ScatterNeRF:
Seeing Through Fog with Physically-Based Inverse Neural Rendering
(Supplementary Material)

Andrea Ramazzina[1] Mario Bijelic[2] Stefanie Walz[1] Alessandro Sanvito[1] Dominik Scheuble[1] Felix Heide[2,3]
[1]Mercedes-Benz [2]Princeton University [3]Algolux

The here presented supplementary material provides additional details for the results shown in the main document. In Section 1, we derive the Koschmieder fog equivalence and the NeRF rendering equation in Section 3.1 of the main document. Section 2 introduces the made approximations used in Section 3.2 of the main document to solve the integral form of the rendering Equation (10) in the main document. Section 3 provides additional training details for ScatterNeRF and further details on training and evaluation of baseline methods. Section 4 outlines network details of ScatterNeRF. Qualitative results for altering fog densities and color can be found in Section 5. In Section 6 and Section 7, we present further qualitative comparisons with baseline methods for scene reconstruction and descattering, respectively. Finally, in Section 8, additional details on the In-the-Wild and controlled environment dataset are shown.

Contents

1. Equivalence of Volumetric Rendering and Koschmieders Model

An overview figure of the neural rendering process is given in Figure 1. Following the volume rendering equation used in [14], the expected color C for each pixel (traversed by a ray ($\mathbf{r}$)) is computed as,

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t))dt \ , \tag{1}$$

Figure 1: Schematic visualization of the neural rendering approach. For a camera position, the scene is integrated along one camera ray $\mathbf{r}$ (gray) at the sampled red positions. Hereby, the interactions along the ray model the scene's appearance in the image.

where,

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right), \tag{2}$$

and $\mathbf{r}_i = [\mathbf{x}_i; \mathbf{d}_i] \in \mathbb{R}^5$ is modeled by the position along the ray $\mathbf{x} \in \mathbb{R}^3$ and direction $\mathbf{d} \in \mathbb{R}^2$.

Assuming the presence of a media with volume density $\sigma_p(\mathbf{r}(t))$ and color $c_p(\mathbf{r}(t))$ from the camera origin until a distance D , we can separate the integral of the expected color in,

$$C(\mathbf{r}) = \int_{t_n}^{D} T(t)\sigma_p(\mathbf{r}(t))c_p(\mathbf{r}(t))dt \; + \int_{D}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt. \tag{3}$$

In an ideal clear-weather setting, it can be assumed for the clear scene C_c to have

$$\sigma_p(\mathbf{r}(t)) = \sigma_p = 0,$$

hence bringing to zero the first defined integral on the right side of equation (3):

$$C_c(\mathbf{r}) = \int_{D}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt. \tag{4}$$

We now consider the presence of participating media in the scene (instead of clear scene) C_F.
Adopting the simplifications used in the Koschmieder model, we assume constant attenuation coefficient $\sigma_p(\mathbf{r}(t)) = \sigma_p$, constant airlight color $c_p(\mathbf{r}(t)) = c_p$ and so we can simplify Equation 3 to,

$$C_F(\mathbf{r}) = \int_{t_n}^{D} T(t)\sigma_p c_p dt \; + \int_{D}^{t_f} T(t)\sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt, \tag{5}$$

$$= \sigma_p c_p \int_{t_n}^{D} \exp\left(-\int_{t_n}^{t} \sigma_p ds\right) dt \; + \int_{D}^{t_f} \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(t))ds\right) \sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt, \tag{6}$$

$$= \sigma_p c_p \int_{t_n}^{D} \exp\left(-\sigma_p \int_{t_n}^{t} ds\right) dt \; + \int_{D}^{t_f} \exp\left(-\int_{t_n}^{D} \sigma_p ds - \int_{D}^{t} \sigma(\mathbf{r}(t))ds\right) \sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt. \tag{7}$$

For simplification, we have the ray starting from the origin, i.e. $t_n = 0$,

$$C_F(\mathbf{r}) = \sigma_p c_p \int_{0}^{D} \exp\left(-\sigma_p t\right) dt \; + \int_{D}^{t_f} \exp\left(-\sigma_p \int_{0}^{D} ds - \int_{D}^{t} \sigma(\mathbf{r}(t))ds\right) \sigma(\mathbf{r}(t))c(\mathbf{r}(t))dt, \tag{8}$$

$$= \sigma_p c_p \left[\frac{\exp(-\sigma_p t)}{-\sigma_p} \right]_0^D + \int_D^{t_f} \exp(-\sigma_p D) \exp\left(- \int_D^t \sigma(\mathbf{r}(t)) ds \right) \sigma(\mathbf{r}(t)) c(\mathbf{r}(t)) dt, \tag{9}$$

$$= -c_p (\exp(-\sigma_p D) - 1) + \exp(-\sigma_p D) \int_D^{t_f} \exp\left(- \int_D^t \sigma(\mathbf{r}(t)) ds \right) \sigma(\mathbf{r}(t)) c(\mathbf{r}(t)) dt. \tag{10}$$

Substituting the last part of the above formulation according to equation (4) we get:

$$C_F(\mathbf{r}) = c_p (1 - \exp(-\sigma_p D)) + \exp(-\sigma_p D) C_c(\mathbf{r}) \tag{11}$$

which is equal to the Koschmieder model in [11],

$$\mathbf{C}_F = l\mathbf{C}_c + (1 - l)\mathbf{C}_p, \tag{12}$$

where c_p is the airlight, the transmittance map is $l = \exp(-\sigma_p D)$ and the fog attenuation coefficient is σ_p.

2. Derivation of the Neural Radiance Model

Starting from Equations (7,8) in the main paper, which were defined as follows,

$$\begin{aligned} \mathbf{C}_F(\mathbf{r}) = \int_{t_n}^{t_f} T_F(t) (&\sigma_p(\mathbf{r}(t)) \mathbf{c}_p(\mathbf{r}(t)) \\ &+ \sigma_c(\mathbf{r}(t)) \mathbf{c}_c(\mathbf{r}(t))) dt, \end{aligned} \tag{13}$$

with,

$$T_F(t) = \exp\left(- \int_{t_n}^t (\sigma_c(\mathbf{r}(s)) + \sigma_p(\mathbf{r}(s))) ds \right), \tag{14}$$

$$T_F(t) = T_p(t) T_c(t), \tag{15}$$

we have to find numerical approximations to solve the integral. Hereby, we define that the scattering volume and scene are separable and additive. Hence the integrated ray in a hazed scene can be written as,

$$\mathbf{C}_F(\mathbf{r}) = \int_{t_n}^{t_f} T_F(t) (\sigma_p(\mathbf{r}(t)) \mathbf{c}_p(\mathbf{r}(t)) + \sigma_c(\mathbf{r}(t)) \mathbf{c}_c(\mathbf{r}(t))) dt, \tag{16}$$

with,

$$T_F(t) = \exp\left(- \int_{t_n}^t (\sigma_c(\mathbf{r}(s)) + \sigma_p(\mathbf{r}(s))) ds \right). \tag{17}$$

To solve the integral, we approximate it as a Riemann sum, considering $\sigma_p(\mathbf{r}(t))$, $\mathbf{c}_p(\mathbf{r}(t))$, $\sigma_c(\mathbf{r}(t))$, and $\mathbf{c}_c(\mathbf{r}(t))$ as a piece-wise constant for small segments $\delta_i = t_{t+1} - t_i$. Therefore, we get $\hat{C}_F$ as,

$$\hat{\mathbf{C}}_F(\mathbf{r}) = \sum_i^N \left(\int_{t_i}^{t_{i+1}} (\sigma_{p_i} \mathbf{c}_{p_i} + \sigma_{c_i} \mathbf{c}_{c_i}) T_F(t) dt \right), \tag{18}$$

$$= \sum_i^N \left((\sigma_{p_i} \mathbf{c}_{p_i} + \sigma_{c_i} \mathbf{c}_{c_i}) \int_{t_i}^{t_{i+1}} T_F(t) dt \right), \tag{19}$$

where $t_n = t_0$ and $t_N = t_f$.
Further simplifying $T_F(t)$ from Eq. (17) we rewrite it by splitting the integral in $t_i \in (t_n, t)$ as,

$$T_F(t) = \exp\left(- \int_{t_n}^{t_i} (\sigma_{p_i}(\mathbf{r}(s)) + \sigma_{c_i}(\mathbf{r}(s))) ds \right) \exp\left(- \int_{t_i}^t (\sigma_{p_i} + \sigma_{c_i}) ds \right), . \tag{20}$$

Then the integral until t_i can be approximated as,

$$T_F(t_i) = \exp\left(-\sum_{j=1}^{i-1}\left(\sigma_p(\mathbf{r}_i) + \sigma_c(\mathbf{r}_i)\right)\delta_i\right), \tag{21}$$

leaving us with,

$$T_F(t) = T_F(t_i)\exp\left(-\int_{t_i}^{t}(\sigma_{p_i} + \sigma_{c_i})ds\right), \tag{22}$$

$$= T_F(t_i)\exp\left(-(\sigma_{p_i} + \sigma_{c_i})(t - t_i)\right). \tag{23}$$

Finally, plugging Eq. (22) in Eq. (18) yields,

$$\hat{\mathbf{C}}_F(\mathbf{r}) = \sum_i^N(\sigma_{p_i}\mathbf{c}_{p_i} + \sigma_{c_i}\mathbf{c}_{c_i})T_{F_i}\int_{t_i}^{t_{i+1}}\exp\left(-(\sigma_{p_i} + \sigma_{c_i})(t - t_i)\right)dt, \tag{24}$$

$$= \sum_i^N \frac{\sigma_{p_i}\mathbf{c}_{p_i} + \sigma_{c_i}\mathbf{c}_{c_i}}{\sigma_{p_i} + \sigma_{c_i}}T_{F_i}\left(\exp\left(-(\sigma_{p_i} + \sigma_{c_i})(t - t_i)\right)\Big|_{t_{i+1}}^{t_i}\right), \tag{25}$$

$$= \sum_i^N \frac{\sigma_{p_i}\mathbf{c}_{p_i} + \sigma_{c_i}\mathbf{c}_{c_i}}{\sigma_{p_i} + \sigma_{c_i}}T_{F_i}(1 - \exp\left(-(\sigma_{p_i} + \sigma_{c_i})\delta_i\right)), \tag{26}$$

which maps to Eq. (12) of the main paper,

$$\mathbf{C}_F(\mathbf{r}) = \sum_i^N w_F(\mathbf{r}(t_i))\mathbf{c}_F(\mathbf{r}(t_i)), \tag{27}$$

with,

$$w_F(\mathbf{r}_i) = T_F(\mathbf{r}_i)(1 - \exp\left(\left(\sigma_p(\mathbf{r}_i) + \sigma_c(\mathbf{r}_i)\right)\delta_j\right), \tag{28}$$

$$T_F(\mathbf{r}_i) = \exp\left(-\sum_{j=1}^{i-1}\left(\sigma_p(\mathbf{r}_i) + \sigma_c(\mathbf{r}_i)\right)\delta_i\right), \tag{29}$$

$$\mathbf{c}_F(\mathbf{r}_i) = \frac{\sigma_c(\mathbf{r}_i)\mathbf{c}_c(\mathbf{r}_i) + \sigma_p(\mathbf{r}_i)\mathbf{c}_p(\mathbf{r}_i)}{\sigma_p(\mathbf{r}_i) + \sigma_c(\mathbf{r}_i)}. \tag{30}$$

3. Additional Training Details

This section contains information regarding the loss and training details.

