3D Neural Edge Reconstruction
Supplementary Material

Lei Li[1] Songyou Peng[1,2†] Zehao Yu[3,4] Shaohui Liu[1] Rémi Pautrat[1,6]
Xiaochuan Yin[5] Marc Pollefeys[1,6]
[1]ETH Zurich [2]MPI for Intelligent Systems, Tübingen [3]University of Tübingen
[4]Tübingen AI Center [5]Utopilot [6]Microsoft
neural-edge-map.github.io

A. More Method Details

A.1. Unbiased Density-based Edge Neural Rendering

The detailed derivation of $\hat{E}$ along the camera ray $\mathbf{r}$ in the main paper is given as

$$\begin{aligned}\hat{E}(\mathbf{r}) &= \int_0^{+\infty} w(t)dt \\ &= \int_0^{+\infty} T(t)\cdot\sigma_u(t)dt \\ &= \int_0^{+\infty} \exp\left(-\int_0^{+\infty}\sigma_u(v)dv\right)\cdot\sigma_u(t)dt \\ &= \int_0^{+\infty} \frac{d}{dv}\left[-\exp\left(-\int_0^{t}\sigma_u(v)dv\right)\right]dt \\ &= 1-\exp\left(-\int_0^{+\infty}\sigma_u(v)dv\right) \\ &= 1-T(+\infty)\,.\end{aligned} \tag{1}$$

Note that this equation occurs in the continuous domain. To obtain the discrete counterparts of the edge rendering function, we adopt the same discrete accumulated transmittance approach used in NeRF [8].

$$\hat{E}(\mathbf{r}) = 1-T_N,\quad T_N = \exp\left(-\sum_{j=1}^{N-1}\sigma_{uj}\cdot(t_{j+1}-t_j)\right), \tag{2}$$

where N is the number of samples on the ray, T_N denotes the accumulated transmittance at the furthest sampling point from the camera center, and t_j is the j-th sample on the camera ray $\mathbf{r}$.

As outlined in the main paper, NEF [14] utilizes edge volume density for representing edges, assigning high-density values σ_e for edge points. However, this approach, similar to the naive solution of NeuS [12], does not align the local maximum of the weight function to the actual intersection point of the camera ray with the edges (Fig. 1 (a)), leading to inaccuracies in 3D edge reconstruction. To address this issue, we incorporate unbiased UDF rendering [7] into our density-based edge rendering framework, as depicted in Fig. 1 (b). As mentioned in the main paper, the monotonically increasing density function σ_u [7] is formulated as

$$\sigma_u(t) = \Psi(t)\cdot\Omega_s\left(f_u(\mathbf{r}(t))\right)+(1-\Psi(t))\cdot\Omega_s\left(-f_u(\mathbf{r}(t))\right), \tag{3}$$

where $\Psi(t)$ is a differentiable visibility function [7] that identifies the intersection point of the camera ray with the edges, and Ω_s is a monotonic density function that is introduced in NeuS [12] using the Sigmoid function $\Phi_k(x) = \left(1+e^{-kx}\right)^{-1}$. The functions $\Psi(t)$ and Ω_s are formulated as

$$\Psi(t_j) = \prod_{i=1}^{j-1}\left(1-h(t_i)\cdot m(t_i)\right), \tag{4}$$

$$m(t_j) = \begin{cases}0, & \cos(\theta_{j+1})<0\\ 1, & \cos(\theta_{j+1})\geq 0\end{cases}, \tag{5}$$

$$\Omega_s = \max\left(\frac{-\frac{\mathrm{d}\Phi_k}{\mathrm{d}t}(f_u(\mathbf{r}(t)))}{\Phi_k(f_u(\mathbf{r}(t)))},0\right), \tag{6}$$

where $m(t_j)$ is a masking function that masks out the points behind the intersection point, with θ_{j+1} being the angle between the ray and the gradient of f_u at $\mathbf{r}(t+1)$. The probability of intersection $h(t_j)$ is 1 when the ray at t_j intersects with edges and is given by a logistic distribution $\phi_\beta(x)$,

$$h(t_j) = 1-\exp\left(-\alpha\cdot\phi_\beta\left(f_u\left(\mathbf{r}(t_j)\right)\right)\cdot\delta_j\right), \tag{7}$$

$$\phi_\beta(x) = \beta e^{-\beta x}/\left(1+e^{-\beta x}\right)^2, \tag{8}$$

where $\alpha = 20$ and β is a learned parameter.

