An Efficient Iterative Method for Reconstructing Surface from Point Clouds

Dong Wang

Received: 8 December 2020 / Revised: 23 February 2021 / Accepted: 2 March 2021 / Published online: 15 March 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract
Surface reconstruction from point clouds is a fundamental step in many applications in computer vision. In this paper, we develop an efficient iterative method on a variational model for the surface reconstruction from point clouds. The surface is implicitly represented by indicator functions and the energy functional is then approximated based on such representations using heat kernel convolutions. We then develop a novel iterative method to minimize the approximate energy and prove the energy decaying property during each iteration. Asymptotic expansion is also performed to illustrate the dynamics of the surface during iterations. Extensive numerical experiments are performed in both 2- and 3-dimensional Euclidean spaces to show that the proposed method is simple, efficient, and accurate.

Keywords Iterative method · Thresholding · Surface reconstruction · Point cloud

1 Introduction
The problem of reconstructing surfaces from point clouds has attracted tremendous attention for the past decades [3]. Point clouds are usually obtained using optical measuring devices such as laser scanners. It is a fundamental step in many applications such as computer graphics [4,38], medical imaging [17], manufacturing applications [2], and many others [1,3].

In this paper, we consider the reconstruction of $n - 1$-dimensional manifold from a point cloud $C \in \mathbb{R}^n$ (for example, a curve in $\mathbb{R}^2$ or a surface in $\mathbb{R}^3$). To be specific, motivated from [40], we consider the following optimization problem:

$$\Gamma^* = \min_{\Gamma} E(\Gamma) = \left( \int_{\Gamma} |d|^{p} \ ds \right)^{\frac{1}{p}}$$

(1)

where $d(x) = \min_{y \in C} |x - y|$ is the distance from any point $x \in \mathbb{R}^n$ to the point cloud $C$, $p$ is a positive number, $ds$ is the line/surface integral element, and $\Gamma^*$ is the surface to be reconstructed.

Dong Wang
wangdong@cuhk.edu.cn

1 School of Science and Engineering, The Chinese University of Hong Kong, Shenzhen 518172, Guangdong, People’s Republic of China
The goal of (1) is to find an optimal surface in the sense of minimizing the $p$-norm of the distance function on the surface. In the continuous limit, when $d(x)$ is the distance function to a smooth surface $\Gamma_0$, it is easy to see that there are two global minimizers (i.e.: $\Gamma = \Gamma_0$ and $\Gamma = \emptyset$). However, in general and practical situations, $C$ is a discrete set and may have noise or missing data. The problem is then interesting and complicated. In this case, the only one global minimizer is the trivial solution $\Gamma = \emptyset$. However, the nontrivial local minimizer is more interesting.

There have been many successful developments along the direction on the surface reconstruction from point cloud: mainly on different objective functionals and numerical methods; for instance, the Poisson surface reconstruction method [16], moving least square projections [29], reconstructing surfaces using anisotropic radius basis functions [5], polygonal surface reconstruction [26], and reconstruction using image segmentation formulations [18]. Besides these, there are many data driven approaches developed for surface reconstruction with priors for considering sampling density, level of noise, missing alignment, local surface smoothness, volumetric smoothness, absence of boundaries, symmetries, shape primitives, or global regularity. We refer to [1] and references therein for a detailed survey on data-driven approaches.

In this paper, we use the indicator function to implicitly represent the surface and approximate the energy in (1) using indicator functions. Based on the new approximation, we derive an unconditionally stable and efficient method to minimize the approximate energy to approximate the optimized solution.

The method is motivated by the threshold dynamics method [21–23] for simulating the motion by mean curvature. Recently, the method is interpreted as a minimizing movement scheme of a Lyapunov functional of indicator functions in [7]. The novel derivation and interpretation in [7] can then be directly generalized to multiphase mean curvature motion with arbitrary surface tensions. The method has attracted much attention due to its simplicity and unconditional stability. It has subsequently been extended to many problems, including the problem of area or volume preserving interface motion [13,31], wetting dynamics [14,32,37,39], image processing [9,20,33,36], target-valued harmonic maps [27,28,34,35], high-order geometric motions [8], and so on.

Threshold dynamics methods can also be generalized and extended to modeling anisotropic interface motions via constructing a specific kernel (instead of heat kernel) (see [6,24,30] for more details). It is obvious that the gradient flow of (1) is a spatially inhomogeneous interface motion where the inhomogeneity comes from varying $|d|^p$ in the domain. One can interpret this as a special “anisotropic” motion where the “anisotropy” comes from the spatial inhomogeneity. In this case, it is difficult to see how approaches from constructing kernels would apply.

In this work, we derive a novel threshold dynamics type method for the application in reconstructing surface from point clouds. We understand this as an iterative approach for minimizing a given surface energy. Also, we perform asymptotic expansions to formally analyze the spatially inhomogeneous dynamics of the interface during each iteration, to build a connection between the method and interface dynamics.

The rest of this paper is organized as follows. In Sect. 2, we introduce new approximations of the energy, derive the numerical method based on the approximation, and prove the unconditional stability property of the method. We discuss some interpretations of dynamics of the interface in Sect. 3. In Sect. 4, we provide an accelerated version of the proposed algorithms. In Sect. 5, we describe the numerical implementation and illustrate the performance of the method using extensive numerical experiments. We draw some conclusions and discussions in Sect. 6.
2 Derivation of the Method

2.1 Approximation of the Energy in (1)

In this paper, since we focus on the codimension 1 interface (e.g., a closed curve in $\mathbb{R}^2$, a surface in $\mathbb{R}^3$, or higher dimensions), we use indicator functions to implicitly represent the interface. That is, we denote

$$u(x) = \begin{cases} 1 & \text{if } x \in \Omega_\Gamma \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $\Omega_\Gamma$ is the domain bounded by an interface $\Gamma$.