3.1. Additional Loss Details

The total loss used to train the three MLPs $f_{c_{coarse}}$,$f_{c_{fine}}$,f_p is,

$$L_{tot} = \psi_1 L_{rgbF} + \psi_2 L_A + \psi_3 L_{ec} + \psi_4 L_{eF} + \psi_5 L_{depth}. \tag{31}$$

Here, the chosen weighting loss parameters $\psi_{1,\dots,5}$ are $\psi_1 = 2.5$, $\psi_2 = 0.5$, $\psi_3 = 10^{-3}$, $\psi_4 = 10^{-4}$ and $\psi_5 = 10^{-1}$. The training is performed for 250'000 steps and batch size of 4096 rays, using as optimizer ADAMW [13] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate $5 \cdot 10^{-4}$ and weight decay factor of 10^{-2}. The implementation is based on Pytorch [16], training the model on four NVIDIA RTX A6000.

3.2. Sampling Procedure

To perform volume rendering, NeRFs must be evaluated across many sampled positions for each pixel to approximate best the volumetric rendering integral in Eq. (1) [14]. Since it is reasonable to assume that in clear weather conditions most of the space will contribute negligibly to the final pixel color due to close to null air density, in [14] as well as many subsequent works [3, 4, 26] it is chosen to bias the learning and rendering procedure by first sampling a rough geometry from a coarse network f_{coarse} and use it to guide the selection of an additional set of points closer to the surfaces of the solids in the scene.
This scheme, named hierarchical sampling, usually involves selecting, for each ray, a set of points N_{coarse} with stratified sampling. Then, for each location in N_{coarse}, f_{coarse} provides a density σ_{coarse} used to compute the weights w. The weights are normalized as,

$$\hat{w}_i = \frac{w_i}{\sum_{j=1}^{N_{coarse}} w_j} \tag{32}$$

to obtain a discretized probability distribution, which informs a second round of sampling of a point set N_{fine}. Finally, the pixel color is the result of volumetric rendering over the set of points $(N_{coarse} \cup N_{fine})$ of a second network f_{fine}.

However, as shown in Fig. 2, the presence of a scattering media undermines the assumption behind hierarchical sampling since the scattering media occupies the usually-empty portions of space and contribute to the final pixel color. On the other hand, we are interested in allocating as much of the network f_c capacity as possible toward reconstructing details at the surfaces of the clear scene, while we expect the fog to be intrinsically more coarse. Therefore, we do not use w_F to guide the fine sampling procedure, but we use w_c instead. Moreover, we do not use separate coarse and fine networks to model the fog, but we use just one evaluated once in $(N_{coarse} \cup N_{fine})$.

3.3. Training and Evaluation Details of Reconstruction Baseline Methods

This section provides additional details for the training and evaluation of the baseline scene reconstruction methods.

We compare the proposed ScatterNeRF with NeRF [14]; Mip-NeRF [3]; Ref-NeRF [23], a variant of Mip-NeRF with better ability for modeling reflective surfaces; voxel-based methods [9, 21], famous for their improved training and inference speed; methods for unbounded scenes [4, 26]; and two NeRFs with auxiliary regularizations [6, 7], supposed to converge faster and generalize better with fewer training views.

All baselines are trained with the same image resolution as ScatterNeRF and according to their original paper implementations. The training of the reference methods was conducted on an NVidia Geforce RTX 3090Ti GPU. All methods are trained for 250'000 steps for each In-the-Wild and controlled environmental sequence, similar to the training procedure of ScatterNeRF. The batch size was set to 5000 for Plenoxels [9], to 4096 for Aug-NeRF [6], Mip-NeRF [3], and Mip-NeRF360 [4], and to 2048 for DVGO [21], NeRF [14], NeRF++ [26], Ref-NeRF [23], and DS-NeRF [7]. We extend ScatterNeRF to Neural Scene Graphs [15] by adding the scattering media module f_p in the global frame coordinates. We follow [15] and evaluate the sampled ray points inside the dynamic bounding boxes in local normalized coordinates. In contrast, we query f_p for the participating media in global coordinates and compose the overall density and color per the NSG approach. We find that this straightforward integration is

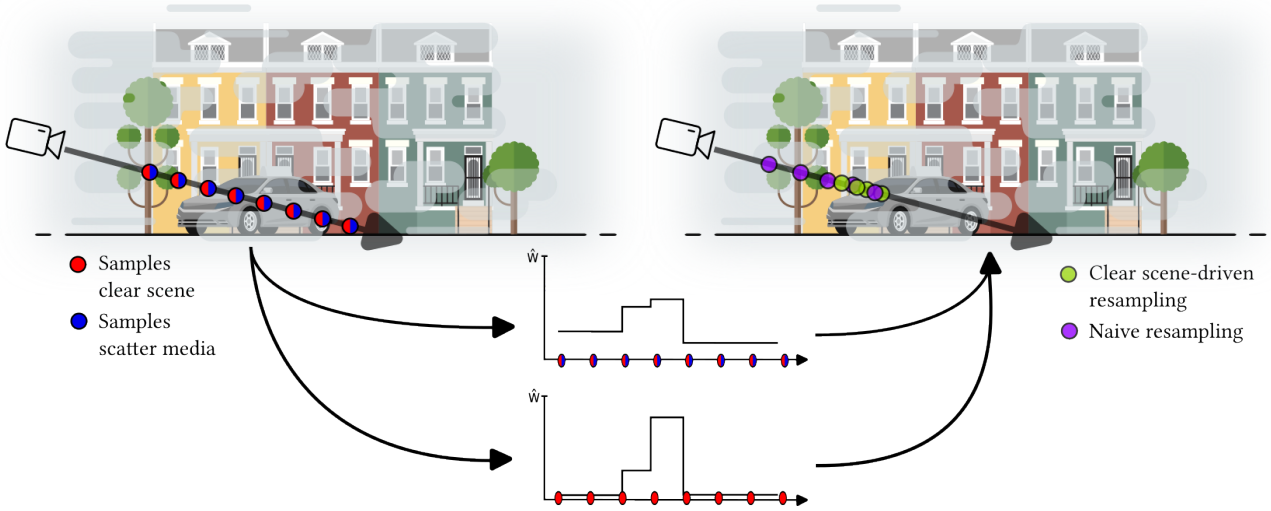


Figure 2: After the first stage of hierarchical sampling, using both the clear (in red) and scattering media (in blue) densities to compute the sampling probability for the second stage leads to drawn points (in violet) spread both in the scattering media and close to the object's surface. However, sampling only with clear scene densities corrects this behavior and yields points (in green) concentrated near the surface.

enough to learn the scattered scene and provide a disentangled representation. The overall training loss remains the same as ScatterNeRF, outlined in Section 3.1

All models are supervised by the mean-square image reconstruction error between ground-truth and reconstructed haze image. In addition, the scene representation of DS-NeRF [7] is supervised with a ground-truth depth predicted by [12]. For fair comparison to NeRF methods relying on two MLPs, for coarse and fine sampling, we reduced the number of layers of Mip-NeRF360 from three to two.

Furthermore, we can expand the Mip-NeRF360 in an analogous manner to ScatterNeRF and ScatterNSG, by querying an MLP f_p for the media radiance and volume density on the first and last sets of sampled points. As Mip-NeRF 360 already has a regularization for the weights w_c, we drop the regularization loss L_{ec} while keeping the other losses to guide f_p as well. These changes allow ScatterMipNeRF360 to reach a PSNR of 42.15 dB over the In-the-Wild subset, an improvement of 3.0% over the original Mip-NeRF360.

We rely on three commonly used image reconstruction metrics for quantitative evaluation: PSNR, SSIM, and LPIPS. While PSNR calculates per-pixel errors between ground-truth and reconstructed image, SSIM and LPIPS perform patch-wise comparisons. SSIM is able to compare the luminance, contrast, and structure of the images, and LPIPS compares internal activations of trained CNNs. For the SSIM metric, we use a window size of 11, and for LPIPS, we use AlexNet, which was trained on ImageNet.

3.4. Training Details of Dehazing Baseline Methods

All baseline methods [2, 17, 10, 20, 24] are trained according to the original paper implementations. PFF-Net [2] and EPDN [17] are trained on a variety of simulated fog scenes covering a wide spectrum of outdoor scenes, wide field of view areal imaginary, automotive street scenes. Both methods have to overcome the challenging synthetic to real-world generalization challenge and therefore cause a magnitude of visual artifacts visible as flickering in the supplemental video. ZeroScatter [20] is trained on a mixture of real and simulated foggy automotive scenes and uses multi-modal feedback from a gated camera and geometric constraints from temporal and stereo multi-view cues. Therefore, ZeroScatter achieves some of the most consistent dehazing results among the reference methods. Nonetheless, the method also models an ISP to predict good weather conditions. This leads to a similar "bright" color tone, not taking into account the true illumination conditions within the scene. ZeroRestore [10] runs a test-time optimization on every image independently. Therefore, per design, the method is reinitialized for each image separately, leading to flickering in illumination and the amount of dehazing if the optimization process is stuck in a local minimum. Hence, it has to be noted that ZeroRestore does not have to perform any generalization task

in the sense of generalizing to prior unseen scenes, novel views, or different fog types. The test-time optimization stops as soon as the self-consistency is reached. In terms of implicitly wrongly predicted scene depth, which is an arduous task to be learned from one single image, the fog effects can be projected onto the scene itself, causing the optimization to stop early. Furthermore, the optimization is performed statistically, such that a single enhancement might change with a different seed for a particular optimization run.
Lastly, MAP-Net [24] is trained using 3'500 videos with synthetic haze, collected and processed from multiple automotive datasets. By using a memory-based physical prior guidance module to enhance the scene radiance recovery and a recurrent multi-range scene radiance recovery module to capture long-range temporal haze and scene clues from the adjacent frames, this method reaches the best results among the baselines. However, as it is still a forward model which does not optimize one scene for all the frames, its results have still some inaccuracies and inconsistencies. Furthermore, such method is being trained using automotive videos, and hence can not function properly in case of other scenes or when there is only a set of sparse images (like for the scenes in the fog chamber).

