ClimateNeRF: Extreme Weather Synthesis in Neural Radiance Field —Supplementary Material —

Yuan Li[1,2*] Zhi-Hao Lin[1*] David Forsyth[1] Jia-Bin Huang[3] Shenlong Wang[1]
[1]University of Illinois Urbana-Champaign [2]Zhejiang University
[3]University of Maryland, College Park

In this supplementary material, we describe implementation details of the simulation framework in Section 1 and provide an ablation study to validate our design choices in Section 2. ClimateNeRF is highly controllable and is demonstrated in Sect 3. We present extensive qualitative and quantitative results in Section 4 and Section 5, respectively.

1. Implementation Details

3D Scene Reconstruction Our implementation is based on [21]. We use spatial hashing grid [18] to represent the 3D scene. The entire space contains a multi-resolution feature grid $\{\text{enc}(\mathbf{x};\theta^l)\}_{l=1}^{L}$, where L is the total number of resolution levels; θ^l are learnable parameters at each level. For a 3D point $\mathbf{x} \in R^3$, we index grids by a spatial hash function [31] and fetch feature by interpolation and concatenation: $\gamma = \text{cat}\{\text{interp}(\mathbf{x}, \text{enc}(\mathbf{x};\theta^l))\}_{l=1}^{L}$. To keep spatial features intact at the stylization stage and separate gradient flows between geometry and color, we develop a disentangled version of instant-NGP inspired by [3, 28]. Each voxel maintains a *geometry code* γ_σ and *appearance code* γ_{app}. The geometry code γ_σ encodes opacity and the appearance code γ_{app} contains color, semantic and normal information:

$$\gamma_\sigma = \text{cat}\{\text{interp}(\mathbf{x}, \text{enc}(\mathbf{x};\theta_\sigma^l))\}_{l=1}^{L_\sigma}, \quad \gamma_{\text{app}} = \text{cat}\{\text{interp}(\mathbf{x}, \text{enc}(\mathbf{x};\theta_{app}^l))\}_{l=1}^{L_{app}}, \tag{1}$$

where interp stands for linear interpolation, cat denotes concatenation.

Inspired by [18], we use shallow MLPs to predict densities σ, colors $\mathbf{c}$, semantic logits $\mathbf{s}$ and surface normal values $\mathbf{n}$ respectively from geometry features γ_σ and appearance features γ_{app}. We also incorporate low-dimensional per-frame appearance embeddings $\{\ell_i^{(a)}\}_{i=1}^{N}$ [14] to balance different lighting conditions across images. With hash grids and MLPs, we have the following function to reconstruct the original scene:

$$(\sigma, \mathbf{c}, \mathbf{s}, \mathbf{n}) = F_\theta(\mathbf{x}, \mathbf{d}, \ell_i^{(a)}, \gamma_\sigma, \gamma_{\text{app}}) \tag{2}$$

Following volume rendering and alpha blending [16], we render the color $C(\mathbf{r})$ for ray $\mathbf{r}$ and its semantic logit $S(\mathbf{r})$ [39].

$$C(\mathbf{r}) = \sum_{i=1}^{N} T_i(1-\exp(-\sigma_i\delta_i))\mathbf{c}_i; \quad S(\mathbf{r}) = \sum_{i=1}^{N} T_i(1-\exp(-\sigma_i\delta_i))\mathbf{s}_i, \text{where } T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j\delta_j\right) \tag{3}$$

We apply softmax to semantic logits $S(\mathbf{r})$ to obtain semantic probabilities $\{p(\mathbf{r})^l\}_{l=1}^{L}$ for all labels. During training, we perform MSE loss L_C and cross-entropy loss L_S for rendered colors and semantic logits using ground truth color $C_{gt}(\mathbf{r})$ and 2D semantic logits $\{\hat{p}(\mathbf{r})^l\}_{l=1}^{L}$ predicted by segformer [35] pretrained on cityscape dataset [6]. Moreover, since we leverage pseudo-semantic labels, we detach densities σ when rendering $S(\mathbf{r})$. Similar to Ref-NeRF [32], we use the density gradient

normals $\hat{\mathbf{n}} = -\frac{\nabla\sigma}{\|\nabla\sigma\|}$ [2, 26] to guide the predicted surface normals $\mathbf{n}$ using a weighted MSE loss:

$$L_C = \sum_{\mathbf{r}\in\mathcal{R}} \|C(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2 \,, \tag{4}$$

$$L_S = -\sum_{\mathbf{r}\in\mathcal{R}} [\sum_{l\in L} \hat{p}(\mathbf{r})^l \log p(\mathbf{r})^l] \,, \tag{5}$$

$$L_{\mathbf{n}} = \sum_{\mathbf{r}\in\mathcal{R}} \sum_{i=N}^{N} w_i \|\mathbf{n_i} - \hat{\mathbf{n}}_i\|_2^2 \,, \tag{6}$$

where $w_i = T_i(1 - \exp(-\sigma_i\delta_i))$ denotes the detached weight in Eq.3. We also leverage distortion loss L_{dist} [1, 29] to mitigate floaters in the reconstruction results:

$$L_{\text{dist}} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} w_i w_j \left|\frac{t_i + t_{i+1}}{2} - \frac{t_j + t_{j+1}}{2}\right| + \frac{1}{3}\sum_{i=0}^{N-1} w_i^2 \left(t_{i+1} - t_i\right) \tag{7}$$

However, ngp model tends to create big 'blobs' in the sky with distortion loss. We alleviate this by applying a simple penalty $L_{sky} = \sum_{\mathbf{r}\in\mathcal{R}} \mathrm{e}^{-D(\mathbf{r})} \cdot \mathbb{1}\{\hat{p}(\mathbf{r}) = \hat{p}_{sky}\}$ where $D(\mathbf{r})$ denotes the depth following Eq. 3 [37] and $\mathbb{1}\{\cdot\}$ is an indicator function using 2D predicted semantic logits $\hat{p}$. Moreover, we incorporate opacity loss $L_O = -\sum_{\mathbf{r}\in\mathcal{R}} O(\mathbf{r}) \log O(\mathbf{r})$ [21] to encourage ray opacity being either 0 or 1 to avoid semi-transparent regions in reconstruction results. During our training time, our model's total loss is a weighted sum of the aforementioned losses:

$$L = L_C + \lambda_S L_S + \lambda_{\mathbf{n}} L_{\mathbf{n}} + \lambda_{dist} L_{dist} + \lambda_{sky} L_{sky} + \lambda_O L_O \tag{8}$$

Transient Object Occlusion To occlude transient objects like pedestrians or vehicles across views in tanks and temples dataset [10], we follow [4] and create per-image learnable masks:

$$M_i(u, v) = F_\psi(u, v, i, \gamma_M) \,, \tag{9}$$

where $(u, v) \in \mathbb{R}^2$, i denotes image coordinates and image index in all training images. F_ψ denotes a shallow MLP and γ_M is the output of hash grids.

