Fast Algorithms for Surface Reconstruction from Point Cloud

(Article begins on next page)

15 October 2024

Fast Algorithms for Surface Reconstruction from Point Cloud

Yuchen He,* Martin Huska †, Sung Ha Kang‡ and Hao Liu§

Abstract

We consider constructing a surface from a given set of point cloud data. We explore two fast algorithms to minimize the weighted minimum surface energy in [Zhao, Osher, Merriman and Kang, Comp.Vision and Image Under., 80(3):295-319, 2000]. An approach using Semi-Implicit Method (SIM) improves the computational efficiency through relaxation on the time-step constraint. An approach based on Augmented Lagrangian Method (ALM) reduces the run-time via an Alternating Direction Method of Multipliers-type algorithm, where each sub-problem is solved efficiently. We analyze the effects of the parameters on the level-set evolution and explore the connection between these two approaches. We present numerical examples to validate our algorithms in terms of their accuracy and efficiency.

1 Introduction

Acquisition, creation and processing of 3D digital objects is an important topic in various fields, e.g., medical imaging [22], computer graphics [11, 8], industry [4], and preservation of cultural heritage [16]. A fundamental step is to reconstruct a surface from a 3D scanned point cloud data [1], denoted as $\mathcal{D} \subseteq \mathbb{R}^m$ for $m = 2$ or 3, such as in Figure 1.
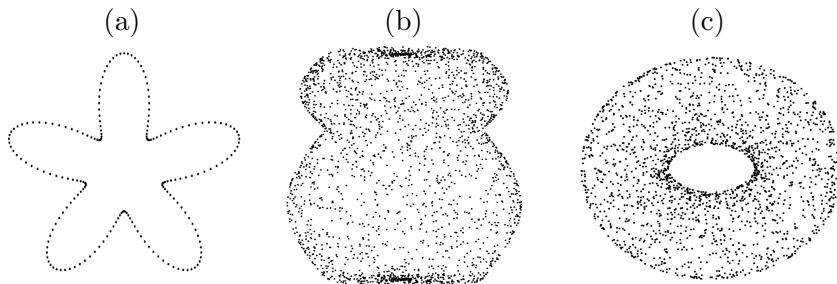
Figure 1: Test point clouds. (a) Five-fold circle (200 points). (b) Jar (2100 points). (c) Torus (2000 points).

We focus on reconstructing an m-dimensional manifold Γ, a curve in $\mathbb{R}^2$ or a surface in $\mathbb{R}^3$, from the point cloud $\mathcal{D}$. We assume only the point locations are given, and no other geometrical information such as normal vectors at each point are known. We explore fast algorithms for minimizing the following energy proposed in [38]:

$$E_p(\Gamma) = \left(\int_\Gamma |d(\mathbf{x})|^p \, d\mathbf{x} \right)^{\frac{1}{p}}, \tag{1}$$

where $d(\mathbf{x}) = \min_{\mathbf{y} \in \mathcal{D}} \{|\mathbf{x} - \mathbf{y}|\}$ is the distance from any point $\mathbf{x} \in \mathbb{R}^m$ to $\mathcal{D}$, p is a positive integer, and $d\mathbf{x}$ is the surface area element. This energy is the p-norm of the distance function

restricted in Γ. In [38], the authors used the fast sweeping scheme to solve Eikonal equation, and the Euler-Lagrange equations are solved by gradient descent algorithm.

Among many ways to represent the underlying surface, e.g., (moving) least square projection [2, 30], radial basis function [13, 9, 10, 11, 18], poisson reconstruction [20, 5, 24, 21, 14], we use the level set method as in [15, 37, 38]. The level set formulation allows topological changes and self-intersection during the evolution [29, 28], and gain popularity in many applications [12, 36, 31] just to mention a few. We represent the surface as a zero level set of $\phi : \mathbb{R}^{m+1} \to \mathbb{R}$:

$$\Gamma = \phi^{-1}(0) = \{\mathbf{x} \in \mathbb{R}^m \mid \phi(\mathbf{x}) = 0\}.$$

There are various related works on surface reconstruction from point cloud data: a convection model proposed in [37], a data-driven logarithmic prior for noisy data in [33], using surface tension to enrich the Euler-Lagrange equations in [17], and using principal component analysis to reconstruct curves embedded in sub-manifolds in [27]. A semi-implicit scheme is introduced in [34] to simulate the curvature and surface diffusion motion of the interface. In [25], the authors defined the surface via a collection of anisotropic Gaussians centered at each entry of the input point cloud, and used TVG-L1 model [7] for minimization. A similar strategy addresses an ℓ_0 gradient regularization model proposed in [23]. Some models incorporate additional information. In [26], the authors proposed a novel variational model, consisting of the distance, the normal, and the smoothness term. Euler's Elastica model is incorporated for surface reconstruction in [32] where graph cuts algorithm is used. The model in [15] extends the active contours segmentation model to 3D and implicitly allows controlling the curvature of the level set function.

In this paper, we explore fast algorithms to minimize the weighted minimum surface energy (1) for $p = 1$ and 2. We propose a Semi-Implicit Method (SIM) to relax the time-step constraint for $p = 2$, and an Augmented Lagrangian Method (ALM) based on the alternating direction method of multipliers (ADMM) approach for $p = 1$. These algorithms minimize the weighted minimal surface energy (1) with high accuracy and superior efficiency. We analyze the behavior of ALM in terms of parameter choices and explore its connection to SIM. Various numerical experiments are presented to discuss the effects of the algorithms.

We organize this paper as follows. In Section 2, we present the two methods: SIM and ALM, and explore their connection. Numerical experiments are presented in Section 3, and effects of the parameters are discussed in Subsection 3.3. We conclude our paper in Section 4.

2 Proposed Algorithms

Let $\Omega \subset \mathbb{R}^m$ ($m = 2$ or 3) denote a bounded domain containing the given m-dimensional point cloud data, $\mathcal{D}$. Using the level-set formulation for Γ, the d-weighted minimum surface energy (1) can be rewritten as:

$$E_p(\phi) = \left(\int_\Omega |d(\mathbf{x})|^p \delta(\phi) |\nabla \phi| \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}. \tag{2}$$

Here $\delta(x)$ is the Dirac delta function which takes $+\infty$ when $x = 0$, and 0 elsewhere. Compared to (1), this integral is defined on Ω, which makes the computation flexible and free from explicitly tracking Γ. We use $p = 2$ for SIM introduced in Section 2.1, and $p = 1$ for ALM in Section 2.2. In general, $p = 2$ is a natural choice, since it provides better stability and efficiency for a semi-implicit type PDE-based method. For ALM, we explore $p = 1$ to take advantage of an aspect of fast algorithm in ADMM setting such as shrinkage, similarly to the case in [3]. Visually, the numerical results of surface reconstruction are similar for $p = 1$ or $p = 2$ (see Section 3).

2.1 Semi-Implicit Method (SIM) to minimize E_2

We introduce a gradient-flow-based semi-implicit method to Minimize

$$E_2(\phi) = \left(\int_\Omega d(\mathbf{x})^2 \delta(\phi) |\nabla \phi| \, \mathrm{d}\mathbf{x} \right)^{\frac{1}{2}}. \tag{3}$$

Following [38], the first variation of $E_2(\phi)$ with respect to ϕ is characterized as a functional:

$$\left\langle \frac{\partial E_2(\phi)}{\partial \phi}, v \right\rangle = \int_\Omega \frac{1}{2} \delta(\phi) \left[\int_\Omega d^2(\mathbf{x}) \delta(\phi) |\nabla \phi| \, \mathrm{d}\mathbf{x} \right]^{-1/2} \nabla \cdot \left[d^2(\mathbf{x}) \frac{\nabla \phi}{|\nabla \phi|} \right] v \, \mathrm{d}\mathbf{x}$$

for any test function v from the Sobolev space H^1. Minimizing (3) is equivalent to finding the critical point ϕ such that $\left\langle \frac{\partial E_2(\phi)}{\partial \phi}, v \right\rangle = 0, \forall v \in H^1$. This is associated with solving the following initial value problem:

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = \bar{f}(d, \phi) \nabla \cdot \left[d^2(\mathbf{x}) \dfrac{\nabla \phi}{|\nabla \phi|} \right], \\ \phi(\mathbf{x}, 0) = \phi^0, \end{cases} \tag{4}$$

where ϕ^0 is an initial guess for the unknown ϕ, and $\bar{f}(d, \phi) = \dfrac{1}{2} \delta(\phi) \left[\int_\Omega d^2(\mathbf{x}) \delta(\phi) |\nabla \phi| \, \mathrm{d}\mathbf{x} \right]^{-1/2}$. The steady state solution of (4) gives a minimizer ϕ^* of $E_2(\phi)$.

