

A Zero-Level Set Preserving Technique for Signed Distance Function Computation from an Implicit Surface

Pierre-Alain Fayolle
University of Aizu

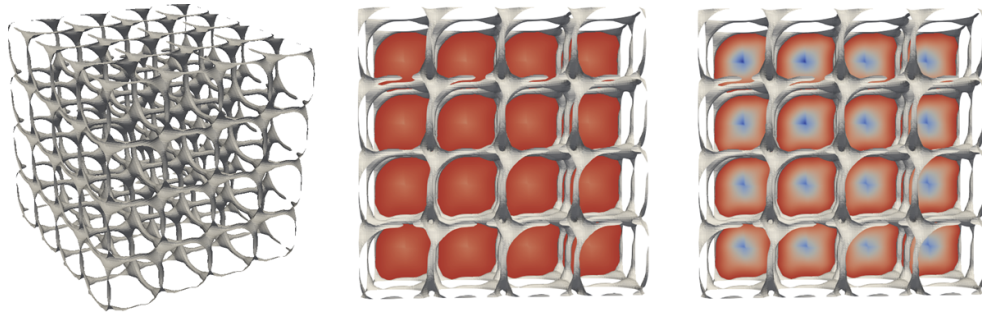


Figure 1. Left: A complex implicit surface modeled using torii, the sawtooth function, and a blending union. Middle: A filled contour plot of the model illustrating the un-normalized nature of the function (the norm of its gradient is different than one). Right: The signed distance to the model surface obtained after re-distancing using the techniques presented in this article.

Abstract

We describe in this short article a technique to convert an implicit surface into a signed distance function (SDF) while exactly preserving the zero-level set of the implicit surface. The proposed approach relies on embedding the input implicit surface in the final layer of a neural network by multiplying this neural network by the implicit surface function or a smoothed version of its sign function. The parameters of the corresponding function are then trained to minimize a loss function characterizing the SDF, either exactly by a term corresponding to an eikonal equation or approximately by a term involving the p -Laplace operator.

1. Introduction

We consider in this article the problem of computing the signed distance function (SDF) to an implicit surface. Let a surface S be defined as the zero-level set of a function f , $S = \{x \in \mathbb{R}^3 : f(x) = 0\}$. We propose a method for computing the

signed distance to S ,

$$d(\mathbf{x}) = \pm \text{dist}(\mathbf{x}, S) = \pm \min_{\mathbf{y} \in S} |\mathbf{x} - \mathbf{y}|, \quad (1)$$

while preserving exactly the zero-level set of f . Without loss of generality, we assume that $d > 0$ inside the domain bounded by S and that $d < 0$ outside. Here and everywhere else in this note, $|\cdot|$ stands for the vector ℓ_2 norm.

In the literature on level-set methods, techniques for computing the SDF d from f are called re-initialization (or re-distancing) methods [Sussman et al. 1994; Osher and Fedkiw 2003]. The usual re-initialization method, due to Sussman et al. [1994], considers the problem

$$\frac{\partial g(\mathbf{x}, t)}{\partial t} + (1 - |\nabla_{\mathbf{x}} g(\mathbf{x}, t)|) = 0, \quad g(\mathbf{x}, 0) = f(\mathbf{x}), \quad (2)$$

and solves it to a steady state. Typically, Equation (2) is solved by discretizing the time derivative with explicit or implicit Euler and the spatial derivative by finite difference [Sussman et al. 1994]. Another popular approach for computing Equation (1) is the Fast Marching Method [Sethian 1996] used to solve the eikonal equation $|\nabla g(\mathbf{x})| = 1$.

In each case, the approach relies on computing a regular grid for the domain and sampling functions and derivatives on this grid. Thus, the zero-level set of g does not correspond exactly to the zero-level set of f .

Conversely, the method described in this article preserves exactly the zero-level set of f . It does so by computing the signed distance $d(\mathbf{x}; \boldsymbol{\theta})$ as $f(\mathbf{x})g(\mathbf{x}; \boldsymbol{\theta})$, where $g(\cdot)$ is a parametric function whose parameters $\boldsymbol{\theta}$ are fitted so that d corresponds to the distance function.

2. Related Work

2.1. Distance to an Implicit Surface

The simplest method for turning f into an SDF consists in meshing the zero-level set S , for example with the Marching Cubes algorithm [Lorensen and Cline 1987], and then computing the distance to the polygonal mesh. Typically, accelerating data structures, such as a Kd-tree or a bounding volume hierarchy (BVH), are used for practical computations.