In such case, we change color reconstruction loss 4 following [4]:

$$L_C = \sum_{\mathbf{r}\in\mathcal{R}} M(\mathbf{r})\|C(\mathbf{r}) - C_{gt}(\mathbf{r})\|_2^2 + \lambda_M(1 - M(\mathbf{r})) \,, \tag{10}$$

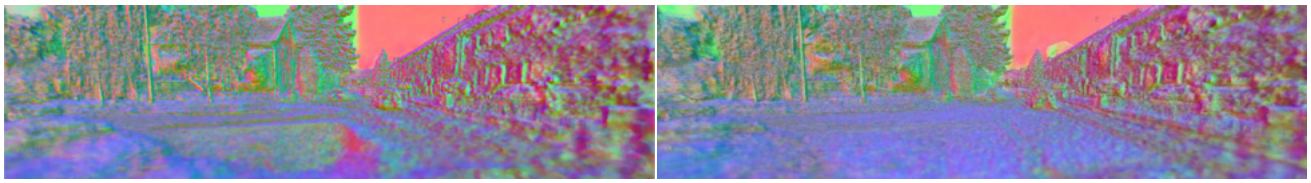
where the second term is used to prevent the mask from predicting everything transient.

Geometry improvements As mentioned in the limitation subsection in the main paper and can be seen in Fig. 15, ClimateNeRF strongly relies upon high-quality geometry. To improve geometry estimations for KITTI-360 dataset [12], we further use the monocular dense depth D_{mono} and impose a depth loss:

$$L_{\text{depth}} = \sum_{\mathbf{r}\in\mathcal{R}} \|((wD(\mathbf{r}) + d) - D_{\text{mono}}(\mathbf{r}))\|_2^2 \,, \tag{11}$$

where w and d are used to align the predicted depth from NeRF $D(\mathbf{r})$ and monocular depth cue $D_{\text{mono}}(\mathbf{r})$ with a least-square criterion [37, 7, 22]. As shown in Fig.1, holes on the ground are "filled" thanks to the supervisions from monocular dense depth.

Training Details Our implementations of the hash grids and distortion loss follow [17, 21]. For the scale and resolution of the hash grid in our model, we get the inspiration from [28], where appearance features allocate more grids with finer resolutions. Geometry hash grids have 16 levels and 2^{19} entries at most per level, while appearance hash grids have 32 levels and 2^{21} entries at most per level. For Tanks and Temples dataset [10] and MipNeRF-360 dataset [1], side length of hash grids is 16. Geometry MLP and appearance MLP have 128 neurons per layer and one hidden layer, while semantic MLP and normal predicting MLP have 32 neurons per layer and one hidden layer. For our default loss weights in Eq. 8, we set $\lambda_S = 4^{-2}$, $\lambda_{\mathbf{n}} = 7^{-4}$, $\lambda_{dist} = 3^{-4}$, $\lambda_{sky} = 1^{-1}$ and $\lambda_O = 2^{-4}$.



(a) w/o monocular dense depth D_{mono} (b) w/ monocular dense depth D_{mono}

Figure 1: **Ablation on monocular dense depth supervision** Normal estimations in (b) shows that leveraging monocular dense depth removes artifacts on the road.

1.1. Flood Simulation

Given that water is mostly muddy and non-transparent, we approximate the opacity by checking a point *above* or *under* the water's surface:

$$O_\phi(\mathbf{x}; F'_\theta) = \begin{cases} \infty & \text{if } \mathbf{n}_w(\mathbf{x} - \mathbf{o}_w) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

Water is highly reflective, yet the microfacet ripples sometimes make the water look glossy. To simulate these effects, we leverage a Spherical Gaussian (SG) to approximate the BRDF on the reflective water surface:

$$B_\phi(\mathbf{x}, -\mathbf{d}, \boldsymbol{\omega}_i, N_\phi(\mathbf{x})) = \exp^{\lambda(-\boldsymbol{\omega}_i \cdot \mathbf{d}_r - 1)} \tag{13}$$

where SG lobe axis is $\mathbf{d}_r = \mathbf{d} - 2(\mathbf{d} \cdot N_\phi(\mathbf{x}))N_\phi(\mathbf{x})$,, and $\lambda \in \mathbb{R}_+$ is the lobe sharpness, controlling the glossy effects.

We use sigma point-based sampling for rendering water. Specifically, the camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$ is cast from the camera and hits the water surface at position $\mathbf{x}$. We compute the observed color by a sampling-based approximation of the rendering equation:

$$\mathbf{c}_\phi = (1 - R)\mathbf{c}_w + R \sum_{i=1}^{5} L(\mathbf{x}, \boldsymbol{\omega}_i) e^{\lambda(-\boldsymbol{\omega}_i \cdot \mathbf{d}_r - 1)}, \tag{14}$$

where $L(\mathbf{x}, \boldsymbol{\omega}_i) = C(\mathbf{r})$ is the NeRF ray color (Eq. 3), representing the incident light color hitting $\mathbf{x}$ along direction $\boldsymbol{\omega}_i$. $R \in (0, 1)$ is the reflectance index determined by viewing direction d and normal $N_\phi(\mathbf{x})$, which enables the system to simulate the Fresnel effect on the water. To approximate the integral in rendering equation [9], we adopt the sigma-point method [15, 33] and sample 5 rays from the position $\mathbf{x}$, including reflection direction $\mathbf{d}_r$ and nearby four rays. In short, ClimateNeRF simulates Fresnel effect, glossy reflection, and wave dynamics.]