Here the delta function δ is realized as the derivative of the one dimensional Heaviside function $H : \mathbb{R} \to \{0, 1\}$. We adopt the smooth approximation of $H(\phi)$ as in [6]:

$$H(\phi) \approx H_\varepsilon(\phi) = \frac{1}{2} + \arctan(\phi/\varepsilon)/\pi \quad \text{and} \quad \delta(\phi) \approx H_\varepsilon'(\phi) = \frac{\varepsilon}{\pi(\varepsilon^2 + \phi^2)} \tag{5}$$

with $\varepsilon > 0$ as the smoothness parameter. Then $\bar{f}$ is approximated by its smoothed version f expressed as

$$f(d, \phi) = \frac{1}{2} \frac{\varepsilon}{\pi(\varepsilon^2 + \phi^2)} \left[\int_\Omega d^2(\mathbf{x}) \frac{\varepsilon}{\pi(\varepsilon^2 + \phi^2)} |\nabla \phi| \, \mathrm{d}\mathbf{x} \right]^{-1/2}.$$

We add a stabilizing diffusive term $-\beta \Delta \phi$ for $\beta > 0$ on both sides of the PDE in (4) to consolidate the computation, similarly to [34]:

$$\frac{\partial \phi}{\partial t} - \beta \Delta \phi = -\beta \Delta \phi + f(d, \phi) \nabla \cdot \left[d^2(\mathbf{x}) \frac{\nabla \phi}{|\nabla \phi|} \right]. \tag{6}$$

Employing a semi-implicit scheme, we solve ϕ from (6) by iteratively updating ϕ^{n+1} using ϕ^n via the following equation:

$$\frac{\phi^{n+1}}{\Delta t} - \beta \Delta \phi^{n+1} = \frac{\phi^n}{\Delta t} - \beta \Delta \phi^n + f(d, \phi^n) \nabla \cdot \left[d^2(\mathbf{x}) \frac{\nabla \phi^n}{|\nabla \phi^n|} \right], \tag{7}$$

where Δt is the time-step. This equation can be efficiently solved by the Fast Fourier Transform (FFT). Denoting the discrete Fourier transform by $\mathcal{F}$ and its inverse by $\mathcal{F}^{-1}$, we have:

$$\mathcal{F}(\phi)(i \pm 1, j) = e^{\pm 2\pi \sqrt{-1}(i-1)/M} \mathcal{F}(\phi)(i, j), \quad \mathcal{F}(\phi)(i, j \pm 1) = e^{\pm 2\pi \sqrt{-1}(j-1)/N} \mathcal{F}(\phi)(i, j).$$

Accordingly, the discrete Fourier transform of $\Delta \phi$ is

$$\mathcal{F}(\Delta \phi)(i, j) = \left[2 \cos(\pi \sqrt{-1}(i-1)/M) + 2 \cos(\pi \sqrt{-1}(j-1)/N) - 4 \right] \mathcal{F}\phi(i, j).$$

3

Here the coefficient in front of $\mathcal{F}\phi(i,j)$ represents the diagonalized discrete Laplacian operator in the frequency domain. Let g_1 be the right side of (7), then the solution $\phi^{n+1}(i,j)$ of (7) is computed via

$$\phi^{n+1}(i,j) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(g_1)(i,j)}{\left(1 - \beta\Delta t\left[2\cos(\pi\sqrt{-1}(i-1)/N) + 2\cos(\pi\sqrt{-1}(j-1)/N) - 4\right]\right)}\right). \quad (8)$$

As for the stopping criterion, we exploit the mean relative change of the weighted minimum surface energy (1). At the nth iteration, the algorithm terminates if

$$\frac{|\bar{e}_{n-1}^k - \bar{e}_n^k|}{\bar{e}_n^k} < 10^{-4}, \quad \text{where} \quad \bar{e}_n^k = \frac{1}{k}\sum_{i=n-k}^{n} E_p(\phi^i). \quad (9)$$

Here the quantity $\bar{e}_n^k$ represents the average of the energy values computed from the $(n-k)^{\text{th}}$ to the nth iteration for some $k \in \mathbb{N}$, $k \geq 1$. We fix $k = 10$, and set $p = 2$ for SIM. We summarize the main steps of SIM in Algorithm 1.

Algorithm 1: SIM for the weighted minimum surface (3)

Initialization: d, ϕ^0 and $n = 0$.
while *the stopping criterion* (9) *with* $p = 2$ *is greater than* 10^{-4} **do**
  | Update ϕ^{n+1} from ϕ^n solving (8);
  | Update $n \leftarrow n + 1$;
end
Output: ϕ^n such that $\{\phi^n = 0\}$ approximates $\{\phi^* = 0\}$.

2.2 Augmented Lagrangian Method (ALM) to Minimize $\mathbf{E_1}$

In this section, we present an augmented Lagrangian-based method to minimize the weighted minimum surface energy (2) for $p = 1$, i.e.,

$$E_1(\phi) = \int_\Omega d(\mathbf{x})\delta(\phi)|\nabla\phi|\,\mathrm{d}\mathbf{x}. \quad (10)$$

For the non-differentiable term $|\nabla\phi|$ in (10), we utilize the variable-splitting and introduce an auxiliary variable $\mathbf{p} = \nabla\phi$. We rephrase the minimization of $E_1(\phi)$ as a constrained optimization problem:

$$\{\phi^*, \mathbf{p}^*\} \leftarrow \arg\min_{\phi,\mathbf{p}} \int_\Omega \frac{\varepsilon d|\mathbf{p}|}{\pi(\varepsilon^2 + \phi^2)}\,\mathrm{d}\mathbf{x}, \quad \text{subject to} \quad \mathbf{p} = \nabla\phi, \quad (11)$$

here we replace $\delta(\phi)$ by its smooth approximation $H_\varepsilon'(\phi)$ as in (5). To solve problem (11), we formulate the augmented Lagrangian function:

$$\mathcal{L}(\phi, \mathbf{p}, \boldsymbol{\lambda}; r) = \int_\Omega \frac{\varepsilon d|\mathbf{p}|}{\pi(\varepsilon^2 + \phi^2)}\,\mathrm{d}\mathbf{x} + \frac{r}{2}\int_\Omega |\mathbf{p} - \nabla\phi|^2\,\mathrm{d}\mathbf{x} + \int_\Omega \boldsymbol{\lambda}\cdot(\mathbf{p} - \nabla\phi)\,\mathrm{d}\mathbf{x}, \quad (12)$$

where $r > 0$ is a scalar penalty parameter and $\boldsymbol{\lambda} : \mathbb{R}^{m+1} \to \mathbb{R}^{m+1}$ represents the Lagrangian multiplier. Minimizing (12) amounts to considering the following saddle-point problem:

$$\begin{aligned}
&\text{Find } (\phi^*, \mathbf{p}^*, \boldsymbol{\lambda}^*) \in \mathbb{R} \times \mathbb{R}^{m+1} \times \mathbb{R}^{m+1} \\
&\text{s.t. } \mathcal{L}(\phi^*, \mathbf{p}^*, \boldsymbol{\lambda}; r) \leq \mathcal{L}(\phi^*, \mathbf{p}^*, \boldsymbol{\lambda}^*; r) \leq \mathcal{L}(\phi, \mathbf{p}, \boldsymbol{\lambda}^*; r); \\
&\quad \forall (\phi, \mathbf{p}, \boldsymbol{\lambda}) \in \mathbb{R} \times \mathbb{R}^{m+1} \times \mathbb{R}^{m+1}.
\end{aligned} \quad (13)$$

