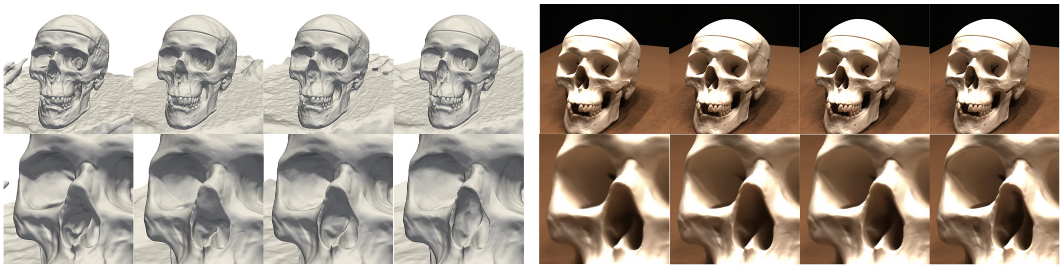
# Volume Rendering of Neural Implicit Surfaces
## Supplementary material

## A Additional results

### A.1 Sampling ablation study

Figure A1 depicts an ablation study that we performed for evaluating the sampling algorithm, by replacing it with other sampling strategies. We compared with the following alternatives: Uniform stands for an uniform sampling of 256 samples along each ray ; 2-networks denotes a hierarchical sampling using coarse and fine networks as suggested in [6]; our sampling algorithm as suggested in section 3.4 where the maximal number of iterations is set to 1 or 5 (the choice in the paper). We note that using 1 iteration resembles one level of sampling as in the hierarchical sampling of [6].

As can be seen in Figure A1 (see also reported Chamfer and PSNR scores) alternative sampling procedures lead to lower accuracy and some artifacts in the geometry (see e.g., head top, and nose areas) and rendering (see e.g., salt and pepper noise and over-smoothed areas).

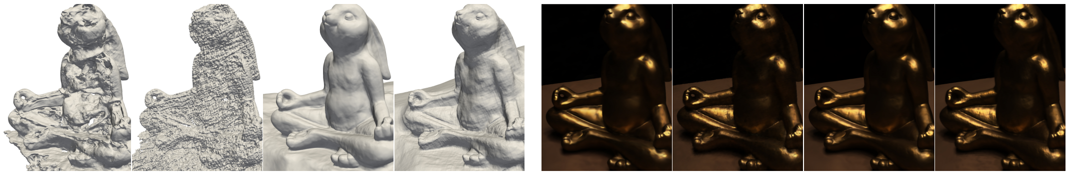


Uniform 1.19 2-networks 0.96 Ours$_{(1\ \text{iter.})}$ 0.89 **Ours$_{(5\ \text{iters.})}$ 0.7** Uniform 30.2 2-networks 31.32 Ours$_{(1\ \text{iter.})}$ 31.27 **Ours$_{(5\ \text{iters.})}$ 31.5**

Figure A1: Sampling ablation study: the left side depicts the geometry reconstruction results with the corresponding Chamfer distances, whereas the right side presents rendering results with their corresponding PSNRs.

### A.2 Positional encoding ablation

We perform an ablation study on the level of positional encoding used in the geometry network. We note that VolSDF use level 6 positional encoding, while NeRF use level 10. In Figure A2 we show the DTU Bunny scene with positional encoding levels 6 and 10 for both NeRF and VolSDF; we report both PSNR for the rendered images and Chamfer distance to the learned surfaces. Note that higher positional encoding improves specular highlights and details of VolSDF but adds some undesired noise to the reconstructed surface.



NeRF - PE 6 3.02 NeRF - PE 10 2.88 **VolSDF** - PE 6 1.08 **VolSDF** - PE 10 2.37 NeRF - PE 6 31.38 NeRF - PE 10 31.68 **VolSDF** - PE 6 31.49 **VolSDF** - PE 10 31.8

Figure A2: Positional Encoding (PE) ablation. We note the tradeoff between smoother geometry with PE level 6 versus detailed rendering and slightly higher noise with PE level 10. Note that in both cases NeRF fails to decompose correctly density and radiance field.

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

## A.3 Multi-view 3D reconstruction

Figures A3 and A4 show additional qualitative results for DTU and BlendedMVS datasets, respectively. We further provide a **video** with qualitative rendering results of the learned geometries and radiance fields from simulated camera paths (including novel views); note that VolSDF rendering alleviates NeRF's salt and pepper artifacts, while producing higher fidelity geometry approximation.