Fresnel Effect When the light hits the water's surface, the amount of reflection and transmission is determined by the incident and normal directions and is described by the Fresnel effect. The angle between normal and incident rays is denoted by θ_i, and the angle between normal and refracted ray in water is θ_t. According to Snell's Law: $r_i\theta_i = r_t\theta_t$, where $r_i = 1$ is the refraction index of air and $r_t = 1.33$ is the refraction index of water in our experiments, which is also consistent with real-world water properties. Next, the reflectance R in Eq. 14 is computed by:

$$R = \frac{R_s + R_p}{2}, \quad R_s = \left[\frac{\sin(\theta_t - \theta_i)}{\sin(\theta_t + \theta_i)}\right]^2, \quad R_p = \left[\frac{\tan(\theta_t - \theta_i)}{\tan(\theta_t + \theta_i)}\right]^2 \tag{15}$$

where R_s and R_p are the reflectances for s-polarized and p-polarized light, respectively. Modeling the Fresnel effect in our flood simulation pipeline makes the water far from the camera (larger θ_i) have higher R and looks more like a mirror. The water nearby (smaller θ_i) has lower R and shows watercolor, enhancing the simulation's realism.

Refraction Effect Refraction occurs when light transports from one medium to another. Modeling such an effect helps simulate clear water. With a ray direction, surface normal, and refraction index r_t of the water, ClimateNeRF calculates the refraction direction $\boldsymbol{\omega}_t$ according to Snell's Law, and retrieves color with volume rendering along $\boldsymbol{\omega}_t$. Next, we can simulate underwater scenes following [25].

$$\mathbf{c}_\phi = (1 - R)(t_c(\mathbf{x})L(\mathbf{x}, \boldsymbol{\omega}_t) + (1 - t_c(\mathbf{x})\mathbf{c}_w) + R \sum_{i=1}^{5} L(\mathbf{x}, \boldsymbol{\omega}_i) e^{\lambda(-\boldsymbol{\omega}_i \cdot \mathbf{d}_r - 1)}, \tag{16}$$

where $t_c(\mathbf{x}) = e^{-\beta_c D(x,\boldsymbol{\omega}_t)}$ and $D(x, \boldsymbol{\omega}_t)$ denotes the geometry depth alone refraction direction. Consequently, ClimateNeRF can simulate reflection and refraction, simulating clear and muddy water with controllable parameter β_c.

1.2. Snow Simulation

For any point $\mathbf{x}$ in the space, we calculate the snow's density of $\mathbf{x}$ in a particle-based manner. We first figure out a set of N particles as metaballs' centers $\{\mathbf{x}_i^{(p)}\}_{i=1}^N$ with densities $\{\sigma_i^{(p)}\}_{i=1}^N$ and metaball radius $\{R_i^{(p)}\}_{i=1}^N$ around $\mathbf{x}$. Then we sum up the densities calculated by kernel function $\mathbf{K}(r, R, \sigma_o)$.

$$\begin{aligned} \sigma_{\text{snow}}(\mathbf{x}) &= \sum_{i=1}^{N} \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)}), \\ \text{where } \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)}) &= \mathbf{K}(\|\mathbf{x} - \mathbf{x}_i^{(p)}\|_2, R_i^{(p)}, \sigma_i^{(p)}) V_g(\mathbf{x}_i^{(p)}, \boldsymbol{\omega}_s), \end{aligned} \tag{17}$$

where $\sigma_i^{(p)}$ is defined by weights during volume rendering 3 for σ_θ of F'_θ. More details are shown in Section 1.3. We denote $V_g(\mathbf{x}_i^{(p)}, \boldsymbol{\omega}_s)$ as the transparency along snow falling direction $\boldsymbol{\omega}_s$ retrieved from a pre-trained visibility network based on pre-trained NeRF's geometry to simulate snow occlusions. When training the visibility network, we perturb snow-falling directions within a small angle to soften snow boundaries. During rendering, we identify snow surface by a threshold τ_{snow} and a hyperparameter a:

$$O_\phi(\mathbf{x}; F'_\theta) = \frac{1}{1 + e^{-a(x\sigma_{\text{snow}} - \tau_{\text{snow}})}} \sigma_{\text{snow}}, \tag{18}$$

The BRDF of snow particles is set as spatially-varying diffuse color $\mathbf{c}_\phi(\mathbf{x}_i^{(p)})$ close to pure white multiplied by the average illumination of the scene. Furthermore, since the snow is semi-transmissive, the subsurface scattering effect [19] will light the snow's shadowed part. To simulate such effect, we leverage warp lighting function [8] $\Phi(\mathbf{n}_\mathbf{K}, \mathbf{n}_l, \gamma_\Phi)$ based on normalized surface normal $\mathbf{n}_\mathbf{K}$, light vector $\mathbf{n}_l$ and hyperparameter γ_Φ. We use 2D shadow predictions [5] and bake shadow confidence $V_s(\mathbf{x}) \in [0, 1]$ on pre-trained NeRF's geometry using L_2 loss. For an arbitrary point $\mathbf{x}$ in space, the color of point $\mathbf{x}$ is a weighted sum of $\{\mathbf{c}_i^{(p)} \in \mathbb{R}^1\}_{i=1}^N$ based on kernel function:

$$\mathbf{c}_\phi(\mathbf{x}) = \frac{\sum_{i=1}^N \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)}) \frac{\mathbf{c}_i^{(p)} + \mathbf{c}_0}{1 + \mathbf{c}_0} (\beta_s V_s(\mathbf{x}_i^{(p)}) + V_{s0})}{\sum_{i=1}^N \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)})} \Phi(\mathbf{n}_\mathbf{K}(\mathbf{x}), \mathbf{n}_l, \gamma_\Phi) \, , \tag{19}$$

where $\Phi(\mathbf{n}_\mathbf{K}(\mathbf{x}), \mathbf{n}_l, \gamma_\Phi) = \frac{\mathbf{n}(\mathbf{x}) \cdot \mathbf{n}_l + \gamma_\Phi}{1 + \gamma_\Phi}$ and $\frac{\mathbf{c}_i'^{(p)} + \mathbf{c}_0}{1 + \mathbf{c}_0}$ is used to approximate a high albedo for snow and $\mathbf{c}_0$ is a hyperparameter. To recast shadow on snow, we let β_s and V_{s0} in Eq. 19 be hyperparameters. See Figure 11 for example. Surface normal values of metaballs are still calculated in a gradient-based manner. Moreover, we stylize the scene to match the color tones of different weather conditions, as shown in Fig. 12.

