Leveraging 2D Priors and SDF Guidance for Dynamic Urban Scene Rendering

Supplementary Material

1. Losses for SDF Network

RGB and Depth Loss.

The per-ray rendering losses for RGB and depth are defined as follows:

$$\ell_{rgb}^r = \|\mathbf{c}_r^t - \hat{\mathbf{c}}_r^t\|, \quad \ell_d^r = |d_r^t - \hat{d}_r^t| \tag{1}$$

where:

- $\mathbf{c}_r^t$ is the observed RGB value of ray r in the image.
- d_r^t is the observed depth value of ray r in the corresponding depth map.
- $\hat{\mathbf{c}}_r^t$ is the predicted RGB value for ray r.
- $\hat{d}_r^t$ is the predicted depth value for ray r obtained from sampling.

SDF supervision. Following prior works [1, 10, 16], we approximate the ground truth signed distance function (SDF) value using the distance to the observed depth along the ray direction $\mathbf{d}_r^t$. Specifically, we define the bound as

$$b_r(\mathbf{x}_i^t) = d_r^t - d(\mathbf{x}_i^t),$$

where d_r^t is the observed depth along ray r, and $d(\mathbf{x}_i^t)$ represents the depth of the sampled point $\mathbf{x}_i^t$.

Using this bound, we divide the sampled points into two disjoint sets:

- **Near-surface points:** $S_{tr}^r = \{\mathbf{x}_i^t \mid b_r(\mathbf{x}_i^t) \leq \epsilon\}$, where ϵ is a truncation threshold that determines proximity to the surface.
- **Free-space points:** $S_{fs}^r = \{\mathbf{x}_i^t \mid b_r(\mathbf{x}_i^t) > \epsilon\}$, which are points far from the surface.

For the set of near-surface points S_{tr}^r, we define the following SDF loss to encourage accurate SDF predictions near the surface:

$$\mathcal{L}_{sdf}^r = \frac{1}{|S_{tr}^r|} \sum_{\mathbf{x}_s \in S_{tr}^r} \left|\varphi(\mathbf{x}_s^t) - b_r(\mathbf{x}_s^t)\right|, \tag{2}$$

where $\varphi(\mathbf{x}_s^t)$ is the predicted SDF value at point $\mathbf{x}_s^t$.

For the set of free-space points S_{fs}^r, we apply a free-space loss similar to [10, 16] to encourage free-space prediction and provide more direct supervision than the rendering terms in Eq. (1):

$$\mathcal{L}_{fs}^r = \frac{1}{|S_{fs}^r|} \sum_{\mathbf{x}_s \in S_{fs}} \max\left(0, e^{-\alpha\varphi(\mathbf{x}_s^t)} - 1, \varphi(\mathbf{x}_s^t) - b_r(\mathbf{x}_s^t)\right). \tag{3}$$

This loss applies:

- An exponential penalty for negative SDF values.
- A linear penalty for positive SDF values exceeding the bound.
- No penalty when the SDF value is within the bound.

SDF regularization. To ensure valid SDF values, particularly in regions without direct supervision, we incorporate the Eikonal regularization term ℓ_{eik}, which promotes a uniform gradient norm for the SDF, encouraging it to grow smoothly away from the surface [5, 10, 17]. Specifically, for any query point $\mathbf{x}_i'^t$ in the canonical space $\mathbb{R}^3$, the gradient of the SDF with respect to the 3D point is encouraged to have unit length:

$$\mathcal{L}_{eik}^r = \frac{1}{|S_{fs}^r|} \sum_{\mathbf{x}_s \in S_{fs}} \left(1 - \|\nabla\varphi(\mathbf{x}_s'^t)\|\right)^2. \tag{4}$$

Surface smoothness regularization. To enhance surface smoothness, we enforce nearby points to have similar normals. Unlike [16], which samples uniformly within a grid, we sample only surface points $\mathbf{x}_s^t \in S_{surf}$, significantly reducing computation. The smoothness loss is defined as:

$$\mathcal{L}_{sm} = \frac{1}{R} \sum_{\mathbf{x}_s \in S_{surf}} \|\nabla\varphi(\mathbf{x}_s^t) - \nabla\varphi(\mathbf{x}_s^t + \delta)\|^2, \tag{5}$$

where $\mathbf{x}_s^t$ is back-projected using depth maps, δ is a small perturbation sampled from a Gaussian distribution with standard deviation δ_{std}, and R is the total number of sampled rays.