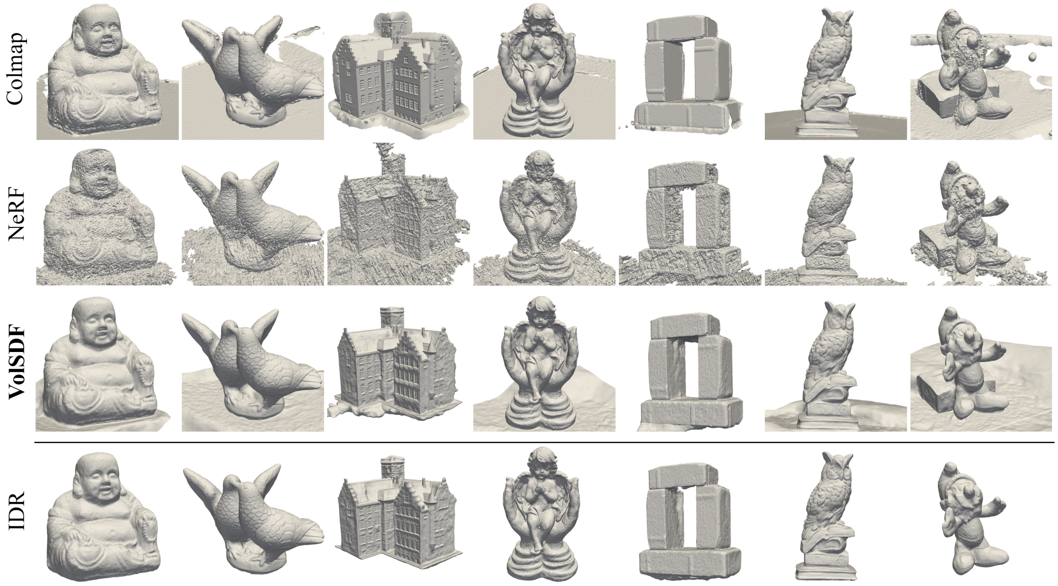


Figure A3: Qualitative results for the reconstructed geometries of objects from the DTU dataset.

## A.4 Limitations

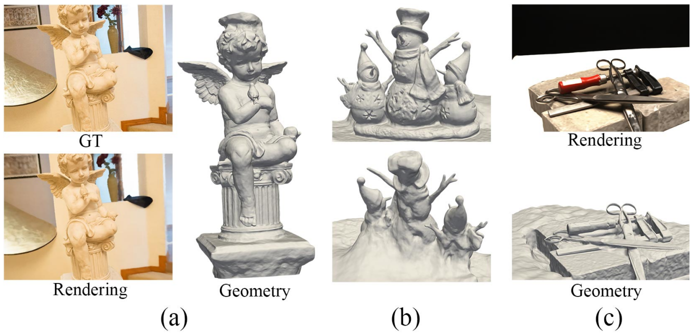


Figure A5: Failure cases, see details in the text.

Figure A5 shows the main failure cases of our method: First, we observe that the geometry for unseen regions is not well defined and can be completed arbitrarily by the algorithm, see e.g., the angle statue head top in (a), and the reverse side of the snow man statue in (b). Second, homogeneous texture-less areas are hard to reconstruct faithfully, see e.g., the white background desk in (c). These limitations can potentially be alleviated with the addition of extra assumptions such as minimal surface reconstruction [3] and/or defining background color or predefined geometry (e.g., a plane).

## A.5 Rendering comparison

Figure 2 in the main paper shows a comparison of NeRF and VolSDF rendering for the same scene using the same random sampling strategy of the inverse opacity function. For NeRF, replacing random sampling with regularly-spaced sampling introduces different artifacts as shown in Figure A6. In contrast, VolSDF produces consistent rendering results regardless of the sampling strategy.

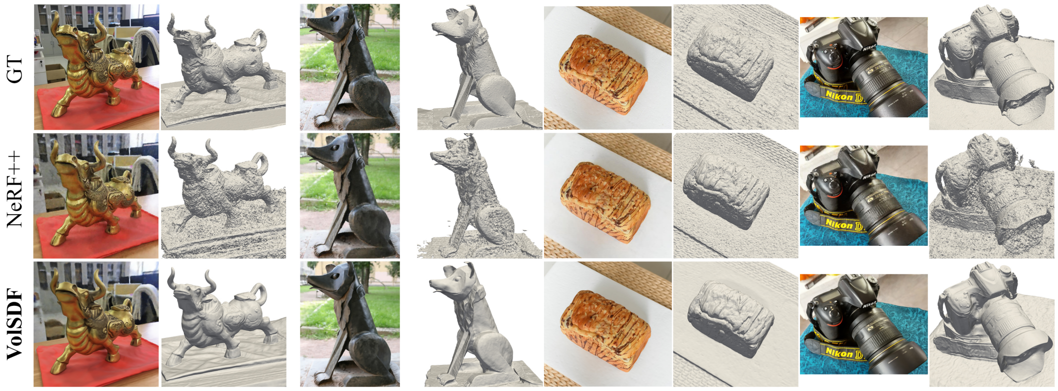
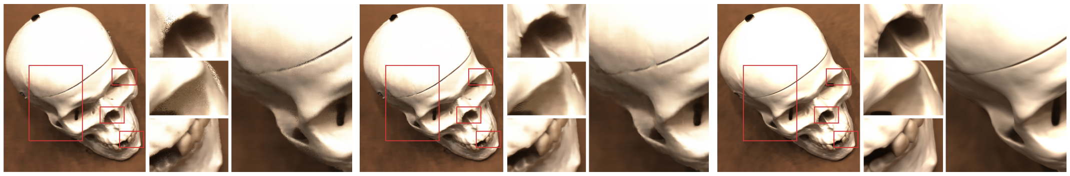