4. Networks Details

For the network implementation, we follow [14]. First, the positional arguments are encoded through higher-order spherical harmonics and fed into the radiance field. In practice, such positional encoding is a non-learnable transformation from R to R^{2L} in the form,

$$\gamma(x) = [\sin(x), \cos(x), ..., \sin(2^{L-1}x), \cos(2^{L-1}x)]^T, \tag{33}$$

where L is a hyperparameter controlling the bandwidth of the projection [22]. Positional encoding is applied to each component of the ray position $\mathbf{r}$ both 3D position $\mathbf{x}$ and direction $\mathbf{d}$, allowing the network to capture higher frequency details and rapid changes in the scene. We set L to 10 for $\gamma(x)$ and 4 for $\gamma(d)$, thus $n\_pos_x = 20$ and $n\_pos_x = 8$.

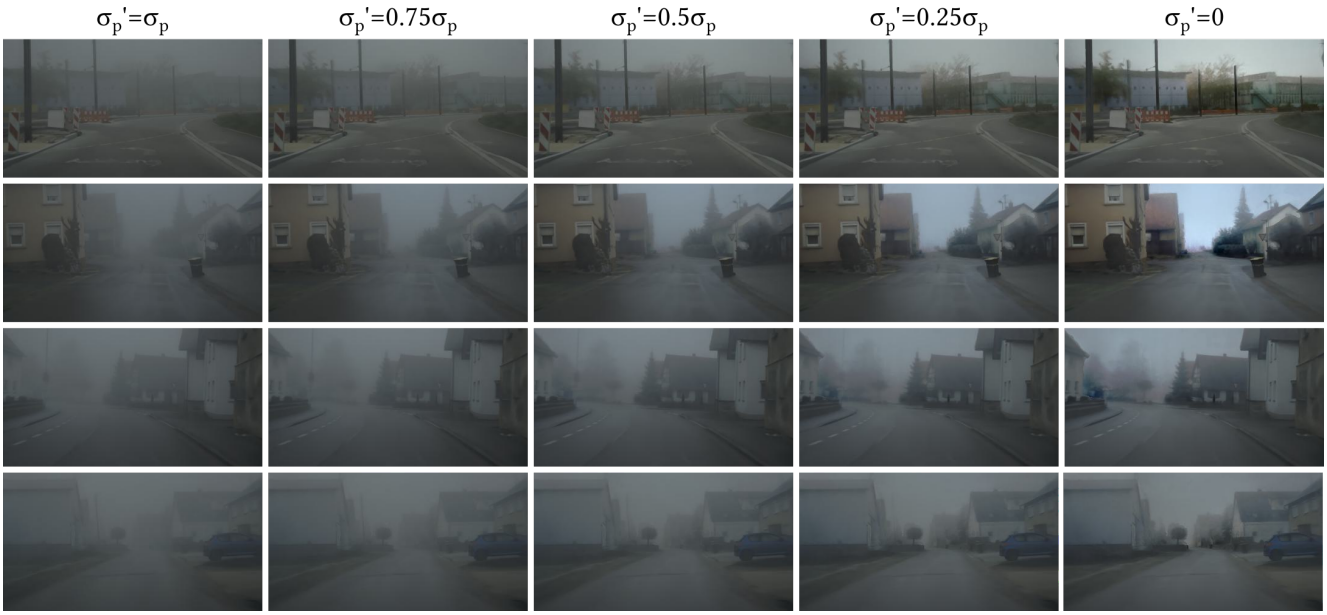
Clear Scene network f_c				
Layer #	**Layer**	**Activation**	**Input Shape**	**Output Shape**
0	Linear	ReLu	$3 \cdot n\_pos_x$	256
1	Linear	ReLu	256	256
2	Linear	ReLu	256	256
3	Linear	ReLu	256	256
4	Concat+Linear	ReLu	$256 + 3 \cdot n\_pos_x$	256
5	Linear	ReLu	256	256
6	Linear	ReLu	256	256
7	Linear	ReLu	256	256
8	Linear	/	256	256
9a	Linear	ReLu	256	1
9b	Concat+Linear	ReLu	$256 + 3 \cdot n\_pos_d$	128
10	Linear	Sigmoid	128	3

Table 1: Detail for the clear scene MLP $f_{c_{clear}}$ and $f_{c_{fine}}$, replicating the architecture used in [14]
.

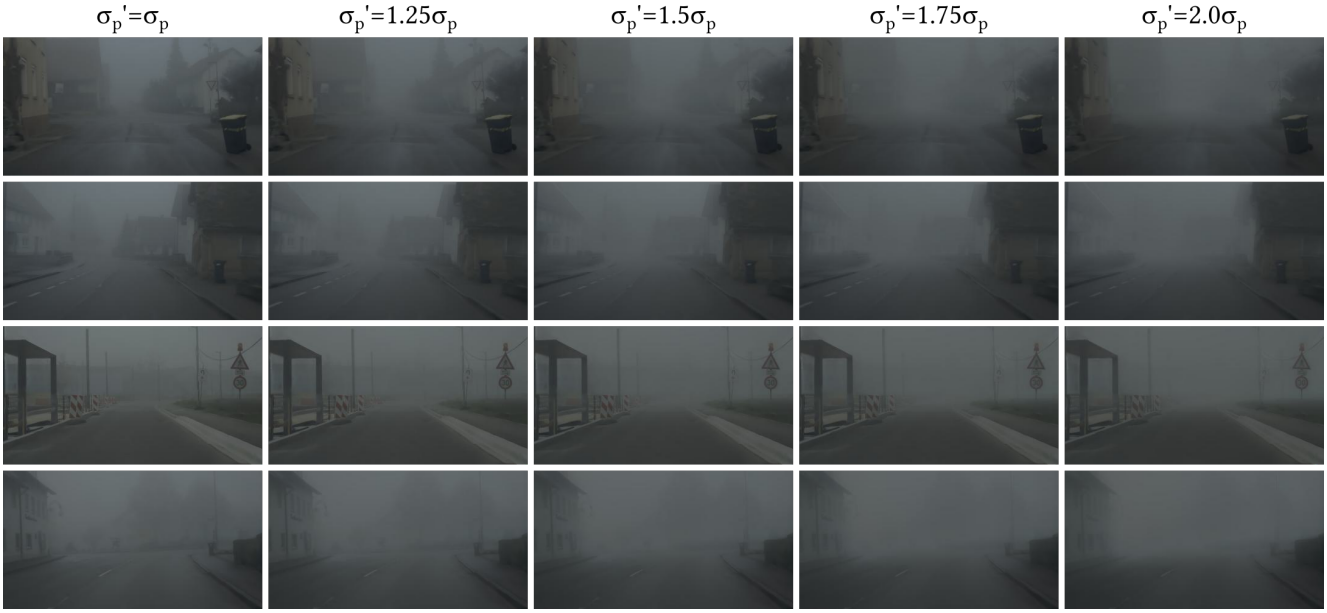
Participating media network f_p				
Layer #	**Layer**	**Activation**	**Input Shape**	**Output Shape**
0	Linear	ReLu	$3 \cdot n\_pos_x$	128
1	Linear	ReLu	128	128
2	Linear+Concat	ReLu	$128 + 3 \cdot n\_pos_x$	128
3	Linear	/	128	128
4*a*	Linear	ReLu	128	1
4*b*	Concat+Linear	ReLu	$128 + 3 \cdot n\_pos_d$	128
6	Linear	Sigmoid	64	3

Table 2: Detail for the participating media f_p. The architecture is similar to f_c but with fewer layers and half the layer width. This prevents any overfitting on the participating media and lowers the computational requirement for this additional network.

5. Rendering Different Fog Densities and Colors



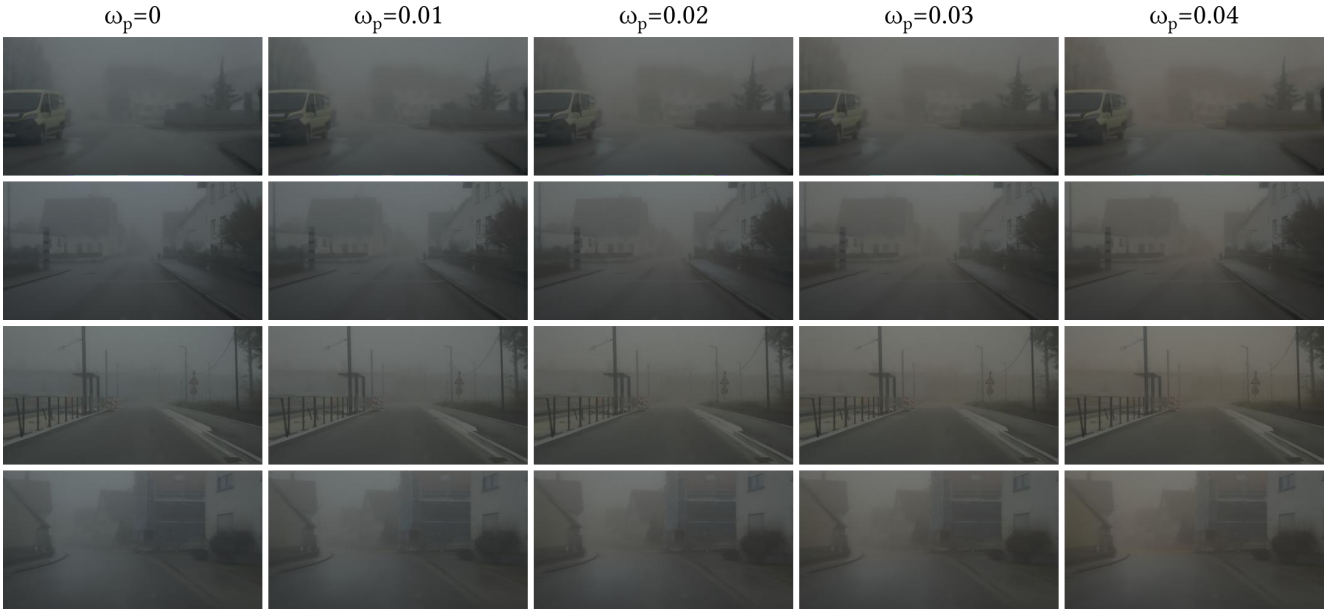
(a) Reduction of the fog density from $\sigma'_p = \sigma_p$ to $\sigma'_p = 0$.



