High–Fidelity Haptic Rendering through Implicit Neural Force Representation

Christoforos Vlachos$^{1[0000-0003-2863-7823]}$ and Konstantinos Moustakas$^{1[0000-0001-7617-227X]}$

Department of Electrical and Computer Engineering, University of Patras, Rion-Patras, Greece
{chris.vlachos,moustakas}@ece.upatras.gr

Abstract. Recent research has demonstrated that neural networks using periodic nonlinearities may be used for implicit representation and reconstruction of continuous-time signals. Starting with a previously published network for representing the Signed Distance Function (SDF) of a mesh surface, we extend the concept and lay the foundation for introducing the additional representation of the Unit Normal Function (UNF). With the representation of these two functions at hand, we construct a penalty-based haptic rendering method. Our experiments suggest that this proposed method is able to handle very large meshes better than other competing alternatives, producing high-fidelity forces, free of discontinuities, by sampling a continuous implicit force function at the desired spatial accuracy.

Keywords: Haptic Rendering · Force Rendering · Implicit Representation · Neural Networks · Human–Computer Interaction.

1 Introduction

Human sensory organs receive many different forms of input. Strengthening Human–Computer Interaction (HCI) can be definitely attained by mimicking these inputs to a sufficient degree. Although visual and auditory input to a person is nowadays considered mostly trivial, haptic input remains challenging. Consequently, the development of haptic devices, able to output haptic signals which are then sensed by a person, has attracted much attention. However, the haptic rendering methods that drive these devices feature computationally expensive operations [16]. Therefore, it raises no eyebrows how the importance of haptic rendering has been highlighted in many publications.

1.1 Related Work

There have been many attempts to categorize haptic rendering algorithms. One such categorization, which is presented in [27] is surface-based and volume-based techniques. This distinction, which is based on graphical rendering categorization, separates algorithms between those using polygonal representations (explicit) or parametric representations (implicit) and those opting to represent meshes using voxels. Another way of classifying haptic algorithms, mentioned in the same article, is point-based vs. ray-based algorithms. This classification serves to differentiate between algorithms that take into account only the contact point of the haptic device, ignoring the rest of the probe, and those that model the entire probe as a ray before proceeding to calculate the response force. Finally, another grouping of haptic rendering techniques that may be encountered in recent publications [12] is the 3-degree-of-freedom (3DOF) vs. the 6-degree-of-freedom (6DOF) algorithms. 3DOF algorithms only account for the probe's position in 3D space, whereas 6DOF algorithms also consider the probe's rotation in determining the response force of the collision.

Similar ways of categorization can be carried out for haptic interfaces, but that falls beyond the scope of this paper. We refer the reader to [27] as well as [12] for obtaining further information regarding haptic devices.

In the last few years, a substantial assortment of haptic algorithms have surfaced. Since the god-object method was proposed in 1995 [30] and the voxel method in 1999 [14], numerous innovations have been made. A highly detailed account of the advancements in haptic rendering techniques in the last few years can be found in [11]. More recent methods have tackled the problem of continuous collision detection with deformable meshes [4], pseudo-rendering of constraints [13], haptic rendering of point clouds [29], leading to new and exciting fields of research.

Another domain that has been of much interest to the haptic rendering community has been data-driven haptic rendering. Introduced by Hover et al. in 2008 [6] and expanded by the same group in the following years [5,7,24,25]. Additional research on the area was performed by Choi's group [2,18,23] has elevated the subject and can nowadays be considered a mature field of research.

Despite those innovations, there is a weakness common to most haptic algorithms: their inability to handle very detailed meshes, with lots of vertices and triangles, without giving up either quality or speed of calculations. Moreover, interpolation-based methods commonly handle discontinuities badly, undesirably eliminating them where present or introducing them where absent. To resolve those issues, it was early on that the researchers' focus shifted to implicitly represented surfaces for haptic rendering [10, 16, 17, 22]. Furthermore, recent work has focused on the use of neural networks to represent an object in 3D space [1,8,15,20]. Fusing these concepts has been the inspiration for our proposed method.

1.2 Motivation and Contribution

In their recent work [26], Sitzmann et al. have introduced the SInusoidal REpresentation Network (SIREN), a type of neural network which uses periodic activation functions in order to implicitly represent various types of signals. One such signal, demonstrated on that paper, is the Signed Distance Function (SDF) of a mesh surface.

