A Level Set Theory for Neural Implicit Evolution under Explicit Flows: Appendix

Ishit Mehta Manmohan Chandraker Ravi Ramamoorthi

University of California San Diego

1 Outline

We organize the Appendix according to the section numbers used in the main manuscript. In Section 5.2, the proofs for Result 3 and Result 4 are outlined. In Section 6.1, implementation details are added along with additional results in Figure 1 and Figure 2. In Section 6.2, implementation details for our inverse rendering experiments are discussed and qualitative comparisons with previous works are shown in Figure 5. An inverse rendering experiment using LSIG [6] with different geometry initializations is shown in Figure 4. Section 6.3 describes the user-defined shape editing method in more detail. A comparison with NFGP [12] is in Figure 6.

5.2 Differentiable Surface Rendering

We first draw comparisons between Differentiable Volumetric Rendering[1] [7], Implicit Differentiable Renderer (IDR) [13] and Differentiable Iso-Surface Extraction in Result 3. Next, in Result 4, we show that methods in [7,13] deviate from the level-set theory for tangential flow fields.

Result 3 *Surface evolution using differentiable ray-marching of parametric implicit surfaces [5,13] is the same as using differentiable iso-surface extraction [8] when the viewing direction* $\mathbf{v}_u$ *is parallel to the normal* $\mathbf{n}$ *at the intersection point* $\mathbf{x}_u$*. The parameters* θ *for the level-set function* Φ *are updated as:*

$$\theta \leftarrow \theta - \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}.$$

Proof. Inverse rendering methods in [7,13] find a surface-intersection point by marching a along ray that is spawned from the camera centre $\mathbf{c}$, in the direction $\mathbf{v}_u$, for each pixel u. The intersection point $\mathbf{x}_u$ is analytically defined as:

$$\mathbf{x}_u = \mathbf{c} + d_u \mathbf{v}_u, \tag{1}$$

where d_u is the depth for $\mathbf{x}_u$. The distance d_u in [7] is estimated from the camera centre. For IDR [13], the distance is computed from a point $\mathbf{y}$ close to the surface

[1] Here, volumetric rendering is a misnomer. It's really surface rendering for geometry defined with volume/occupancy.

which modifies (1) to $\mathbf{x}_u = \mathbf{y}_u + d_u\mathbf{v}_u$ ($\mathbf{y}_u$ is denoted as $\mathbf{x}_0$ in [13]). In both the methods, a rendering loss function $\mathcal{L}$ (photometric error) is computed for pixels $u \in U$. To update the geometry parameters θ, the gradient $\frac{\partial \mathcal{L}}{\partial \theta}$ is computed:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{u \in U} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_u}\frac{\partial \mathbf{x}_u}{\partial \theta} = \sum_{u \in U} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_u}\frac{\partial \mathbf{x}_u}{\partial d_u}\frac{\partial d_u}{\partial \theta} = \sum_{u \in U} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u \frac{\partial d_u}{\partial \theta} \tag{2}$$

To compute $\frac{\partial d_u}{\partial \theta}$, we slightly deviate from the derivations in [7,13] for clarity. For points $\mathbf{x}_u$ on the implicit surface, we know $\Phi(\mathbf{x}_u) = 0$. Using implicit differentiation:

$$\begin{aligned}
&\frac{\partial \Phi}{\partial \theta} + \frac{\partial \Phi}{\partial \mathbf{x}_u}\frac{\partial \mathbf{x}_u}{\partial \theta} = 0 \\
\iff &\frac{\partial \Phi}{\partial \theta} + \frac{\partial \Phi}{\partial \mathbf{x}_u}\frac{\partial \mathbf{x}_u}{\partial d_u}\frac{\partial d_u}{\partial \theta} = 0 \\
\iff &\frac{\partial \Phi}{\partial \theta} + \frac{\partial \Phi}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u \frac{\partial d_u}{\partial \theta} = 0 \qquad \triangleleft \text{ From (1)} \\
\iff &\frac{\partial d_u}{\partial \theta} = -\frac{1}{\frac{\partial \Phi}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u}\frac{\partial \Phi}{\partial \theta}
\end{aligned} \tag{3}$$