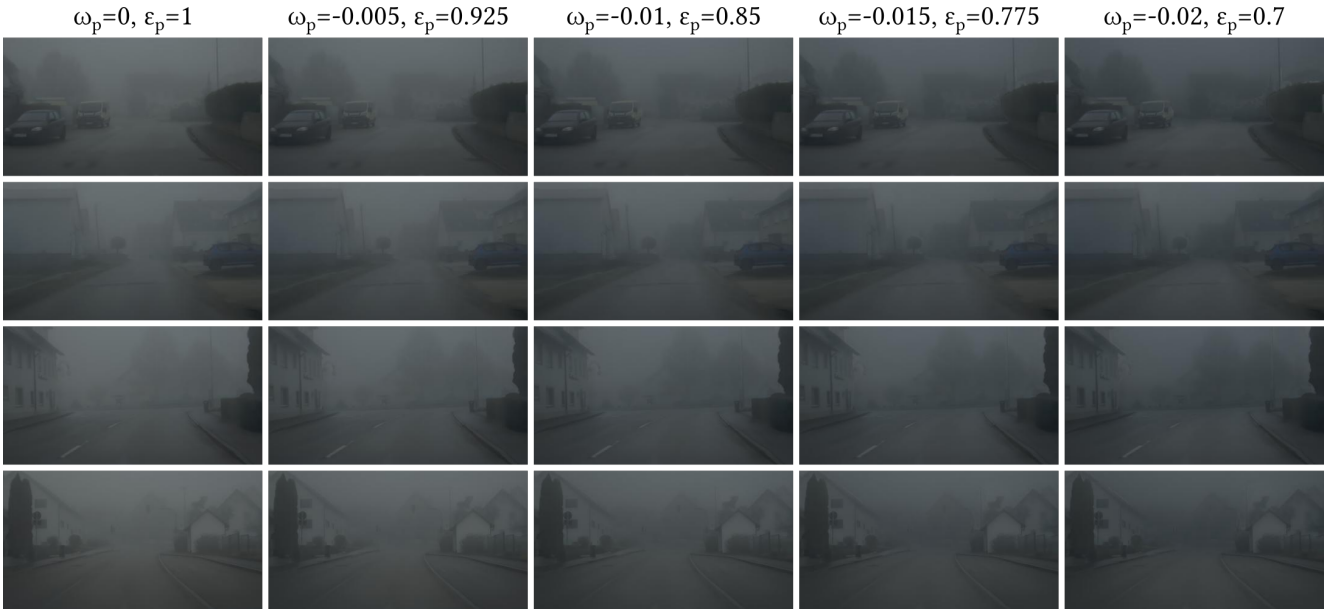
(b) Increase of the fog density from $\sigma'_p = \sigma_p$ to $\sigma'_p = 2\sigma_p$

Figure 3: The proposed method allows a simple change of the fog density. By increasing or decreasing σ_p, images with different physically correct rendered fog levels are generated. Please note how buildings and objects in far distances appear and disappear when varying the fog density.

Based on our rendering pipeline, we can not only remove the fog as presented in the main document but also alter it arbitrarily, as shown in Figure 3. Changing the fog density can be achieved by scaling the predicted



(a) Changing the fog color to an orange tone allows us to generate images captured in sunrise or sunset conditions.



(b) Lowering the brightness and changing the fog color to a blue tone allows us to generate images captured in dawn or twilight conditions.

Figure 4: The proposed method allows re-rendering the scenes with a change of the fog's brightness and color. This is achieved by varying the RGB values of the airlight $c_p = [R, G, B]$ with the formula (R,G,B) = ϵ_p(R+ω_p,G,B−ω_p). Please note that, especially in sky regions, the color change of the airlight is visible, representing the light source

scattering volume density σ_p by a factor α. Our new fog density σ'_p can be written as,

$$\sigma'_p = \alpha\sigma_p, \tag{34}$$

with $\alpha \in \mathbb{R}^+$. By plugging in the new σ'_p in the rendering formulas, the effect of the scattering media can be re-weighted, and we can render a realistic scene in higher or lower fog densities compared to the ground-truth

sequence while using the learned true fog distribution within the scene. Figure 3 shows an example of changing the fog density.
Furthermore, also the airlight color and scene illumination can be changed as shown in Figure 4. We experiment with $c_p = [R, G, B] \in \mathbb{R}^3$ with R being the red, G the green, and B the blue color channel. All three channels are $R, G, B \in [0, 1]$ and can be rescaled using two parameters for color balance ω_p and brightness ϵ_p,

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \min \left(\max \left(\epsilon_p \begin{bmatrix} R + \omega_p \\ G \\ B - \omega_p \end{bmatrix}, \ 0 \right), \ 1 \right). \tag{35}$$

6. Additional Qualitative Comparison for Scene Reconstruction

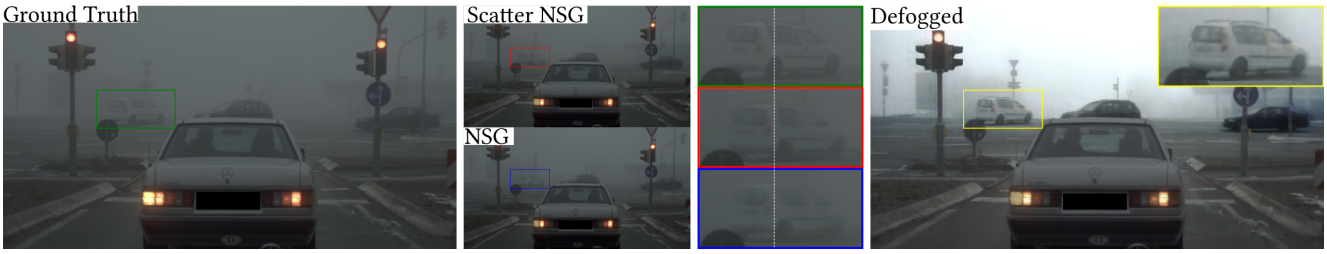


Figure 5: Qualitative comparison of ScatterNSG and baseline NSG for the foggy scene. The white car in the background is rotated, as indicated by the dashed white line in the cutouts. The ScatterNSG is able to learn a more consistent representation of the car compared to the baseline NSG. Defoggification results for ScatterNSG are provided on the right.

In this section, we provide additional qualitative results for scene reconstruction and compare them with different baselines.

As indicated by Fig. 6, ScatterNeRF learns a disentangled representation of fog and scene. This becomes evident, for example, in the first and second row of Fig. 6 for the In-The-Wild dataset. ScatterNeRF clearly separates the car and house from the fog and does not produce a blurred result as, for instance, Plenoxels, Aug-NeRF, or DVGO. Methods like NeRF++, Mip-NeRF, or Mip-NeRF360 do not suffer as much from blurring. Nevertheless, details like the rear of the blue van still are increasingly blurry compared to ScatterNeRF. Furthermore, ScatterNeRF learns that fog is a semi-homogeneous scattering media and subsequently avoids grey floating artifacts on objects in the fog. This can, for instance, be seen in the third and fourth row of Fig. 6 on the trash can. Additionally, it can be seen that the color of colorful objects, like the cones in the fourth and fifth row of Fig. 6 are not cast on solid surfaces as the road. ScatterNeRF distinguishes well between the different surfaces and learns sharp contours and rich contrast for these objects, whereas other baseline methods only produce blurry reconstructions.

Qualitative results for ScatterNSG can be found in Fig. 5. Here, the scene is altered by rotating the compact white car in the back, as visible in the cutouts. ScatterNSG learns a better representation and details, while the car rendered by the baseline NSG blends in with the fog and surroundings. Additionally, ScatterNSG provides the opportunity to defog the scene and preserve these details.

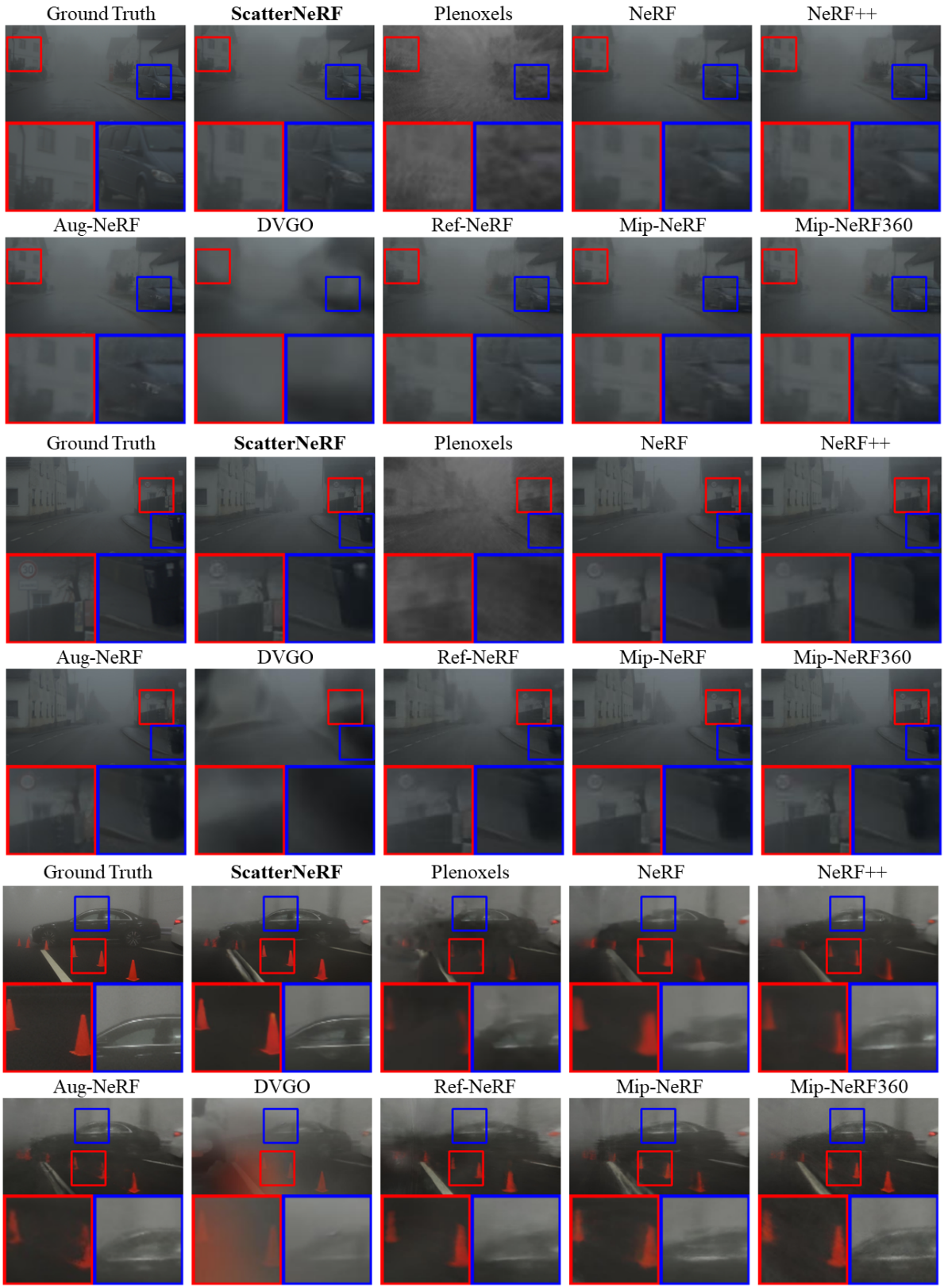


Figure 6: Qualitative comparison of the proposed ScatterNeRF and state-of-the-art scene reconstruction methods.

