

AD-GS: Object-Aware B-spline Gaussian Splatting for Self-Supervised Autonomous Driving

Jiawei Xu [1], Kai Deng [1], Zexin Fan [1], Shenlong Wang [2], Jin Xie [3] and Jian Yang [1]

[1] College of Computer Science, Nankai University

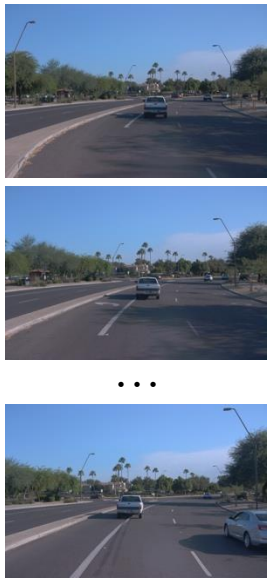
[2] University of Illinois Urbana-Champaign

[3] School of Intelligence Science and Technology, Nanjing University

Problem Definition

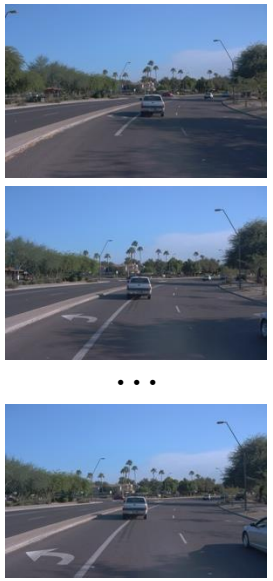
Autonomous Driving Rendering without Manual Labeling

- Input: Multi-view and multi-frame images and lidar points.
- Output: Any-view and any-frame Images.
- ***Note: Without any manually labeled object positions or poses.***



Images

Lidar Points

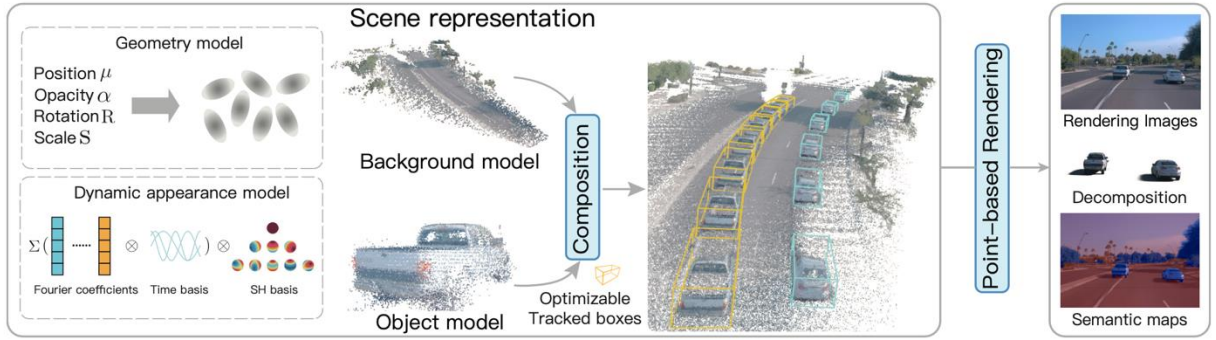


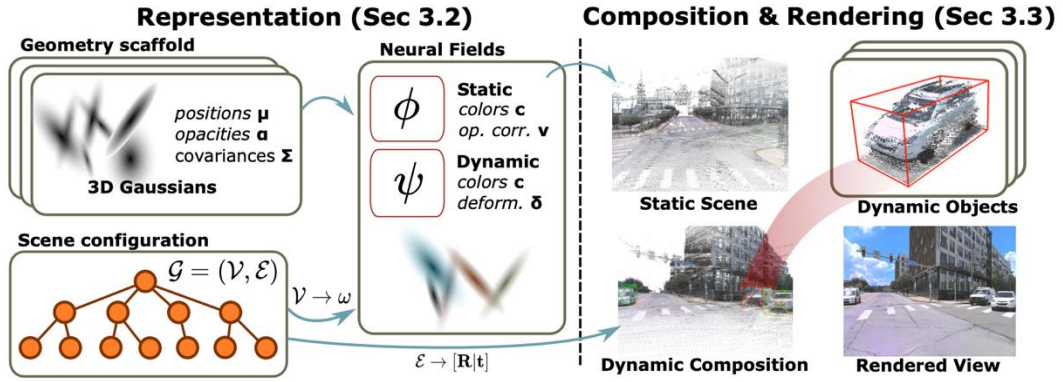
Any-View and Any-Frame Images

Related Work

Autonomous Driving Rendering with Manual Labeling

- Street-GS[1]: Apply Gaussians to model the scenes and objects in manual labeled boxes.
- 4DGF[2]: Make Gaussians deformable for each objects.
- ***Require precise manually labeled object positions and poses.***