From (2) and (3) we have:

$$\frac{\partial \mathcal{L}}{\partial \theta} = -\sum_{u \in U} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_u} \cdot \frac{\mathbf{v}_u}{\frac{\partial \Phi}{\partial \mathbf{x}_u} \cdot \mathbf{v}_u}\frac{\partial \Phi}{\partial \theta} \approx -\sum_{u \in U} \frac{\partial \mathcal{L}}{\partial \mathbf{x}_u} \cdot \frac{\nabla \Phi}{|\nabla \Phi|^2}\frac{\partial \Phi}{\partial \theta} \text{ (when } \mathbf{v}_u \to \nabla \Phi) \tag{4}$$

Taking $-\frac{\partial \mathcal{L}}{\partial \mathbf{x}_u} = \mathbf{V}$, we can update the parameters θ using (4) and gradient descent as:

$$\theta \leftarrow \theta - \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \nabla \Phi \frac{\partial \Phi}{\partial \theta}. \tag{5}$$

Equation (5) shows that DVR/IDR and MeshSDF$^\theta$ are closely related, while Result 1 shows that these methods do not agree with the level-set theory. □

Result 4 *Differentiable ray-marching of parametric implicit surfaces [5,13] disagrees with the level-set equation for tangential components $\mathbf{V}^\perp$ of the flow field $\mathbf{V}$. The change in parameters $\Delta\theta$ is:*

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \pm|\mathbf{V}^\perp| \tan(\arccos(\nabla \Phi \cdot \mathbf{v}_u))\frac{\partial \Phi}{\partial \theta}. \tag{6}$$

Proof. From (4), we know the change in parameters θ for methods in [7,13] is:

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \mathbf{V} \cdot \frac{\mathbf{v}_u}{\nabla \Phi \cdot \mathbf{v}_u}\frac{\partial \Phi}{\partial \theta}. \tag{7}$$

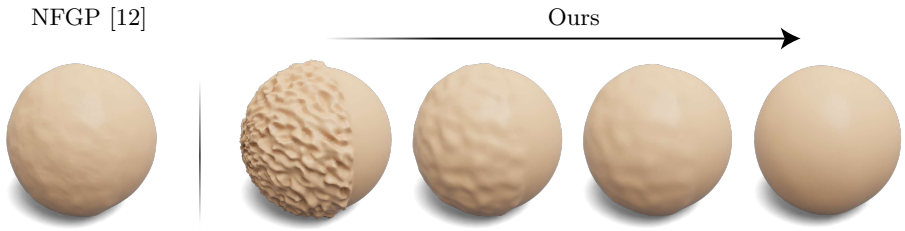

Fig. 1: **Surface smoothing evolution on a half-noisy sphere.** (*Right*) Our method can evolve a noisy surface using an explicit diffusion flow-field. This allows smoothing on implicitly defined surfaces with increasing levels of smoothness. (*Left*) Yang *et al.*'s [12] method uses an optimization objective for a single level of smoothness.

When the flow-field is tangential to the surface, $\mathbf{V} = \mathbf{V}^{\perp}$, and $\mathbf{V}^{\perp} \perp \nabla\Phi$. The change in parameters for this flow-field is:

$$\Delta\theta = \lambda \sum_{\mathbf{x}_u} \frac{\mathbf{V}^{\perp} \cdot \mathbf{v}_u}{\nabla\Phi \cdot \mathbf{v}_u} \frac{\partial\Phi}{\partial\theta}. \tag{8}$$

When $\Phi : \mathbb{R}^2 \mapsto \mathbb{R}$ is a level-set function defined in 2D, we can modify (8) as:

$$\begin{aligned} \Delta\theta &= \lambda \sum_{\mathbf{x}_u} \pm|\mathbf{V}^{\perp}| \frac{\sin\alpha}{\cos\alpha} \frac{\partial\Phi}{\partial\theta} \qquad \text{where, } \alpha = \arccos(\nabla\phi \cdot \mathbf{v}_u) \text{ and } \Phi \text{ is an SDF} \\ &= \lambda \sum_{\mathbf{x}_u} \pm|\mathbf{V}^{\perp}| \tan\alpha \ \frac{\partial\Phi}{\partial\theta} \\ &= \lambda \sum_{\mathbf{x}_u} \pm|\mathbf{V}^{\perp}| \tan(\arccos(\nabla\Phi \cdot \mathbf{v}_u)) \ \frac{\partial\Phi}{\partial\theta}. \end{aligned} \tag{9}$$

The parameters here could change depending on the angle subtended between the normal $\nabla\Phi$ and the viewing direction $\mathbf{v}_u$. This dependence on the viewing direction is a result of using differentiable ray-marching. When the function is deformed using the level-set method described in the main paper, the change in parameters $\Delta\theta$ does not depend on the viewing direction and is 0. For 3D level-set functions, the parameters could still change as the term $\mathbf{V}^{\perp} \cdot \mathbf{v}_u$ in (8) could be non-zero.

