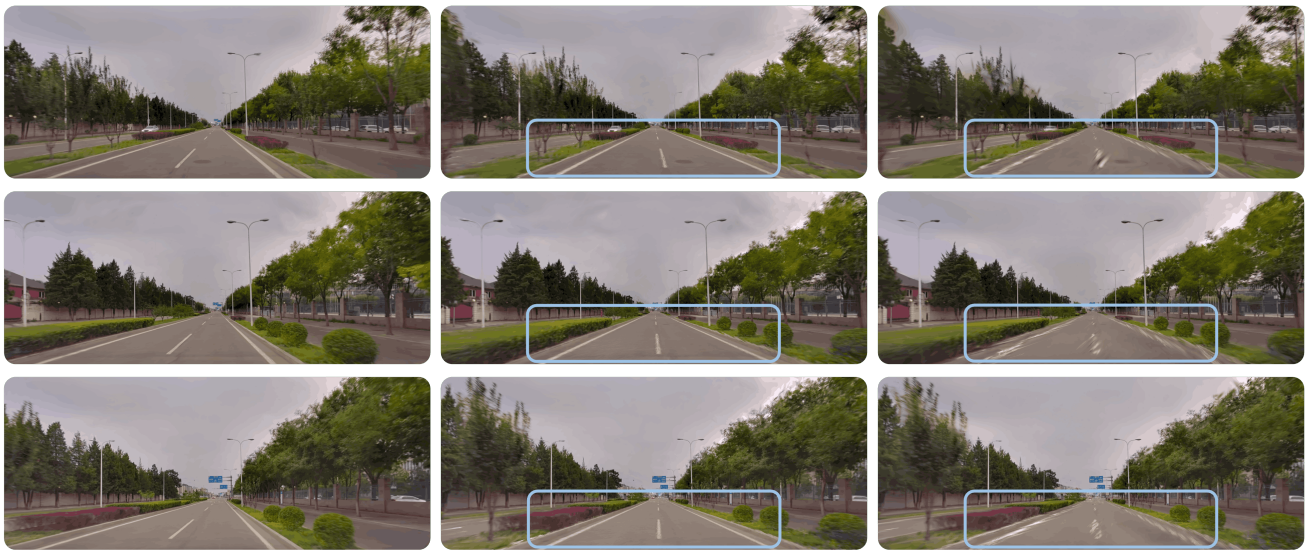
Training Unlike large-scale scenes, in typical scenes, our background model undergoes UGP adjustment based on scale. For the learning rates of the UGP parameters, we align with 3DGS. For non-rigid objects, we introduce a strategy to adjust based on t-grad, with a threshold of 2e-4. All model adjustments stop after 15,000 iterations, but non-rigid objects continue to be pruned until 20,000 steps. Our training is completed within two hours on an RTX 4090 GPU.

4. Baseline

All baseline methods are implemented utilizing their official code releases. Next, we will provide a brief overview of the baseline methods and discuss their limitations.

4DGS [17] extends 3DGS into four-dimensional space. During rendering, it first computes the conditional 3DGS and marginal t, followed by alpha blending. Dynamic reconstruction is achieved by incorporating temporal information into the model. However, it is unable to reconstruct large-scale scenes, and the reconstruction quality deteriorates in regions with significant motion.

HierarchicalGS [9] achieves large-scale static scene reconstruction by partitioning the scene and designing a hierarchical structure. However, it is limited to static scenes and cannot reconstruct dynamic elements.



GT | Novel View w/ Virtual Warping View | Novel View w/o Virtual Warping View

Figure 2. **ablation on virtual warping view.** The ground truth represents the original view, while the novel view is obtained by shifting the original view 2 meters to the left. The results in the figure demonstrate that the virtual warping view supervision can effectively improve the visual quality of the novel view.

