Revisiting the Redistancing Problem using the Hopf-Lax Formula

Byungjoon Lee[a], Jérôme Darbon[c], Stanley Osher[b], Myungjoo Kang[a]

[a]*Department of Mathematical Sciences, Seoul National University, Seoul, Korea.*
[b]*Department of Mathematics, University of California at Los Angeles, California 90095-1555 USA*
[c]*CNRS / CMLA Ecole Normale Supérieure de Cachan, France.*

Abstract

This article presents a fast new numerical method for redistancing objective functions based on the Hopf–Lax formula [1]. The algorithm suggested here is a special case of the previous work in [2] and an extension that applies the Hopf–Lax formula for computing the signed distance to the front. We propose the split Bregman approach to solve the minimization problem as a solution of the eikonal equation obtained from Hopf–Lax formula. Our redistancing procedure is expected to be generalized and widely applied to many fields such as computational fluid dynamics, the minimal surface problem, and elsewhere.

Keywords: Reinitialization, Level Set Method, Hopf-Lax formula, Hamilton-Jacobi equations, Split Bregman Method.

1. Introduction

The level set framework [3] has been proven to be a successful technique for describing the motion of a front $\Gamma \in \mathbb{R}^n$ represented by the zero level set of a continuous level set function $\phi : \mathbb{R}^n \to \mathbb{R}$. The main principle of the level set method is that for the function ϕ defined everywhere in the domain Ω, the following evolution equation is solved:

$$\frac{\partial \phi}{\partial t} + V \cdot \nabla \phi = 0, \tag{1}$$

where V is the velocity of the movement of the front. We can obtain the information of the designated time level from the solution of (1). Depending on the meaning of V in (1), its applications can be a part of computational fluid dynamics [4, 5, 6], the minimal surface problem [7], image processing [8, 9], etc.

Although it is well established, the level set function obtained as the solution of (1) suffers from distortion. This unwanted phenomenon can be justified by the observation that the gradient of the level set function is too large or too small in magnitude near the interface Γ. In such cases, the level set function may fail to hold enough regularity near the front; that is, tremendous changes in the zero level set can arise from even small perturbations. Therefore, it is reasonable to replace the level set function with a function that has better properties, i.e., the signed distance function to the front. This process is called *redistancing* or *reinitialization* [4, 10]. There have been many studies on redistancing a given function to a signed distance function. One of the famous attempts is to use the advantages of the signed distance function: the signed distance function is uniquely determined as the viscosity solution of the following eikonal equation [11]

$$\begin{cases} \|\nabla \phi\|_2 = 1 \\ sgn(\phi) = sgn(\phi_0) \end{cases}, \tag{2}$$

where the magnitude of its gradient is always 1. Here, $\phi_0 : \mathbb{R}^n \to \mathbb{R}$ is the given initial level set function, *sgn* denotes the signum function that is equal to 1, -1, and 0 when the input value is positive, negative, and 0, respectively. There are

Email addresses: asone30@snu.ac.kr (Byungjoon Lee), jerome@math.ucla.edu (Jérôme Darbon), sjo@math.ucla.edu (Stanley Osher), mkang@snu.ac.kr (Myungjoo Kang)

well-known algorithms for solving (2), e.g., the fast marching method [12] and fast sweeping method [13]. These two methods differ in the updating order and the number of passes, but they are similar in sense that they solve the finite difference approximation of (2) with a monotone Godunov Hamiltonian [14, 15], guaranteeing the convergence of a discrete equation to the viscosity solution of (2) in [11]. Although these time-marching-type methods are simple and sufficiently fast, these methods produce first-order accurate results only, which can affect the accuracy of important geometric quantities such as the normals to the front and the mean curvature.

For this reason, the following higher order PDE-based redistancing procedure suggested by Sussman, Smereka, and Osher [4] is well-used for the time-dependent problems.

$$\begin{cases} \phi_t + sgn(\phi_0)(\|\nabla\phi\|_2 - 1) = 0 \\ \phi(\cdot, 0) = \phi_0 \end{cases}. \tag{3}$$

The solution of (3) is known to converge to the signed distance function as $t \to \infty$ [16]. The signum term, $sgn(\phi_0)$, plays an important role in fixing the interface for correct positioning. This method has been successfully used for many applications and modified in better ways [10, 17]. In spite of the success of redistancing based on (3), the time-dependent PDE-based method has a trade-off between the computational cost and the accuracy. It converges to a signed distance function for t increasing, with speed almost proportional to distance from the original level set. We will produce a method here which is orders of magnitude faster, more accurate and is embarrassingly parallel.