1.3. Extensions

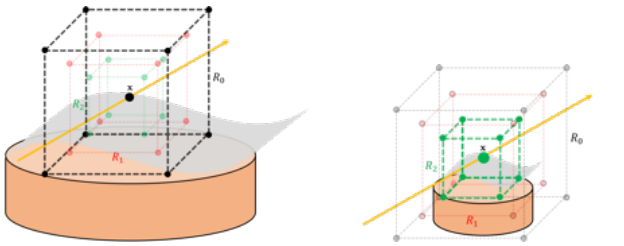
Bake Editing When simulating physical entities, especially snow, rendering will be time-consuming if we straightly let snow fall from the sky and do collision detection. Moreover, a lack of supervision in a bird's eye view makes depth estimation for rays cast from the sky inaccurate. To mitigate the aforementioned issues, we fit the distribution of metaballs' densities and colors in a new model and fetch them in a particle-based manner. The new model outputs high densities where metaballs locate. We use $w_i = T_i(1 - \exp(-\sigma_i \delta_i))$ in Eq. 3 for surface detection since $w(\mathbf{x}) \in [0, 1]$ is close to 1 when $\mathbf{x}$ is close to surfaces. To identify surfaces where snow accumulates, we incorporate surface normals $\mathbf{n}$ and vertical axis $\mathbf{n}_\perp$ to figure out metaballs' density weights: $w_i^{(p)} = \frac{1}{1 + e^{-a'(\mathbf{n}_i \cdot \mathbf{n}_\perp - \cos(\theta_0))}} w_i$ where θ_0 is a hyperparameter. We then bake $w_i^{(p)}$ into a new model G_{ϕ_w}:

$$w_{\phi_w}^{(p)}(\mathbf{x}) = G_{\phi_w}(\mathbf{x}) \tag{20}$$

We also bake the gray scale [27] of $\mathbf{c}_\theta$ from F_θ into a new model $G_{\phi_\mathbf{c}}$ to capture an approximation for light intensities and shadows:

$$\mathbf{c}_{\phi_\mathbf{c}}^{(p)}(\mathbf{x}) = G_{\phi_\mathbf{c}}(\mathbf{x}) \tag{21}$$

Then, we leverage the pre-trained G_{ϕ_w}, $G_{\phi_\mathbf{c}}$ and do voxel sampling to fetch $\{\sigma_i^{(p)}\}_{i=1}^N$ and $\{\mathbf{c}_i^{(p)}\}_{i=1}^N$ from 8 vertices. Also, to automatically alter metaballs' radiuses according to the size of the surface, we sample nested grids with different side



(a) Sample near a large surface (b) Sample near a tiny surface

Figure 2: **Visualization of voxel+nest sampling for $N_n = 3$.** Instead of storing metaball information in point cloud, we approximate metaball's center density distributions with G_{ϕ_w}. Our voxel+nest sampling helps capture different geometric level of details [30], which makes large metaballs dominate at flat surfaces, and put smaller snow balls on thin structures.

lengths defined in a geometric progression. Moreover, we define metaballs' radiuses by girds' side lengths. Hence, Eq. 17 and Eq. 19 can be rewritten as:

$$\sigma_{\text{snow}}(\mathbf{x}) = \sum_{i=1}^{8N_n} \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)}); \quad \mathbf{c}_\phi(\mathbf{x}) = \frac{\sum_{i=1}^{8N_n} \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)}) \frac{\mathbf{c}_i^{(p)} + \mathbf{c}_0}{1+\mathbf{c}_0} (\beta_s V_s(\mathbf{x}_i^{(p)}) + V_{s0})}{\sum_{i=1}^{8N_n} \sigma_{\mathbf{K}}(\mathbf{x}, \mathbf{x}_i^{(p)})} \Phi(\mathbf{n}_{\mathbf{K}}(\mathbf{x}), \mathbf{n}_l, \gamma_\Phi), \tag{22}$$

where N_n is the number of nests. We calculate density $\sigma^{(p)}$ and albedo color $\mathbf{c}^{(p)}$ for metaball centered at $\mathbf{x}^{(p)}$ by $\sigma^{(p)} = w_{\phi_w}^{(p)}(\mathbf{x}^{(p)})\sigma_0$ and $\mathbf{c}^{(p)} = \mathbf{c}_{\phi_\mathbf{c}}^{(p)}(\mathbf{x})$ where σ_0 is a hyperparameter. See Fig. 2 for a visualization of this sampling strategy. If stylization is done on the scene, we leverage the stylized model F'_θ and finetune the $G_{\phi_\mathbf{c}}$ to match new illumination conditions while remaining G_{ϕ_w} intact since F'_θ shares the same spatial information with F_θ.

Anti-Aliasing When rendering with simulation, the high-frequency normal map changes on the physical entity surface would lead to an aliasing effect. To alleviate such artifacts, we can render four times larger images with higher resolution and perform anti-aliasing downsampling to the original resolution.

2. Ablation Study

To justify our design choices, we perform an ablation study of flood simulation, and the results are shown in Fig 3. Specifically, we report the simulation results without certain technical components depicted in Sec. 1.1 of the main paper.

Fig 3 shows that all components are essential for realism. For example, vanishing point detection [13] makes the water plane follow gravity direction; wave simulation adds ripples to the water surface; the Fresnel effect makes the water reflectance view-dependent and physically plausible; the Glossy effect mimics realistic microfacet water surfaces with ripples; anti-aliasing removes far-away high-frequency noises. In short, all components contribute to the realism of the simulation.