Under this representation, as shown in [25], as $\tau \downarrow 0$, the boundary integral $\int_\Gamma |d|^P \, ds$ is approximated by a short time heat flow (i.e.; Gaussian convolution):

$$\int_\Gamma |d|^P \, ds \approx \sqrt{\pi \tau} \int_{\mathbb{R}^n} |d|^P \, u \, G_\tau \ast (1 - u) \, dx, \quad (3)$$

or

$$\int_\Gamma |d|^P \, ds \approx \sqrt{\pi \tau} \int_{\mathbb{R}^n} |d|^P \, (1 - u) \, G_\tau \ast u \, dx \quad (4)$$

where

$$G_\tau(x) = \frac{1}{(4\pi \tau)^{n/2}} \exp \left( -\frac{|x|^2}{4\tau} \right).$$

* denotes the convolution, and $\tau$ is a free parameter. To keep the symmetry of the formula with respect to $u$ and $1 - u$, we approximate $\int_\Gamma |d|^P \, ds$ by $E_\tau(u)$:

$$E_\tau(u) := \frac{1}{2} \sqrt{\frac{\pi}{\tau}} \left( \int_{\mathbb{R}^n} |d|^P \, u \, G_\tau \ast (1 - u) \, dx + \int_{\mathbb{R}^n} |d|^P \, (1 - u) \, G_\tau \ast u \, dx \right) \quad (5)$$

or

$$E_\tau(u) := \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^P \, u \, G_\tau \ast \left( |d|^P (1 - u) \right) \, dx. \quad (6)$$

Note that in the special case when $|d|^P = 1$, the formula reduces to the perimeter or the surface area. It can be used to model multiphase motion with arbitrary surface tensions [7], wetting dynamics [37,39] and image segmentation [33,36]. The convergence of (6) to $\int_\Gamma |d|^P \, ds$ as $\tau \downarrow 0$ is rigorously proved in [12].

Now, one arrives at the following approximate problem: finding $u^{\tau,*}$ such that

$$u^{\tau,*} = \arg \min_{u \in B} E_\tau(u) \quad (7)$$

where

$$B := \{ u \in BV(\Omega, \mathbb{R}) \mid u = \{0, 1\} \}$$

and $BV(\Omega, \mathbb{R})$ denotes the bounded-variation functional space.
2.2 Derivation of the Method Based on (5)

In this section, we iteratively solve (7). Note that problem (7) is a minimization problem of a functional on a nonconvex set $\mathcal{B}$. Using the relaxation approach in [7], we relax this problem to an equivalent problem: finding $u^{\tau,*}$ such that

$$u^{\tau,*} = \arg\min_{u \in \mathcal{K}} E^\tau(u)$$

(8)

where

$$\mathcal{K} := \{ u \in BV(\Omega, \mathbb{R}) | u \in [0, 1] \}.$$ 

The equivalence between (7) and (8) is guaranteed in the following lemma.

**Lemma 1** Problem 7 is equivalent to problem 8. That is,

$$\arg\min_{u \in \mathcal{B}} E^\tau(u) = \arg\min_{u \in \mathcal{K}} E^\tau(u).$$

**Proof** It is easy to see that

$$\min_{u \in \mathcal{K}} E^\tau(u) \leq \min_{u \in \mathcal{B}} E^\tau(u)$$

from the fact that $\mathcal{B} \subseteq \mathcal{K}$.

To finish the proof, we need only to prove

$$\arg\min_{u \in \mathcal{K}} E^\tau(u) \in \mathcal{B}.$$

Assume it is not true and the minimizer is $u^*$, then there exists a set $A \subset \Omega$ with nonzero measure and $c > 0$ such that

$$u^*(x) \in (c, 1 - c), \quad \forall x \in A.$$ 

Denote $u' = u^* + t \chi_A$ where $\chi_A$ is the indicator function of $A$, we have $u' \in \mathcal{K}$ for any $|t| < c$. Direct computation yields

$$\frac{d^2 E^\tau(u')}{dt^2} = -2 \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^p \chi_A G_{\tau} \ast \chi_A \, dx.$$ 

Because $|d|^p \geq 0$ and $|d|^p = 0$ only on a set with zero measure, we have $\frac{d^2 E^\tau(u')}{dt^2} < 0$, especially at $t = 0$ (i.e., $u^*$). This contradicts with the assumption that $u^*$ is a minimizer.

Therefore, we have that the minimizer of $\min_{u \in \mathcal{K}} E^\tau(u)$ must be attained in $\mathcal{B}$. \qed

Now, we use an iterative method to solve (8). Without loss of generality, assume the $k$-th iteration is known, we find $k + 1$-th iteration as follows. At the $k$-th iteration $u^k$, we compute the linearization of $E^\tau(u)$ at $u^k$:

$$L^\tau(u, u^k) = \frac{1}{2} \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} u \phi^k \, dx$$

(9)

where

$$\phi^k = |d|^p G_{\tau} \ast (1 - 2u^k) + G_{\tau} \ast \left(|d|^p (1 - 2u^k)\right).$$
Based on the sequential linear programming [7], we then compute the \( k + 1 \)-th iteration \( u^{k+1} \) by solving the following linearized problem:

\[
\begin{align*}
    u^{k+1} &= \arg \min_{u \in K} L^\tau (u, u^k).
\end{align*}
\]

(10)

Since \( u \) only takes value in \([0, 1]\) (i.e., a bounded set), the minimization problem (10) can be solved in a point-wise manner. That is, at each point \( x \),

\[
    u^{k+1}(x) = \arg \min_{u(x) \in [0,1]} u(x) \phi^k(x).
\]

This can be exactly solved via the following thresholding step:

\[
    u^{k+1}(x) = \begin{cases} 
        1, & \text{if } \phi^k(x) < 0, \\
        0, & \text{otherwise}.
    \end{cases}
\]

The algorithm is summarized into Algorithm 1.

**Algorithm 1:** The iterative method for approximating minimizers of (5).

**Input:** \( \Omega \): computational domain; \( d \): distance function to the point cloud; \( \tau > 0 \); and \( u^0 \in \mathcal{B} \).

**Output:** \( u^* \in \mathcal{B} \).

while not converged do

1. For the fixed \( u^k \), compute

\[
    \phi^k(x) = |d|^p G_\tau * \left( 1 - 2u^k \right) + G_\tau * \left( |d|^p \left( 1 - 2u^k \right) \right).
\]

2. Set

\[
    u^{k+1}(x) = \begin{cases} 
        1, & \text{if } \phi^k(x) \leq 0, \\
        0, & \text{otherwise}.
    \end{cases}
\]

**Remark 1** The criteria for “convergence” in all proposed algorithms is that \( u^{k+1}(\bar{x}) = u^k(\bar{x}) \) on each grid point \( \bar{x} \) in the discretized domain. In other words, the value of \( u \) at no grid point is changing (from 1 to 0 or 0 to 1).

**2.3 Derivation of the Method Based on (6)**

Similar to the derivation in Algorithm 1, we use the same relaxation and linearization approach to derive another unconditional stable method in Algorithm 2. The details are omitted here.

**Remark 2** At each iteration, the computational complexity of Algorithm 2 is about the half of the computational complexity of Algorithm 1 because only one convolution is computed.

As for Algorithm 2, we discuss the stability of the method in the sense of the monotonicity of the approximate energy (6) (i.e.; \( E^\tau (u^{k+1}) \leq E^\tau (u^k) \)).

