Implicitly and Differentiably Representing Protein Surfaces and Interfaces

Cory B. Scott[†], Charlie Rothschild, and Benjamin E. Nye
Department of Mathematics and Computer Science, Colorado College,
Colorado Springs, CO 80909, USA
[†]*E-mail: cbs@coloradocollege.edu*
`https://sites.coloradocollege.edu/cbs/`

We introduce a pipeline for implicitly representing a protein, or protein complex, as the union of signed distance functions (SDFs) by representing each atom as a sphere with the appropriate van der Waals radius. While this idea has been used previously as a way to render images of proteins, it has not, to our knowledge, been widely adopted in a machine learning setting. Mirroring recent successful work applying SDFs to represent 3D geometry, we present a proof of concept that this representation of proteins could be useful in several biologically relevant applications. We also propose further experiments that are necessary to validate the proposed approach.

Keywords: Signed Distance Function; Differentiability; Protein Structure; Solvent Accessible Surface

1. Introduction

Modeling the three-dimensional shape of proteins is crucially important for understanding how they interact with other molecules. Advances in representations of protein structure have lead to corresponding advances in the ability for machine learning (ML) methods to predict biologically relevant properties of proteins, such as: binding affinity with bioactive molecules; prediction of protein conformation; and simulation of protein behavior under a variety of external conditions. One common representation of proteins that has received much attention is the *protein surface*,[1] also sometimes called the *solvent accessible surface* (SAS), which is a 2D manifold that represents the portion of the protein's surface that is physically accessible to a solvent. Classical methods[2] for computing the SAS involve rolling an imaginary probe (with the radius of a solvent molecule, typically water) over the surface of the protein and storing the points of contact between the probe and the surface.

In contrast, in this work, we propose an alternate protein surface representation: an isosurface of a signed distance function (SDF), produced by smoothing the boolean union of individual spherical SDFs for each of the protein's component atoms. This approach naturally produces the same biologically relevant notion of a protein's solvent-accessible surface, but represented as an alternative object that naturally integrates with current ML methods and tooling. Historically, changes in the way an object is represented can significantly affect the tractability of modeling it for ML tasks. We present a proof-of-concept demonstration of how this representation can be applied to predict protein-protein interactions; further work is needed to quantify the benefits of the proposed approach. The main contributions of this work are as follows: 1) we discuss prior work training machine learning models on protein surface geometry; 2) we demonstrate a way to represent this surface as the zero-level set of a signed distance function, constructed with boolean operations; 3) we investigate using acceleration structures to query a protein SDF more efficiently; 4) we produce a dataset of protein-protein interface meshes, and provide code to reproduce our results; 5) we illustrate two potential applications to analyzing proteins in motion as well as binding sites.

1.1. *Mathematical and Biological Background*

We first introduce some necessary background in protein biology, geometry, and geometric machine learning. Throughout this paper, variables with an arrow ($\vec{x}$) represent points in nD Euclidean space, lowercase letters represent constants, and $||\cdot||$ is the nD Euclidean norm.

Signed Distance Functions (SDFs) Let $\Omega \subset \mathbb{R}^n$ be a compact subset of Euclidean space, and let $\partial\Omega$ represent its boundary. For any point $\vec{x} \in \mathbb{R}^n$, the signed distance $d_\Omega(\vec{x})$ is defined as:

$$d_\Omega(\vec{x}) = \begin{cases} 0 & x \in \partial\Omega \\ -||\vec{x}-\vec{y}|| & \vec{x} \in \Omega \\ ||\vec{x}-\vec{y}|| & \vec{x} \notin \Omega \end{cases} \quad \text{where} \quad \vec{y} = \arg\min_{\vec{y}\in\partial\Omega} ||\vec{x}-\vec{y}|| \tag{1}$$

In other words, the signed distance measures the distance between any point and the boundary of Ω, with the sign indicating whether a point is inside the shape or not (some authors take the opposite sign as convention, i.e. positive values denote an object's interior). Signed distance functions have many properties that make them useful in a machine learning context: 1) they have unit gradient everywhere the gradient is defined; 2) analytic formulae have been found for a wide variety of SDFs for specfic shapes;[3] and most significantly for this work, 3) simple SDFs can be combined into SDFs for more complex shapes using basic boolean operations. We will specifically make use of the exponential *smooth-min* operation, one of a family of SDF blending operations originally proposed by Quilez.[4] If $d_1(x), d_2(x), \ldots d_n(x)$ are a set of SDFs, then their smooth-min $\mathbf{d}$ is given by

$$\mathbf{d}(\vec{x}) = -k \log\left(\sum_{i=1}^{n} e^{-\frac{1}{k}d_i(\vec{x})}\right) \text{where } k \text{ determines the smoothing radius.} \tag{2}$$

As a relevant example of a specific closed-form analytic SDF, the SDF d for a n-dimensional sphere of radius r centered at $\vec{y}$ is given by $d(\vec{x}) = ||\vec{x}-\vec{y}|| - r$.

When SDFs are combined with boolean operations, the resulting function may not be a true SDF, in the sense that it may not strictly satisfy Equation 1, but in practice the resulting distance fields are approximate enough for typical applications including training ML models.

Protein Surfaces Protein surfaces[5] are a common way to represent the parts of a protein that are available to react/bind with other proteins or small molecules. A protein surface is typically calculated by representing each atom as a hard-shell sphere of a given radius, and rolling a simulated spherical probe (typically taken to have the radius of a single hydrogen atom, 1Å) over the collection of atoms. The final mesh representing the protein is produced by discretizing the surface traced by the probe. If we trace the center of the probe sphere, we get the Solvent Accessible Surface (SAS) of the protein. Tracing instead all the contact points between the probe and the atoms of our molecule produces the Solvent Excluded Surface. See Figure 1 for examples of protein solvent excluded surfaces.

