IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 16, 2020, accepted September 21, 2020, date of publication September 25, 2020, date of current version October 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3026757

Molecular Surface Estimation by Geometric Coupled Distance Functions

YUANHAO GONG[1] AND YONG CHEN[2]

[1]College of Electronics and Information Engineering, Shenzhen University, Shenzhen 518060, China
[2]School of Energy and Power Engineering, Wuhan University of Technology, Wuhan 430070, China

Corresponding author: Yong Chen (yongchen@whut.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61907031.

ABSTRACT Estimating the surface from given atoms with location and size information is a fundamental task in many fields, such as molecular dynamics and protein analysis. In this paper, we present a novel method for such surface estimation. Our method is based on level set representations, which can efficiently handle complex geometries. The proposed method is analyzed from mathematical point of view and from computation point of view. The method does not require any prior information about the surface. This property is fundamentally important for the surface estimation task. The presented method is evaluated on both synthetic and real data. Several numerical experiments confirm that our method is effective and computationally efficient. Finally, the method is applied on protein surface estimation. This method is suitable for high performance molecular dynamics study, protein surface analysis, etc.

INDEX TERMS Molecule, surface, level set, mean curvature.

I. INTRODUCTION

Given the location and size information of some atoms, it is important to know the geometric surface that compactly contain these atoms. Such geometric surface is important for its function, such as protein docking and Molecular Hydrophobicity Potential. Therefore, surface estimation from such atoms becomes fundamentally important for related research fields. One example is shown in Fig. 1, where the surface is estimated by our method and the color indicates its mean curvature property.

Before showing the molecular surface estimation, we first introduce a surface reconstruction method from point cloud. Different from the molecular surface, these points exactly live on the surface and do not have a size or radius. We show this method and its accuracy on both synthetic and real data. Then, we extend this method for molecular surface estimation in Section V.

A. POINT CLOUD REPRESENTATION

A point cloud $P = \{\vec{x}_i \in \mathcal{S}\}$ is a fundamental representation of a surface $\mathcal{S}$. If an object is small compared to $\mathcal{S}$, it can be treated as a point. For example, nuclei are treated as points to represent cells in the tissue; a small region is treated as a point during tissue development; a protein is treated as a point on the cell membrane. All points together can represent the geometry where they live on. This idea has been used in super resolution microscopy techniques such as Photo-Activated Localization Microscopy (PALM, invented by Eric Betzig who won Nobel prize because of PALM in 2014) and Stochastic Optical Reconstruction Microscopy (STORM).

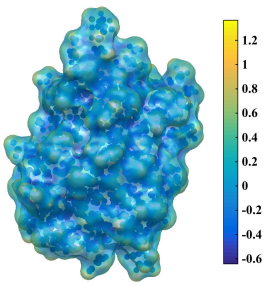


FIGURE 1. Protein surface estimation by our method (UBQ protein surface). The color on the surface indicates the mean curvature value of the surface.

However, surface reconstruction from unstructured point clouds is challenging due to the absence of connectivity information between the points, which may lead to topological ambiguities. Even if there is a model to choose

The associate editor coordinating the review of this manuscript and approving it for publication was Jenny Mahoney.

the topology according to the sample's geometry, this requires prior knowledge about the sample. Moreover, high curvature (sharp corners or edges), irregular sampling, and noise in the point positions often complicate the task.

Conceptually, there are two approaches to recovering the surface $\mathcal{S}$ from which the points have been sampled: interpolation (find missing data) and model fitting (reduce error). Traditionally, both approaches require predefined basis functions that ideally reflect geometric properties of the true surface, such as connectivity, smoothness, sparsity, and curvature. These implicitly assumed properties constitute the prior knowledge (regularization) about the unknown geometry. Even though their imposition may render the reconstruction problem well-posed, these priors may mask details or patterns in the signal, such as smoothing out texture, or bias the reconstruction result toward the imposed priors.

Regularization-free geometry reconstruction is desirable, specially for biological data, where prior knowledge is scarce or needs to be investigated. If a prior was imposed, it would be difficult to decide whether a property of resulting geometry results from the prior or from the signal. This prior-freeness requirement makes most of state-of-the-art approaches in surface reconstruction fail in this context. We present a regularization-free method for geometry reconstruction from point clouds to tackle this challenge.

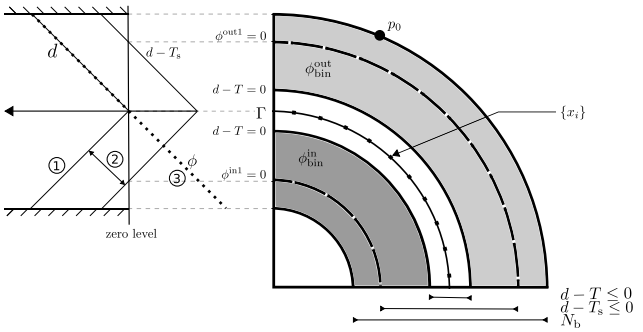
B. PREVIOUS WORKS ON SURFACE RECONSTRUCTION

Numerous methods have been proposed for surface reconstruction from unstructured point clouds. This includes approaches based on FFT (e.g., [1]), Poisson surfaces (e.g., [2]), and moving least squares (e.g., [3]). Most of those methods require that the surface normals have been first estimated from the data. From a differential geometry point of view, normals are first-order information about the geometry, while position is zeroth-order information. However, estimating normals from unstructured point sets is challenging because global consistency across the entire surface must be ensured. Traditionally, normals are estimated using Principal Component Analysis (PCA), a local method that can not guarantee global consistency.

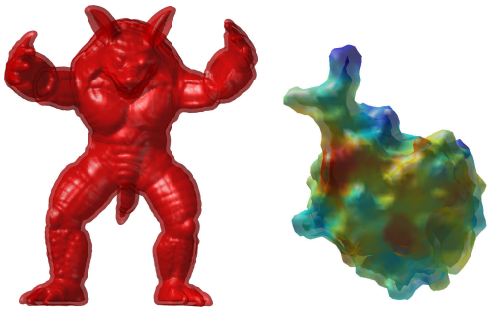
Several methods, for example level-set methods [4], [5], do not require normal information. Instead, they attempt to minimize the energy [5]

$$\min_{\mathcal{S}} \left\{ \mathcal{E}(\mathcal{S}) = \|d(\vec{x})\|_q + \lambda \mathcal{R}(\mathcal{S}) \right\}, \tag{1}$$

where q is a real number, and $d(\cdot)$ is the distance field to the unknown implicit surface $\mathcal{S}$ (see Fig. 2(a)), and λ is the regularization coefficient. In this energy-based approach, methods from image segmentation have also been adapted to surface reconstruction [6]. Thanks to the convexity of the energy, fast solvers (e.g., split-Bregman [7], [8] or Primal/Dual) can be used. The regularization term $\mathcal{R}(\mathcal{S})$ in the model, however, tends to smooth the result and remove details from the surface. Moreover, the computational cost depends on proper initialization and stepsize control.



(a) Geometric situation



(b) Coupled signed-distance fields

FIGURE 2. (a) Illustration of coupled signed distance functions and its notions. (b) The two coupled fields ϕ^{out} and ϕ^{in}: the standard "Armadillo" computer-graphics test model and a protein molecular surface. The protein surface is color coded by the Molecular Hydrophobicity Potential for better visualization. Details are in section IV.