The RGB rendering loss $\mathcal{L}_{rgb}$ measures the difference between ground truth and predicted ray colors, while the depth rendering loss $\mathcal{L}_d$ evaluates the depth error over valid rays R_d. Both losses utilize the object mask M_r to focus on the object of interest:

$$\mathcal{L}_{rgb} = \frac{1}{|R_{rgb}|} \sum_{r \in R_{rgb}} M_r^t \ell_{rgb}^r, \tag{6}$$

$$\mathcal{L}_d = \frac{1}{|R_d|} \sum_{r \in R_d} M_r^t \ell_d^r. \tag{7}$$

The SDF loss $\mathcal{L}_{sdf}$ is applied to points in the truncation region S_{tr}:

$$\mathcal{L}_{sdf} = \frac{1}{|R_d|} \sum_{r=1}^{R} \mathcal{L}_{sdf}^r. \tag{8}$$

The free-space loss $\mathcal{L}_{fs}$ and Eikonal loss $\mathcal{L}_{eik}$ are applied to the remaining points S_{fs}:

$$\mathcal{L}_{fs} = \frac{1}{|R_d|} \sum_{r=1}^{R} \mathcal{L}_{fs}^r, \tag{9}$$

16	21	22	25	31	34	35	49
53	80	84	86	89	94	96	102
111	222	323	382	402	427	438	546
581	592	620	640	700	754	795	796

Table 1. Scene IDs of 32 dynamic scenes from the NOTR [20] Dataset which is a subset of the Waymo dataset used for evaluation.

2011_09_26_drive_0005 (City)	2011_09_26_drive_0009 (City)
2011_09_26_drive_0011 (City)	2011_09_26_drive_0013 (City)
2011_09_26_drive_0014 (City)	2011_09_26_drive_0015 (Road)
2011_09_26_drive_0018 (City)	2011_09_26_drive_0022 (Residential)
2011_09_26_drive_0032 (Road)	2011_09_26_drive_0036 (Residential)
2011_09_26_drive_0056 (City)	2011_09_26_drive_0059 (City)
2011_09_26_drive_0060 (City)	2011_09_26_drive_0091 (City)

Table 2. KITTI raw sequences.

$$\mathcal{L}_{eik} = \frac{1}{|R_d|} \sum_{r=1}^{R} \mathcal{L}_{eik}^{r}. \tag{10}$$

2. Dataset Details

We evaluate our method on the NOTR Dataset [**?**], which uses sequences from the Waymo Open Dataset [15]. The scene IDs used in our experiments are listed in Table 1.

We also evaluate on the KITTI MOT sequences for which 3D tracklets are available, as the other two baselines ([2, 3]) utilize these tracklets. The specific sequence IDs used for this evaluation can be seen in Table 2

3. Runtime-Analysis

We show the runtime analysis for our method relative to other methods in Tab. 3. Our method is competitive with other methods both in terms of frame rate and training time. This is because the background Gaussians take up the most amount of time for the rendering operation.

Training for our method takes 3-5 hours for a single sequence. 60% of the time is typically taken to train the SDF networks, 5% for initialization, with the remaining 35% by rasterization approximately. At inference time, our method runs at about 20 fps, which is similar to 4DGF [3] and S3Gaussians [6].

Method	Ours	OmniRe [2]	S3Gaussians [6]	4DGF [4]	StreetGS [19]
Train Time	3-5	3-5	8-10	3-5	1-2
Frame Rate	20	24	20	20	68

Table 3. Train time (in hrs) and frame rate (in fps) comparison for our method.

3D BBox Type	PSNR	SSIM	LPIPS
GT	31.34	0.945	0.026
Ours	30.55	0.931	0.028
[11]	30.67	0.943	0.035

Table 4. Comparison of GT Bounding Boxes (GT), bounding boxes predicted from our tracking method (Ours) and from [11].

4. Tracking Evaluation

To evaluate the tracking off the combination of the depth network UniDepth [12] and the point tracker CoTracker V3 [7], we performing a tracking evaluation. To do so, we compare the rendering results of using bounding boxes derived from our tracking methodology, to that of [11] and the ground truth on the KITTI MOT dataset. Remarkably the combination of point tracking and depth yields only slightly inferior results to that of GT bounding boxes while being almost identical to [11] which makes an assumption of object rigidity.

5. Additional Results

5.1. Image and Depth Rendering Results

Figure 1 shows the results of rendered scenes for both depth and images with moving objects for StreetGaussians, 4DGF and our method without LiDAR inputs. Using point tracking and depth only, our network is able to preserve the details of moving objects in greater detail than that of methods that make use of LiDAR.