7. Comparison to State-of-the-Art Descattering Algorithms

In this Section we illustrate additional qualitative descattering results in Figure 8a, Figure 8b, Figure 7, 9 and Figure 10. Figure 8a and Figure 8b show the temporal consistency of image enhancement results. The learning of the scene representation explicitly enforces multiple views to be consistent with one another which enforces temporal consistency for our ScatterNeRF approach. This is not the case for reference methods such as ZeroRestore [10], PFF [2] and EPDN [17] which suffer heavily from such distortions. ZeroRestore [10] gets stuck in local optimization minima and leads to different degrees of removed fog, causing brightness flicker as explained in Section 3. PFF [2] and EPDN [17] suffer from blacked out areas, which are propagating through the scene. For a better illustration of the temporal flickering see the supplemental video.

Figure 7, Figure 9, and Figure 10 show the qualitative performance of the proposed ScatterNeRF approach in comparison to ZeroScatter [20], EPDN [17], D4 [25], ZeroRestore [10] and PFF [2]. We illustrate the image enhancement performance for the full scene and zoom ins in long distances. In Figure 7 ScatterNeRF is the only method to reconstruct the tree, rooftop and side buildings fully. In Figure 9, only ScatterNeRF and ZeroScatter [20] are able to reconstruct the building in the long distance while for the other methods, the building is darkened to much. ZeroScatter is additionally able to reconstruct the windows on the building but misses due to the predefined ISP the true colortone of the scene. In Figure 10 similar performances as in the previous two Figures can be observed.

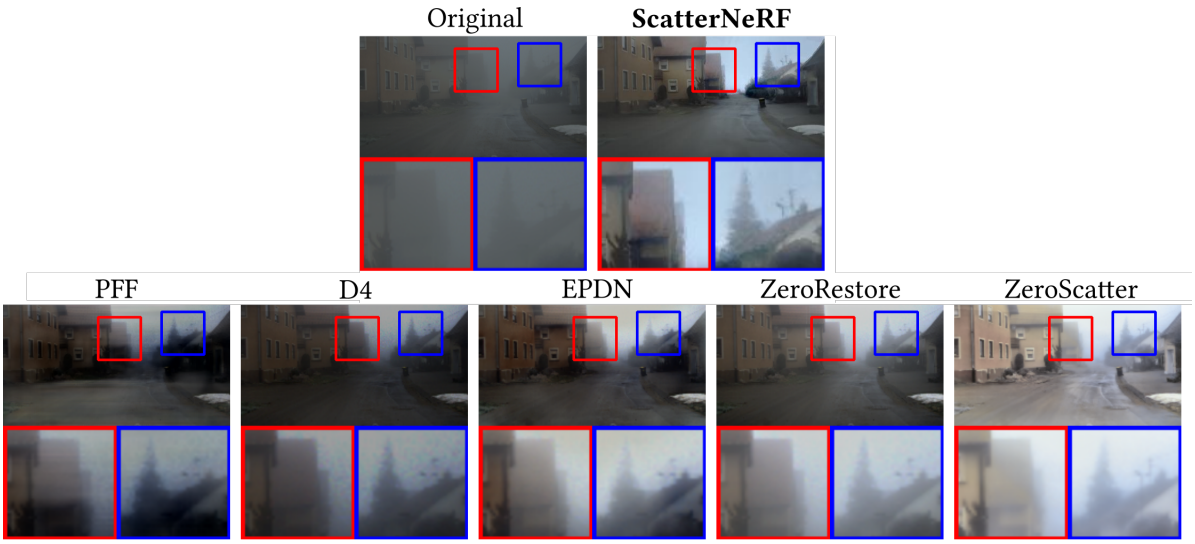
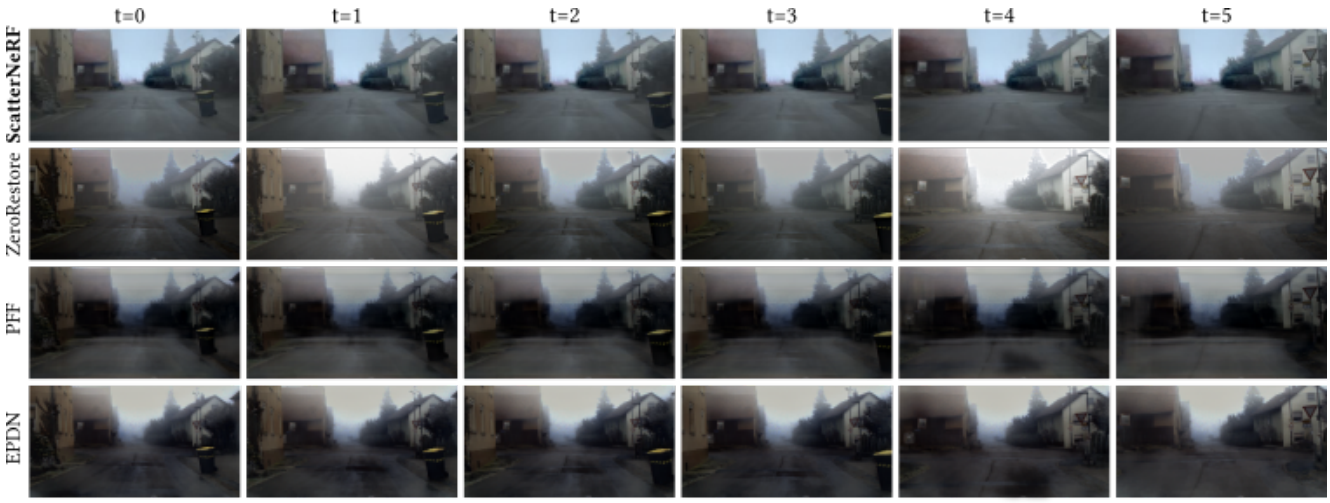
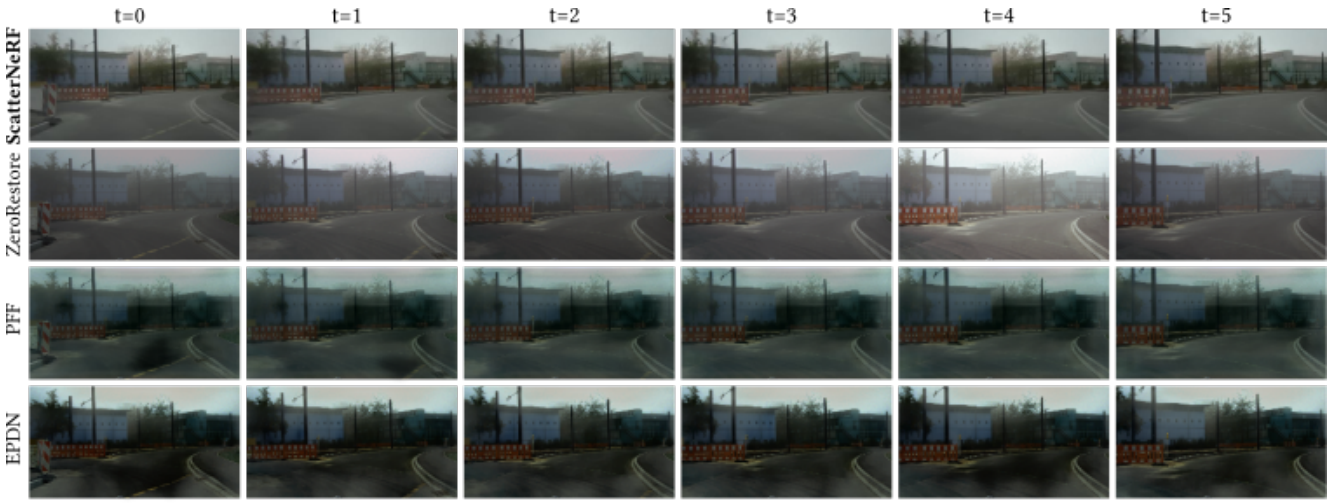


Figure 7: Qualitative comparison of the proposed ScatterNeRF and state-of-the-art dehazing methods. ScatterNeRF is able to remove fog while preserving colors much better than the other methods.



(a) PFF and EPDN are not able to learn a consistent representation of the road, resulting in randomly appearing black spots on the road.



(b) ZeroRestore is a zero-shot single image-based dehazing method that is not able to predict the same ambient light for consecutive frames, visible through color changes for the same objects in different frames.

Figure 8: ScatterNeRF learns a representation of the entire sequence and is consistent across consecutive frames. In contrast, other descattering methods rely upon single images and are affected by flickering effects. For further comparison, please have a look at the supplementary video.

8. ScatterNeRF Datasets

This section provides additional details on the In-the-Wild and controlled environment sequences captured to train and evaluate this method.

8.1. Pose Estimation

For estimating camera poses, we choose to perform Hierarchical Localization with hloc [18], and SuperGlue [19] on every sequence of the ScatterNeRF dataset due to the proven [19, 1] reliability of the method under challenging conditions. At first, we extract features from each image in the dataset with SuperPoint [8]. We use configuration settings already present in the open sourced code, namely the *superpoint_aachen* configuration for the In-the-Wild

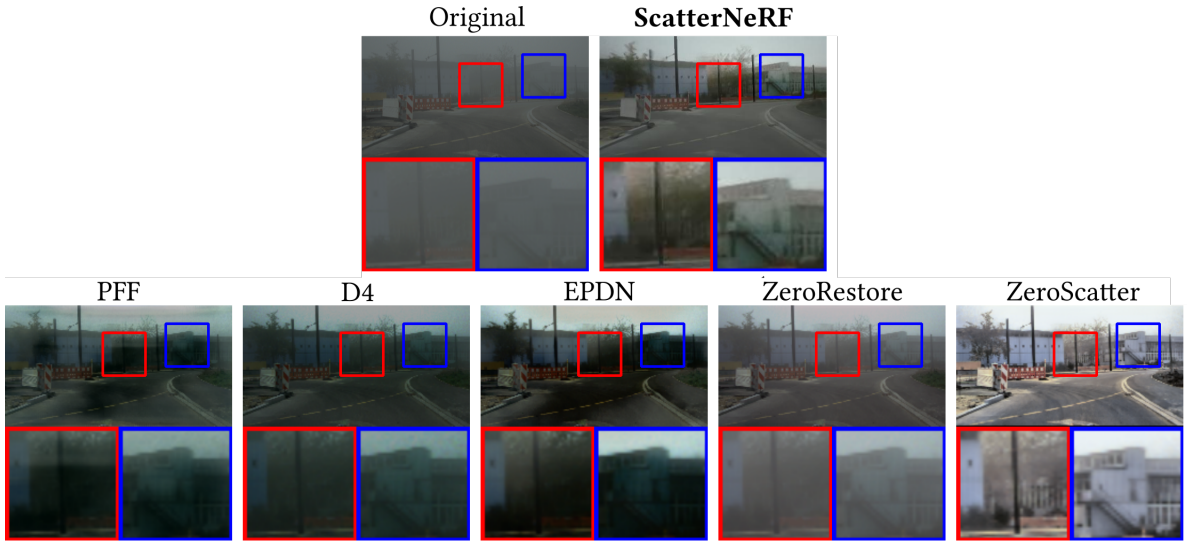


Figure 9: Qualitative comparison of the proposed ScatterNeRF and state-of-the-art dehazing methods. PFF, EPDN and D4 tend to predict foggy areas too dark, while ZeroScatter has a yellow cast, leading to unrealistic looking defogged colors.