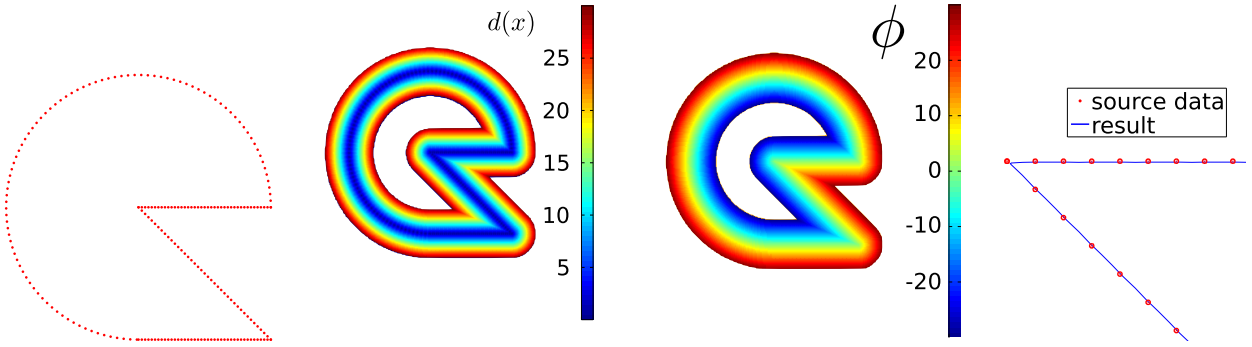
Previous works also demonstrated several strategies to save memory and reduce the computational complexity of level-set algorithms. This includes narrow-band formulations [9], multi-scale methods [10], [11], and DT-grids [12]. The need for computationally expensive level-set re-initialization has been overcome by adding an additional penalty (regularization) term in the energy [13].

II. OUR METHOD

Our work is motivated by the symmetry property of a distance field and the antisymmetry of the level set representation [14]. It is inspired by coupled level-set methods [15]–[17], but addresses some of their shortcomings when reconstructing high-curvature regions, while guaranteeing the signed-distance property. The connection of our method with others can be found in section II-D.

We are given an unordered point set $P = \{\vec{x}_i : \vec{x}_i \in \mathbb{R}^n, i = 1, \ldots, N\}$. Usually, $n = 2$ for image processing problems, such as estimating a contour from feature points or filling gaps between edge fragments, and $n = 3$ for computer graphics problems, such as constructing a surface model from a point cloud, or for stereo vision problems. Even though we focus on two- and three-dimensional problems, the method presented here also works in higher dimensions, for example for constructing surfaces in manifold learning.

For surface reconstruction, we present a coupled Signed Distance Functions (cSDF) method. The key idea in cSDF



(a) Input points (b) Distance $d(\vec{x})$ (c) Result $\phi(\vec{x})$ (d) Sharp corner

FIGURE 3. 2D example with sharp corners and uneven sampling to illustrate the individual steps of the present method.

is to use two spatially coupled signed distance functions ϕ^{in} and ϕ^{out} to capture the surface $\mathcal{S}$. This geometric constraint is in contrast to coupled level-set methods, which are based on topological constrains [18]. The cSDF idea is inspired by the difference between the reflection symmetry of the distance field d and the reflection antisymmetry of the level-set function ϕ.

From a variational point of view, the cSDF method attempts to minimize the energy functional

$$\min_{\mathcal{S}} \left\{ \mathcal{E}(\mathcal{S}) = \int_{\Omega} \left[(\phi^{\text{in}})^2 + (\phi^{\text{out}})^2 + d^2 - \phi_{\max} \right] \mathrm{d}x \right\}, \qquad (2)$$

where $\phi_{\max} = \max\left\{(\phi^{\text{in}})^2, (\phi^{\text{out}})^2, d^2\right\}$. However, as described in the following, we do not need to evolve any Partial Differential Equation (PDE) in order to compute the result. Instead, simple thresholding is enough. The reason for this becomes apparent from the Eikonal equation (Eq. 3), where thresholding is equivalent to wave propagation if the wave speed is constant.

We first illustrate cSDF in 2D and then apply it to synthetic and real-world data in 3D. The method proceeds in three steps, as detailed by Algorithms 1 to 3 below. The corresponding steps are illustrated as in Fig. 2(a). The coupled level set functions ϕ^{out} and ϕ^{in} are illustrated in Fig. 2(b) for two examples. A simple example of the intermediate results after every processing step is shown in Fig. 3.

A. STEP 1: DISTANCE FIELD

For a given point cloud P (exemplified in Fig. 3(a)), we compute the distance field $d(\vec{x})$ on a predefined Cartesian grid $G = m \times n$ of uniform resolution h. This amounts to solving the following Eikonal equation:

$$\begin{cases} \|\nabla d(\vec{x})\| = v(\vec{x}) & \forall \vec{x} \in G \\ \text{s.t. } d(\vec{x}_i) = 0 & \forall \vec{x}_i \in P \end{cases} \qquad (3)$$

as the boundary-value formulation of the Hamilton-Jacobi problem associated with Eq. 2.

Several methods are available to numerically solve this equation, including the Fast Marching Method [9], the Group Marching Method [19], the Fast Sweeping Method (FSM) [20], the Fast Iteration Method [21], and direct Hamilton-Jacobi solvers [22].

Here, we show three alternative methods to compute the distance field. The first one uses an extended-window FSM restricted to a narrow band of width b:

$$N_{\text{b}} = \{\vec{x} \in G : \exists \vec{x}_i \in P \; s.t. \; ||\vec{x}_i - \vec{x}|| < b\} \qquad (4)$$

where Eq. 3 is only solved for $\vec{x} \in N_b$. We ensure the distance property by setting $v(\vec{x}) \equiv 1$. This method was first published in [23]. The second method directly computes the distance field from the point cloud using a Sparse Voxel Oct-tree (SVO) data structure in the narrow band. The third method solves an inhomogeneous Helmholtz equation using FMM [24], similar to how it is done in a Schrödinger distance transform [25].

1) EXTENDED-WINDOW FAST SWEEPING METHOD

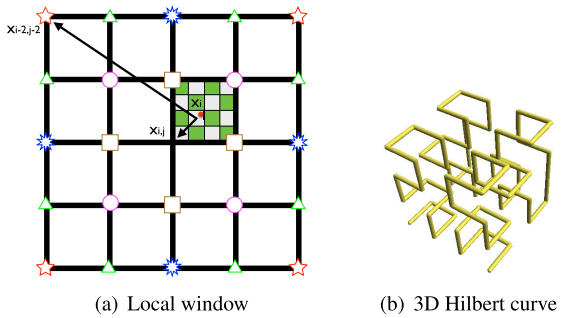
FSM sweeps the grid until convergence, which can be inefficient for points far from the interface. Fast iterative methods relax this by using locks [21]. These locks, however, cause additional serialization. Here, we avoid these locks and accelerate FSM by using a larger window size $w > 1$ (see Algorithm 1). The min method of FSM [20] is hence extended to account for all points in a w-neighborhood, as illustrated in Fig. 4(a) for $w = 2$. We initialize the algorithm with:

$$\begin{cases} d(\vec{x}) = +\infty & \forall \vec{x} \in N_{\text{b}} \setminus P \\ d(\vec{x}_i) = 0 & \forall \vec{x}_i \in P. \end{cases} \qquad (5)$$

The original FSM [20] is recovered for $w = 1$.

For $w > 1$ the present extended-window FSM is more accurate than the original FSM, because the integrated error is reduced when the local window size w gets larger. Moreover, it converges faster since the larger window causes a wave of higher speed (not further elucidated in this paper). The local update cost increases from 2 to $w(w+3)/2$ (not $(2w+1)^2$, thanks to the symmetry property).