5.2. Scene Editing Results

Figure 3 shows some scene editing results on different video sequences. We are able to add and remove both rigid and non-rigid objects from the scene. (b), (d) and (e) add a car, pedestrian and car respectively. (a), (c) and (f) remove instances.

5.3. Results on IPhone Dataset

As our method is not restricted to urban scene datasets but can work on more casually captured datasets, we show qualitative results on the IPhone Dataset. We compare the results to Shape Of Motion [18], Dynamic Gaussian Marbles [14].

6. Implementation Details

Initialization For the background model, we follow OmniRe [2], combining LiDAR points with 4×10^5 random samples, which are divided into 2×10^5 near samples uniformly distributed by distance to the scene's origin and 2×10^5 far samples uniformly distributed by inverse distance. To initialize the background, we filter out the LiDAR samples of dynamic objects. For canonicalization

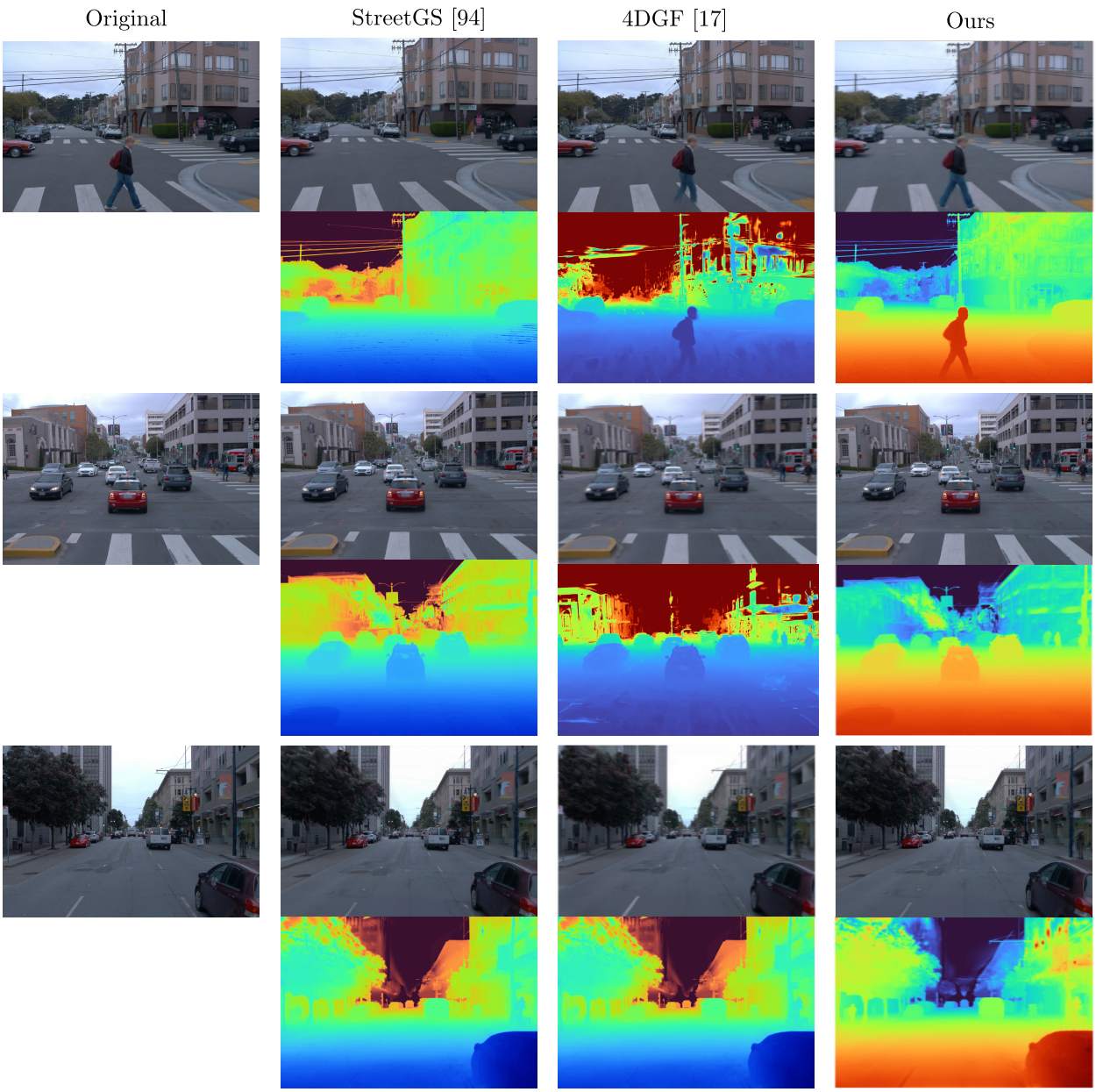

Figure 1. **Image and Depth Rendering Results for the NOTR Dataset** Our method is rendered without LiDAR and is compared to StreetGS and 4DGF. Even though the rendered images look similar, the depth achieved varies by mehod. Our method is able to capture the details (feet of the pedestrian) and smoothness of moving objects (cars) with greater accuracy. StreetGS cannot model pedestrians, hence it fails to render in the top strip. *Citation numbers in the figure correspond to the main paper.*

