ALPHASURF: ON-THE-FLY SURFACE COMPUTATIONS FOR PROTEIN REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Several protein surfaces have been proposed, notably for visualization purposes. Machine learning methods have incorporated these surfaces as protein representations, resulting in strong performance, at the cost of heavy computations. In this paper, we show that this burden can be avoided by introducing AlphaSurf, a coarse meshing method tailored for learning on proteins that leverages α-complexes. We characterize the surfaces theoretically, including their topological relation to the SAS surface, and evaluate them experimentally on downstream tasks. Our method runs on-the-fly during training while maintaining performance. This enables use cases previously out of reach for surface-based methods: large-scale training as well as coordinate-noise data augmentation and training on molecular dynamics frames which improves performance. The code is available here `https://anonymous.4open.science/r/AlphaSurf-C5A1/README.md`

1 INTRODUCTION

1.1 MOLECULAR SURFACES

Various protein surfaces have been developed in the last four decades, following their initial introduction in Connolly (1983). They represent a foundational tool for visualization (DeLano et al., 2002) and geometry processing (Gerstein & Richards, 2001). The use of protein surfaces within a learning pipeline was pioneered by MaSIF (Gainza et al., 2020) and followed by numerous methods achieving state-of-the-art performance.

Modeling atoms as intersecting spheres of different radii, the most natural molecular surface is the **outer boundary of the union of balls** (Gavezzotti, 2007). This boundary consists of spherical caps around atoms, and their intersections: circle arcs between two spheres and points between three. This set of primitives is termed the *combinatorial structure* of the boundary of the union.

This outer boundary defines the van der Waals surface when using van der Waals radii, which are obtained from half the pairwise distance between atoms in organic molecules (Gavezzotti, 2007). Given the importance of water solvent (Lee & Richards, 1971), the **Solvent Accessible Surface (SAS)** is defined as the boundary of the atomic balls, enlarged by the radius of a water probe (Gerstein et al., 1995; Gerstein & Richards, 2001), taken as $r_w = 1.4$Å in general.

Another classical surface is obtained by rolling a water probe on the atoms (Connolly, 1983). Known as Connolly's surface, molecular surface or **Solvent-Excluded Surface (SES)**, it consists of spherical caps (resp. toroidal and spherical concave) patches contributed by the probe in contact with one (resp. two and three) atoms. This surface has the advantage of being smooth: a tangent plane exists everywhere.

Finally, **triangulated molecular surfaces** are often used for visualization purposes. Such meshes are delivered, for example, by `MSMS` (Sanner et al., 1995) and widely used in tools such as PyMOL (DeLano et al., 2002).

1.2 SURFACE COMPUTATION

Two main strategies exist to compute the combinatorial structure of the boundary of the union of balls. The first one leverages a generic mathematical object called the **α-complex** $\mathcal{K}_\alpha$ (Akkiraju & Edelsbrunner, 1996), described in detail in Section 2.1. The complexity of computing the α-complex is $O(n \log n + k)$, with k the number of simplices, empirically equal to $O(n \log n)$ for

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

proteins (Cazals et al., 2011). A numerically robust implementation is provided by the Computational Geometry Algorithm Library (The CGAL Project).

The second strategy for computing molecular surfaces leverages atom properties. Atoms have comparable radii and their distances are lower bounded due to repulsion forces. For such a set, the number of edges and faces of the molecular surface is linear in the number of spheres and can be found in $O(n \log n)$ (Halperin & Overmars, 1994). This property is exploited by the `MSMS` program (Sanner et al., 1995) using an iterative algorithm to compute the solvent-excluded surface.

Once equipped with the boundary of the union of balls, several methods exist to obtain a triangular mesh of the surface. `MSMS` discretizes circle arcs into segments, computes constrained meshes with those segments on each patch and stitches the results into a single mesh. In contrast, EDTSurf (Xu & Zhang, 2009) and NanoShaper (Decherchi & Rocchia, 2013) project the level-set specification of the target SAS or SES onto a regular grid. To do so, EDTSurf uses the Euclidean distance transform while NanoShaper relies on ray casting. Both methods then extract triangular meshes using the marching cubes algorithm, so the grid resolution introduces a geometry-cost trade-off that can affect the resulting shape and mesh complexity.

Protein triangular meshes have been pioneered for protein representation learning by `MaSIF` (Gainza et al., 2020), incurring high computational cost. To limit this burden, point clouds sampled at the surface with approximate normals have also been proposed as a surface proxy (dMaSIF (Sverrisson et al., 2021)), as well as coarsened triangular meshes in AtomSurf (Mallet et al., 2024). In this paper, we propose to directly extract a coarse mesh from the α-complex for learning applications. The different molecular surfaces, their computation, and their use in machine learning pipelines are illustrated in Figure 1.

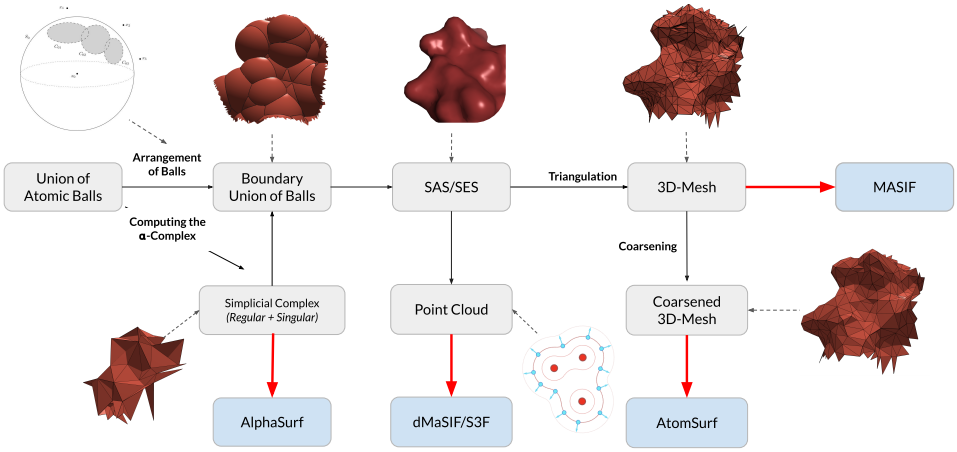


Figure 1: Comparison of the surface-processing pipelines used by MaSIF, AtomSurf, and AlphaSurf. MaSIF operates on a triangulated 3D mesh, AtomSurf uses a coarsened version of this mesh, whereas AlphaSurf extracts an atom-centered simplicial complex directly from the weighted α-complex. Blue boxes indicate the final representations used by each method.

1.3 Machine Learning on Protein Surfaces

Learning on surfaces has emerged as a subfield of geometric deep learning with computer vision applications (Bruna et al., 2013; Masci et al., 2015; Bronstein et al., 2017; Sharp et al., 2022). `MaSIF` (Gainza et al., 2020) pioneered the application of such methods to molecules, extracting patches from `MSMS` triangulations. It was followed by several others directly using the whole triangulation along with coarsening (Mylonas et al., 2021; Riahi et al., 2023; Wang et al., 2023; Banerjee et al., 2025). Other approaches relied on points sampled at the surface (Zhang et al., 2024; Wu & Li, 2024), following the sampling method of Sverrisson et al. (2021).

Using surface meshes in isolation disregards important protein features and often results in underwhelming performance (Mallet et al., 2024). However, combining surface encoders with other modalities, such as geometric graphs, results in powerful embeddings (Somnath et al., 2021; Mallet et al., 2024; Zhang et al., 2024). Surface-based models have been successfully used for protein

design (Li et al., 2025; Wu et al., 2026a) and protein-protein docking (Sverrisson et al., 2023), as well as to predict solubility (Qian et al., 2025), driver mutations (Zhou et al., 2025), binding site prediction (Wu et al., 2026b) and small-molecule binding (Cao et al., 2025).

All the aforementioned surface methods rely on triangulations of the surface. However, once the combinatorial structure has been computed, producing a triangle mesh is a costly extra step resulting in large meshes. We believe the numerous small triangles present on the surface are an unnecessary proxy for the semantic information of the surface, incurring avoidable overheads.

We argue this limitation has no foundational basis and stems only from the widespread adoption of MSMS as the default tool to build surfaces. We hence propose AlphaSurf, a pipeline that directly extracts protein meshes for learning applications from the α-complex.

Our contributions are as follows.

- We propose AlphaSurf, a meshing technique that drastically cuts preprocessing time.
- We validate that our meshes sustain state-of-the-art performance on known applications.
- We showcase novel use cases for protein surface encoders: learning at scale, learning on molecular dynamics and data augmentation on-the-fly.

2 AlphaSurf: Rapid protein surface meshing using the α-complex

2.1 Mathematical background on α-complexes

A k-simplex is defined as the convex hull of $k+1$ affinely independent points. In $\mathbb{R}^3$, a 0-simplex is a point, a 1-simplex a segment, a 2-simplex a triangle, and a 3-simplex a tetrahedron. The face of a simplex is a simplex of smaller dimension defined by a strict subset of its vertices, and one defines a coface similarly. A simplicial complex is a collection of simplices such that the intersection of any two simplices is either empty or a face of each.

For a set of balls $\{B_i(x_i, r_i)\}$, the power distance $\pi(x, B_i) = \|x - x_i\|^2 - r_i^2$ induces a *power diagram* with cells $V_i = \{x \mid \pi(x, B_i) \leq \pi(x, B_j), \forall j\}$, generalizing the Euclidean Voronoi diagram. Intersections between these regions define the so-called *regular triangulation* $\mathcal{K}_\infty$, a simplicial complex generalizing the usual Delaunay triangulation for balls. Indeed, two, three and four incident power regions define an edge, a triangle and a tetrahedron, respectively, in the regular triangulation. Remarkably, both diagrams are invariant upon considering the set of enlarged balls $\mathcal{S}_\alpha = \{B_i(x_i, \sqrt{r_i^2 + \alpha})\}_{i=1,\dots,n}$ for $\alpha \geq 0$. This invariance enables defining the *restriction of a ball* as $R_{i,\alpha} = V_i \cap B_{i,\alpha}$. The **$\alpha$-complex**, denoted $\mathcal{K}_\alpha$, is the subcomplex of $\mathcal{K}_\infty$ that contains all simplices $\Delta \in \mathcal{K}_\infty$ such that $\bigcap_{j \text{ vertex of } \Delta} R_{j,\alpha} \neq \emptyset$. Note that growing α results in larger restrictions, hence more simplices. For α large enough, $\mathcal{K}_\alpha$ therefore matches the regular triangulation.

Geometrically, the union of simplices in $\mathcal{K}_\alpha$ forms an underlying polyhedron known as the α-shape $|\mathcal{K}_\alpha| \subset \mathbb{R}^d$. In $\mathbb{R}^3$, all tetrahedra in $\mathcal{K}_\alpha$ are classified as *interior*. For a lower-dimensional simplex $\Delta \in \mathcal{K}_\alpha$, Δ is interior if all its cofaces in $\mathcal{K}_\infty$ belong to $\mathcal{K}_\alpha$ (no coface removed around it) and Δ is not on the convex hull boundary. Otherwise, Δ lies on the *topological boundary* $\partial|\mathcal{K}_\alpha|$, which is itself a complex. There, it is classified as *regular* if it has at least one coface in $\mathcal{K}_\alpha$, and *singular* otherwise. These concepts are illustrated in the 2D case in Figure 2.

2.2 Molecular surfaces and α-complexes

On proteins, α-complexes determine atomic packing (Gerstein & Richards, 2001) and interface (Janin et al., 2008; Loriot & Cazals, 2010) properties. They can be used to compute molecular surfaces easily. For example, the SAS surface reads directly from $\partial|\mathcal{K}_0|$ for the set of balls $\mathcal{S} = \{B_i(x_i, r_i + r_w)\}_{i=1,\dots,n}$. For a more detailed discussion of the link between the α-complex and surfaces, see Appendix A.

Current learning methods rely on fine triangulations of the SAS or the SES, because existing machine learning methods operate on triangular meshes. AlphaSurf promotes a different viewpoint by performing the diffusion on a coarse simplicial complex. Since the SAS is directly encoded in the boundary complex $\partial|\mathcal{K}_0|$, which consists of triangles and edges for proteins, we aim to turn this complex into a triangular mesh amenable to learning. Note that the vertices of this complex correspond to atoms, which constitutes an offset compared to SAS or SES vertices. In addition, $\partial|\mathcal{K}_\alpha|$ is not, in general, a 2-manifold.

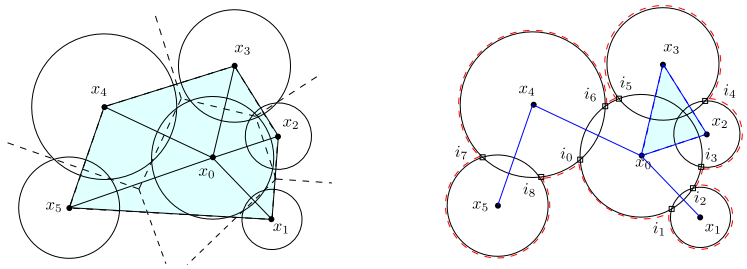


Figure 2: 2D illustration of the α-complex and the boundary of a union of balls. (Left) Regular triangulation of six balls (vertices, edges, cyan triangles), and the corresponding power diagram. (Right) Corresponding complex $\mathcal{K}_0$ (vertices, blue edges, cyan triangle), and boundary of the union of balls (red circle arcs). The ball centered at x_0 contributes three circle arcs–vertex x_0 is a regular 0-simplex in $\mathcal{K}_0$. The singular 1-simplex (x_4, x_5) contributes the two intersection points i_7 and i_8 on the boundary.

However, we establish that a 2-manifold can be obtained from a **proper offset** $\mathcal{T}_r$ of $\mathcal{K}_\alpha$, and that this offset can be continuously deformed (without tearing or self-intersecting) into the SAS. Here, a proper r-offset denotes the regularized boundary of the Minkowski sum $\partial(|\mathcal{K}_\alpha| \oplus \bar{B}_r(0))$. The precise statement reads as follows:

Theorem. 1. *There exists $r_0 > 0$ such that for any $r \in]0, r_0]$, a proper r-offset $\mathcal{T}_r$ of the alpha shape $|\mathcal{K}_0|$ is isotopic to the boundary of the union of balls $\partial \cup_i S_i$.*

The proof, provided in Appendix B.1, uses an intermediate surface SES_{r_s}(SAS) defined as the SES with probe radius r_s, of the SAS. We show that both the SAS and $\mathcal{T}_r$ are isotopic to SES_{r_s}(SAS). Both isotopies are constructed by defining a partition of the volume where each cell is centered on an atom or a probe center.

