Supplementary material: Dynamic LiDAR Re-simulation using Compositional Neural Fields

Hanfeng Wu[1,2] Xingxing Zuo[2,*] Stefan Leutenegger[2]
Or Litany[3,4] Konrad Schindler[1] Shengyu Huang[1,*]
[1] ETH Zurich [2] TU Munich [3] Technion [4] NVIDIA

In this supplementary material, we first provide additional information about the datasets for our evaluations and implementation details of our proposed method in Sec. A. Next, we present more qualitative and quantitative results in Sec. B. Please also check the supplemental video for more results showcasing our performance. Finally, we provide the complete derivations of the SDF-based volume rendering for active sensor in Sec. C.

A. Datasets and implementation details

A.1. Datasets

Waymo Dynamic. For the *Waymo Dynamic* dataset, we take them from 4 scenes of *Waymo Open Dataset* [5]. There are multiple moving vehicles inside each scene. 50 consecutive frames are taken from each scene for our evaluation. The vehicles are deemed as *dynamic* if the speed is > 1 m/s. in any of the 50 frames. The corresponding scene IDs on *Waymo Open Dataset* for our selected scenes are shown as follows:

	Scene ID
Scene 1	1083056852838271990_4080_000_4100_000
Scene 2	13271285919570645382_5320_000_5340_000
Scene 3	10072140764565668044_4060_000_4080_000
Scene 4	10500357041547037089_1474_800_1494_800

Waymo Dynamic NVS. For the *Waymo Dynamic NVS* dataset, we use the same 4 scenes as chosen in *Waymo Dynamic*. We change the evaluation paradigm similar to *Waymo NVS* [1] such that we first train the model on all 50 consecutive LiDAR frames then we synthesize 50 novel LiDAR frames with a sensor shift of 2 meters. We then train a new model on the new 50 synthetic LiDAR scans and evaluate against the original 50 LiDAR scans.

A.2. Implementation details

DynNFL. Our model is implemented based on nerfstudio[7]. For the static neural field, we sample $N_s = 512$ points in total, with $N_u = 256$ uniformly sampled points and $N_i = 256$ weighted sampled points with 8 upsample steps. In each upsample step, 32 points are sampled based on the weight distribution of the previously sampled points. For each dynamic neural field, we sample $N_s = 128$ points in total, with $N_u = 64$ uniformly sampled points and $N_i = 64$ weighted sampled points with 4 upsample steps. During training, we minimize the loss function using the Adam [2] optimiser, with an initial learning rate of 0.005. It linearly decays to 0.0005 towards the end of training. For the loss weights, we use $w_\zeta = 3, w_e = 50, w_{\text{drop}} = 0.15, w_s = 1$, and $w_{\text{eik}} = 0.3$. The batch size is 4096 and we train the model for 60000 iterations on a single RTX3090 GPU with float32 precision.

LiDARsim. We re-implement the LiDARsim [3] as one of our baselines. First, we estimated point-wise normal vectors by considering all points within a 20 cm radius ball within the training set. Following this, we applied voxel down-sampling [8], employing a 4 cm voxel size to reconstruct individual disk surfels at each point. The surfel orientation is defined based on the estimated normal vector. During inference, we apply the ray-surfel intersections test to determine the intersection points, thus the range and intensity values. We select a fixed surfel radius of 6 cm for the *Waymo* dataset and 12 cm for the *Town* dataset. To handle dynamic vehicles, we follow LiDARsim [3] by aggregating the LiDAR points for each vehicle from all the training frames and representing them in the *canonical* frame of each vehicle. During inference, we transform all the aggregated vehicle points from their *canonical* frames to the world frame and run ray-surfel intersection.

UniSim. We re-implement UniSim's [10] rendering process for LiDAR measurements by replacing our ray-drop test-based neural fields composition method with its joint

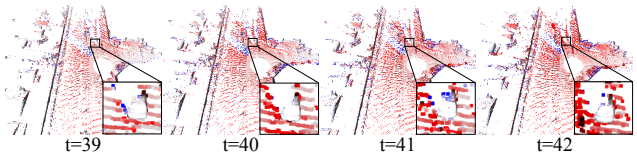

Figure 1. Qualitative results of LiDAR future frame simulation.