We also perform an ablation study on snow simulation to validate our approximate scattering rendering in Fig. 4. We compare 1) pure white metaballs with spatial variant colors, 2) metaballs in a fully diffuse model, and 3) our full simulation. Results demonstrate that our choice provides a more realistic rendering of accumulated snow.

To identify how scene geometry improvements benefit simulations, the ablation study of various technical components is illustrated in Fig. 5. For example, appearance embeddings mitigate the floaters caused by varying exposures in training views, and sky loss reduces blobs in the sky. Results show that our improvements in geometry reconstructions help simulate dense snow covering in both foreground and background, thus offering more visually pleasing results.

3. Controllability

We further demonstrate that ClimateNeRF is highly controllable during the simulation process. In Fig. 6, our method simulate different colors of smog and flood, water clearness, varying spatial frequency of water ripples, and distinct heights of accumulated snow. The results show that our simulation framework is highly controllable by the users. Consequently, scientists can use this framework to simulate accurate climate conditions depending on the projected climate in the future and visualize the consequences corresponding to different actions taken by policymakers and the general public.

(a) Inaccurate plane geometry (b) No wave simulation (c) No Fresnel effect

(d) No glossy effect (e) No anti-aliasing (f) Full simulation

Figure 3: **Ablation Study of Flood Simulation.** (a) Without accurate plane geometry estimation with vanishing point detection [13], the water surface deviates from gravity direction. (b) The surface is perfect planar without wave simulation, which is not natural. (c) Fresnel effect makes the water far from the camera (with a larger incident angle) have higher reflectance, and is consistent with physical rules. (d) The glossy effect makes the reflection more blurry and realistic (e) There is much high-frequency noise around the water border without an anti-aliasing trick. (f) Our full flood simulation results.

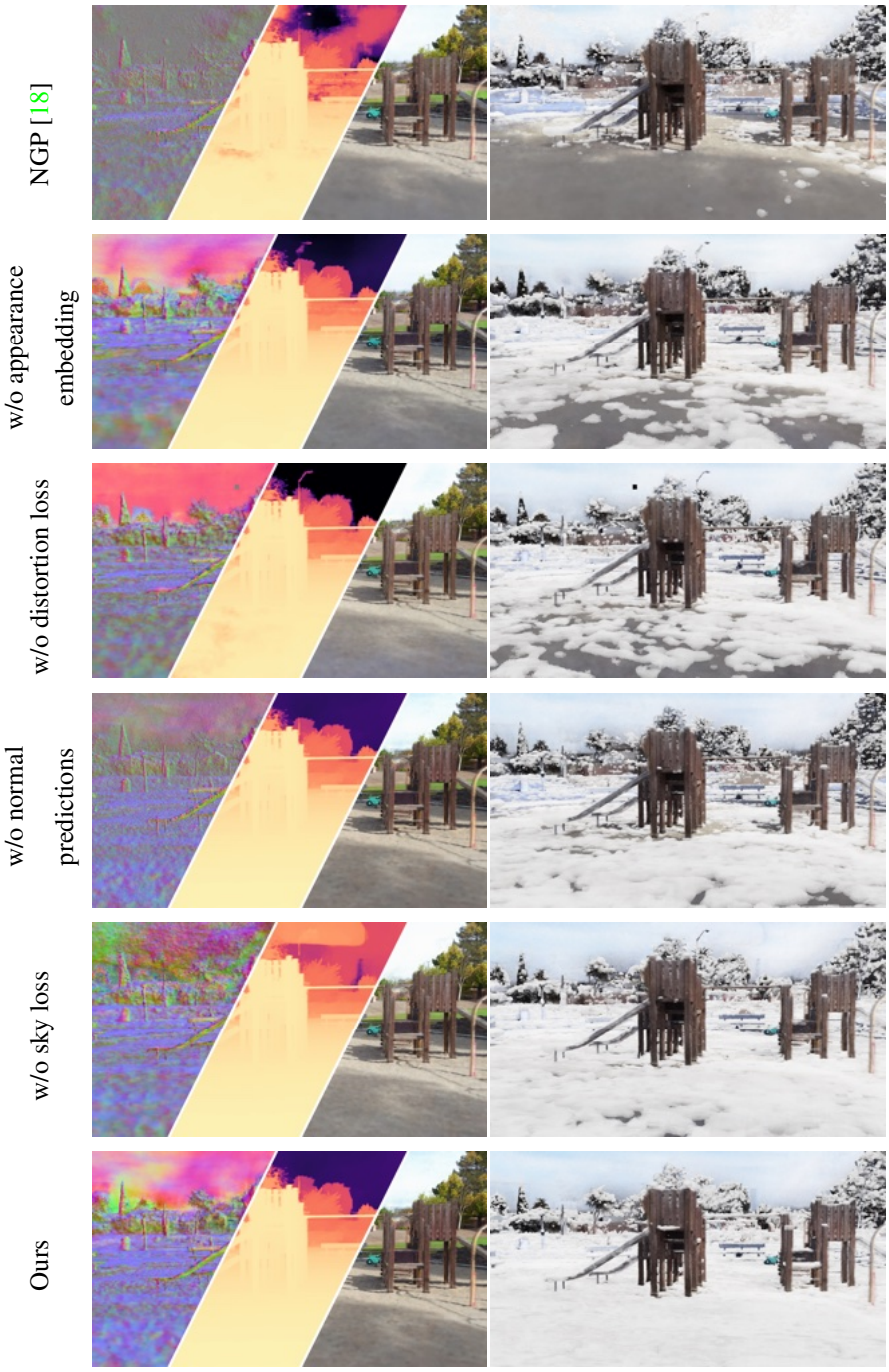
Pure White Snow Only spatial variant colors Only diffuse model Full model

Figure 4: **Ablation Study for Snow Simulation** Though spatial variant colors capture local illumination conditions, they fail to offer a sense of depth for snow. Moreover, the diffuse model cannot simulate snow's scattering effects.