Figure A4: Qualitative results sampled from the BlendedMVS dataset. For each scan we present a visualization of a rendered image and the 3D geometry.

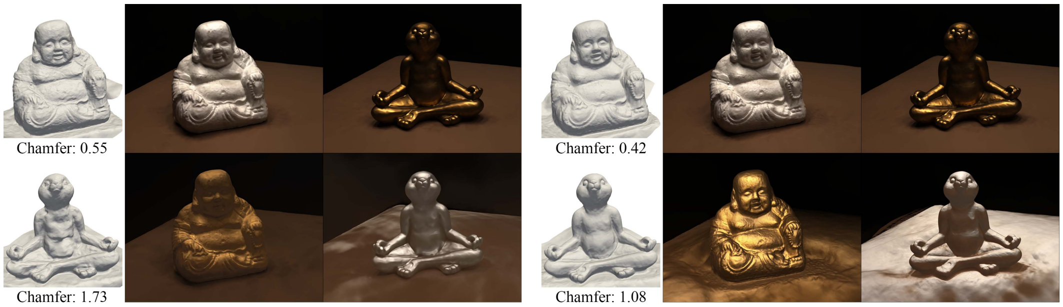


NeRF *random sampling* NeRF *regularly-spaced sampling* **VolSDF**

Figure A6: Comparison of random versus regularly-spaced sampling in NeRF. VolSDF produces consistent results in both cases.

### A.6 Normal dependency in radiance field

As presented in [8], incorporating the zero level set normal in surface rendering improves both reconstruction quality and disentanglement of geometry and radiance. Incorporating the level set's normal has similar effect in the radiance field representation of VolSDF; see Figure A7 where we compare using radiance field with and without normal dependency.



VolSDF *without normal* **VolSDF**

Figure A7: Geometry and radiance field disentanglement is successful with normal information incorporated in the radiance field, and partially fails without it.

### A.7 Disentanglement of geometry and appearance

Figure A8 shows additional results for unsupervised disentanglement of geometry and radiance field (switching the radiance fields of three independently trained scenes). Note how the material and light from one scene is gracefully transferred to the two other scenes. We further provide in the supplementary **video** the 360 degrees camera path for one of the swaps.

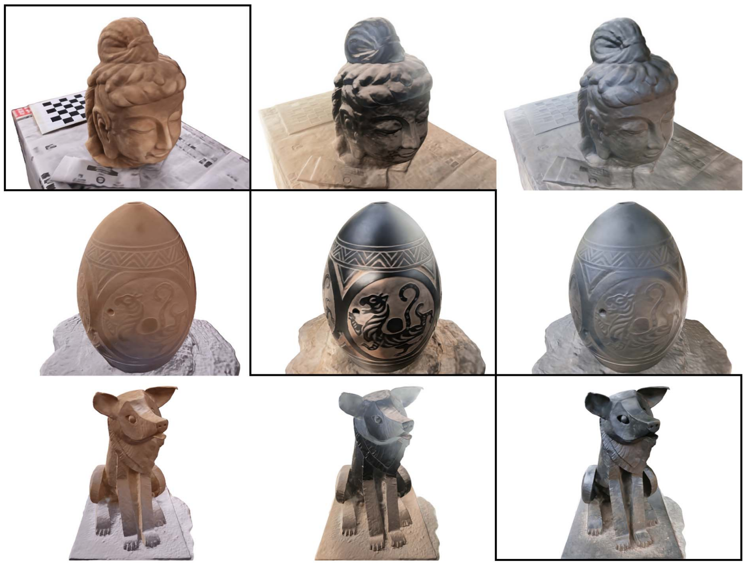


Figure A8: Additional results of geometry and radiance field disentanglement in BlendedMVS. The diagonal depicts the trained geometry and radiance, while the off-diagonal demonstrates post-training mix and match of geometries and radiance fields.