1.2. *ML Analysis of Protein Surfaces*

Protein design and analysis is a rapidly evolving application area of machine learning.[6,7] For a more thorough review of ML for protein design and concepts in geometric machine learning, we refer the reader to Cheng et al.[8] and Bronstein et. al[9] respectively. Protein surfaces have been widely adopted as a representation of proteins that facilitates training machine learning models to identify possible protein-protein and protein-ligand interactions.[10–15] While much of the prior work on protein surfaces has focused on meshes as an intermediate representation, there is a complimentary vein of work that uses signed distance functions to represent proteins. SDFs have been thoroughly examined in the context of rendering 2D and 3D images of proteins.[16–18] Much of this prior work uses machine learning or an SDF representation of proteins, but not both. A notable exception is Sverrisson et al.[19] which uses a similar atomic union SDF to the one we propose. However, that work mainly examines the definition of a convolution-like operation on implicitly defined protein surfaces, and does not consider combinations of such surfaces with constructive solid geometry operations as we do in the present work.

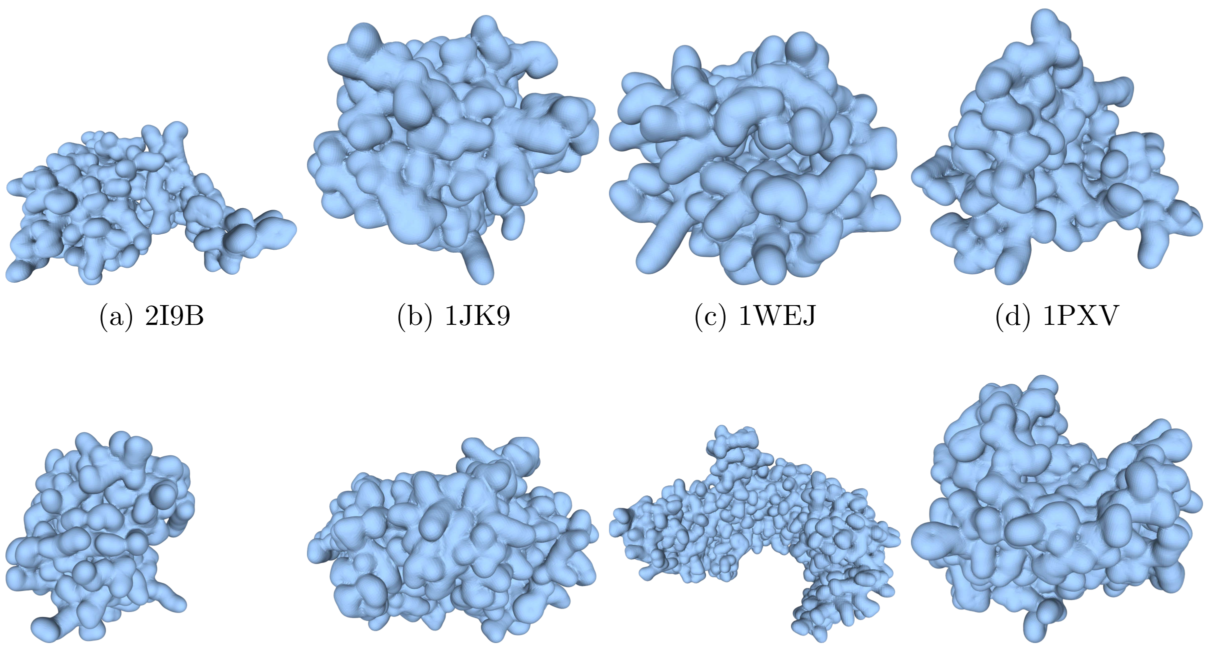
2. Method

2.1. *Protein Representation*

We propose representing a protein as the union of spherical SDFs of each of its component atoms, where each atom is a sphere sized according to its van der Waals radius (as reported by Bondi[20]). These spherical SDFs are combined using the smooth-min operation, which we and others have observed in practice[19,21] to closely resemble SAS and SES computed via other means. See Figure 1 for several examples of protein SDFs computed according to this method. We implement these SDFs in Pytorch,[22] enabling backpropagation of error through loss functions composed of protein SDF queries, and provide the code for operationalizing ML tasks using SDF protein surface representations.

2.1.1. *Possible Issues*

As noted above, boolean operations on SDFs are not guaranteed to produce a function that satisfies Equation 1. The effect of this on our proposed protein SDF can be seen in Figure 2. The overlapping SDFs in the protein interior produce a lower bound instead of the true SDF value. However, for the union operation, the resulting SDF has the correct $d = 0$ isosurface and exterior distance values. We expect that this limitation should have little impact on downstream applications, since protein-protein and protein-ligand interactions occur exclusively on the exterior.


(a) 2I9B (b) 1JK9 (c) 1WEJ (d) 1PXV

(e) 7CEI (f) 2GAF (g) 1IBR (h) 1A2K

Fig. 1: $d = 0$-level set meshes of multiple proteins from the DB5 dataset. Each protein imaged here is the "left" chain of the protein complex, in its bound confirmation. See Table 1 for a detailed explanation of these proteins.