Corollary. 1. *The SAS surface and $\mathcal{T}_r$ are homeomorphic: they have the same number of connected components and the same genus on each component.*

From the proper offset to the α-complex. As $r \to 0$, the 2-manifold $\mathcal{T}_r$ collapses onto the polyhedral boundary of $|\mathcal{K}_\alpha|$. While this limit is the natural piecewise-linear candidate for a triangulated molecular mesh, it fails to be a topological 2-manifold at specific degenerate configurations. These defects fall into an exhaustive classification detailed in Section C):

1. Singular triangles: in $\mathcal{T}_r$, such a simplex is enclosed by two distinct sheets separated by $2r$; at $r \to 0$, both sheets collapse onto a single unoriented triangle.
2. Singular dangling edges, incident to zero triangles, where the cylindrical offset boundary collapses into a 1-dimensional segment.
3. Finned edges shared by more than two triangles.
4. Pinch vertices that bridge two connected components through an annulus in the offset.
5. Vertices whose boundary neighborhood decomposes into multiple disconnected patches.

Two avenues open to us: either keep $\mathcal{T}_r$ and pay the cost of a triangulation or try to perform local operation on the α-complex to mitigate those defects. In this paper, we choose the second option.

2.3 Building a Mesh from the Boundary of the α-complex

We propose AlphaSurf, a fast procedure to build triangle meshes amenable to learning from the α-complex. First, we construct our α-complex using the robust algorithms in CGAL (The CGAL Project). Then, we select the regular and singular simplices contributing to $\partial|\mathcal{K}_0|$.

The previous section identifies topological defects of this boundary compared to the SAS. We analyze those defects in light of the learning approaches we aim to use. Those approaches rely on heat diffusion on the cotan Laplace-Beltrami Operator. Defect 1) amounts to merging the faces of singular triangles but allows heat diffusion. Similarly, defect 4) does not prevent heat diffusion since the cotan sums the contributions from connected components. Defect 3) distorts heat diffusion, but does

not prevent it, hence creating artificial local shortcuts. We consider those acceptable in a machine learning setting.

Handling defect 2) can be done by locally expanding the dangling edge as a triangle or, to be more thorough, to locally triangulate the cylinder and wire it back around the supporting vertex. In practice, dangling edges mostly occur at the end of side chains, which frequency-cropped heat diffusion ignores anyway. Occasionally, these edges connect two connected components, as illustrated in Section E.3, but these are extremely rare as detailed in the next sections. In our algorithm, we ignore this defect.

The last defect (5) occurs when an atom might contribute multiple disconnected spherical patches to the boundary of the union of balls. In the α-complex, these are associated with the same vertex, artificially connecting those disconnected SAS patches in our mesh. This happens relatively frequently and leads to strong disruptions of the heat diffusion, creating major shortcuts through the protein.

To restore the right connectivity, we introduce as many copies of the atom-centered vertex as there are disconnected patches and connect each copy with its neighbors based on the boundaries of each patch (see Figure 3). The resulting vertices remain geometrically coincident but are combinatorially distinct, restoring the connectivity of the union-of-balls boundary without altering its geometry. Note that this fixes some 1) and 2) defects (95% of singular faces in practice). We call this process *tufting* in reference to Sharp & Crane (2020). The complete geometric procedure is given in Algorithm 2 and the impact of this step is investigated in Appendix D.

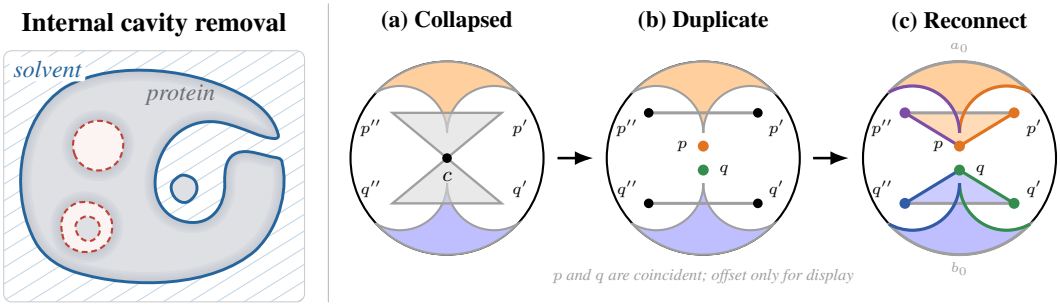


Figure 3: *Left:* Shaded area represents the protein and blue hatching marks the solvent-connected region. Surfaces drawn in solid blue are retained; enclosed cavities, and any surface they contain (dashed red), are discarded. The open pocket on the right is connected to the solvent and is kept. *Right:* Two disconnected patches (orange and blue) contribute through the same atom/vertex c. We create one copy of c for each patch and restore its connectivity based on the patch connectivity.

Our last step is to remove internal cavities that are not solvent-exposed, and isolate the outer layer. To do so, we consider the exterior three-dimensional cells of the weighted regular triangulation and connect two such cells when they share an exterior facet. The connected component containing the infinite cells defines the solvent-connected exterior. Finite exterior-cell components that cannot reach infinity are therefore excluded as enclosed cavities. The surface facets are the interfaces between this infinity-connected exterior component and the remaining cells, together with singular facets between two infinity-connected exterior cells. This is illustrated on the left of Figure 3.

2.4 Empirical validation

Efficiency and usability. To test AlphaSurf, we compute protein surfaces on monomers of the large-scale PINDER dataset (Kovtun et al., 2024) (details in Appendix F.1). We compare our method to MSMS, the most popular protein meshing tool, and to the faster and more recent EDTSurf and NanoShaper (v2). Those tools were developed for large complex visualization, and were not optimized for on-the-fly processing for learning.

We evaluate the computational cost and geometric validity of the resulting surfaces. Moreover, modern surface encoders (e.g. DiffusionNet (Sharp et al., 2022) and PoissonNet (Maesumi et al., 2025)) rely on spectral methods, requiring meshes with a single connected component. Thus, in learning pipelines, all but the largest connected component are discarded. We define a protein surface mesh to be *usable at* $x\%$ when the discarded vertices represent at most x% of the original ones. The

usability at different cutoffs can be found in Supplementary Figure S12 and we use 1% in what follows.

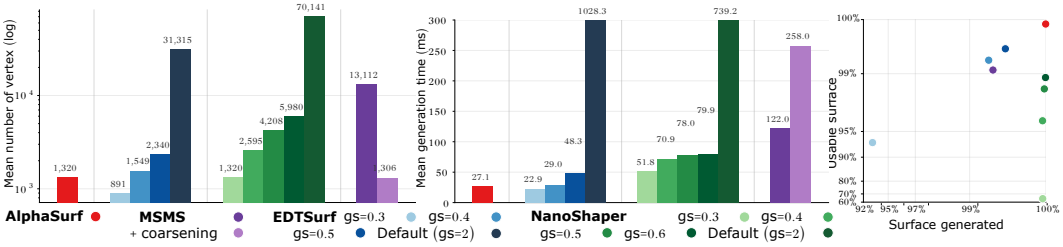


Figure 4: Efficiency, surface size, and usability of the different molecular surface-generation methods on the PINDER dataset.

Figure 4 shows that AlphaSurf is significantly faster and coarser than existing tools under their recommended setup (27ms vs 258ms, 1,300 vs 13,000 vertices), without any parameter tuning. To reach a similar number of vertices, AtomSurf coarsens MSMS meshes, worsening surface creation time. By substantially reducing the grid scale (grid points per Å) of NanoShaper and EDTSurf, we can obtain coarser meshes competitive with AlphaSurf in size and computing time. Representative examples of the generated meshes are provided in Appendix E.2. Note that these aggressive low-resolution settings lie outside the regimes in which these methods were originally evaluated.

Tractability gains come at a price for these methods. To reach competitive runtime, NanoShaper must be used with a grid scale of 0.3, resulting in fragmented surfaces and less than 70% of usable meshes, whereas AlphaSurf produces close to 100% of usable meshes, see Appendix H.3 for more details. Despite retaining 99% of usability at a grid scale of 0.4, EDTSurf yields oversmoothed meshes. Altogether, AlphaSurf displays favorable properties for ultra-fast, coarse protein surface meshing.

Heat diffusion on coarse meshes. Surface learning models propagate features through heat diffusion on meshes. Having assessed the efficiency and overall usability of those meshes, we now ask whether coarse meshes keep the diffusion faithful. Remember that our surfaces have an offset since they directly use atoms as vertices.

We compare heat diffusion on each mesh to a full-resolution MSMS reference, using both the HKS (as in DiffusionNet) and kernel values $K_t(i, j)$ between points sampled on the reference. We use $t = 10$, after matching each method's mean surface area to the reference and we consider pairs of points below 15 Å based on the decay of the kernel (Supplementary Figure S10). We additionally plot the results of diffusion for the different methods.

AlphaSurf is more faithful than NanoShaper at grid scale 0.3 and than EDTSurf. Only larger meshes, or MSMS simplified, which is decimated from the reference itself, do better. Tufting prevents diffusion from leaking across disconnected sheets and improves the worst cases. NanoShaper also suffers from leaking. Overall, despite their offset and coarse resolutions, our meshes empirically allow faithful heat diffusion.

3 Learning on AlphaSurf meshes

In this section, we first show that the coarse meshes produced by AlphaSurf are sufficient to retain state-of-the-art performance of surface-based approaches on existing tasks (Section 3.1). We then scale to use cases previously unreachable for surface-based methods (learning on dynamics and with data augmentation) in Section 3.2.

We follow the setup of AtomSurf (Mallet et al., 2024). The model combines ProNet (Wang et al., 2022) (a residue-level protein graph encoder) with DiffusionNet (Sharp et al., 2022) (a surface encoder), followed by bidirectional communication between the graph and surface representations. Input features and hyperparameters are retained without modification, except for the surface mesh used. We include details for each task setup in Appendix G.

For learning applications, it should be noted that extra preprocessing steps are needed on the triangular meshes, such as computing Laplace-Beltrami operator eigenvectors. The timing of these steps

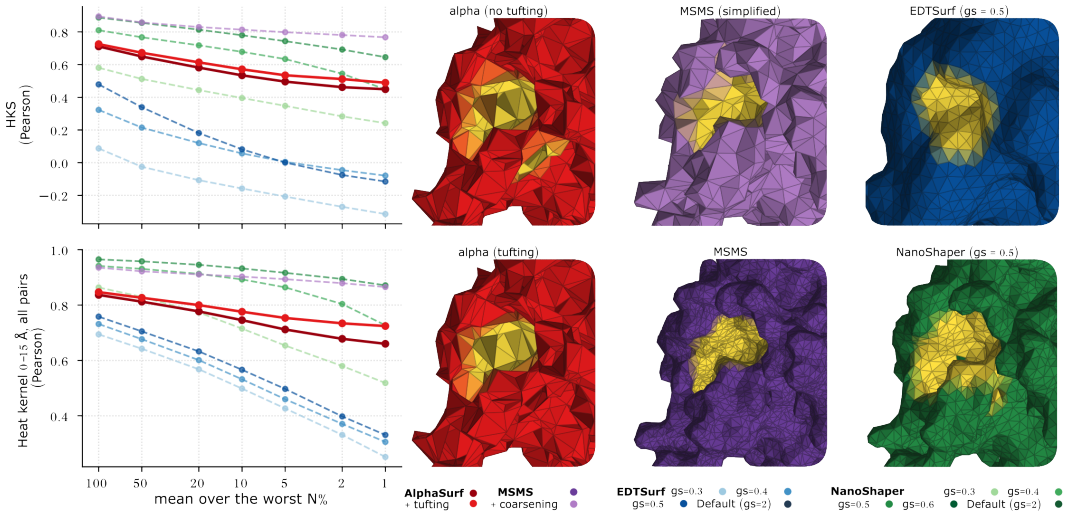


Figure 5: Faithfulness of heat diffusion on coarse meshes, relative to the full-resolution MSMS surface. *Left:* Pearson correlation with the reference of the heat kernel signature (top) and of the heat kernel $K_t(i, j)$ at $t = 10$ for pairs closer than 15 Å (bottom), averaged over the worst $N\%$ of proteins. *Right:* heat kernel from a source in a binding pocket on each mesh.

is broken down in Section H.5. Below, we report the amortized processing throughput on twenty CPUs and compare it to the training loop GPU processing time.

3.1 AlphaSurf retains state-of-the-art performance

MaSIF-Ligand. We first evaluate different surfaces on the MaSIF-Ligand task (2k pockets). An analysis of the resulting surfaces on this dataset, similar to Section 2.4, is available in Appendix H.4. The objective of the task is to classify binding pockets according to their binder among seven possible cofactors. We report the performance of different meshes in Figure 6, presented as balanced accuracy as a function of throughput. Shaded areas indicate that data preparation exceeds processing time. AlphaSurf, NanoShaper and MSMS obtain similar performance, with EDTSurf lagging 2 to 5 points behind. MSMS is far too slow for on-the-fly preprocessing.

Pinder-Pair. We then evaluate surface representations on the Pinder-Pair task (42k holo training dimers) introduced in AtomSurf. Given the individual monomers of a protein dimer, the task consists in predicting which pairs of residues are closer than 5Å in the bound complex. We use the same throughput-performance analysis as above. We split performance as a function of the nature of the dimers (homo or hetero), as well as the nature of monomers by increasing difficulty: holo, apo, AF2-predicted.

Only AlphaSurf and EDTSurf can be generated on-the-fly on this task. NanoShaper becomes more expensive on larger inputs, due to its tendency to generate more vertices. In terms of performance too, two groups emerge. NanoShaper and MSMS display strong Holo performance, outperforming other methods by 3 AUROC points. However, their performance drops sharply on the Apo and AF2 splits, falling behind the second group by 9 points. The reduced overfitting behavior of the second group could be explained by their coarser meshes, preventing exact surface comparison.

3.2 AlphaSurf unlocks new use cases

Large-scale tasks S3F (Zhang et al., 2024) proposed a surface-based method for the mutation stability task, and showed that encoding the surface improved their performance compared to their S2F method. To encode their surface, they generate a surface point cloud with dMaSIF. They pretrain a structure encoder on demasking on a large training set, and score amino-acid substitutions with predicted logits. They validate their prediction by comparing to deep mutational scanning experiment results, on the ProteinGym benchmark, covering over 2.4M mutations in 217 assays.