† Corresponding author

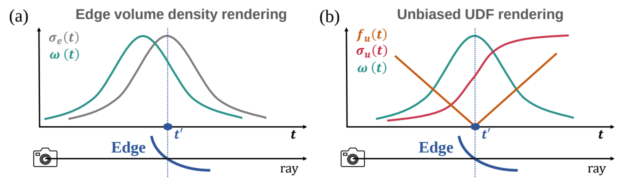

Figure 1. **Illustration of edge volume density rendering (left) and unbiased UDF rendering (right).** In edge volume density rendering, the weight function does not peak at the intersection point t'. Conversely, in unbiased UDF rendering, the weight function's peak aligns precisely with t'.

Algorithm 1 **Point Connection**

Input: Set of potential edge points $\mathcal{U}$ with position and local edge direction, search distance threshold d_t, direction similarity threshold s_t, and NMS ratio n_r.

Output: Collection of edge point groups.

```
 1: while $\mathcal{U} \neq \emptyset$ do
 2:     Select a point $p$ randomly from $\mathcal{U}$ and remove $p$ from $\mathcal{U}$.
 3:     Initialize a new edge point group $G$ with $p$.
 4:     for each direction case: forward $(+)$ and backward $(-)$ do
 5:         Initialize an empty list $S$ for storing similarities.
 6:         for all points $\hat{p}$ in $\mathcal{U}$ within distance $d_t$ of $p$ do
 7:             Calculate similarity $s_{\hat{p}} \leftarrow \cos(\vec{p}, \vec{\hat{p}}) \cdot \mathrm{sign}(direction)$.
 8:             Append $s_{\hat{p}}$ to $S$.
 9:         Find point $\hat{p}_i$ with the highest similarity in $S$.
10:         if $s_{\hat{p}_i} > s_t$ then
11:             Update $p$ to $\hat{p}_i$, remove $\hat{p}_i$ from $\mathcal{U}$.
12:             Add $\hat{p}_i$ to group $G$.
13:             Apply NMS: Remove points from $\mathcal{U}$ with similarity greater than $n_r \cdot s_{\hat{p}_i}$.
14: return the set of all edge point groups formed.
```

A.2. Point Connection Algorithm

In the point initialization step, we set the UDF threshold to ϵ' and normalize the 3D voxel grid to a range of [-1, 1], with a grid size of M^3. In the point shifting step, we perform T iterations. The point connection algorithm is detailed in Alg. 1.

In this algorithm, we establish thresholds for search distance (d_t), direction similarity (s_t), and Non-Maximum Suppression (NMS) ratio (n_r). The process begins with initializing the set of unvisited points ($\mathcal{U}$) with all potential edge points. A random target point p is then selected from $\mathcal{U}$, removed from the set, and added to an edge point group. We identify adjacent points $\hat{p}$ from $\mathcal{U}$ within the search distance d_t from p. The edge point group, being undirected, is extended in two subgroups: one extending forward along the edge direction, and the other extending backward along the inverse edge direction. For each adjacent point, we compute its similarity (s) with the target point as follows

$$s_j = \begin{cases} \cos(\overrightarrow{pp_j}, l(p)), & \text{forward} \\ -\cos(\overrightarrow{pp_j}, l(p)), & \text{backward} \end{cases}, \tag{9}$$

where j indexes the adjacent points. The similarity s_j is calculated using the edge direction for the forward case and the inverse edge direction for the backward case. The adjacent point with the highest similarity (s_i) is selected as the candidate. If s_i exceeds s_t, the candidate point is updated as the new target point, removed from $\mathcal{U}$, and added to the edge point group. Redundant points with similarity greater than $n_r \cdot s_i$ are also removed from $\mathcal{U}$. If s_i is not greater than s_t, the growth of the current point connection halts, and a new target point is selected from $\mathcal{U}$ to initiate a new edge point group. This process is repeated for each edge point group, progressively extending it until no further target points can be added. Finally, we obtain connected edge points from each edge point group.