In this paper, we show that it is possible to extend their proposed network to include computation of the Unit Normal Function (UNF) as well. This way, we arrive at an elegant solution for quickly and accurately approximating the response force generated by collision with the mesh, to be used in a haptic rendering context where high-fidelity discontinuity-free forces are necessary.

Specifically, our contribution is creating a haptic rendering method, which we call PANDIS (Periodic Activations for Normals and DIStances), that:

- is able to estimate the collision response force that should be generated and fed back to the user, in a predictable time.
- can be easily implemented and adapted to be used with any mesh. The haptic rendering method is mesh and resolution agnostic during run-time.
- does not contain unwanted discontinuities. Both the SDF and the UNF are implicitly represented continuous functions that can be sampled at the desired spatial accuracy. This implicitly generates smooth force fields and haptic interaction that leads to force feedback without discontinuities or the need for post-processing in the haptic rendering pipeline as is the case, for example, in the force shading method.

2 Methods

2.1 Formulation

Penalty-based algorithms model the response force between objects as a spring force whose magnitude increases as the objects penetrate deeper into one another [12]. Thus, to determine the collision response force, we will need to have knowledge of both the magnitude and the direction of the spring force. To acquire these, we will use two implicit functions, the Signed Distance Function (SDF) and the Unit Normal Function (UNF).

Given a space Ω and a boundary $\partial\Omega$ within that space, we can divide the space into an exterior region Ω^+ and an interior region Ω^-. A distance function $d(x)$ is an implicit function equal to the distance between x and the closest point on the boundary [19]. From this definition, it is apparent that on the boundary itself $d(x) = 0$, therefore $\partial\Omega$ represents the zero-level set of $d(x)$ and we can refer to it simply as Ω_0.

It should also be mentioned that, since d represents Euclidean distance, the following equation should hold:

$$|\nabla d| = 1 \tag{1}$$

This is intuitive, since moving twice as far away from the boundary, we would expect the distance function's value to double.

We can now adapt the definition of the (unsigned) distance function in order to define its signed counterpart. Whereas the unsigned distance function $d(x)$ is always non-negative, the Signed Distance Function (commonly abbreviated as SDF) $\phi(x)$ is positive in Ω^+ and negative in Ω^-, while maintaining its magnitude. In other words:

$$\phi(x) = \begin{cases} d(x) = 0 & x \in \partial\Omega \\ d(x) & x \in \Omega^+ \\ -d(x) & x \in \Omega^- \end{cases}$$

Similarly to the unsigned distance function, for the SDF it holds that:

$$|\nabla\phi| = 1 \tag{2}$$

Regarding Equations 1 and 2, it should be mentioned that distance functions are differentiable almost everywhere, but not completely. In the case of the unsigned distance function one quickly realizes that the boundary constitutes a local minimum of d and the gradient cannot be defined there [28]. Despite this being a problem only for the unsigned distance function (since SDFs are monotonic), there exist other points where both kinds of distance functions are not differentiable, specifically those where the closest boundary point is ambiguous (it is a "tie" between at least two points).

We also define the Unit Normal Function (UNF) $\psi(x)$ as the vector function that always points towards or away from the closest point of the boundary. For all points in Ω that the SDF is differentiable, $\psi = \nabla\phi$.

We are interested in determining an implicit approximation of the SDF of a mesh surface. We call this approximation $\Phi(x)$. An appropriate loss function for the training of a SIREN that is used to predict the SDF at a specific point in 3D space would then be [26]:

$$\begin{aligned} \mathcal{L}_{sdf} = &\int_{\Omega} \| |\nabla_x \Phi(x)| - 1 \| dx \\ &+ \int_{\Omega_0} (1 - \langle \nabla_x \Phi(x), n(x) \rangle) dx \\ &+ \int_{\Omega_0} \|\Phi(x)\| dx \\ &+ \int_{\Omega \setminus \Omega_0} \exp(-a \cdot |\Phi(x)|) dx \end{aligned} \tag{3}$$

This loss function treats determining the SDF as a Boundary Value Problem (BVP). Ω represents the entire 3D space whereas Ω_0 represents only those points where the SDF is zero (the zero-level set). In the final integral, $a \gg 1$.