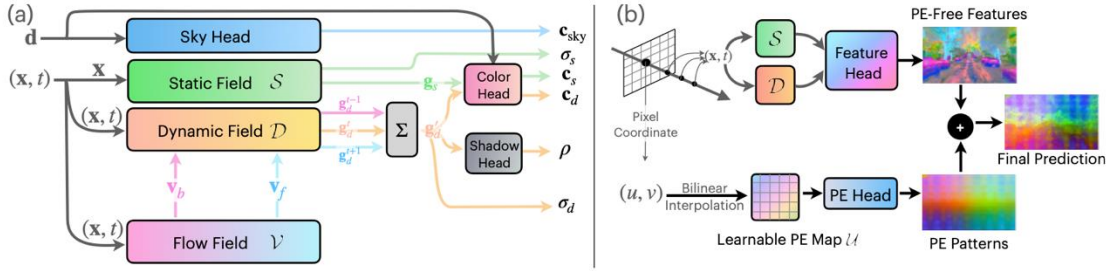


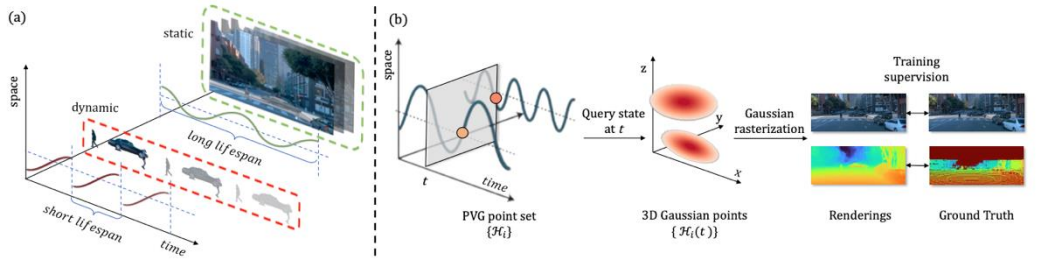


Related Work

Autonomous Driving Rendering without Manual Labeling

- EmerNeRF[3]: Self-Supervise rendering models with the features from DINO[4].
- PVG[5]: Fit the motions with trigonometric function and make Gaussians have life peaks.
- ***Unsatisfactory rendering quality.***



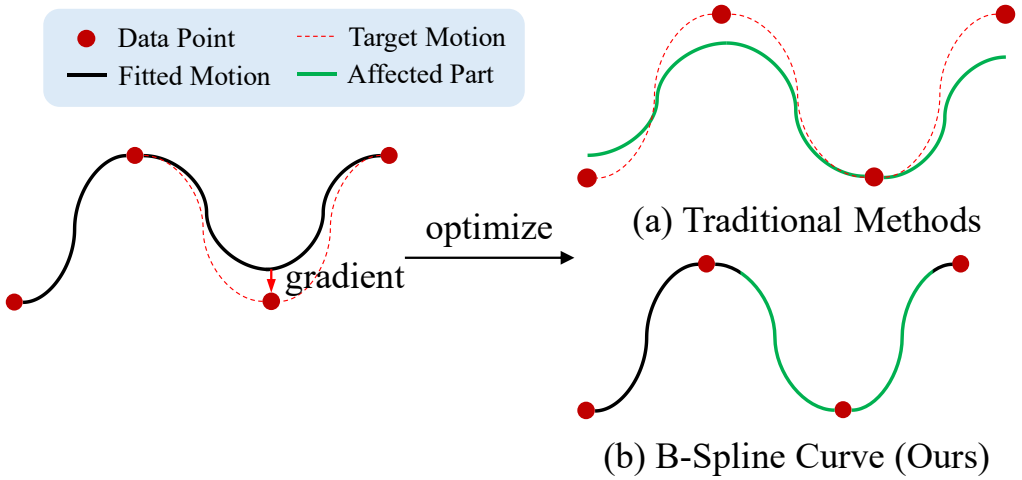


Methods

Modeling Motion with B-Spline Curve

- Traditional methods use trigonometric functions and neural networks for motion fitting.
- ***These methods cannot offer local detail fiting, resulting in artifacts.***
- ***B-splines only optimize the nearby control points of a given training sample rather than all points.*** **ADD** B-Spline functions could handle these problems.