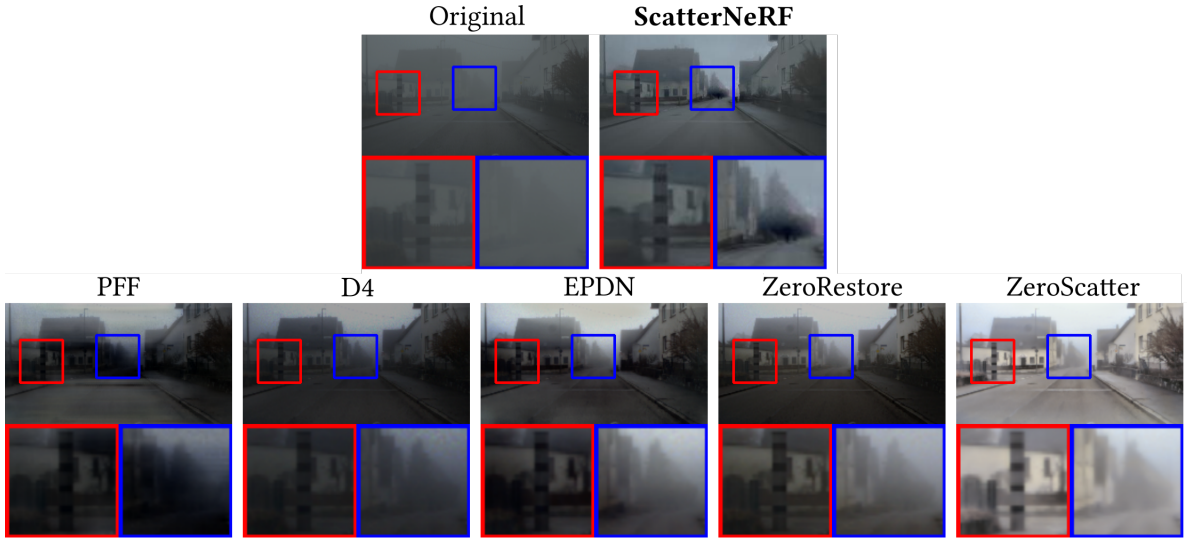


Figure 10: Qualitative comparison of the proposed ScatterNeRF and state-of-the-art dehazing methods.

dataset and the *superpoint_inloc* for the controlled environment dataset. Then, we perform exhaustive feature matching using SuperGlue [19] and reconstruct a 3D model of the sequence and reference poses. For In-the-Wild sequences, the reconstruction is performed with all the available rectified frames due to their visual consistency, and we use our calibrated pinhole camera model as prior. On the other hand, the 3D model for the controlled environment sequences is reconstructed from clear images alone, and we let the algorithm estimate a unified camera model with simple radial distortion. We then undistort the haze-free and foggy images with the estimated parameters and perform localization on the hazed images with Hierarchical Localization [18]. Fig. 11 provides an example of the estimated poses for the controlled environment dataset.

8.2. In-the-Wild Dataset

For the In-the-Wild dataset, we capture 9 different sequences in real fog conditions with an Aptina-0230 camera stereo pair mounted behind the car windshield, running at 20 Hz. The sequences are collected on different days

(a) Toyota (b) Car Accident

Figure 11: Distribution of camera poses in the controlled environment dataset. The position and heading of the camera is visualized with red frustums. For visualization purposes we show the camera poses in context of the sparse point cloud estimated by hloc.

and in different locations around Germany, ensuring a diverse dataset with different fog conditions. Furthermore, we select frames from urban and suburban scenarios to create scenes with diverse backgrounds suitable for our evaluations. An overview of the number of collected frames is provided in Tab. 3.

Sequence	Tram Station	Farm	Intersection	Suburb	Speed Control	Road Fork	Adds	Construction	Countryside
Num. frames	228	376	254	212	390	302	280	214	422

Table 3: Number of captured frames per In-the-Wild sequence.

For each frame, the ground-truth depth is estimated with a pretrained stereo algorithm introduced in [12]. We employ the open-sourced model trained on a large datasets ensemble without any fine-tuning, as we found the generated depth maps to be sufficient for the necessary depth guidance. These results can be seen in Fig. 12-14, where ground-truth depths are shown with their respective RGB image.

8.3. Controlled Environment Dataset

The controlled environment dataset is captured in the fog chamber of the Japan Automobile Research Institute (JARI)[1]. The chamber is 200m long, 15m wide and can reproduce adverse weather effects ranging from rain to strong oncoming sunlight and fog. The fog visibility can be controlled from 10m to 80m and we kept it to 20m for the Toyota sequence and 40m for the Car Accident sequence. Furthermore, we set the scene illumination of the ceiling lights to 800 Lux to ensure comparable lighting conditions for foggy and clear reference sequences.

As Camera we use a Nikon D5200 DSLR camera with fixed exposure 20 ms, aperture f/10, ISO 800 and focal length 18 mm.

In contrast to the In-the-Wild scenes, we are able to obtain for each fog scene captured in the controlled environment its corresponding clear reference. This allows to compare dehazed images directly to their clear reference equivalent. For this, we used a Leica ScanStation P30 to obtain dense depth maps and warp clear reference images to a foggy image. This procedure is illustrated in details in Fig. 15. Similar to the In-the-Wild dataset, we use hloc to estimate the camera poses as well as the camera intrinsics [18]. We opt to use the clear reference images as database images since we find that this improves the quality of the estimated poses for both clear and hazed captures significantly. From the clear reference images, hloc is able to generate a sparse point cloud. We effectively replace this point cloud with a dense laser scan produced by the Leica scanner. An example of a dense scan can be seen in Fig. 15, where the color of the points encodes the distance. Laser scans are performed from multiple positions in the fog chamber and we overlay multiple scans to produce high-resolution scans that suffer from little occlusions. Next, we use the Iterative-Closest-Point (ICP) algorithm [5] to estimate the transform consisting of scale, rotation and

[1]https://www.jari.or.jp

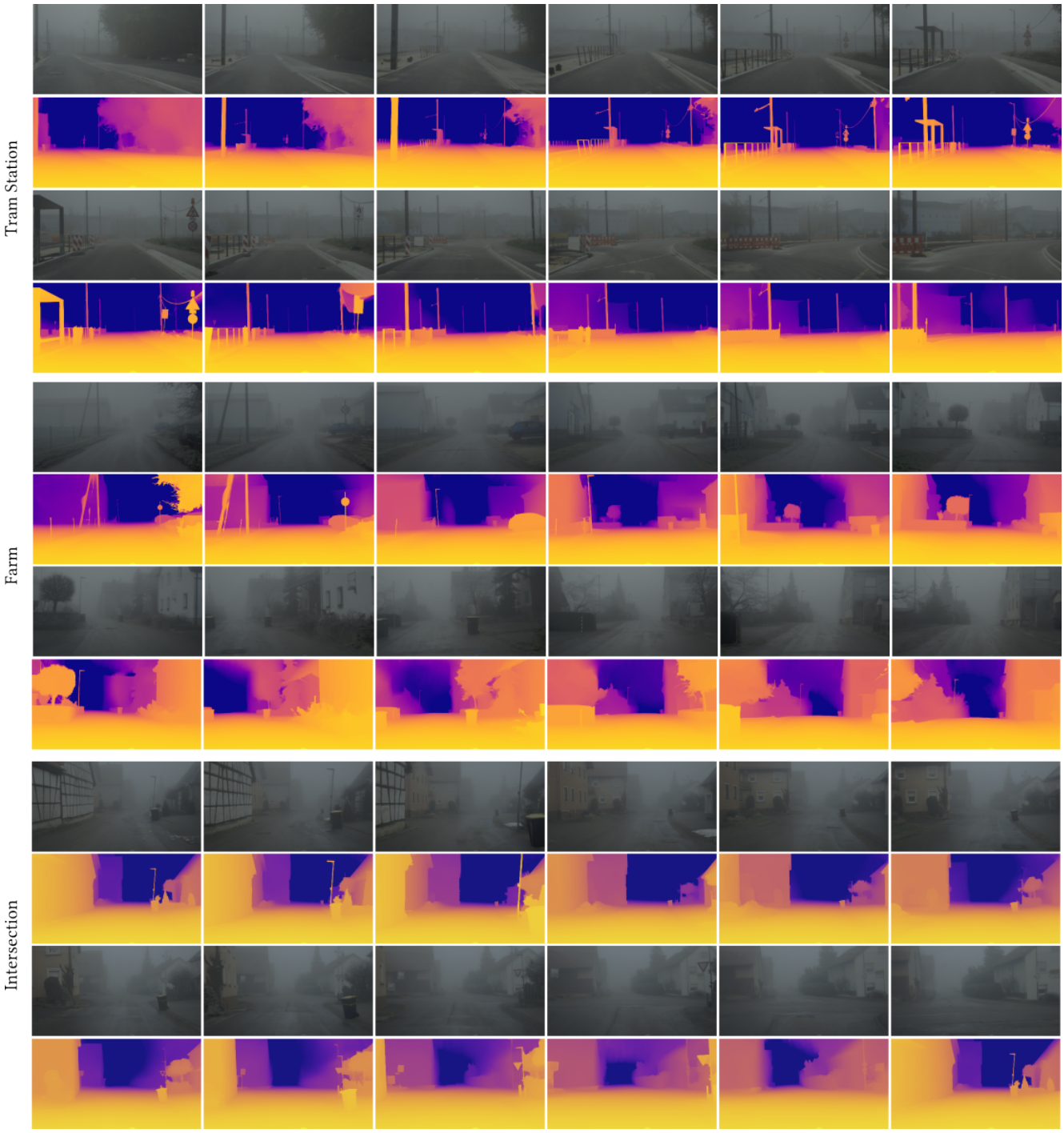


Figure 12: Examples of our In-the-Wild dataset. We show RGB image and estimated ground-truth depth.

translation - between dense laser scan coordinates and hloc coordinates. An example overlay of sparse hloc point cloud and dense laser scan can be found in Fig. 15. The transformed dense laser scan in combination with the estimated camera intrinsics allows us to produce dense depth maps for arbitrary camera poses. In order to produce depth maps for foggy images, we query hloc with the foggy image to obtain the respective camera poses and subsequently produce the corresponding depth map.