Method	MAE ↓	MedAE ↓	CD ↓	MedAE Dyn ↓	Intensity RMSE ↓
LiDARsim [3]	448.4	55.1	77.0	38.7	0.13
Unisim [10]	115.1	9.7	33.5	24.3	0.19
Ours	**72.9**	**3.8**	**22.9**	**14.0**	**0.07**

Table 1. Evaluation of LiDAR NVS on *Waymo Dynamic NVS*.

rendering method. For every ray $\mathbf{r}(\mathbf{o}, \mathbf{d})$, we begin by conducting an intersection test with all dynamic bounding boxes in the scene to identify the near and far limits. We then uniformly sample 512 points along each ray, assigning each point to either a dynamic neural field, if it falls within a dynamic bounding box, or to the static neural field otherwise. After sampling, we query the SDF and intensity values from the relevant neural fields. Finally, using the SDF-based volume rendering formula in Eq. 41 for active sensors, we calculate the weights and perform the rendering. Note that we use the same neural field architecture as in our method.

B. Additional results

B.1. Waymo Dynamic NVS evaluation

To demonstrate the robustness of our method, we extend the evaluation paradigm to not only focus on interpolation performance. We incorporate Waymo NVS evaluation introduced in Sec. A to focus on close-loop novel view synthesis performance. As illustrated in Tab. 1, our method outperforms LiDARsim and Unisim in all aspects.

B.2. Future frame generation

We trained DyNFL using the initial 40 frames and assessed its performance against the last 10 frames. The results are presented on the *Waymo Dynamic* dataset in Tab. 2 and Fig. 1. Unsurprisingly, the performance is comparatively inferior to the original setting (*cf*. Tab. 1), as it requires extrapolation beyond the observed environment, and thus again a (possibly learned) scene prior. Nevertheless DyNFL continues to outperform LiDARsim. The degradation on dynamic vehicles is marginal, attributable to our precise pose interpolation and high-quality asset reconstruction. We will incorporate these findings in the final version.

Method	MAE ↓	MedAE ↓	CD ↓	MedAE Dyn ↓
LiDARsim	333.3	25.3	67.8	13.0
Ours	**81.8**	**8.6**	**26.4**	**9.3**

Table 2. Results of future frame simulation.

B.3. Runtime analysis

DyNFL training takes ≈7 hours on average on a single RTX 3090 GPU with fp16 precision and 16 hours with fp32 previsions, inference takes 2.2 seconds per LiDAR scan using fp16 precision and 7 seconds using fp32 precision. The envisioned offline use for counterfactual re-simulation prioritizes realism over efficiency. Runtime can potentially be improved for high-throughput applications by reducing rendering complexity.

B.4. More qualitative results

In this section, we provide more qualitative results. In Fig. 2, we showcase the 4 scenes from *Waymo dynamic* dataset. We show additional scene editing results in Fig. 3. Please check the supplementary videos for more qualitative results.

C. SDF-based LiDAR volume rendering

In this section, we start by introducing the preliminary of NeRF [4] following terminology as described in [6]. Then we provide the full derivation of the SDF-based volume rendering for active sensor.

C.1. Preliminary

Density. For a ray emitted from the origin $\mathbf{o} \in \mathbb{R}^3$ towards direction $\mathbf{d} \in \mathbb{R}^3$, the *density* σ_ζ at range ζ indicates the likelihood of light interacting with particles at that point $\mathbf{r}_\zeta = \mathbf{o} + \zeta\mathbf{d}$. This interaction can include absorption or scattering of light. In passive sensing, density σ is a critical factor in determining how much light from the scene's illumination is likely to reach the sensor after passing through the medium.

Transmittance quantifies the likelihood of light traveling through a given portion of the medium without being scattered or absorbed. Density is closely tied to the transmittance function $\mathcal{T}(\zeta)$, which indicates the probability of a ray traveling over the interval $[0, \zeta)$ without hitting any particles. Then the probability $\mathcal{T}(\zeta+d\zeta)$ of *not* hitting a particle when taking a differential step $d\zeta$ is equal to $\mathcal{T}(\zeta)$, the likelihood of the ray reaching ζ, times $(1 - d\zeta \cdot \sigma(\zeta))$, the probability of not hitting anything during the step:

$$\mathcal{T}(\zeta + d\zeta) = \mathcal{T}(\zeta) \cdot (1 - d\zeta \cdot \sigma(\zeta)) \quad (1)$$