Given ϕ^n, $\mathbf{p}^n$, and $\boldsymbol{\lambda}^n$, for $n = 0, 1, 2, \cdots$, the $(n+1)^{\text{th}}$ iteration of an ADMM-type algorithm for (13) consists of solving a series of sub-problems:

$$\phi^{n+1} = \arg\min_{\phi} \mathcal{L}(\phi, \mathbf{p}^n, \boldsymbol{\lambda}^n; r); \tag{14}$$

$$\mathbf{p}^{n+1} = \arg\min_{\mathbf{p}} \mathcal{L}(\phi^{n+1}, \mathbf{p}, \boldsymbol{\lambda}^n; r); \tag{15}$$

$$\boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^n + r \left(\mathbf{p}^{n+1} - \nabla \phi^{n+1} \right). \tag{16}$$

Each sub-problem can be solved efficiently. First, we find the minimizer of the sub-problem (14) by solving its Euler-Lagrange equation:

$$-r\Delta\phi^{n+1} = \frac{2d\varepsilon|\mathbf{p}^n|\phi^n}{\pi(\varepsilon^2 + (\phi^n)^2)^2} - \nabla \cdot (r\mathbf{p}^n + \boldsymbol{\lambda}^n). \tag{17}$$

Following [3], we introduce a frozen-coefficient term $\eta\phi$, for $\eta > 0$, on both sides of (17) to stabilize the computation; thus, (14) is solved using the following equation:

$$\eta\phi^{n+1} - r\Delta\phi^{n+1} = \eta\phi^n + \frac{2d\varepsilon|\mathbf{p}^n|\phi^n}{\pi(\varepsilon^2 + (\phi^n)^2)^2} - \nabla \cdot (r\mathbf{p}^n + \boldsymbol{\lambda}^n). \tag{18}$$

Here Δ is the Laplacian operator, and we solve this via FFT, similar to (8) for SIM. Thus, the ϕ sub-problem is solved via:

$$\phi^{n+1}(i,j) = \mathcal{F}^{-1}\left(\frac{\mathcal{F}(g_2)(i,j)}{\left(\eta - r\left[2\cos(\pi\sqrt{-1}(i-1)/N) + 2\cos(\pi\sqrt{-1}(j-1)/N) - 4 \right] \right)} \right). \tag{19}$$

Second, the $\mathbf{p}$ sub-problem (15) is equivalent to a weighted Total Variation (TV) minimization, whose solution admits a closed-form expression using the shrinkage operator [35]. Explicitly, the updated $\mathbf{p}^{n+1}$ is computed via:

$$\mathbf{p}^{n+1} = \max\left\{ 0, 1 - \frac{d\,\varepsilon}{\pi(\varepsilon^2 + (\phi^{n+1})^2)|r\nabla\phi^{n+1} - \boldsymbol{\lambda}^n|} \right\} \left(\nabla\phi^{n+1} - \frac{\boldsymbol{\lambda}^n}{r} \right). \tag{20}$$

Finally, the Lagrangian multiplier $\boldsymbol{\lambda}$ is updated by (16). The stopping criterion for the ALM iteration is the same as that for SIM (9), but with $p = 1$. We summarize the main steps of ALM in Algorithm 2.

Algorithm 2: ALM for the weighted minimum surface (10)

Initialization: d, ϕ^0, $\mathbf{p}^0$, $\boldsymbol{\lambda}^0$, and $n = 0$.
while *the stopping criterion* (9) *with $p = 1$ is greater than* 10^{-4} **do**
    Update $\phi^{n+1} = \arg\min_{\phi} \mathcal{L}(\phi, \mathbf{p}^n, \boldsymbol{\lambda}^n; r)$ via (19) ;
    Update $\mathbf{p}^{n+1} = \arg\min_{\mathbf{p}} \mathcal{L}(\phi^{n+1}, \mathbf{p}, \boldsymbol{\lambda}^n; r)$ via (20);
    Update $\boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^n + r(\mathbf{p}^{n+1} - \nabla\phi^{n+1})$;
    Update $n \leftarrow n + 1$;
end
Output: ϕ^n such that $\{\phi^n = 0\}$ approximates $\{\phi^* = 0\}$.

2.3 Connection between SIM and ALM algorithms

Note that both SIM and ALM involve solving elliptic PDEs of the form:

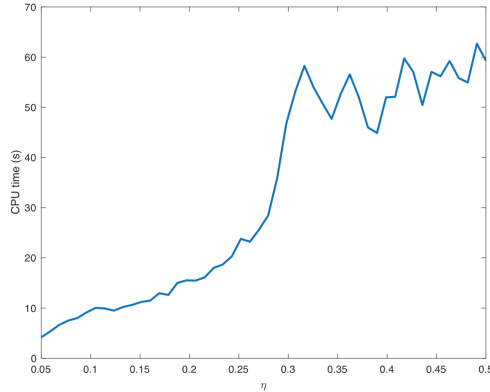$$a\phi - b\Delta\phi = g, \tag{21}$$

Figure 2: The CPU-time (s) of ALM until convergence, for the five-fold circle point cloud in Figure 1 (a). Here $r = \varepsilon = 1$ and η varies from 0.05 to 0.5. The connection between SIM and ALM indicates that large η slows down ALM. In this graph, as η increases, the time required to reach the convergence increases.

for some constants $a, b > 0$, and a function g defined on Ω. For SIM, it is equation (7):

$$\underbrace{\frac{1}{\Delta t}}_{a} \phi^{n+1} - \underbrace{\beta}_{b} \Delta\phi^{n+1} = \underbrace{\frac{\phi^n}{\Delta t} - \beta\Delta\phi^n + f(d, \phi^n)\nabla \cdot \left[d^2(\mathbf{x})\frac{\nabla\phi^n}{|\nabla\phi^n|}\right]}_{g},$$

and for ALM, it is equation (18):

$$\underbrace{\eta}_{a} \phi^{n+1} - \underbrace{r}_{b} \Delta\phi^{n+1} = \underbrace{\eta\phi^n + \frac{2d\varepsilon|\mathbf{p}^n|\phi^n}{\pi(\varepsilon^2 + (\phi^n)^2)^2} - \nabla \cdot (r\mathbf{p}^n + \boldsymbol{\lambda}^n)}_{g}.$$

We remark interesting connections between SIM and ALM. First, both methods have stabilizing terms but in different positions on the left side of (21). For SIM, it is $-\beta\Delta\phi$, while for ALM, it is $\eta\phi$. Second, relating the coefficients of ϕ, $1/\Delta t$ in SIM gives insight to the effect of η in ALM. In general, a large η slows down the convergence of ALM, while a small η accelerates it (as the effect of $\frac{1}{\Delta t}$ on SIM). Figure 2 shows convergence behaviors of ALM for different η, using the five-fold circle point cloud in Figure 1 (a). It displays the CPU time (in seconds) for $r = 1$, $\varepsilon = 1$, and η varying from 0.05 to 0.5. Note that as η increases, the time required to reach the convergence increases almost quadratically at first, then stays around the same level. Third, the correspondence between $b = \beta$ in SIM, and $b = r$ in ALM allows another interpretation of the parameter r in ALM. In SIM, a large β smears the solution and avoids discontinuities or sharp corners, and for ALM, large r also allows to pass through fine details. Figure 7 in Section 3 presents more details, where we experiment with different r and ε values for the five-fold circle point cloud shown in Figure 1 (a).

