6. Ray tracing algorithm pseudocode

Algorithm 1 Ray tracing algorithm

```
procedure RENDER($\mathbf{o}, \mathbf{d}$)                          ▷ ray parameters
    $t_0 \leftarrow 0$
    $i \leftarrow \mathrm{nn}(\mathbf{o})$                ▷ initial cell (nearest neighbour)
    $T \leftarrow 1$
    $\mathbf{C} \leftarrow \mathbf{0}$
    while $T > \epsilon$ do
        $x \leftarrow v_i$                         ▷ $v_i$: primal vertex of cell $i$
        $t_1 \leftarrow \infty$
        $i' \leftarrow \varnothing$
        for all $j \in \mathrm{N}(i)$ do  ▷ $\mathrm{N}(i)$: neighbours of cell $i$
            $x' \leftarrow v_j$                    ▷ $v_j$: primal vertex of cell $j$
            $(t_j, \text{front}) \leftarrow$ INTERSECT($\mathbf{o}, \mathbf{d}, x, x'$)
            if front and $(t_j < t_1)$ then
                $t_1 \leftarrow t_j$
                $i' \leftarrow j$
            end if
        end for
        $c \leftarrow \mathbf{c}_i$                           ▷ $\mathbf{c}_i$: color of cell $i$
        $\sigma \leftarrow \sigma_i$                        ▷ $\sigma_i$: density of cell $i$
        $\alpha \leftarrow 1 - e^{-\sigma(t_1 - t_0)}$
        $\mathbf{C} \leftarrow \mathbf{C} + T\,\alpha\,c$
        $T \leftarrow T\,(1 - \alpha)$
        $t_0 \leftarrow t_1$
        $i \leftarrow i'$
    end while
    return $\mathbf{C}$
end procedure
```

Figure 8. **Ray tracing –** Our ray tracing algorithm is simple, and based on the method proposed by Weiler et al. [57]. Unlike common algorithms for tracing triangle meshes and other unstructured scene representations, we do not require a hierarchical acceleration structure, and thus avoid the associated $O(\log(n))$ query operation.

7. Additional implementation details

Training. Our training pipeline uses the Adam [22] optimizer, and similarly to 3DGS, directly optimizes per-point position, density and view-dependent color via spherical harmonics of degree three. We use the softplus activation function with β=10 for density to constrain it within the $[0, \infty)$ range, while keeping smooth gradients. For the location of points, we start at a learning rate of $2e^{-4}$ and decay it using a cosine annealing scheduler to a final learning rate of $2e^{-6}$. For point density and spherical harmonics, we start at a learning rate of $1e^{-1}$ and $5e^{-3}$ respectively and decay it with a cosine annealing scheduler by a factor of 0.1. Following 3DGS [19], we optimize only the zero-order component of SH coefficients and the high-order coefficients with a warmup for the first 25% of the total training iterations. Similarly to [4, 21], after initialization and warm-up training, we gradually grow the number of Voronoi sites so that points are placed at useful locations. We progressively increase the number of points up to half the total training iterations, linearly increasing the number of points until the maximum desired number of points is obtained.

Voronoi optimization. We maintain an adjacency data structure throughout training, which defines the Voronoi cells for rendering. Whenever the primal vertex positions are changed we must update the adjacency by performing an incremental Delaunay triangulation. While much faster than a complete rebuild, the incremental update is still computationally expensive for large point sets, so we allow the optimizer to take multiple steps between mesh rebuilds. We start at a 1:1 ratio after each densification and increase to 1:100, as the frequency of discrete changes to the mesh decreases over time with the converging optimization. This strategy balances the overall speed of training with maintaining a relatively accurate mesh structure. All our experiments are optimized for 20k iterations, with the last 2k only updating radiance and density attributes while positions are frozen. This process takes, as an example, 77 minutes on the "bonsai" scene with an RTX 4090 GPU.

8. Per Scene metrics

Tables 3 and 4 summarize the error metrics collected for our evaluation of all considered techniques. These include results for both Mip-NeRF360 [2] and Deep Blending [14] scenes. However, 3DGRT [30] is excluded from per-scene comparisons as these values are not reported in the original paper, and the code is not publicly available.