Since $\Omega \subseteq R^3$, the SDF is differentiable almost everywhere and its gradient satisfies the following Eikonal equation:

$$|\nabla_x \Phi(x)| = 1$$

Looking back at the loss function defined in Equation 3, the first integral simply enforces the Eikonal equation on Ω.
The second integral ensures that, on the surface, the gradient of the SDF, $\nabla_x \Phi(x)$, and the surface normals, $n(x)$, point in the same direction. After all, on Ω_0 these

two should be identical.
The third integral penalizes on-surface points that have an SDF that is not equal to zero, while the fourth penalizes off-surface points having an SDF close to zero.

While the SDF is crucial to calculating the magnitude of the response force used for haptic rendering, the direction is also needed. For that, we could use the same SIREN to implicitly represent the UNF of the mesh. Let's call this implicit representation $\Psi(x)$. It is important to note that, while $\Phi(x) \in R$, this is not the case for $\Psi(x)$, as $\Psi(x) \in R^3$.

Inspired by $\mathcal{L}_{sdf}$, we can begin to formulate our own loss function, to aid with the UNF prediction:

$$\begin{aligned}\mathcal{L}_{unf} = &\int_{\Omega} (\|\Psi(x)\| - 1)dx \\ &+ \int_{\Omega_0} (1 - \langle\Psi(x), n(x)\rangle)dx \\ &+ \int_{\Omega \setminus \Omega_0} (1 - \langle\Psi(x), \nabla_x \Phi(x)\rangle)dx\end{aligned} \tag{4}$$

The first integral forces the UNF to have unit values, while the other two integrals have it pointing in the same direction as the mesh normals, where available, or the gradient of the SDF, in places where the normals are not defined. By carefully inspecting the two loss functions, it can be understood that, ideally, $\Psi(x) \equiv \nabla_x \Phi(x)$.

It should be noted that, instead of using some numerical method to differentiate the SDF, we opted to encode it as part of the same neural network. In this way, we arrive at an elegant solution where, adding barely any complexity, we are able to obtain both the SDF and the UNF with a single inference pass.

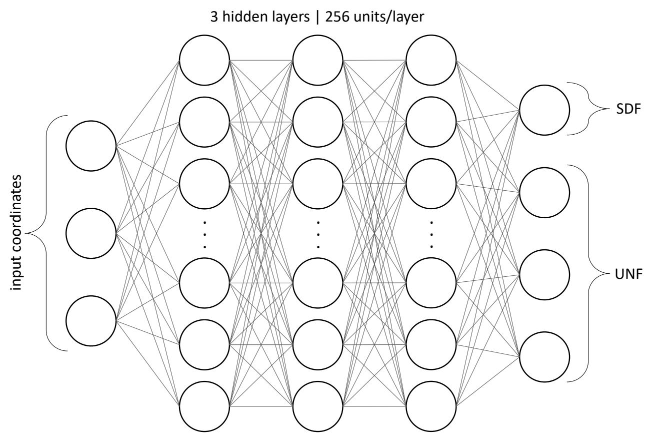

Fig. 1. The architecture of our proposed SIREN. The probe coordinates are entered in the Cartesian form. The first output is the SDF and the remaining 3 are the UNF.

A SIREN is now constructed, following the architecture shown in Fig. 1. That is, a Multi-Layer Perceptron (MLP) made up of one input layer, one output layer, as well as 3 hidden layers, each consisting of 256 sinusoidally-activated units. All layers are Fully Connected (FC). The 3D space coordinates of each mesh's vertices are input to the network and after passing through the three hidden layers, both $\Phi(x)$ and $\Psi(x)$ are output. For the training, we use the Adam optimizer and as a loss function a combination of the ones we defined previously:

$$\mathcal{L}_{total} = \mathcal{L}_{sdf} + \mathcal{L}_{unf} \tag{5}$$

One advantage of using a SIREN instead of an ordinary ReLU-based neural network for implicitly representing the SDF and the UNF is that the SIREN is able to represent these functions in such a way that their respective derivatives are also retained. Thus, we expect response forces that do not suffer from discontinuities as much as other haptic rendering algorithms.

It is moreover worth mentioning that each component (i.e., each integral) of the final loss function was given a weight, each discovered by trial and error and was validated experimentally. These are 50, 100, 3000, 100, 50, 100, and 100, respectively.

2.2 Training

All training took place on a system with an AMD Ryzen 5 3600X CPU and an NVIDIA GeForce RTX 2060 SUPER GPU with 8GB GDDR6 VRAM. The installed memory was 32GB DDR4 SDRAM. The operating system used was Windows 10, version 22H2.