Re-initialization (or re-distancing) a scalar field f sampled on a grid is a common operation in level-set methods [Sussman et al. 1994; Osher and Fedkiw 2003]. It relies on solving Equation (2) by finite differences for the spatial derivatives and explicit Euler for time stepping. Several variants were proposed over the years that use higher-order schemes for approximating the spatial derivatives or for time stepping. Related approaches include the Fast Marching Method of Sethian [1996] or the fast sweeping method of Zhao [2005].

All these approaches rely on sampling the function and its derivatives on a grid (generally a regular grid, but methods working on hierarchical grids exist as well).

2.2. Deep Neural Networks for Computational Science

While deep neural networks were originally designed to tackle regression or classification problems, their use has now been extended to deal with different types of problems in computational sciences and applications.

Deep neural networks can be trained to compute the SDF to a given point cloud or triangle mesh [Atzmon et al. 2019; Atzmon and Lipman 2020, 2021; Chibane et al. 2020; Gropp et al. 2020; Davies et al. 2021; Sitzmann et al. 2020b,a; Takikawa et al. 2021]. All of these methods differ in how they model the loss function or the network architecture. For example, Sitzmann et al. [2020b] proposes using sine activation functions.

In this work, instead of starting from a point cloud or a polygonal mesh, we assume that the input is a function f whose zero-level set describes the surface of interest. Typically, this implicit surface would have been designed with a modeling system, and the distance to its zero-level set could be needed for certain types of modeling operations (for example, as a design parameter) or for further simulations (for example, in computational mechanics). The surface information is embedded in the neural network by multiplying the final layer by the function f itself or by a smoothed version of $\text{sign}(f)$.

Related to our approach are recent techniques for solving partial differential equations (PDE), including high-dimensional and stochastic PDE, with deep neural networks [Weinan and Yu 2018; Blechschmidt and Ernst 2021].

3. Approach

Given a function f whose zero-level set defines a surface S , our goal is to compute d , the signed distance to S from Equation (1). We look for the distance function d under the form

$$d(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{x})g(\mathbf{x}; \boldsymbol{\theta}), \quad (3)$$

where g is a parametric function with parameters $\boldsymbol{\theta}$. We use a fully connected feed-forward neural network for g . For fitting $\boldsymbol{\theta}$, the parameters of g , we express d as the solution to a variational problem (see Sections 3.2 and 3.3).

An alternative to Equation (3) is to use

$$d(\mathbf{x}; \boldsymbol{\theta}) = \text{sign}(f(\mathbf{x}))g(\mathbf{x}; \boldsymbol{\theta}), \quad (4)$$

where $\text{sign}(u)$ is a smoothed version of the sign function. For example, one can use $\text{sign}(u) \equiv \tanh(\alpha u)$, where α is a user-specified parameter.

3.1. Feedforward Neural Network

As an ansatz for $d(\mathbf{x})$, we use $f(\mathbf{x})g(\mathbf{x}; \boldsymbol{\theta})$, where $y_L = g(\mathbf{x}; \boldsymbol{\theta})$, $y_L \in \mathbb{R}$, is a deep, fully connected neural network defined by

$$\begin{aligned} y_L &= W_L \sigma(y_{L-1}) + b_L, \\ &\dots \\ y_2 &= W_2 \sigma(y_1) + b_2, \\ y_1 &= W_1 \mathbf{x} + b_1, \end{aligned}$$

where L corresponds to the number of layers in the deep neural network, $\mathbf{W} = \{W_i\}_{i=1,\dots,L}$ and $\mathbf{b} = \{b_i\}_{i=1,\dots,L}$ correspond to the parameters $\boldsymbol{\theta}$ of the neural network, and $\sigma(\cdot)$ is a nonlinear activation function.

3.2. Variational Problems

The distance function d in Equation (1) is the viscosity solution to the eikonal equation [Crandall and Lions 1983]

$$|\nabla_{\mathbf{x}} d| = 1 \quad (5)$$

with boundary condition $d(\mathbf{x}) = 0$ for $\mathbf{x} \in S$. See, e.g., Belyaev and Fayolle [2015, 2020] for recent numerical techniques to solve Equation (5).