6.1 Curvature-based Deformation

For each shape, we optimize a SIREN [9] MLP with 5 layers, each of which has a 512-sized vector output. We use the publicly-released code[2] by Yang *et al.* [12] for SDF queries and optimization of the network. The flow-field defined in Equation

[2] https://github.com/stevenygd/NFGP

(14) is used for smoothing. The learning rate is 10^{-6}, the time-delta Δt is 0.95 and an eikonal regularization term (enforcing $\nabla \Phi = 1$) [2] is used with the error function defined in Equation (5). The regularization term is weighed 10^{-3} times against the main objective. We use Adam optimizer [3] and use 200 gradient steps for each time step. The Lagrangian surface is extracted using Marching Cubes at $(120 \pm 3)^3$ resolution. Figure 1 shows the surface evolution with respect to time on a half-noisy sphere (similar to [10]). In addition to the results in the main manuscript, we show qualitative comparisons with other methods in Figure 2.

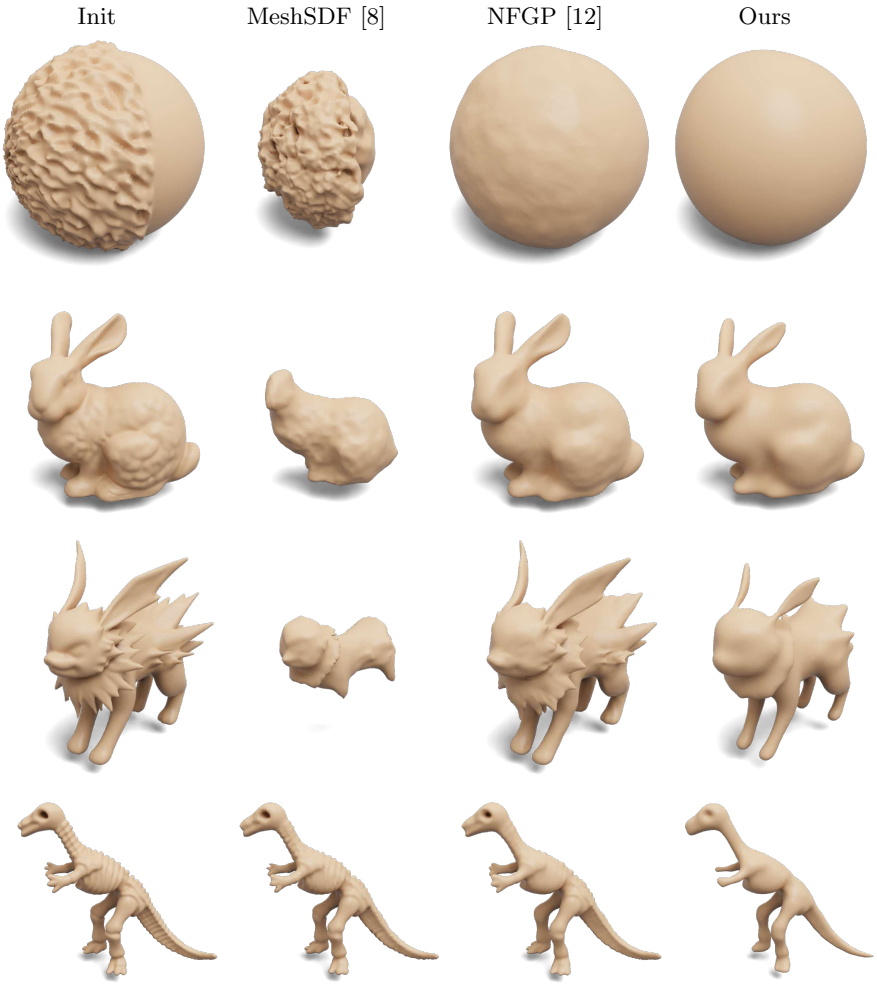

Fig. 2: **Qualitative comparisons for surface smoothing.** We apply surface smoothing on four computer graphics models encoded as parametric level-sets. Best viewed when zoomed-in.