# B Experiments setup

## B.1 Camera Normalization

We used the known camera poses to shift the coordinate system, locating the object at the origin. This is done using a least squares solution for the intersection point of all camera principal axes. Let $R_{max}$ be the maximal camera center norm, we further apply a global scale of $\frac{3}{R_{max}*1.1}$ to place all camera centers inside a sphere of radius 3, centered at the origin.

## B.2 Datasets

**DTU** We used the formal evaluation script to measure the Chamfer $l_1$ distance between each reconstructed object to its corresponding ground truth point cloud. For fare comparison with [8], we used their masks (with a dilation of 50 pixels) to remove non visual hull parts from each output of both ours and [6]. Specifically, from an output mesh, we remove a 3D point (and its adjacent triangles) if it is projected to a zero pixel label in any image. Finally, we used the largest connected component mesh for evaluation. The results for COLMAP and IDR are taken from [8].

**BlendedMVS** We used the ground truth meshes supplied by the authors to evaluate the Chamfer $l_1$ distances from the output surfaces. For each mesh we evaluated the largest connected surface component above the ground plane. To measure the Chamfer $l_1$ distance we used $100K$ random point samples from each surface.

## B.3 Additional implementation details

**Architecture and hyper-parameters** As described in section 3.5, our model architecture consists of two MLP networks, where the geometry network $\boldsymbol{f}_\varphi$ has 8 layers with hidden layers of width 256, and a single skip connection from the input to the 4th layer. The geometry MLP gets a 3D position, $\boldsymbol{x}$, and outputs the SDF value and an extra feature vector $\boldsymbol{z}$ of size 256, i.e., $\boldsymbol{f}_\varphi(\boldsymbol{x}) = (d(\boldsymbol{x}), \boldsymbol{z}(\boldsymbol{x})) \in \mathbb{R}^{1+256}$. We initialized $\boldsymbol{f}_\varphi$ using the geometric initialization presented in [1], so that $d$ produces an approximated SDF to the unit sphere. The radiance field network $L_\psi$ receives a 3D position $\boldsymbol{x}$, the normal to its level set, $\boldsymbol{n}$, and the (geometry) feature vector $\boldsymbol{z}$, as well as the view direction $\boldsymbol{v}$, and outputs a RGB value. It consists of 4 layers of width 256, and ends with a Sigmoid activation for providing valid color values. In addition to these two networks we have an additional scalar parameter $\beta$ that is initialized to $0.1$.

To capture the high frequencies of the geometry and radiance field, we exploit Positional Encoding (PE) [6] for the position $\boldsymbol{x}$ and view direction $\boldsymbol{v}$ in the geometry and radiance field. For the position $\boldsymbol{x}$ we use 6 PE levels, and for the the view direction $\boldsymbol{v}$ we use 4 PE levels, as in [8]. The influence of different levels of frequencies applied to the position is presented in the Section A.2.

**Training details** We trained our networks using the ADAM optimizer [2] with a learning rate initialized with $5\mathrm{e}{-4}$ and decayed exponentially during training to $5\mathrm{e}{-5}$. Each model was trained for $100K$ iterations on scenes from the DTU dataset, and for $200K$ on scenes from the BlendedMVS dataset. Training one model from the DTU dataset takes approximately 12 hours. Training was done on a single Nvidia V-100 GPU, using PYTORCH deep learning framework [7].

**Mesh extraction** We use the Marching Cubes algorithm [4] for extracting each surface mesh from the zero level set of the signed distance function defined by $d(\boldsymbol{x})$.

**Modeling the background** To satisfy the assumption that all rays, including rays that do not intersect any surface, are eventually occluded (i.e., $O(\infty) = 1$), we model our SDF as:

$$d_\Omega(\boldsymbol{x}) = \min\{d(\boldsymbol{x}), r - \|\boldsymbol{x}\|_2\}, \tag{A1}$$

where $r$ denotes a predefined scene bounding sphere (in out experiments $r = 3$, see Section B.1 for more details about camera normalization). During the rendering process, the maximal depth for ray samples, denoted by $M$ in Sec. 3.2, is set to $2r$. Intuitively, this modification can be considered as modeling the background using a 360 degrees panorama on the scene boundaries.
For modeling more complex backgrounds as in the blendedMVS dataset, we follow the parametrization presented in NeRF++ [10]: the volume outside a radius 3 sphere (the background) is modeled using an additional NeRF network, predicting the background point density and radiance field. Each $3D$ background point $(x_b, y_b, z_b)$ is modeled using a $4D$ representation $(x', y', z', \frac{1}{r})$ where $\|(x', y', z')\| = 1$ and $(x_b, y_b, z_b) = r \cdot (x', y', z')$. For rendering the background component of a pixel, we use 32 points calculated by sampling $\frac{1}{r}$ uniformly in the range $(0, \frac{1}{3}]$. More details regarding this parametrization (referred as the "inverted sphere parametrization") can be found in [10].