From the point of view of previous approaches on reinitialization, we note that the redistancing procedures are heavily related to Hamilton–Jacobi equation

$$\begin{cases} \frac{\partial\phi}{\partial t}(x,t) + H(\nabla_x\phi(x,t)) = 0, & \text{in } \mathbb{R}^n \times (0,\infty) \\ \phi(x,0) = J(x) & \forall x \in \mathbb{R}^n \end{cases}. \tag{4}$$

For efficient computation of the solution to Hamilton–Jacobi equation, we focus on a recent work [2] (see also [18] for connections between Hamilton-Jacobi equations and optimization) related to the solution of (4) in an exact way but fast, in which the authors suggest a method for solving the Hamilton–Jacobi equation by making use of the Hopf formula [1] when J is convex

$$\phi(x,t) = -\min_{v\in\mathbb{R}^n}\{J^*(v) + tH(v) - \langle x, v\rangle\}, \tag{5}$$

where $J^* : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is the Fenchel–Legendre transform of a convex, proper, lower semicontinuous function $J : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ defined for any $y \in \mathbb{R}^n$ by [19]

$$J^*(y) := \sup_{x\in\mathbb{R}^n}\{\langle y, x\rangle - J(y)\}. \tag{6}$$

They proposed the split Bregman iteration [20] for solving the optimization problem in (5) to obtain the solution of (4). However, the initial function J may not be convex in general, especially for reinitialization problems. This leads us to choose an alternative, the Hopf–Lax formula, to obtain the solution of (4). The Hopf–Lax formula is a formula that also gives an exact solution of (4) when $H : \mathbb{R}^n \to \mathbb{R}$ is convex but does not require the initial data J to be convex.

Throughout this paper, we propose a totally different and new approach for redistancing the objective function based on the Hopf–Lax formula. The outline of this article is as follows. In Section 2, we introduce an algorithm for redistancing based on the Hopf–Lax formula as a special case of the Hamilton–Jacobi equation solver. A numerical method—in particular, an optimization technique—will be presented and discussed in Section 3. Numerical experiments will be presented to show the effectiveness and accuracy of our approach in Section 4. Finally, the conclusions and the plans for future work will be discussed in Section 5.

2. An algorithm for redistancing and Hopf–Lax formula

In this work, we propose an algorithm for redistancing to a non-empty closed set $\Omega \subset \mathbb{R}^n$ based on the level set method [3] and the Hopf–Lax formula [1]. Let us consider the following eikonal equation

$$\frac{\partial\phi}{\partial t}(x,t) + \|\nabla_x\phi(x,t)\|_2 = 0, \quad (x,t) \in \mathbb{R}^n \times (0,\infty), \tag{7}$$

as a special case of the Hamilton–Jacobi equation, where the Hamiltonian is $\|\cdot\|_2$. Here, $\frac{\partial\phi}{\partial t}(x,t)$ and $\nabla_x\phi(x,t)$ denote the partial derivatives with respect to time t and space x of ϕ at $(x,t) \in \mathbb{R}^n \times (0,\infty)$, respectively. Moreover, we assume that we can find an initial data J such that

$$\phi(x,0) = J(x), \tag{8}$$

where $J : \mathbb{R}^n \to \mathbb{R}$ satisfies

$$\begin{cases} J(x) < 0 & x \in \text{int}\,\Omega \\ J(x) > 0 & x \in (\mathbb{R}^n \setminus \Omega) \\ J(x) = 0 & x \in (\Omega \setminus \text{int}\,\Omega) \end{cases}, \tag{9}$$

where int Ω denotes the interior of Ω. For a given $t > 0$, we define the set

$$\Gamma(t) = \{x \in \mathbb{R}^n \,|\, \phi(x,t) = 0\}, \tag{10}$$

which represents all points that travel a distance t from the interface $\Gamma(0)$ with a speed of 1 in the normal direction. The set $\Gamma(t)$ corresponds to all points that are at a distance t to Ω. We abuse notation and denote by $t(x)$ the distance of x to Ω.