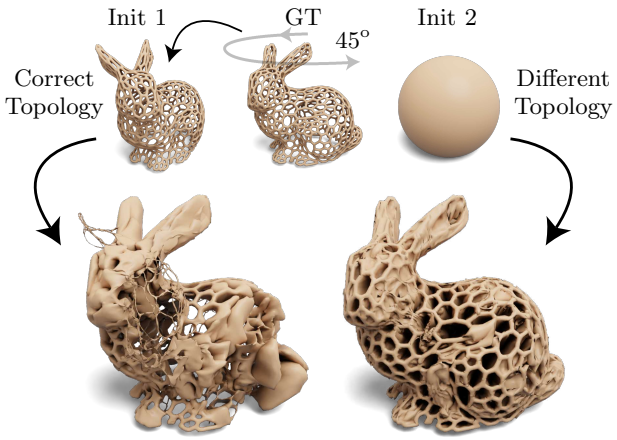

Fig. 4: **LSIG [6] converges to a poor estimate with correct topology at initialization.** Optimizing a triangle mesh for inverse rendering problems is non-trivial even when the initialization is close to the ground truth and has correct topology. For `VBunny` we rotate the target geometry and use it as initialization. The shape optimized using LSIG [6] has correct topology but with poor mesh quality and reconstruction accuracy.

6.2 Inverse Rendering

Our method uses the same spherical initialization for all the shapes. The network is composed of 4 layers, with 512 neurons in each layer. The learning-rate is 2×10^{-6}, the weight-decay factor is 0.1 and the time-delta Δt is 10^{-4}. Each object is confined in a 2^3 volume and is rendered with Nvdiffrast [4]. The weight for the smoothness regularization term linearly decreases from 10^{-3} to 0. An ablation is shown in Figure 3. We do not observe reliable improvements quantitatively with the regularizer, but do observer better convergence. The Phong shading model is used with 0.55 as the albedo value. We show an experiment with different topological initializations for LSIG [6] in Figure 4. All the qualitative comparisons for results in Table 1 (main) are in Figure 5.

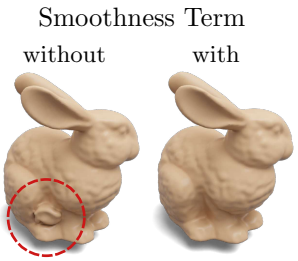

Fig. 3: Ablation on smoothness regularization term in the flow-field for inverse rendering.

6.3 User-defined Deformation

As discussed in the main paper, we use the thin-shell energy [11] loss to densify a sparsely defined flow-field from user inputs. It induces a flow field which is

Fig. 5: **Qualitative comparisons for Inverse Rendering of Geometry.**

characterized using an Euler-Lagrange equation:

$$-k_s \Delta \mathbf{V} + k_b \Delta^2 \mathbf{V} = 0. \tag{10}$$

We know $\mathbf{V}$ for the regions on the surface where the user specifies the deformation and would like to estimate it for the entire surface such that there is minimum bending and stretching. Equation 10, however is a result of a linearized version of the thin-shell energy loss. In this case, the resultant deformations can be unpleasant. This is required for explicit surface deformation methods as they are expected to be conducive to real-time editing. On the contrary, our goal is to have plausible deformations which can be slightly more time consuming. For each editing operation ($\mathbf{x} \rightarrow \mathbf{x}'$), we break the deformation in T time steps as $\mathbf{x} \rightarrow \mathbf{x}^1 \rightarrow \mathbf{x}^2 \rightarrow \mathbf{x}^3 \dots \mathbf{x}^T$. For each time-step t, we solve for (10) (using a sparse linear solver) with the following user constraints:

$$\mathbf{V}(\mathbf{x}_h^{t-1}) = \mathbf{x}_h^t - \mathbf{x}_h^{t-1}, \tag{11}$$

where $\mathbf{x}_h \in \mathcal{H}$ belongs to a set of handle vertices for which the user defines deformation. Having obtained the flow-field for all the points on the surface, we evolve the surface using an Euler step as $\mathbf{x}^t = \mathbf{x}^{t-1} + \mathbf{V}(\mathbf{x}^{t-1})$. Note the absence of a delta term (Δt) in the Euler step. This is because of how we define the flow-field; we know exactly where the surface is expected to be at a given time step (From (11)). This is different from gradient-based Euler integration where we take small steps in the direction of the flow-field. Since we know *exactly* where the surface is at time t (can assume $\Phi(\mathbf{x}) = 0$ for surface points), we can tweak the objective defined in Equation 5 (main):

$$\min_{\theta} J = \frac{1}{|\partial \Omega_L^t|} \sum_{\mathbf{x} \in \Omega_L^t} ||\Phi(\mathbf{x})||^2 + \beta \sum_{\mathbf{x} \in \Omega} (|\nabla \Phi(\mathbf{x})| - 1)^2, \tag{12}$$

where we also add Eikonal regularization. An example deformation is shown in Figure 6 including a comparison with NFGP [12]. Note that our goal with these