We propose to replace their point cloud generation with a point cloud derived from the alpha complex while strictly preserving all other aspects. Because of the stochastic nature of dMaSIF, S3F

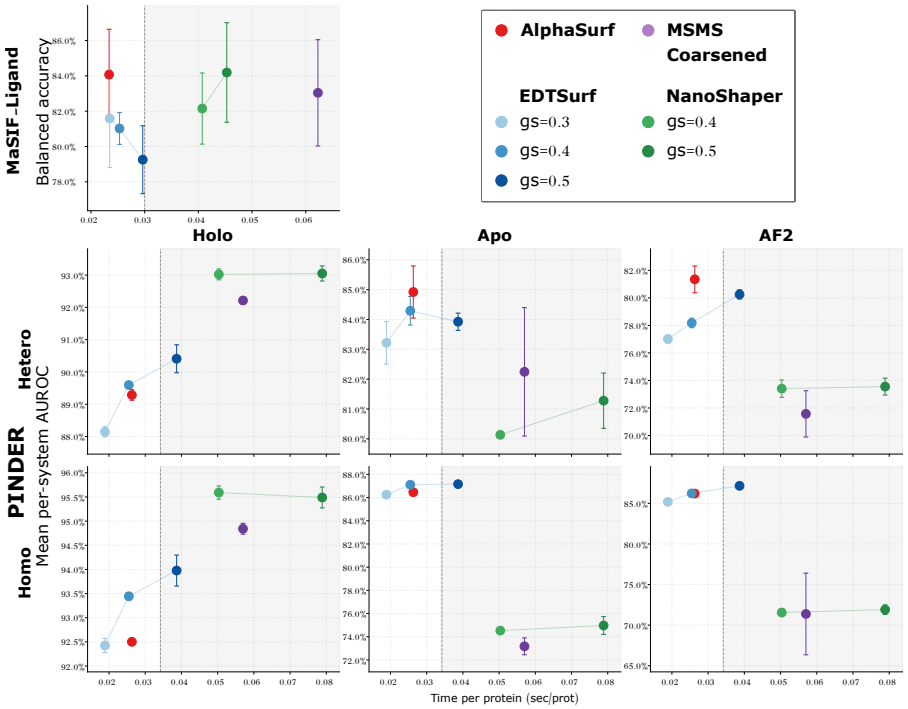


Figure 6: MaSIF-Ligand and Pinder-Pair predictive performance versus input-construction throughput for the evaluated molecular surface representations. Vertical line = model training throughput.

samples numerous points and spectral feature computation becomes a bottleneck. In practice, S3F precomputed these operators resulting in millions of files and terabytes of data. AlphaSurf obtains similar performance using far fewer points, resulting in roughly 80 times faster, on-the-fly data generation.

Method	Spearman correlation	# Vertices	Surface generation	Forward + backward
S3F	0.470	4,136	424	56.5
α-S3F	0.474	**563**	**5.2**	**23.8**

Table 1: ProteinGym Spearman correlation, then average number of surface vertices and surface generation + featurization time per protein on the CATH pretraining set (31,303 domains; 20 workers, one H100), and forward + backward time (batch size 8). Times are in ms per protein.

Data Augmentation. On-the-fly preprocessing significantly facilitates adoption, by avoiding a separate preprocessing step and saving gigabytes of precomputed operators to disk. Beyond moderate-size datasets, we also propose use cases where precomputation is not tractable. First, we consider data augmentation, in the form of light noising of the input coordinates. Such augmentation is common practice in geometric graph learning pipelines. For instance, our graph encoder (ProNet) uses Gaussian noise of scale 0.5 Å. We inject noise into protein coordinates before computing the surface, and also perturb the surface along normals before computing the Laplacian operator. We perform our experiment on the Pinder dataset, and report our results in Figure 7. This noising procedure results in a modest but significant performance increase across most setups. This boost is more marked on the Apo and AF2 splits, indicating enhanced generalization.

Our new surface also permits a unique data augmentation technique: instead of adding noise to the input coordinates, we sample α uniformly in $[0, \alpha_{\max}]$. When using positive α, the validity guarantees we provide still hold. This results in small gains for $\alpha_{\max} = 1$ and on the AF2 split, but collapses on Apo and AF2 for larger values of $\alpha_{\max}$ while enhancing Holo.

Learning on Dynamics. Beyond learning on static protein structure, the next frontier lies in taking the dynamics of protein into account. The MISATO (Siebenmorgen et al., 2024) dataset contains

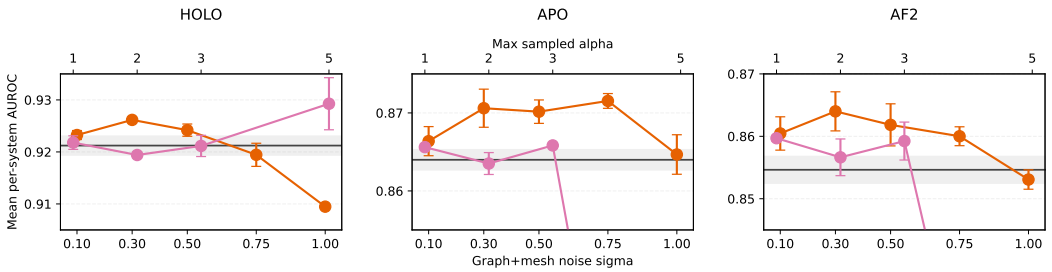


Figure 7: Augmentation on PINDER-Pair: coordinate noise (orange, bottom axis) versus alpha sampling (pink, top axis), against the unaugmented baseline (grey).

15k short MD simulations of protein-ligand complexes starting from the holo conformation, each containing 100 MD frames. Guo et al. (2025) defined a binding-site prediction task that aims to predict residues within 10 Å of the ligand in the holo structure.

We train a model taking a single conformation as input by randomly sampling a frame at every epoch. This can be seen as another data augmentation strategy, exposing the model to alternative conformations rather than artificial coordinate noise. We compare to a model trained only on holo conformations. Both models are tested on holo conformations. We compare to the best model reported by Portal et al. (2026) (R-EGNN, distance + distogram edges); AUROC and AUPRC are not reported. We present the results in Table 2.

Training / model	**F1**	**AUROC**	**AUPRC**
DistoDyn†	46.4%	–	–
Holo conformations	$48.9 \pm 0.5\%$	$85.7 \pm 0.3\%$	$47.7 \pm 0.8\%$
All-frames	$\mathbf{50.3 \pm 0.2\%}$	$\mathbf{86.8 \pm 0.1\%}$	$\mathbf{50.4 \pm 0.5\%}$

Table 2: Binding-site prediction performances on the MISATO dataset

Random-frame training achieves the best performance on all metrics. These results support molecular-dynamics frame sampling as a principled form of structural data augmentation. The precomputation and storage of 1.5M protein meshes, Laplacian operators and their eigenvectors is avoided by AlphaSurf's on-the-fly computation.

4 Discussion

Going back to the foundations of molecular surface computations, we notice that current surface-based learning methods could use coarser and faster surfaces. We implement this approach in AlphaSurf, which retains performance while drastically cutting preprocessing times by using the α-complex and bypassing triangulation. AlphaSurf builds a mesh in 27 ms with about 1.3k vertices, without any parameter tuning, and remains competitive with MSMS-based meshes on MaSIF-Ligand and PINDER-Pair. Our work allows on-the-fly surface computations, facilitating adoption of surface-based protein representation learning and opening the door to surface data augmentation and learning on molecular dynamics. Noise augmentation improves PINDER-Pair AUROC by 0.5–0.9 points, training on MD frames improves MISATO F1 by 1.4 points, and the surface generation of S3F becomes about 80 times cheaper with unchanged ProteinGym performance.

Our work also includes several limitations. Our algorithm does not currently handle singular edges (Fig. S9), so that the triangle mesh generated is not strictly speaking homeomorphic to the SAS surface. A dedicated routine could be implemented to restore topological equivalence. Moreover, most of the preprocessing time lies in spectral computations. In the future, using smaller models and fewer eigenvectors would result in an even lighter approach, cutting the runtime by another order of magnitude. Investigating the performance of such an approach represents an interesting follow-up. Finally, since AlphaSurf meshes can be recomputed whenever atoms move, surface encoders could be integrated into generative models, either to score candidate binders or as a message-passing component of the model, towards surface-aware protein.

REPRODUCIBILITY STATEMENT

Code for AlphaSurf and for all experiments is available in an anonymous repository linked in the abstract. The meshing procedure is described in Section 2.3, and the tufting step in Appendix D (Algorithm 2); α-complexes are computed with CGAL 6.2, as detailed in Appendix A. The assumptions and proof of Theorem 1 are given in Appendix B. PINDER preprocessing and connected-component filtering are described in Appendix F.1. Data splits, hyperparameters and optimization settings for MaSIF-Ligand, PINDER-Pair, MISATO, on-the-fly noising, and S3F pretraining with ProteinGym evaluation are listed in Appendix G, including the per-method surface-label thresholds and the reduced PINDER test sets on which all methods are compared. Baseline tools and grid scales are given in Section 2.4 and Figure 4 (MSMS with the coarsening used in AtomSurf, EDTSurf, and NanoShaper v2); example meshes are shown in Appendix E.2. The heat-kernel evaluation protocol, including the per-method area normalization, is described in Appendix H.1, and the tufting ablation in Appendix H.7. Per-stage preprocessing timings and the hardware used are reported in Appendix H.5, and the forward–backward timings that define the on-the-fly threshold of Figure 6 in Appendix H.6; S3F timings follow the setup given in Table 1. Results on MaSIF-Ligand, PINDER-Pair and MISATO are reported over three seeds (2024, 2025, 2026); S3F pretraining uses a single seed (2024).

AI USE STATEMENT

In this work, we used generative AI tools for feature and plotting implementations, to proof check the overall syntax and orthography of the paper. We have not used generative AI tools for any mathematical proofs, research claims and interpretations. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness by the first author. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

N. Akkiraju and H. Edelsbrunner. Triangulating the surface of a molecule. *Discrete Appl. Math.*, 71:5–22, 1996.

Sharmi Banerjee, Mostafa Karimi, Melih Yilmaz, Tommi Jaakkola, Bella Dubrov, Shang Shang, and Ron Benson. Pi-sage: Permutation-invariant surface-aware graph encoder for binding affinity prediction. *arXiv preprint arXiv:2508.01924*, 2025.

J-D. Boissonnat, O. Devillers, and J. Duquesne. Computing connolly surfaces. In *IFIP Conf. on algorithms and efficient computation*, 1992.

J-D. Boissonnat, C. Wormser, and M. Yvinec. Curved voronoi diagrams. In J.-D. Boissonnat and M. Teillaud (eds.), *Effective Computational Geometry for curves and surfaces*. Springer-Verlag, Mathematics and Visualization, 2006.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

K. Buchin and J. Giesen. Flow complex: General structure and algorithm. In *Proc. 16th Canad. Conf. Comput. Geom*, pp. 270–273, 2005.

Duanhua Cao, Mingan Chen, Runze Zhang, Zhaokun Wang, Manlin Huang, Jie Yu, Xinyu Jiang, Zhehuan Fan, Wei Zhang, Hao Zhou, and others. SurfDock is a surface-informed diffusion generative model for reliable and accurate protein–ligand complex prediction. *Nature Methods*, 22 (2):310–322, 2025.

F. Cazals, A. Parameswaran, and S. Pion. Robust construction of the three-dimensional flow complex. In *ACM Symposium on Computational Geometry*, pp. 182–191, 2008.

F. Cazals, H. Kanhere, and S. Loriot. Computing the volume of union of balls: a certified algorithm. *ACM Transactions on Mathematical Software*, 38(1):1–20, 2011. doi: 10.1145/2049662.2049665. URL http://hal.inria.fr/hal-00849809.

F. Chazal, D. Cohen-Steiner, and A. Lieutier. A sampling theory for compact sets in Euclidean space. *Discrete and Computational Geometry*, 41(3):461–479, 2009. ISSN 0179-5376.

M. Connolly. Molecular surfaces: A review. *Network Science*, 14, 1996. URL http://www.netsci.org/Science/Compchem/feature14.html.

M. L. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221:709–713, 1983.

Sergio Decherchi and Walter Rocchia. A general and robust ray-casting-based algorithm for triangulating surfaces at the nanoscale. *PloS one*, 8(4):e59744, 2013.

Warren L DeLano et al. Pymol: An open-source molecular graphics tool. *CCP4 Newsl. protein crystallogr*, 40(1):82–92, 2002.

H. Edelsbrunner. Weighted alpha shapes. Technical Report UIUCDCS-R-92-1760, Dept. Comput. Sci., Univ. Illinois, Urbana, IL, 1992. was e-was-92.

H. Edelsbrunner. The union of balls and its dual shape. *Discrete Comput. Geom.*, 13:415–440, 1995.

Herbert Edelsbrunner. *Geometry and Topology for Mesh Generation*, volume 7 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, UK, 2001. ISBN 978-0-521-79309-4. doi: 10.1017/CBO9780511530067.

Rida T Farouki. Exact offset procedures for simple solids. *Computer Aided Geometric Design*, 2 (4):257–279, 1985.

Efi Fogel and Dan Halperin. Exact and efficient construction of minkowski sums of convex polyhedra with applications. *Computer-Aided Design*, 39(11):929–940, 2007.

P. Gainza, F. Sverrisson, F. Monti, E. Rodolà, D. Boscaini, M. M. Bronstein, and B. E. Correia. Deciphering interaction fingerprints from protein molecular surfaces using geometric deep learning. *Nature Methods*, 17(2):184–192, 2020.

Angelo Gavezzotti. *Molecular aggregation: structure analysis and molecular simulation of crystals and liquids*, volume 19. OUP Oxford, 2007.

M. Gerstein and F.M. Richards. Protein geometry: volumes, areas, and distances. In M. G. Rossmann and E. Arnold (eds.), *The international tables for crystallography (Vol F, Chap. 22)*, pp. 531–539. Springer, 2001.

Mark Gerstein, Jerry Tsai, and Michael Levitt. The volume of atoms on the protein surface: calculated from simulation, using voronoi polyhedra. *Journal of molecular biology*, 249(5):955–966, 1995.

Pengkang Guo, Bruno Correia, Pierre Vandergheynst, and Daniel Probst. Boosting protein graph representations through static-dynamic fusion. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 20777–20792. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/guo25b.html.

Dan Halperin and Mark H Overmars. Spheres, molecules, and hidden surface removal. In *Proceedings of the tenth annual symposium on Computational geometry*, pp. 113–122. ACM, 1994.

J. Janin, R. P. Bahadur, and P. Chakrabarti. Protein-protein interaction and quaternary structure. *Quarterly reviews of biophysics*, 41(2):133–180, 2008.

L. Kettner, K. Mehlhorn, S. Pion, S. Schirra, and C. Yap. Classroom examples of robustness problems in geometric computations. *Computational Geometry*, 40(1):61–78, 2008.