	Large 001				Large 002			
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑
OmniRe [4]	24.48	0.776	0.336	19	22.17	0.734	0.386	20
Hierarchical GS [9] τ=1	27.00	0.870	0.171	2	25.33	0.852	0.185	4
Ours τ=1	27.29	0.871	0.164	13	25.46	0.853	0.182	8
Hierarchical GS τ=3	26.96	0.869	0.172	11	25.31	0.851	0.186	9
Ours τ=3	27.24	0.870	0.165	23	25.43	0.852	0.183	13
Hierarchical GS τ=6	26.85	0.865	0.178	21	25.23	0.847	0.192	12
Ours τ=6	27.13	0.866	0.171	28	25.34	0.849	0.189	18
Hierarchical GS τ=9	26.74	0.861	0.188	28	25.14	0.843	0.202	15
Ours τ=9	27.01	0.862	0.180	32	25.25	0.844	0.198	21

Table 1. **Quantitative Comparison on Dynamic City.** We selected every 10th frame as a test frame and computed visual quality metrics and frames per second (FPS) of our method compared to Hierarchical GS on Large 001 and Large 002. We loaded the entire hierarchy structure of the large scene and used LOD level $\tau = 1, 3, 6, 9$. Each cell is colored to indicate the best.

StreetGS [16] represents the scene as a static background and dynamic foreground, organizing them into a scene graph. It models both components separately using 3DGS, enabling efficient and high-quality dynamic street view reconstruction. However, it is unable to model deformable dynamic objects and struggles with reconstructing large-scale scenes.

OmniRe [4] represents the scene as a Gaussian graph, dividing it into four types of nodes: background, sky, rigid objects, and non-rigid objects. It employs 3DGS to model the background and rigid objects, SMPL and deformable-GS to model non-rigid objects, and a cubemap to model the sky, achieving comprehensive street view reconstruction. However, it is incapable of handling large-scale scenes, and its ability to reconstruct arbitrary dynamic regions is limited, particularly in areas with significant motion.

5. Evaluation

We use PSNR, SSIM, and LPIPS to quantitatively evaluate the quality of our reconstruction and interpolation. Every tenth frame is selected as a test frame. Additionally, we compute pedestrian-specific metrics on the Waymo dataset to demonstrate the accuracy of our method in modeling

	Large 001			Large 002			Large 003			Large 004		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
PVG [3]	25.03	0.825	0.321	23.27	0.792	0.373	23.79	0.788	0.341	21.82	0.743	0.366
OmniRe [4]	24.48	0.776	0.336	22.17	0.734	0.386	23.00	0.729	0.323	23.07	0.737	0.222
Hierarchical GS [9]	27.00	0.870	0.171	25.33	0.852	0.185	24.73	0.840	0.191	25.68	0.857	0.186
Ours	27.29	0.871	0.164	25.46	0.853	0.182	25.24	0.845	0.185	26.5	0.860	0.181

Table 2. **More Quantitative Comparison on Dynamic City.** We selected every 10th frame as a test frame and computed visual quality metrics in four large scenes. **used LOD level** $\tau = 1$. Each cell is colored to indicate the best.

Methods	**StreetGS**				**OmniRe**				**Ours**			
	PSNR↑	LPIPS↓	PSNR*↑	SSIM*↑	PSNR↑	LPIPS↓	PSNR*↑	SSIM*↑	PSNR↑	LPIPS↓	PSNR*↑	SSIM*↑
327	32.46	0.120	26.66	0.823	36.24	0.090	28.56	0.839	36.43	0.085	34.52	0.947
614	30.95	0.161	30.74	0.897	30.97	0.169	26.36	0.795	30.76	0.158	35.41	0.964
621	32.01	0.102	24.98	0.810	35.05	0.074	27.23	0.825	35.04	0.071	32.72	0.936
703	31.16	0.142	25.98	0.820	33.37	0.094	27.09	0.806	33.51	0.098	32.65	0.933
788	30.71	0.161	25.92	0.824	31.24	0.150	25.43	0.789	31.73	0.139	32.96	0.949

Table 3. **More Quantitative Comparison on Waymo.** We computed visual quality metrics, where * denotes the metrics for the pedestrian regions. Each cell is colored to indicate the best.