2.2. *Intersection of Protein Chains*

In this section, we use properties of SDFs to produce an implicit function that represents the shared interface between two protein chains. In addition to the smooth min defined in Equation 2, we make use of three operations on SDFs: 1) the rounding operation $d^{(r)}(\vec{x}) = d(\vec{x}) - r$, which expands the zero level set of an SDF by r in the positive direction; 2) the intersection operation, where the intersection of two SDFs d_1 and d_2 is given by $d_{d_1 \cap d_2}(\vec{x}) = \max(d_1(\vec{x}), d_2(\vec{x}))$, and finally 3) the union operation, where the union of two SDFs d_1 and d_2 is given by $d_{d_1 \cup d_2}(\vec{x}) = \min(d_1(\vec{x}), d_2(\vec{x}))$. Let d_1 and d_2 be the SDFs for two protein chains. Then we calculate an interface SDF as:

$$\begin{aligned} d_{\text{INTER}} = \min\big(& \max\left(d_1\left(\vec{x}\right) - r, d_2\left(\vec{x}\right)\right), \\ & \max\left(d_1\left(\vec{x}\right), d_2\left(\vec{x}\right) - r\right)\big) \end{aligned} \tag{3}$$

This produces an SDF that represents the area that is within r of both protein surfaces, i.e. the space between the two chains. See Figures 3 and 5 for examples of these interface meshes with an intersection radius of $r = 4$Å.

2.3. *Accelerating Queries*

We investigate the use of two data structures for accelerating spatial queries of protein SDFs: Bounding Volume Hierarchies (BVHs) and K-D Trees. For further details about these data

Table 1: PDB ID, atom count, and short description for each of the proteins used in figures in this paper. For more information, we refer the reader to the individual PDB entries, or to the description pages for the DB5[23] or PLINDER[24] datasets.

Source	PDB ID	# Atoms	Description
DB5	2I9B	11696	ATF-urokinase receptor complex
DB5	1JK9	5954	Heterodimer between H48F-ySOD1 and yCCS
DB5	1WEJ	4166	IGG1 fragment with horse cytochrome
DB5	1PXV	4816	Staphostatin-staphopain complex
DB5	7CEI	1724	Colicin E7 in complex with inhibitor
DB5	2GAF	6011	Vaccinia polyadenylate polymerase
DB5	1IBR	9739	RAN with Beta
DB5	1A2k	6732	NTF2:RanGDP complex
DB5	1WDW	29252	Tryptophan synthase a2b2 complex
PLINDER	1GOL	2901	Rat MAP kinaser ERK2
PLINDER	2CJM	8856	CDK2 Y15p T160p in complex with cyclin
PLINDER	2PY7	4133	E. coli phosphoenolpyruvate carboxykinase
PLINDER	3DAW	4056	Actin-depolymerizing factor
PLINDER	4EOM	8710	Human cyclin A3 complex with ATP
PLINDER	4EOQ	8717	Human cyclin A3 complex with ATP
PLINDER	6AEC	2579	Human DNA Polymerase Mu with MnATP
PLINDER	8AEC	2748	Compound 17 bound to CK2alpha
Fan et al.[25]	2QF1	4410	Zinc transporter YiiP

structures, we refer the reader to several surveys on data structures.[26–28] BVHs in particular have been previously explored as a technique to accelerate SDF queries.[29] Both of these data structures use spatial subdivision to accelerate finding the k nearest points to a given query point. See Figure 4 for an illustration of the BVH approach. Both of these data structures yield an approximation (see Figure 6) of the original signed distance function, but with a substantially lower memory footprint, which we analyze in Section 3.4. We also note here that the object-oriented implementation of a protein as a Torch module may add overhead in comparison to computing the all-pairs distance between a query point and all of a protein's atoms. A further potential benefit of the object-oriented approach, which we hope to explore in future work, is the ability to represent protein subunits like residues and secondary structure.

2.4. *Implementation Details*

Code used to build the SDFs described in this paper is available at `https://github.com/cory-b-scott/protein_sdfs`. Our code is built on top of the `torch_sdf` package, which is available at `https://github.com/cory-b-scott/torch_sdf`. For visualizing SDFs, we first convert each to a mesh using the Marching Cubes algorithm[30] as implemented in SciKit-Image.[31] We render images of each mesh using the `meshplot` package. The final meshes for each interface in the DB5 dataset may be downloaded from `https://osf.io/xzgyc/?view_only=6634b1aa0bf6401990ac2f217cef068b`.