### B.4 Baselines

**NeRF** For running NeRF [6] we used a slightly modified version of the PYTORCH implementation suggested by [9]. Following the official implementation code[1] of NeRF we extracted the 50 level set of the learned density as NeRF geometry reconstruction.

**NeRF++** We used the official code[2] of NeRF++ [10]. All the cameras are normalized inside a unit sphere, and we train two networks (one foreground network in normal coordinates, one background network in inverse sphere coordinates). For foreground, we used 64 uniform samples and 128 hierarchical samples; for background, we used 32 uniform and 64 hierarchical samples. Similar to NeRF experiments, we extracted the 50 level set of the foreground density as reconstruction.

## C Proofs and additional lemmas

As $d_\Omega$ is not everywhere differentiable we need to bound the Lipschitz constant of $\sigma$, rather than its derivative. The Lipshcitz constant is defined as a constant $K_i > 0$ so that $|\sigma(\boldsymbol{x}(s)) - \sigma(\boldsymbol{x}(t))| \leq K_i|s - t|$, for all $s, t \in [t_i, t_{i+1}]$.

**Theorem 1.** *The Lipshcitz constant of the density $\sigma$ within a segment $[t_i, t_{i+1}]$ satisfies*

$$K_i \leq \frac{\alpha}{2\beta} \exp\left(-\frac{d_i^\star}{\beta}\right), \text{ where } d_i^\star = \min_{\substack{s \in [t_i, t_{i+1}] \\ \boldsymbol{y} \notin B_i \cup B_{i+1}}} \|\boldsymbol{x}(s) - \boldsymbol{y}\|, \tag{A2}$$

*and* $B_i = \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x}(t_i)\| < |d_i|\}$, $d_i = d_\Omega(\boldsymbol{x}(t_i))$.

---

[1] `https://github.com/bmild/nerf`

[2] `https://github.com/Kai-46/nerfplusplus`

The constant $d_i^*$ can be computed explicitly using $d_i, d_{i+1}, t_i, t_{i+1}$ as described in the next proposition.

**Proposition 1.** *The lower distance bound $d_i^*$ can be computed using the following formulas:*

$$d_i^\star = \begin{cases} 0 & |d_i| + |d_{i+1}| \leq \delta_i \\ \min\{|d_i|, |d_{i+1}|\} & \left||d_i|^2 - |d_{i+1}|^2\right| \geq \delta_i^2 \\ h_i & \textit{otherwise} \end{cases}, \tag{A3}$$

$d_i = d_\Omega(\boldsymbol{x}(t_i))$, $h_i = \frac{2}{\delta_i}\sqrt{s(s-\delta_i)(s-|d_i|)(s-|d_{i+1}|)}$, $s = \frac{1}{2}(\delta_i + |d_i| + |d_{i+1}|)$, $\delta_i = t_{i+1} - t_i$.

*Proof of Theorem 1.* We denote by $\Phi_\beta$ the Probability Density Function (PDF) of the Laplace distribution (the CDF of which is given in equation 3),

$$\frac{d}{dt}\Psi_\beta(s) = \Phi_\beta(s) = \frac{1}{2\beta}\exp\left(-\frac{|s|}{\beta}\right). \tag{A4}$$

Let $s, t \in [t_i, t_{i+1}]$,

$$\begin{aligned} |\sigma(\boldsymbol{x}(s)) - \sigma(\boldsymbol{x}(t))| &= \alpha\,|\Psi_\beta(-d_\Omega(\boldsymbol{x}(s))) - \Psi_\beta(-d_\Omega(\boldsymbol{x}(t)))| \\ &\leq \alpha\,|d_\Omega(\boldsymbol{x}(s)) - d_\Omega(\boldsymbol{x}(t))| \max_{t\in[t_i,t_{i+1}]} |\Phi_\beta(-d_\Omega(\boldsymbol{x}(t)))| \\ &\leq \alpha\,|s-t|\,\Phi_\beta\left(\min_{t\in[t_i,t_{i+1}]} |d_\Omega(\boldsymbol{x}(t))|\right) \\ &\leq \alpha\,|s-t|\,\Phi_\beta(d_i^*) \end{aligned}$$

where the first equality is by definition of $\sigma$ (equation 2), the first inequality uses the fact that the Lipschitz constant of a continuously differential function is the maximum of (absolute value of) its derivative. The second inequality uses the following Lemma A1 and the fact that $\Phi_\beta$ is symmetric w.r.t. zero, positive, and monotonically decreasing at $[0,\infty)$. The last inequality can be justified by noting that by definition of the distance function $\mathcal{M} \subset (B_i \cup B_{i+1})^c$ and therefore $\min_{t\in[t_i,t_{i+1}]} |d_\Omega(\boldsymbol{x}(t))| \geq d_i^*$. ☐