3 Numerical Implementations, Experiments and Effects of Parameters

In this section, we describe the implementation details and present numerical experiments. For both SIM and ALM, we vary ε from 0.5 to 1. For SIM, we use $\Delta t = 500$. When $\mathcal{D}$ is in 2D, we set $\beta = 0.1$, and $\beta = 0.01$ for 3D. For ALM, η ranges from 0.05 to 1, and r from 0.5 to 2.

The code is written in MATLAB and executed without additional machine support, e.g. parallelization or GPU-enhanced computations. All the experiments are performed on Intel® Core™4-Core 1.8GHz (4.0GHz with Turbo) machine, with 16 GB/RAM and Intel® UHD

6

Graphics 620 graphic card under Windows OS. The contours and isosurfaces are displayed using MATLAB visualization engine. No post-processing, e.g., smoothing or sharpening, is applied.

3.1 Implementation Details

We illustrate the details for planar point clouds, i.e., $\mathcal{D} \subseteq \mathbb{R}^2$, and the extension to $\mathbb{R}^3$ is straightforward. Let the computational domain $\Omega = [0, M] \times [0, N]$, $M, N > 0$, be discretized by a Cartesian grid with $\Delta x = \Delta y = 1$. For any function u (or a vector field $\mathbf{v} = (v^1, v^2)$) defined on Ω, we use $u_{i,j}$ or $u(i, j)$ to denote $u(i\Delta x, i\Delta y)$. We use the usual backward and forward finite difference schemes:

$$\partial_1^- u_{i,j} = \begin{cases} u_{i,j} - u_{i-1,j}, \ 1 < i \le M; \\ u_{1,j} - u_{M,j}, \ i = 1. \end{cases} \qquad \partial_1^+ u_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j}, \ 1 \le i < M - 1; \\ u_{1,j} - u_{M,j}, \ i = M. \end{cases}$$

$$\partial_2^- u_{i,j} = \begin{cases} u_{i,j} - u_{i,j-1}, \ 1 < j \le N; \\ u_{i,1} - u_{i,N}, \ j = 1. \end{cases} \qquad \partial_2^+ u_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j}, \ 1 \le j < N - 1; \\ u_{i,1} - u_{i,N}, \ j = N. \end{cases}$$

The gradient, divergence and the Laplacian operators are approximated as follows:

$$\nabla u_{i,j} = ((\partial_1^- u_{i,j} + \partial_1^+ u_{i,j})/2, (\partial_2^- u_{i,j} + \partial_2^+ u_{i,j})/2);$$
$$\nabla \cdot \mathbf{v}_{i,j} = (\partial_1^+ v_{i,j}^1 + \partial_1^- v_{i,j}^1)/2 + (\partial_2^+ v_{i,j}^2 + \partial_2^- v_{i,j}^2)/2;$$
$$\Delta u_{i,j} = \partial_1^+ u_{i,j} - \partial_1^- u_{i,j} + \partial_2^+ u_{i,j} - \partial_2^- u_{i,j}.$$

The distance function d is computed once at the beginning and no update is needed. It satisfies an Eikonal equation:

$$\begin{cases} |\nabla d| = 1 \text{ in } \Omega, \\ d(\mathbf{x}) = 0 \text{ for } \mathbf{x} \in \mathcal{D}, \end{cases} \tag{22}$$

and discretizing (22) via the Lax-Friedrich scheme leads to an updating formula:

$$d_{i,j}^{n+1} = \frac{1}{2}\left(1 - |\nabla d_{i,j}^n| + \frac{d_{i+1,j}^n + d_{i-1,j}^n}{2} + \frac{d_{i,j+1}^n + d_{i,j-1}^n}{2}\right). \tag{23}$$

We solve (23) using the fast sweeping method [19] with complexity $O(G)$ for G grid points.

Keeping ϕ^n to be a signed distance function during the iteration improves the stability of level-set-based algorithms. We reinitialize ϕ^n at the nth iteration by solving the following PDE:

$$\begin{cases} \psi_\tau + \text{sign}(\psi)(|\nabla \psi| - 1) = 0, \\ \psi(\mathbf{x}, 0) = \phi^n. \end{cases} \tag{24}$$

Here subscript τ represents artificial time and partial derivative, and sign $: \mathbb{R} \to \{-1, 0, 1\}$ is the sign function. In practice, ψ^n being a signed distance function near 0-level-set is important; thus, it is sufficient to solve (24) for a few steps. We fix 10 steps of reinitialization throughout this paper.

3.2 Numerical Experiments of 2D and 3D Point Clouds

The first experiment, Figure 3, is reconstruction of planar curves from 2D point clouds confined within a square $\Omega = [0, 100]^2 \subset \mathbb{R}^2$. We generate the data using four different shapes: a triangle, an ellipse, a square whose corners are missing, and a five-fold circle. For these cases, we use a centered circle with radius 30 as the initial guess, shown in Figure 3 (a). Figure 3 (b) and (c) display the given $\mathcal{D}$, as well as the curves identified by SIM and ALM with $r = 1.5$, respectively. Both methods produce comparably accurate results. In the triangle example, corners get as
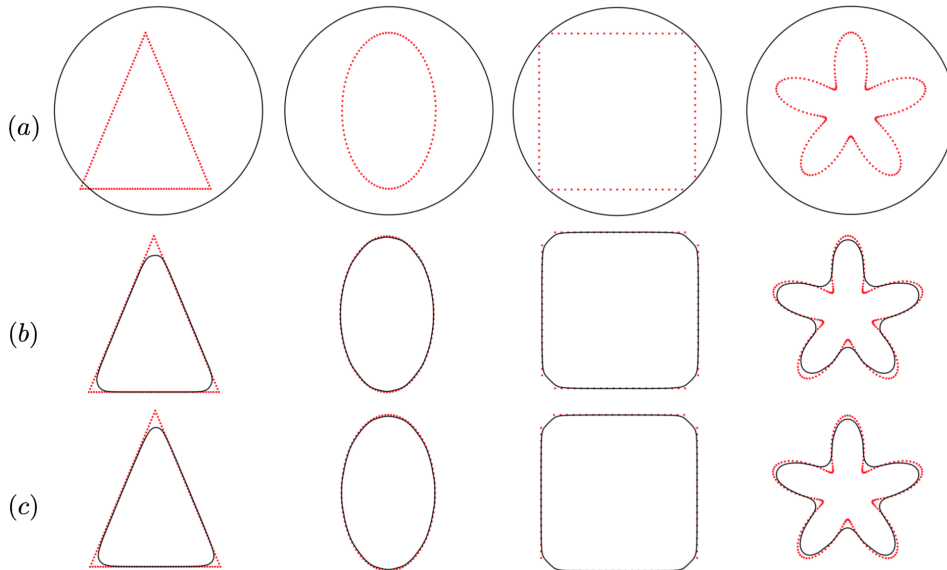
Figure 3: The test point clouds: triangle with 150 number of points, ellipse with 100 points, square with 80 points, and five-fold-circle with 200 points. (a) The top row, identical initial condition applied to SIM and ALM for different $\mathcal{D}$. (b) The middle row, the results obtained by SIM. (c) The bottom row, the results obtained by ALM using $r = 1.5$. Both methods give compatible results.

close as the approximated delta function (with parameter ε) allows for both methods. The ellipse and square results fit very closely to the given point clouds. For the five-fold-circle, there is a slight difference in how the curve fits the edges, yet the results are very compatible.