around dynamic objects, we use the depth map estimated from UniDepth to calculate a bounding box around the object. We use the 2D tracks generated from [7] to warp lidar and depth information from neighboring frames into the initialization frame, typically chosen as the frame where the object is initially detected via SAM2 [13].

Optimization Our 3DGS pipeline trains for $30,000$ iterations with all scene nodes optimized jointly. The learning rate for Gaussian properties aligns with the default settings of 3DGS [8]. Instead of using spherical harmonics, we just use a constant color value for the Gaussians. For the SDF Network, for the initialization, we train the network for 2000 iterations. We train at the lowest-resolution for the first 500 iterations, adding an additional resolution every 200 iterations during the initialization. Subsequently, we train the iteration a 1000 iterations every 2000 training

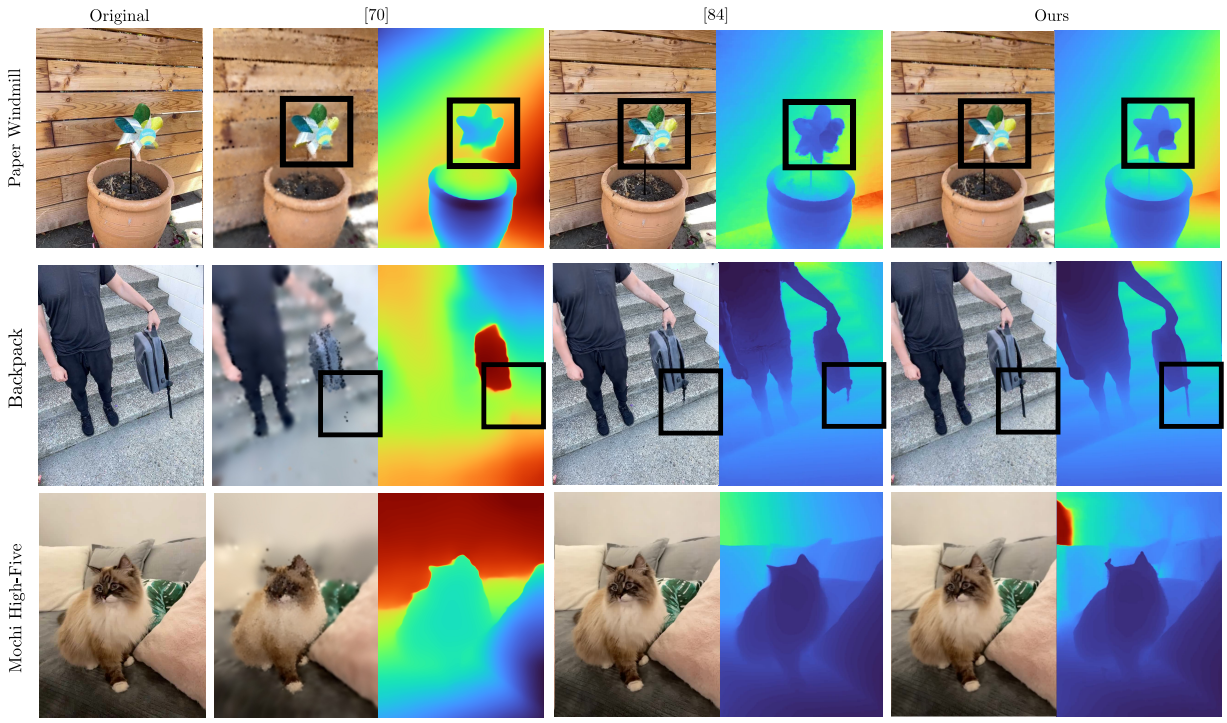

Figure 2. **Qualitative Results on the IPhone Dataset** We show the rendering results of Dynamic Gaussian Marbles [14], Shape of Motion [18] and our method on 3 sequences of the IPhone Dataset. The bounding boxes highlight regions where our method generates a more high-fidelity rendering of the scene. *Citation numbers in the figure correspond to the main paper.*

Figure 3. **Scene Editing** We show original and edited pairs of images, with the region of interest highlighted in a green bounding box. (a) and (c) show the white car and van removed from the scene respectively. (b), (d) and (e) show duplicated cars and pedestrians in the scenes. (f) shows the rendered scene after removing all moving objects and the sky. Videos for the edited scenes are in the ***supp. video***.

iterations for the Gaussians. We train the SDF network using the Adam optimizer [9] with a learning rate 5×10^{-4}. As mentioned in the main paper training alternates between the SDF network and the Gaussians and is done progressively.