In many fields where redistancing is required, $x \mapsto J(x)$ can be nonconvex and nonformulated data in general. In this work, we only assume that a formula $x \mapsto J(x)$ describing the given data as a zero level set of $J(x)$ can be found such that $J(x)$ can be computed.

For any $x \mapsto J(x)$, the following Hopf–Lax formula [1] gives the solution ϕ of the Hamilton-Jacobi partial differential equation with initial data (4) and is given for any $x \in \mathbb{R}^n$ and $t > 0$ by

$$\phi(x,t) = \min_{y\in\mathbb{R}^n} \left\{ J(y) + tH^*\left(\frac{x-y}{t}\right)\right\}. \tag{11}$$

In our special case in (4), $H = \|\cdot\|_2$, and $H^* : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is known as the indicator function I_Ω of the unit ball that is defined for any $v \in \mathbb{R}^n$ by

$$I_\Omega(v) = \begin{cases} 0 & \text{if } \|v\|_2 \le 1 \\ \infty & \text{otherwise} \end{cases}. \tag{12}$$

Now, we can express the solution ϕ to the system in (7)–(8) for any $x \in \mathbb{R}^n$ and $t > 0$ as

$$\phi(x,t) = \min_{y\in\mathbb{R}^n} \left\{ J(y) + t\, I_\Omega\left(\frac{x-y}{t}\right)\right\}, \tag{13}$$

or more explicitly,

$$\phi(x,t) = \min_{y\in\mathbb{R}^n,\, \|y-x\|_2 \le t} J(y). \tag{14}$$

With the aid of the Hopf–Lax formula, we can summarize our algorithm as follows. Define a level set function $x \mapsto J(x)$ corresponding to the initial data $x \mapsto \phi(x,0)$. Solve the eikonal equation in (7) using the Hopf–Lax formula in (13)–(14). Once we have the solution of (7), find $t(x)$ satisfying (10) for each $x \in \mathbb{R}^n$ using any suitable root finding method. Finally, properly change the sign of $t(x)$ so that $t(x)$ is the signed distance function of the set $\{x \in \mathbb{R}^n \,|\, \phi(x,0) = 0\}$.

The details of the numerical technique for each procedure are presented in the next section.

Remark. *Let $Y(x,t)$ be the minimizer of* (14)*. Then an easy calculation shows that*

$$Y(x,t) = x - t\frac{\nabla\phi(x,t)}{\|\nabla\phi(x,t)\|_2} = x - t\vec{n}(x,t) \tag{15}$$

where $\vec{n}(x,t)$ is the unit normal to the level set function $\phi(x,t) =$ constant at x,t. This means that we can get the normal for free from the Hopf-Lax formula :

$$\vec{n}(x,t) = \frac{x - Y(x,t)}{t} \tag{16}$$

3. Numerical method based on optimization techniques

3.1. Split Bregman method and eikonal equation

We propose the split Bregman solver [20, 21, 22, 23] for the optimization problem. Recall that the solution of (7)–(8) is

$$\phi(x,t) = \min_{y\in\mathbb{R}^n}\left\{J(y) + t\, I_\Omega\left(\frac{x-y}{t}\right)\right\}$$

from the Hopf–Lax formula. The split Bregman technique generates sequences $(v^k)_{k\in\mathbb{N}}$ $(d^k)_{k\in\mathbb{N}}$ and $(b^k)_{k\in\mathbb{N}}$ defined as follows

$$v^{k+1} = \arg\min_{v\in\mathbb{R}^n}\left\{J(v) + \frac{\lambda}{2}\left\|d^k - v - b^k\right\|_2^2\right\}, \tag{17}$$

$$d^{k+1} = \arg\min_{d\in\mathbb{R}^n}\left\{I_\Omega\left(\frac{x-d}{t}\right) + \frac{\lambda}{2}\left\|d - v^{k+1} - b^k\right\|_2^2\right\}, \tag{18}$$

$$b^{k+1} = b^k + v^{k+1} - d^{k+1}, \tag{19}$$

or, in more simple form,

$$v^{k+1} = \arg\min_{v\in\mathbb{R}^n}\left\{J(v) + \frac{\lambda}{2}\left\|d^k - v - b^k\right\|_2^2\right\}, \tag{20}$$

$$d^{k+1} = \arg\min_{d\in\mathbb{R}^n, \|d-x\|_2\le t}\left\|d - v^{k+1} - b^k\right\|_2^2, \tag{21}$$

$$b^{k+1} = b^k + v^{k+1} - d^{k+1}, \tag{22}$$

utilizing the definition of I_Ω.