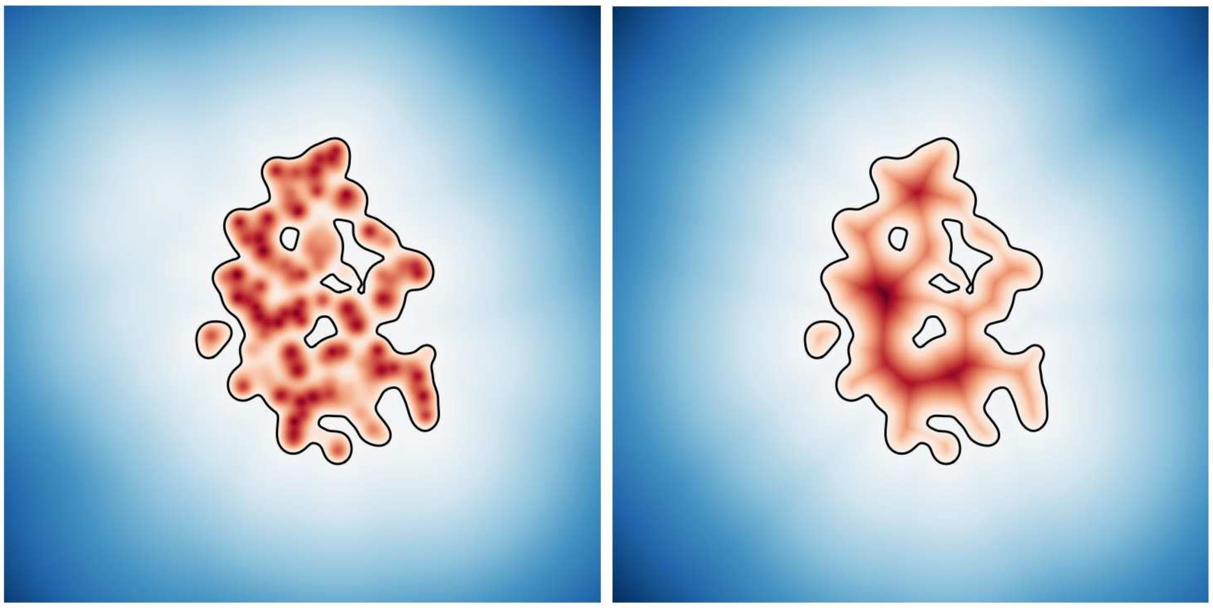


Fig. 2: An illustration of one possible issue when representing a protein as a collection of spherical SDFs. The left shows the union SDF, while the right shows the true distance to the $d = 0$ isosurface. The union SDF is incorrect in the protein's interior.

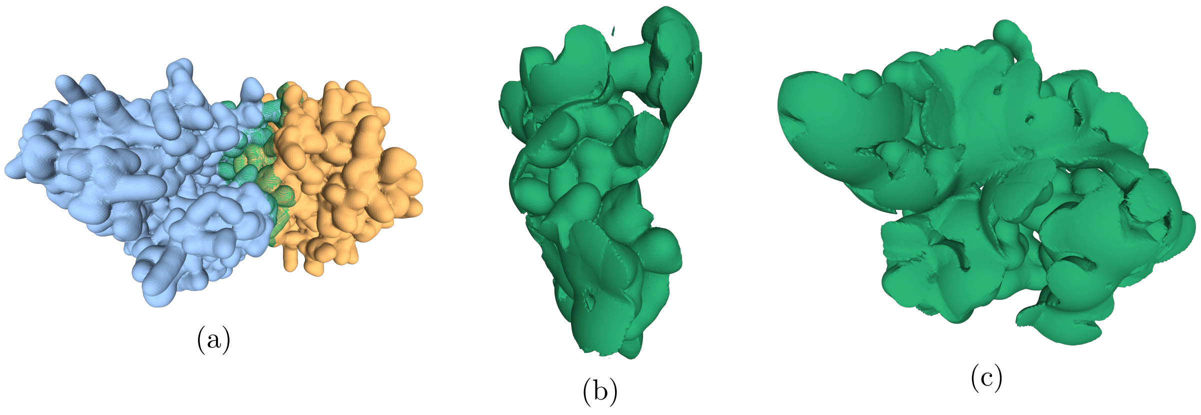


Fig. 3: An example of the $d(\mathbf{x}) = 0$ isosurface generated by our method for the protein 1WDW from the DB5 dataset. Left: The intersection isosurface rendered alongside both chains of the protein complex (color denotes chain ID). Nodes in the intersection region have been highlighted in green. Right: two views of the isolated intersection mesh.

3. Results and Applications

For this paper, we evaluate our model on the Docking Benchmark 5 (DB5)[23] and PLINDER[24] datasets. DB5 consists of protein complexes, where each complex includes two chains. Each chain is represented with PDB files[32] with atom coordinates for both its undocked pose (i.e.

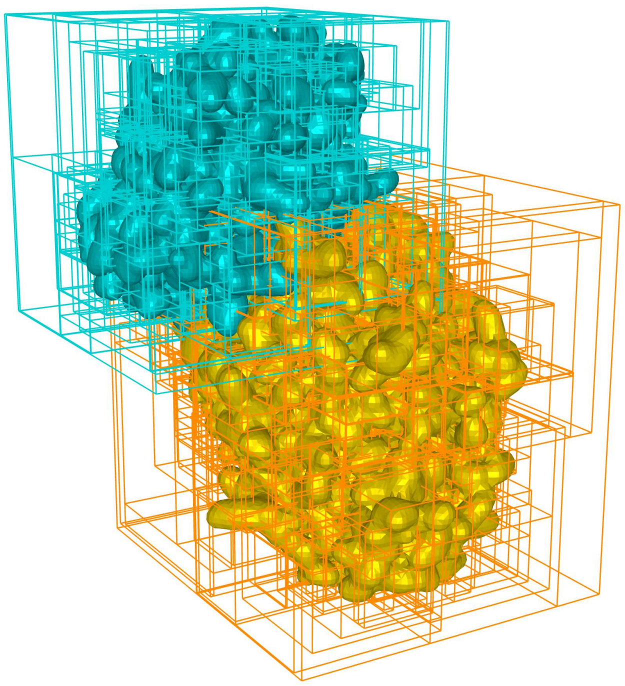


Fig. 4: A visualization of the bounding volume hierarchy built by our method for the protein 1WDW.