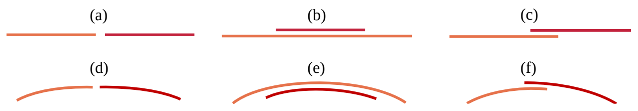

Figure 2. **Illustration of edge merging.** (a)-(c) demonstrate line segment merging, and (d)-(f) show curve merging. In all cases, the candidate edges are closely aligned in terms of shortest distance, with similar line directions or curvatures at their nearest points.

A.3. Edge Fitting

As introduced in the main paper, the merging of line segments and Bézier curves is based on two primary criteria: the shortest distance (d_s) between candidate edges and the curvature similarity (s_c) at their nearest points. Fig. 2 illustrates distinct cases for both line segments and curves. These criteria ensure that merging occurs only between closely situated edges with similar features.

B. More Experiment Details

B.1. Implementation Details of UDF Field

We utilize one MLP with an absolute activation function in the last layer to learn UDF values. Following previous works [12, 13], we include a skip connection that links the output of the fourth layer with its input. Positional encoding is applied using 10 frequencies. Following NeuralUDF [7], we initially sample 64 points uniformly and then perform iterative importance sampling five times to refine the sampling based on the UDF values. In each importance sampling iteration, 16 points are sampled. Additionally, we normalize all scenes into a unit box. This normalization ensures that the hyperparameters for 3D parametric edge extraction are generally invariant to the scale of the scenes. λ in the loss function is set to 0.1 for ABC-NEF dataset, and 0.01 for other dataset.

B.2. Implementation Details of 3D Parametric Edge Extraction

Similar to LIMAP [6], our parametric edge extraction approach involves several hyperparameters within each module. However, as we normalize all scenes into a unit box within our UDF field, our default settings typically suffice. The UDF threshold (σ') and voxel grid size (M) are adjusted according to scene complexity. For example, we use $\sigma' = 0.02$ and $M = 128$ for the synthetic ABC-NEF [14] dataset, while for other datasets like DTU [1], Replica [11], and Tanks & Temples [3], we opt for $\sigma' = 0.01$ and $M = 256$. We set the point shifting iteration (T) to 2, which yields optimal performance, with further details in Sec. C.3. In the edge direction extraction step, the shift set δ_N is randomly sampled from the range $\left[-5 \times 10^{-3}, 5 \times 10^{-3}\right]$ with 50 samples. In point connection, the parameters d_t, s_t, and n_r are set to $\frac{10}{M}$, 0.97, and 0.95, respectively. For edge fitting, we require a minimum of 5 inlier points for robust line segment fitting and at least 4 for Bézier curve fitting. During edge merging, we apply criteria of $d_s = \frac{5}{M}$ and $s_c = 0.98$. Endpoints within a distance of $d_e = \frac{2}{M}$ are merged into shared endpoints. Edge refinement is applied specifically to real-world datasets to mitigate the effect of noise in the input edge maps.

B.3. Ground-truth Edge Point Generation on DTU

As mentioned in the main paper, the DTU dataset provides dense ground-truth point clouds that can be further processed into edge points. To meet multi-view consistent edge requirements, we select scan 37, scan 83, scan 105, scan 110, scan 118, and scan 122 as our evaluation scans. Inspired by [2] that builds edge point matching using the sparse 3D points from Structure-from-Motion, we generate edge points by projecting the ground-truth dense points onto images and then cross-comparing these projections with observations on 2D edge maps. This process allows us to filter out points that do not correspond to edges. To ensure accuracy in the ground-truth edge points, we manually set thresholds for each scan and meticulously remove any floating points.