Daniel Kovtun, Mehmet Akdel, Alexander Goncearenco, Guoqing Zhou, Graham Holt, David Baugher, Dejun Lin, Yusuf Adeshina, Thomas Castiglione, Xiaoyun Wang, Céline Marquet, Matt McPartlon, Tomas Geffner, Gabriele Corso, Hannes Stärk, Zachary Carpenter, Emine Kucukbenli, Michael Bronstein, and Luca Naef. PINDER: The protein interaction dataset and evaluation resource. *bioRxiv*, 2024. doi: 10.1101/2024.07.17.603980.

Byungkook Lee and Frederic M Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of molecular biology*, 55(3):379–IN4, 1971.

Yanting Li, Jiyue Jiang, Zikang Wang, Ziqian Lin, Dongchen He, Yuheng Shan, Yanruisheng Shao, Jiayi Li, Xiangyu Shi, Jiuming Wang, et al. Ds-progen: A dual-structure deep language model for functional protein design. *arXiv preprint arXiv:2505.12511*, 2025.

S. Loriot and F. Cazals. Modeling Macro-Molecular Interfaces with Intervor. *Bioinformatics*, 26 (7):964–965, 2010. doi: 10.1093/bioinformatics/btq052. URL http://hal.inria.fr/hal-00849822.

Arman Maesumi, Tanish Makadia, Thibault Groueix, Vladimir Kim, Daniel Ritchie, and Noam Aigerman. Poissonnet: A local-global approach for learning on surfaces. *ACM Transactions on Graphics (TOG)*, 44(6):1–16, 2025.

Vincent Mallet, Souhaib Attaiki, Yangyang Miao, Bruno Correia, and Maks Ovsjanikov. Atomsurf : Surface representation for learning on protein structures, 2024. URL https://arxiv.org/abs/2309.16519.

Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 37–45, 2015.

J.W. Milnor. *Morse Theory*. Princeton University Press, Princeton, NJ, 1963. m-mt-63.

Stelios K Mylonas, Apostolos Axenopoulos, and Petros Daras. Deepsurf: a surface-based deep learning approach for the prediction of ligand binding sites on proteins. *Bioinformatics*, 37(12): 1681–1690, 2021.

Nicolas Portal, Wissam Karroucha, Vincent Mallet, and Massimiliano Bonomi. Learning dynamic protein representations at scale with distograms. In *ICLR 2026 Workshop on Learning Meaningful Representations of Life (LMRL)*, 2026.

Jie Qian, Lin Yang, Renxiao Wang, and Yifei Qi. Pro4s: prediction of protein solubility by fusing sequence, structure, and surface. *bioRxiv*, pp. 2025–11, 2025.

Saleh Riahi, Jae Hyeon Lee, Taylor Sorenson, Shuai Wei, Sven Jager, Reza Olfati-Saber, Yanfeng Zhou, Anna Park, Maria Wendt, Hervé Minoux, and Yu Qiu. Surface ID: a geometry-aware system for protein molecular surface comparison. *Bioinformatics*, 39(4):btad196, 2023.

Colin P. Rourke and Brian J. Sanderson. *Introduction to Piecewise-Linear Topology*, volume 69 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, Berlin, Heidelberg, 1982. ISBN 978-3-540-11102-3. doi: 10.1007/978-3-642-81735-9.

Michel F Sanner, Arthur J Olson, and Jean-Claude Spehner. Fast and robust computation of molecular surfaces. In *Proceedings of the eleventh annual symposium on Computational geometry*, pp. 406–407, 1995.

Nicholas Sharp and Keenan Crane. A Laplacian for Nonmanifold Triangle Meshes. *Computer Graphics Forum*, 39(5):69–80, August 2020. ISSN 1467-8659. doi: 10.1111/cgf.14069. URL https://onlinelibrary.wiley.com/doi/10.1111/cgf.14069.

Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

Till Siebenmorgen, Filipe Menezes, Sabrina Benassou, Erinc Merdivan, Kieran Didi, André Santos Dias Mourão, Radosław Kitel, Pietro Liò, Stefan Kesselheim, Marie Piraud, and others. MISATO: machine learning dataset of protein–ligand complexes for structure-based drug discovery. *Nature Computational Science*, 4:367–378, 2024.

Vignesh Ram Somnath, Charlotte Bunne, and Andreas Krause. Multi-scale representation learning on proteins. *Advances in Neural Information Processing Systems*, 34:25244–25255, 2021.

Freyr Sverrisson, Jean Feydy, Bruno E Correia, and Michael M Bronstein. Fast end-to-end learning on protein surfaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15272–15281, 2021.

Freyr Sverrisson, Mehmet Akdel, Dylan Abramson, Jean Feydy, Alexander Goncearenco, Yusuf Adeshina, Daniel Kovtun, Céline Marquet, Xuejin Zhang, David Baugher, Zachary Carpenter, Luca Naef, Michael M Bronstein, and Bruno Correia. DiffMaSIF: Surface-based Protein-Protein Docking with Diffusion Models. In *Machine Learning in Structural Biology workshop at NeurIPS 2023*, New Orleans, United States, December 2023. URL https://hal.science/hal-04360638.

The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 6.2 edition. URL https://www.cgal.org.

Limei Wang, Haoran Liu, Yi Liu, Jerry Kurtin, and Shuiwang Ji. Learning hierarchical protein representations via complete 3d graph networks, 2022. URL https://arxiv.org/abs/2207.12600.

Yiqun Wang, Yuning Shen, Shi Chen, Lihao Wang, Fei Ye, and Hao Zhou. Learning harmonic molecular representations on riemannian manifold. *arXiv preprint arXiv:2303.15520*, 2023.

Ron Wein. Exact and approximate construction of offset polygons. *Computer-Aided Design*, 39(6): 518–527, 2007.

Fang Wu and Stan Z. Li. Surface-VQMAE: Vector-quantized masked auto-encoders on molecular surfaces. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=szxtVHOh0C.

Fang Wu, Shuting Jin, Xiangru Tang, Mark Gerstein, Xiangxiang Zeng, Yejin Choi, Jure Leskovec, and Jinbo Xu. SurfDesign: Effective Protein Design on Molecular Surfaces, May 2026a. URL http://arxiv.org/abs/2606.07567. arXiv:2606.07567 [q-bio.BM].

Fang Wu, Weihao Xuan, Jure Leskovec, Yejin Choi, and Li Erran Li. Deciphering Fingerprints of 3D Molecular Surfaces For Accurate Epitope Prediction. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, KDD '26, pp. 12408–12419, New York, NY, USA, 2026b. Association for Computing Machinery. ISBN 979-8-4007-2259-2. doi: 10.1145/3770855.3818825. URL https://dl.acm.org/doi/10.1145/3770855.3818825.

Dong Xu and Yang Zhang. Generating triangulated macromolecular surfaces by euclidean distance transform. *PloS one*, 4(12):e8140, 2009.

Zuobai Zhang, Pascal Notin, Yining Huang, Aurelie C Lozano, Vijil Chenthamarakshan, Debora Marks, Payel Das, and Jian Tang. Multi-scale representation learning for protein fitness prediction. *Advances in Neural Information Processing Systems*, 37:101456–101473, 2024.

Wanyao Zhou, Yuchengze Song, Jixiao He, Mingyu Li, Jian Zhang, and Shaoyong Lu. Multialldriver: a multi-model method to predict and identify cancer driver mutations. *bioRxiv*, pp. 2025–06, 2025.

Appendix

A Further mathematical explanations on surface computations

Relation between the $\mathcal{K}_\alpha$ and the boundary of a union of balls The theory of weighted α-complexes is non-trivial, see (Edelsbrunner, 1992, Table 5.1) and (Akkiraju & Edelsbrunner, 1996). Assume we want to compute the boundary $\partial \bigcup B_i$ of a collection of balls $\{B_i\}$. As mentioned in the main text, we use $\mathcal{S} = \{B_i(x_i, r_i + r_w)\}$ to define the SAS. In the following, we just provide the link between the so-called *singular* and *regular* simplices of $\mathcal{K}_0$ for these balls, and $\partial \bigcup B_i$.

In short, singular and regular simplices make the following contributions to $\partial \bigcup B_i$:

A *singular* simplex in $\mathcal{K}_\alpha$ is a simplex which does not have any coface. That is, this simplex is not the face of a higher dimensional simplex. Cases of interest contributing to $\partial \bigcup B_i$ are:

- singular vertex: contributes a full sphere. (Does not happen in proteins since all protein atoms have covalent bonds.)
- singular edge: contributes the full circle defined by two intersecting spheres. (Typically happens to atoms at the *tip* of side-chains.)
- singular triangle: contributes the two intersection points defined by its three spheres. (Happens in *thin* regions of the proteins, when atoms form a layer.)

A *regular* simplex in $\mathcal{K}_\alpha$ is a simplex which is not singular and satisfies either condition: it belongs to the convex hull (the boundary of the Delaunay/regular triangulation), or it belongs to the interior and does not have all its cofaces in $\mathcal{K}_\alpha$. Cases of interest contributing to $\partial \bigcup B_i$ are:

- regular vertex: contributes one or several spherical caps.
- regular edge: contributes one or several circle arcs.
- regular triangle: contributes one of the two intersection points defined by its three spheres.

Computing the α-complex. It should also be noted that the computation of the (regular) Delaunay triangulation can be unstable, as rounding errors in (nearly) degenerate configurations confuse the logic of the algorithms (Kettner et al., 2008). For this reason, the effective calculations of α-complexes only became possible with the engineering on number types and geometric predicates, as performed within the Computational Geometry Algorithms Library (CGAL) (The CGAL Project). In practice, we use the Fixed_alpha_shape_3 class from CGAL.

B Isotopy

B.1 Preliminaries

Notations. Consider a collection of atoms $A_i(x_i, R_i)$ and a water probe r_w. The enlarged atoms are denoted $S_i(x_i, R_i + r_w)$, and the union of the balls $\cup_i B_i$ associated with the spheres S_i is denoted U. We take for granted the construction of the Solvent Accessible Surface SAS_{r_w} and the Solvent Excluded Surface SES_r using the α-complex (Boissonnat et al., 1992; Edelsbrunner, 1995; Akkiraju & Edelsbrunner, 1996; Connolly, 1996; Cazals et al., 2011).

Define:

- The set K of singular and regular simplices of $\mathcal{K}_0$ coding the boundary ∂U.
- The proper r-offset $\mathcal{T}_r$ of $\mathcal{K}_0$ with a suitable value of r. Let $B(\cdot, r)$ be a ball of radius r. The r-offset is defined using the Minkowski sum (pointwise addition) of the polygon or polyhedron with $B(\cdot, r)$, and the proper offset is its boundary:

$$\mathcal{T}_r = \partial(|\mathcal{K}_0| \oplus B(0, r)). \tag{1}$$

  Since $\mathcal{K}_0$ is a simplicial complex, its proper offset is a cell complex whose structure is well known:

  - 2D case: the proper offset involves segments and circle arcs, see (Wein, 2007).

  - 3D case: the proper offset involves planar facets, cylindrical parts, and spherical caps, see (Farouki, 1985; Fogel & Halperin, 2007).

  The determination of r is handled in Section B.2.

- The solvent accessible surface, that is the boundary of the union of balls:

$$\text{SAS}_{r_w}(\{A_i\}).$$

  For the sake of conciseness, we simply write SAS_{r_w} in the sequel.

- The solvent excluded surface (aka Connolly) obtained from SAS_{r_w} using a probe of radius r_s:

$$\text{SES}_{r_s}(\text{SAS}_{r_w}(\{A_i\})).$$

  For the sake of conciseness, we simply write SES_{r_s} in the sequel.

Given a curve (2D) or surface (3D), we use decompositions into a disjoint union of open cells denoted $\mathcal{X}(\cdot)$. For example, for $\mathcal{T}_r$:

$$\mathcal{T}_r = \coprod_{e \in \mathcal{X}(\mathcal{T}_r)} e. \tag{2}$$

The topological closure of a cell is denoted $\overline{e}$.

Proof outline. We establish an isotopy between $\mathcal{T}_r$ and the SAS_{r_w} in two steps:

- Step 1: build an isotopy between $\mathcal{T}_r$ and SES_{r_s} as follows:
  1. (Simplicial complex coding the boundary of the union of balls) Compute the simplicial complex K involving the singular and regular faces of the 0-complex of the balls $\{S_i(x_i, R_i + r_w)\}$.
  2. (Offset) Compute a valid offset radius r_s for $\{S_i\}$ and build $\mathcal{T}_r$. See Sec. B.2.
  3. (Decompositions) Build decompositions $\mathcal{X}(\mathcal{T}_r)$ of $\mathcal{T}_r$, $\mathcal{X}(\text{SES}_{r_s})$ for SES_{r_s}, whose fully dimensional cells are in 1-1 correspondence. See Algorithm 1.
  4. (Explicit isotopy) Use the prismatic cells defined by the decompositions to define the isotopy.
- Step 2: build an isotopy between SES_{r_s} and SAS_{r_w}.

B.2 Intermezzo: tangent spheres, Apollonius diagrams, and offset radii

Our constructions involve probe spheres tangent to a set of spheres corresponding to atoms. We note that we use a radius r_s determined as explained below rather than the radius r_w of the water probe defining SAS_{r_w} and the classical Connolly surface SES_{r_w}.

We therefore review basic properties of Apollonius diagrams coding tangencies between spheres.

Apollonius diagrams. The Apollonius diagram of a set of spheres $\{A_i(x_i, R_i)\}$ is defined using the so-called *additively weighted* distance (Boissonnat et al., 2006):

$$\forall x \in \mathbb{R}^d : f(x) = \min_i(\|x - x_i\| - R_i). \tag{3}$$

The Apollonius diagram is the Voronoi diagram obtained with this distance. For spheres of different radii in 3D (resp. 2D), the bisector is a single sheet of a two-sheeted hyperboloid of revolution (resp. a branch of hyperbola) – (Boissonnat et al., 2006) and Fig. S1. This is in stark contrast with power diagrams whose bisectors are flats.

Computationally, the Apollonius diagram of n spheres in $\mathbb{R}^d$ is in correspondence with the power diagram of spheres in $\mathbb{R}^{d+1}$. Its complexity is $O(n^{\lfloor d/2 \rfloor + 1})$ and it can be computed in time $O(n^{\lfloor d/2 \rfloor + 1}) \log n$ (Boissonnat et al., 2006).