9. Computation of positional gradients

As stated in Section 3, in a piecewise constant field, the gradient for the primal points $\mathbf{p}_i$ depends only on the segment widths δ_n. These segment widths are determined by the points where the ray intersects the boundaries of the cells. Specifically,δ_n is given by $\delta_n = \mathbf{t}_n - \mathbf{t}_{n-1}$, where $\mathbf{t}_{n-1}, \mathbf{t}_n$ represent the entry intersection and exit intersection of the ray with a given cell $\mathbf{c}_i$. Without loss of generality, we will derive the derivative for $\mathbf{t}_n$ with respect to the primal point $\mathbf{p}_i$.

We will first express the cell boundaries(hyperplanes) in terms of the primal points and then derive the intersections in terms of the primal points. Since our mesh is a Voronoi diagram, each cell boundary is a plane positioned at the midpoint between two neighboring primal points, $\mathbf{p}_i$ and $\mathbf{p}_j$. The normal to this face is the vector pointing from $\mathbf{p}_i$ to $\mathbf{p}_j$,

	3DGS [19]	Mip-Splatting [60]	3DGS-MCMC [21]	Plenoxels [45]	iNGP-Big [31]	MipNerf360 [2]	Ours
	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓
Room	30.63 / 0.91 / 0.27	31.74 / **0.93** / 0.27	**32.30** / **0.93** / **0.25**	27.59 / 0.84 / 0.42	29.69 / 0.87 / 0.26	**31.63** / **0.91** / 0.21	30.87 / **0.91** / **0.19**
Counter	28.70 / 0.91 / 0.24	**29.16** / **0.92** / 0.24	**29.16** / 0.91 / **0.23**	23.63 / 0.76 / 0.44	26.69 / 0.82 / 0.31	**29.55** / **0.89** / 0.20	28.59 / 0.88 / **0.19**
Bonsai	31.98 / 0.94 / 0.24	32.31 / **0.95** / **0.23**	**32.67** / **0.95** / **0.23**	24.67 / 0.81 / 0.40	30.69 / 0.91 / 0.21	**33.46** / **0.94** / 0.18	32.15 / 0.93 / **0.17**
Kitchen	30.32 / 0.92 / **0.14**	31.55 / 0.93 / 0.15	**32.23** / **0.94** / **0.14**	23.42 / 0.65 / 0.45	29.48 / 0.86 / 0.20	**32.23** / **0.92** / **0.13**	31.40 / 0.91 / **0.13**
Bicycle	25.25 / 0.77 / 0.23	25.72 / **0.78** / **0.19**	**26.06** / **0.78** / **0.19**	21.92 / 0.50 / 0.51	22.17 / 0.51 / 0.45	**24.37** / **0.69** / **0.30**	24.19 / 0.68 / 0.31
Garden	27.41 / 0.87 / 0.12	27.76 / **0.88** / **0.11**	**27.99** / 0.87 / **0.11**	23.49 / 0.61 / 0.39	25.07 / 0.70 / 0.26	**26.98** / 0.81 / 0.17	26.58 / **0.82** / **0.16**
Stump	26.55 / 0.78 / 0.24	26.94 / **0.79** / 0.21	**27.67** / 0.78 / **0.20**	20.66 / 0.52 / 0.50	23.47 / 0.59 / 0.42	**26.40** / **0.74** / **0.26**	25.48 / 0.71 / 0.29

Table 3. **Per-scene metrics –** PSNR, SSIM, and LPIPS scores for Mip-NeRF360 [2] scenes.