This splitting technique assures fast convergence for both sequences $(v_k)_{k\in\mathbb{N}}$ and $(d_k)_{k\in\mathbb{N}}$ to the same quantity, a minimizer of (13). Note that (20) can cause problematic behaviors for convergence since the initial function $\phi(x,0)$ may not be convex in general. Even though the split Bregman method for a nonconvex optimization problem remains an area of further research to understand, it is known that the split Bregman method works fairly well for some practical problems such as phase retrieval [24], matrix completion [25, 26], and signal processing [27]. To avoid problems due to the nonconvexity of J in (20), we choose λ to be sufficiently large so that (20) can be a convex optimization problem.

A remarkable observation here is that the second step in (21) can be solved analytically. Indeed, it can be computed as

$$d^{k+1} = x + z, \tag{23}$$

where

$$z = \begin{cases} v^{k+1} + b^k - x & \|v^{k+1} + b^k - x\|_2 \le t \\ t\frac{v^{k+1}+b^k-x}{\|v^{k+1}+b^k-x\|_2} & otherwise \end{cases} \tag{24}$$

from the following lemma, replacing $z = d - x$ in (21).

Lemma 1. *The minimizer* $z_{\min} = \arg\min_{\|z\|_2\le t}\|z-a\|_2$ *is given by*

$$z_{\min} = \begin{cases} t\frac{a}{\|a\|_2} & if\ \|a\|_2 > t \\ a & otherwise \end{cases}. \tag{25}$$

Proof. Let us define $f(z) = \|z-a\|_2^2$ and $g(z) = \|z\|_2^2 - t^2$. Then, the above minimization problem can be converted to minimize $f(z)$ subject to $g(z) \le 0$. Since the feasible region $g(z)$ is bounded and the objective function $f(z)$ is convex, a unique global minimum must exist and satisfy the strong duality condition. Hence, we can find the minimizer of the given problem using the following KKT conditions:

$$\lambda \ge 0, \tag{26}$$

$$\lambda(\|z\|_2^2 - t^2) = 0, \tag{27}$$

$$2(z_i - a_i) + 2\lambda z_i = 0, \quad for\ \ i = 1, \cdots, n, \tag{28}$$

$$g(z) \le 0. \tag{29}$$

1. The case when $\lambda > 0$
   In this case, we have $\|z\|_2 = t$ from (27). From (28), we have $(1+\lambda)z_i = a_i$; that is, $\lambda = \|a\|_2/\|z\|_2 - 1 = \|a\|_2/t - 1$, which tells us $\|a\|_2 > t$ with the assumption $\lambda > 0$. Finally, the substitution of $\lambda = \|a\|_2/t - 1$ into (28) shows that
   $$z_i = t\frac{a_i}{\|a\|_2}, \quad for \;\; i = 1, \cdots, n.$$
2. The case when $\lambda = 0$
   This case simply gives us the following by letting $\lambda = 0$ in (28):
   $$z_i = a_i, \quad for \;\; i = 1, \cdots, n.$$

□

Note that with the aid of Lemma 1, the second step in (21) can be immediately computed, and it can highly affect the efficiency of the entire split Bregman procedure in (20)–(22).

3.2. Remarks on choosing the initial functions and conditions

3.2.1. The initial function $J(x)$

The only assumptions we require on the initial $x \mapsto J(x)$ in the Hopf–Lax formula in (11) is the continuity and 1-coercivity of J, i.e., $\lim_{\|x\|_2 \to +\infty} \frac{J(x)}{\|x\|_2} = +\infty$. Hence, we can choose the initial data suggested in [2] as

$$J(R\theta) = \frac{1}{2m}\left(\left(\frac{R}{W(\theta)}\right)^{2m} - 1\right), \tag{30}$$

where $R \geq 0, \theta \in S^{n-1}$, and m is a positive integer. Here, we use the fact that any given closed bounded domain Ω with non empty interior can be expressed in terms of Wulff shapes [28] by the function $W : S^{n-1} \to \mathbb{R}$ as

$$\Omega = \{(R\theta) \in \mathbb{R}^n \mid R \geq 0, \theta \in S^{n-1}, R \leq W(\theta)\}, \tag{31}$$

where $S^{n-1} = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$ is the $(n-1)$ sphere.