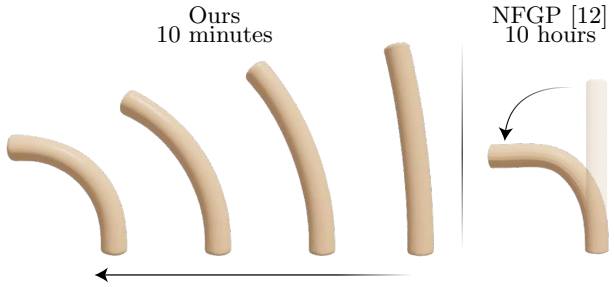

Fig. 6: **User-defined rotation on an implicitly defined cylinder.** (*Left*) User editing using our method is faster (takes 10 min) than (*Right*) NFGP which takes 10 hr for the same editing operation.

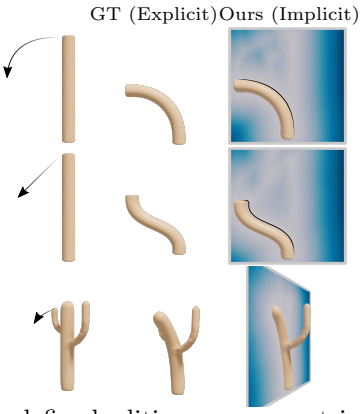

Fig. 7: User-defined editing on parametric level-sets.

experiments is to showcase the versatility of the level-set method and not propose a method which performs accurate and real-time deformations; we would still pose this application of user-editing as a proof of concept. Several explicit-surface based methods [1] exist which can probably generate better deformations.

Implementation Details We use pre-trained neural implicit representations provided in the publicly-released code by Yang *et al.* [12]. The number of time steps T is 20. The learning rate is 2×10^{-6} and 750 gradient steps are taken to minimize Equation 12 using Adam [3]. The regularization weighting term $\beta = 10^{-4}$.

References

1. Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B.: Polygon mesh processing. CRC press (2010)
2. Gropp, A., Yariv, L., Haim, N., Atzmon, M., Lipman, Y.: Implicit geometric regularization for learning shapes. arXiv preprint arXiv:2002.10099 (2020)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
4. Laine, S., Hellsten, J., Karras, T., Seol, Y., Lehtinen, J., Aila, T.: Modular primitives for high-performance differentiable rendering. ACM Transactions on Graphics **39**(6) (2020)
5. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (2019)
6. Nicolet, B., Jacobson, A., Jakob, W.: Large steps in inverse rendering of geometry. ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) **40**(6) (Dec 2021). https://doi.org/10.1145/3478513.3480501, `https://rgl.epfl.ch/publications/Nicolet2021Large`
7. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)

8. Remelli, E., Lukoianov, A., Richter, S., Guillard, B., Bagautdinov, T., Baque, P., Fua, P.: Meshsdf: Differentiable iso-surface extraction. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 22468–22478. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/fe40fb944ee700392ed51bfe84dd4e3d-Paper.pdf
9. Sitzmann, V., Martel, J.N., Bergman, A.W., Lindell, D.B., Wetzstein, G.: Implicit neural representations with periodic activation functions. In: arXiv (2020)
10. Taubin, G.: A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques. pp. 351–358 (1995)
11. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. In: Proceedings of the 14th annual conference on Computer graphics and interactive techniques. pp. 205–214 (1987)
12. Yang, G., Belongie, S., Hariharan, B., Koltun, V.: Geometry processing with neural fields. In: Thirty-Fifth Conference on Neural Information Processing Systems (2021)
13. Yariv, L., Kasten, Y., Moran, D., Galun, M., Atzmon, M., Ronen, B., Lipman, Y.: Multiview neural surface reconstruction by disentangling geometry and appearance. Advances in Neural Information Processing Systems **33** (2020)