In our implementation, we further accelerate initialization and fast sweeping by using a Look-Up Table (LUT) for the distances in the local window. Instead of directly computing the distances of each sample point $\vec{x}_i$ to each grid node $\vec{x}_{i,j}$



(a) Local window (b) 3D Hilbert curve

FIGURE 4. (a) Illustration of a 5 × 5 local window used in the extended-window FSM with $w = 2$. Different symbols mark symmetric points with respect to the center point $\vec{x}_{i,j}$. The green/white checkerboard illustrates the resolution of the distance look-up table that may optionally be used to further accelerate the algorithm (see main text for details). (b) Illustration of a Hilbert curve in 3D to enumerate all voxels of the grid.

within the local window, we subdivide each grid cell into 4 × 4 bins, represented by the green/white checkerboard pattern in Fig. 4(a). We then build a LUT of the distances between each checkerboard bin center and each grid node in the local window. The distance of a sample point to a grid node is then taken to be the distance between the bin center that it resides in and the grid node (black arrows in Fig. 4(a)). This can be used to further accelerate the initialization and sweeping steps as follows:

- Initialization Step: There are two ways to initialize the distance field. The sample point $\vec{x}_i$, represented by a red dot in Fig. 4(a), can be used to directly compute the distance to its all neighbors on the grid. The other way is to use a LUT. As shown in Fig. 4(a), $\vec{x}_i$ must be in one of the grid cell neighboring $\vec{x}_{i,j}$. All distances in the LUT are computed with respect to the center of the checkerboard cell containing $\vec{x}_i$.
- Sweeping Step: We determine $d(\vec{x})$ in the local window using a distance LUT with respect to the center node $\vec{x}_{i,j}$. Thanks to the symmetry property of $d(\vec{x})$, only $w(w+3)/2$ real numbers need to be stored in the LUT. The sweeping process only uses addition and comparison operations, which are fast on modern CPUs.

The computational complexity of FSM is $O(N_b)$, which is less than the $O(N_b \log(N_b))$ of FMM. Fig. 3(b) shows an example $d(\vec{x})$ computed using FSM with $w = 3$.

2) DIRECT COMPUTATION

The distance field can alternatively be directly computed using a SVO data structure [26]. SVO has attracted a lot of attention in the computer-graphics literature recently for its nice properties in ray tracing, which essentially also is a distance computation task.

We use a Hilbert curve (illustrated Fig. 4(b)) to encode the narrow band grid points. Then, the so-encoded narrow band is organized in an oct-tree, which is constructed bottom-up. The distance is then directly determined by a nearest-neighbor search in the tree. For more details, we refer to Ref. [26].

3) HELMHOLTZ EQUATION

We can embed the distance field $d(\vec{x})$ into a homogeneous Helmholtz equation:

$$\begin{cases} \tau^2 \Delta\psi(\vec{x}) - \psi(\vec{x}) = 0 \\ \psi(\vec{x}_i) = 1, \end{cases} \tag{6}$$

where $\psi(\vec{x}) = \exp(d(\vec{x})/\tau)$, $\delta \leq \tau < 0$ and δ is a negative number close to zero. Δ is the Laplace operator. The solution of this Helmholtz equation automatically also satisfies the Eikonal Eq. 3. For $\tau \to 0(\tau < 0)$,

$$\tau \Delta d(\vec{x}) + \|\nabla d(\vec{x})\|^2 - 1 = 0 \to \|\nabla d(\vec{x})\|^2 = 1. \tag{7}$$

After solving Eq. 6, the distance field can be recovered as:

$$d(\vec{x}) = \tau \log(\psi). \tag{8}$$

The advantage of this method is that there are several very efficient solvers for Eq. 6, such as FFT(DCT, DST)-based solvers, Multigrid solvers, and Fast Multipole Methods [24]. Here, we use a DCT-based solver in order to directly impose homogeneous Neumann boundary conditions, i.e., the gradient of ψ is zero in the normal direction at the band edge.

B. STEP 2: COUPLED SIGNED-DISTANCE FUNCTIONS

We aim to compute $\phi(\vec{x})$, the signed-distance function associated with $d(\vec{x})$. The key idea in cSDF is to apply distance-preserving shift transformations to the output of Algorithm 1, thus solving the boundary-value problem in Eq. 3 without (pseudo-)time evolution. Specifically, we shift d by an offset T in order to determine the functions $\phi^{\text{in}}_{\text{bin}}$ and $\phi^{\text{out}}_{\text{bin}}$ that indicate whether the shifted level set $d - T$ is inside or outside of $\mathcal{S}$ (see shaded areas in Fig. 2(a)). The threshold T defines the separation between the regions to be labeled. Therefore, $T > \sqrt{3}h$.

Algorithm 1 Extended-Window Fast Sweeping Method in 2D

1: **INPUT:** threshold tol, window size w, S, U, V
2: set $w^{+1} = w + 1$
3: initialize $d_k(\vec{x})$ using Eq. 5
4: define the loop sets
$\{(i,j) : i = w^{+1} \dots m - w, j = w^{+1} \dots n - w\}$,
$\{(i,j) : i = m - w \dots w^{+1}, j = w^{+1} \dots n - w\}$,
$\{(i,j) : i = m - w \dots w^{+1}, j = n - w \dots w^{+1}\}$,
$\{(i,j) : i = w^{+1} \dots x_1 - m, j = n - w \dots w^{+1}\}$
5: **while** $\max\{|d_{k+1}(\vec{x}) - d_k(\vec{x})|\} > tol$ **do**
6: go through the loop sets and do
$d_{k+1}(\vec{x}) = \min_{m \in [-w,w] \times [-w,w]}\{d_k(\vec{x}_m) + |\vec{x} - \vec{x}_m|_2^2\}$
7: **end while**
8: $d(\vec{x}) = \sqrt{d_{k+1}(\vec{x})}$
9: **OUTPUT:** $d(\vec{x})$

After thus identifying $\phi^{\text{in}}_{\text{bin}}$ and $\phi^{\text{out}}_{\text{bin}}$, we shift d down by T_s, yielding the level sets ϕ^{in1} and ϕ^{out1}. Then, the function $d - T_s$ is shifted up by exactly the **same** T_s, yielding ϕ^{in2} and ϕ^{out2}. It is clear that $T < T_s < b$ in order to keep the two

layers separate. As shown later, T_s is a scale parameter. The complete procedure is given in Algorithm 2.

Algorithm 2 cSDF Construction

1: **INPUT:** threshold T, T_s, $d(\vec{x})$
2: $d_0(\vec{x}) = d(\vec{x}) - T$
3: select any point p_0 on the outer boundary of the narrow band.
4: starting from p_0, label as $\phi^{\text{out}}_{\text{bin}}$ the connected component where $d_0 > 0$; label the rest of the region where $d_0 > 0$ as $\phi^{\text{in}}_{\text{bin}}$.
5: $d_1(\vec{x}) = d(\vec{x}) - T_s$
6: compute ϕ^{in1} and ϕ^{out1} using Algorithm 1 on $\phi^{\text{in}}_{\text{bin}}$ and $\phi^{\text{out}}_{\text{bin}}$, respectively, with input $d_1(\vec{x})$
7: $\phi^{\text{in2}} = \phi^{\text{in1}} - T_s$, $\phi^{\text{out2}} = \phi^{\text{out1}} - T_s$
8: **OUTPUT:** ϕ^{in2} and ϕ^{out2}

C. STEP 3: SURFACE RECONSTRUCTION USING cSDF

After computing ϕ^{in2} and ϕ^{out2}, a joint estimation of the signed-distance function ϕ of the reconstructed surface $\mathcal{S}$ is computed from ϕ^{in2}, ϕ^{out2}, and $d(\vec{x})$ as described in Algorithm 3.

Algorithm 3 Surface Reconstruction Using cSDF

1: **INPUT:** ϕ^{in2}, ϕ^{out2}, d
2: $d^{\text{in}}_{\text{out}} = ||\phi^{\text{in2}}| - |\phi^{\text{out2}}||$, $d^{\text{in}}_{\text{edge}} = ||\phi^{\text{in2}}| - d|$, $d^{\text{out}}_{\text{edge}} = ||\phi^{\text{out2}}| - d|$
3: **for** all $x \in N_b$ **do**
4: $t = \min\{d^{\text{in}}_{\text{out}}, d^{\text{in}}_{\text{edge}}, d^{\text{out}}_{\text{edge}}\}$
5: **if** $d^{\text{in}}_{\text{out}} == t$ **then** $\phi = (\phi^{\text{in2}} - \phi^{\text{out2}})/2$
6: **if** $d^{\text{in}}_{\text{edge}} == t$ **then** $\phi = -\phi^{\text{in2}}$
7: **if** $d^{\text{out}}_{\text{edge}} == t$ **then** $\phi = \phi^{\text{out2}}$
8: **end for**
9: **OUTPUT:** ϕ

There are only three possible curvature (c) cases for any point $\vec{x}_i$ on $\mathcal{S}$: positive, negative, or zero curvature (as illustrated in Fig. 3(a)), corresponding to the three cases in Algorithm 3:

- $c > 0$: $\mathcal{S}$ is convex at $\vec{x}_i$. Therefore, $\mathcal{S}$ is captured by d and ϕ^{out2}.
- $c < 0$: $\mathcal{S}$ is concave at $\vec{x}_i$. Therefore, $\mathcal{S}$ is captured by d and ϕ^{in2}.
- $c = 0$: $\mathcal{S}$ is planar at $\vec{x}_i$. Therefore, $\mathcal{S}$ is captured by ϕ^{in2} and ϕ^{out2}.