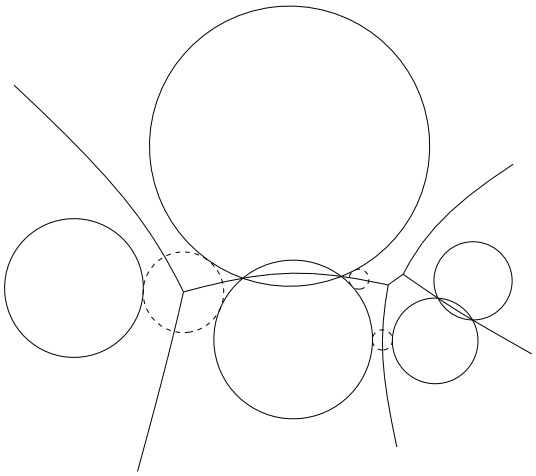


Figure S1: **The Apollonius diagram of five circles in 2D, with bi/tri tangent circles dashed.** The bisectors are branches of hyperbola.

Weak feature size. The classical Morse-Smale theory for smooth functions (Milnor, 1963) generalizes to the setting of *distances* used to build Voronoi diagrams (Chazal et al., 2009; Buchin & Giesen, 2005; Cazals et al., 2008).

For a point x outside the union of balls U, let the *active set* be the set of balls at minimal distance from x:

$$I(x) = \{i \mid \|x - x_i\| - R_i = f(x)\}. \tag{4}$$

A point x is termed *critical* provided that the following holds

$$x \text{ critical } \Leftrightarrow x \in \text{ConvexHull}\{x_i, i \in I(x)\}. \tag{5}$$

Following (Chazal et al., 2009), we define the *weak feature size* (wfs)

$$\text{wfs}(U) = \inf f(x) : x \notin U, x \text{ critical }. \tag{6}$$

Recall that the notion of wfs is more stringent than that of reach: the reach is defined by a number of *nearest neighbors* ≥ 2. At a reach point, one can in general *move away* from the shape to increase the distance. At a critical point, this possibility does not hold: the point is precisely critical because the gradient is not defined and the direction to move away is ambiguous.

In the constructions below, we use a quantity r_s smaller than the wfs of the balls S_i. Enumerating its Voronoi vertices suffices to determine the wfs.

B.3 The 2D case

Decompositions of $\mathcal{T}_r$ and SES_{r_s}. In representing the decompositions $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\text{SES}_{r_s})$, the following conventions are used (Fig. S2 and Fig. S3):

- Points defining the natural cell decomposition of the offset $\mathcal{T}_r$ and the solvent excluded surface SES_{r_s} are represented with circles.
- Points used to define a partition of the previous decompositions, obtained via projections and intersections, are represented with colored squares (orange and purple).
- Cells of the final decomposition $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\text{SES}_{r_s})$ are represented with colors (green, orange, purple).

The construction of the decompositions $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\text{SES}_{r_s})$ actually mimics the computation of the boundary of a union of balls using the α-complex (Akkiraju & Edelsbrunner, 1996). It is described in Algorithm 1 and mainly consists of calling a procedure $\text{CircSweep}(S_i, (a, b))$ to handle the circle arcs contributed by a given atom.

For the 2D case, we summarize our construction with the following–see (Fig. S2(C):

Algorithm 1 The algorithm building the decompositions $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\mathbf{SES}_{r_s})$.

procedure BUILDDECOMPOSITIONS($\mathcal{T}_r$, SES$_{r_s}$)
  for each atom S_i **do** Sweep procedure CircSweep$(S_i, (a, b))$ for each circle arc contributed by A_i to SES$_{r_s}$
    for each circle arc $c = (a, b)$ of S_i contributing to SES$_{r_s}$ **do**
      Back project a and b onto $\mathcal{T}_r$ ▷ Yields orange squares on $\mathcal{T}_r$
      Project the 0-cells of $\mathcal{T}_r$ onto the arc (a, b) ▷ Yields purple squares on SES$_{r_s}$
      for Every pair of consecutive points a_i, a_{i+1} on the circle arc **do**
        Form a prismatic cell V_i with floor and ceiling on $\mathcal{T}_r$ and SES$_{r_s}$ respectively
      end for
    end for
  end for
  for pairs of atoms (S_i, S_j) defining a re-entrant cap on SES$_{r_s}$ **do**
    Form a prismatic cell W_i using the points constructed to build the cells V_i
  end for
end procedure

Proposition. 1. *There is a one-to-one correspondence between the 1-cells of $\mathcal{X}(\mathcal{T}_r)$ and those of $\mathcal{X}(SES_{r_s})$. Moreover, to each pair (P_i, Q_i) of such cells is associated a prismatic cell V_i and a central projection π_a from the center of an atom x_i, or a prismatic cell W_i and a central projection π_p from the center of a probe.*

Since we shall map the $\mathcal{T}_r$ onto SES$_{r_s}$, we simply denote a projection as π, implicitly referring to π_a or π_p^{-1}.

Isotopy between $\mathcal{T}_r$ and SES$_{r_s}$. Consider two compact embedded surfaces S and T in $\mathbb{R}^d$, each decomposed into finitely many closed patches. In our case, $S = \mathcal{T}_r$ and $T =$ SES$_{r_s}$, and the decompositions are given by $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}$(SES$_{r_s}$). Note that by patches, we refer to the topological closure of cells of dimension $d-1$ in $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}$(SES$_{r_s}$).

Consider two pairs of patches (P_i, P_j) on S and (Q_i, Q_j) on T, and $C_{ij} = P_i \cap P_j$ on S and likewise $D_{ij} = Q_i \cap Q_j$ on T. We have defined above central projections π_i and π_j:

$$\pi_i : P_i \to Q_i; \pi_j : P_j \to Q_j.$$

By construction, these projections are identical on the shared boundary C_{ij} since the boundary corresponds to tangency points between the probe and the spheres S_i.

By the pasting lemma, the continuous maps π_i, which agree pointwise on all overlaps, define a continuous map $\pi : S \to T$ whose restriction to P_i is precisely π_i, that is $\pi_{|P_i} = \pi_i$.

We now use this global map to define an isotopy, that is a continuous map

$$H : S \times [0, 1] \to \mathbb{R}^d \tag{7}$$

such that for every t, the map $H_t : x \mapsto H(x, t)$ is an embedding, namely a homeomorphism from S onto its image $H_t(S)$. Using the global map π, define

$$\forall x \in S : H(x, t) = (1 - t)x + t\pi(x), \forall t \in [0, 1]. \tag{8}$$

We note that the following holds

$$\forall x \in C_{ij} : (1 - t)x + t\pi_i(x) = (1 - t)x + t\pi_j(x), \forall t \in [0, 1]. \tag{9}$$

We conclude with the following

Theorem. 2. *The mapping of Eq. 8 is an isotopy, that is the surfaces $\mathcal{T}_r$ and SES$_{r_s}$ are isotopic.*

Proof.

The map $H(x,t)$ which uses the global continuous map π is jointly continuous in x and t.

For $x \neq y$ in S, the segments $[x, \pi(x)]$ and $[y, \pi(y)]$ defined from central projections associated with cells V_i and W_i are disjoint, and since $H_t(x)$ and $H_t(y)$ belong to these respective segments, they are disjoint. Thus, H_t is injective for every $t \in [0,1]$.

Because S is compact and $\mathbb{R}^2$ is Hausdorff, each continuous injection H_t is an embedding.

We have that $H_0 = \mathrm{Id}_S$; and also $H_1(S) = \pi(S) = \cup_i \pi_i(P_i) = \cup_i Q_i = T$.

Therefore, H is an isotopy from S to T. □

Step 2: isotopy between SES_{r_s} and SAS_{r_w}. To finalize the construction, we need to build an isotopy between SES_{r_s} and SAS_{r_w}. These surfaces differ since the former has been smoothed by replacing two circle arcs by a re-entrant arc (Fig. S2(A)). The two circle arcs excised meet at the intersection point between the original spheres S_i. The re-entrant arc can be split into two sub-arcs by computing the intersection with the Apollonius bisector.

These points can be used to build an explicit isotopy between the re-entrant arc and the two circle arcs excised.

B.4 THE 3D CASE

2D versus 3D. The 3D case also requires manipulating the three surfaces $\mathcal{T}_r$, SAS_{r_w} and SES_{r_s}. The main difficulty is to construct the partitions of $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\mathrm{SES}_{r_s})$, together with cells within which the central projections are defined.

In the sequel, we only present the construction of the decompositions of $\mathcal{T}_r$ and SES_{r_s} for singular triangles – the generalization of Fig. S2, leaving the general case to the reader.

The key players of the construction are presented in Fig. S4.

Decompositions $\mathcal{T}_r$ and SES_{r_s} for singular triangles. The r-offset $\mathcal{T}_r((x_i, x_j, x_k))$ of a triangle involves spherical caps, cylindrical caps, and triangles (Fig. S5(Left)). (NB: the spherical caps are denoted with a superscript (r), namely $\mathrm{Cap}_i^{(r)}$, to distinguish from those found on the SES_{r_s}.)

The solvent excluded surface SES_r involves the following cells:

- Spherical caps Cap_i from the spheres S_i;
- Toroidal patches Torus_{ij} contributed by a probe in contact with two enlarged atoms S_i and S_j.
- Re-entrant spherical caps ReCap_{ijk} contributed by a probe in contact with three enlarged atoms S_i, S_j and S_k.

The difficulty is to elucidate the structure of the cell complexes $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\mathrm{SES}_{r_s})$, and the correspondence between the cells.

With a focus on singular triangles, consider the following partition $\mathcal{X}(\mathcal{T}_r)$ (Fig. S5):

- Vertex x_i: a spherical digon Dig_i.
- Edge (x_i, x_j): a cylindrical part Cyl_{ij}.
  The decomposition of this cylindrical part in $\mathcal{X}(\mathcal{T}_r)$ will read as:

$$\mathrm{Cyl}_{ij} = \mathrm{Cyl}_{ij}^{(1)} \cup \mathrm{Cyl}_{ij}^{(0)} \cup \mathrm{Cyl}_{ji}^{(0)} \tag{10}$$

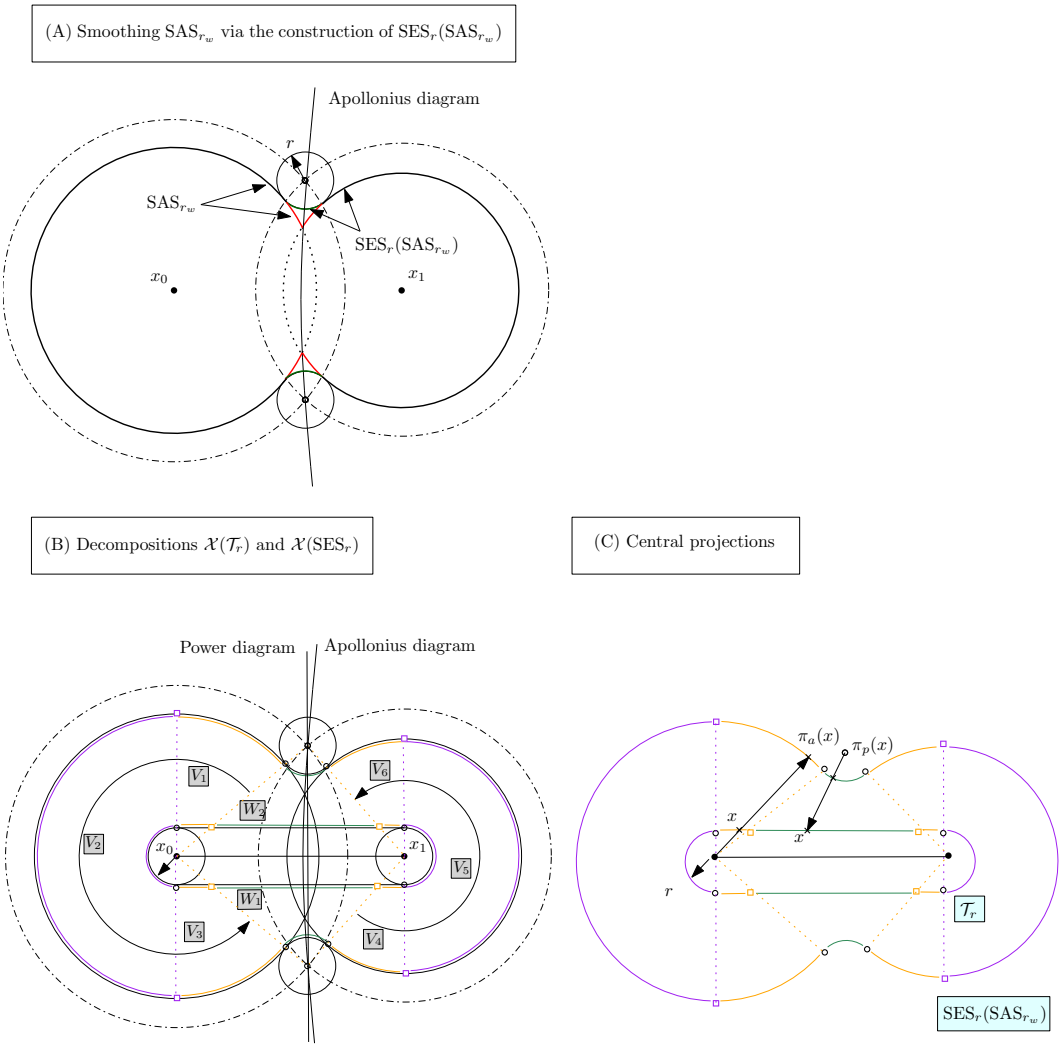


Figure S2: **Constructing the decompositions $\mathcal{X}(\mathcal{T}_r)$ for the proper offset and $\mathcal{X}(\mathbf{SES}_{r_s})$ for the SES in 2D: method illustrated with two balls centered at x_0 and x_1 whose convex hull defines a singular 1-simplex (x_0, x_1).** **(A)** The solid balls represent the balls $S_i(x_i, R_i + r_w)$ whose boundary defines the solvent accessible surface SAS_{r_w}. The probes of radius r are used to smooth SAS_{r_w}, yielding $\mathrm{SES}_r(\mathrm{SAS}_{r_w})$. **(B)** The probes are also used to define $\mathcal{T}_r$. The circular sweeps $\mathrm{CircSweep}_i(A_i)$ applied to each atom are indicated by the black circle arcs with a forward arrow. These result in the creation of the prismatic cells V_i and W_i. **(C)** The central projections π_a and π_p, respectively defined from an atom center and a probe center.