**Theorem 1** Suppose \( u^k (k = 1, 2, \cdots) \) are computed from Algorithm 2, we have

\[
    E^\tau (u^{k+1}) \leq E^\tau (u^k)
\]

with \( E^\tau (u) \) being defined in (6).
Algorithm 2: The iterative method for approximating minimizers of (6).

**Input:** $\Omega$: computational domain; $d$: distance function to the point cloud; $\tau > 0$; and $u^0 \in B$.

**Output:** $u^* \in B$.

**while** not converged **do**

1. For the fixed $u^k$, compute
   \[ \psi^k(x) = G_\tau \ast \left( |d|^\frac{p}{2} (1 - 2u^k) \right). \]

2. Set
   \[ u^{k+1}(x) = \begin{cases} 1 & \text{if } \phi^k(x) \leq 0, \\ 0 & \text{otherwise.} \end{cases} \]

**Proof** As for $E^\tau(u)$ defined in (6), the linearization of $E^\tau(u)$ at $u^k$ is defined by:

\[ L^\tau(u, u^k) = \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^\frac{p}{2} u^k G_\tau \ast \left( |d|^\frac{p}{2} (1 - 2u^k) \right) dx. \]

Note that we have

\[ E^\tau(u^k) = L^\tau(u^k, u^k) + \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^\frac{p}{2} u^k G_\tau \ast \left( |d|^\frac{p}{2} u^k \right) dx \]

and

\[ E^\tau(u^{k+1}) = L^\tau(u^{k+1}, u^k) + 2 \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^\frac{p}{2} u^{k+1} G_\tau \ast \left( |d|^\frac{p}{2} u^k \right) dx \]

\[ - \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^\frac{p}{2} u^{k+1} G_\tau \ast \left( |d|^\frac{p}{2} u^{k+1} \right) dx. \]

Because $u^{k+1}$ is the solution from the sequential linear programming, we have $L^\tau(u^{k+1}, u^k) \leq L^\tau(u^k, u^k)$. Then, we compute

\[ E^\tau(u^{k+1}) - E^\tau(u^k) = L^\tau(u^{k+1}, u^k) - L^\tau(u^k, u^k) + \mathcal{L} \]

where

\[ \mathcal{L} = - \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} |d|^\frac{p}{2} (u^{k+1} - u^k) G_\tau \ast \left( |d|^\frac{p}{2} (u^{k+1} - u^k) \right) dx \]

\[ = - \sqrt{\frac{\pi}{\tau}} \int_{\mathbb{R}^n} \left[ G_{\tau/2} \ast \left( |d|^\frac{p}{2} (u^{k+1} - u^k) \right) \right]^2 \leq 0. \]

Therefore, we are led to $E^\tau(u^{k+1}) - E^\tau(u^k) \leq 0$. \hfill $\square$

### 3 Interpretations of Interface Dynamics

Note that, at each iteration, both Algorithms 1 and 2 start with an indicator function and end by another indicator function, which implicitly determines a motion of a front. The free parameter $\tau$ can be interpreted as the time step in the dynamics of the interface.

When $|d|^p = 1$, $\forall x \in \Omega$, it is easy to see that both algorithms reduce to the original MBO method [22] where the front evolves along its normal direction by mean curvature. In this case, if there is no other constraint (e.g., volume constraint), the front evolves to the unique minimizer (i.e., the global minimizer $\emptyset$).
However, when \( d(x) \) is varying in the space, we perform the following asymptotic expansions to compute the motion law of the front during the iterations. For the convenience, we write \( \psi = |d|^2 \) and use an asymptotic expansion to expand \( \phi^k(x) \) in Algorithm 2 with respect to a small parameter \( \tau \). Without loss of generality, we set up the interface as that in Fig. 1. Specifically, we assume that \( u^k \) is the indicator function for the region where \( x_2 \geq g(x_1) \) in Fig. 1. Furthermore, we assume the point of interest is the origin and \( g'(0) = 0 \).

In the following, we focus on expanding \( \phi^k \) into a series with respect to \( \tau \) and find the zero level set of \( \phi^k(x) \) which corresponds to the new position of the front (according to Algorithm 2):

\[
\phi^k(x) = G_\tau * \left( \psi(1 - 2u^k) \right)
\]

\[
= \frac{1}{4\pi \tau} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left( -\frac{|x-y|^2}{4\tau} \right) \psi(y)(1 - 2u^k(y)) \, dy \, dy_1
\]

\[
= \frac{1}{4\pi \tau} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left( -\frac{|x-y|^2}{4\tau} \right) \psi(y) \, dy_2 \, dy_1
\]

\[
= \frac{1}{2\pi \tau} \int_{-\infty}^{\infty} \int_{g(y_1)}^{\infty} \exp \left( -\frac{|x-y|^2}{4\tau} \right) \psi(y) \, dy_2 \, dy_1.
\]

Evaluating \( \phi^k(x) \) at \((0, x_2)\) (i.e., the front moves along its normal direction), we have

\[
\phi^k(0, x_2) = I_1 - I_2
\]

where

\[
I_1 = \frac{1}{4\pi \tau} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left( -\frac{y_1^2 + (x_2 - y_2)^2}{4\tau} \right) \psi(y_1, y_2) \, dy_2 \, dy_1
\]

and

\[
I_2 = \frac{1}{2\pi \tau} \int_{-\infty}^{\infty} \int_{g(y_1)}^{\infty} \exp \left( -\frac{y_1^2 + (x_2 - y_2)^2}{4\tau} \right) \psi(y_1, y_2) \, dy_2 \, dy_1.
\]

Assume \( \psi(y_1, y_2) \) is smooth almost everywhere and expand \( \psi(y_1, y_2) \) into its Taylor series around \((0, x_2)\):

\[
\psi(y_1, y_2) = \psi_{0,0} + y_1 \psi_{1,0} + (y_2 - x_2) \psi_{0,1} + \frac{y_1^2}{2} \psi_{2,0} + \frac{(y_2 - x_2)^2}{2} \psi_{0,2} + y_1(y_2 - x_2) \psi_{1,1} + \cdots
\]
where \( \psi_{m,n} \) denotes the mixed partial derivative at \((0, x_2)\) with \(m\)-th order derivative with respect to \(y_1\) and \(n\)-th order derivative with respect to \(y_2\).