	3DGS [19]	Mip-Splatting [60]	3DGS-MCMC [21]	Plenoxels [45]	iNGP-Big [31]	MipNerf360 [2]	Ours
	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓	PSNR↑ / SSIM↑ / LPIPS↓
Dr Johnson	28.77 / **0.90** / 0.33	28.76 / **0.90** / **0.32**	**29.05** / 0.89 / 0.33	23.14 / 0.79 / 0.52	28.26 / 0.85 / 0.35	**29.14** / **0.90** / **0.24**	28.33 / /0.88 / 0.27
Playroom	30.04 / **0.91** / 0.32	30.17 / **0.91** / 0.33	**30.37** / 0.90 / **0.31**	22.98 / 0.80 / 0.50	21.67 / 0.78 / 0.43	**29.66** / **0.90** / **0.25**	29.56 / 0.89 / 0.26

Table 4. **Per-scene metrics –** PSNR, SSIM, and LPIPS scores for Deep Blending [14] scenes.

given by:

$$\mathbf{x} = \frac{\mathbf{p}_i + \mathbf{p}_j}{2} \tag{8}$$

$$\mathbf{n} = \mathbf{p}_j - \mathbf{p}_i \tag{9}$$

The intersection $\mathbf{t}_n$ of the ray with this boundary can be written as:

$$\mathbf{t}_n = \frac{(\mathbf{x} - \mathbf{o}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \tag{10}$$

where $\mathbf{o}$ is the ray origin, and $\mathbf{d}$ is the ray direction. Using the chain rule, we differentiate $\mathbf{t}_n$ with respect to $\mathbf{p}_i$:

$$\frac{d\mathbf{t}_n}{d\mathbf{p}_i} = \frac{\mathbf{n}/2 - (\mathbf{x} - \mathbf{o})}{\mathbf{d} \cdot \mathbf{n}} + \frac{(\mathbf{x} - \mathbf{o}) \cdot \mathbf{n}\, \mathbf{d}}{(\mathbf{d} \cdot \mathbf{n})^2} \tag{11}$$

$$= \frac{\mathbf{o} - \mathbf{p}_j + \mathbf{t}_n\, \mathbf{d}}{\mathbf{d} \cdot \mathbf{n}} \tag{12}$$

Following the same procedure, we can compute the derivative of $\mathbf{t}_{n-1}$ with respect to $\mathbf{p}_i$.

10. Scalability of traversal loop

While one could construct pathological inputs where cell valence (number of cells that share a face) grows arbitrarily large – potentially harming performance – such cases do not arise in our trained models, as illustrated in Figure 9 (left). As shown in Figure 9 (right), this trend holds even as the total number of cells increases, suggesting that resolution does not introduce excessive local complexity.

11. Point budget analysis

Figure 10 shows how reconstruction quality (PSNR) varies with the number of points used in the representation for the *bonsai* scene. The experiment was conducted over a fixed training schedule of 20,000 iterations for each setting, with the point budget up to 2 million points.

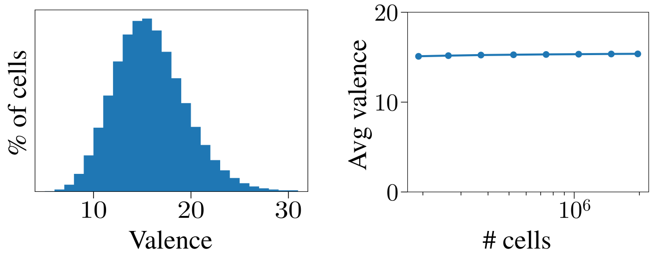

Figure 9. **Cell valence –** (left) High valence is rare in a trained model, with 99.9% of cells having valence below 30. Most cells have valence in the range of 12 to 18, which ensures that the number of neighboring faces remains bounded. This makes the search for the exit face during traversal nearly constant-time for every cell along a ray. (right) The average valence remains approximately constant across varying cell budgets, indicating that increased resolution does not significantly affect local connectivity complexity.

12. Detailed training time breakdown

Table 5 provides a detailed breakdown of the total training time for the bonsai scene, computed over 20k iterations, which results in a final point cloud containing 2 million points. These are computed with a batch size of 1 million rays and constitute the primary computational load during optimization.

13. Density slices

In Figure 11, we visualize a cross-sectional slice through our volumetric density field along the plane indicated by the red laser in the inset image. This slice reveals the internal structure of our representation, where high-density regions (shown in black) are concentrated on the surfaces of the 3D geometry, while the surrounding free space maintains near-zero density.