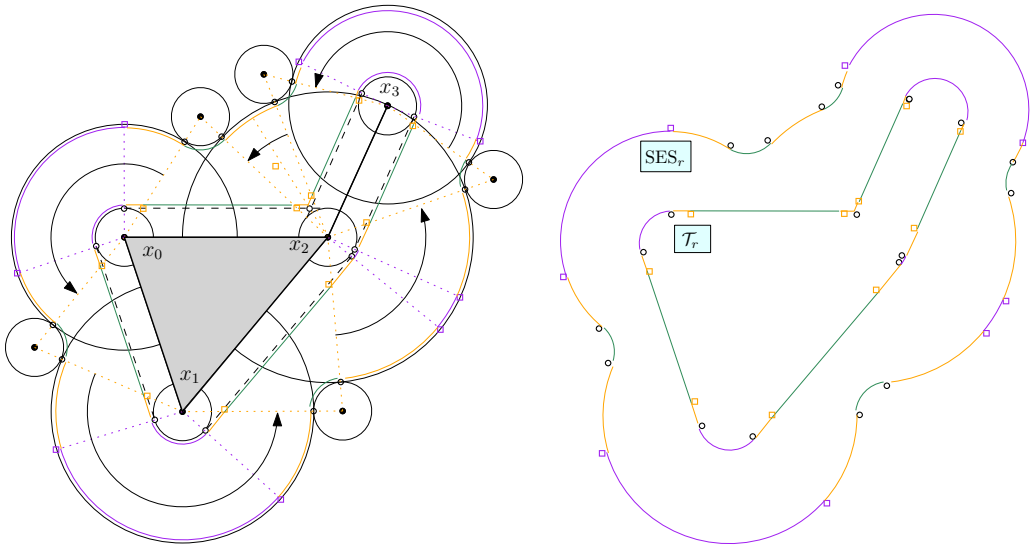


Figure S3: **Constructing the decompositions $\mathcal{X}(\mathcal{T}_r)$ for the offset and $\mathcal{X}(\mathbf{SES}_{r_s})$ for the SES: example.** In this example, the complex K coding SAS_{r_w} consists of the three edges bounding the triangle – which are regular edges in the α-complex, and the singular edge (x_2, x_3). Note that the angle $\angle x_0x_2x_3$ is concave, so that $\mathcal{T}_r$ does not have a spherical cap. The 1-to-1 correspondence between cells of $\mathcal{T}_r$ and cells SES_{r_s} is suggested by the color code.

- Triangle (x_i, x_j, x_k): a triangle Δ_{ijk}.
  The decomposition of this triangle in $\mathcal{X}(\mathcal{T}_r)$ will read as

$$\begin{aligned}\Delta_{ijk} = \Delta^{(2)}_{ijk} \cup \Delta^{(1)}_{ij} \cup \Delta^{(0)}_{ij} \cup \Delta^{(0)}_{ji} \\ \cup \Delta^{(1)}_{jk} \cup \Delta^{(0)}_{jk} \cup \Delta^{(0)}_{kj} \\ \cup \Delta^{(1)}_{ik} \cup \Delta^{(0)}_{ik} \cup \Delta^{(0)}_{ki}\end{aligned} \tag{11}$$

Also consider the following partition of SES_{r_s} as follows–to be elucidated below:

- Spherical caps Cap_i. Decomposed as

$$\mathrm{Cap}_i = \mathrm{Cap}_i{}^- \cup \mathrm{Cap}_i{}^+ \tag{12}$$

- Torus stripes Torus_{ij}
- Re-entrant caps ReCap_{ijk}

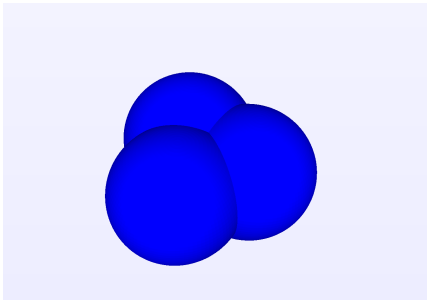
The correspondence between these regions, which is obtained via projections similar to those used in 2D, is presented on Fig. S5:

- (Fig. S5(A)) First, one intersects the triangles defined by the probe centers (two of them for a singular triangle) and the edges of the triangle (x_i, x_j, x_k) with $\mathcal{T}_r$. Note that the

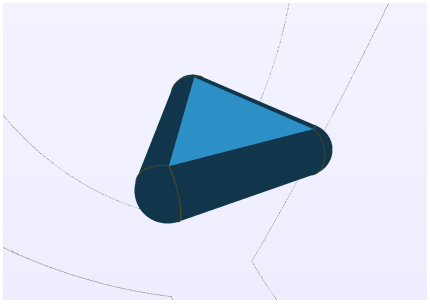
intersection between this plane and the three spheres $B(c, r)$, S_i and S_j defines three great circles on these spheres, so that the section is exactly that from Fig. S2.

- (Fig. S5(B)) The previous projection induces a partition of $\mathcal{T}_r$ specified by Eq. 11 for triangles and Eq. 10 for cylinders.
- (Fig. S5(C,D)) These regions are in one-to-one correspondence with regions of the surface SES_{r_s}.

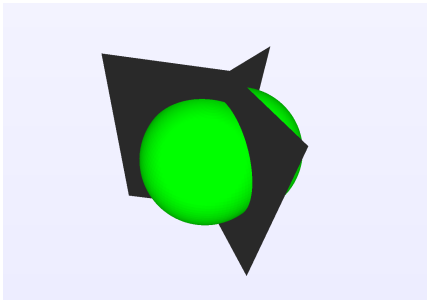
As in the 2D case, the general construction mimics that of the boundary of the union from the α-complex (Akkiraju & Edelsbrunner, 1996). And similarly to the 2D case, the resulting decompositions of $\mathcal{X}(\mathcal{T}_r)$ and $\mathcal{X}(\text{SES}_{r_s})$ can be used to define an explicit isotopy. The very details are omitted from this version.



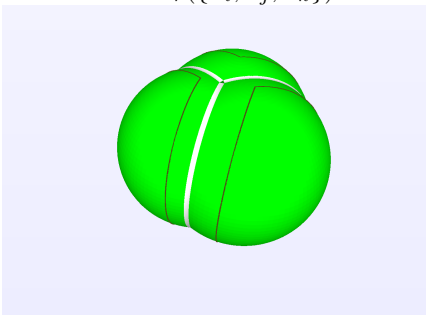
VdW surface $\partial \cup_i A_i$



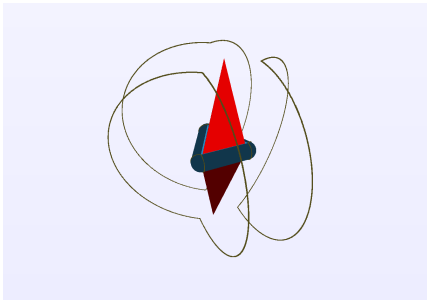
Offset $\mathcal{T}_r(\{x_i, x_j, x_k\})$.



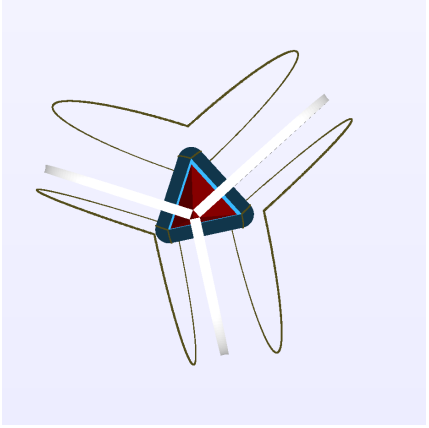
SAS_{r_w} and its radical flats



SES (Connolly) of the SAS *i.e.* $\text{SES}_r(\text{SAS}_{r_w})$



Triangles $(c_{\text{top}}, x_i, x_{i+1})$



ditto

Figure S4: **Partitioning the region between $\mathcal{T}_r$ and SES_{r_s}: key constructions illustrated for the particular case of an acute singular triangle.**

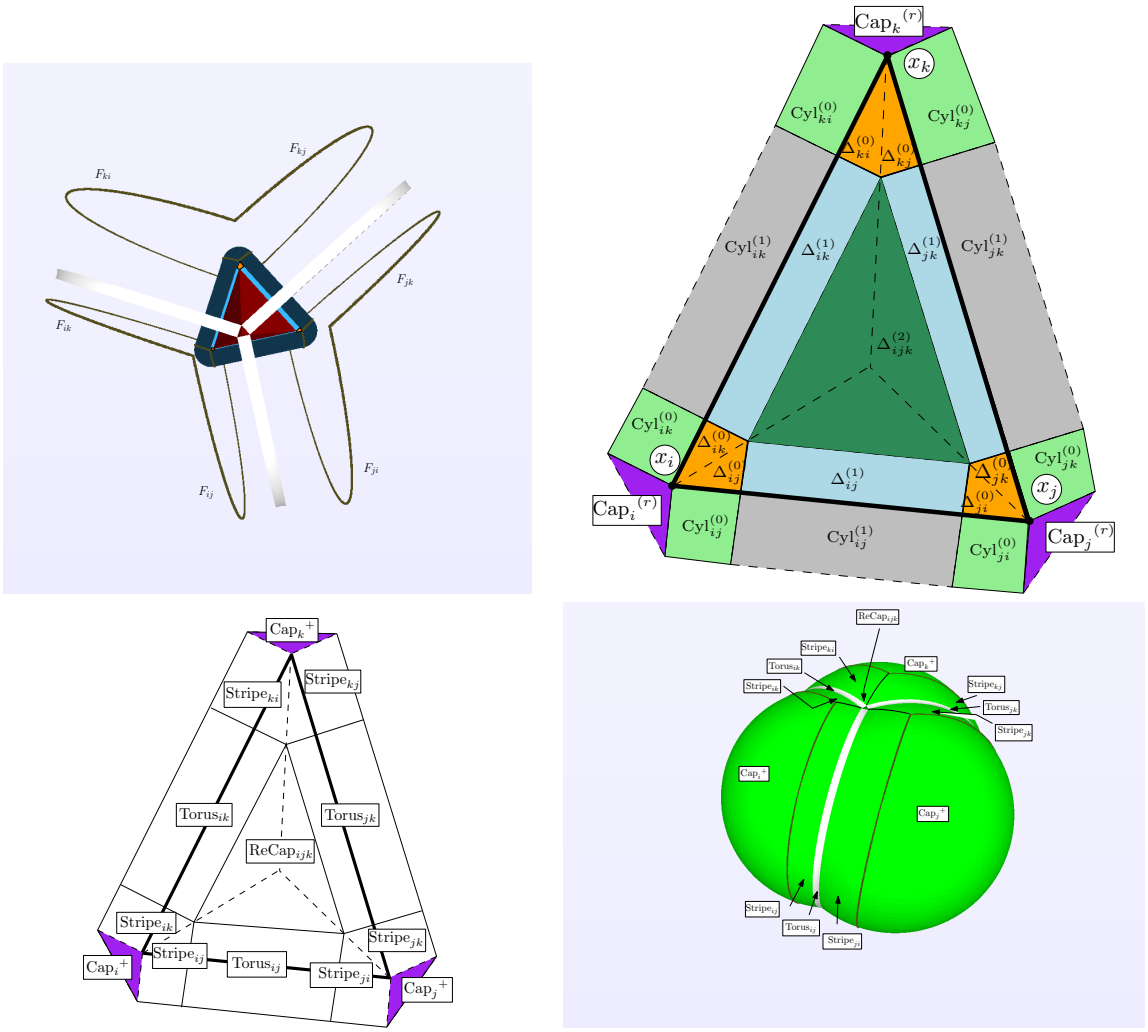


Figure S5: **Decomposition of $\mathcal{T}_r((x_i, x_j, x_k))$.** Color codes identical to those in Fig. S2. **(A)** Top view of the central projection from the probe center c through the atomic centers. The triangles (c, x_i, x_j) are displayed in red/burgundy. Note the cyan/blue and orange regions on the triangular region of the offset. The outer curves F_{ab} are the central projections on the spheres S_i of the curves bounding the digons on $\mathcal{T}_r$. **(B)** Decomposition $\mathcal{X}(\mathcal{T}_r)$ of the offset $\mathcal{T}_r$. See also Eq. 10 and Eq. 11. **(C)** Mapping between regions of $\mathcal{X}(\mathcal{T}_r)$ to those of $\mathcal{X}(\mathrm{SES}_{r_s})$. **(D)** Geometric realization of the regions of $\mathcal{X}(\mathrm{SES}_{r_s})$.

C Boundary Degeneracies

Let $\mathcal{K}$ be a simplicial complex, and let $\sigma \in \mathcal{K}$ be a simplex.

(i) The **star** of σ in $\mathcal{K}$, denoted $\mathrm{St}(\sigma, \mathcal{K})$, is the collection of all cofaces containing σ:

$$\mathrm{St}(\sigma, \mathcal{K}) := \{\tau \in \mathcal{K} \mid \sigma \subseteq \tau\}. \tag{13}$$

Its topological closure in $\mathcal{K}$ is the **closed star**, denoted $\overline{\mathrm{St}}(\sigma, \mathcal{K})$, defined as the smallest subcomplex containing all simplices of $\mathrm{St}(\sigma, \mathcal{K})$ and their faces.

(ii) The **link** of σ in $\mathcal{K}$, denoted $\mathrm{Lk}(\sigma, \mathcal{K})$, is the subcomplex consisting of all simplices in the closed star that are disjoint from σ:

$$\mathrm{Lk}(\sigma, \mathcal{K}) := \{\tau \in \overline{\mathrm{St}}(\sigma, \mathcal{K}) \mid \tau \cap \sigma = \emptyset\}. \tag{14}$$

Equivalently, $\tau \in \mathrm{Lk}(\sigma, \mathcal{K})$ if and only if $\tau \cap \sigma = \emptyset$ and the join $\sigma * \tau$ is a simplex in $\mathcal{K}$.

For instance, for a 2-dimensional complex, for an edge $e = [u, v] \in \mathcal{M}$, the link $\mathrm{Lk}(e, \mathcal{M})$ is a 0-dimensional complex (a discrete set of vertices) comprising the opposite vertices w for all triangles $[u, v, w] \in \mathcal{M}$ incident to e.

C.1 Exhaustive Analysis of Boundary Degeneracies via Simplicial Links

To establish why the list of topological defects in Section 2.2 is exhaustive, we analyze the local neighborhood of the candidate boundary complex using the combinatorial tools of piecewise-linear (PL) topology (Rourke & Sanderson, 1982; Edelsbrunner, 2001).