As for \(I_1\), direct computation yields

\[
I_1 = \frac{1}{4\pi \tau} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left( -\frac{y_1^2 + y_2^2}{4\tau} \right) \psi(y_1, y_2 + x_2) \, dy_2 \, dy_1
\]

\[
= \sum_{m,n=0}^{\infty} \psi_{m,n} \frac{1}{4\pi \tau} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp \left( -\frac{y_1^2 + y_2^2}{4\tau} \right) y_1^m y_2^n \, dy_2 \, dy_1
\]

\[
= \sum_{m,n=0}^{\infty} \psi_{m,n} \eta_{m,n}
\]

where \(\eta_{m,n} = \xi_m \xi_n\) and

\[
\xi_m = \frac{1}{2\sqrt{\pi \tau}} \int_{-\infty}^{\infty} e^{-\frac{y^2}{4\tau}} y^m \, dy
\]

which can be explicitly computed:

\[
\xi_m = \begin{cases} 
0 & \text{if } m \text{ is odd}, \\
(m - 1)!! \frac{2^{m/2} \tau^{m/2}}{m/2} & \text{if } m \text{ is positive and even,} \\
1 & \text{if } m = 0 
\end{cases}
\]

where \((m - 1)!! = (m - 1)(m - 3) \cdots 3 \cdot 1\) is the double factorial. Therefore,

\[
I_1 = \psi_{0,0} + 2(\psi_{2,0} + \psi_{0,2}) \tau + 4(\psi_{2,2} + 3\psi_{4,0} + 3\psi_{0,4}) \tau^2 + o(\tau^2).
\]

As for \(I_2\), because \(g(0) = g'(0) = 0\), we expand \(g(y_1)\) into its Taylor series around 0 by

\[
g(y_1) = \frac{g^{(2)}}{2!} y_1^2 + \frac{g^{(3)}}{3!} y_1^3 + \frac{g^{(4)}}{4!} y_1^4 + \cdots
\]

where \(g^{(n)}\) denotes the \(n\)-th order derivative at 0. After changing of variables, \(I_2\) is then written into

\[
I_2 = II_1 - II_2
\]

where

\[
II_1 = \frac{1}{2\pi \tau} \int_{-\infty}^{\infty} \int_{0}^{\infty} \exp \left( -\frac{y_1^2 + y_2^2}{4\tau} \right) \psi(y_1, y_2 + x_2) \, dy_2 \, dy_1
\]

and

\[
II_2 = \frac{1}{2\pi \tau} \int_{-\infty}^{\infty} \int_{0}^{\infty} \exp \left( -\frac{y_1^2 + y_2^2}{4\tau} \right) \psi(y_1, y_2 + x_2) \, dy_2 \, dy_1.
\]
For $II_1$, we have

$$II_1 = \sum_{m,n=0}^{\infty} \psi_{m,n} \eta'_{m,n},$$

where $\eta'_{m,n} = \xi_m \xi'_n$ and

$$\xi'_n = \frac{1}{\sqrt{\pi} \tau} \int_0^\infty e^{-\frac{y^2}{\sqrt{\pi}}} y^n \, dy$$

which can be explicitly computed as follows:

$$\xi'_n = \begin{cases} 
\frac{1}{\sqrt{\pi} \tau} \left( n - 1 \right)!! (2\tau)^{(n+1)/2} & \text{if } n \text{ is odd}, \\
\xi_n & \text{if } n \text{ is even}
\end{cases}$$

where $(n-1)!! = (n-1)(n-3) \cdots 2$ is the double factorial when $n$ is odd. Therefore,

$$II_1 = \psi_{0,0} + \frac{2\psi_{0,1}}{\sqrt{\pi}} \tau^{1/2} + 2(\psi_{2,0} + \psi_{0,2}) \tau + \frac{4(2\psi_{0,3} + \psi_{2,1})}{\sqrt{\pi}} \tau^{3/2} + 4(\psi_{2,2} + 3\psi_{4,0} + 3\psi_{0,4}) \tau^2 + o(\tau^2).$$

For the computation of $II_2$, we further expand $e^{-\frac{y^2}{\sqrt{\pi}}}$ into its Taylor series and compute:

$$II_2 = \sum_{m,n=0}^{\infty} \psi_{m,n} \xi_{m,n}$$

where

$$\xi_{m,n} = \frac{1}{2\pi \tau^2} \int_{-\infty}^{\infty} e^{-\frac{y^2}{\sqrt{\pi}}} y^m \, dy \int_0^\infty e^{-\frac{y^2}{\sqrt{\pi}}} y^n \, dy$$

$$= \left( y^2 - \frac{1}{4\tau} y^4 + \frac{1}{32\tau^2} y^6 + \cdots \right) dy dy_1.$$ 

Assume $x_2 \sim O(\tau)$ and write $x_2 = \tau \tilde{x}_2$ with $\tilde{x}_2 \sim O(1)$, then the leading orders in $\xi_{m,n}$ are computed:

$$\begin{align*}
\xi_{0,0} &= \frac{-\tilde{x}_2}{\sqrt{\pi}} \tau^{1/2} + \frac{g^{(2)}}{2\sqrt{\pi}} \tau^{1/2} + \left( \frac{g^{(4)}}{2\sqrt{\pi}} + \frac{\tilde{x}_2^3}{12\sqrt{\pi}} - \frac{g^{(2)} \tilde{x}_2}{4\sqrt{\pi}} \right) \tau^{3/2} + o(\tau^{3/2}) \\
\xi_{1,0} &= \frac{2g^{(3)}}{\sqrt{\pi}} \tau^{3/2} + o(\tau^{3/2}) \\
\xi_{0,1} &= \left( \frac{\tilde{x}_2^2}{2\sqrt{\pi}} - \frac{g^{(2)}}{2\sqrt{\pi}} + \frac{3(g^{(2)})^2}{2\sqrt{\pi}} \right) \tau^{3/2} + o(\tau^{3/2}) \\
\xi_{2,0} &= \left( -\frac{2\tilde{x}_2^3}{\sqrt{\pi}} + 6g^{(2)} \frac{1}{\sqrt{\pi}} \right) \tau^{3/2} + o(\tau^{3/2}) \\
\xi_{0,2} &= o(\tau^{3/2}) \\
\xi_{1,1} &= o(\tau^{3/2})
\end{align*}$$

Therefore,

$$II_2 = \psi_{0,0} \left( \frac{-\tilde{x}_2}{\sqrt{\pi}} + \frac{g^{(2)}}{\sqrt{\pi}} \right) \tau^{1/2} + O(\tau^{3/2}).$$
Collecting all leading terms in $I_1$, $II_1$, and $II_2$ into $I_1 - (II_1 - II_2)$ yields

$$\phi^k(0, \tilde{x}_2) = -\frac{2\psi_{0,1}}{\sqrt{\pi}} \tau^{1/2} + \psi_{0,0}(\frac{-\tilde{x}_2}{\sqrt{\pi}} + \frac{g^{(2)}}{\sqrt{\pi}}) \tau^{1/2} + O(\tau^{3/2}).$$

Set $\phi^k(0, \tilde{x}_2) = 0$, we then have

$$\psi_{0,0} \tilde{x}_2 = \psi_{0,0} g^{(2)} - 2\psi_{0,1}. \quad (19)$$

Note that above analysis is only valid when the origin $(0, 0)$ is not on the zero level set of $\psi$ because $\psi$ is not differentiable at points where $\psi = 0$. If $\psi$ is constant along the normal direction of the front (i.e., $\psi_{0,1} = 0$), the leading order behavior of the front motion is $g^{(2)}$ which denotes mean curvature. This reduces to the formal analysis to the original MBO method [19].