Methods	**MSE↓**	**MAE↓**
Hierarchical GS [9]	0.0220	0.1042
Ours	**0.0161**	**0.0887**

Table 4. **Depth Quantitative Comparison on Dynamic City.** We present the quantitative tests comparing rendering depth and lidar depth using normalized depth metrics (MSE and MAE).

pedestrians. For the pedestrian metrics, we use OmniRe's code to generate pedestrian masks. To quantitatively assess our extrapolation quality, we use FID. Specifically, we translate the camera viewpoint by two meters (changing lanes) and render new viewpoint images, then compute the FID against the ground truth. We use pytorch-FID [11] for the FID calculation.

6. Additional Experiments and Anlysis

We present additional comparative experimental results and analysis to further demonstrate the superiority of our method over the baseline methods.

6.1. Qualitative Comparison

The results in Figure 3 and Figure 4 show the interpolation and extrapolation comparison on Dynamic City dataset, , while Figure 5 highlights the comparison of humans in the Waymo dataset. The comparative experimental results indicate that our method achieves state-of-the-art performance in both large-scale dynamic scenes and large-motion scenarios.

6.2. Quantitative Comparison

Table 1 presents a comparative analysis between our method and HierarchicalGS on the Large 001 and Large 002, evaluating visual quality metrics and rendering performance across varying levels of granularity denoted by τ [9] . Experiments were conducted with τ set to 1, 3, 6, and 9. The results demonstrate that our method consistently surpasses HierarchicalGS across all levels of detail (LOD) in both visual quality and rendering efficiency. Furthermore, even at the largest granularity level (τ=9), our method significantly exceeds the visual quality of StreetGS and OmniRe, while achieving real-time rendering.

We present additional quantitative experiments on Dynamic City in Table 2 , where the experimental results demonstrate that our method achieves state-of-the-art performance.

The quantitative results in Table 3 presents additional comparative experiments on the Waymo dataset. The results demonstrate that our method achieves state-of-the-art performance on the Waymo dataset. Notably, in terms of pedestrians reconstruction, our UGP significantly outperforms other algorithms.

6.3. Analysis of Human Results

The advantages of UGP over the SMPL-based approach, such as OmniRe [4], for pedestrian modeling can be identified in two key aspects:

First, OmniRe [4] uses 4DHuman [7] to track pedestrians within the scene and extract relevant motion parameters. However, in many scenarios, particularly when pedestrians occupy a small spatial area or exhibit atypical motion poses

Figure 3. **Interpolation Comparison on Dynamic City Dataset.** We conducted more qualitative experiments on the Dynamic City Dataset, selecting every 10th frame as a test frame. The results show that our method significantly outperforms others in reconstruction quality, achieving high-quality reconstruction of large-scale dynamic scenes.