Using PyTorch, we implemented the network we described and trained it on three different, quite large meshes, creating three models. These meshes were taken off the Stanford 3D Scanning Repository and are the Stanford Bunny, the Stanford Armadillo, and the Stanford Dragon. Each model was trained on its respective mesh for 10,000 epochs. The number of vertices and faces of each mesh, along with the training time (for 10,000 epochs), is displayed in Table 1.

Table 1. Number of vertices, faces, and training time of the meshes that were used for training.

Model	Vertices	Faces	Training Time
Bunny	35,947	69,451	09:29
Armadillo	172,947	345,944	47:11
Dragon	437,645	1,132,830	2:11:47

3 Results

To confirm the quality of our results, our testing was focused on the models of the Stanford Bunny, the Stanford Armadillo, and the Stanford Dragon. Firstly, we created a vector plot (also called a "quiver plot"), plotting the response force according to Hooke's Law as:

$$F = -kx \tag{6}$$

where x is the SDF. The direction of the response force is indicated by the UNF. The generated vector plots along a planar section of each of the three meshes are shown in Fig. 2.

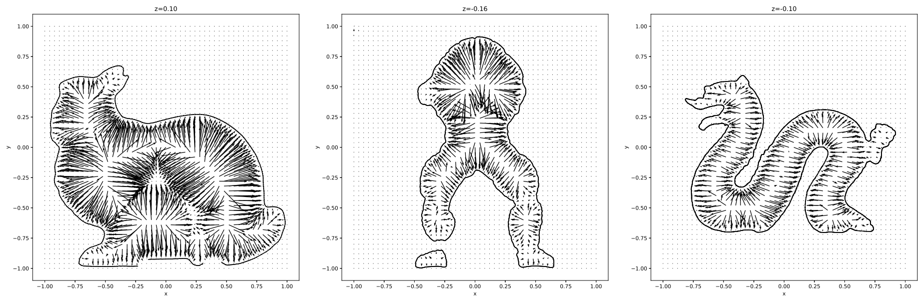

Fig. 2. Vector plot of the response force generated along a planar section of the (Left) Stanford Bunny, (Center) Stanford Armadillo, and (Right) Stanford Dragon.

After verifying that the model generates seemingly correct forces (2), we serialized the model, for use outside of PyTorch. We proceeded to write a C/C++ program using OpenHaptics® and LibTorch to enable utilizing a haptic rendering device to render the Stanford Armadillo using the forces produced and calculated by the neural network. Two such devices were used in our tests, the first being 3D SYSTEMS's flagship haptic device, the *Touch™* (formerly the *Phantom Omni*), the other being the oldest and less accurate *Touch™ 3D stylus*. Both devices as part of our experimental setup are visible in Fig. 3.

Although some steps have been taken towards standardizing the benchmarking of haptic rendering, they have either largely focused on the benchmarking of the haptic hardware itself, suffer from limitations, or ultimately failed to see widespread use [21]. It is because of this that we decided to focus our benchmarking on comparison with widespread, easily–implemented algorithms and objective statistical analyses.

We needed some comparison targets. To this extent, we implemented a few other algorithms for calculating the SDF and the UNF of a triangle mesh. First, we implemented a method for analytically calculating the above functions in order to obtain ground-truth results (GT). Then, we ran that method on a 3D grid, effectively splitting the mesh's AABB into voxels. In this way, we can report additional interpolated results. We implemented Nearest Neighbor(NN-i) as well as Trilinear interpolation(Tri-i) options. It should be noted that trilinear interpolation from a 3D grid could be considered State-of-the-Art, as featured in recent publications [3, 9]. We used two grid sizes: 32^3 and 64^3.

Table 2. Setup time (Top) and space (Bottom) required per method, per model. GT stands for Ground Truth, NN-i for Nearest Neighbor Interpolation, Tri-i for Trilinear Interpolation and PANDIS is the method we are proposing.