Now, we consider the case where the front is perturbed from the zero level set of $\psi$ and assume $\psi_{0,0} \sim O(\varepsilon)$ and $\psi_{0,1} \sim O(1)$. Then, the value of $\tilde{x}_2$ (the velocity of the front) is dominated by $-2\psi_{0,1}$. Therefore, every point on the front moves to the direction which decreases the value of $\psi$ (see, for example, a diagram in Fig. 2). Since $\psi = |d|^{p/2}$, in the leading order, the front evolves to $\psi = 0$.

**Remark 3** Above analysis only gives the formal leading order velocity of motion of the front. In (19), there is still one term regarding the mean curvature which may play an important role when $\psi_{0,0}$ is not small (i.e., the front is far away from the point cloud). This observation gives several suggestions on the setting of the algorithms: 1.) The initial guess should contain full set of the point cloud, then the interface moves inward based on (19) and 2.) the initial choice $\tau$ should not be very large, otherwise, the front evolves to the global minimizer $\emptyset$ except the local minimizer.

**Remark 4** All above computations can be done for Algorithm 1 and (19) is also expected similarly.

### 4 Accelerations on Algorithms 1 and 2

As we mentioned in Sect. 3, both Algorithms 1 and 2 are related to interface dynamics and $\tau$ is a free parameter which can be interpreted as the time step. Because the method is unconditional stable as shown in Sect. 2.3, the algorithms work for any large $\tau$. However, $\tau$ is also related to the approximation of the energy functional. We do need to use small $\tau$. 
to obtain a good approximation. In the discrete case, when the domain $\Omega$ is discretized by a uniform mesh, the interface may be frozen with a small $\tau$, where the indicator functions between two iterations are exactly same. The whole algorithm is then frozen.

Based on above observations, we propose to accelerate Algorithms 1 and 2 with building a sequence $\tau_1 > \tau_2 > \tau_3 > \cdots > \tau_N$. We start with solving Algorithms 1 and 2 with $\tau_1$ until convergence and then we use the obtained solution to build an initial guess of the Algorithms with $\tau_2$. This is repeated until we have obtained two exactly same solutions with $\tau_\ell$ and $\tau_{\ell+1}$. The algorithm is summarized into Algorithm 3.

**Algorithm 3**: The accelerated version of Algorithm 1 or 2.

**Input**: $\Omega$: computational domain; $d(x)$: distance function to the point cloud; a sequence $\tau_1 > \tau_2 > \tau_3 > \cdots > \tau_N$; $\tau = \tau_1$; $s = 1$; and $u^0 \in B$.

**Output**: $u^* \in B$.

**while** not converged **do**

1. Run Algorithm 1 or 2 to obtain the stationary solution $u^*$ at current $\tau$.
2. $s = s + 1$
3. $\tau = \tau_{s}$

**Remark 5** We remark here that $\tau_1, \tau_2, \cdots$ are not required to be chosen specifically or initially. One could simply choose $\tau_{s+1} = \frac{\tau_s}{2}$ each time after the algorithms with $\tau_s$ converges.

The advantages of Algorithm 3 can be understood as follows:

1. It avoids having the solution become frozen.
2. The initial large $\tau$ makes the solution converge faster.
3. Algorithm 3 converges at the a relatively small $\tau$, which guarantees the accuracy of the solution.

## 5 Numerical Implementation and Experiments

### 5.1 Numerical Implementation

In this section, we discuss the implementation of the whole algorithm. When the point cloud is fixed, $d(x)$ is fixed and only needs to be evaluated once. That is, from the point cloud $C$, we expect to solve

$$
\begin{aligned}
|\nabla d| &= 1, \quad x \in \Omega, \\
\quad d(x) &= 0, \quad x \in C.
\end{aligned}
$$

Following [10], we simply choose a first order Lax–Friedrich scheme to discretize the relaxed dynamical equation of (20):

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

and use a fast sweeping method [15] to solve it with a linear complexity.

Because the point cloud is discrete, if one plots the $\varepsilon$-level set (relatively small $\varepsilon$) of the computed $d(x)$, the contours are small circles around each points in the cloud (see Fig. 3).
Figure 3 tells that a distance function computed from discrete point cloud does not imply the surface reconstruction from point cloud, especially in high dimensional cases where the ordering of points are much more complicated than 2-dimensional cases. This is the reason that we consider minimizing an approximate objective function to obtain the reconstructed surface instead of direct interpolation from the computed distance function.

Besides the computation of \( d(x) \) in above Algorithms, one only needs to compute convolutions in each iteration. They can be efficiently computed using the fast Fourier transform (FFT) based on a uniformly discretized computational domain. In practical implementation, one can also truncate the Gaussian kernel into a narrow band in the discretization domain and calculate the convolution by direct discrete convolution.

### 5.2 Numerical Experiments

In this section, we provide a variety of numerical experiments to show the performance of the proposed algorithm. We implemented all algorithms in MATLAB. All reported results were obtained on a laptop with a 2.7GHz Intel Core i5 processor and 8GB of RAM. In all experiments, the domain \( \Omega = [-\pi, \pi]^n \) is discretized with uniform meshes. If there is no other statement, the CPU time we report is the total CPU time for iterations after \( d(x) \) is computed and \( p = 2 \). For the convenience, we denote Algorithm 3_1 (or 3_2) as Algorithm 3 combined with Algorithm 1 (or 2).

**Visualization of all results:** Indicator functions are piecewise constant functions. Even though indicator functions are used in each iteration, the visualization of the steady solution is based on the zero level set of \( \varphi(x) \) in Step 1.