4. Qualitative Results

We demonstrate more qualitative results is Fig. 7, Fig. 8, Fig. 9, Fig. 10, and Fig. 11. For smog scene images in Fig. 7, ClimateGAN [24] generates visually plausible results but fails to provide sharp boundaries, and 3D stylization attempts to change the surface texture but makes the images overall darker. Our method simulates realistic visibility reduction effects caused by smog, thanks to the geometry reconstruction.

The flood images are shown in Fig. 8. ClimateGAN++ [24] cannot reconstruct realistic reflection on the water surface, Stable Diffusion [23] synthesize realistic water appearance but also produce random objects (e.g., cars, signs) in the scene, which is not consistent across views. ClimateNeRF simulates realistic reflection and water ripples while being view-consistent. This is better demonstrated in the supp video and website. In Fig. 9, we compare with other NeRF-editing baselines. We first edit training images with ClimateGAN [24] or 2D stylization [11] and optimize the neural radiance field. While these



(a) Normal/Depth/RGB (d) Snow

Figure 5: **Geometry Improvement for Simulation.** Appearance embeddings [14] and distortion loss [1] mitigate floaters or incomplete geometries while normal prediction loss [32] smooths predicted normals and leads a well interpolation in background(see the snow coverage in the background). Sky loss alleviates 'blobs' in the sky.

baseline methods produce more view-consistent rendering, they either generate severe floating artifacts to compensate for the image inconsistency or cannot perform geometric editing, which sabotages realism. We also compare our FastPhotoStyle [11] based stylization method with Artistic Radiance fields [38]. As shown in Fig. 13, we sustain more appearance details from the original scene.