A simple average of the corresponding two fields provides the estimate for ϕ. Fig. 3(c) shows an example ϕ computed using this procedure.

D. RELATIONS TO OTHER METHODS

The cSDF method is related to coupled level set methods and to Hilbert-Huang transforms. However, cSDF uses a geometric coupling, while coupled level sets and the Hilbert-Huang transform only provide topology control. This difference is illustrated in Fig. 5. For example, the inner level set in coupled level sets can evolve arbitrarily as long as it stays inside the outer level set (left panel of Fig. 5). This is not possible in cSDF, where the two layers must be geometrically shifted by T_s (right panel of Fig. 5).

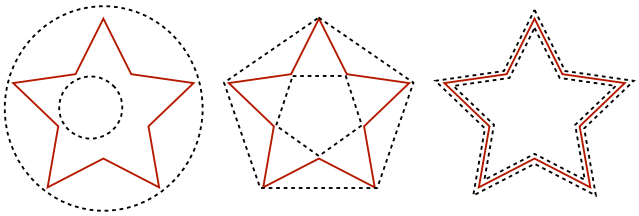


FIGURE 5. Illustration of the relations between the inner and outer level sets in coupled level-set methods (left), Hilbert-Huang transforms (middle), and the present cSDF (right). The solid red line is the true surface and the dash lines are the coupled inner and outer layers.

E. NOISE, OUTLIERS, AND PARAMETER

Since the present cSDF method is free of regularization and intends to reconstruct even minute details of the surface, it is sensitive to noise in the input point set. If the input point positions are noisy, or contain outliers, we hence use a different algorithm for computing the distance field $d(\vec{x})$. This robust algorithm is presented below. All downstream processing, in particular the cSDF construction, then remains unaffected.

We also analyze below how the parameter T_s controls the scale of the surface details to be recovered. This parameter hence is a scale-space parameter for the cSDF method.

1) ROBUST DISTANCE FIELD

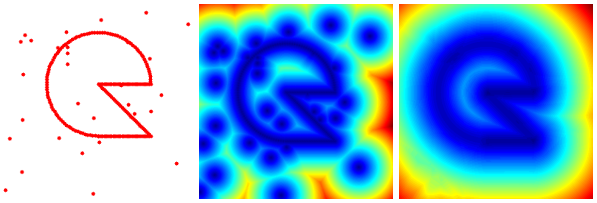
In the presence of outliers or noise in the input point set, we use a local variance-weighted method to compute the distance field, which is robust against noise and outliers. This choice is motivated by the fact that in practice the positions $\vec{x}_i$ are often uncertain due to, e.g., measurement errors. We model this uncertainty as Gaussian noise of mean zero and standard deviation σ_i i.i.d. added to the point positions. Usually, this σ_i can directly be obtained from the measurement or imaging device that acquired the data.

If σ_i is unknown, we estimate it by K_n-nearest neighbor clustering or by singular value decomposition. In the clustering method, we first compute the K_n-nearest neighbor distances of all points and compute their average. Then, we use the difference between the distance and this average as the weight. When using singular value decomposition, we first compute the variance matrix of the point set and use SVD to compute the distance to the plane defined by the first eigenvectors. In what follows, we use the clustering-based method, which is also called ''inverse distance weighting'' in the literature.

The weighted distance field is then defined as:

$$d(\vec{x}) = \frac{1}{\hat{D}} \sum_{i=1}^{K_n} \frac{1}{D_i + 1} (\vec{x} - \vec{x}_i)^2, \tag{9}$$

where $\hat{D} = \sum_{i=1}^{K_n} 1/(D_i + 1)$ is a normalization factor, $D_i = |x - \vec{x}_i|$ is the distance between x and $\vec{x}_i$, and x is an arbitrary position on the grid. A comparison between the so-obtained robust distance field and the one obtained using Algorithm 1 is shown in Fig. 6.



(a) point set (b) standard (c) robust (K_n=10)

FIGURE 6. Comparison of un-robust and robust distance fields.

2) SCALE PARAMETER T_S

The cSDF coupling parameter T_s controls the scale space in which the interpolated signal lives. Let

$$r = \frac{1}{NK} \sum_{i=1}^{N} \sum_{j=1}^{K_n} \sqrt{(\vec{x}_i - \vec{x}_{i_j})^2} \tag{10}$$

be the average (across all data points) distance between K_n-nearest neighbors. Then, T_s has to satisfy the condition:

$$T_s > r \tag{11}$$

in order for neighboring points to meet.

This defines the lower bound on how small of details can possibly be recovered from the samples by cSDF. An illustration is shown in Fig. 7. The green dot represents a grid point $\vec{x}_{i,j}$, the red dots represent the input samples $\vec{x}_i$. The two red samples within the shaded disk around the green dot are indistinguishable by $\vec{x}_{i,j}$. Therefore, T_s must be larger than the radius of the shaded disk in order to ensure that it is unnecessary to distinguish between those two samples (lower bound).

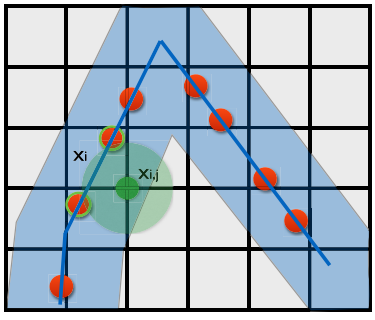
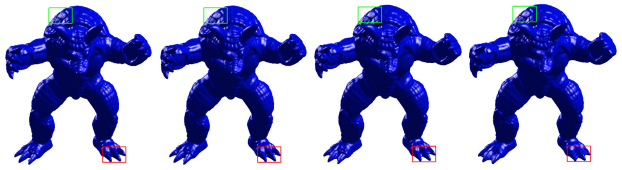


FIGURE 7. Illustration of T_s. Any samples on the circle will not be distinguished by $\vec{x}_{i,j}$. To make sure the gap is closed, $T_s > r$.

As mentioned, thresholding with T_s is equivalent to wave propagation. Thus, this step turns the explicit discrete-sample representation of the surface into an implicit continuous representation.

Increasing T_s, however, is not equivalent to smoothing or to a regularizer. T_s only defines the scale space for the interpolation, but does not limit the curvature within that space. As seen in the area highlighted by the green rectangle in Fig. 8, surface details are not lost when increasing T_s. However, between closely apposed surfaces, a too coarse scale space may lead to topological problems, as for example shown in the red rectangle in Fig. 8. This issue can be avoided by adaptively changing T_s in a standard scale-space approach, or by using a spatially adapted $T_s(\vec{x})$.



(a) $T_s = 4$ (b) $T_s = 5$ (c) $T_s = 6$ (d) $T_s = 8$

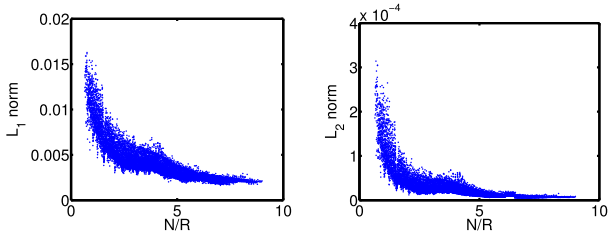
FIGURE 8. Changing the scale parameter T_S does not introduce surface smoothing (green rectangle), but may lead to topological problems (red rectangle) due to scale-space coarsening.