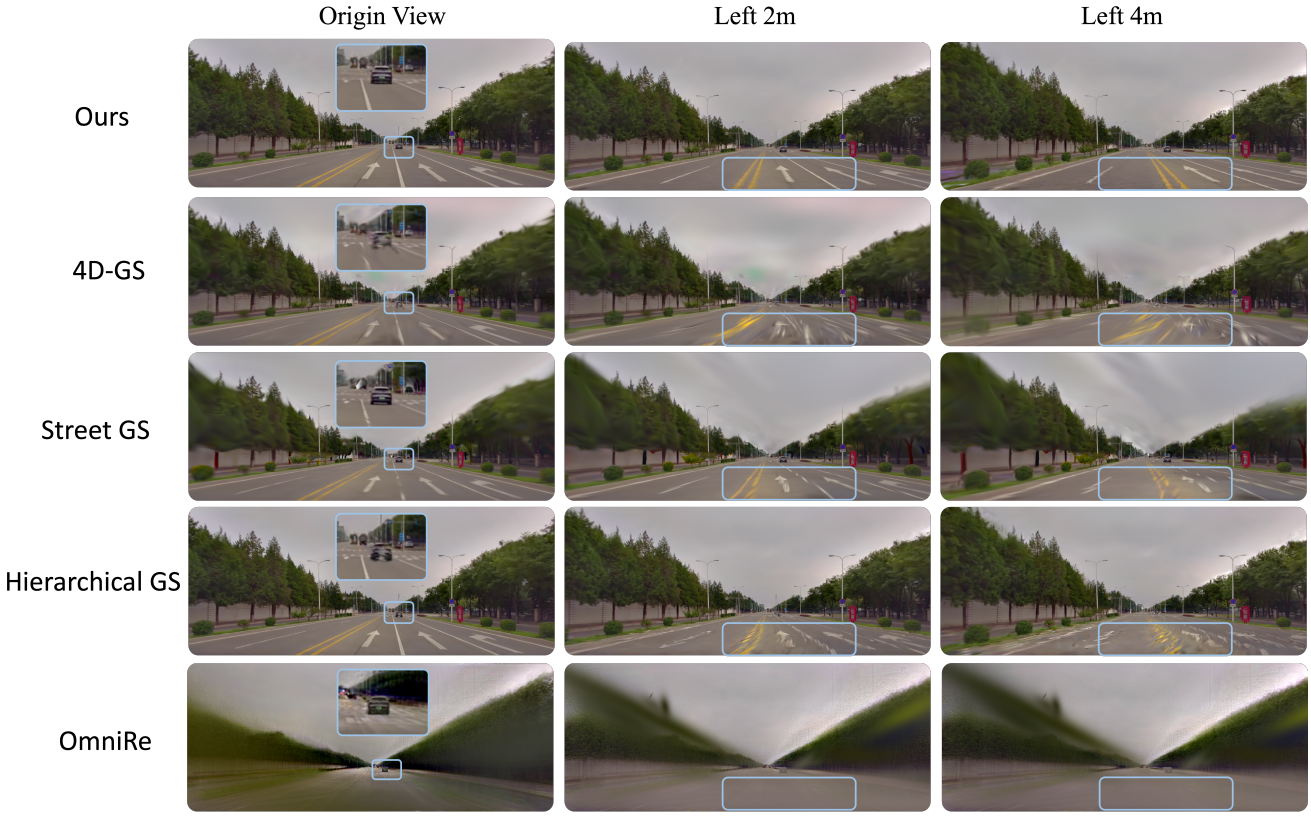


Figure 4. **Extrapolation Comparison on Dynamic City Dataset.** We shifted the camera 2 meters and 4 meters to the left from the original viewpoint to evaluate the extrapolation performance. The results show that our method significantly outperforms others in extrapolation.

Figure 5. **Qualitative Comparison on Waymo Dataset.**

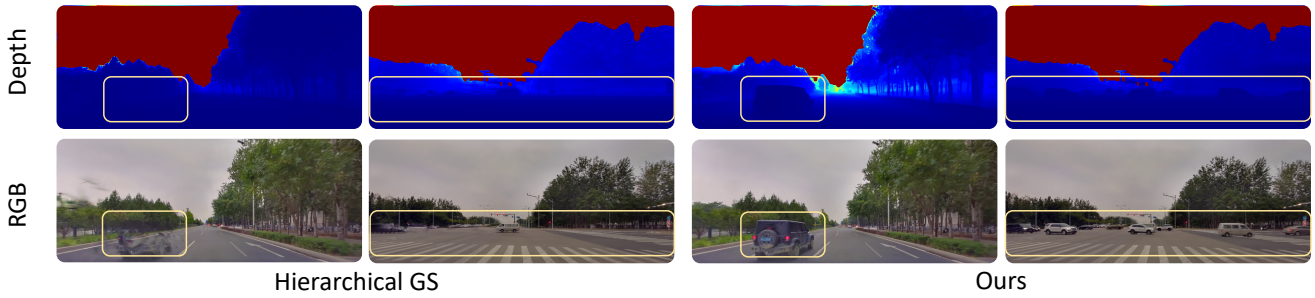


Figure 6. **Rendering Depth Results on the Dynamic City Dataset.**

(such as cycling or using a wheelchair), these pedestrians are often missed by 4DHuman. In such cases, missed pedestrians are fitted using the deformable GS [14] model within the OmniRe algorithm. Experimental results indicate that this fitting process often produces suboptimal results.