For some applications, one may be interested instead in smooth approximations to the distance function; in these cases an alternative approach consists in considering the p -Poisson problem

$$\Delta_p u = -1, \quad (6)$$

subject to $u = 0$ on S and where $\Delta_p u \equiv \operatorname{div}(|\nabla_{\mathbf{x}} u|^{p-2} \nabla_{\mathbf{x}} u)$ is the p -Laplacian, with p being an integer.

Let u_p be the solution to Equation (6); then $u_p(\mathbf{x}) \rightarrow \operatorname{dist}(\mathbf{x}, S)$ as $p \rightarrow \infty$ and thus u_p can be used to approximate the signed distance function to S [Belyaev and Fayolle 2015; Fayolle and Belyaev 2018].

For a fixed value of p , the solution u_p to Equation (6) only delivers an approximation of the distance function to S . The approximation quality can be improved by a simple normalization scheme as described by Belyaev and Fayolle [2015].

3.3. Loss Functions

Let $g(\mathbf{x}; \boldsymbol{\theta} = (\mathbf{W}, \mathbf{b}))$ be a deep and fully connected neural network with parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$. We use the ansatz

$$d(\mathbf{x}; \boldsymbol{\theta}) = f(\mathbf{x})g(\mathbf{x}; \boldsymbol{\theta}) \quad (\text{or} \quad d(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign}(f(\mathbf{x}))g(\mathbf{x}; \boldsymbol{\theta}))$$

for the distance function, where the parameters $\boldsymbol{\theta}$ need to be fitted.

Given that the distance function is the viscosity solution to the eikonal Equation (5), we consider the following loss function (to be minimized)

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim D} (|\nabla_{\mathbf{x}} d(\mathbf{x}; \boldsymbol{\theta})| - 1)^2, \quad (7)$$

where $\mathbb{E}$ is the expected value and D is the uniform distribution over the given computational domain.

Similarly, if one wants to solve the p -Poisson problem in Equation (6) instead, the corresponding loss function is given by

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim D} (\Delta_p d(\mathbf{x}; \boldsymbol{\theta}) + 1)^2. \quad (8)$$

It is possible to add other constraints to the loss in Equation (7) or the loss in Equation (8). For example, if necessary, one could prevent eventual extra zero-level sets away from the surface by adding a term such as

$$\mathbb{1}_{\{\mathbf{x}: f(\mathbf{x}) \neq 0\}} \exp(-\gamma |d(\mathbf{x}; \boldsymbol{\theta})|), \quad (9)$$

where $\mathbb{1}_X$ is the indicator function for the set X and γ is a large value (for example, $\gamma = 10^2$).

Stochastic Gradient Descent [Robbins and Monro 1951] or a variant, such as, e.g., ADAM [Kingma and Ba 2014], is used to minimize the loss function in Equation (7) or the loss function in Equation (8), with eventual additional constraints such as Equation (9).

The corresponding minimizer $d(\mathbf{x}; \boldsymbol{\theta} = \tilde{\boldsymbol{\theta}})$, with fitted parameters $\tilde{\boldsymbol{\theta}}$, provides the SDF to S (or an approximation in the case of Equation (8)), while preserving exactly the zero-level set of f .

3.4. Derivatives Computation

We rely on automatic differentiation to compute the spatial derivatives $\nabla_{\mathbf{x}}$, div , or Δ_p exactly. Note that the derivatives are taken here with regards to the input $\mathbf{x}$ of the function.

For training the deep neural network, we also need to compute the derivatives of the loss function in Equation (7) or (8) with regards to the parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{b})$. This is also done by automatic differentiation, as is usually done when training deep neural networks.

4. Numerical Results

4.1. Setup

We use PyTorch [Paszke et al. 2019] for specifying the deep neural network, computing the derivatives (with regards to the network's parameters and the coordinates) and minimizing the loss function.

For the network g we use 8 fully connected linear layers with width 512. We use a skip connection from the input to the middle layer. The architecture is similar to the one described by [Park et al. \[2019\]](#) and [Gropp et al. \[2020\]](#).

The softplus activation function is used for the nonlinear activation functions

$$\sigma(x) = \frac{1}{\beta} \ln(1 + \exp(\beta x)).$$

We used $\beta = 100$ in the experiments. Using the softplus activation function behaved better in our experiments than using the Rectified Linear Unit (RELU), which is typically used with deep neural networks. Note that it is also possible to consider alternatives such as the SIREN layer [\[Sitzmann et al. 2020b\]](#) for the activation function. We did not experiment with this alternative. In the final layer, we multiply the output of the neural network $g(x; \theta)$ by $\tanh(\alpha f(x))$, the smoothed sign function of $f(x)$. We used $\alpha = 0.1$.