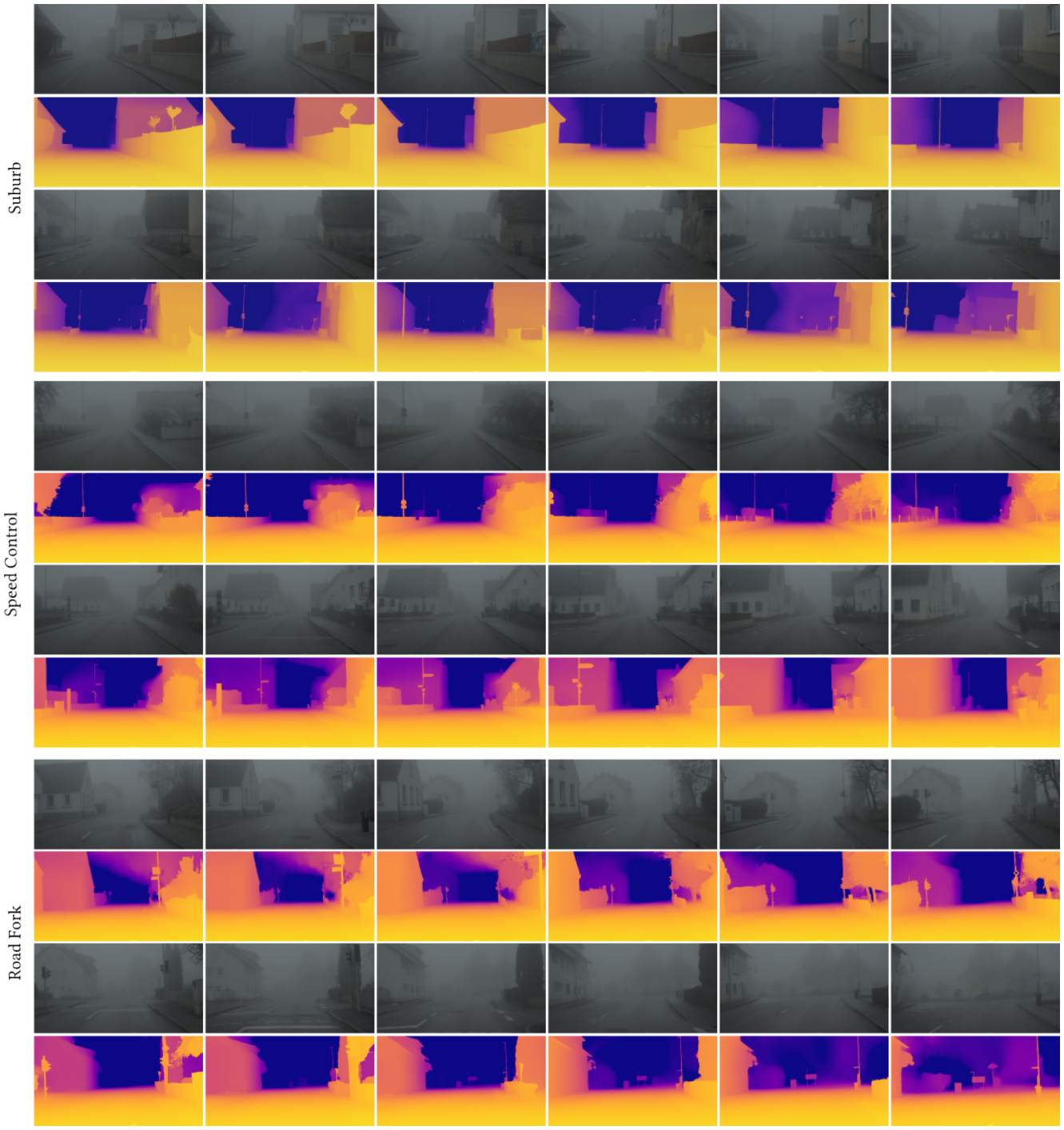


Figure 13: Examples of our In-the-Wild dataset. We show RGB image and estimated ground-truth depth.

For evaluation purposes, clear reference images for every foggy image are advantageous. To this end, we warp the closest clear reference images to foggy camera poses using the previously generated dense depth maps. An example image can be found in Fig. 15.

We use this procedure to generate two different sequences as described by Tab. 4. Furthermore, Fig. 11 illustrates the diverse poses captured for both sequences. Fig. 16 and Fig. 17 show clear reference and foggy images with

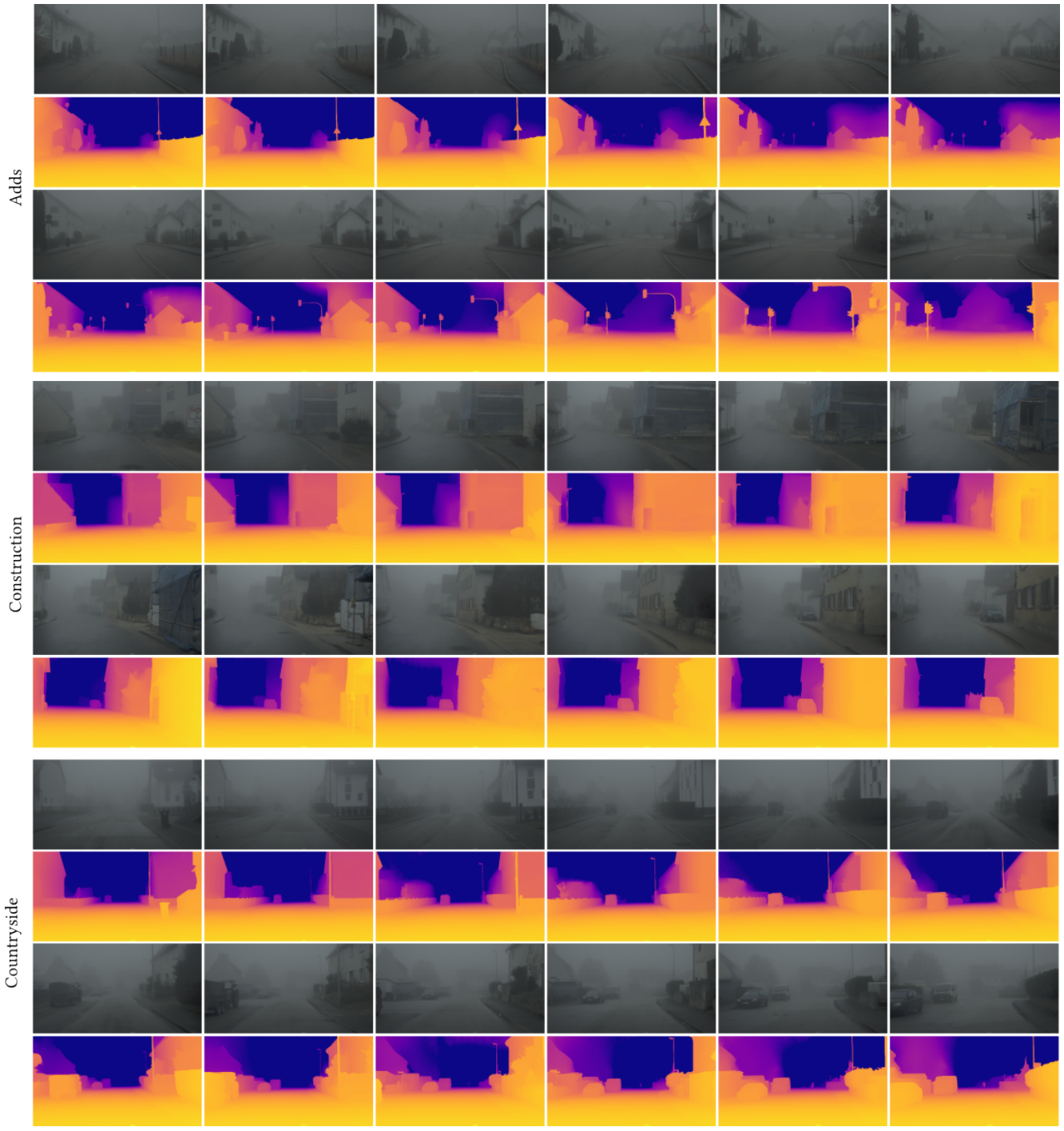


Figure 14: Examples of our In-the-Wild dataset. We show RGB image and estimated ground-truth depth.

their respective depth maps.

Sequence	Toyota	Car Accident
Num. frames	187	195

Table 4: Number of captured frames per controlled environment sequence.

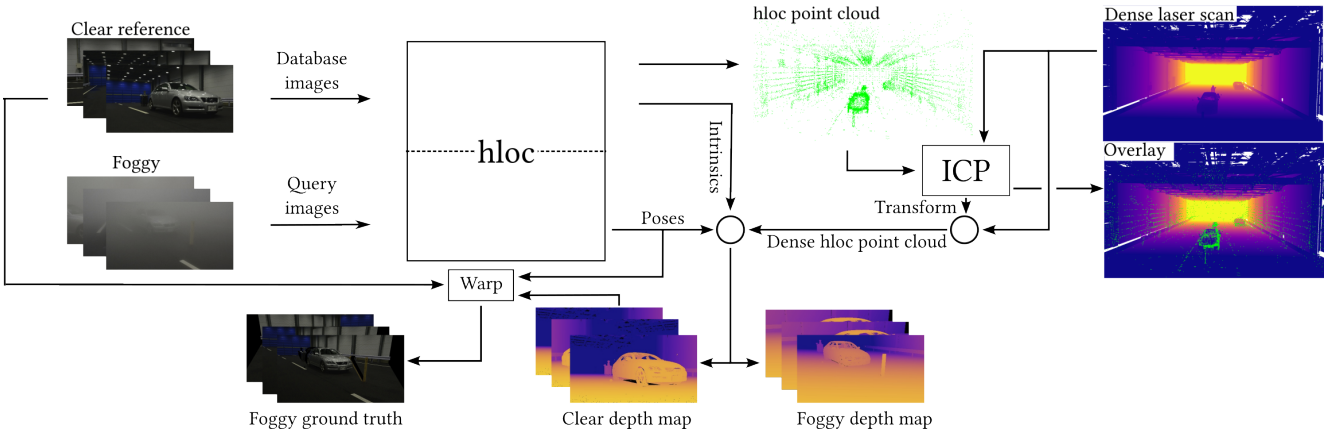


Figure 15: Depth maps for foggy images are obtained by projecting a dense laser scan to the repective camera poses. Camera poses are estimated by hloc. Clear reference images can be warped to a foggy pose to allow better evaluation of dehazing algorithms.