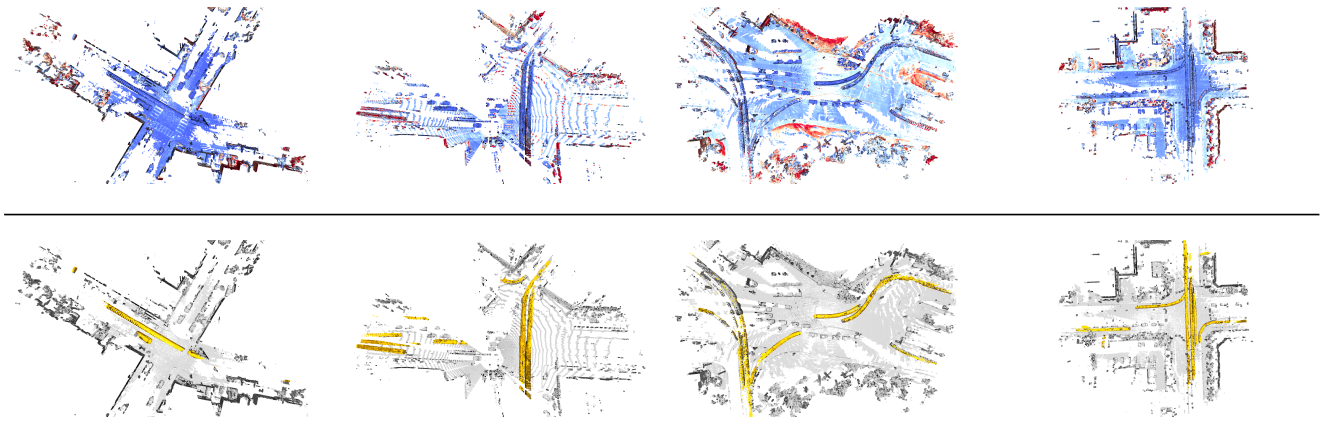

Figure 2. Visualization of 4 selected scenes from *Waymo Dynamic* dataset. For each scene, we aggregate 50 frames. In the first row, points are color-coded by the intensity values(0 0.25). In the second row, dynamic vehicles are painted as yellow.

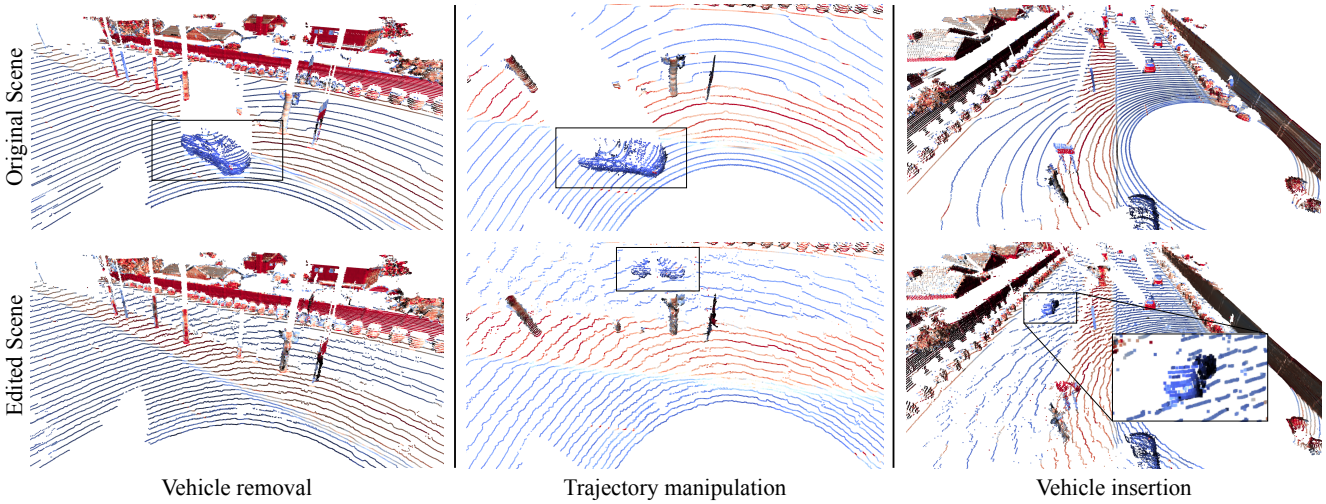

Figure 3. Visualization of scene editing capabilities. We showcase 3 kinds of scene editing capabilities including vehicle removal(left), trajectory manipulation(middle) and vehicle insertion(right). The first row represents the original scenes, the second row demonstrates the scenes after editing. All points are color-coded by the intensity values(0 0.25).