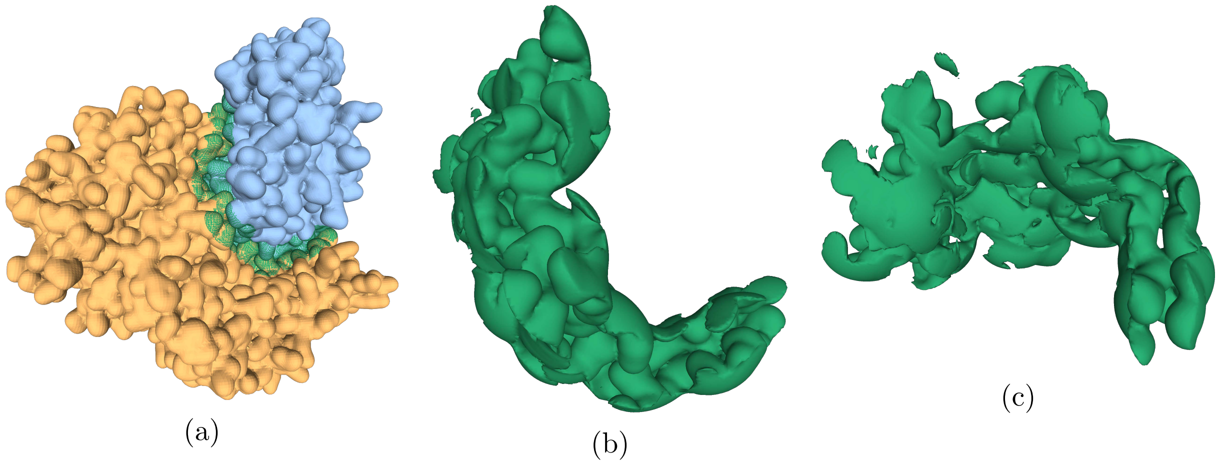
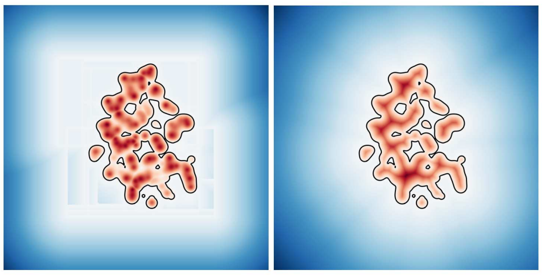


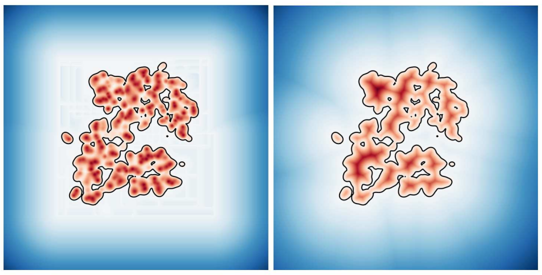
Fig. 5: As in Figure 3, but for protein 2GAF.

Table 2: Comparison of time/memory usage by two acceleration structures.

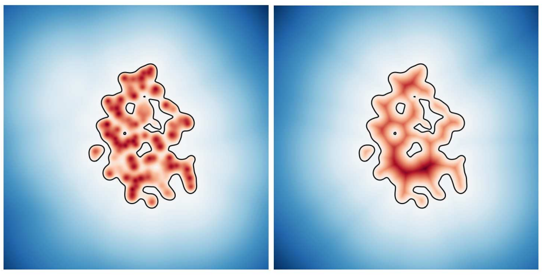
	Avg time (ms)	Peak memory
Basic	48.01	1.91MB
KDTree	77.43	11.15 kB
BVH	45.77	7.06 kB



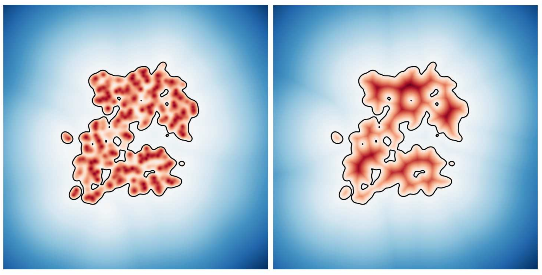
(a) 1A2K, BVH rendering.



(b) 1WDW, BVH rendering.



(c) 1A2K, kd-tree rendering.



(d) 1WDW, kd-tree rendering.

Fig. 6: Illustration of SDFs produced by the two acceleration structures we examine in this paper. Each picture shows a 2D slice through the signed distance field at $z = 0$. We see that both accelerated SDFs are only approximations of the SDFs in Figure 2. In particular, the BVH approach leads to rectilinear artifacts on the exterior of the SDF; this could likely be ameliorated by using a different shape as the bounding volume (as opposed to axis-aligned rectangular prisms).

the pose it folds into naturally) and its docked pose with the other protein in the complex. We use the "bound" version of each chain in the complex. PLINDER is a searchable catalogue of protein-ligand complexes which can be filtered by protein/ligand/complex properties. We use the holo conformation of several proteins bound to ATP for the SDFs we render in Figure 8.

3.1. *Calculation of Protein Interaction Surfaces Using Smooth-Min SDFs*

We use the SDF defined in Equation 3 to compute signed distance functions for all of the protein complexes in the DB5 dataset, as well as SDFs and meshes for their interfaces. We use an interaction radius of 4.0Å in line with previous work on protein-protein interaction.[33,34] This data supports ML tasks using SDF representations, as well as explicit modeling of the biologically active surfaces. See Figures 3 and 5 for example protein interface meshes.