C. More Experiment Results

C.1. Edge Representations

We evaluate the effectiveness of SDF and UDF for edge representation. As illustrated in Fig. 3, the edge map and normal map rendered using SDF tend to be noisy and lack precision. This issue primarily arises because SDF rendering requires a clear definition of inside and outside, which edges do not inherently possess. In contrast, UDF excels at representing both closed and open surfaces in volume rendering, making them more suitable for modeling the edge distance field. Consequently, UDF-rendered edge and normal maps exhibit greater precision. In comparison, representations based on edge volume density, like those in NEF [14] that often rely on fitting-based extraction methods, are more prone to inaccuracies. Therefore, we select UDF as our edge representation.

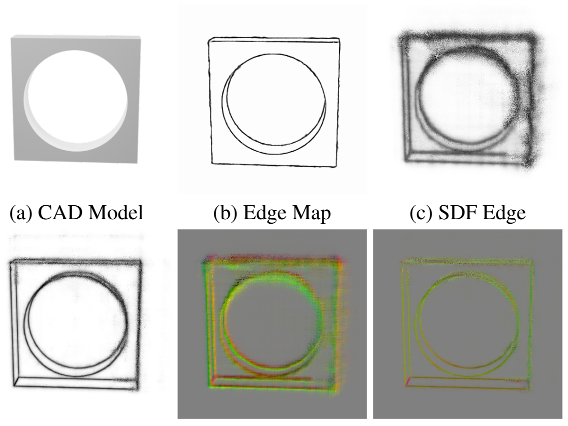

Figure 3. **Ablation study on SDF and UDF.** Representing edges using SDF leads to blurred edge rendering, as edges do not have a clear definition of inside and outside. In contrast, UDF is well suited for modeling edge distance field.

	Ratio	Acc↓	Comp↓	Norm ↑	R5↑	P5↑	F5↑
a	25%	7.9	8.9	95.3	**57.1**	**64.7**	**60.3**
b	50%	8.8	**8.9**	**95.4**	56.4	62.9	59.1
c	75%	**7.7**	10.3	94.8	53.2	58.3	55.3

Table 1. Ablation studies on ray sampling strategy on ABC-NEF. Performance metrics for 25% sampling (a) and 75% sampling (c) are averaged over *valid scans* only, as 25% sampling results in invalid outcomes for scan 9685, and 75% sampling is ineffective for scans 2412 and scan 5109.

C.2. Ray Sampling Strategy

To evaluate the effectiveness of our ray sampling strategy, we conduct ablation studies on the sampling ratio in Table 1. The sampling ratio in edge regions is then incrementally increased from 25% to 75%. The results indicate that a 50% sampling ratio is the only strategy that consistently yields complete results across all scans, while 25% and 75% sampling ratios fail to reconstruct all scans. Besides, the commonly used uniform sampling strategy is largely ineffective, predominantly due to occlusion-related issues.

	Iter.	Acc↓	Comp↓	Norm↑	R5↑	P5↑	F5↑
a	0	14.4	9.4	94.3	32.7	26.2	28.3
c	1	9.9	9.0	95.1	49.6	53.8	51.2
d	2	**8.8**	**8.9**	**95.4**	**56.4**	**62.9**	**59.1**
e	3	9.1	9.1	95.4	53.5	59.8	56.1

Table 2. Ablation studies on point shifting iterations on ABC-NEF.

Res	w/o point shifting			w/ point shifting		
	Acc↓	Comp↓	F5↑	Acc↓	Comp↓	F5↑
64	15.1	31.3	13.8	8.8	29.0	37.8
128	14.4	9.4	28.3	**8.8**	8.9	**59.1**
256	16.5	**8.0**	24.6	10.5	8.2	55.2

Table 3. Ablation studies on point-shifting with various grid resolutions on ABC-NEF.

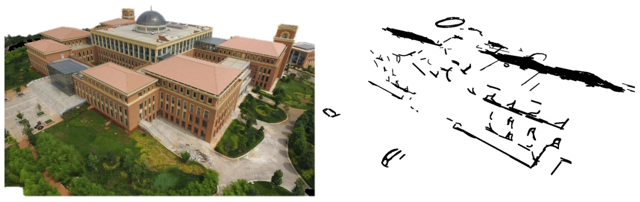

Figure 4. **A failure case in the large-scale scene.** *left*: RGB image, *right*: reconstructed 3D edges.