Table 1 shows the CPU time (in seconds) for SIM, ALM using $r = 0.5, 1, 1.5$, and 2, and the explicit method in [38] with $\Delta t = 20$ on the same data sets. With proper choices of r, ALM outperforms the other methods in terms of computational efficiency. SIM is stable without any dependency on the choice of parameters, and its run-times are comparable to the best performances of ALM in most cases. Both methods are faster than the explicit method in all the examples, and computational efficiency is similar between ALM and SIM.

Object	ALM($r = 0.5$)	ALM($r = 1$)	ALM($r = 1.5$)	ALM($r = 2$)	SIM	[38]
Triangle	–	1.45	1.31	1.48	1.50	5.25
Ellipse	1.22	1.03	1.33	1.37	1.49	3.89
Square	–	–	0.94	1.20	1.09	2.07
Five-fold circle	0.83	1.44	1.86	1.22	1.96	4.18

Table 1: CPU time (s) for SIM, ALM using $r = 0.5, 1, 1.5$, and 2, and the explicit method in [38] with $\Delta t = 20$ for the point cloud data sets in Figure 3. Both SIM and ALM shows fast convergence.

The second set of experiments reconstruct surface from the point cloud $\mathcal{D}$ in 3D: a jar in Figure 1 (b) and a torus in Figure 1 (c) within $\Omega = [0, 50]^3$. In Figure 4, we show the reconstructed surfaces using SIM and ALM. A portion of the given point cloud is superposed for validation in each case. Both methods successfully capture the overall shapes and non-convex features of the jar, as well as the torus. There are only slight differences in the reconstruction between using SIM with $p = 2$ and using ALM with $p = 1$.

Table 2 shows the efficiency of SIM and ALM compared to the explicit method in [38] for the experiments in Figure 4. Thanks to the semi-implicit scheme, Δt can be large and we used
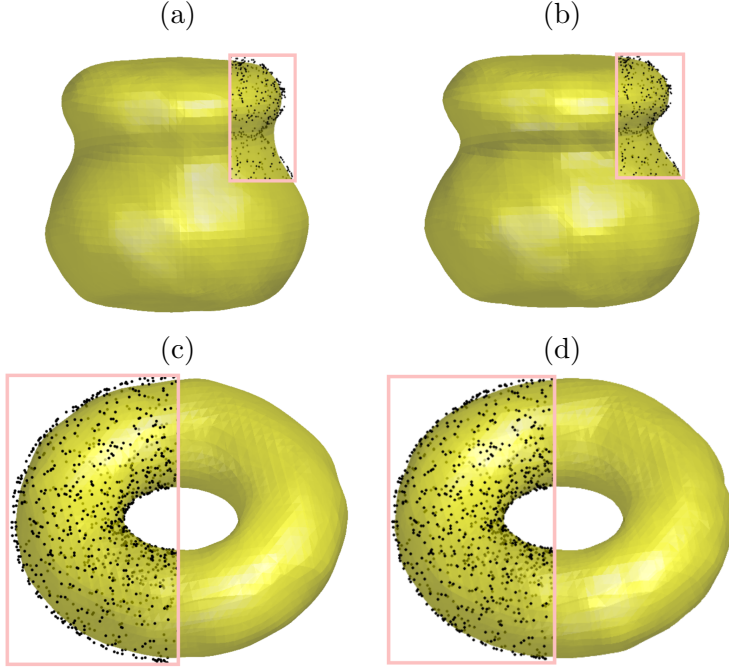
Figure 4: The first row shows ALM and SIM applied to the 3D jar point cloud in Figure 1 (b). (a) The result of ALM with $r = 1.3, \varepsilon = 0.5, \eta = 0.6$. (b) The result of SIM. The second row shows the methods applied to the 3D torus point cloud in Figure 1 (c). (c) The result of ALM with $r = 1.3, \varepsilon = 0.5, \eta = 0.6$. (d) The result of SIM. Both methods are compatible and shows good results.

500 in SIM; in the explicit method, we are forced to use much smaller time step $\Delta t = 20$ to maintain the stability. The improvement of run-time in ALM is carefully controlled by the parameters r, ε and η. We choose $r = 1.3$, $\varepsilon = 0.5$ and $\eta = 0.6$ for both cases. Both SIM and ALM efficiently provide accurate reconstruction.

Object	ALM	SIM	[38]
Jar	29.69	29.42	74.44
Torus	47.32	33.58	114.20

Table 2: CPU time (s) of SIM and ALM compared to the explicit method in [38] for the point cloud data sets of Figure 4. Both SIM and ALM show fast convergence.

The third set of examples show the effect of the distance function d. Notice that the weighted minimal surface energy (1) is mainly driven by the distance function d, that is, the given point cloud $\mathcal{D}$ determines the landscape of d, which affects the behavior of the level-set during the evolution. Figure 5 shows the evolution using ALM, applied to different subsets of point clouds sampled from the same bunny face shape. The densities of the point cloud vary for the three different regions: the face with n_1 points, the head with n_2 points, and each ear with n_3 points. Figure 5 (a) shows the given point cloud for $(n_1, n_2, n_3) = (20, 10, 20)$, with the 0-level-set of ϕ^n at 15th iteration, (b) for $(n_1, n_2, n_3) = (50, 10, 20)$, at 18th iteration, and(c) for $(n_1, n_2, n_3) = (20, 10, 40)$, at 20th iteration. These three curves eventually degenerate to a point. (d) for $(n_1, n_2, n_3) = (50, 10, 40)$ and shows the converged solution. In (a)–(c), denser parts of the point cloud attract the curve with stronger forces, the sparser parts of the point cloud fail to lock the curve. Then, the energy model (2) drives curves to have short lengths, i.e., the level set tends to shrink. In (d), with a more balanced distribution of points, the curve converges to correct shape.
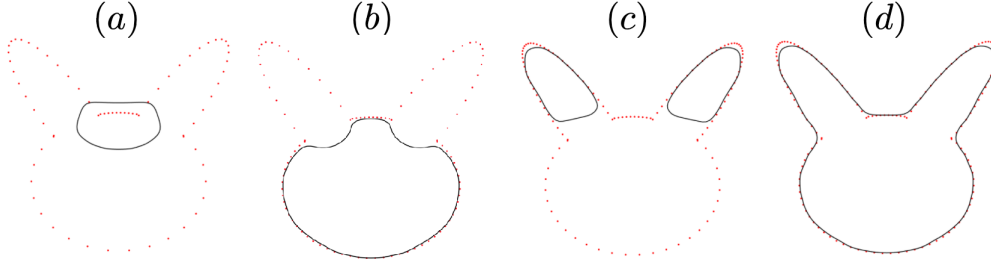
9

Figure 5: The effect of the distance function for varying-density point clouds: the face with n_1 points, the head with n_2 points, and each ear with n_3 points. (a) the given point cloud is with $(n_1, n_2, n_3) = (20, 10, 20)$, and shows the 0-level-set of ϕ^n at 15th iteration, (b) $(n_1, n_2, n_3) = (50, 10, 20)$, and shows 18th iteration, and (c) $(n_1, n_2, n_3) = (20, 10, 40)$, and shows 20th iteration. These three curves eventually degenerate to a point. (d) is with $(n_1, n_2, n_3) = (50, 10, 40)$ and shows the converged solution. The potential energy (1) is mainly driven by the distance function d, which affects the level-set evolution.

The fourth set of examples demonstrate the robustness of ALM and SIM against noise. Figure 6 shows the reconstructed curves from clean and noisy data: (a)-(c) are results of ALM, and (d)-(f) are results of SIM. (a) and (d) in the first column show results obtained from the clean data, which has 200 points sampled from a three-fold circle. Gaussian noise with standard deviation 1 is added to both x and y coordinates to generate noisy point cloud in the second column, (b) and (e). To show the differences, the third column superposes both results reconstructed from clean and noisy point clouds. Both ALM and SIM provide compatible results. For the noisy data, although the reconstructed curves show some oscillation, they are very close to the solutions using the clean data, respectively.