3.2. *Visualizing Molecular Dynamics Trajectories*

In this section, we demonstrate using protein SDFs to visualize a protein's shape over time during a molecular dynamics (MD) simulation. We generate an SDF for an entire protein trajectory as follows: first, we generate an SDF $f_t(\vec{x})$ for the protein at each timestep using the approach introduced in Section 2.1. Then we take the smooth-min of all of these SDFs over t: $F(\vec{x}) = \mathrm{smin}_t f_t(\vec{x})$. This produces an outer envelope showing the shape of the protein over the course of the entire simulation. Visualizing a protein in this way can tell us which structural features of the protein are persistent over time. To illustrate this technique we use a 9ns trajectory of the YiiP membrane protein[25] (See Figure 7). This illustrates that this SDF representation extends naturally to time-series protein data.

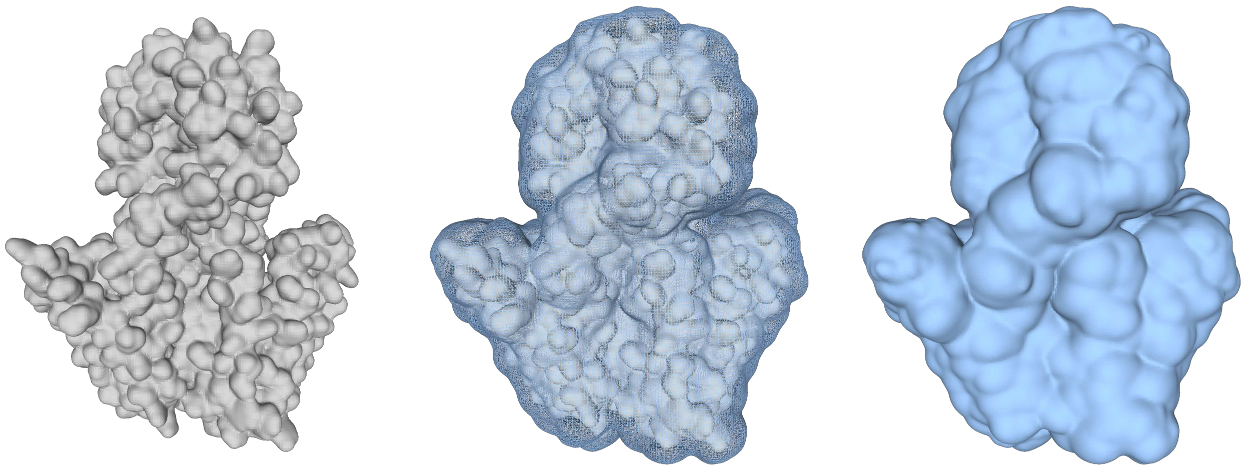


Fig. 7: An SDF computed over all timesteps of a MD simulation of the YiiP membrane protein in water. Left: the protein SDF for the first timestep only. Right: the SDF computed by taking the smooth-min blend of all timesteps. Center: left and right views overlaid.

3.3. *Visualizing Ligand Pockets*

We use the method described in Section 3 to generate visualizations of ligands and their associated binding sites. We download a subset of arbitrarily chosen protein-ligand complexes from the PLINDER training dataset,[24] filtering for complexes which consist of a single protein bound to ATP. We extract an SDF representing the binding site by building the smooth-min SDF for the protein, and then taking the intersection of that SDF with that of a 10Åsphere centered on the ATP molecule. We visualize these binding site SDFs in Figure 8, both with and without the ATP molecule visualized. Because each of these SDFs is constructed in a fully differentiable way, we could train a neural net to memorize and reproduce these pocket shapes, in the manner of a DeepSDF[35] model.

3.4. *Comparison of Acceleration Structures*

We compare the performance of the two acceleration structures described in Section 2.3. For each protein complex in the DB5 dataset, we benchmark the time and GPU memory needed to evaluate all grid points in a 256^3 grid laid over the bounding box of the complex. In Figure 9, we compare the accelerated version of each protein structure to the time and memory required to query all of the atomic SDFs for a given complex. We see that the BVH

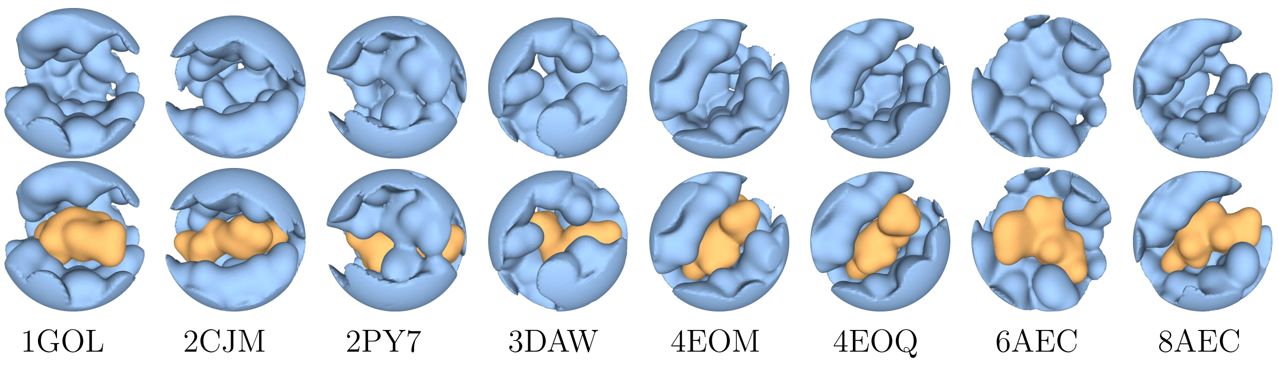


Fig. 8: Visualization of binding pockets for ATP for several proteins. Top row shows the pocket only; bottom row shows the pocket occupied by the ATP molecule (orange). See Table 1 for a description of each protein.