#### 5.2.1 Comparisons Among Algorithms 1–3

In this section, we perform several experiments to show the comparisons among Algorithms 1–3. We use two point clouds in this experiment as displayed in Fig. 4. The 2-dimensional point cloud is generated using \( N = 200 \) uniform grids \( \theta_i \) (\( i \in [N] \)) in \([0, 2\pi]\):

\[
\begin{align*}
x_i &= r_i \cos(\theta_i), \\
y_i &= r_i \sin(\theta_i)
\end{align*}
\]

where \( r_i = 1 + 0.5 \cos(5(\theta_i - \pi/2)) \). The 3-dimensional point cloud is generated from \( N = 2000 \) random points \((u_i, v_i), i \in [N]\) (i.i.d. from uniform distribution) in \([0, 2\pi]^2\) by:
Fig. 4  Left: Point cloud of five folds. Right: Point cloud of a torus. See Sect. 5.2.1

Fig. 5  First row: Computed results from Algorithm 1 with different \( \tau \). Second row: Computed results from Algorithm 2 with different \( \tau \). Blue curves: Initial guess. Black curves: Reconstructed surface. See Sect. 5.2.1

Fig. 6  Left: The result computed with Algorithm 3_1. Right: The result computed with Algorithm 3_2. The \( \tau \) used in both experiments are 0.02, 0.01, 0.005, 0.0025, and 0.00125; respectively. See Sect. 5.2.1

\[
\begin{align*}
    x_i &= (1 + 0.5 \cos(u_i)) \cos(v_i), \\
    y_i &= (1 + 0.5 \cos(u_i)) \sin(v_i), \\
    z_i &= 0.5 \sin(u_i).
\end{align*}
\]

Figure 5 displays results from Algorithms 1 and 2 with different values of \( \tau \) for the 2-dimensional point cloud. Figure 6 displays the results obtained using Algorithms 3_1 and 3_2. In Fig. 5, for a small \( \tau = 0.0025 \), the solution is frozen at an incorrect solution which is just because of the spatial discretization. In Algorithm 3, this is easily avoided by using the adaptive in time strategy. In all experiments, the computational domain \([-\pi, \pi]^2\) is discretized by 128 \( \times \) 128 uniform grids.

Figure 7 displays the 3-dimensional results obtained from Algorithms 1 and 2 with different choices of \( \tau \). In all experiments, the initial guess is simply set to be the indicator function of a rectangular box:

\[
\{(x, y, z) : |x| < 1.6, |y| < 1.6, \text{ and } |z| < 0.6\}.
\]

The computational domain \([-\pi, \pi]^3\) is discretized with 128\(^3\) uniform grids. It’s easy to see that small \( \tau \) makes both Algorithms be frozen at some local minimizers of the discretized problem. However, smaller \( \tau \) does give a closer surface to the point cloud. In Fig. 8, we
Table 1

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Algorithm} & \tau = 0.02 & \tau = 0.018 & \tau = 0.016 & \tau = 0.01 \\
\hline
\text{Algorithm 1} & \text{Fig. 7 1st row: Computed results from Algorithm 1 with different } \tau. 2nd row: \text{Computed results from Algorithm 2 with different } \tau. \text{ See Sect. } 5.2.1 \text{ Fig. 8 Left: The result computed with Algorithm 3_1. Right: The result computed with Algorithm 3_2. The } \tau_i (i = 1, 2, 3, 4) \text{ used in both experiments are 0.02, 0.01, 0.005, and 0.0025; respectively. See Sect. } 5.2.1 \end{array}
\]

5.2.2 The Energy Decaying Property

In this section, we show the energy decaying property (especially for Algorithm 3) via a 2-dimensional three-fold example. The point cloud is generated using \( N = 100 \) uniform points \( \theta_i \) in \( [0, 2\pi] \):

\[
\begin{align*}
x_i &= r_i \cos(\theta_i), \\
y_i &= r_i \sin(\theta_i)
\end{align*}
\]

where \( r_i = 1 + 0.5 \cos(3(\theta - \pi/2)) \).

Figure 9 displays the energy decaying curves of Algorithms 1 and 2. Even though we only have the theoretical proof of the energy decaying property for Algorithm 2, numerical experiments indicate that Algorithm 1 also has the energy decaying property. In both figures, for different choices of \( \tau \), the curves have similar profiles with only differences in numbers of iterations. This is also consistent with the fact that \( \tau \) plays the role of time step in the algorithm for the evolution of the interface, as we discussed in Sect. 3.

Figure 10 displays energy decaying curves during iterations for Algorithms 3_1 and 3_2. In both figures, relatively big jumps occur at the iteration when \( \tau \) is changed. Several snapshots at the jumps are shown in both figures. From Fig. 10, we observe that the initially choice of \( \tau \) (relatively large) can quickly give a solution in the regime of the local minimizer and smaller \( \tau \) refines the solution.
Fig. 9  Energy decaying curves for Algorithms 1 and 2 (left: Algorithm 1, right: Algorithm 2). See Sect. 5.2.2

Fig. 10  Energy decaying curves for Algorithms 3_1 and 3_2 (top: Algorithm 3_1, bottom: Algorithm 3_2). See Sect. 5.2.2

Fig. 11  Results obtained from Algorithm 3_2 using different values of $p$. Left to right: $p = 1, 2, 3, 4,$ and $5$. See Sect. 5.2.3

5.2.3 Sensitivity to $p$

In this experiment, we check the dependency of the results on the value of $p$. In Fig. 11, we list the results obtained from Algorithm 3_2 for different choices of $p$. The values of $p$ used in Fig. 11 are 1, 2, 3, 4, and 5 from the left to the right, respectively. We observe that for $p \geq 2$, the results show little difference. However, the result obtained from Algorithm 3_2 using $p = 1$ deviates a little from the correct curve. Actually, we did a lot similar experiments for different types of point clouds with different values of $p$; the behavior of the solutions are similar as those in Fig. 11. In other words, $p = 1$ gives worse results from our numerical observation. This is reasonable because $|d|^{p/2}$ is not Lipschitz on the front when $p = 1$. 
5.2.4 Sensitivity to Noisy Data

In this experiment, we perform several experiments to show the capability of proposed algorithms in noisy data. Because Algorithm 3 gives better results than Algorithms 1 and 2, in the following and subsequent experiments, we only show results from Algorithm 3\_2.

We consider the noisy data $\tilde{x}$ generated from the pure data $x$: $\tilde{x} = x + \mu v$, where $v$ is a vector whose entries are independently and identically distributed random variables from a normal distribution and $\mu$ is a parameter to control the intensity of noise. In Fig. 12, we observe that the method still works for noisy data, especially when the noise intensity is not very high. In addition, the two-circle case indicates that the algorithm works well for topological changing cases. In these experiments, the computational domain $[-\pi, \pi]^2$ is discretized by $128 \times 128$ grids.

5.2.5 Sensitivity to Resolutions

In this experiment, we check the sensitivity of the proposed algorithms to the resolution of the computational domain and the number of points in the point cloud. In Fig. 13, we list the
Fig. 13 Results obtained from Algorithm 3 using different discretization of the computational domain and different sizes of point clouds. Top to bottom: Computational domain discretized by $64 \times 64$, $128 \times 128$, $256 \times 256$, and $512 \times 512$ grids. Left to right: Point clouds with 100, 200, 300, and 400 points. See Sect. 5.2.5.

results obtained from Algorithm 3_2, using different sizes of the computational domain and different size of the point cloud. In Fig. 3, the size of point clouds from the left to the right are 100, 200, 300, and 400, respectively. The computational domains from the top to the bottom are discretized by $64 \times 64$, $128 \times 128$, $256 \times 256$, and $512 \times 512$ grids. We observe that as the mesh is refined, the result is closer to the desired curve. In addition, point clouds with larger size can give better results. These observations agree with our expectations.