We employ step-based weighting for the RGB, depth, and regularization losses, prioritizing RGB and regularization losses early in training and gradually reducing their weights as training progresses. To begin with, we randomly sample an image and select 1024 rays per batch, sampling 128 points along each ray. Subsequently, we take feedback from the Gaussian splatting to direct the ray sampling improving upon random ray sampling. Overall optimization time for our is around 3-4 hours per scene.

Hyper-Parameters:

- **SDF Guidance for Gaussians** $\tau_s = 0.01$, $\tau_n = 0.02$, τ_{pr}=0.02
- **Gaussian Guidance for SDFs**: $\gamma = 3$

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6290–6301, 2022. 1

[2] 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. 2

[3] Tobias Fischer, Jonas Kulhanek, Samuel Rota Bulò, Lorenzo Porzi, Marc Pollefeys, and Peter Kontschieder. Dynamic 3d gaussian fields for urban areas. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 2

[4] Tobias Fischer, Lorenzo Porzi, Samuel Rota Bulo, Marc Pollefeys, and Peter Kontschieder. Multi-level neural scene graphs for dynamic urban environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21125–21135, 2024. 2

[5] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning*, pages 3789–3799. PMLR, 2020. 1

[6] Nan Huang, Xiaobao Wei, Wenzhao Zheng, Pengju An, Ming Lu, Wei Zhan, Masayoshi Tomizuka, Kurt Keutzer, and Shanghang Zhang. S3 gaussians: Self-supervised street gaussians for autonomous driving. *arXiv preprint arXiv:2405.20323*, 2024. 2

[7] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *arXiv preprint arXiv:2307.07635*, 2023. 2, 3

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 3

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[10] Joseph Ortiz, Alexander Clegg, Jing Dong, Edgar Sucar, David Novotny, Michael Zollhoefer, and Mustafa Mukadam. isdf: Real-time neural signed distance fields for robot perception. In *Robotics: Science and Systems*, 2022. 1

[11] Ziqi Pang, Zhichao Li, and Naiyan Wang. Simpletrack: Understanding and rethinking 3d multi-object tracking. In *European Conference on Computer Vision*, pages 680–696. Springer, 2022. 2

[12] Luigi Piccinelli, Yung-Hsu Yang, Christos Sakaridis, Mattia Segu, Siyuan Li, Luc Van Gool, and Fisher Yu. Unidepth: Universal monocular metric depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10106–10116, 2024. 2

[13] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. 3

[14] Colton Stearns, Adam W Harley, Mikaela Uy, Florian Dubost, Federico Tombari, Gordon Wetzstein, and Leonidas Guibas. Dynamic gaussian marbles for novel view synthesis of casual monocular videos. In *SIGGRAPH Asia 2024 Conference Papers*, pages 1–11, 2024. 2, 4

[15] 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. 2

[16] Jingwen Wang, Tymoteusz Bleja, and Lourdes Agapito. Go-surf: Neural feature grid optimization for fast, high-fidelity rgb-d surface reconstruction. In *2022 International Conference on 3D Vision (3DV)*. IEEE, 2022. 1

[17] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems*, pages 27171–27183. Curran Associates, Inc., 2021. 1

[18] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4d reconstruction from a single video. 2024. 2, 4

[19] Yunzhi Yan, Haotong Lin, Chenxu Zhou, Weijie Wang, Haiyang Sun, Kun Zhan, Xianpeng Lang, Xiaowei Zhou, and Sida Peng. Street gaussians: Modeling dynamic urban scenes with gaussian splatting. In *European Conference on Computer Vision*, pages 156–173. Springer, 2024. 2

[20] Jiawei Yang, Boris Ivanovic, Or Litany, Xinshuo Weng, Seung Wook Kim, Boyi Li, Tong Che, Danfei Xu, Sanja Fidler, Marco Pavone, et al. Emernerf: Emergent spatial-temporal scene decomposition via self-supervision. *arXiv preprint arXiv:2311.02077*, 2023. 2