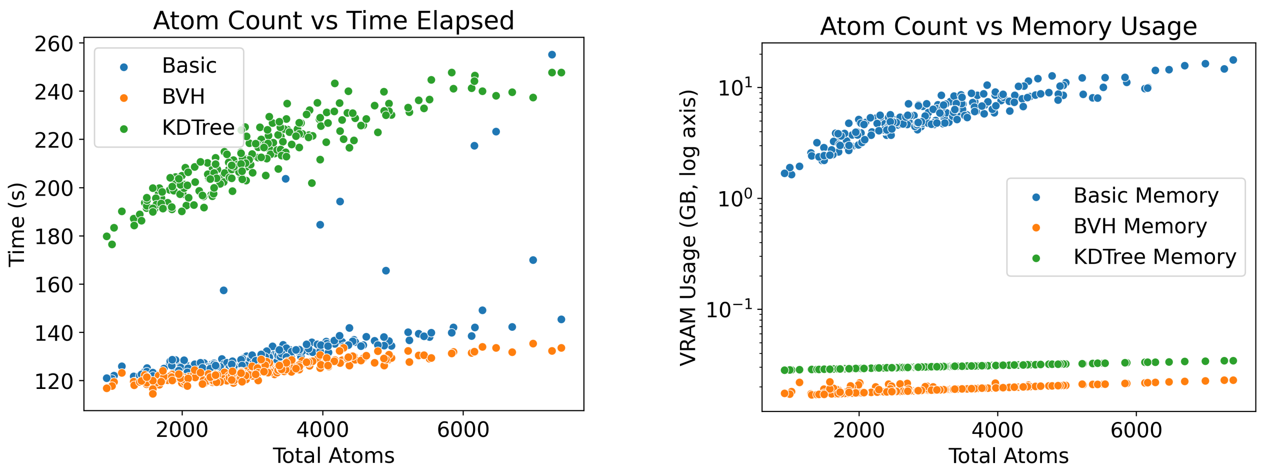


Fig. 9: Time/memory benchmarking of acceleration structures for querying protein SDFs.

approach is slightly faster than the naive approach, but the KDTree is much slower, likely due to CPU-GPU latency (at time of writing, there was no functional Torch+CUDA KDTree implementation, so the KDTree approach necessitates copying data back to the CPU. We hope to ameliorate this inefficiency in future work). All timings were collected on a consumer NVIDIA RTX A6000 GPU. Both the KDTree and BVH approaches offer significant memory savings (see Table 2). However, this approximation does incur some rectangular artifacts in the BVH tree approach; this could likely be resolved by using a different shape primitive in the volume hierarchy (e.g. bounding spheres instead of axis-aligned rectangles).

4. Conclusion and Future Work

This paper demonstrates the utility of representing a protein using the signed distance to its solvent-accessible surface (as approximated by the smooth-min function to the protein's component atoms). We also show how this can facilitate constructive solid geometry operations on protein interfaces. While this technique is promising, further work is needed to validate the proposed approach. One of the major advantages of building our protein representation in PyTorch[22] is the ability to optimize over parts of the protein representation. In future work, we hope to use our framework to optimize protein shape, conformation, and position according to SDF-based loss functions.

Recent work[35] has used neural networks as maps between vectors of latent states and signed distance functions, producing a single model that is able to predict SDF values for a dataset of multiple 3D shapes. We hope to investigate a combination of this approach and our SDF construction, perhaps by utilizing embedding vectors from a pretrained protein transformer architecture.[36] We also hope to incorporate atomic characteristics as features of the generated surfaces. Finally, the literature includes examples of data structures like the ChainTree[37] which have been specifically developed for fast querying of energy potentials of proteins. It is likely that our proposed approach could be further enhanced by one of these protein-specific acceleration structures.

Finally, we note that constructing an SDF that implicitly stores the SAS shape is only the first step. We anticipate that this approach could be useful in any case where a machine learning model operates on the solvent accessible surface (AKA, where differentiability is needed). An example could include optimizing the position and pose of amino acids to produce an SAS that fits some other structural motif; or to provide some other machine learning model with a training signal that takes protein shape into account. We hope to use our SAS representation in one or more of these applications.

Acknowledgements

This work was supported in part by the Euclid Fund of the Department of Mathematics and Computer Science at Colorado College, and in part by the Dean of the Faculty at Colorado College's Mrachek Research Grant.