Stage	Total Time (s)	Time per iteration (ms)	Percentage (%)
Triangulation	1600	80.3	34.8
Initial cell	128	6.41	2.78
Forward pass	730	36.5	15.8
Backward pass	2150	108	46.7
Training time	4620	231	100

Table 5. **Detailed Training Time Breakdown** - Stage-wise breakdown of training time for the *bonsai* scene over 20,000 iterations. The table reports total time, average time per iteration, and percentage contribution for each stage.

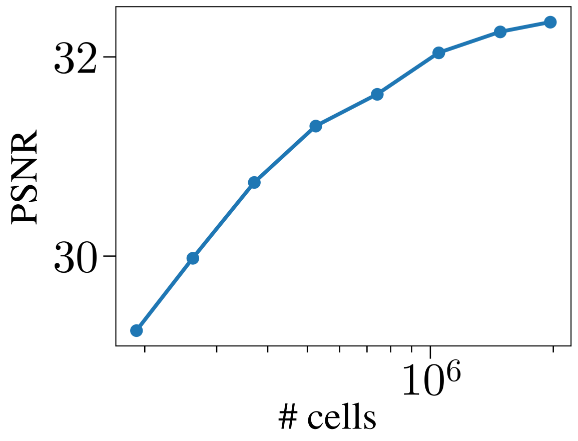

Figure 10. **Performance with varying point budget –** PSNR improves consistently as the number of points increases, demonstrating that more cells yield more accurate reconstructions. Results are shown for the *bonsai* scene with budgets ranging up to 2 million points.

14. Geometric quality

In Figure 12, we present qualitative visualizations of predicted depth maps on a variety of scenes from the *Mip-NeRF360* dataset. Each depth image includes an inset RGB view from the same scene for spatial context. The visualizations demonstrate that the model accurately captures fine scene geometry and preserves spatial consistency across complex environments. Depth discontinuities at object boundaries are well preserved, and smooth gradients are maintained across planar surfaces. These examples highlight the effectiveness of our method in producing detailed and geometrically consistent reconstructions across both indoor and outdoor scenes.

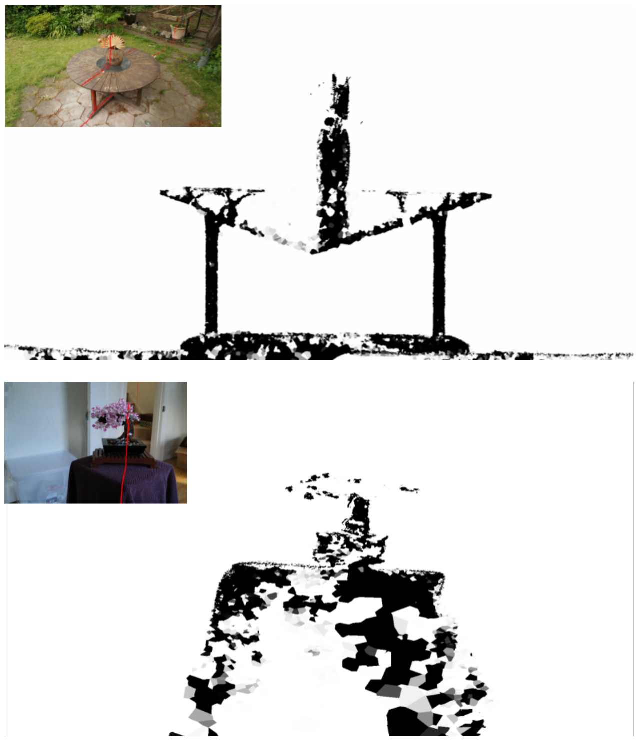

Figure 11. **Volumetric density slice –** Cross-sectional visualization of the volumetric density field for the *garden* scene. The slice is taken along the plane indicated by the red laser in the inset image. High-density regions (shown in black) correspond to surfaces of the 3D geometry, while empty space maintains near-zero density (white), highlighting the surface-localized nature of the representation.