5.2.6 Efficiency Comparison with Level Set Approaches

In this example, we compare the efficiency between Algorithm 3_2 and level set approaches. A recent work in [10] has carefully studied the efficiency comparison between the semi-implicit method (SIM) and the classic level set approach in [40], showing great improvement in the efficiency. Therefore, in this section, we simply compare the efficiency between Algorithm 3_2 and SIM in [10]. We consider different point clouds (different $m$) generated using $N = 200$ uniform points $\theta_i$ in $[0, 2\pi]$. 
Algorithm 3

Table: CPU times for the computations. See Sect. 5.2.6

| SIM in [10] | 3.2 s | 4.47 s | 5.26 s | 6.40 s | 7.18 s | 4.56 s |
|-------------|-------|--------|--------|--------|--------|--------|
| Algorithm 3.2 | 0.029 s | 0.036 s | 0.029 s | 0.026 s | 0.031 s | 0.030 s |

Fig. 14 Comparisons between Algorithm 3.2 and SIM in [10]. Blue curve: results obtained from Algorithm 3. Black curve: results obtained from SIM in [10]. Green curve: initial guess. Red points: point cloud with 200 points. Table: CPU times for the computations. See Sect. 5.2.6

\[
x_i = r_i \cos(\theta_i), \\
y_i = r_i \sin(\theta_i),
\]

where \( r_i = 1 + 0.4 \sin(m\theta_i) \); see Fig. 14 for the cases when \( m = 3, 4, 5, 6, 7, \) and 8. To make the comparison fair, we use the same initial guess as shown in Fig. 14 and same discretization (128 \times 128 grids) of the computational domain. The table at the bottom of Fig. 14 shows the dramatrical acceleration in the computational CPU time and figures indicate the improvement on the accuracy.

5.2.7 3-Dimensional Examples

In the last example, we show the performance of the proposed algorithms in reconstructing a 2-dimensional surface from the 3-dimensional point cloud. In the following, we choose the \( \sigma \)-level set from \( d(x) \) as the initial guess of the Algorithm with a relatively large \( \sigma \). We note that \( \sigma \) cannot be very small because the \( \sigma \)-level sets are then small balls around each point, similar to those in Fig. 3.

Figure 15 displays the reconstructed results from different point clouds: noisy torus, two tori, bumpy torus, bunny, and pig. The computational domain \([-\pi, \pi]^3\) is discretized by \(128 \times 128 \times 128\) grids. We observe that for the most part in these point clouds, they are well reconstructed especially when the curvature is not that large. However, when the curvature is very big, it is very difficult to have the reconstruction match the point cloud (e.g., the ears of the bunny and pig) because of the diffusion effect. This is consistent with the dynamical motion law we derived in Sect. 3. We expect that this could be improved by considering some more terms (for example, a curvature term as that in [11]) in the objective energy. The CPU time of each experiment is printed on each figure respectively. All of results are completed in seconds on \(128 \times 128 \times 128\) grids, demonstrating the efficiency of the method.
Fig. 15 Results obtained from Algorithm 3.2. Left to right: Reconstructed surface from point cloud, xy-view, yz-view, and xz-view. See Sect. 5.2.7

6 Conclusion and Discussions

In this paper, we developed a novel iterative method to minimize an objective energy functional to reconstruct codimension-1 surfaces from point clouds in both 2- and 3-dimensional Euclidean spaces. The method is simple and unconditional stable in the sense of energy decay. We carefully checked the properties and efficiency using a variety of numerical experiments. The proposed algorithms show great advantages over level set based approaches.

From numerical experiments, we observe that thin parts with large curvature are difficult to reconstruct, especially for 3-dimensional point clouds. We expect this could be improved by considering more terms in the objective functional and this will be investigated and reported in the future. As for the Algorithm 1, all numerical results that were performed so far imply the unconditional stability. However, a theoretical proof is still needed; we believe requires new mathematical tools. Since the surfaces are represented by indicator functions, the accuracy is dependent on the resolution of the discretization mesh. One could use a multi-scale strategy
to refine the mesh after obtaining the results on a coarse mesh and set the results as the initial condition of the refined mesh. In addition, for those applications requiring finer design, the proposed methods can be used as an efficient tool for initialization which can be directly combined with a high order accurate method.

The CPU time for both 2- and 3-dimensional experiments have clearly shown the fast convergence of the algorithm. One could directly extend all proposed algorithms to higher dimensional problems.

Acknowledgements This work is partially supported by the University Development Fund from The Chinese University of Hong Kong, Shenzhen (UDF01001803). D. Wang would like to thank Wei Hu, Hao Liu, and Jun Ma for helpful discussions and Xiao-Ping Wang for constant support, help and encouragement.