$$\frac{\mathcal{T}(\zeta + d\zeta) - \mathcal{T}(\zeta)}{d\zeta} \equiv \mathcal{T}'(\zeta) = -\mathcal{T}(\zeta) \cdot \sigma(\zeta) \ . \quad (2)$$

We solve the differential equation as follows:

$$\mathcal{T}'(\zeta) = -\mathcal{T}(\zeta) \cdot \sigma(\zeta) \quad (3)$$

$$\frac{\mathcal{T}'(\zeta)}{\mathcal{T}(\zeta)} = -\sigma(\zeta) \quad (4)$$

$$\int_a^b \frac{\mathcal{T}'(\zeta)}{\mathcal{T}(\zeta)} \, d\zeta = -\int_a^b \sigma(\zeta) \, d\zeta \quad (5)$$

$$\log \mathcal{T}(\zeta)|_a^b = -\int_a^b \sigma(\zeta) \, d\zeta \quad (6)$$

$$\mathcal{T}(a \to b) \equiv \frac{\mathcal{T}(b)}{\mathcal{T}(a)} = \exp\left(-\int_a^b \sigma(\zeta) \, d\zeta\right) \ . \quad (7)$$

Hence, for a ray segment between ζ_0 and ζ, transmittance is given by:

$$\mathcal{T}_{\zeta_0 \to \zeta} \equiv \frac{\mathcal{T}_\zeta}{\mathcal{T}_{\zeta_0}} = exp(-\int_{\zeta_0}^{\zeta} \sigma_t dt) \ , \quad (8)$$

which leads to following factorization of the transmittance:

$$\mathcal{T}_\zeta = \mathcal{T}_{0 \to \zeta_0} \cdot \mathcal{T}_{\zeta_0 \to \zeta} \ . \quad (9)$$

Opacity is the complement of transmittance and represents the fraction of light that is either absorbed or scattered in the medium. In a homogeneous medium with constant density σ the opacity for a segment $[\zeta_j, \zeta_{j+1}]$ of length $\Delta\zeta$ is given by $\alpha_{\zeta_j} = 1 - exp(-\sigma \cdot \Delta\zeta)$

C.2. SDF-based volume rendering for active sensor

NeuS[9] derives the opaque density based on the SDF which is:

$$\begin{aligned}\sigma_{\zeta_i} &= \max\left(\frac{-\frac{d\Phi_s}{d\zeta_i}(f(\zeta_i))}{\Phi_s(f(\zeta_i))}, 0\right) \\ &= \max\left(\frac{-(\nabla f(\zeta_i)\cdot \mathbf{v})\phi_s(f(\zeta_i))}{\Phi_s(f(\zeta_i))}, 0\right) ,\end{aligned} \tag{10}$$

where Φ_s represents the Sigmoid function, f is the SDF function that maps a range ζ to the SDF value of the point position $\mathbf{o} + \mathbf{d} * \zeta$. Note that the integral term is computed by

$$\int \frac{-(\nabla f(\zeta)\cdot \mathbf{v})\phi_s(f(\zeta))}{\Phi_s(f(\zeta))} d\zeta = -\ln(\Phi_s(f(\zeta))) + C \, . \tag{11}$$

We extend the density-based volume rendering for active sensor to SDF-based. Starting from the passive SDF-based volume rendering [9], We substitute the density $\tilde{\sigma}$ with opaque density in 10 and evaluate the radiant power integrated from ray segment [a,b] with constant reflectivity ρ_a.