III. NUMERICAL VALIDATION

We validate cSDF in 2D and show its accuracy by measuring the error under the ℓ^1 and ℓ^2 norms with uniformly sampled points on a circle. We further perform 3D benchmarks on computer-graphics models.

A. 2D BENCHMARKS

We test the accuracy of cSDF by sampling N points uniformly on a circle of radius R and comparing the reconstructed circle to the ground truth for decreasing N. We use a 200×200 grid for all $N \in [45, 360] \times R \in [40, 70]$ and directly compute distances without LUT. We linearly interpolate the resulting ϕ at each of the original $\vec{x}_i \in S$. The correct value would be $\phi^c = 0$ for all $\vec{x}_i$. We then compute the overall (reconstruction plus interpolation) ℓ^1 and ℓ^2 errors as $[\sum_{i=1}^{i=N} |\phi(\vec{x}_i)|]/N$ and $\left([\sum_{i=1}^{i=N} \phi(\vec{x}_i)^2]/N\right)^{1/2}$, respectively. The result is shown in Fig. 9.



(a) ℓ^1-error vs. N/R (b) ℓ^2-error vs. N/R

FIGURE 9. Reconstruction errors for clouds of N points on circles of different radii R without LUT.

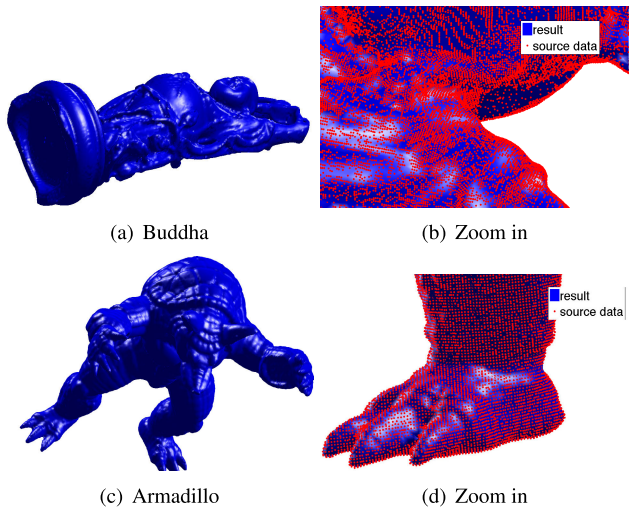
Figure 3 shows a synthetic example with sharp corners to illustrate cSDF's capability to represent them without introducing excessive surface smoothing. The points in this example are moreover irregularly distributed on $\mathcal{S}$.

TABLE 1. cSDF 3D benchmarks (narrow band width $b = 6$ in all cases) with Extended-window Fast Sweeping.

Surface	# Points	Grid	CPU	CPU (LTU)	ℓ^1 error	ℓ^2 error
Armadillo	172 974	$175 \times 208 \times 159$	39.9 s	12.8 s	0.0523	0.0755
Buddha	543 652	$82 \times 199 \times 82$	32.6 s	15.9 s	0.0673	0.1122

B. 3D BENCHMARKS

We benchmark cSDF in 3D by using the vertices of the triangulated surfaces of the well-known computer-graphics models "Armadillo" and "Buddha" as input point clouds. The number of points for each model, the CPU time for cSDF reconstruction of the implicit surface representation, and the resulting errors against the known ground truth at the vertex positions are given in Table 1. The code is implemented in C and run on a 2GHz Intel Core i7. When the distance LUT is used, the CPU time is further reduced to 12.8s and 15.9s for "Armadillo" and "Buddha", respectively. The timings compare favorably with the $>$ 200s CPU time of an efficient Bregman code [27] for "Buddha" with similar resolution. Figure 10 shows the resulting reconstructions and close-ups (Fig. 10(b) and (d)) with the input point cloud overlaid to demonstrate the method's capability to represent high-curvature regions without grid refinement (see, for example, the "Armadillo" claws).



(a) Buddha (b) Zoom in

(c) Armadillo (d) Zoom in

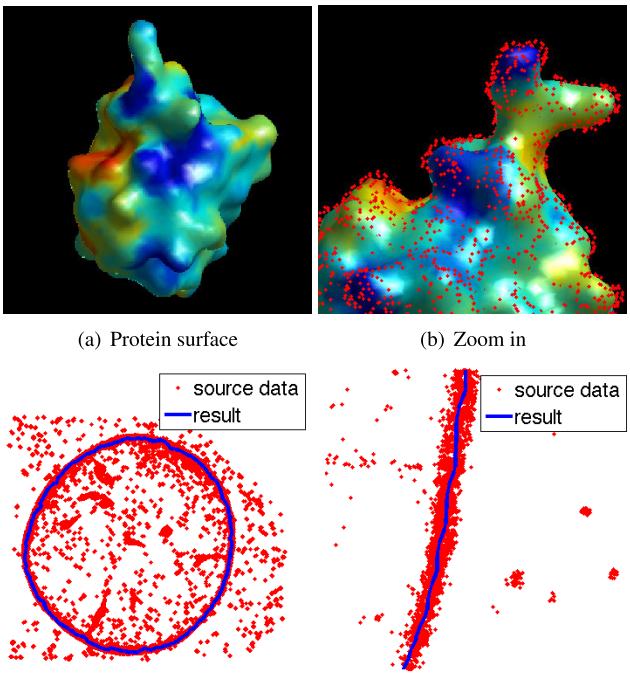
FIGURE 10. Computer-graphics model surface reconstruction using cSDF.

IV. SURFACE RECONSTRUCTION

In this section, we illustrate the use of cSDF in several applications on real biological data. Since the cSDF is prior-free, the resulted geometry is only based on the point cloud. Therefore, the property of the surface, such as normal or curvature distribution, is only from the geometry itself, without being corrupted by any prior.

The first data set comprises 3D positions of atoms in a protein conformation obtained from molecular-dynamics simulations.[1] This is an example of noise-free data where the surface is to be reconstructed as accurately as possible. We use cSDF to reconstruct the molecular surface of the protein and to locally shade it according to the Molecular Hydrophobicity Potential (MHP). The result is shown in Fig. 11(a) and (b).

The second case considers a 2D PALM (photo-activated localization microscopy) super-resolution image. PALM intrinsically produces point clouds, as it detects the centroids of single fluorescent molecules. cSDF can then be used to reconstruct the surface (e.g., the membrane) on which the fluorescent molecules live. The PALM image in Fig. 11(c) and (d) shows fluorescent lamin proteins of the nuclear lamina.[2] We use cSDF to reconstruct the nuclear envelope from these point detections. Due to the large amount of outliers and noise in this data set, we use the robust distance field method presented above.



(a) Protein surface (b) Zoom in

(c) 2D PALM data (d) Zoom in

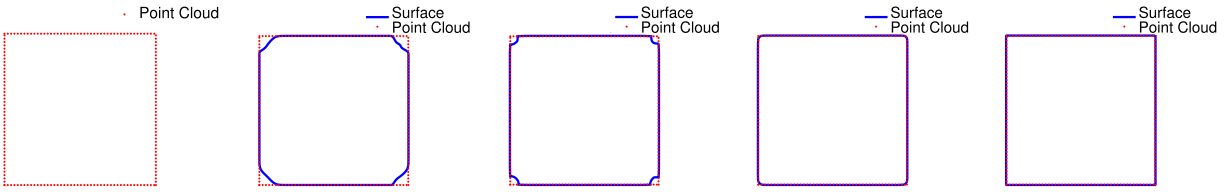
FIGURE 11. Biological surface reconstruction using cSDF in the absence and in the presence of noise and outliers. The protein surface is colored by MHP value.