Model	Preprocessing/Training time					
	GT	NN-i		Tri-i		PANDIS
		32^3	64^3	32^3	64^3	
Bunny	0	07:38	58:52	07:38	58:52	09:29
Armadillo	0	40:21	5:25:00	40:21	5:25:00	47:11
Dragon	0	1:41:11	12:54:56	1:41:11	12:54:56	2:11:47

Model	Storage space					
	GT	NN-i		Tri-i		PANDIS
		32^3	64^3	32^3	64^3	
Bunny	0	384 KB	3 MB	384 KB	3 MB	800 KB
Armadillo	0	384 KB	3 MB	384 KB	3 MB	800 KB
Dragon	0	384 KB	3 MB	384 KB	3 MB	800 KB

3.1 Setup

The Analytical algorithm does not require any preparatory computations, since it computes everything in real time. Thus it requires no time or space in advance (Table 2). The story changes when it comes to the next algorithms. Both the Nearest Neighbor and the Trilinear Interpolation methods need a precomputed grid. Both the computation time and space change cubically as the grid size increases, as long as we are talking about the same model. We observe that, for a grid size of 32^3, the interpolation–based methods need preprocessing time that is similar to the training time of PANDIS, while PANDIS falls behind when it comes to storage requirements. That being said, increasing the grid size to 64^6 leaves the interpolation-based methods well outperformed by the SIREN–based method on both regards. It should be noted that PANDIS, just as the others chosen, is mesh-agnostic when it comes to the required storage space.

Table 3. Average computation time (in milliseconds) of response force. Times on top are from the *Touch™*, times below (inside parentheses) are from the *Touch™ 3D stylus*.

Model	GT	NN-i		Tri-i		PANDIS
		32^3	64^3	32^3	64^3	
Bunny	13.851	<0.001	<0.001	<0.001	<0.001	0.745
	(14.053)	(<0.001)	(<0.001)	(<0.001)	(<0.001)	(0.741)
Armadillo	73.125	<0.001	<0.001	<0.001	<0.001	0.769
	(78.443)	(<0.001)	(<0.001)	(<0.001)	(<0.001)	(0.743)
Dragon	182.085	<0.001	<0.001	<0.001	<0.001	0.756
	(181.604)	(<0.001)	(<0.001)	(<0.001)	(<0.001)	(0.750)

3.2 Update Rate

Having finished preparing all the methods for testing, we measured the update rate capability of each one and report the average computation time, in milliseconds, for each method in Table 3. We reiterate that, for haptic rendering, the target rate is 1 kHz, which corresponds to a computation time of at most 1 ms.

From our testing, it is abundantly clear that the analytical method is too slow in dealing with large meshes. In contrast, the interpolation methods both respond very quickly. While PANDIS's speed is nowhere near that of the interpolation methods, it should be noted that it is still within target, 1 ms being the 97th percentile rank (i.e., less than 3% of the updates fall outside the target rate). Besides, we expect PANDIS to make up any lost ground when we look at the quality of the predicted forces.

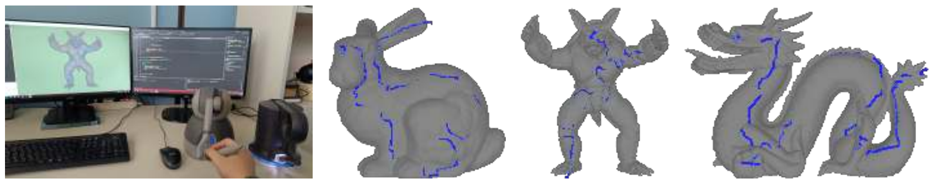

Fig. 3. (Leftmost) Our experimental setup, featuring the haptic devices used. (Left center – Rightmost) The synthetic trajectories used for testing the quality of the responses generated by each algorithm: (Left center) Bunny. (Right center) Armadillo. (Rightmost) Dragon.

3.3 Quality

To measure the quality and accuracy of the forces produced, we traced and recorded a trajectory on the surface of each mesh (actually, penetrating the surface slightly, since we are testing penalty-based algorithms), making sure that the trajectory contained a variety of areas of the mesh's surface with vastly different gradients. The specific traces were generated using the state-of-the-art trilinear interpolation-based method and can be viewed in Fig. 3.

Table 4. Average error, as defined by Equation 7, per method, per model.

Model	GT	NN-i		Tri-i		PANDIS
		32^3	64^3	32^3	64^3	
Bunny	0	0.086	0.031	0.057	0.026	0.006
Armadillo	0	0.139	0.081	0.099	0.046	0.022
Dragon	0	0.176	0.054	0.171	0.041	0.068

We then applied all methods to the synthetic trajectories we had collected. Defining an Average Error as

$$\text{Average Error}_i = \sum_j \frac{1}{j} \left(\|\text{force}_{i,j} - \text{force}_{\text{GT},j}\| \right), \tag{7}$$

where $i = \{\text{GT, NN-i, Tri-i, PANDIS}\}$, enables us to determine how each method performed, with respect to the accuracy of the predicted collision response force. These results can be seen in Table 4.