We minimize the loss function in Equation (7) or (8) by ADAM [\[Kingma and Ba 2014\]](#), using a learning rate of 0.0001. The parameters of the neural network are initialized with the geometric initialization of [Atzmon and Lipman \[2020\]](#).

For the computation of the loss function in Equation (7) or (8), we use the uniform distribution over the computational domain for the distribution D .

4.2. Results

4.2.1. 1D Example

We start our numerical experiments with a simple example in 1D: We consider the domain $[0, 1]$ with the points $\{0, 1\}$ as the boundary. We build an implicit for this domain as

$$f(x) = x + (1 - x) + \sqrt{x^2 + (1 - x)^2},$$

i.e., the intersection of half-spaces x and $(1 - x)$ with the intersection implemented by an R-function [\[Shapiro 2007; Pasko et al. 1995\]](#).

Note that, in this case, the exact SDF to the boundary is known and is given, of course, by

$$\min(x, 1 - x).$$

Figure 2 shows a plot of the function f , the distance computed by solving the eikonal Equation (5), and the approximate distance obtained by solving the p -Poisson problem in Equation (6) with $p = 2$ and $p = 8$.

Solving Equation (5) or (6) is done by minimizing the corresponding loss function in Equation (7) or (8), respectively. We used 15,000 iterations in both cases.

To verify the quality of the solutions, we also show in Figure 3 the distribution of the p -Laplacian $\Delta_p d(x)$, $p = 2, 8$, for the solution to Equation (6), and the norm of the gradient $|\nabla_x d(x)|$, for the solution to Equation (5). Notice how the distributions

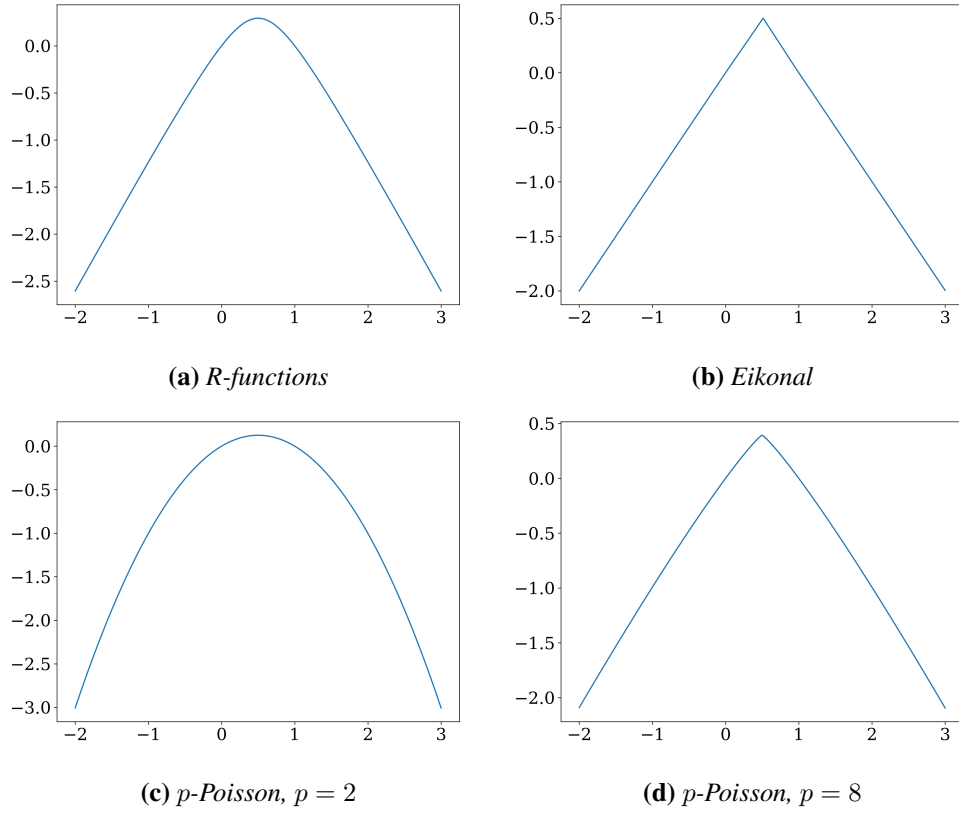


Figure 2. Results in 1D. Plots of (a) $f(x) = x + (1 - x) + \sqrt{x^2 + (1 - x)^2}$, (b) minimizing the eikonal Equation (7) and minimizing Equation (8) with (c) $p = 2$ and (d) $p = 8$. Note that the zero-level set of f (the points $\{0, 1\}$) is exactly preserved.