C.3. Point Shifting Iteration

We conduct analytical experiments to assess the impact of point-shifting iterations, as presented in Table 2. We empirically find that both accuracy and completeness are maximized when the iteration count, T, is set to 2. As discussed in the main paper, the absence of point shifting ($T = 0$) results in significantly lower accuracy due to the prediction of redundant edge points. Conversely, increasing the iteration count ($T = 3$) leads to a degradation in performance, attributable to the introduction of noise from over-iteration.

C.4. Point Shifting vs. Grid Resolution.

We perform ablation studies to analyze the effectiveness of point-shifting step with various voxel grid resolutions. Table 3 shows that point shifting consistently boosts the performance across various resolutions. Moreover, increasing grid resolution does improve completeness, but could harm accuracy. Additionally, the cubic growth in query points with higher resolutions leads to a substantial computation increase.

D. Limitations and Future Work

Our approach successfully achieves comprehensive edge reconstruction across various datasets. However, its application to large-scale datasets is somewhat restricted due to our reliance on edge maps without texture features and the use of vanilla NeRF [8] MLP, leading to ill-posed reconstruction scenarios (Fig. 4). Incorporating texture information or monocular depth maps [15], and applying more powerful volume rendering methods [4, 5, 9] can potentially enhance our method's reconstruction capabilities on large-scale datasets.

Additionally, our method is limited to scenes with only view-consistent edges. To handle view-inconsistent edges, one future work is to apply semantic edge detector [10] to detect those view-inconsistent edges and remove them from 2D edge maps. Alternatively, in our UDF training phase, we could also add a regularization loss to enforce that 3D edges maintain consistency across multiple views.

References

[1] Henrik Aanæs, Rasmus Ramsbøl Jensen, George Vogiatzis, Engin Tola, and Anders Bjorholm Dahl. Large-scale data for multiple-view stereopsis. *International Journal of Computer Vision (IJCV)*, 2016.

[2] Andrea Bignoli, Andrea Romanoni, and Matteo Matteucci. Multi-view stereo 3d edge reconstruction. In *Proc. of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[3] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Trans. on Graphics (ToG)*, 2017.

[4] Ruilong Li, Matthew Tancik, and Angjoo Kanazawa. Nerfacc: A general nerf acceleration toolbox. In *Proc. of the International Conf. on Computer Vision (ICCV)*, 2023.

[5] Zhaoshuo Li, Thomas Müller, Alex Evans, Russell H Taylor, Mathias Unberath, Ming-Yu Liu, and Chen-Hsuan Lin. Neuralangelo: High-fidelity neural surface reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[6] Shaohui Liu, Yifan Yu, Rémi Pautrat, Marc Pollefeys, and Viktor Larsson. 3d line mapping revisited. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[7] Xiaoxiao Long, Cheng Lin, Lingjie Liu, Yuan Liu, Peng Wang, Christian Theobalt, Taku Komura, and Wenping Wang. Neuraludf: Learning unsigned distance fields for multi-view reconstruction of surfaces with arbitrary topologies. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[8] 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 *Proc. of the European Conf. on Computer Vision (ECCV)*, 2020.

[9] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *SIGGRAPH*, 2022.

[10] Mengyang Pu, Yaping Huang, Qingji Guan, and Haibin Ling. Rindnet: Edge detection for discontinuity in reflectance, illumination, normal and depth. In *Proceedings*

of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 6879–6888, 2021.

[11] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[12] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[13] Lior Yariv, Jiatao Gu, Yoni Kasten, and Yaron Lipman. Volume rendering of neural implicit surfaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[14] Yunfan Ye, Renjiao Yi, Zhirui Gao, Chenyang Zhu, Zhiping Cai, and Kai Xu. Nef: Neural edge fields for 3d parametric curve reconstruction from multi-view images. In *Proc. of the Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[15] Zehao Yu, Songyou Peng, Michael Niemeyer, Torsten Sattler, and Andreas Geiger. Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.