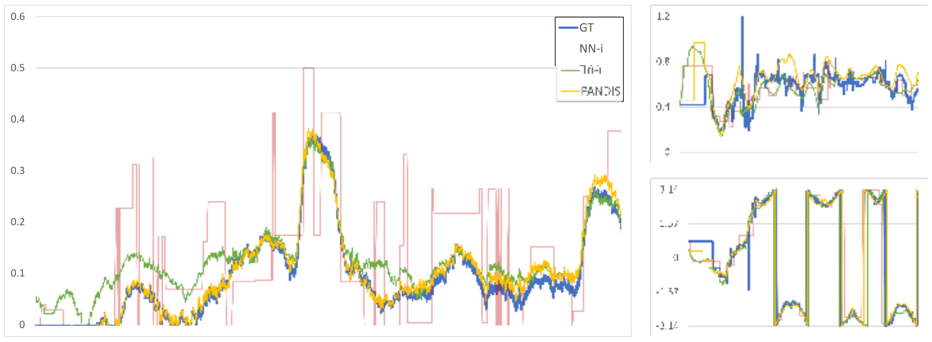

Fig. 4. Response force returned per method for a subset of the Armadillo's traced path, converted to spherical coordinates. (Left) r. (Top right) θ. (Bottom right) ϕ.

One can immediately notice that on both the Bunny and the Armadillo our method produces more accurate results, even on the increased grid resolution of 64^3. In the case of the Dragon, PANDIS loses—only barely—to the interpolation-based methods on a resolution of 64^3, albeit at the cost of significantly more preprocessing time and storage space.

Aiming to provide the reader with a better visual representation, we plot a subset of the response force produced by each algorithm in the case of the Armadillo model, in Fig. 4. It is evident that our proposed method tracks the Ground Truth results well, eliminating discontinuities present in other methods (such as nearest neighbor interpolation) and should be a serious consideration where a haptic representation of large meshes is required.

3.4 Theoretical Validation

In order to have a more complete picture of the quality and accuracy of the generated forces, we extended our testing to a sphere, whose SDF as well as the UNF can be defined in closed-form. Regarding a sphere located on the origin with radius r, it is well-known that its SDF is given by:

$$\phi(x,y,z) = \sqrt{x^2+y^2+z^2} - r$$

Consequently, the UNF comes to:

$$\psi(x,y,z) = \nabla\phi(x,y,z) = \left(\frac{x}{\sqrt{x^2+y^2+z^2}}, \frac{y}{\sqrt{x^2+y^2+z^2}}, \frac{z}{\sqrt{x^2+y^2+z^2}}\right)$$

Since an implicit representation of the sphere was readily available, we were able to also extend our arsenal of methods to include the original implicit haptic rendering method of Salisbury and Tarr [22] as well as two other state-of-the-art methods of the same family [10, 16]. We synthesized a trajectory on the sphere and the Ground Truth which we could trivially calculate from the closed-form SDF and UNF. We calculated the forces in the trajectory using all methods we had implemented, for more rigorous testing, and assessed the quality using the error defined in Equation 7. This led to the results of Table 5.

Table 5. Average error, as defined by Equation 7, per method, on a sphere. For the interpolation-based methods (as well as [10]) a resolution of 100^3 was used, for [16] 642 support planes were created, and for PANDIS the underlying SIREN was trained on 2737 uniformly sampled points on the surface of the sphere.

Method	GT	NN-i	Tri-i	[22]	[10]	[16]	PANDIS
Error	0	0.097	0.082	0.083	0.367	0.296	0.162

One can see that PANDIS outperforms the other state-of-the-art methods ([10, 16]) and produces results that are comparable to the remaining methods. It is obvious that interpolation-based methods play well with simple shapes, such as a sphere but, as previously proven, fail to keep that lead in complex meshes. On the other hand, the original implicit haptic rendering method of Salisbury and Tarr [22] does very well, but it should be noted that it only works on shapes whose implicit representation can be defined implicitly.