From the algorithm suggested in Section 3.1, the zero level set of ϕ is traveling in its outward normal direction as ϕ evolves along with (7). In this case, we can only have information related to the outside to the front Γ. In order to obtain the inward normal directional information, we need a special treatment for the Hamiltonian $H(\nabla\phi)$ in (7). However, changing the Hamiltonian can cause problems when solving (13) using (20)–(22) since another nonconvex minimization may arise in the step in (21).

Thus, in this article, we suggest a proper modification of the initial level set function $\phi(x, 0) = J(x)$ rather than changing the Hamiltonian in (7). On the region inside the given interface Γ, we define the flipped level set function $x \mapsto \tilde{J}(x)$

$$\begin{cases} \tilde{J}(x) > 0 & x \in \text{int}\,\Omega \\ \tilde{J}(x) < 0 & x \in (\mathbb{R}^n \setminus \Omega) \\ \tilde{J}(x) = 0 & x \in (\Omega \setminus \text{int}\,\Omega) \end{cases} \tag{32}$$

so that the normal direction of the inside region to the front Γ is in the inward direction. With this simple modification, we can solve (7) in the inside region of Γ with the split Bregman technique. Although the introduction of this modified level set function may give rise to a nonconvex optimization problem in the step in (20), the selection of the proper λ can easily overcome this problem. In our algorithm, we set $\tilde{J}(x) = -J(x)$ when $J(x) < 0$.

3.2.2. The initial condition for the split Bregman technique

With the possibility of $x \mapsto J(x)$ being nonconvex in practical problems, we need to be concerned with choosing the initial data for the split Bregman iteration in (20)–(22). In our setting, this occurs when we solve (7) in an interior region of Γ, where $\tilde{J}(x) = -J(x)$. Although our proposed split Bregman procedure in (20)–(22) seems to have no problem with a sufficiently large λ, some potential problems reside in the solution form in (14) obtained by the Hopf–Lax formula. Since (14) is a minimization problem over a ball centered at x with a radius of t, there is a possibility

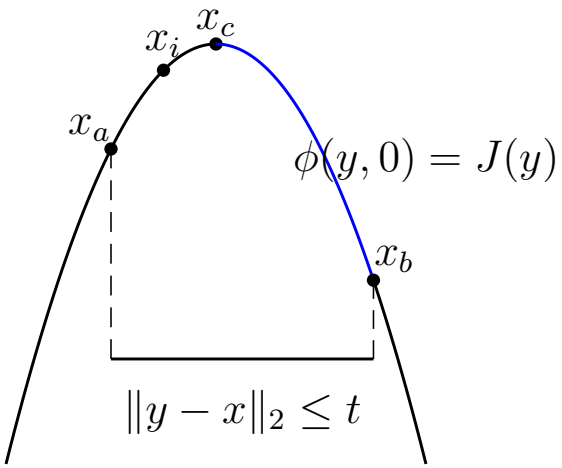

Figure 1: A graph of $\phi(x,t) = \min_{\|y-x\|_2 \leq t} J(y)$ when J is not convex for some x, t.

that the split Bregman procedure in (20)–(22) converges to another local minimizer and not a solution depending on the values x and t.

For a simple and concrete description of this behavior, see Figure 1. In the situation in Figure 1, if we choose a point x_i as an initial v_0 or d_0 for (20)–(22), the computed solution will converge to the point x_a, which is the local minimizer of $J(y)$ over the given domain (designated by the dashed lines) but not the desired solution x_b. This implies that the ideal initial value for (20)–(22) must be chosen from the blue region in Figure 1 that is located to the right of the maximizer x_c of $J(y)$. However, it is not an easy task to find the blue region in general, especially for the 3D problems. To overcome this difficulty, we set the initial guess for the interior region of Γ utilizing an approximated solution of (7) based on a finite difference approximation. For fixed time t_f, solve (7) approximately to have $\phi(x, t_f)$ for all $x \in \mathbf{R}^n$. Then set the initial guess v_0 or d_0 for (20)–(22) as $x - \frac{\nabla_x \phi}{\|\nabla_x \phi\|_2} t$ for each x and t. With this setting, we can enforce our initial guess to be close enough to the global minimum of (14). Note that this technique can be used when $J(x)$ is nonformulated data, i.e., grid based data without any modification.
Since what we need is the approximated solution of (7), we solve (7) on a coarsen grid rather than solving on finest grid for computational efficiency. Then we use an interpolant of this coarse approximated solution as in above consideration. In this article, we use the first-order Euler method in time and the fifth-order WENO apporximation[29] in the space discretization for the finite difference solution of (7).