3.3   Choice of Parameters for ALM and its Effects

The proposed ALM has one parameter r, and the model (2) uses the delta function, where the smoothness parameter ε is added to stabilize the computation. Both parameters have straightforward effects on the level-set evolution from (17). For example, consider a set of points within a thin-band around the 0-level-set of ϕ^n, denoted by $B_\varepsilon = \{\mathbf{x} \mid -2\varepsilon/\sqrt{3} < \phi^n(\mathbf{x}) < 2\varepsilon/\sqrt{3}\}$. By the continuity of ϕ^n, there exist $\mathbf{y}$ and $\mathbf{z} \in B_\varepsilon$ such that $\phi^n(\mathbf{y}) = -\varepsilon/\sqrt{3}$ and $\phi^n(\mathbf{z}) = \varepsilon/\sqrt{3}$; these values are the minimum and maximum of the function $h(x) = \frac{2\varepsilon x}{\pi(\varepsilon^2+x^2)^2}$, respectively. At these points, (17) takes the following forms:

$$\Delta\phi^{n+1} = \begin{cases} -9\, d\, |\mathbf{p}^n|/(8\sqrt{3}\pi\, \varepsilon^2 r) + \nabla \cdot (\mathbf{p}^n + \frac{\lambda^n}{r}) \text{ at } \mathbf{y}. \\ 9\, d\, |\mathbf{p}^n|/(8\sqrt{3}\pi\, \varepsilon^2 r) + \nabla \cdot (\mathbf{p}^n + \frac{\lambda^n}{r}) \quad \text{at } \mathbf{z}. \end{cases} \tag{25}$$

The first terms in the right hand side of (25) show that with a smaller ε, there are less number of points in B_ε, but influence from d is stronger. With a larger ε, d affects more number of points in B_ε, but the influence becomes weaker. Varying r also modifies the effect of d, while the size of B_ε is not changed.

We also find that ε interacts with r and effectively modifies the shape of the level-set. Figure 7 shows the results for ALM using different combinations of r and ε, on the five-fold circle point cloud in Figure 1 (a). For a fixed r, increasing ε makes the approximated delta function smoother; consequently, narrow and elongated shapes are omitted, and the reconstructed curve becomes more convex. For a fixed ε, larger r loses more details, as discussed in Subsection 2.3. The speed of convergence varies for different combinations of r and ε. When the choices are reasonable, the algorithm converges fast within 2 seconds. When both r and ε are large, results are not as good, and the convergences are slow.
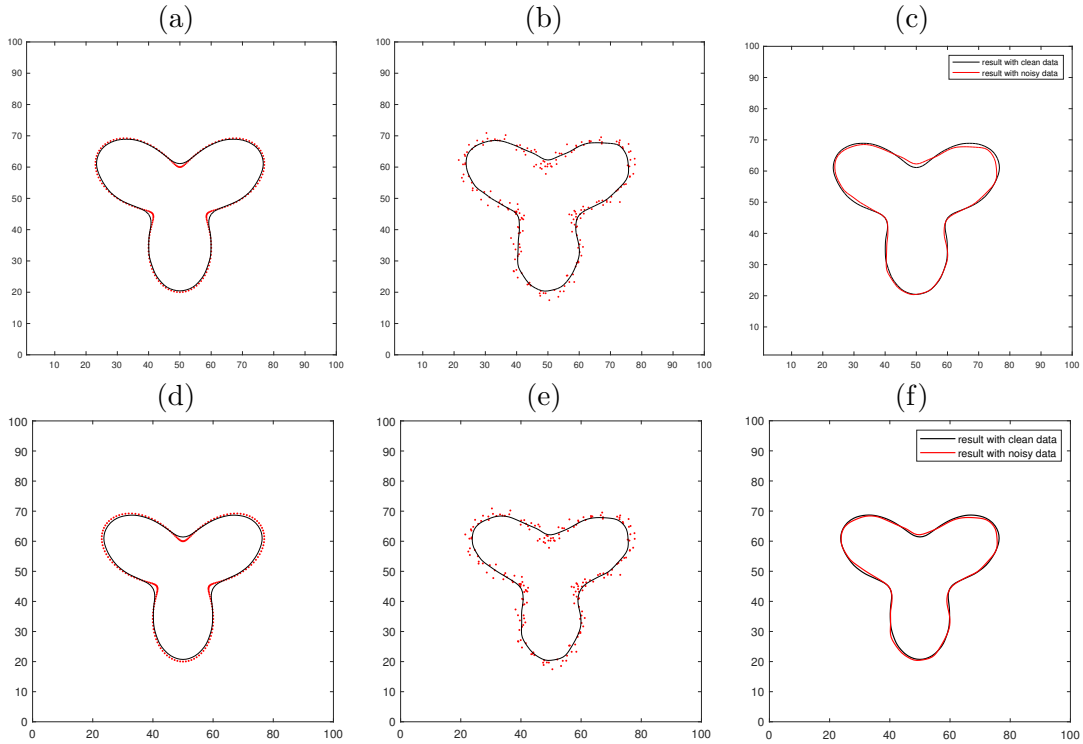
Figure 6: The influence of noise on reconstructing three-fold circle with 200 points: (a)-(c) ALM and (d)-(f) SIM. The first column shows the reconstructed curves from clean data, and the second column the reconstructions from noisy data. The third column shows the comparison between the two reconstructed curves in first two columns.
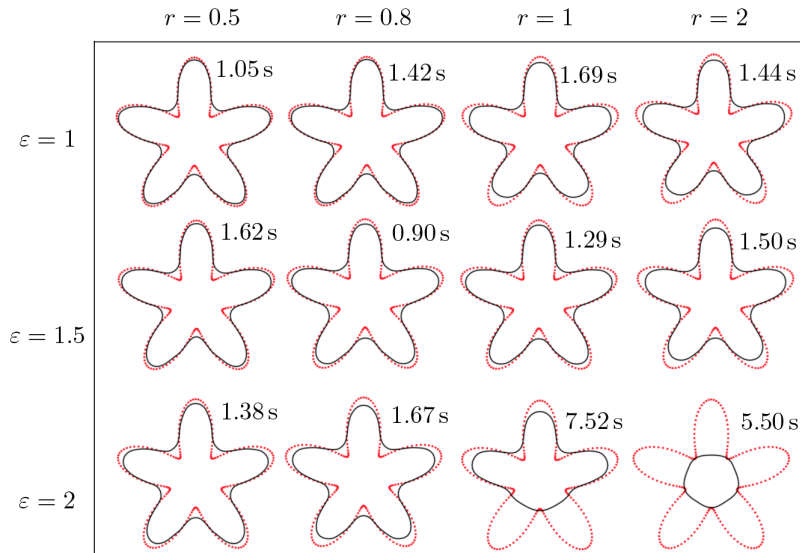

Figure 7: Results by ALM with different r and ε. For each column, from top to bottom, $\varepsilon = 1, 1.5, 2$; and for each row, from left to right, $r = 0.5, 0.8, 1, 2$. Increasing ε renders the curve less sharp and more convex. Increasing r induces a stronger diffusion effect on ϕ^n.