Consider the case where $-(\nabla f(\zeta)\cdot \mathbf{v}) > 0$ within the ray segment $[a, b]$. We have

$$P(a \to b) = \int_a^b \mathcal{T}^2(a \to t)\cdot \tilde{\sigma}_t \cdot \rho(t)\, dt \tag{12}$$

$$= \rho_a \int_a^b \mathcal{T}^2(a \to t)\cdot \tilde{\sigma}_t \, dt \tag{13}$$

$$= \rho_a \int_a^b \exp\left(-\int_a^t 2\tilde{\sigma}(u)\, du\right)\cdot \tilde{\sigma}_t \, dt \tag{14}$$

$$= \rho_a \int_a^b \exp\left(-2\int_a^t \tilde{\sigma}(u)\, du\right)\cdot \tilde{\sigma}_t \, dt \tag{15}$$

$$= \rho_a \int_a^b \exp\left(2\ln(\Phi_s(f(u)))\Big|_a^t\right)\cdot \tilde{\sigma}_t \, dt \, . \tag{16}$$

Let $\Omega_x = \Phi_s(f(x))$, then

$$P(a \to b) = \rho_a \int_a^b \exp\left(2\ln(\Omega_t) - 2\ln(\Omega_a)\right)\cdot \tilde{\sigma}_t \, dt \tag{17}$$

$$= \rho_a \int_a^b \frac{{\Omega_t}^2}{{\Omega_a}^2}\cdot \tilde{\sigma}_t \, dt \tag{18}$$

$$= \frac{\rho_a}{{\Omega_a}^2}\int_a^b {\Omega_t}^2 \cdot \tilde{\sigma}_t \, dt \tag{19}$$

$$= \frac{\rho_a}{{\Omega_a}^2}\int_a^b -\frac{d\Phi_s}{dt}(f(t))\cdot \Phi_s(f(t))\, dt \tag{20}$$

$$= \frac{\rho_a}{{\Omega_a}^2}\left(-\frac{1}{2}\Phi_s(f(t))^2\Big|_a^b\right) \tag{21}$$

$$= \frac{\rho_a}{{\Omega_a}^2}\left(\frac{1}{2}\Phi_s(f(a))^2 - \frac{1}{2}\Phi_s(f(b))^2\right) \tag{22}$$

$$= \frac{\Phi_s(f(a))^2 - \Phi_s(f(b))^2}{2\Phi_s(f(a))^2}\cdot \rho_a \, . \tag{23}$$

Consider the case where $-(\nabla f(\zeta)\cdot \mathbf{v}) < 0$ within the ray segment $[a, b]$. Then

$$P(a \to b) = \int_a^b \mathcal{T}^2(a \to t)\cdot \tilde{\sigma}_t \cdot \rho(t)\, dt \tag{24}$$

$$= \int_a^b \mathcal{T}^2(a \to t)\cdot 0 \cdot \rho(t)\, dt \tag{25}$$

$$= 0 \, . \tag{26}$$

Hence we conclude

$$P(a \to b) = \max\left(\frac{\Phi_s(f(a))^2 - \Phi_s(f(b))^2}{2\Phi_s(f(a))^2}, 0\right)\cdot \rho_a \, . \tag{27}$$

Volume rendering of piecewise constant data. Combining the above, we can evaluate the volume rendering integral through a medium with piecewise constant reflectivity:

$$P(\zeta_{N+1}) = \sum_{n=1}^{N}\int_{\zeta_n}^{\zeta_{n+1}} \mathcal{T}^2(\zeta)\cdot \tilde{\sigma}_\zeta \cdot \rho_{\zeta_n}\, d\zeta \tag{28}$$

$$= \sum_{n=1}^{N}\int_{\zeta_n}^{\zeta_{n+1}} \mathcal{T}_{\zeta_n}^2 \cdot \mathcal{T}^2(\zeta_n \nrightarrow \zeta)\cdot \tilde{\sigma}_\zeta \cdot \rho_{\zeta_n}\, d\zeta \tag{29}$$

$$= \sum_{n=1}^{N}\mathcal{T}_{\zeta_n}^2 \int_{\zeta_n}^{\zeta_{n+1}} \mathcal{T}^2(\zeta_n \to \zeta)\cdot \tilde{\sigma}_\zeta \cdot \rho_{\zeta_n}\, d\zeta \tag{30}$$

$$= \sum_{n=1}^{N}\mathcal{T}_{\zeta_n}^2 P(\zeta_n \to \zeta_{n+1}) \tag{31}$$

$$= \sum_{n=1}^{N}\mathcal{T}_{\zeta_n}^2 \cdot \tilde{\alpha}_{\zeta_n}\cdot \rho_{\zeta_n}, \tag{32}$$

where

$$\tilde{\alpha}_{\zeta_n} \equiv \max\left(\frac{\Phi_s(f(\zeta_n)^2 - \Phi_s(f(\zeta_{n+1}))^2}{2\Phi_s(f(\zeta_n))^2}, 0\right) \, . \tag{33}$$