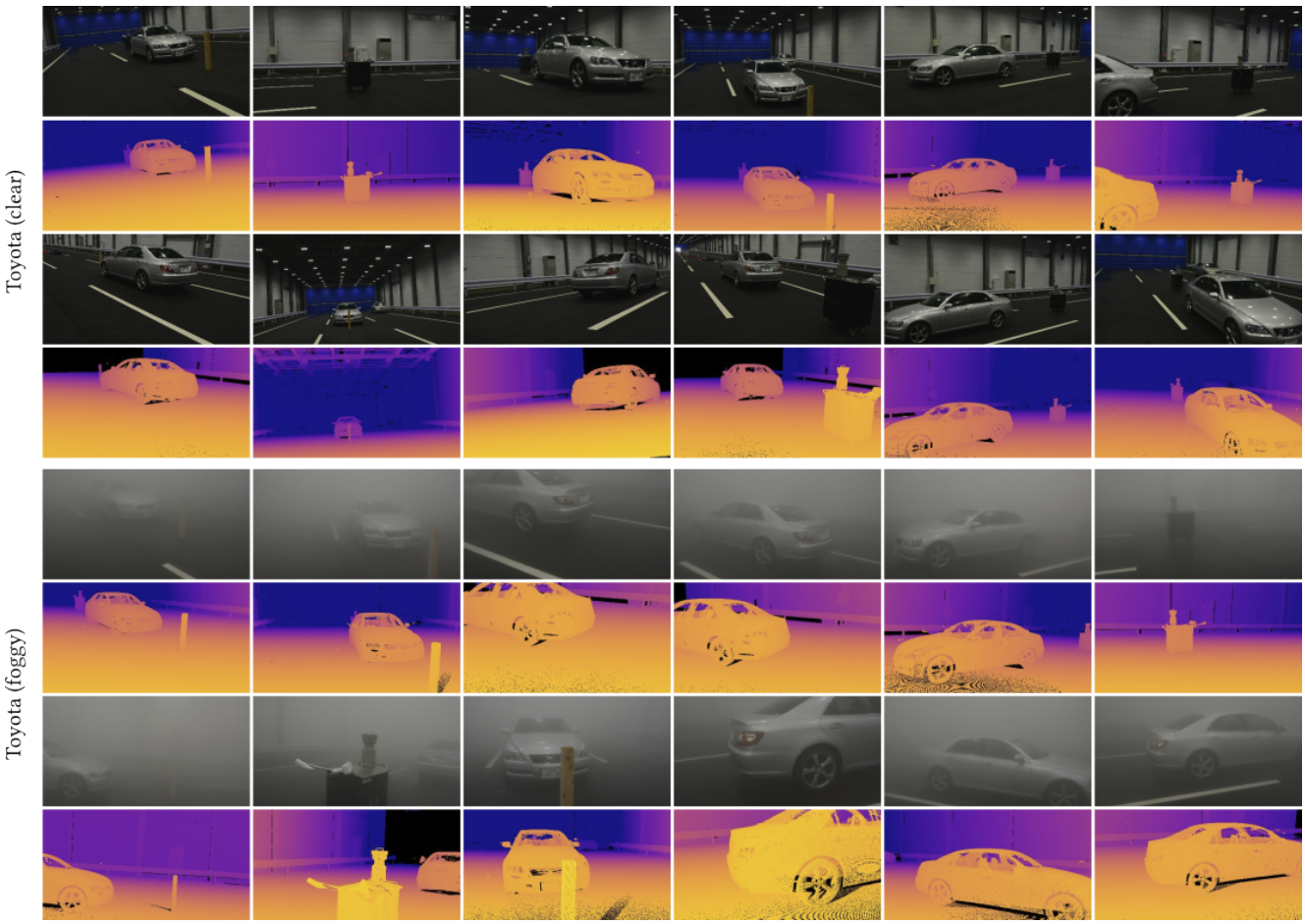


Figure 16: Examples of our proposed controlled environment dataset for the Toyota scene. For both scenes, we capture foggy and clear reference images to allow image enhancement evaluation.

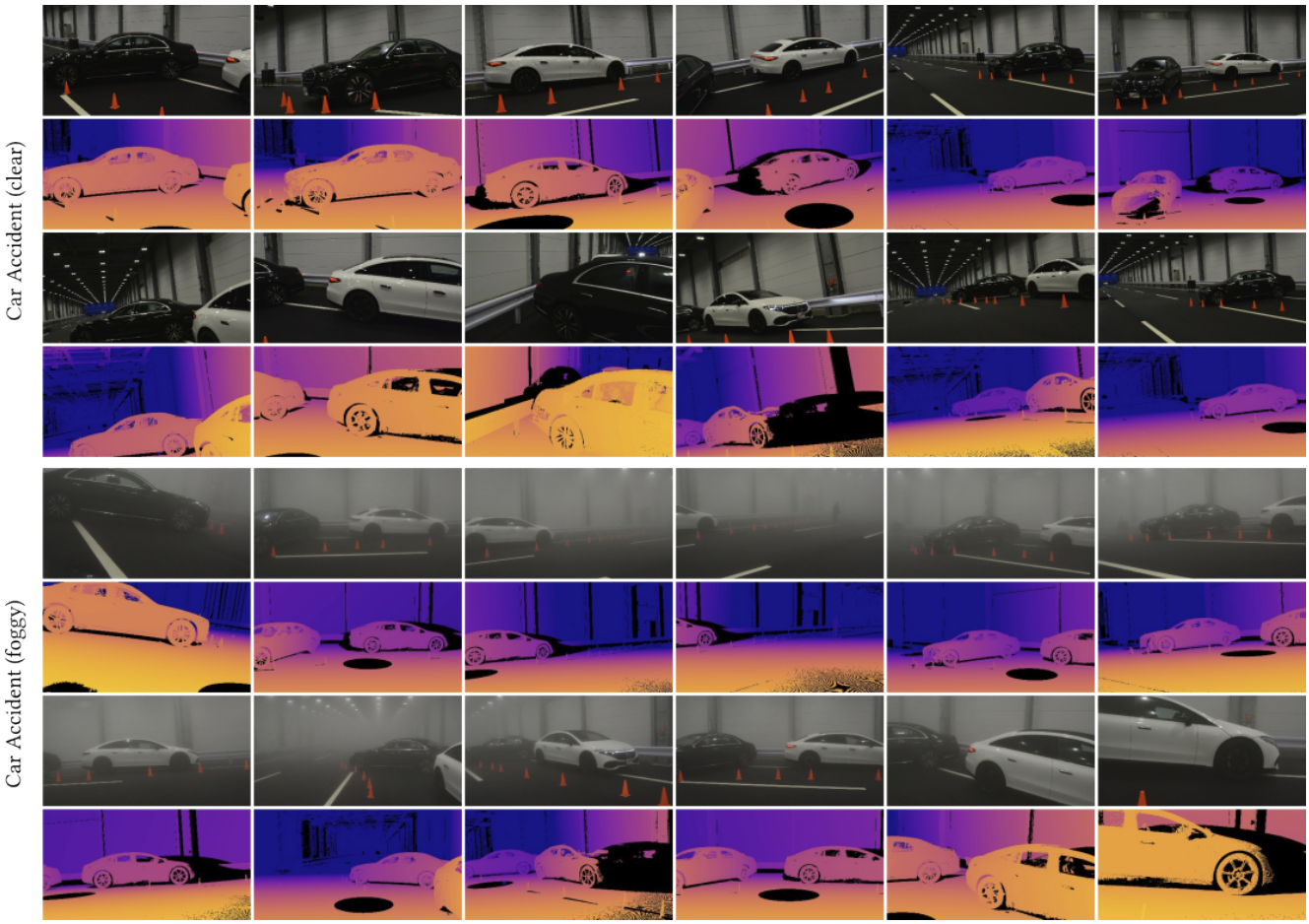


Figure 17: Examples of our proposed controlled environment dataset for the Car Accident scene. For both scenes, we capture foggy and clear reference images to allow image enhancement evaluation.

References

[1] Eccv 2020 workshop on long-term visual localization under changing conditions, 2020. 14

[2] Haoran Bai, Jinshan Pan, Xinguang Xiang, and Jinhui Tang. Self-guided image dehazing using progressive feature fusion. *IEEE Transactions on Image Processing*, 31:1217 – 1229, 2022. 6, 13

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 5

[4] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 5

[5] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 16

[6] Tianlong Chen, Peihao Wang, Zhiwen Fan, and Zhangyang Wang. Aug-nerf: Training stronger neural radiance fields with triple-level physically-grounded augmentations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 5

[7] Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12882–12891, 2022. 5, 6

[8] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description, 2017. 14

[9] Sara Fridovich-Keil, Alex Yu, Matthew Tancik, Qinhong Chen, Benjamin Recht, and Angjoo Kanazawa. Plenoxels: Radiance fields without neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5501–5510, 2022. 5

[10] Aupendu Kar, Sobhan Kanti Dhara, Debashis Sen, and Prabir Kumar Biswas. Zero-shot single image restoration through controlled perturbation of koschmieder's model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16205–16215, June 2021. 6, 13

[11] Harald Koschmieder. Theorie der horizontalen sichtweite. *Beitrage zur Physik der freien Atmosphare*, pages 33–53, 1924. 3

[12] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16263–16272, 2022. 6, 16

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *International Conference on Learning Representations*, 2018. 5

[14] 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. *Communications of the ACM*, 65(1):99–106, 2021. 1, 5, 7

[15] Julian Ost, Fahim Mannan, Nils Thuerey, Julian Knodt, and Felix Heide. Neural scene graphs for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2856–2865, 2021. 5

[16] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 5

[17] Yanyun Qu, Yizi Chen, Jingying Huang, and Yuan Xie. Enhanced pix2pix dehazing network. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8152–8160, 2019. 6, 13

[18] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 14, 15, 16

[19] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 14, 15

[20] Zheng Shi, Ethan Tseng, Mario Bijelic, Werner Ritter, and Felix Heide. Zeroscatter: Domain transfer for long distance imaging and vision through scattering media. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 6, 13

[21] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 5

[22] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 7

[23] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T Barron, and Pratul P Srinivasan. Ref-nerf: Structured view-dependent appearance for neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5481–5490. IEEE, 2022. 5

[24] Jiaqi Xu, Xiaowei Hu, Lei Zhu, Qi Dou, Jifeng Dai, Yu Qiao, and Pheng-Ann Heng. Video dehazing via a multi-range temporal alignment network with physical prior. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 6, 7

[25] Yang Yang, Chaoyue Wang, Risheng Liu, Lin Zhang, Xiaojie Guo, and Dacheng Tao. Self-augmented unpaired image dehazing via density and depth decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2037–2046, June 2022. 13

[26] Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. 5