Second, UGP is defined in 4D space, which enhances its ability to accurately fit object motion and deformation. In contrast, SMPL-based methods are inherently constrained by the limitations of their data-driven nature.

6.4. Depth Rendering Results

We conduct a comparison experiment of rendering depth with Hierarchical GS [9] , the method closest to ours in large-scale scenes. We show the comparison of rendering depth in Figure 6 and present the quantitative tests on rendering depth and lidar depth in Table 4. The experimental

Figure 7. **Decomposition results on the Dynamic City Dataset.**

Figure 8. **Scene Editing on the Dynamic City Dataset.**

results demonstrate that our rendering depth is more accurate than that of Hierarchical GS [9] .

7. Application

Hierarchy UGP enables foreground-background decomposition and scene editing.

As demonstrated in Figure 7, our method effectively decomposes the scene, with both the foreground and background displaying exceptional visual quality. Additionally, as shown in Figure 8, our approach enables scene editing, a feature that is essential for autonomous driving simulation applications.

8. Limitations

Hierarchy UGP still has some limitations. First, our method does not explicitly model lighting conditions, which limits its ability to simulate diverse environments, such as rain or nighttime. Incorporating lighting modifications into im-

age rendering using diffusion models presents a potential research direction. Additionally, training Hierarchy UGP still requires approximately three hours. Directly predicting UGPs in a feed-forward manner might serve as a prospective avenue for future research.

References

[1] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006. 2

[2] Ivan Bogoslavskyi and Cyrill Stachniss. A comparative study of 3d segmentation algorithms for lidar data. In *ISARC*, pages 1069–1074, 2017. 1

[3] Yurui Chen, Chun Gu, Junzhe Jiang, Xiatian Zhu, and Li Zhang. Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. *arXiv preprint arXiv:2311.18561*, 2023. 5

[4] Ziyu Chen, Jiawei Yang, Jiahui Huang, Riccardo de Lutio, Janick Martinez Esturo, Boris Ivanovic, Or Litany, Zan Gojcic, Sanja Fidler, Marco Pavone, et al. Omnire: Omni urban scene reconstruction. *arXiv preprint arXiv:2408.16760*, 2024. 1, 4, 5

[5] Kai Cheng, Xiaoxiao Long, Wei Yin, Jin Wang, Zhiqiang Wu, Yuexin Ma, Kaixuan Wang, Xiaozhi Chen, and Xuejin Chen. Uc-nerf: Neural radiance field for under-calibrated multi-view cameras in autonomous driving. *arXiv preprint arXiv:2311.16945*, 2023. 3

[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, 96:226–231, 1996. 1

[7] Shubham Goel, Georgios Pavlakos, Jathushan Rajasegaran, Angjoo Kanazawa, and Jitendra Malik. Humans in 4D: Reconstructing and tracking humans with transformers. In *ICCV*, 2023. 5

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 3

[9] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024. 1, 2, 3, 4, 5, 7, 8

[10] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 1

[11] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, 2020. Version 0.3.0. 5

[12] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020. 1

[13] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8248–8258, 2022. 1, 2

[14] 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. 7

[15] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101. IEEE, 2021. 1

[16] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians for modeling dynamic urban scenes. In *ECCV*, 2024. 4

[17] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4d gaussian splatting. In *International Conference on Learning Representations (ICLR)*, 2024. 3