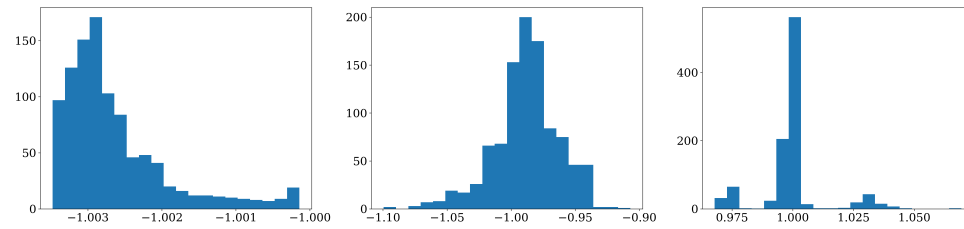


Figure 3. Left: Distribution of the values of the Laplacian Δd ($\Delta_p d$ with $p = 2$). Middle: Distribution of the values of the p -Laplacian $\Delta_p d$, $p = 8$. Right: Distribution of the values of $|\nabla d|$. The computational domain is the interval $[-2, 3]$. The values are centered around -1 for $\Delta_p d$ and 1 for $|\nabla d|$.

are centered around -1 for the solution to the p -Poisson problem and around 1 for the solution to the eikonal equation.

Another observation is that the solution to the p -Poisson problem in Equation (6)

allows us to obtain a smooth approximation to the exact distance function, where p acts as a parameter. Computing smooth approximants to the distance function has applications in domains such as computational mechanics.

4.2.2. 2D Example

In 2D, we consider the function

$$f(x, y) = 1 - x^2 - y^2,$$

whose zero-level set corresponds to a unit circle. The exact SDF to the unit circle is, of course, known and given by

$$1 - \sqrt{x^2 + y^2}.$$

Figure 4 illustrates the results obtained by our approach. The top row shows filled contour plots for the input function $f(x, y)$ and the solution to the eikonal Equation (5) obtained by minimizing Equation (7). The bottom row shows the solution to the p -Poisson problem for $p = 2$ (left) and $p = 8$ (right). In all cases, we used 15,000 iterations of ADAM. Notice again how the zero-level set of f is exactly preserved.