$$\boldsymbol{\mu}' = \boldsymbol{\mu} + \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(t) + \sum_{l=1}^{L} \mathbf{a}_l \sin(t \cdot l\pi) + \mathbf{b}_l \cos(t \cdot l\pi)$$



Methods

Modeling Motion with B-Spline Curve

- A k order B-Spline curve with $n+1$ $(k \leq n)$ control points p_i can be formulated as

$$p(t) = \sum_{i=0}^{n} p_i B_{i,k}(t).$$

$$B_{i,k}(t) = \frac{t - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(t) \,. \quad B_{i,1}(t) = \begin{cases} 1 & \text{if } t_i < t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

- Similarly The quaternion B-Spline curve can be formulated as[6]

$$q(t) = q_0^{\widetilde{B}_{0,k}(t)} \prod_{i=1}^{n} \exp(\omega_i \widetilde{B}_{i,k}(t)) \,,$$

$$\begin{aligned} \widetilde{B}_{i,k}(t) &= \sum_{j=i}^{n} B_{j,k}(t) \\ &= \begin{cases} \sum_{j=i}^{i+k} B_{j,k}(t) & \text{if } t_i < t < t_{i+k-1} \\ 1 & \text{if } t \geq t_{i+k-1} \\ 0 & \text{if } t \leq t_i \end{cases} \end{aligned}$$

- The basic function can be quickly calculated through the matrix format[7].

Methods

Object-Aware Splatting with Temporal Mask

- ***Traditional methods split the scenes under noisy pseudo segmentations.***
- ***Simplify the segmentation into to two class: objects and background***.
- The points on an object will be deformed by the B-spline curves, otherwise they will be considered stationary.
- Render and self-supervise object mask through α-blending:

$$M_{obj} = \sum_{i} 1\{i \in \Omega\}\alpha_i \prod_{j=0}^{i-1}(1-\alpha_j)$$

$$\mathcal{L}_{obj} = -M_{obj}^{gt} \log M_{obj} - \left(1 - M_{obj}^{gt}\right) \log(1 - M_{obj})$$

- $1\{i \in \Omega\}$ means whether the Gaussian represents an object.

Methods

Object-Aware Splatting with Temporal Mask

- Some objects like cars may not appear in all frames.
- Add ***a temporal visibility mask*** for the object Gaussians:

$$\sigma'(t) = \sigma \cdot e^{-\frac{1}{2}\left(\frac{t-\mu_t}{s}\right)^2}, s = \begin{cases} s_1, t < \mu_t \\ s_2, t \geq \mu_t \end{cases}$$

- To make the object Gaussians appear as longer as possible, add a regularization term:

$$\mathcal{L}_s = \left\| \frac{2 \cdot \Delta_f}{s_1 + s_2} \right\|_1$$

- Δ_f means the average time gap between two adjacent frames.

Methods

Self-Supervised Optimization

- Render and self-supervise the inverse depth map with shift-and-scale loss[8, 9]:

$$\mathcal{L}_d = \left\| w * \frac{1}{D} + q - \frac{1}{D^{gt}} \right\|_1, \frac{1}{D} = \sum_i \frac{1}{d_i} \alpha_i \prod_{j=0}^{i-1} (1 - \alpha_j)$$

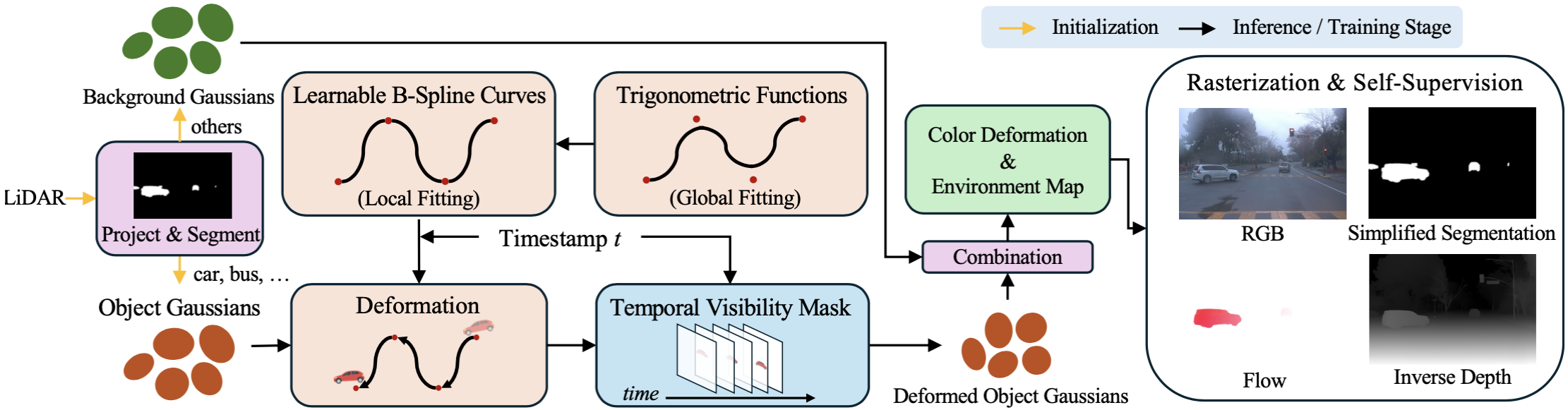
- Render and self-supervise the optical flow:

$$\mathcal{L}_f = \left\| f_{proj}(X) - X^{gt} \right\|_1, X = \sum_i (\mu_i + \Delta\mu_i) \alpha_i \prod_{j=0}^{i-1} (1 - \alpha_j)$$

- Regularization makes nearby Gaussians have similar deformations and temporal masks:

$$\mathcal{L}_r = \sum_{\boldsymbol{\beta} \in \{\mathbf{p}, \mathbf{a}, \mathbf{b}, \{s_0, s_1\}\}} \sum_{\beta_i \in \boldsymbol{\beta}} var(\beta_i)$$

Pipeline



$$\mathcal{L} = (1-\lambda_c)\mathcal{L}_1 + \lambda_c\mathcal{L}_{D-SSIM} + \lambda_d\mathcal{L}_d + \lambda_f\mathcal{L}_f + \lambda_{obj}\mathcal{L}_{obj} + \lambda_{sky}\mathcal{L}_{sky} + \lambda_s\mathcal{L}_s + \lambda_r\mathcal{L}_r$$

Experiments

Table 1. Quantitative comparisons on the KITTI [7] dataset. We follow the experimental setup of SUDS [32], and the color of each cell shows the best and the second best. "Annotations" means whether the model is assisted by the manual 3D annotations.

		KITTI-75%			KITTI-50%			KITTI-25%		
Model	Annotations	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
StreetGS [35]	✓	25.79	0.844	0.081	25.52	0.841	0.084	24.53	0.824	0.090
ML-NSG [6]	✓	28.38	0.907	0.052	27.51	0.898	0.055	26.51	0.887	0.060
4DGF [5]	✓	31.34	0.945	0.026	30.55	0.931	0.028	29.08	0.908	0.036
SUDS [32]		22.77	0.797	0.171	23.12	0.821	0.135	20.76	0.747	0.198
Grid4D [34]		23.79	0.877	0.064	24.07	0.880	0.061	22.25	0.846	0.074
PVG [2]		27.13	0.895	0.049	25.96	0.885	0.053	22.59	0.841	0.078
AD-GS (Ours)		29.16	0.920	0.033	28.51	0.912	0.035	24.12	0.868	0.065

Table 2. Quantitative comparisons on the Waymo [29] dataset. We mainly follow the experimental setup of StreetGS [35] with a higher resolution 1280 × 1920, and the color of each cell shows the best and the second best. "Annotations" means whether the model is assisted by the manual 3D annotations. * denotes the metric only for moving objects.

Model	Annotations	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR* ↑
StreetGS [35]	✓	33.97	0.926	0.227	28.50
4DGF [5]	✓	34.64	0.940	0.244	29.77
PVG [2]		29.54	0.895	0.266	21.56
EmerNeRF [36]		31.32	0.881	0.301	21.80
Grid4D [34]		32.19	0.921	0.253	22.77
AD-GS (Ours)		33.91	0.927	0.228	27.41

Table 3. Quantitative comparisons on the nuScenes [1] dataset. We select six sequences with the resolution of 900 × 1600. The color of each cell shows the best and the second best.

Model	PSNR ↑	SSIM ↑	LPIPS ↓
EmerNeRF [36]	27.17	0.853	0.268
PVG [2]	29.49	0.900	0.211
Grid4D [34]	30.29	0.920	0.172
AD-GS (Ours)	31.06	0.925	0.164

Experiments

Experiments

Table 4. Loss ablation on the KITTI [7] dataset. The color of each cell shows the best and the second best.

obj&sky	flow&depth	reg	PSNR ↑	SSIM ↑	LPIPS ↓
			26.52	0.896	0.053
✓			26.98	0.902	0.048
✓	✓		28.03	0.910	0.042
✓	✓	✓	29.16	0.920	0.033