A. COMPARED WITH TV

In its current implementation, cSDF is about 6 times faster than a highly efficient Bregman code for surface reconstruction [27]. This runtime can be further reduced by parallelizing

[1] Data courtesy of Dr. Anton Polyansky, Zagrovic group, MFPL, Vienna.

[2] Data courtesy of Dr. Jonas Ries, Ewers group, ETH Zürich.



(a) Input (b) TV results [6] (c) cSDF result

FIGURE 12. 2D example with sharp corners, compared with TV regularization [6] with parameter $\mu = 10^{-6}$ (left), 2×10^{-6} (middle) and 3×10^{-6} (right) and presented method.

the code on multi- or many-core hardware, such as GPUs or computer clusters. cSDF computes the distance field three times and then estimates the signed distance field. Thus, the key point of cSDF's parallelization is to compute the distance field in parallel, which has been extensively studied in computer graphics [28], [29].

Due to its regularization-free nature, cSDF can capture high curvature without adaptive grid. It also does not bias the reconstruction result toward a prior, nor does it excessively smooth the reconstructed surface. This way, cSDF is for example able to perfectly reconstruct and represent the sharp corners and edges of a cube, whereas regularization-based methods will round them even for the smallest amount of regularization, as shown in Fig. 12.

B. MEAN CURVATURE ESTIMATION

From ϕ, thanks to the signed-distance property of cSDF, the mean curvature can directly be computed as:

$$H = \frac{\Delta\phi}{\|\nabla\phi\|} = \Delta\phi. \tag{12}$$

Examples are shown in Figs. 13. Based on the prior-free property, the normal and curvature are guaranteed to be features of the surface itself, without being corrupted by any prior.

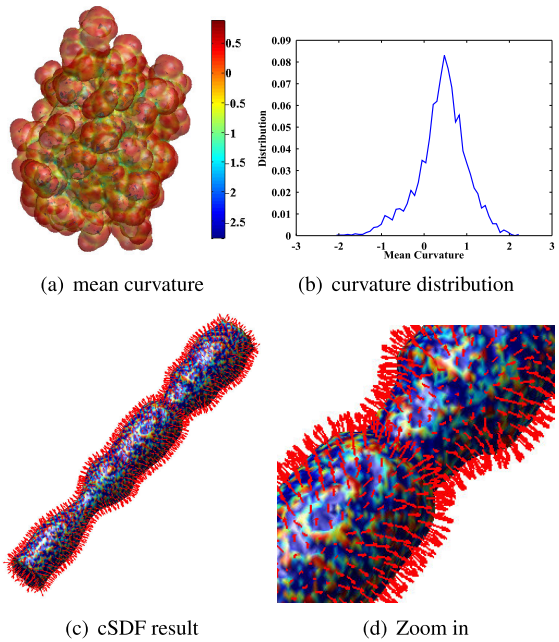
V. PROTEIN SURFACE ESTIMATION

cSDF can be extended to volume data, where the inner distance field vanishes. The surface of the volume is captured by the outer distance field and the distance field of the point cloud. Since the inner distance field does not exist, the concave region can not be accurately recovered (sharp inside corners get smoothed). The algorithm is summarized in Algorithm 4. The process is very similar with shrinking effect in traditional level set method.

cSDF can be further extend to volume data, where a point $\vec{x}_i$ becomes a sphere centered at $\vec{x}_i$ with given radius r_i. Algorithm 4 can be used to handle this case by letting $d(\vec{x})$ inside the sphere be negative.

A. MEAN CURVATURE DISTRIBUTION PRIOR

Since cSDF is regularization-free, we can use it to obtain prior knowledge about the surfaces. We prepare 17 different proteins from Molecular Dynamics Simulations with 1000 time



(a) mean curvature (b) curvature distribution

(c) cSDF result (d) Zoom in

FIGURE 13. Mean curvature and its distribution as estimated using cSDF for a molecular protein surface. (a) The protein surface colored by curvature. (b) The mean curvature distribution of (a). (c) Surface reconstruction, normal estimation, and curvature estimation for a part of a human aorta point cloud (data courtesy of Dr. George Bourantas, MPI-CBG). The mean curvature is color coded after curvature histogram equalization for better visualization. (d) Zoom of (c). It worth pointing out that the distribution is only relied on the surface without being corrupted by any prior since cSDF is prior-free.

step.[3] The proteins are summarized in Table 2. These names are the same as they are in the Protein Data Bank.[4] We have five independent runs for ubiquitin (UBQ), and five independent runs for UBM2. In total, we have 25 trajectories, each of which has 1000 time steps. The number of points for each protein is shown in Table 2.

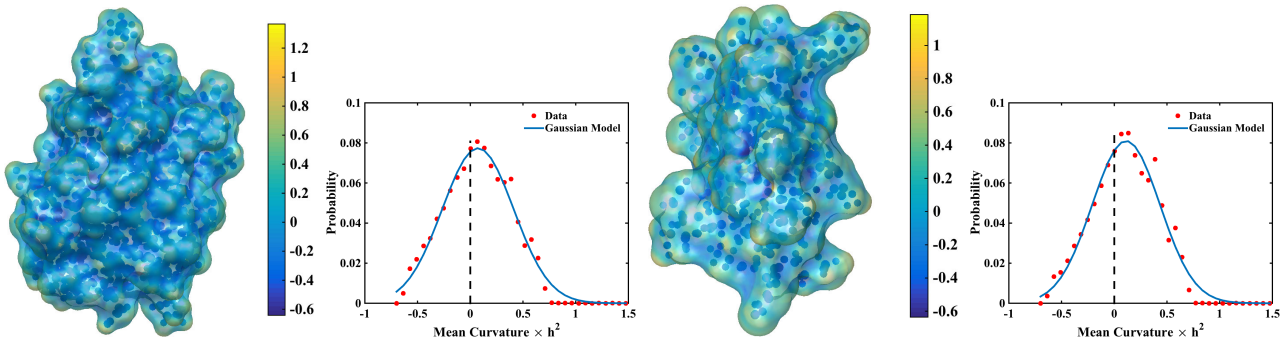
We use Algorithm 4 to construct the 25,000 protein surfaces and estimate their mean curvature. To reduce the resolution effect, instead of studying H, we study $H \cdot h^2$, which is called mean curvature half density [30], [31] or weighted curvature [32]. And it is independent on resolution h.

Two distributions of $H \cdot h^2$ from the examples are shown in Fig. 14. Even though these two surfaces are very different,

[3]Data courtesy of Dr. Anton Polyansky, Zagrovic group, MFPL, Vienna.
[4]http://www.rcsb.org/pdb

TABLE 2. First row: the index of 17 proteins. Second row: the names of 17 proteins. Third row: the number of atoms in this protein. The fourth row: protein volume with unit 10^{-30} cubic meters, estimated by VOSS-VOLUME-VOXELATOR program. The protein names are the same as they are in the Protein Data Bank.

Index	1	2	3	4	5	6	7	8	9
Name	1AKL	1AKZ	1KPP	1SCD	1UCH	1UGI	1W8V	1YD8_unb	2BVB
N_p	6736	3499	2324	3776	3559	1285	2471	1469	1927
Volume	59931	31360	20400	33660	42760	11034	21996	12561	17012
Index	10	11	12	13	14	15	16	17	
Name	2GKR	2K2T	2OOA	2PXR	2RN4	3R3Q	UBM2	UBQ	
N_p	725	737	721	2199	1568	2340	512	1226	
Volume	6380	6413	6036	19484	13566	20645	4320	10530	



(a) UBQ (b) $p(H \cdot h^2)$ (c) UBM2 (d) $p(H \cdot h^2)$

FIGURE 14. cSDF on Volume Point Cloud. (a) UBQ protein surface; (b) distribution of $H \cdot h^2$ for (a), the Gaussian model is $0.0774 \exp(\frac{-(x-0.0727)^2}{0.4795^2})$ with fitting accuracy $SSE = 5.8 \times 10^{-4}$ and $R^2 = 0.9793$; (c) UBM2 protein surface; (d) distribution of $H \cdot h^2$ for (c), the Gaussian model is $0.081 \exp(\frac{-(x-0.1121)^2}{0.4550^2})$ with fitting accuracy $SSE = 8.2 \times 10^{-4}$ and $R^2 = 0.9731$. We use $h = 0.4$Å and set the range of $H \cdot h^2$ to be $[-0.7, 1.5]$. We use $\frac{1.5+0.7}{h^3} \approx 35$ bins to represent $p(H \cdot h^2)$.