*Proof of Proposition 1.* We wish to find the distance of the two sets: $A = [\boldsymbol{x}(t_i), \boldsymbol{x}(t_{i+1})]$ and $B = \partial((B_i \cup B_{i+1})^c)$. That is $d_i^* = \min_{\boldsymbol{a}\in A, \boldsymbol{b}\in B} \|\boldsymbol{a} - \boldsymbol{b}\|$. Since these two sets are compact, their cartesian product is compact, and the minimum is achieved. Denote this minimum $(\boldsymbol{a}^*, \boldsymbol{b}^*)$, where $\boldsymbol{a}^* \in A$ and $\boldsymbol{b}^* \in B$.

First, if $B_i \cap B_{i+1} = \emptyset$ then $d_i^* = 0$. This case is characterized by $|d_i| + |d_{i+1}| \leq \delta_i$. Henceforth we assume $|d_i| + |d_{i+1}| > \delta_i$.

Now we consider several cases. If $\boldsymbol{a}^* = \boldsymbol{x}(t_i)$ the minimal distance is $|d_i|$, and similarly the distance is $|d_{i+1}|$ if $\boldsymbol{a}^* = \boldsymbol{x}(t_{i+1})$. Otherwise, if $\boldsymbol{a}^* \in (\boldsymbol{x}(t_i), \boldsymbol{x}(t_{i+1}))$ Lagrange Multipliers imply that $\boldsymbol{a}^* - \boldsymbol{b}^* \perp \boldsymbol{v}$, namely is orthogonal to the ray. In this case we have two options to consider: First, $\boldsymbol{b}^* \in \bar{B}_i \cap \bar{B}_{i+1}$, where $\bar{B}_i$ is the closure of the ball $B_i$. In this case Heron's formula provides the minimal distance $h_i$, as the height of the triangle $\Delta(\boldsymbol{x}(t_i), \boldsymbol{b}^*, \boldsymbol{x}(t_{i+1}))$.

Otherwise $\boldsymbol{b}^* \in \bar{B}_i$ or $\boldsymbol{b}^* \in \bar{B}_{i+1}$. Lets assume the former (the latter is treated similarly). Lagrange multipliers imply that $\boldsymbol{b}^* - \boldsymbol{a}^* \| \boldsymbol{b}^* - \boldsymbol{x}(t_i)$. This necessarily means that $\boldsymbol{a}^* = \boldsymbol{x}(t_i)$, leading to a contradiction with the assumption that $\boldsymbol{a}^* \in (\boldsymbol{x}(t_i), \boldsymbol{x}(t_{i+1}))$.

To check if $\boldsymbol{a}^* \in (\boldsymbol{x}(t_i), \boldsymbol{x}(t_{i+1}))$ and $\boldsymbol{b}^* \in \bar{B}_i \cap \bar{B}_{i+1}$ it is enough to check that the angles at $\boldsymbol{x}(t_i)$ and $\boldsymbol{x}(t_{i+1})$ in the triangle $\Delta(\boldsymbol{x}(t_i), \boldsymbol{b}^*, \boldsymbol{x}(t_{i+1}))$ are acute, or equivalently that $\left||d_i|^2 - |d_{i+1}|^2\right| < \delta_i^2$ (the latter can be shown with the Law of Cosines).

☐

**Lemma A1.** *The distance function $d = d_\Omega$ to a compact surface $\mathcal{M}$ is Lipschitz with constant* 1, *i.e.,* $|d(\boldsymbol{x}) - d(\boldsymbol{y})| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$, *for all* $\boldsymbol{x}, \boldsymbol{y}$.

*Proof of Lemma A1.* Let $\boldsymbol{x}^* \in \mathcal{M}$ be the closest point to $\boldsymbol{x}$, and $\boldsymbol{y}^* \in \mathcal{M}$ the closets point to $\boldsymbol{y}$. If $d(\boldsymbol{x})d(\boldsymbol{y}) < 0$, then $|d(\boldsymbol{x}) - d(\boldsymbol{y})| = \|\boldsymbol{x} - \boldsymbol{x}^*\| + \|\boldsymbol{y} - \boldsymbol{y}^*\| \leq \|\boldsymbol{y} - \boldsymbol{z}\| + \|\boldsymbol{z} - \boldsymbol{x}\|$, for all $\boldsymbol{z} \in \mathcal{M}$. However, from mean value theorem we know there exists $\boldsymbol{z} \in \mathcal{M}$ on the straight line $[\boldsymbol{x}, \boldsymbol{y}]$, and therefore $|d(\boldsymbol{x}) - d(\boldsymbol{y})| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$. If $d(\boldsymbol{x})d(\boldsymbol{y}) \geq 0$, then $|d(\boldsymbol{y}) - d(\boldsymbol{x})| = ||d(\boldsymbol{y})| - |d(\boldsymbol{x})||$. Now,