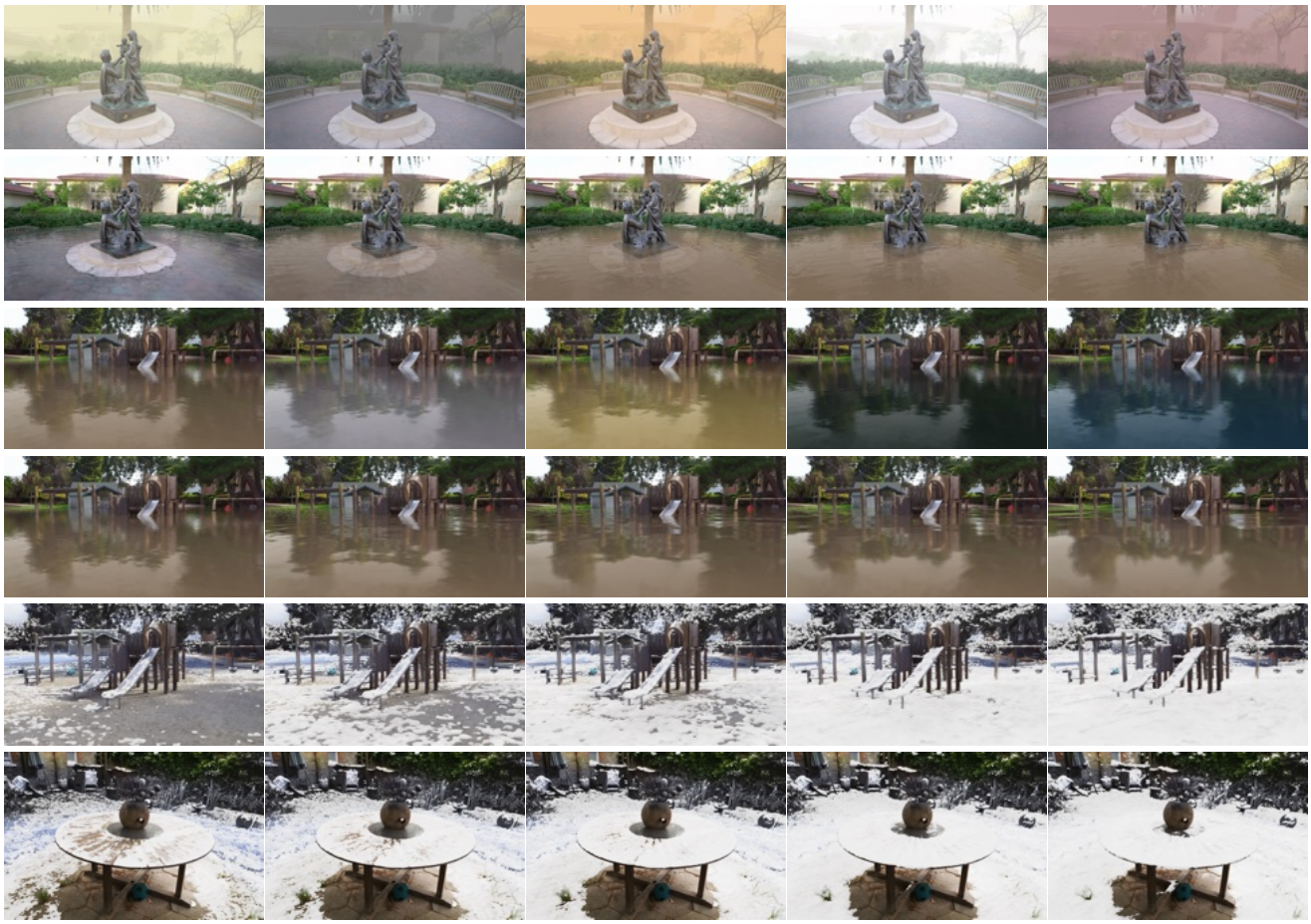


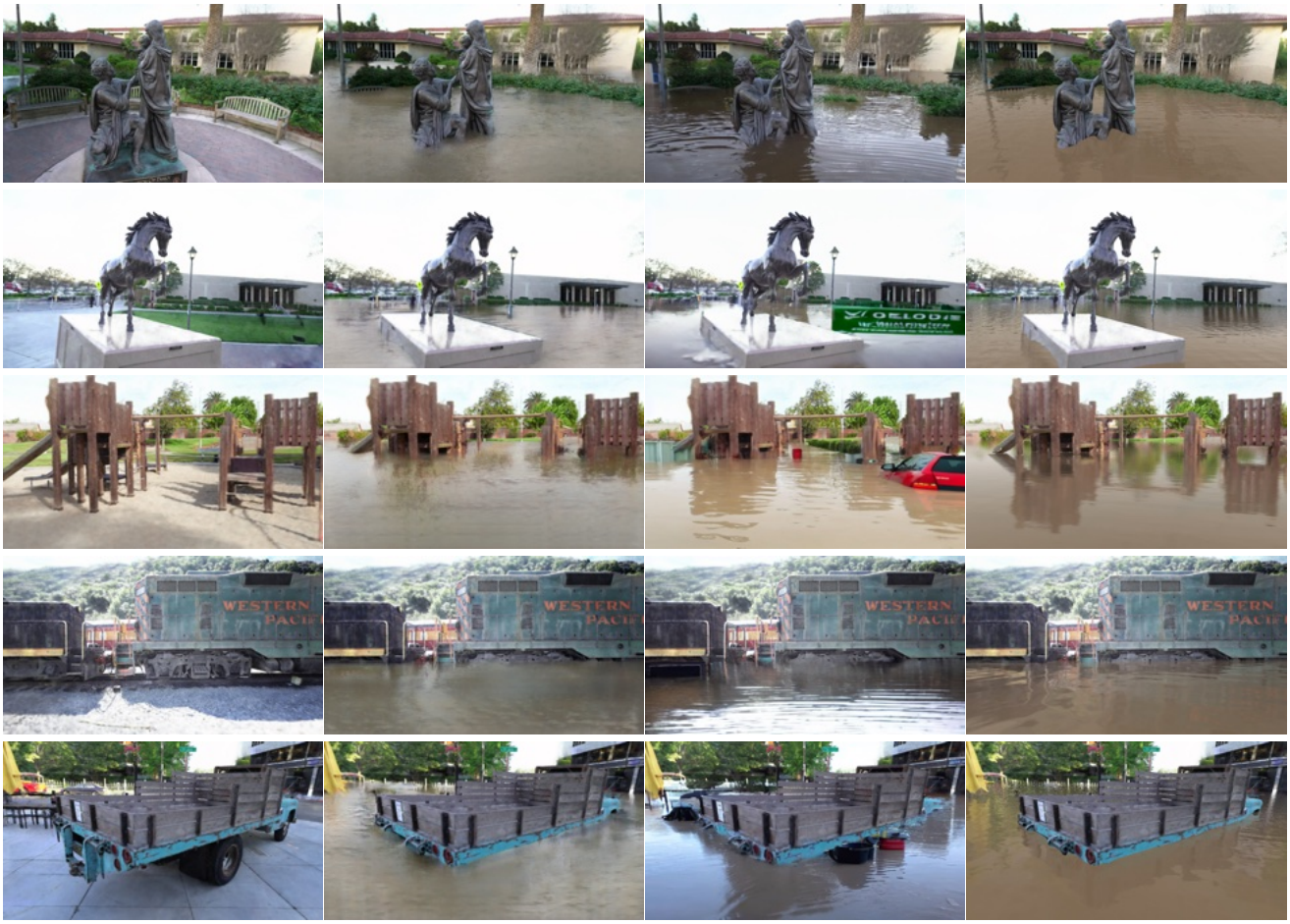
Figure 6: **Controllable Climate Simulation.** From the top row to the bottom row, we control (1) smog color (2) water clearness (3) watercolor (4) spatial frequency of the wave. The frequency decreases from left to right. (5) snow height. We control snow height by adjusting the threshold density τ_{snow}.

5. Quantitative Results

No automatic quantitative score can holistically evaluate the quality of our weather-simulated movies. In this project, we evaluate the synthesized videos with the state-of-the-art video quality assessment model UVQ [34] and report the results in Table 1. The score ranges between interval $(1, 5)$, where 1 indicates the lowest quality and 5 indicates the highest quality. As the table shows, our smog simulation outperforms all other baselines, while it does not win Stable Diffusion [23] in flood simulation and ClimateGAN [24] in snow simulation. That being said, UVQ prefers sharp videos instead of measuring holistic realism. As shown in Fig. 10, baselines get a better quality score despite providing low-quality snow simulation results, suggesting UVQ might not be a good metric for our task. Hence, despite demonstrating UVQ [34] results, we want to emphasize that such metrics mainly focus on measuring the amount of low-level degradation (such as blurriness and noise), which cannot faithfully reproduce human evaluation on realism. Having a good video quality score on simulation remains an open topic.

Original | ClimateGAN [24] | 3D Stylization | Ours

Figure 7: **Smog simulation comparison.**



Original | ClimateGAN++ [24] | Stable Diffusion [23] | Ours

Figure 8: **Flood simulation comparison.**

Original | ClimateGAN [24]+NeRF | 3D Stylization | Ours

Figure 9: **NeRF editing for flood simulation.** We tested the *ClimateGAN image + NeRF training* baseline, but the results are blurry due to inconsistent inputs. *3D stylization* baseline is a NeRF trained on stylized images.

Original | Swapping Autoencoder [20] | 3D stylization | Ours

Figure 10: **Snow simulation comparison.**

Figure 11: **Simulation on Urban Driving Scenes.**

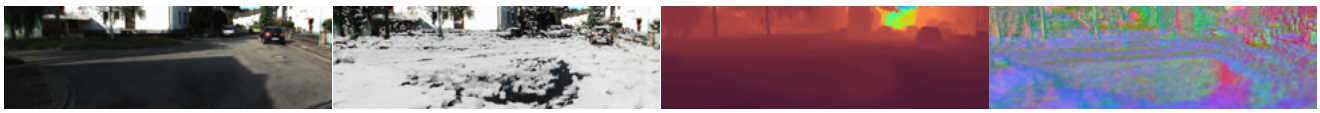
(a) Original NeRF rendering (b) Original NeRF + snow (c) NeRF with stylization (d) NeRF with stylization + snow

Figure 12: **Stylization.** Performing snow simulation with the original NeRF model leads to incompatible scene appearance (e.g., snow on green vegetations). We address this issue by first finetuning the appearance of the NeRF model to match the style of the provided style image. Our snow simulation on the stylized NeRF model shows visually more appealing results.