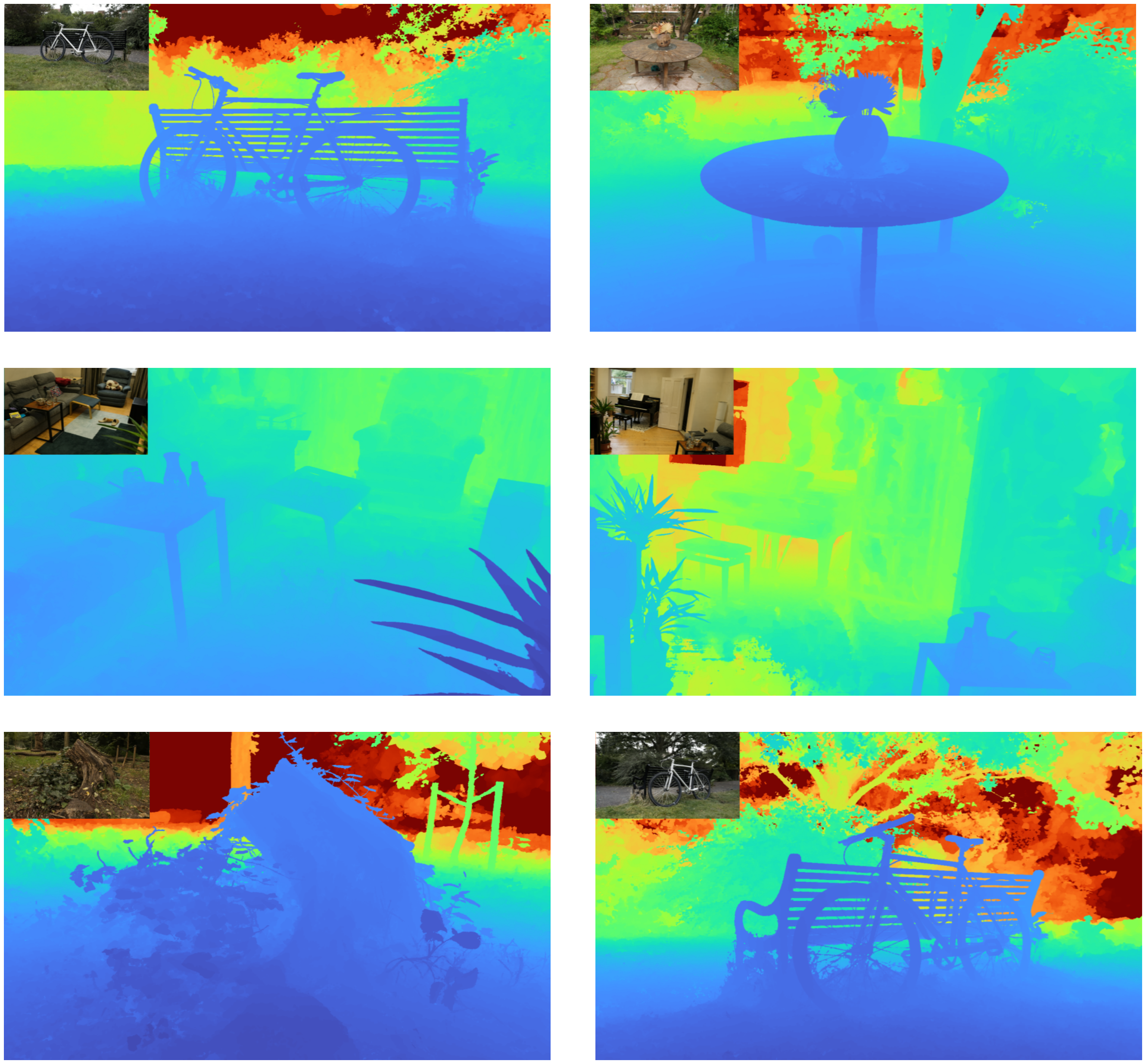

Figure 12. **Depth maps –** Qualitative visualizations of depth maps across diverse scenes from the *mipnerf360* dataset. Each pair shows the predicted depth (in color) with an inset of the corresponding RGB image for reference. Cooler colors (blue) indicate closer surfaces, while warmer colors (red/yellow) represent farther regions.