$$\begin{aligned} |d(\boldsymbol{y})| &= \|\boldsymbol{y} - \boldsymbol{y}^*\| \leq \|\boldsymbol{y} - \boldsymbol{x}^*\| \\ &\leq \|\boldsymbol{y} - \boldsymbol{x}\| + \|\boldsymbol{x} - \boldsymbol{x}^*\| \\ &= \|\boldsymbol{x} - \boldsymbol{y}\| + |d(\boldsymbol{x})|. \end{aligned}$$

Therefore $|d(\boldsymbol{y})| - |d(\boldsymbol{x})| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$. The other direction follows by switching the roles of $\boldsymbol{x}, \boldsymbol{y}$. We proved $|d(\boldsymbol{x}) - d(\boldsymbol{y})| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$. □

*Derivation of equation 12.* For a single interval $[t_k, t_{k+1}]$,

$$\left| \int_{t_i}^{t_{i+1}} \sigma(\boldsymbol{x}(s))ds - \delta_i \sigma_i \right| \leq \int_{t_i}^{t_{i+1}} |\sigma(\boldsymbol{x}(s)) - \sigma(\boldsymbol{x}(t_i))| \, ds \leq K_i \frac{\delta_i^2}{2}$$

Plugging the Lipschitz bound from Theorem 1 (equation A3) provides that the error in equation 9 for $t \in [t_k, t_{k+1}]$ can be bounded by

$$|E(t)| \leq \widehat{E}(t) = \frac{\alpha}{4\beta} \left( \sum_{i=1}^{k-1} \delta_i^2 e^{-\frac{d_i^\star}{\beta}} + (t - t_k)^2 e^{-\frac{d_k^\star}{\beta}} \right). \tag{A5}$$

□

**Theorem 2.** *For $t \in [0, M]$, the error of the approximated opacity $\hat{O}$ can be bounded as follows:*

$$\left| O(t) - \widehat{O}(t) \right| \leq \exp\left(-\widehat{R}(t)\right) \left( \exp\left(\widehat{E}(t)\right) - 1 \right) \tag{A6}$$

*Proof of Theorem 2.* This bound is derived as follows:

$$\begin{aligned} \left| O(t) - \widehat{O}(t) \right| &= \left| \exp\left(-\widehat{R}(t)\right) - \exp\left(-\int_0^t \sigma(\boldsymbol{x}(s))ds\right) \right| = \exp\left(-\widehat{R}(t)\right) |1 - \exp\left(-E(t)\right)| \\ &\leq \exp\left(-\widehat{R}(t)\right) \left(\exp(\widehat{E}(t)) - 1\right), \end{aligned}$$

where the last inequality is due to the inequality $|1 - \exp(r)| \leq \exp(|r|) - 1$ and the bound $|E(t)| \leq \widehat{E}(t)$ from equation 12. □

**Lemma 1.** *Fix $\beta > 0$. For any $\epsilon > 0$ a sufficient dense sampling $\mathcal{T}$ will provide $B_{\mathcal{T},\beta} < \epsilon$.*

*Proof of Lemma 1.* Noting that $\exp(-\widehat{R}(t)) \leq 1$, equation 15 leads to

$$\begin{aligned} B_{\mathcal{T},\beta} \leq \left(\exp\left(\widehat{E}(t_n)\right) - 1\right) &= \left( \exp\left( \frac{\alpha}{4\beta} \sum_{i=1}^{n-1} \delta_i^2 e^{-\frac{d_i^\star}{\beta}} \right) - 1 \right) \\ &\leq \left( \exp\left( \frac{\alpha}{4\beta} \sum_{i=1}^{n-1} \delta_i^2 \right) - 1 \right). \end{aligned}$$

Lastly we note that the inner sum satisfies $\sum_i \delta_i^2 \leq M \max_i \delta_i$, and in turn this implies that dense sampling can achieve arbitrary low error bound. □