Therefore, we conclude that even playing from a losing position, on a scenario than is far from ideal for PANDIS (which shines brightest in the case of very complex meshes), our proposed method is able to keep up and even surpass other state-of-the-art methods.

4 Conclusion and Discussion

We have used a sinusoidally-activated neural network (SIREN) to create an implicit representation of both the Signed Distance Function (SDF) and the Unit Normal Function (UNF) with great accuracy. We have then proposed a new haptic rendering algorithm that makes use of this representation and succeeds in estimating the collision response force quickly, with high fidelity, and free of discontinuities. Furthermore, this method is easily scalable to very large meshes with no decrease in the update rate.

Although the aforementioned advances tackle critical and basic issues in haptic rendering, further advances have to take place in order to make machine learning based haptic rendering a game changer in haptic interaction. The basic current limitation is the fact that a deformation of the object would require a retraining of the SDF and UNF, making real-time deformation not feasible at the present stage. Moreover, both the SDF and UNF can be strictly theoretically defined only on closed manifold surfaces. Extension of the proposed approach solving these limitations through modern adaptive neural networks and utilization of pseudo-manifold surfaces are challenging directions of future work.

Acknowledgments. The research project is implemented in the framework of H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union – NextGenerationEU (H.F.R.I. Project Number: 16469.).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Atzmon, M., Lipman, Y.: Sal: Sign agnostic learning of shapes from raw data. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2565–2574 (2020). https://doi.org/10.1109/cvpr42600.2020.00264
2. Cha, H., Bhardwaj, A., Choi, S.: Data-driven haptic modeling and rendering of viscoelastic behavior using fractional derivatives. IEEE Access **10**, 130894–130907 (2022). https://doi.org/10.1109/ACCESS.2022.3230065

3. Corenthy, L., Otaduy, M.A., Pastor, L., Garcia, M.: Volume haptics with topology-consistent isosurfaces. IEEE Transactions on Haptics **8**(4), 480–491 (2015). https://doi.org/10.1109/TOH.2015.2466239
4. Ding, H., Mitake, H., Hasegawa, S.: Continuous collision detection for virtual proxy haptic rendering of deformable triangular mesh models. IEEE Transactions on Haptics **12**(4), 624–634 (2019). https://doi.org/10.1109/TOH.2019.2934104
5. Höver, R., Harders, M.: Measuring and incorporating slip in data-driven haptic rendering. In: 2010 IEEE Haptics Symposium. pp. 175–182. IEEE (2010). https://doi.org/10.1109/HAPTIC.2010.5444658
6. Hover, R., Harders, M., Székely, G.: Data-driven haptic rendering of visco-elastic effects. In: 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. pp. 201–208. IEEE (2008). https://doi.org/10.1109/HAPTICS.2008.4479943
7. Hover, R., Kosa, G., Szekely, G., Harders, M.: Data-driven haptic rendering—from viscous fluids to visco-elastic solids. IEEE Transactions on Haptics **2**(1), 15–27 (2009). https://doi.org/10.1109/TOH.2009.2
8. Jiang, C., Sud, A., Makadia, A., Huang, J., Nießner, M., Funkhouser, T., et al.: Local implicit grid representations for 3d scenes. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6001–6010 (2020). https://doi.org/10.1109/cvpr42600.2020.00604
9. Kaluschke, M., Yin, M.S., Haddawy, P., Srimaneekarn, N., Saikaew, P., Zachmann, G.: A shared haptic virtual environment for dental surgical skill training. In: 2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW). pp. 347–352 (2021). https://doi.org/10.1109/VRW52623.2021.00069
10. Kim, L., Sukhatme, G., Desbrun, M.: A haptic-rendering technique based on hybrid surface representation. IEEE Computer Graphics and Applications **24**(2), 66–75 (2004). https://doi.org/10.1109/MCG.2004.1274064
11. Laycock, S.D., Day, A.: A survey of haptic rendering techniques. In: Computer graphics forum. vol. 26, pp. 50–65. Wiley Online Library (2007). https://doi.org/10.1111/j.1467-8659.2007.00945.x
12. Lin, M.C., Otaduy, M.: Haptic rendering: foundations, algorithms, and applications. CRC Press (2008). https://doi.org/10.1201/b10636
13. Lobo, D., Otaduy, M.A.: Rendering of constraints with underactuated haptic devices. IEEE Transactions on Haptics **13**(4), 699–708 (2020). https://doi.org/10.1109/TOH.2020.2981932
14. McNeely, W.A., Puterbaugh, K.D., Troy, J.J.: Six degree-of-freedom haptic rendering using voxel sampling. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques. pp. 401—408. SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co. (1999). https://doi.org/10.1145/311535.311600
15. Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., Geiger, A.: Occupancy networks: Learning 3d reconstruction in function space. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4460–4470 (2019). https://doi.org/10.1109/CVPR.2019.00459
16. Moustakas, K.: 6dof haptic rendering using distance maps over implicit representations. Multimedia Tools and Applications **75**(8), 4543–4557 (2016). https://doi.org/10.1007/s11042-015-2490-z
17. Moustakas, K., Tzovaras, D., Strintzis, M.G.: Sq-map: Efficient layered collision detection and haptic rendering. IEEE Transactions on Visualization and Computer Graphics **13**(1), 80–93 (2006). https://doi.org/10.1109/TVCG.2007.20