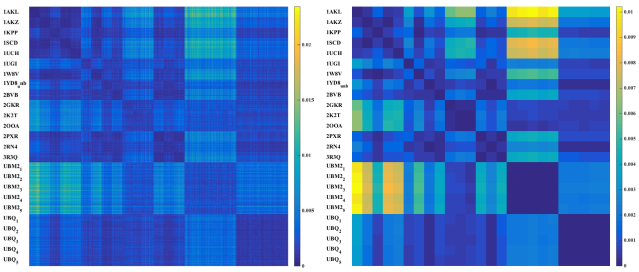
Algorithm 4 Surface Reconstruction From Volume Point Cloud

INPUT: $\{\vec{x}_i\}, T_s$
compute $d(\vec{x})$ by Algorithm 1
$d_1 = d - T_s$
compute d^{out1} by Algorithm 1 on d_1
$d^{\text{out2}} = d^{\text{out1}} - T_s$
if $d > T_s$ **then**
  $\phi = d$
else
  $\phi = -d^{\text{out2}}$
end if
OUTPUT: ϕ

their distributions of $H \cdot h^2$ are similar. This fact inspires us to study $p_t^i(H \cdot h^2)$ across all trajectories, where $t \in [1, 1000]$ is the time step index and i is the protein index in Table 2 (the index starts from left to right and from up to down).

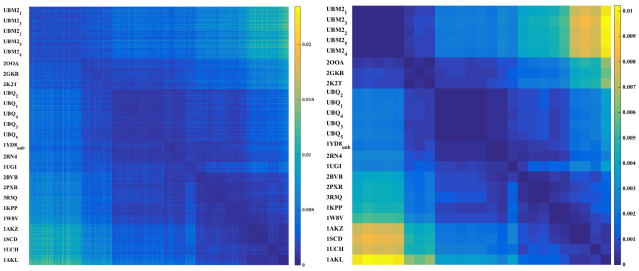
For all proteins, the radii of their atoms are between 1.2Å and 1.9Å, where Å $= 10^{-10}$ meter. We set $h = 0.4$Å and $T_s = 5h$. We set the range of $H \cdot h^2$ to be $[-0.7, 1.5]$. We use $\frac{1.5+0.7}{h^3} \approx 35$ bins to represent $p(H \cdot h^2)$ because H has second-order accuracy $O(h^2)$.

We compute a distance matrix $\vec{M}$ between all proteins at each time step. $\vec{M}(j,k)$ are the χ^2 distances between $p_{t_1}^{i_1}$ and $p_{t_2}^{i_2}$, where $j = (i_1 - 1) * 1000 + t_1$ and $k = (i_2 - 1) * 1000 + t_2$. The result is shown in Fig. 15a.



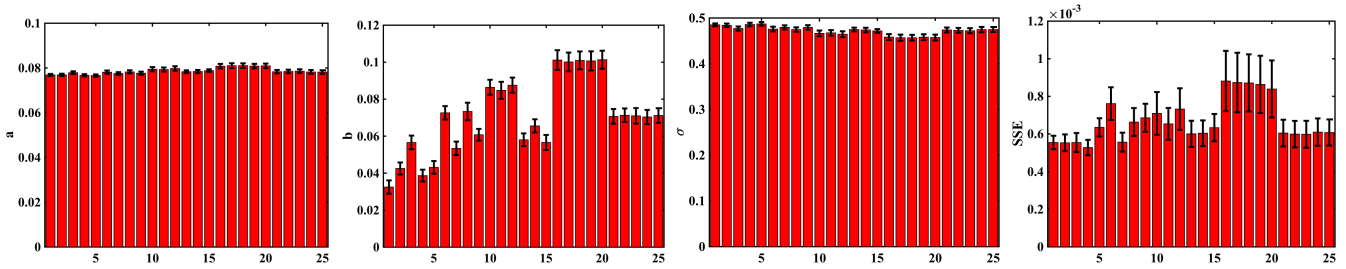
(a) Distance Matrix $\vec{M}$ (b) Distance Matrix $\tilde{\vec{M}}$

FIGURE 15. Distance matrix for all trajectories and their average.



(a) Rearrangement of $\vec{M}$ (b) Rearrangement of $\tilde{\vec{M}}$

FIGURE 16. Rearrangement of two distance matrices.



(a) average a and its variance (b) average b and its variance (c) average σ and its variance (d) fitting accuracy

FIGURE 17. Fitting results with Gaussian models. The x-axis is the protein index. The error bars show the standard deviations for the time steps.

We define the average distribution for each protein as

$$\tilde{p}^i(H \cdot h^2) = \frac{\int_t p_t^i(H \cdot h^2)\mathrm{d}t}{\int \mathrm{d}t}. \tag{13}$$

Similarly, we can compute a distance matrix $\tilde{\tilde{M}}$, where $\tilde{\tilde{M}}(j,k)$ is the χ^2 distance between $\tilde{p}^j$ and $\tilde{p}^k$.

We use isomap algorithm [33] to reorder the proteins, showing the relationship between each other. The reordered distance matrix is shown in Fig. 16a and b. The rearrangement shows the relationship between proteins.

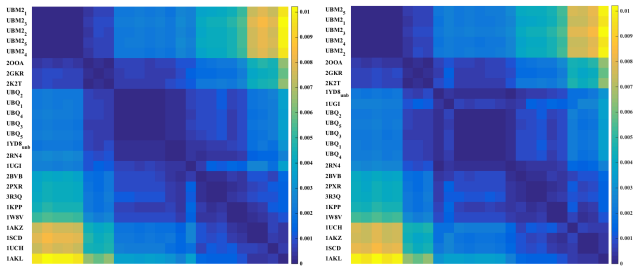
B. CURVATURE DISTRIBUTION MODELING

Noticing the distributions of $H \cdot h^2$ can be well approximated by a Gaussian distribution in Fig. 14, we use a Gaussian model to approximate each distribution of $H \cdot h^2$ for all 25,000 protein surfaces. The Gaussian model is defined as

$$f(x) = a \exp\left(-\left(\frac{x-b}{\sigma}\right)^2\right). \tag{14}$$

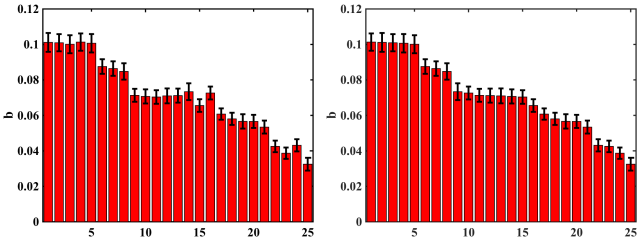
The modeling results are shown in Fig. 17. The parameters a and σ are stable for all tested proteins. The parameter b is stable for each protein during time steps. This stability guarantees that mean curvature distribution can be used as priors.

We can also rearrange the proteins by simply sorting the parameter b. The result is shown in Fig. 18b. The reordered parameter b is shown in Fig. 19b. The similarity between the



(a) distance matrix after reordering by Isomap (b) distance matrix after reordering by sorting b

FIGURE 18. Distance matrix after the rearrangement of these proteins.



(a) parameter b after protein rearrangement by Isomap (b) parameter b after protein rearrangement by sorting

FIGURE 19. The parameter b in different rearrangement methods. The error bars show the standard deviations for the time steps.

result from Isomap and the result from sorting parameter b suggests that b is a dominant parameter in this modeling.