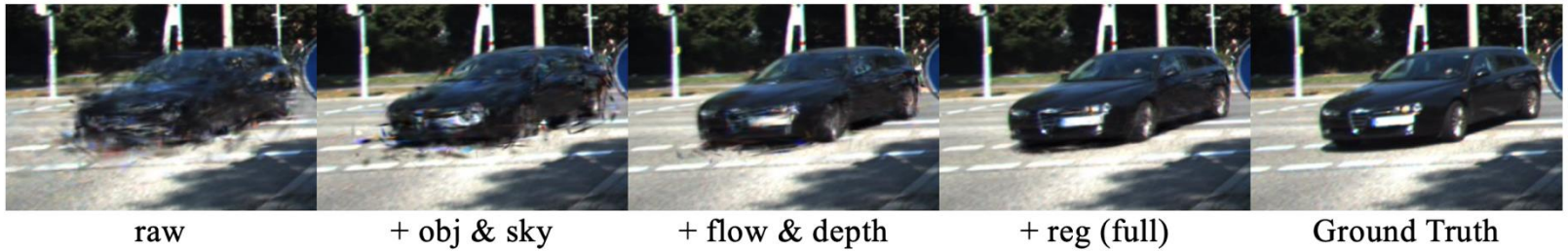


Figure 7. Visualization of loss ablation on the KITTI [7] dataset by gradually adding the losses.

Table 5. Object modeling module ablation on the Waymo [29] dataset. The color of each cell shows the best and the second best. * denotes the metric only for moving objects.

sin&cos	B-spline	t-mask	PSNR* ↑	PSNR ↑	SSIM ↑	LPIPS ↓
	✓		24.28	32.61	0.922	0.234
✓			25.70	33.38	0.925	0.232
✓	✓		26.65	33.65	0.926	0.231
✓	✓	✓	27.41	33.91	0.927	0.228

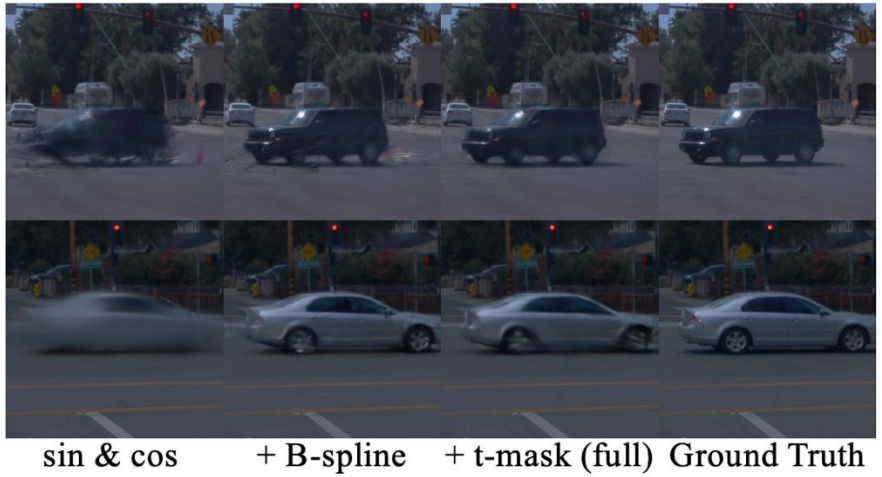


Figure 8. Visualization of object modeling module ablation on the Waymo [29] dataset by gradually adding the modules.

More details and experiments

Project page: https://jiaweixu8.github.io/AD-GS-web/

References

[1] Yan et al. Street Gaussians: Modeling Dynamic Urban Scenes with Gaussian Splatting. European Conference on Computer Vision. 2024.

[2] Fischer et al. Dynamic 3D Gaussian Fields for Urban Areas. The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024.

[3] Yang et al. EmerNeRF: Emergent Spatial-Temporal Scene Decomposition via Self-Supervision. International Conference on Learning Representations, 2024

[4] Oquab et al. DinoV2: Learning Robust Visual Features Without Supervision. arXiv preprint arXiv:2304.07193, 2023.

[5] Chen et al. Periodic Vibration Gaussian: Dynamic Urban Scene Reconstruction and Real-time Rendering arXiv preprint arXiv:2106.13228, 2021.

[6] Kim et al. A General Construction Scheme for Unit Quaternion Curves with Simple High Order Derivatives. Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. 1995.

[7] Qin et al. General Matrix Representations for B-Splines. Sixth Pacific Conference on Computer Graphics and Applications, 1998.

[8] Yu et al. MonoSDF: Exploring Monocular Geometric Cues for Neural Implicit Surface Reconstruction. NeurIPS 2022.

[9] Kerbl et al. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. ACM Transactions on Graphics (TOG), 2024.