18. Osgouei, R.H., Kim, J.R., Choi, S.: Data-driven texture modeling and rendering on electrovibration display. IEEE Transactions on Haptics **13**(2), 298–311 (2019). https://doi.org/10.1109/TOH.2019.2932990
19. Osher, S., Fedkiw, R.: Signed distance functions. In: Level Set Methods and Dynamic Implicit Surfaces, pp. 17–22. Springer New York (2003). https://doi.org/10.1007/0-387-22746-6_2
20. Park, J.J., Florence, P., Straub, J., Newcombe, R., Lovegrove, S.: Deepsdf: Learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 165–174 (2019). https://doi.org/10.1109/cvpr.2019.00025
21. Ruffaldi, E., Morris, D., Edmunds, T., Barbagli, F., Pai, D.K.: Standardized evaluation of haptic rendering systems. In: 2006 14th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems. pp. 225–232. IEEE (2006). https://doi.org/10.1109/haptic.2006.1627081
22. Salisbury, K., Tarr, C.: Haptic rendering of surfaces defined by implicit functions. In: ASME International Mechanical Engineering Congress and Exposition. vol. 18244, pp. 61–67. American Society of Mechanical Engineers (1997). https://doi.org/10.1115/IMECE1997-0378
23. Shin, S., Osgouei, R.H., Kim, K.D., Choi, S.: Data-driven modeling of isotropic haptic textures using frequency-decomposed neural networks. In: 2015 IEEE World Haptics Conference (WHC). pp. 131–138. IEEE (2015). https://doi.org/0.1109/WHC.2015.7177703
24. Sianov, A., Harders, M.: Data-driven haptics: Addressing inhomogeneities and computational formulation. In: 2013 World Haptics Conference (WHC). pp. 301–306. IEEE (2013). https://doi.org/10.1109/WHC.2013.6548425
25. Sianov, A., Harders, M.: Exploring feature-based learning for data-driven haptic rendering. IEEE Transactions on Haptics **11**(3), 388–399 (2018). https://doi.org/10.1109/TOH.2018.2817483
26. Sitzmann, V., Martel, J., Bergman, A., Lindell, D., Wetzstein, G.: Implicit neural representations with periodic activation functions. Advances in Neural Information Processing Systems **33**, 7462–7473 (2020), https://proceedings.neurips.cc/paper_files/paper/2020/hash/53c04118df112c13a8c34b38343b9c10-Abstract.html
27. Srinivasan, M.A., Basdogan, C.: Haptics in virtual environments: Taxonomy, research status, and challenges. Computers & Graphics **21**(4), 393–404 (1997). https://doi.org/10.1016/s0097-8493(97)00030-7
28. Venkatesh, R., Sharma, S., Ghosh, A., Jeni, L., Singh, M.: Dude: Deep unsigned distance embeddings for hi-fidelity representation of complex 3d surfaces. arXiv preprint arXiv:2011.02570 (2020). https://doi.org/10.48550/arXiv.2011.02570
29. Zhu, L., Xiang, Y., Song, A.: Visible patches for haptic rendering of point clouds. IEEE Transactions on Haptics **15**(3), 497–507 (2022). https://doi.org/10.1109/TOH.2022.3165119
30. Zilles, C., Salisbury, J.: A constraint-based god-object method for haptic display. In: Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots. vol. 3, pp. 146–151. IEEE (1995). https://doi.org/10.1109/IROS.1995.525876