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

References

1. Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A., Silva, C.T.: A survey of surface reconstruction from point clouds. Comput. Graph. Forum 36(1), 301–329 (2016). https://doi.org/10.1111/cgf.12802
2. Bi, Z., Wang, L.: Advances in 3d data acquisition and processing for industrial applications. Robot. Comput. Integr. Manuf. 26(5), 403–413 (2010). https://doi.org/10.1016/j.rcim.2010.03.003
3. Bolle, R., Vemuri, B.: On three-dimensional surface reconstruction methods. IEEE Trans. Pattern Anal. Mach. Intell. 13(1), 1–13 (1991). https://doi.org/10.1109/34.67626
4. Calakli, F., Taubin, G.: SSD: smooth signed distance surface reconstruction. Comput. Graph. Forum 30(7), 1993–2002 (2011). https://doi.org/10.1111/j.1467-8659.2011.02058.x
5. Dinh, H.Q., Turk, G., Slabaugh, G.: Reconstructing surfaces using anisotropic basis functions. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, vol. 2, pp. 606–613. IEEE (2001)
6. Elsey, M., Esedoglu, S.: Threshold dynamics for anisotropic surface energies. Math. Comput. 87(312), 1721–1756 (2017). https://doi.org/10.1090/mcom/3268
7. Esedoglu, S., Otto, F.: Threshold dynamics for networks with arbitrary surface tensions. Commun. Pure Appl. Math. 68(5), 808–864 (2014). https://doi.org/10.1002/cpa.21527
8. Esedoglu, S., Tsai, R., Ruuth, S.: Threshold dynamics for high order geometric motions. Interfaces Free Bound. 10, 263–282 (2008)
9. Esedoglu, S., Tsai, Y.H.R., et al.: Threshold dynamics for the piecewise constant Mumford–Shah functional. J. Comput. Phys. 211(1), 367–384 (2006). https://doi.org/10.1016/j.jcp.2005.05.027
10. He, Y., Huska, M., Kang, S.H., Liu, H.: Fast algorithms for surface reconstruction from point cloud. arXiv:1907.01142 (2019)
11. He, Y., Kang, S.H., Liu, H.: Curvature regularized surface reconstruction from point clouds. SIAM J. Imaging Sci. 13(4), 1834–1859 (2020)
12. Hu, W.: Threshold dynamics: analysis and applications. Ph.D. Thesis, Hong Kong University of Science and Technology (2020)
13. Jacobs, M., Merkurjev, E., Esedoglu, S.: Auction dynamics: a volume constrained MBO scheme. J. Comput. Phys. 354(1), 288–310 (2018). https://doi.org/10.1016/j.jcp.2017.10.036
14. Jiang, S., Wang, D., Wang, X.P.: An efficient boundary integral scheme for the MBO threshold dynamics method via the Nufft. J. Sci. Comput. 74(1), 474–490 (2018)
15. Kao, C.Y., Osher, S., Qian, J.: Lax–Friedrichs sweeping scheme for static Hamilton–Jacobi equations. J. Comput. Phys. 196(1), 367–391 (2004). https://doi.org/10.1016/j.jcp.2003.11.007
16. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing, pp. 61–70. Eurographics Association (2006)
17. Khan, D., Shirazi, M.A., Kim, M.Y.: Single shot laser speckle based 3d acquisition system for medical applications. Opt. Lasers Eng. 105, 43–53 (2018). https://doi.org/10.1016/j.optlaseng.2018.01.001
18. Liang, J., Park, F., Zhao, H.: Robust and efficient implicit surface reconstruction for point clouds based on convexified image segmentation. J. Sci. Comput. 54(2–3), 577–602 (2012). https://doi.org/10.1007/s10915-012-9674-8
19. Mascarenhas, P.: Diffusion generated motion by mean curvature. University of California, Los Angeles (1992)
20. Merkurjev, E., Kostic, T., Bertozzi, A.L.: An MBO scheme on graphs for classification and image processing. SIAM J. Imaging Sci. 6(4), 1903–1930 (2013). https://doi.org/10.1137/120886935
21. Merriman, B., Bence, J., Osher, S.: Diffusion generated motion by mean curvature. In: AMS Selected Letters, Crystal Grower’s Workshop, pp. 73–83 (1993)
22. Merriman, B., Bence, J.K., Osher, S.: Diffusion generated motion by mean curvature. University of California, Los Angeles (1992)
23. Merriman, B., Bence, J.K., Osher, S.J.: Motion of multiple junctions: a level set approach. J. Comput. Phys. 112(2), 334–363 (1994). https://doi.org/10.1016/j.jcp.1994.1105
24. Merriman, B., Ruuth, S.J.: Convolution-generated motion and generalized Huygens’ principles for interface motion. SIAM J. Appl. Math. 60(3), 868–890 (2000). https://doi.org/10.1137/S003613999833397X
25. Miranda, M., Pallara, D., Paronetto, F., Preunkert, M.: Short-time heat flow and functions of bounded variation in \( \mathbb{R}^n \). Ann. Facul. Sci. Toulouse Math. 16(1), 125–145 (2007). https://doi.org/10.5802/afst.1142
26. Nan, L., Wonka, P.: PolyFit: polygonal surface reconstruction from point clouds. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2372–2380. IEEE (2017). https://doi.org/10.1109/iccv.2017.258
27. Osting, B., Wang, D.: Diffusion generated methods for denoising target-valued images. Inverse Probl. Imaging 14(2), 205–232 (2020). https://doi.org/10.3934/ipl.2020010
28. Osting, B., Wang, D.: A diffusion generated method for orthogonal matrix-valued fields. Math. Comput. 89, 515–550 (2020). https://doi.org/10.1090/mcom/3473
29. Öztireli, A.C., Guennebaud, G., Gross, M.: Feature preserving point set surfaces based on non-linear kernel regression. Comput. Graph. Forum 28(2), 493–501 (2009). https://doi.org/10.1111/j.1467-8659.2009.01388.x
30. Ruuth, S.J., Merriman, B.: Convolution-thresholding methods for interface motion. J. Comput. Phys. 169(2), 678–707 (2001). https://doi.org/10.1006/jcph.2000.6580
31. Ruuth, S.J., Wetton, B.T.: A simple scheme for volume-preserving motion by mean curvature. J. Sci. Comput. 19(1–3), 373–384 (2003). https://doi.org/10.1023/A:1025368328471
32. Wang, D., Jiang, S., Wang, X.P.: A fast level set method for the Allen–Cahn equation and its geometric applications. J. Comput. Phys. 236, 544–566 (2013). https://doi.org/10.1016/j.jcp.2012.10.038
33. Wang, D., Jiang, S., Wang, X.P.: A fast level set method for the Allen–Cahn equation and its geometric applications. J. Comput. Phys. 236, 544–566 (2013). https://doi.org/10.1016/j.jcp.2012.10.038
34. Wang, D., Osting, B.: A diffusion generated method for computing Dirichlet partitions. J. Comput. Appl. Math. 351, 302–316 (2019). https://doi.org/10.1016/j.cam.2018.11.015
35. Wang, D., Osting, B., Wang, X.P.: A fast level set method for the Allen–Cahn equation and its geometric applications. J. Comput. Phys. 236, 544–566 (2013). https://doi.org/10.1016/j.jcp.2012.10.038
36. Wang, D., Osting, B., Wang, X.P.: A fast level set method for the Allen–Cahn equation and its geometric applications. J. Comput. Phys. 236, 544–566 (2013). https://doi.org/10.1016/j.jcp.2012.10.038
37. Wang, D., Wang, X.P.: The iterative convolution-thresholding method (ICTM) for image segmentation. arXiv:1904.10917 (2019)
38. Wang, D., Wang, X.P.: An efficient iterative thresholding method for image segmentation. J. Comput. Appl. Math. 351, 302–316 (2019). https://doi.org/10.1016/j.cam.2018.11.015
39. Wang, D., Wang, X.P.: An efficient iterative thresholding method for image segmentation. J. Comput. Appl. Math. 351, 302–316 (2019). https://doi.org/10.1016/j.cam.2018.11.015
40. Zhao, H.K., Osher, S., Merriman, B., Kang, M.: Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method. Comput. Vis. Image Underst. 80(3), 295–314 (2000). https://doi.org/10.1006/cviu.2000.0875

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.