Another observation comes from (20). For any point $\mathbf{x}$ and $n \geq 0$, if the following value:

$$Q^n(\varepsilon, r) := \phi^n \pi |r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}| \varepsilon^2 - d\varepsilon + (\phi^n)^3 \pi |r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}|$$

is positive, then $\mathbf{p}^n(\mathbf{x}) = 0$, and d has no direct effect on (18) at $\mathbf{x}$ in the next iteration. Regarding $Q^n(\varepsilon, r)$ as a quadratic polynomial in terms of ε parameterized by r, the sign of $Q^n(\varepsilon, r)$ depends on the sign of $\phi^n(\mathbf{x})$ and the sign of its discriminant computed via:

$$\operatorname{Disc} Q^n = d^2 - 4(\phi^n)^4 \pi^2 |r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}|^2.$$

The sign of $\phi^n(\mathbf{x})$ is related to the position of $\mathbf{x}$ relative to the 0-level-set. The sign of $\operatorname{Disc} Q^n$ is determined by comparing the length of a vector difference $r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}$ with the quantity $d/(4(\phi^n)^2 \pi)$. By the projection theorem, $|r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}|^2$ is bounded below by $\alpha^n := |\boldsymbol{\lambda}^{n-1}|^2 - |\operatorname{Proj}_{\nabla \phi^n} \boldsymbol{\lambda}^{n-1}|^2 = |\boldsymbol{\lambda}^{n-1}|^2 - |\boldsymbol{\lambda}^{n-1} \cdot \nabla \phi^n|^2 / (|\boldsymbol{\lambda}^{n-1}|^2 |\nabla \phi^n|^2)$, i.e., the squared residual of orthogonal projection of $\boldsymbol{\lambda}^{n-1}$ onto $\nabla \phi^n$; therefore, we can decide the sign of $\operatorname{Disc} Q^n$ using r via the following cases:

1. When $\frac{d^2}{4(\phi^n)^4 \pi^2} < \alpha^n$, for any $r > 0$, $\operatorname{Disc} Q^n < 0$.

2. When $\frac{d^2}{4(\phi^n)^4 \pi^2} \geq \alpha^n$:

   (a) if $r > r_U^n$ or $r < r_L^n$, then $\operatorname{Disc} Q^n < 0$;
   (b) if $\max\{0, r_L^n\} \leq r \leq r_U^n$, then $\operatorname{Disc} Q^n \geq 0$.

   Here,

   $$r_U^n = \frac{|\operatorname{Proj}_{\nabla \phi^n} \boldsymbol{\lambda}^{n-1}| + \sqrt{\frac{d^2}{4(\phi^n)^4 \pi^2} - \alpha^n}}{|\nabla \phi^n|} \quad \text{and} \quad r_L^n = \frac{|\operatorname{Proj}_{\nabla \phi^n} \boldsymbol{\lambda}^{n-1}| - \sqrt{\frac{d^2}{4(\phi^n)^4 \pi^2} - \alpha^n}}{|\nabla \phi^n|}.$$

When $\phi^n(\mathbf{x}) > 0$, Q^n concaves upwards and $Q^n(0, r) \geq 0$ for any r. If $\operatorname{Disc} Q^n < 0$, Q^n is positive for all ε and d has no effect on level set evolution. If $\operatorname{Disc} Q^n \geq 0$, Q^n is positive for ε outside the interval bounded by two roots of Q^n, i.e.,

$$0 < \varepsilon < \frac{d - \sqrt{\operatorname{Disc} Q^n}}{2\phi^n \pi |r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}|} \quad \text{or} \quad \varepsilon > \frac{d + \sqrt{\operatorname{Disc} Q^n}}{2\phi^n \pi |r \nabla \phi^n - \boldsymbol{\lambda}^{n-1}|}.$$

When $\phi^n(\mathbf{x}) < 0$, Q^n concaves downwards, and $Q^n(0, r) \leq 0$ for any r. In this case, Q^n is never positive: either $\operatorname{Disc} Q^n < 0$, i.e., no roots, or $\operatorname{Disc} Q^n \geq 0$ but both roots are negative.

Notice that the bounds, r_L^n and r_U^n, are closely related to the ratio $d/(\phi^n)^2$, which contributes to the adaptive behavior of ALM. For example, for a point $\mathbf{x}$ where $\phi^n(\mathbf{x}) > 0$, when $|\phi^n(\mathbf{x})|$ is close to 0 but $d(\mathbf{x}) \gg 0$, $r_L^n < 0$ and r_U^n becomes extremely large; thus, for a moderate value of r, d has strong influence on the evolution of the level-set near $\mathbf{x}$ and swiftly moves the curve towards the point cloud. For a point $\mathbf{x}$ which is close to both $\mathcal{D}$ and $\{\phi^n = 0\}$, the level-set evolution becomes more stringent about the minimization of the energy (10).

Figure 8 illustrates this effect, for the five-fold circle point cloud in Figure 1 (a) with $r = 2$ and $\varepsilon = 1$. Figure 8 shows (a) $\operatorname{Disc} Q^n$, (b) r_U^n, (c) r_L^n, and (d) the region where d effects the level set evolution. The figures are for iterations $n = 2, 3, 4, 7, 8, 10, 11, 13$ and 38 (converged). The region inside $\{\phi^n = 0\}$ always experiences the influence of d, as described above. Figure 8 (a) shows that the region outside $\{\phi^n = 0\}$ is mostly blue indicating $\operatorname{Disc} Q^n < 0$; hence, for almost every point outside the 0-level-set, as long as $r_L^n \leq r \leq r_U^n$, the landscape of d has strong effects on the evolution. In (b) and (c), observe that high values of r_U^n only concentrate near the 0-level-set while r_L^n remains relatively small in the whole domain; thus, the influence of d is strong near $\{\phi^n = 0\}$. (d) displays the white regions where d explicitly guides the level-set evolution and the black regions where d has no direct effect. These results show that, although ALM evolves the level-set globally, it ignores the effects of d when evolving the regions far away from the level-sets; and it utilizes d to refine the local structures for the regions of the level-sets close to $\mathcal{D}$.
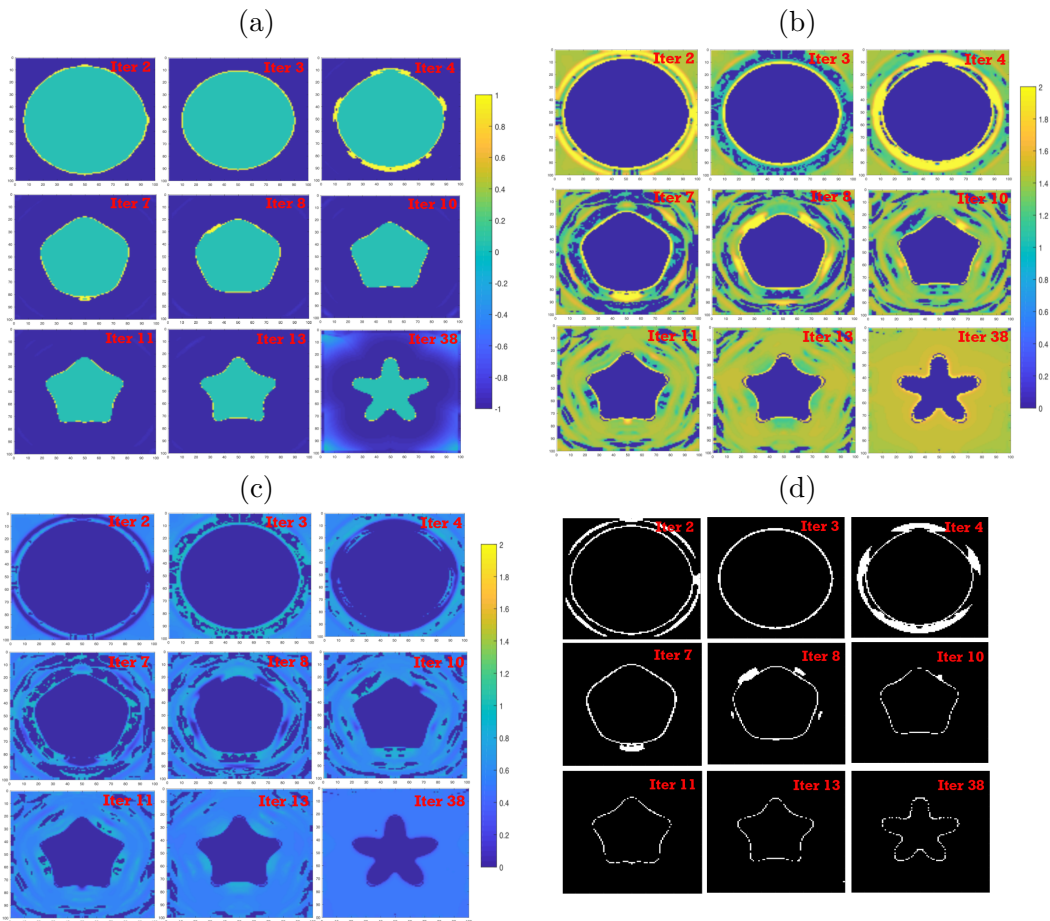
Figure 8: (a) Disc Q^n, (b) r_U^n, (c) r_L^n at certain iterations. (d) The region (in white) where d explicitly guides the level-set evolution by ALM. The distance function d refines the local structures and it is only active near $\{\phi^n = 0\}$. This partially explains the efficiency of ALM.