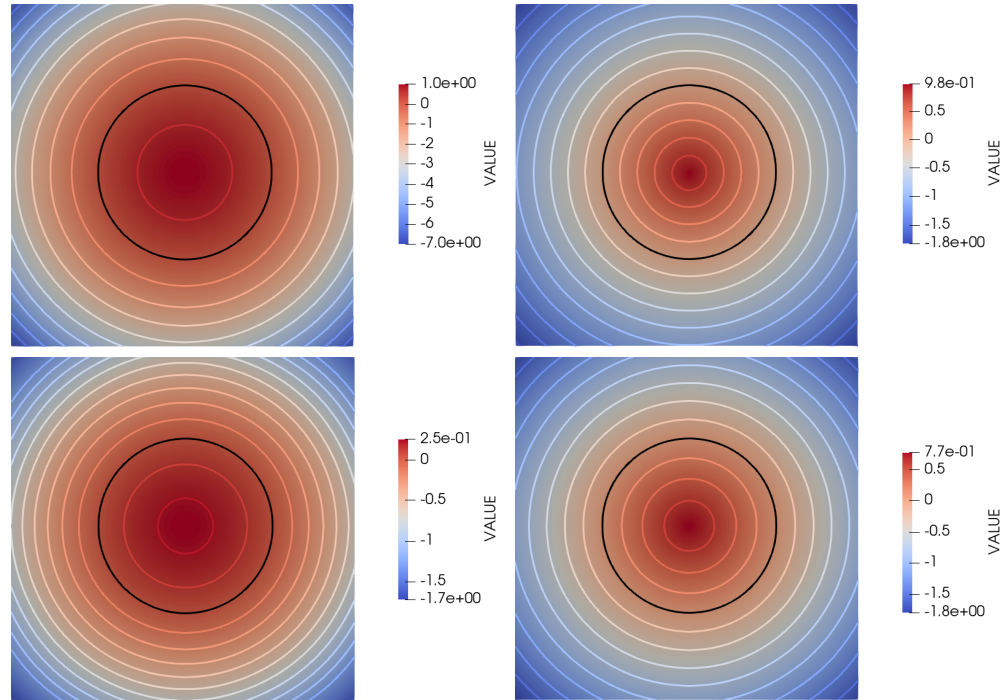


Figure 4. Top: Filled contour plot of $f(x, y) = 1 - x^2 - y^2$ (left) and SDF obtained by minimizing the loss in Equation (7) (right). Bottom: Approximate SDF obtained by minimizing Equation (8) for $p = 2$ (left) and $p = 8$ (right). The zero-level set is shown in black. Contour lines are equidistributed (the difference between two contour lines is a constant within one plot). Note how the zero-level set of f is exactly preserved in all cases.

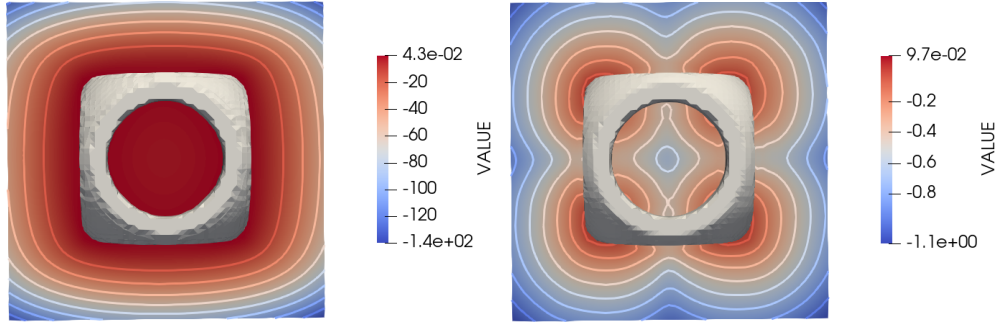


Figure 5. Zero-level set and filled contour plot on a slice for the original implicit (left) and the SDF obtained by minimizing the loss function in Equation (7) (right). The contour lines shown on each slice are equidistributed (the difference between two contour line is a constant within one plot).

4.2.3. 3D Examples

We use a more complex model to illustrate our approach in 3D. See the image on the left in Figure 5. This object is built from algebraic primitives (a sphere, a box, and three cylinders) and constructive solid geometry (CSG) operations implemented by R-functions [Pasko et al. 1995]. The term *algebraic* here refers to the fact that we are not using distance-based primitives. For example, we use

$$1 - x^2 - y^2 - z^2 \quad \text{and} \quad 1 - x^2 - y^2$$

for a sphere and a cylinder, respectively.

Figure 5 (left) shows with a filled contour plot on a given slice of the object that the function f defining the object does not correspond to the distance to the surface S . Figure 5 (right) shows the solution to Equation (5). It corresponds to minimizing the loss function in Equation (7). We used 15,000 iterations of ADAM (similar to the 1D and 2D examples). Notice how the solution delivers a good approximation of the signed distance to the surface. Since the original implicit f is embedded in the last layer of the deep neural network, the zero-level set of f is exactly preserved.

Figure 6 illustrates a more complex object (a microstructure) made of blended torii, tiled along the three main directions. This model is also built from algebraic implicits, the sawtooth function, and blend functions, and it does not correspond to the distance function (left image). The right image shows the result obtained by re-distancing, where we can see in the filled contour plot on the cross-section the normalized behavior of the field.

5. Conclusion

We proposed in this short article a technique for re-distancing an implicit surface. Given a function f whose zero-level set defines a geometric surface S , we want to