Style Image from [36] Original ARF [38] 3D stylization

Figure 13: **Comparison with Artistic Radiance Fields [38]**

Figure 14: **View Consistency Comparison.** We show flood simulation in Playground scene and time step t increases from left to right. ClimateGAN++ [24] (Top) cannot generate realistic reflection, Stable Diffusion [23] (Middle) synthesizes different objects in each views. Ours (Bottom) simulate photorealistic and consistent reflection and ripples.



(a) Original NeRF rendering (b) Climate NeRF rendering (c) Depth rendering (d) Surface normal rendering

Figure 15: **Limitations.** Snow simulation on KITT-360 [12] dataset fails to cover a shadowed road due to wrong geometry.

		Smog			**Flood**				**Snow**		
	Original	GAN	3D Style	Ours	GAN	GAN++	Diffusion	Ours	GAN	3D Style	Ours
Family	3.434	3.426	3.425	**3.437**	3.408	3.413	3.416	**3.424**	**3.434**	**3.434**	3.431
Horse	3.426	3.422	3.422	**3.432**	3.409	3.412	3.416	**3.422**	**3.435**	3.424	3.421
Playground	3.409	3.402	3.403	**3.412**	3.400	3.402	**3.409**	3.404	**3.427**	3.412	3.415
Train	3.406	3.396	**3.407**	**3.407**	3.401	3.402	**3.410**	3.407	**3.417**	3.411	3.408
Truck	3.425	3.424	3.424	**3.431**	3.404	3.403	**3.424**	3.413	**3.431**	3.424	3.416

Table 1: **Video Quality Assessment.** We evaluate the video quality with Google's Universal Video Quality (UVQ) model [34].

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 2, 7

[2] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *ICCV*, 2021. 2

[3] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. *ECCV*, 2022. 1

[4] Xingyu Chen, Qi Zhang, Xiaoyu Li, Yue Chen, Ying Feng, Xuan Wang, and Jue Wang. Hallucinated neural radiance fields in the wild. In *CVPR*, 2022. 2

[5] Zhihao Chen, Lei Zhu, Liang Wan, Song Wang, Wei Feng, and Pheng-Ann Heng. A multi-task mean teacher for semi-supervised shadow detection. In *CVPR*, pages 5611–5620, 2020. 4

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *NeurIPS*, 2014. 2

[8] Simon Green. Real-time approximations to subsurface scattering. *GPU Gems*, 1:263–278, 2004. 4

[9] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, 1986. 3

[10] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG*, 2017. 2

[11] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. A closed-form solution to photorealistic image stylization. In *ECCV*, 2018. 6, 7

[12] Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE TPAMI*, 2022. 2, 13

[13] Xiaohu Lu, Jian Yaoy, Haoang Li, Yahui Liu, and Xiaofeng Zhang. 2-line exhaustive searching for real-time vanishing point estimation in manhattan world. In *WACV*. IEEE, 2017. 5, 6

[14] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 1, 7

[15] Henrique MT Menegaz, João Y Ishihara, Geovany A Borges, and Alessandro N Vargas. A systematization of the unscented kalman filter theory. *IEEE Transactions on automatic control*, 2015. 3

[16] 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. 1

[17] Thomas Müller. tiny-cuda-nn, 4 2021. 2

[18] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 1, 7

[19] Tomoyuki Nishita, Hiroshi Iwasaki, Yoshinori Dobashi, and Eihachiro Nakamae. A modeling and rendering method for snow by using metaballs. In *Computer Graphics Forum*, 1997. 4

[20] Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. *NeurIPS*, 2020. 11

[21] Chen Quei-An. ngp_pl: a pytorch-lightning implementation of instant-ngp, 2022. 1, 2

[22] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE TPAMI*, 2020. 2

[23] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 6, 8, 10, 13

[24] Victor Schmidt, Alexandra Sasha Luccioni, Mélisande Teng, Tianyu Zhang, Alexia Reynaud, Sunand Raghupathi, Gautier Cosne, Adrien Juraver, Vahe Vardanyan, Alex Hernandez-Garcia, et al. Climategan: Raising climate change awareness by generating images of floods. *ICLR*, 2022. 6, 8, 9, 10, 13

[25] Advaith Venkatramanan Sethuraman, Manikandasriram Srinivasan Ramanagopal, and Katherine A Skinner. Waternerf: Neural radiance fields for underwater scenes. *in arXiv*, 2022. 3

[26] Pratul P Srinivasan, Boyang Deng, Xiuming Zhang, Matthew Tancik, Ben Mildenhall, and Jonathan T Barron. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021. 2

[27] Michael Stokes. A standard default color space for the internet-srgb. *http://www. color. org/contrib/sRGB. html*, 1996. 4

[28] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 1, 2

[29] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Improved direct voxel grid optimization for radiance fields reconstruction. *in arXiv*, 2022. 2

[30] Towaki Takikawa, Joey Litalien, Kangxue Yin, Karsten Kreis, Charles Loop, Derek Nowrouzezahrai, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *CVPR*, pages 11358–11367, 2021. 5

[31] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, 2003. 1

[32] 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. *CVPR*, 2022. 1, 7

[33] Eric A Wan and Rudolph Van Der Merwe. The unscented kalman filter for nonlinear estimation. In *Proceedings of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000. 3

[34] Yilin Wang, Junjie Ke, Hossein Talebi, Joong Gon Yim, Neil Birkbeck, Balu Adsumilli, Peyman Milanfar, and Feng Yang. Rich features for perceptual quality assessment of ugc videos. In *CVPR*, 2021. 8, 13

[35] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. 1

[36] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2014. 12

[37] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. *in arXiv*, 2022. 2

[38] Kai Zhang, Nick Kolkin, Sai Bi, Fujun Luan, Zexiang Xu, Eli Shechtman, and Noah Snavely. Arf: Artistic radiance fields. *ECCV*, 2022. 7, 12

[39] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 1