References

1. M. L. Connolly, Solvent-accessible surfaces of proteins and nucleic acids, *Science* **221**, 709 (1983).
2. M. L. Connolly, Analytical molecular surface calculation, *Applied Crystallography* **16**, 548 (1983).
3. I. Quilez, Modeling with distance functions. `http://iquilezles.org/www/articles/distfunctions/distfunctions.htm`, (2008).
4. I. Quilez, Smooth minimum. `https://iquilezles.org/articles/smin/`, (2013).
5. A. Shrake and J. A. Rupley, Environment and exposure to solvent of protein atoms. lysozyme and insulin, *Journal of Molecular Biology* **79**, 351 (1973).
6. R. Casadio, P. L. Martelli and C. Savojardo, Machine learning solutions for predicting protein–protein interactions, *Wiley Interdisciplinary Reviews: Computational Molecular Science* **12**, p. e1618 (2022).
7. P. Notin, N. Rollins, Y. Gal, C. Sander and D. Marks, Machine learning for functional protein design, *Nature biotechnology* **42**, 216 (2024).
8. J. Cheng, A. N. Tegge and P. Baldi, Machine learning methods for protein structure prediction, *IEEE reviews in biomedical engineering* **1**, 41 (2008).
9. M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst, Geometric deep learning: going beyond Euclidean data, *IEEE Signal Processing Magazine* **34**, 18 (2017).
10. A. J. Bordner and A. A. Gorin, Protein docking using surface matching and supervised machine learning, *Proteins: Structure, Function, and Bioinformatics* **68**, 488 (2007).
11. P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. Bronstein and B. Correia, Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning, *Nature Methods* **17**, 184 (2020).
12. K. Atz, F. Grisoni and G. Schneider, Geometric deep learning on molecular representations, *Nature Machine Intelligence* **3**, 1023 (2021).
13. O. Méndez-Lucio, M. Ahmad, E. A. del Rio-Chanona and J. K. Wegner, A geometric deep learning approach to predict binding conformations of bioactive molecules, *Nature Machine Intelligence* **3**, 1033 (2021).
14. S. K. Mylonas, A. Axenopoulos and P. Daras, DeepSurf: a surface-based deep learning approach for the prediction of ligand binding sites on proteins, *Bioinformatics* **37**, 1681 (2021).
15. V. Mallet, Y. Miao, S. Attaiki, B. Correia and M. Ovsjanikov, AtomSurf: Surface representation for learning on protein structures, in *The Thirteenth International Conference on Learning Representations*,
16. J. Parulek and I. Viola, Implicit representation of molecular surfaces, **1**, 217 (2012).
17. J. Parulek and A. Brambilla, Fast blending scheme for molecular surface representation, *IEEE Transactions on Visualization and Computer Graphics* **19**, 2653 (2013).
18. D. Klepáč, Rendering molecular surfaces using signed distance functions [online] (2023 [cit. 2024-09-16]), SUPERVISOR : Jan Byška.
19. F. Sverrisson, J. Feydy, B. E. Correia and M. M. Bronstein, Fast end-to-end learning on protein surfaces, 15272 (2021).
20. A. v. Bondi, van der Waals volumes and radii, *The Journal of Physical Chemistry* **68**, 441 (1964).
21. G. Patané and M. Spagnuolo, State-of-the-art and perspectives of geometric and implicit modeling for molecular surfaces, *Computational Electrostatics for Biological Applications: Geometric and Numerical Approaches to the Description of Electrostatic Interaction Between Macromolecules* , 157 (2015).
22. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga and A. Lerer, Automatic differentiation in PyTorch (2017).
23. T. Vreven, I. H. Moal, A. Vangone, B. G. Pierce, P. L. Kastritis, M. Torchala, R. Chaleil, B. Jiménez-García, P. A. Bates, J. Fernandez-Recio *et al.*, Updates to the integrated protein–

protein interaction benchmarks: docking benchmark version 5 and affinity benchmark version 2, *Journal of Molecular Biology* **427**, 3031 (2015).
24. J. Durairaj, Y. Adeshina, Z. Cao, X. Zhang, V. Oleinikovas, T. Duignan, Z. McClure, X. Robin, G. Studer, D. Kovtun *et al.*, PLINDER: The protein-ligand interactions dataset and evaluation resource, *bioRxiv* , 2024 (2024).
25. S. Fan and O. Beckstein, Molecular Dynamics trajectories of membrane protein YiiP (5 2019).
26. E. Reinhard, B. Smits and C. Hansen, Dynamic acceleration structures for interactive ray tracing, 299 (2000).
27. T. Foley and J. Sugerman, Kd-tree acceleration structures for a GPU raytracer, 15 (2005).
28. D. Meister, S. Ogaki, C. Benthin, M. J. Doyle, M. Guthe and J. Bittner, A survey on bounding volume hierarchies for ray tracing, **40**, 683 (2021).
29. F. Liu and Y. J. Kim, Exact and adaptive signed distance fieldscomputation for rigid and deformable models on gpus, *IEEE transactions on visualization and computer graphics* **20**, 714 (2013).
30. T. Lewiner, H. Lopes, A. W. Vieira and G. Tavares, Efficient implementation of marching cubes' cases with topological guarantees, *Journal of graphics tools* **8**, 1 (2003).
31. S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart and T. Yu, scikit-image: image processing in Python, *PeerJ* **2**, p. e453 (2014).
32. H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov and P. E. Bourne, The protein data bank, *Nucleic acids research* **28**, 235 (2000).
33. J. Salamanca Viloria, M. F. Allega, M. Lambrughi and E. Papaleo, An optimal distance cutoff for contact-based protein structure networks using side-chain centers of mass, *Scientific reports* **7**, p. 2838 (2017).
34. R. A. Laskowski, J. Jabłońska, L. Pravda, R. S. Vařeková and J. M. Thornton, PDBsum: structural summaries of PDB entries, *Protein science* **27**, 129 (2018).
35. J. J. Park, P. Florence, J. Straub, R. Newcombe and S. Lovegrove, DeepSDF: Learning continuous signed distance functions for shape representation, 165 (2019).
36. C. Dallago, K. Schütze, M. Heinzinger, T. Olenyi, M. Littmann, A. X. Lu, K. K. Yang, S. Min, S. Yoon, J. T. Morton and B. Rost, Learned embeddings from deep learning to visualize and predict protein sets, *Current Protocols* **1**, p. e113 (2021).
37. I. Lotan, F. Schwarzer and J.-C. Latombe, Efficient energy computation for monte carlo simulation of proteins, 354 (2003).