4 Conclusion

We propose two fast algorithms, SIM and ALM, to reconstruct m-dimensional manifold from unstructured point clouds by minimizing the weighted minimum surface energy (2). SIM improves the computational efficiency by relaxing the constraint on the time-step using a semi-implicit scheme. ALM follows an augmented Lagrangian approach and solves the problem by an ADMM-type algorithm. Numerical experiments show that the proposed algorithms are superior at the computational speed, and both of them produce accurate results. Theoretically, we demonstrate the delicate interaction among parameters involved in ALM, and show the connections between SIM and ALM. This explains the behaviors of ALM from the perspective of SIM.

References

[1] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Point set surfaces. In *Proceedings of the Conference on Visualization'01*, pages 21–28. IEEE Computer Society, 2001.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleishman, D. Levin, and C. T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):3–15, January 2003.

[3] E. Bae, X.-C. Tai, and W. Zhu. Augmented Lagrangian method for an Euler's elastica based segmentation model that promotes convex contours. *Inverse Problems & Imaging*, 11(1):1–23, 2017.

[4] Z. Bi and L. Wang. Advances in 3D data acquisition and processing for industrial applications. *Robotics and Computer-Integrated Manufacturing*, 26(5):403–413, 2010.

[5] M. Bolitho, M. Kazhdan, R. Burns, and H. Hoppe. Parallel poisson surface reconstruction. In *International symposium on visual computing*, pages 678–689. Springer, 2009.

[6] R. Bracewell and R. Bracewell. *The Fourier Transform and Its Applications*. Electrical Engineering Series. McGraw Hill, 2000.

[7] X. Bresson, S. Esedoḡlu, P. Vandergheynst, J.-P. Thiran, and S. Osher. Fast global minimization of the active contour/snake model. *Journal of Mathematical Imaging and vision*, 28(2):151–167, 2007.

[8] F. Calakli and G. Taubin. SSD: Smooth signed distance surface reconstruction. In *Computer Graphics Forum*, volume 30, pages 1993–2002. Wiley Online Library, 2011.

[9] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, pages 67–76. ACM, 2001.

[10] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth surface reconstruction from noisy range data. In *Proceedings of the 1st International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 119–ff. ACM, 2003.

[11] G. Casciola, D. Lazzaro, L. B. Montefusco, and S. Morigi. Shape preserving surface reconstruction using locally anisotropic radial basis function interpolants. *Computers & Mathematics with Applications*, 51(8):1185–1198, 2006.

[12] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[13] H. Q. Dinh, G. Turk, and G. Slabaugh. Reconstructing surfaces using anisotropic basis functions. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 606–613. IEEE, 2001.

[14] V. Estellers, M. Scott, K. Tew, and S. Soatto. Robust poisson surface reconstruction. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 525–537. Springer, 2015.

[15] V. Estellers, D. Zosso, R. Lai, S. Osher, J.-P. Thiran, and X. Bresson. Efficient algorithm for level set method preserving distance function. *IEEE Transactions on Image Processing*, 21(12):4722–4734, 2012.

[16] L. Gomes, O. R. P. Bellon, and L. Silva. 3D reconstruction methods for digital preservation of cultural heritage: A survey. *Pattern Recognition Letters*, 50:3–14, 2014.

[17] J. Haličková and K. Mikula. Level set method for surface reconstruction and its application in surveying. *Journal of Surveying Engineering*, 142(3):04016007, 2016.

[18] H. Huang, D. Li, H. Zhang, U. Ascher, and D. Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 28(5):176, 2009.

[19] C. Y. Kao, S. Osher, and J. Qian. Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations. *Journal of Computational Physics*, 196(1):367–391, 2004.

[20] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing*, volume 7, pages 61–70, 2006.

[21] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(3):29, 2013.

[22] D. Khan, M. A. Shirazi, and M. Y. Kim. Single shot laser speckle based 3D acquisition system for medical applications. *Optics and Lasers in Engineering*, 105:43–53, 2018.

[23] H. Li, Y. Li, R. Yu, J. Sun, and J. Kim. Surface reconstruction from unorganized points with ℓ_0 gradient minimization. *Computer Vision and Image Understanding*, 169:108–118, 2018.

[24] X. Li, W. Wan, X. Cheng, and B. Cui. An improved Poisson surface reconstruction algorithm. In *2010 International Conference on Audio, Language and Image Processing*, pages 1134–1138. IEEE, 2010.

[25] J. Liang, F. Park, and H.-K. Zhao. Robust and efficient implicit surface reconstruction for point clouds based on convexified image segmentation. *Journal of Scientific Computing*, 54(2-3):577–602, 2013.

[26] H. Liu, X. Wang, and W. Qiang. Implicit surface reconstruction from 3D scattered points based on variational level set method. In *2008 2nd International Symposium on Systems and Control in Aerospace and Astronautics*, pages 1–5. IEEE, 2008.

[27] H. Liu, Z. Yao, S. Leung, and T. F. Chan. A level set based variational principal flow method for nonparametric dimension reduction on Riemannian manifolds. *SIAM Journal on Scientific Computing*, 39(4):A1616–A1646, 2017.

[28] S. Osher and R. P. Fedkiw. Level set methods: an overview and some recent results. *Journal of Computational Physics*, 169(2):463–502, 2001.

[29] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton–Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[30] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. In *Computer Graphics Forum*, volume 28, pages 493–501. Wiley Online Library, 2009.

[31] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid mechanics, Computer vision, and Materials Science*, volume 3. Cambridge university press, 1999.

[32] J. Shi, M. Wan, X.-C. Tai, and D. Wang. Curvature minimization for surface reconstruction with features. In *International Conference on Scale Space and Variational Methods in Computer Vision*, pages 495–507. Springer, 2011.

[33] Y. Shi and W. C. Karl. Shape reconstruction from unorganized points with a data-driven level set method. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages iii–13. IEEE, 2004.

[34] P. Smereka. Semi-implicit level set methods for curvature and surface diffusion motion. *Journal of Scientific Computing*, 19(1):439–456, 2003.

[35] X.-C. Tai, J. Hahn, and G. J. Chung. A fast algorithm for Euler's elastica model using augmented Lagrangian method. *SIAM Journal on Imaging Sciences*, 4(1):313–344, 2011.

[36] A. Tsai, A. Yezzi Jr, W. Wells, C. Tempany, D. Tucker, A. Fan, W. E. Grimson, and A. Willsky. A shape-based approach to the segmentation of medical imagery using level sets. *IEEE Transactions on Medical Imaging*, 22(2):137, 2003.

[37] H.-K. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction using the level set method. In *Proceedings IEEE Workshop on Variational and Level Set Methods in Computer Vision*, pages 194–201. IEEE, 2001.

[38] H.-K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit, nonparametric shape reconstruction from unorganized points using a variational level set method. *Computer Vision and Image Understanding*, 80(3):295–319, 2000.