**Lemma 2.** *Fix $n > 0$. For any $\epsilon > 0$ a sufficiently large $\beta$ that satisfies*

$$\beta \geq \frac{\alpha M^2}{4(n-1)\log(1+\epsilon)} \tag{A7}$$

*will provide $B_{\mathcal{T},\beta} \leq \epsilon$.*

*Proof of Lemma 2.* Assuming sufficiently large $\beta$ that satisfies equation A7, then following the derivation of Lemma 1 and assuming equidistance sampling, i.e. $\delta_i = \frac{M}{n-1}$, leads to

$$\begin{aligned} B_{\mathcal{T},\beta} &\leq \left( \exp\left( \frac{\alpha}{4\beta} \sum_{i=1}^{n-1} \delta_i^2 \right) - 1 \right) = \left( \exp\left( \frac{\alpha M^2}{4(n-1)\beta} \right) - 1 \right) \\ &\leq \left( \exp\left( \log(1+\epsilon) \right) - 1 \right) = \epsilon \end{aligned}$$

□

# D Numerical integration

Standard numerical quadrature in volume rendering is based on the rectangle rule [5], which we repeat here for completeness. Quantities with ˆ denote approximated quantities. For a set of discrete samples $\mathcal{S} = \{s_i\}_{i=1}^m$, $0 = s_1 < s_2 < \ldots < s_m = M$ we let $\delta_i = s_{i+1} - s_i$, $\sigma_i = \sigma(\boldsymbol{x}(s_i))$, and $L_i = L(\boldsymbol{x}(s_i), \boldsymbol{n}(s_i), \boldsymbol{v})$. Applying the rectangle rule (left Riemann sum) to approximate the integral in equation 7 we have

$$\begin{aligned} I(\boldsymbol{c}, \boldsymbol{v}) &= \int_0^\infty L(\boldsymbol{x}(t), \boldsymbol{n}(t), \boldsymbol{v})\tau(t)dt \\ &= \int_0^M L(\boldsymbol{x}(t), \boldsymbol{n}(t), \boldsymbol{v})\tau(t)dt + \int_M^\infty L(\boldsymbol{x}(t), \boldsymbol{n}(t), \boldsymbol{v})\tau(t)dt \\ &\approx \sum_{i=1}^{m-1} \delta_i \tau(s_i) L_i, \end{aligned} \tag{A8}$$

where we assumed that $M$ is sufficiently large so that the integral over the segement $[M, \infty)$ is negligible, and the integral over $[0, M]$ is approximated with the rectangle rule, and $\tau(s_i) = \sigma_i T(s_i)$ as given in equation 6. The rectangle rule is applied yet again to approximate the transparency $T$:

$$T(s_i) \approx \hat{T}(s_i) = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right).$$

To provide a discrete probability distribution $\hat{\tau}$ that parallels the continuous one $\tau$, [5] defines $p_i = \exp(-\sigma_i \delta_i)$ that can be interpreted as the probability that light passes through the segment $[s_i, s_{i+1}]$. Then, using $\hat{T}(s_i) = \prod_{j=1}^{i-1} p_i$, and the approximation $\delta_i \sigma_i \approx (1 - \exp(-\delta_i \sigma_i)) = 1 - p_i$, the approximation in equation A8 takes the form

$$I(\boldsymbol{c}, \boldsymbol{v}) \approx \hat{I}(\boldsymbol{c}, \boldsymbol{v}) = \sum_{i=1}^{m-1} \left[(1 - p_i) \prod_{j=1}^{i-1} p_j\right] L_i = \sum_{i=1}^{m-1} \hat{\tau}_i L_i$$

where the discrete probability is given by

$$\hat{\tau}_i = (1 - p_i) \prod_{j=1}^{i-1} p_j \quad \text{for } 1 \leq i \leq m-1, \text{ and } \hat{\tau}_m = \prod_{j=1}^{m-1} p_j$$

establishing that $\hat{\tau}_i$, $i = 1, \ldots, m$ is indeed a discrete probability. Note that although $\hat{\tau}_m$ is not used in $\hat{I}$ above, we added it for implementation ease. Since we use a bounding sphere (see equation A1) and $M$ is chosen so that $s_m$ is always outside this sphere, $\hat{\tau}_m \approx 0$.

# References

[1] M. Atzmon and Y. Lipman. Sal: Sign agnostic learning of shapes from raw data. *arXiv preprint arXiv:1911.10414*, 2019.

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

[3] Y. Lipman. Phase transitions, distance functions, and implicit neural representations. *arXiv preprint arXiv:2106.07689*, 2021.

[4] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[5] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[8] L. Yariv, Y. Kasten, D. Moran, M. Galun, M. Atzmon, B. Ronen, and Y. Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *Advances in Neural Information Processing Systems*, 33, 2020.

[9] L. Yen-Chen. Nerf-pytorch. `https://github.com/yenchenlin/nerf-pytorch/`, 2020.

[10] K. Zhang, G. Riegler, N. Snavely, and V. Koltun. Nerf++: Analyzing and improving neural radiance fields. *arXiv:2010.07492*, 2020.