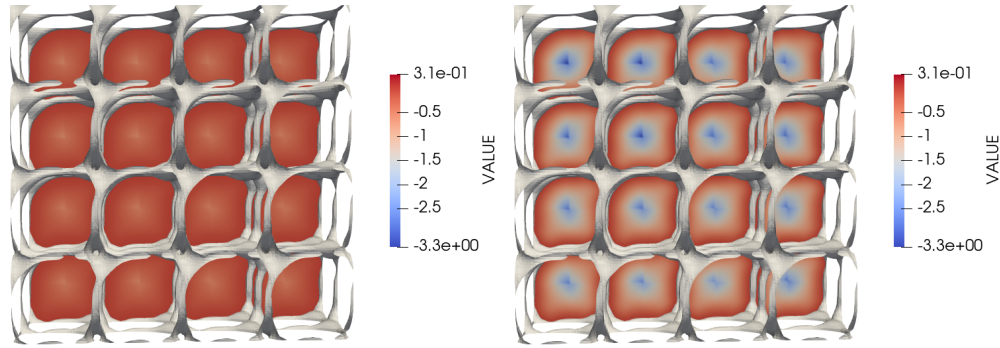


Figure 6. A more complex model made of blended torii, infinitely tiled in three directions. Left: Zero-level set and filled contour plot on a slice for the original implicit. Right: The SDF obtained by minimizing the loss function in Equation (7).

compute the signed distance to S (or an approximation), while preserving exactly the zero-level set of f . We do that by embedding the function f in the last layer of a deep neural network, which is then trained on a loss function derived from an eikonal equation. Alternatively, we propose a different loss function corresponding to the p -Poisson problem $\Delta_p u = -1$, whose solution delivers an approximation to the distance function.

Acknowledgements

The author would like to thank the reviewers for their thoughtful comments and feedback.

References

- ATZMON, M. AND LIPMAN, Y. SAL: Sign agnostic learning of shapes from raw data. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2562–2571. IEEE, 2020. URL: <https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.00264>. 187, 190
- ATZMON, M. AND LIPMAN, Y. SALD: Sign agnostic learning with derivatives. In *International Conference on Learning Representations, ICLR*. Curran Associates Inc., 2021. URL: <https://openreview.net/forum?id=7EDgLu9reQD>. 187
- ATZMON, M., HAIM, N., YARIV, L., ISRAELOV, O., MARON, H., AND LIPMAN, Y. Controlling neural level sets. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, volume 32, pages 2032–2041. Curran Associates Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/b20bb95ab626d93fd976af958fbc61ba-Abstract.html>. 187

- BELYAEV, A. AND FAYOLLE, P.-A. An ADMM-based scheme for distance function approximation. *Numerical Algorithms*, 84(3):983–996, 2020. URL: <https://link.springer.com/article/10.1007/s11075-019-00789-5>. 188
- BELYAEV, A. G. AND FAYOLLE, P.-A. On variational and PDE-based distance function approximations. *Computer Graphics Forum*, 34(8):104–118, 2015. URL: <https://doi.org/10.1111/cgf.12611>. 188
- BLECHSCHMIDT, J. AND ERNST, O. G. Three ways to solve partial differential equations with neural networks—A review. *GAMM-Mitteilungen*, 44(2):e202100006, 2021. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/gamm.202100006>. 187
- CHIBANE, J., MIR, M. A., AND PONS-MOLL, G. Neural unsigned distance fields for implicit function learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33, pages 21638–21652. Curran Associates Inc., 2020. URL: <https://proceedings.neurips.cc/paper/2020/hash/f69e505b08403ad2298b9f262659929a-Abstract.html>. 187
- CRANDALL, M. G. AND LIONS, P.-L. Viscosity solutions of Hamilton–Jacobi equations. *Transactions of the American mathematical society*, 277(1):1–42, 1983. URL: <https://www.ams.org/journals/tran/1983-277-01/S0002-9947-1983-0690039-8/>. 188
- DAVIES, T., NOWROUZEZAHRAI, D., AND JACOBSON, A. On the effectiveness of weight-encoded neural implicit 3D shapes. ArXiv preprint arXiv:2009.09808 [cs.GR], 2021. URL: <https://arxiv.org/abs/2009.09808>. 187
- FAYOLLE, P.-A. AND BELYAEV, A. p -Laplace diffusion for distance function estimation, optimal transport approximation, and image enhancement. *Computer Aided Geometric Design*, 67:1–20, 2018. URL: <https://doi.org/10.1016/j.cagd.2018.09.003>. 188
- GROPP, A., YARIV, L., HAIM, N., ATZMON, M., AND LIPMAN, Y. Implicit geometric regularization for learning shapes. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 3789–3799. PMLR, 2020. URL: <http://proceedings.mlr.press/v119/gropp20a.html>. 187, 190
- KINGMA, D. P. AND BA, J. Adam: A method for stochastic optimization. ArXiv preprint arXiv:1412.6980, 2014. URL: <https://arxiv.org/abs/1412.6980>. 189, 190
- LORENSEN, W. E. AND CLINE, H. E. Marching Cubes: A high resolution 3D surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987. URL: <https://dl.acm.org/doi/10.1145/37402.37422>. 186
- OSHER, S. AND FEDKIW, R. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153 of *Applied Mathematical Sciences*. Springer Science & Business Media, 2003. URL: <https://link.springer.com/book/10.1007/b98879>. 186