Link Criterion for 2-Manifolds. Let $\mathcal{M}$ be a 2-dimensional simplicial subcomplex of $\mathcal{K}_\alpha$ representing the naive geometric boundary. The complex $\mathcal{M}$ is a topological 2-manifold without boundary if and only if the open star of every simplex $\sigma \in \mathcal{M}$ is homeomorphic to an open 2-ball $\mathbb{R}^2$. Equivalently, the simplicial link $\mathrm{Lk}(\sigma, \mathcal{M})$ must satisfy:

$$\mathrm{Lk}(\sigma, \mathcal{M}) \cong S^{1-\dim(\sigma)}. \tag{15}$$

Because any simplex $\sigma \in \mathcal{M}$ has dimension 0, 1, or 2, systematically testing Condition equation 15 across all three dimensions exhausts every possible local configuration:

Dimension 2 (Facets): For any 2-simplex $\tau \in \mathcal{M}$, the link is empty by definition: $\mathrm{Lk}(\tau, \mathcal{M}) = \emptyset \cong S^{-1}$. The relative interior $\mathrm{int}(\tau)$ is homeomorphic to an open disk $\mathbb{R}^2$. Consequently, the interior of a facet is always locally a manifold. The defect of a *singular triangle* does not reside in its interior, but manifests in two ways:

(i) Along its boundary edges, where the absence of adjacent tetrahedra leaves unshared 1-simplices with $|\mathrm{Lk}(e, \mathcal{M})| = 1 \neq 2$ (analyzed below).

(ii) In the limit $r \to 0$, where $\mathcal{T}_r$ consists of two distinct, parallel boundary sheets with opposing outward normals. Collapsing these two sheets onto a single geometric triangle requires facet duplication to preserve manifold orientation.

Dimension 1 (Edges): For an edge $e \in \mathcal{M}$, Condition equation 15 requires $\mathrm{Lk}(e, \mathcal{M}) \cong S^0$, which is the discrete 0-sphere consisting of exactly two points. Combinatorially, $|\mathrm{Lk}(e, \mathcal{M})|$ counts the number of incident boundary triangles. Any deviation violates the manifold criterion:

- **A. Isolated / Dangling edges ($|\mathrm{Lk}(e, \mathcal{M})| = 0$):** The edge has no incident triangles. In $\mathcal{T}_r$, the tubular boundary of radius r around the 1-simplex shrinks into a 1-dimensional segment without 2D area.
- **B. Boundary edges ($|\mathrm{Lk}(e, \mathcal{M})| = 1$):** The edge belongs to exactly one boundary triangle, forming an open slit rather than a closed manifold.
- **C. Branching / Finned edges ($|\mathrm{Lk}(e, \mathcal{M})| \geq 3$):** Three or more surface sheets intersect along e, creating a non-manifold junction line.

Dimension 0 (Vertices): For a vertex $v \in \mathcal{M}$, the criterion demands $\mathrm{Lk}(v, \mathcal{M}) \cong S^1$. Combinatorially, $\mathrm{Lk}(v, \mathcal{M})$ is a 1-dimensional graph whose vertices represent neighbors of v along edges in $\mathcal{M}$, and whose edges represent the incident boundary triangles. A finite graph is homeomorphic to S^1 if and only if it is non-empty, connected, and every vertex has degree 2 in the link:

- **D. Isolated vertices ($\mathrm{Lk}(v, \mathcal{M}) = \emptyset$):** The spherical envelope of radius r around an isolated atom contracts to a point.
- **E. Vertices incident to a problematic edge:** Any vertex in $\mathrm{Lk}(v, \mathcal{M})$ with degree $\neq 2$ corresponds directly to an incident edge with $|\mathrm{Lk}(e, \mathcal{M})| \neq 2$.
- **F. Pinch / Multi-patch vertices:** Every vertex in $\mathrm{Lk}(v, \mathcal{M})$ has degree 2, but the graph is disconnected into $m \geq 2$ cycles ($\bigsqcup_{k=1}^{m} S^1$). Geometrically, the neighborhood of v is a cone over multiple circles. This can happen in two cases: i) multiple solvent-accessible patches of the same atom meet at a common vertex, or ii) an annular neck collapses as $r \to 0$ (splitting a connected component).

In short, we can map those cases to the singular and regular faces of our construction. Singular triangles result from case **B**. Singular and non-manifold edges result from cases **A** and **C**. Problematic vertices originate from case **F**. Case **D** does not happen in proteins and case **E** is handled by handling problematic edges.

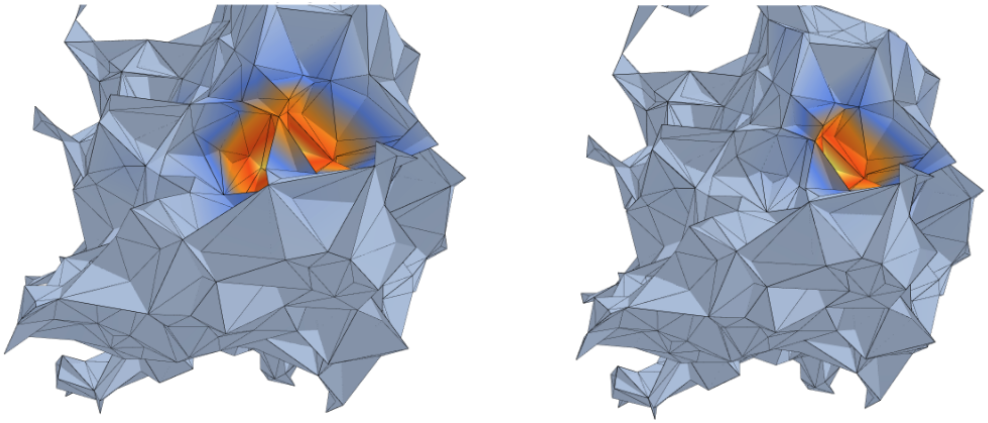
D TUFTING

Tufting construction. An atom is problematic when its sphere contributes more than one disconnected exposed patch to the boundary of the union of balls. Such patches correspond to distinct local surface sheets that are nevertheless represented by the same atom-centered vertex in the surface α-complex. For every problematic atom, we create one coincident vertex copy per exposed patch, remove the original edges and faces incident to that atom, and reconstruct them from the cyclic adjacency of patches across their boundary arcs. Algorithm 2 gives the complete geometric construction.

Algorithm 2 Tufting—geometric version

Require: Surface α-complex $\partial A = (V, E, F)$; boundary of the union of balls $\partial U = (C, \mathcal{A}, P)$, represented by atom centers, boundary arcs, and exposed patches
Ensure: Modified mesh $(\widetilde{V}, \widetilde{E}, \widetilde{F})$
Step 1: Find problematic atoms
$M \leftarrow \emptyset$
for all atoms c on the surface **do**
  if c is covered by more than one exposed patch **then**
    $M \leftarrow M \cup \{c\}$
  end if
end for
Step 2: Split each problematic atom
$V_{\text{new}} \leftarrow \emptyset$
for all $c \in M$ **do**
  for all exposed patches p of c **do**
    Create one new vertex $p.v$ at the center c
    $V_{\text{new}} \leftarrow V_{\text{new}} \cup \{p.v\}$
  end for
end for
$\widetilde{V} \leftarrow (V \setminus M) \cup V_{\text{new}}$
$\widetilde{E} \leftarrow \{e \in E : e \text{ does not touch } M\}$
$\widetilde{F} \leftarrow \{f \in F : f \text{ does not touch } M\}$
Step 3: Reconnect the mesh
for all $c \in M$ **do**
  for all exposed patches p of c **do**
    for all boundary arcs a of p **do**
      $p' \leftarrow$ patch on the other side of a
      $\widetilde{E} \leftarrow \widetilde{E} \cup \{\{p.v, p'.v\}\}$
      $p'' \leftarrow$ patch across the next boundary arc
      $\widetilde{F} \leftarrow \widetilde{F} \cup \{\{p.v, p'.v, p''.v\}\}$
    end for
  end for
end for
return $(\widetilde{V}, \widetilde{E}, \widetilde{F})$

Figure S6 illustrates the practical effect on a surface operator. Merging the coincident copies connects the local sheets and allows a signal initialized on one sheet to diffuse through their shared vertex. Tufting keeps the sheets combinatorially separate, so the signal stays on the selected sheet.



(a) Merged (non-tufted) connectivity. (b) Tufted connectivity.

Figure S6: Heat diffusion on merged and tufted AlphaSurf connectivity. In the merged mesh (left), the signal crosses between surface sheets through the shared vertex. In the tufted mesh (right), a Dirac signal placed on one vertex copy diffuses only within its associated local sheet.

As seen in this figure, the diffusion process on a tufted vertex prevents the heat from crossing the surface in a non-canonical way. Moreover, singular faces that represent 4‰ of the overall number of faces per protein, become two faces at 95‰ after the tufting operation.

E VISUALIZATIONS OF THE SURFACES

E.1 VISUALIZATION OF THE α-COMPLEX OF A MOLECULE

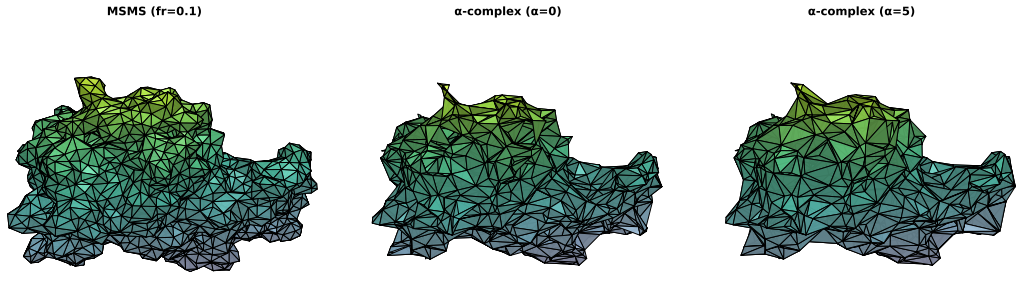


Figure S7: Geometric representations of the 1A27_AB protein complex: a) MSMS coarsened mesh, b) α-complex with $\alpha = 0$, and c) α-complex with $\alpha = 5$.

E.2 VISUALIZATION OF THE DIFFERENT MESHING TECHNIQUES

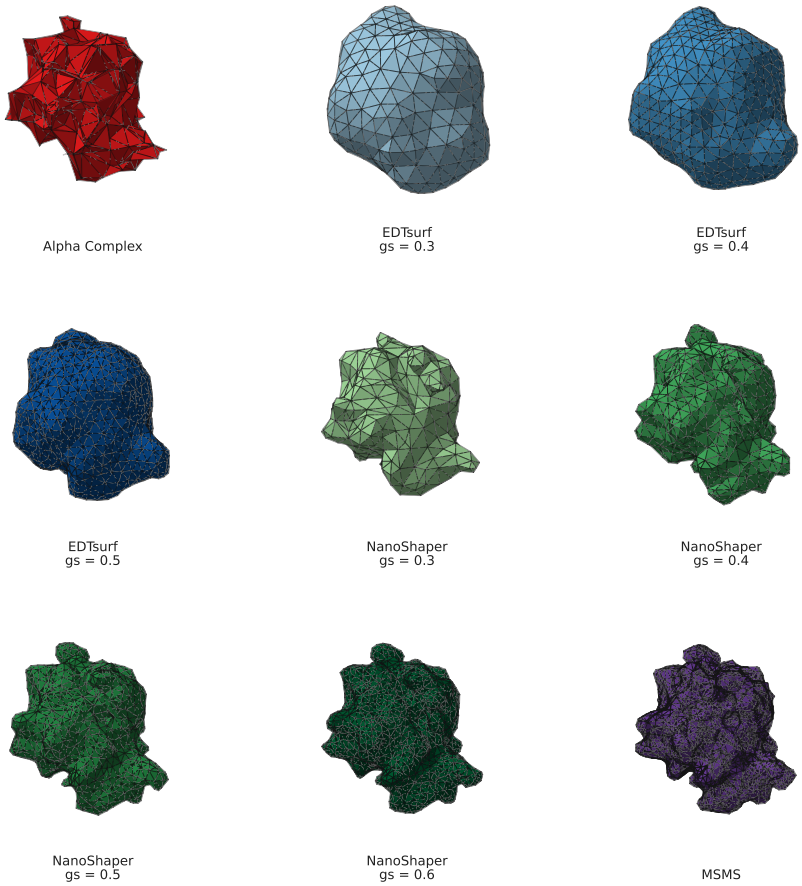


Figure S8: PDB 1A0N, chain B, from the PINDER pair, visualized using different meshing methods.

Taking MSMS as reference, NanoShaper stays visually close to it at all grid scales, whereas EDTSurf appears oversmoothed even at gs = 0.5, losing fine topological features.

E.3 VISUALIZATION OF AN ALPHASURF FRAGMENTATION CASE



Figure S9: Representative AlphaSurf fragmentation case. Removing an edge (in blue) that acts as a connectivity bridge during surface (in red) extraction separates the extracted surface into two connected components.

F DATA FILTERING

F.1 PINDER PREPROCESSING AND CONNECTED-COMPONENT FILTERING

PINDER contains experimentally determined protein–protein complexes organized into sequence-based clusters. To reduce redundancy while retaining structural diversity, we construct our evaluation subset by randomly sampling systems from these clusters.

As a first filtering step, we discard PDB structures represented entirely at the residue level or containing only Cα atoms. Among the remaining structures, 5,807 out of 94,811 PINDER sides, approximately 6%, are split into multiple disconnected atom clouds, which may produce spurious additional components during surface generation. We do not discard these systems, but instead process them to extract the components forming the binding interface.

For each system, we construct a 3 $\mathring{A}$ atom-level neighborhood graph independently for each side and identify its connected components. We define the binding interface as all receptor-ligand heavy-atom pairs within 5 $\mathring{A}$. Component selection is then performed jointly across both sides: for every receptor–ligand component pair, we count the number of interface atom pairs it retains and select the pair maximizing this quantity. This procedure removes disconnected fragments while preserving the pair of components associated with the main binding interface.

G HYPERPARAMETERS

Unless stated otherwise, all experiments use the same AtomSurf backbone: three 128-dimensional ProNet/GVP blocks coupled to DiffusionNet, with bidirectional graph-surface communication. Surface-generation methods and resolutions are varied independently. Only MaSIF-Ligand and S3F uses ESM-2 650M embeddings as an additional residue-level input.

MaSIF-Ligand.

- **Data:** 1,972/238/489 pockets from 1,036/115/287 proteins (train/val/test).
- **Patches:** radius 6 Å, grown to $\geq$100 vertices.
- **Head/loss:** MLP $128 \to 128 \to 7$, dropout 0.25, batch norm, SiLU; cross-entropy.
- **Optimization:** Adam, lr 10^{-4}, ReduceLROnPlateau, 300 epochs, batch 8, 8 workers.
- **Selection/seeds:** val balanced accuracy; seeds 2024, 2025, 2026.

PINDER-Pair.