The discrete accumulated transmittance $\mathcal{T}$ can be calculated as follows:

Consider the case where $-(\nabla f(\zeta)\cdot \mathbf{v}) > 0$ in $[\zeta_n, \zeta_{n+1}]$:

$$\mathcal{T}_{\zeta_n} = \prod_{i=1}^{n-1}(\exp(-\int_{\zeta_n}^{\zeta_{n+1}} \tilde{\sigma}_\zeta \, d\zeta) \tag{34}$$

$$= \prod_{i=1}^{n-1} \left(\frac{\Phi_s(f(\zeta_{n+1}))}{\Phi_s(f(\zeta_n))}\right) \tag{35}$$

$$\mathcal{T}^2_{\zeta_n} = \prod_{i=1}^{n-1} \left(\frac{\Phi_s(f(\zeta_{n+1}))^2}{\Phi_s(f(\zeta_n))^2}\right) \tag{36}$$

$$= \prod_{i=1}^{n-1} (1 - 2\tilde{\alpha}_{\zeta_n}) \ . \tag{37}$$

Consider the case where $-(\nabla f(\zeta) \cdot \mathbf{v}) < 0$ in $[\zeta_n, \zeta_{n+1}]$:

$$\mathcal{T}_{\zeta_n} = \prod_{i=1}^{n-1} (\exp(-\int_{\zeta_n}^{\zeta_{n+1}} \tilde{\sigma}_\zeta \, d\zeta) = \prod_{i=1}^{n-1} (1) \tag{38}$$

$$\mathcal{T}^2_{\zeta_n} = \prod_{i=1}^{n-1} (1^2) = \prod_{i=1}^{n-1} (1 - 2\tilde{\alpha}_{\zeta_n}) \ . \tag{39}$$

In conclusion, the radiant power can be reformulated as:

$$P(\zeta_{N+1}) = \sum_{n=1}^{N} \mathcal{T}^2_{\zeta_n} \cdot \tilde{\alpha}_{\zeta_n} \cdot \rho_{\zeta_n} \ , \tag{40}$$

where $\mathcal{T}^2_{\zeta_n} = \prod_{i=1}^{n-1} (1 - 2\tilde{\alpha}_{\zeta_i})$.

Depth volume rendering of piecewise constant data Note that $\tilde{\alpha}_{\zeta_n} \in [0, 0.5], \mathcal{T}^2_{\zeta_n} \in [0, 1], \sum_{n=1}^{N} \mathcal{T}^2_{\zeta_n} \cdot \tilde{\alpha}_{\zeta_n} = 0.5$, for depth volumetric rendering, we have

$$\zeta = \sum_{n=1}^{N} 2 \cdot \mathcal{T}^2_{\zeta_n} \cdot \tilde{\alpha}_{\zeta_n} \cdot \zeta_n = \sum_{n=1}^{N} w_n \cdot \zeta_n \ , \tag{41}$$

where $w_n = 2\tilde{\alpha}_{\zeta_n} \cdot \prod_{i=1}^{n-1} (1 - 2\tilde{\alpha}_{\zeta_i})$.

References

[1] Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *ICCV*, 2023. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[3] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. LiDARsim: Realistic LiDAR simulation by leveraging the real world. In *CVPR*, 2020. 1, 2

[4] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NerF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2

[5] 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 *CVPR*, 2020. 1

[6] Andrea Tagliasacchi and Ben Mildenhall. Volume rendering digest (for nerf). *arXiv preprint arXiv:2209.02417*, 2022. 2

[7] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, 2023. 1

[8] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. Torchsparse: Efficient point cloud inference engine. *Proceedings of Machine Learning and Systems*, 4:302–315, 2022. 1

[9] 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 *NeurIPS*, 2021. 4

[10] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *CVPR*, 2023. 1, 2