- PARK, J. J., FLORENCE, P., STRAUB, J., NEWCOMBE, R., AND LOVEGROVE, S. DeepSDF: Learning continuous signed distance functions for shape representation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 165–174. IEEE, 2019. URL: <https://ieeexplore.ieee.org/document/8954065>. 190
- PASKO, A., ADZHIEV, V., SOURIN, A., AND SAVCHENKO, V. Function representation in geometric modeling: Concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995. URL: <https://link.springer.com/article/10.1007/BF02464333>. 190, 193
- PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., DESMAISON, A., KOPF, A., YANG, E., DEVITO, Z., RAISON, M., TEJANI, A., CHILAMKURTHY, S., STEINER, B., FANG, L., BAI, J., AND CHINTALA, S. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 8024–8035. Curran Associates, Inc., 2019. URL: <https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html>. 189
- ROBBINS, H. AND MONRO, S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951. URL: <https://doi.org/10.1214/aoms/1177729586>. 189
- SETHIAN, J. A. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC39986/>. 186
- SHAPIRO, V. Semi-analytic geometry with R-functions. *Acta Numerica*, 16:239–303, 2007. URL: <https://doi.org/10.1017/S096249290631001X>. 190
- SITZMANN, V., CHAN, E. R., TUCKER, R., SNAVELY, N., AND WETZSTEIN, G. MetaSDF: Meta-learning signed distance functions. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33, pages 10136–10147. Curran Associates Inc., 2020a. URL: <https://proceedings.neurips.cc/paper/2020/hash/731c83db8d2ff01bdc000083fd3c3740-Abstract.html>. 187
- SITZMANN, V., MARTEL, J., BERGMAN, A., LINDELL, D., AND WETZSTEIN, G. Implicit neural representations with periodic activation functions. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020*, volume 33, pages 7462–7473. Curran Associates, Inc., 2020b. URL: <https://proceedings.neurips.cc/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html>. 187, 190
- SUSSMAN, M., SMEREKA, P., AND OSHER, S. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational physics*, 114(1):146–159, 1994. URL: <https://doi.org/10.1006/jcph.1994.1155>. 186
- TAKIKAWA, T., LITALIEN, J., YIN, K., KREIS, K., LOOP, C., NOWROUZEZAHRAI, D., JACOBSON, A., MCGUIRE, M., AND FIDLER, S. Neural geometric level of detail: Real-time

rendering with implicit 3D shapes. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11353–11362. IEEE, 2021. URL: <https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.01120>. 187

WEINAN, E. AND YU, B. The Deep Ritz Method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018. URL: <https://link.springer.com/article/10.1007/s40304-018-0127-z>. 187

ZHAO, H. A fast sweeping method for Eikonal equations. *Mathematics of Computation*, 74(250):603–627, 2005. URL: <https://www.ams.org/journals/mcom/2005-74-250/S0025-5718-04-01678-3/>. 186

Index of Supplemental Materials

- Sample code is available at
<https://jcgt.org/published/0014/01/09/code.zip>.

Author Contact Information

Pierre-Alain Fayolle
University of Aizu
Ikki machi
Aizu-Wakamatsu, Japan
fayolle@u-aizu.ac.jp

Pierre-Alain Fayolle, A Zero-Level Set Preserving Technique for Signed Distance Function Computation from an Implicit Surface, *Journal of Computer Graphics Techniques (JCGT)*, vol. 14, no. 1, 185–197, 2025
<http://jcgt.org/published/0014/01/09/>

Received: 2022-07-17

Recommended: 2025-02-02

Published: 2025-05-16

Corresponding Editor: Bernd Bickel

Editor-in-Chief: Eric Haines

© 2025 Pierre-Alain Fayolle (the Authors).

The Authors provide this document (the Work) under the Creative Commons CC BY-ND 4.0 license available online at <http://creativecommons.org/licenses/by-nd/4.0/>. The Authors further grant permission for reuse of images and text from the first page of the Work, provided that the reuse is for the purpose of promoting and/or summarizing the Work in scholarly venues and that any reuse is accompanied by a scientific citation to the Work.