- **Data:** 42,220/1,958 train/val systems; test holo/apo/AF2 1,955/342/1,747, reduced to 1,835/309/1,582 (valid meshes for all methods).
- **Geometry:** interface thresholds of 5 Å between residues for graph labels. Surface labels use 2.0 Åbetween vertices of SES-based meshes (MSMS, EDTSurf, NanoShaper) and 3.8 Å for AlphaSurf, whose vertices are atom centers; the latter was chosen to match the label rate of the SES meshes.
- **Head/loss:** graph + surface pair heads, MLP $256 \to 128 \to 1$ on concatenated partner embeddings, dropout 0.1; focal loss ($\gamma = 2$), equal graph/surface weights, 10:1 surface neg:pos sampling. AUROC from graph head only.
- **Optimization:** Adam, lr 5×10^{-4}, cosine (3 warm-up epochs), 30 epochs, adaptive batching ($\leq$30,000 atoms per batch), 20 workers.
- **Selection/seeds:** val AUROC; seeds 2024, 2025, 2026.

MISATO binding-site prediction.

- **Data:** 13,765/1,595/1,612 complexes (train/val/test); Cα positive if $\leq$10 Å from a ligand atom.
- **Frames:** random MD frame (train), frame 0 (val/test); labels from reference frame.
- **Head/loss:** residue MLP $128 \to 128 \to 2$, ReLU, dropout 0.1; unweighted two-class cross-entropy.
- **Optimization:** Adam, lr 10^{-3}, ReduceLROnPlateau, 200 epochs, batch 16, 16 workers.
- **Selection/seeds:** val AUPRC; seeds 2024, 2025, 2026.

Noising on-the-fly.

- **Noise:** Gaussian coordinate noise on graph/surface inputs, training loader only.
- **Labels:** interface pairs recomputed from noised coordinates and vertices.
- **Evaluation:** clean val/test inputs.
- **Optimization:** as PINDER-Pair.

S3F pretraining and ProteinGym.

- **Data:** 31,885 CATH v4.3 S40 domains; S3F seed-0 split $0.97/0.02/0.01$ (30,928/637/320 train/val/test).
- **Input/masking:** random crop at 250 residues; 15% masking (80/10/10 mask/random/unchanged);
- **Model:** S3F architecture: frozen ESM-2 650M, GVP-GNN with $(256, 16)$ node / $(64, 1)$ edge dims, 5 structure + 5 surface GVP layers, dropout 0.1, fusion over 20 nearest surface points.
- **Head/loss:** dropout + linear $256 \to 20$, dropout 0.5; cross-entropy on masked positions.
- **Optimization:** Adam, lr 2×10^{-4}, 100 epochs, batch 8, no weight decay, no schedule.
- **Selection/seeds:** final epoch-100 checkpoint; seed 2024.
- **ProteinGym:** zero-shot, 217 DMS assays, one AF2 structure per assay; ESM-2 fallback for sites with pLDDT $<$70 (as S3F); Spearman averaged per UniProt, per function category, then across 5 categories.

H Additional Results

H.1 Heat Kernel

Sample points. We sample 1000 vertices of the MSMS surface by farthest-point sampling and map each to its nearest vertex on every mesh. A sample is dropped when its alpha-complex vertex is duplicated by tufting, or when it lands on the same vertex as another sample on some mesh, so that every mesh evaluates the kernel at the same distinct points. The heat kernel at fixed t depends on metric scale (Mallet et al., 2024). We therefore rescale each method's meshes by a single factor so that their mean area over the dataset matches that of the MSMS reference; relative sizes across proteins are preserved.

Distance Window. To choose the distance window, we measure how the heat kernel at $t = 10$ decays with geodesic distance on the MSMS surface (Figure S10). For each pair of sampled points, $K_t(i, j)$ is divided by its value at the source and averaged over proteins per distance bin. The kernel falls to 50% at about 6 Å and below 1% beyond 15 Å; we therefore compare pairs 0 to 15 Å apart.

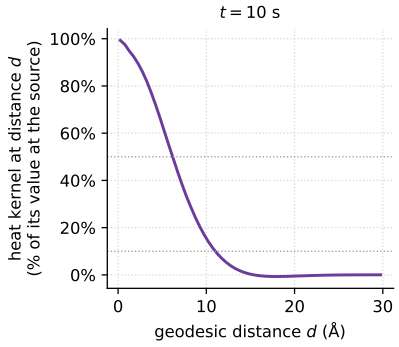


Figure S10: Heat kernel at $t = 10$ against geodesic distance on the MSMS surface, relative to its value at the source, mean over 1000 proteins.

H.2 Distributions

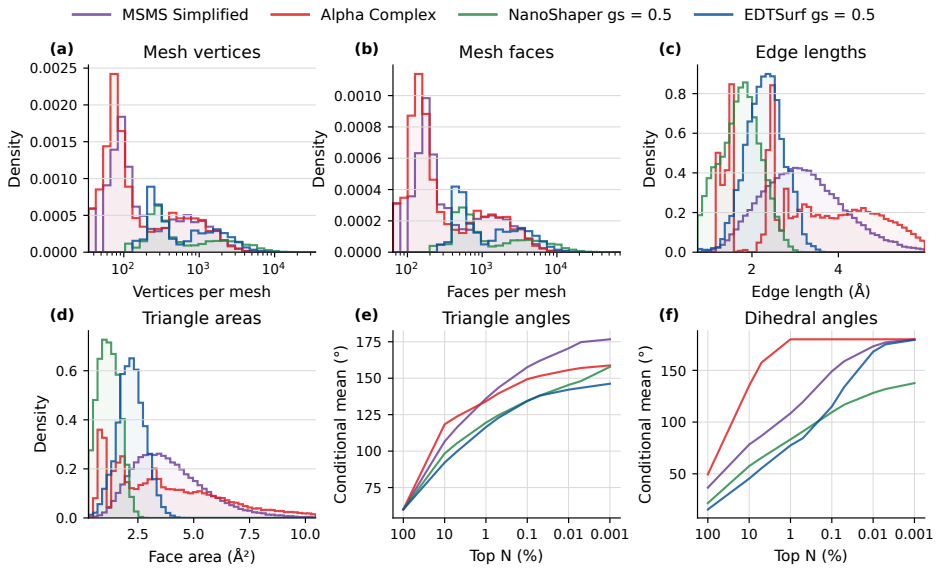


Figure S11: Distributions of mesh statistics across surface-generation methods.

Figure S11 further compares the geometry of the generated meshes. The Alpha Complex produces relatively small meshes, with vertex and face counts concentrated near the lower end of the distributions, while retaining a long tail for larger proteins. Compared with NanoShaper and EDTSurf, it exhibits broader edge-length and triangle-area distributions. Its triangle-angle distribution is broadly comparable to NanoShaper and EDTSurf, whereas its dihedral-angle tail is more pronounced, reaching large angles earlier than all the others, this might be explained by the very coarse mesh, an issue that MSMS simplified does not suffer from the used decimation technique minimizes the dihedral angles. Overall, the Alpha Complex favors compact meshes at the cost of somewhat less regular local triangulations.

H.3 PINDER Surface validity

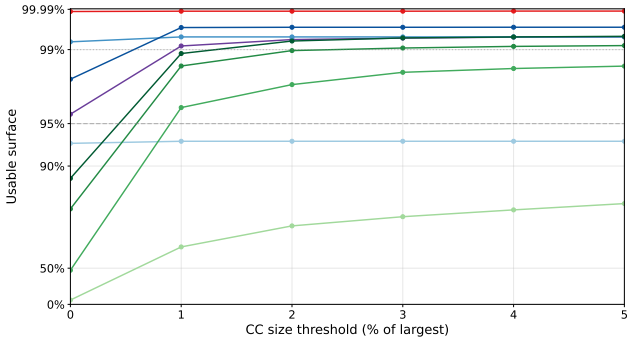


Figure S12: Percentage of the PINDER proteins with a usable surface mesh as a function of the connected-component size threshold. Results are shown for Alpha Complex, MSMS, EDTSurf, and NanoShaper at the indicated grid scales.

H.4 MaSIF-Ligand surface validity

We repeat the validity and fragmentation analysis from Section 2.4 on the MaSIF-Ligand pockets. We use the same failure definition, connected-component criterion, and relative fragmentation threshold so that the behavior of each surface-generation method can be compared consistently across the two datasets.

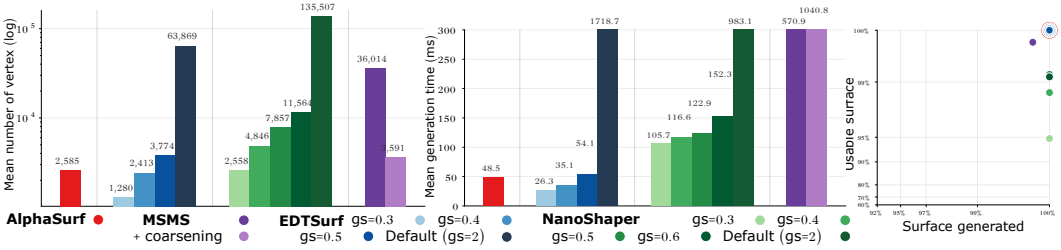


Figure S13: Efficiency, surface size, and usability of the different molecular surface-generation methods on the MaSIF Ligand dataset

In this dataset, proteins are significantly bigger, hence all methods have more vertices and slower generation time. This makes uncoarsened NanoShaper and MSMS meshes even more prohibitive. Moreover, contrary to the Pinder dataset, MaSIF-Ligand exclusively contains globular proteins. This explains the gap in validity, notably for EDTSurf, between the two datasets.

H.5 Preprocessing timings

To isolate the computational cost of each surface representation, we profile the preprocessing pipeline on our datasets. Proteins are processed sequentially using a single data-loading process and no additional worker processes. For triangulated representations, we separately measure the time required for surface generation, mesh processing and simplification, spectral-operator construction,

and geometric-feature computation. Processing times are measured using an NVIDIA H100 GPU and its associated Intel Xeon Platinum 8468 CPU resources.

Table S1: Average preprocessing time per MaSIF–Ligand protein (on the full protein, not the pocket-extracted mesh), broken down by pipeline stage. Times are reported in milliseconds.

Pipeline stage	**α-complex**	**EDTSurf**			**NanoShaper**		**MSMS Simplified**
Grid scale	–	0.3	0.4	0.5	0.4	0.5	–
Surface / point-cloud generation	48.5 ± 38.4	26.3 ± 44.2	35.1 ± 21.9	54.1 ± 36.2	116.6 ± 86.1	122.9 ± 91.2	570.9 ± 583.1
Mesh processing and simplification	25.1 ± 26.1	14.3 ± 20.4	25.0 ± 23.7	39.7 ± 36.0	51.7 ± 47.2	83.1 ± 71.4	469.9 ± 397.9
Spectral-operator construction	391.2 ± 254.8	189.5 ± 106.6	361.5 ± 209.8	564.4 ± 331.6	705.9 ± 514.9	1154.2 ± 911.0	503.6 ± 339.9
Geometric-feature computation	60.5 ± 43.8	25.5 ± 13.8	48.5 ± 28.0	77.9 ± 45.8	107.9 ± 73.2	170.7 ± 115.3	78.6 ± 76.7
Total preprocessing	525.3	255.6	470.1	736.1	982.1	1530.9	1623.0

Table S2: Average preprocessing time per PINDER protein, broken down by pipeline stage. Times are reported in milliseconds.

Pipeline stage	**α-complex**	**EDTSurf**			**NanoShaper**		**MSMS Simplified**
Grid scale	–	0.3	0.4	0.5	0.4	0.5	–
Surface / point-cloud generation	27.1 ± 34.2	22.9 ± 25.7	29.0 ± 30.8	48.3 ± 56.8	70.9 ± 446.4	78.0 ± 100.9	122.0 ± 175.2
Mesh processing and simplification	12.4 ± 18.8	8.9 ± 12.8	12.9 ± 16.8	23.6 ± 28.6	26.3 ± 34.1	39.4 ± 48.9	136.0 ± 151.6
Spectral-operator construction	184.6 ± 188.1	127.6 ± 97.5	185.0 ± 163.8	343.2 ± 312.5	375.5 ± 405.2	552.8 ± 584.9	192.7 ± 192.4
Geometric-feature computation	28.5 ± 32.5	16.7 ± 13.4	24.8 ± 22.5	47.9 ± 42.7	56.8 ± 60.0	81.1 ± 83.6	28.5 ± 29.3
Total preprocessing	252.6	176.1	251.7	463.0	529.5	751.3	479.2

H.6 FORWARD–BACKWARD TIMINGS

In addition to preprocessing cost, we measure the forward-backward pass time for each surface representation. Timings are recorded during training on a single NVIDIA H100 GPU, averaged per protein over the full epoch, and reported separately for the MaSIF-Ligand and PINDER datasets. Those are then used for the on-the-fly vertical lines in Figure 6.

Table S3: Forward–backward training time per MaSIF–Ligand protein.

Measurement	**Alpha-complex**	**EDTSurf**		**NanoShaper**	**MSMS**
Grid scale	–	0.3	0.4	0.4	–
Time per protein (s)	0.0299	0.0308	0.0328	0.0316	0.0299
Mean			**0.0310**		

Table S4: Forward–backward training time per PINDER protein.

Measurement	**Alpha-complex**	**EDTSurf**			**NanoShaper**		**MSMS**
Grid scale	–	0.3	0.4	0.5	0.4	0.5	–
Time per protein (s)	0.0322	0.0320	0.0340	0.0350	0.0350	0.0390	0.0320
Mean				**0.0342**			

H.7 TUFTING ABLATION

We evaluate the impact of tufting on the benchmarks.

	Holo	Apo	AF2
No tufting	$92.02 \pm 0.09\%$	$86.30 \pm 0.17\%$	$\mathbf{85.54 \pm 0.47}\%$
Tufting	$\mathbf{92.12 \pm 0.18}\%$	$\mathbf{86.40 \pm 0.13}\%$	$85.46 \pm 0.22\%$

Table S5: PINDER-Pair AUROC with and without surface tufting.

PINDER-Pair. Surface tufting yields negligible differences in AUROC across all three settings (Holo, Apo, AF2), with changes well within one standard deviation of the corresponding baseline. We therefore conclude that tufting has no meaningful impact on PINDER-Pair prediction performance.

Metric	**No tufting**	**Tufting**
F1	$48.88 \pm 0.39\%$	$\mathbf{50.30 \pm 0.20}\%$
AUROC	$86.17 \pm 0.53\%$	$\mathbf{86.80 \pm 0.10}\%$
AUPRC	$48.00 \pm 0.58\%$	$\mathbf{50.40 \pm 0.50}\%$

Table S6: MISATO binding-site prediction with and without tufting.

MISATO binding-site prediction. In contrast to PINDER-Pair, surface tufting yields a consistent and significant improvement on MISATO binding-site prediction, with gains across all three metrics that exceed one standard deviation of the untufted baseline.