VI. SUMMARY

We have presented a regularization-free method for geometry reconstruction from unstructured point clouds. The result is guaranteed to be a signed-distance function, dispensing with the need for re-initialization and regularization. We benchmarked the method on 2D and 3D artificial datasets and showed its accuracy and computational efficiency. We further showed its application in real-world surface reconstruction for protein molecular surfaces and PALM microscopy data.

Thanks to the regularization-free property, the mean curvature distributions from the estimated surfaces can be obtained and modeled as a prior. We show how to compute and model the mean curvature distributions for a Molecular Dynamic Simulation dataset.

Thanks to the computational efficiency, our method can reach real-time performance and can be applied in many fields, such as protein surface estimation, studying the relationship between structure and function, molecular dynamics, etc.

REFERENCES

[1] M. Kazhdan, "Reconstruction of solid models from oriented point sets," in *Proc. 3rd Eurograph. Symp. Geometry Process.*, 2005, p. 73.
[2] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. 4th Eurograph. Symp. Geometry Process.*, 2006, pp. 61–70.
[3] A. C. Öztireli, G. Guennebaud, and M. Gross, "Feature preserving point set surfaces based on non-linear kernel regression," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 493–501, Apr. 2009.

[4] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method," *Comput. Vis. Image Understand.*, vol. 80, no. 3, pp. 295–314, Dec. 2000.
[5] H. K. Zhao, S. Osher, and R. Fedkiw, "Fast surface reconstruction using the level set method," in *Proc. IEEE Workshop Variational Level Set Methods Comput. Vis.*, Jul. 2001, pp. 194–201.
[6] T. Goldstein, X. Bresson, and S. Osher, "Geometric applications of the split Bregman method: Segmentation and surface reconstruction," *J. Sci. Comput.*, vol. 45, nos. 1–3, pp. 272–293, Oct. 2010.
[7] T. Goldstein and S. Osher, "The split Bregman method for L1-regularized problems," *SIAM J. Imag. Sci.*, vol. 2, no. 2, pp. 323–343, Jan. 2009.
[8] G. Paul, J. Cardinale, and I. F. Sbalzarini, "Coupling image restoration and segmentation: A generalized linear model/Bregman perspective," *Int. J. Comput. Vis.*, vol. 140, no. 1, pp. 69–93, 2013.
[9] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proc. Nat. Acad. Sci. USA*, vol. 93, no. 4, p. 1591, 1996.
[10] I. Tobor, P. Reuter, and C. Schlick, "Multi-scale reconstruction of implicit surfaces with attributes from large unorganized point sets," in *Proc. Shape Modeling Appl.*, Jun. 2004, pp. 19–30.
[11] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk, and H.-P. Seidel, "Multi-level partition of unity implicits," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 463–470, Jul. 2003.
[12] M. B. Nielsen and K. Museth, "Dynamic tubular grid: An efficient data structure and algorithms for high resolution level sets," *J. Sci. Comput.*, vol. 26, no. 3, pp. 261–299, Mar. 2006.
[13] C. Li, C. Xu, C. Gui, and M. D. Fox, "Level set evolution without re-initialization: A new variational formulation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2005, pp. 430–436.
[14] Y. Gong, Q. Wang, C. Yang, Y. Gao, and C. Li, "Symmetry detection for multi-object using local polar coordinate," in *Computer Analysis of Images and Patterns* (Lecture Notes in Computer Science), vol. 5702. Berlin, Germany: Springer, 2009, p. 277.
[15] M. Sussman, "A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles," *J. Comput. Phys.*, vol. 187, no. 1, pp. 110–136, 2003.
[16] N. Paragios and R. Deriche, "Coupled geodesic active regions for image segmentation: A level set approach," in *Proc. Europ. Conf. Comput. Vis.* Berlin, Germany: Springer, 2000, pp. 224–240.
[17] X. Zeng, L. H. Staib, R. T. Schultz, and J. S. Duncan, "Segmentation and measurement of the cortex from 3-D MR images using coupled-surfaces propagation," *IEEE Trans. Med. Imag.*, vol. 18, no. 10, pp. 927–937, Oct. 1999.
[18] X. Han, C. Xu, and J. L. Prince, "A topology preserving level set method for geometric deformable models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 6, pp. 755–768, Jun. 2003.
[19] S. Kim, "An $O(N)$ level set method for Eikonal equations," *SIAM J. Sci. Comput.*, vol. 22, no. 6, pp. 2178–2193, 2001.
[20] H. Zhao, "A fast sweeping method for Eikonal equations," *Math. Comput.*, vol. 74, no. 250, pp. 603–628, 2005.
[21] W.-K. Jeong and R. T. Whitaker, "A fast iterative method for eikonal equations," *SIAM J. Sci. Comput.*, vol. 30, no. 5, pp. 2512–2534, Jan. 2008.
[22] M. Sussman and E. Fatemi, "An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow," *SIAM J. Sci. Comput.*, vol. 20, no. 4, pp. 1165–1191, Jan. 1999.
[23] Y. Gong, G. Paul, and I. F. Sbalzarini, "Coupled signed-distance functions for implicit surface reconstruction," in *Proc. 9th IEEE Int. Symp. Biomed. Imag. (ISBI)*, May 2012, pp. 1000–1003.
[24] L. Greengard and V. Rokhlin, "The rapid evaluation of potential fields in three dimensions," in *Vortex Methods* (Lecture Notes in Mathematics), vol. 1360. Berlin, Germany: Springer, 1988, pp. 121–141.
[25] M. Sethi, A. Rangarajan, and K. Gurumoorthy, "The schrodinger distance transform (SDT) for point-sets and curves," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 198–205.
[26] J. Baert, A. Lagae, and P. Dutré, "Out-of-core construction of sparse voxel octrees," in *Proc. 5th High-Perform. Graph. Conf. HPG*, 2013, pp. 27–32.
[27] J. Ye, X. Bresson, T. Goldstein, and S. Osher, "A fast variational method for surface reconstruction from sets of scattered points," UCLA, UCLA CAM, Los Angeles, CA, USA, Tech. Rep. 10(01), 2010.
[28] N. Cuntz and A. Kolb, "Fast hierarchical 3D distance transforms on the GPU," in *Eurographics Short Papers*. New York, NY, USA: ACM, 2007, pp. 93–96.
[29] T.-T. Cao, K. Tang, A. Mohamed, and T.-S. Tan, "Parallel banding algorithm to compute exact distance transform with the GPU," in *Proc. ACM SIGGRAPH Symp. Interact. 3D Graph. Games I3D*, 2010, pp. 83–90.
[30] G. Kamberov, F. Pedit, and U. Pinkall, "Bonnet pairs and isothermic surfaces," *Mathematics*, vol. 92, no. 3, pp. 637–644, Jun. 1998.
[31] K. Crane, U. Pinkall, and P. Schröder, "Robust fairing via conformal curvature flow," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 1–10, Jul. 2013.
[32] Y. Gong and I. F. Sbalzarini, "Local weighted Gaussian curvature for image processing," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2013, pp. 534–538.
[33] J. B. Tenenbaum, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000.

YUANHAO GONG received the B.Sc. degree from Tsinghua University, Beijing, China, in 2007, and the Ph.D. degree from ETH Zürich, Switzerland, in 2015. From 2017 to 2018, he held a postdoctoral position with the Computer Vision Laboratory, ETH Zürich. Since 2018, he has been an Assistant Professor with Shenzhen University, China. His research interests include curvature-based image processing, high performance image processing, and biomedical image processing. He received several awards, including the Best Paper Award at the 2012 IEEE International Symposium on Biomedical Imaging.

YONG CHEN received the B.Eng. degree in mechanical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2008, and the Ph.D. degree in mechanical engineering from ETH Zürich, Switzerland, in 2018. Since 2019, he has been a Lecturer with the Wuhan University of Technology, China. His research interests include computational fluid mechanics and high performance computing.