3.3. Secant method

From the considerations in Section 2, once we have the solution ϕ of (13) using (20)–(22), we need to find $t(x)$ satisfying $\phi(x, t(x)) = 0$ for $x \in \mathbb{R}^n$. Unlike the case in [2], $\nabla_x \phi(x,t)$ cannot be immediately obtained from the split Bregman procedure in our algorithm. Thus, we choose the secant method for finding the 0 of the function $t \mapsto \phi(x,t)$ as an alternative to Newton's method in a previous work [2]. The secant method can expressed as

$$t_{n+1} = t_n - \phi(x, t_n) \left(\frac{t_n - t_{n-1}}{\phi(x, t_n) - \phi(x, t_{n-1})} \right) \tag{33}$$

for two given initial values t_0 and $t_1 > 0$ and integers $n > 0$. In this work, we set $t_0 = 0$ and $t_1 = 0.2$.

In summary, we present the entire redistancing algorithm in the following table.

Redistancing Algorithm Based on the Hopf–Lax Formula
Step 0 : Initialize $J(x)$, whose zero level set is Γ.
Step 0-1 : Define $\tilde{J}(x) = -J(x)$ when $J(x) \leq 0$; otherwise, $\tilde{J}(x) = J(x)$.
Step 0-2 : Set the initial values for the split Bregman iteration in (20)–(22) depending on the shape of $\tilde{J}(x)$.
Step 1 : Set $t_0 = 0$ and t_1 to be a small time step, solve (20)–(22), and obtain the solution $\phi(x, t_1)$ of (7).
Step 2 : While $\lvert t_{n+1} - t_n \rvert < \delta_{sm}$,
Step 2-1 : Update t_n by (33).
Step 2-2 : Solve the split Bregman iteration in (20)–(22) to obtain $\phi(x, t_{n+1})$.
If $\lvert t_{n+1} - t_n \rvert < \delta_{sm}$, let $t(x) = t_n$.
Step 3 : Set $\phi(x, t) = t(x)$ when $J(x) \leq 0$; otherwise, $\phi(x, t) = -t(x)$.

4. Numerical Experiments

In this section, we will present some numerical results based on our novel proposed algorithm for redistancing the given functions. All results are compared with data computed by PDE-based methods for reinitialization [4, 10], widely used in many applications related to redistancing. Since our method proposes to make use of the computation of the "exact" solution of an eikonal equation, the computing times will be presented as results rather than the accuracy. For a comparable accuracy, we use the third-order TVD Runge–Kutta method in time and the fifth-order WENO approximation in the space discretization for the comparison data from [10, 6]. All computation in this section is performed on the uniform grid Ω^h.

We are presenting two and three dimensional cases here to evaluate the efficiency of the point-wise updating nature of the split Bregman technique. Moreover, in an $N_x \times N_y$ grid, the PDE-based method [10] is iterated $2 \cdot \max(N_x, N_y)$ times and $3 \cdot \max(N_x, N_y, N_z)$ for an $N_x \times N_y \times N_z$ grid with the fictitious time step $\tau = \min(\triangle x, \triangle y)/5$, $\tau = \min(\triangle x, \triangle y, \triangle z)/5$ for 3D, to assure its full convergence. As previously mentioned, we perform a finite difference approximation to solve (7) to obtain good initial guesses for the split Bregman iteration, which is important for redistancing inside a region. The computational coarsen grid is Ω^{2h}.

For all experiments, we set $\delta_{sb} = \delta_{sm} = 10^{-8}$ as the tolerances for the split Bregman and secant methods. All numerical tests were coded in C++ and compiled using Visual C++ while MATLAB is used for graphical purposes. Computations are performed on a desktop personal computer with a 3.30-GHz Intel(R) i7-5820K CPU. All computations are performing using double floating point values. We did not use any multicore parallelization in our simulations and focused on the efficiency of the algorithm itself.

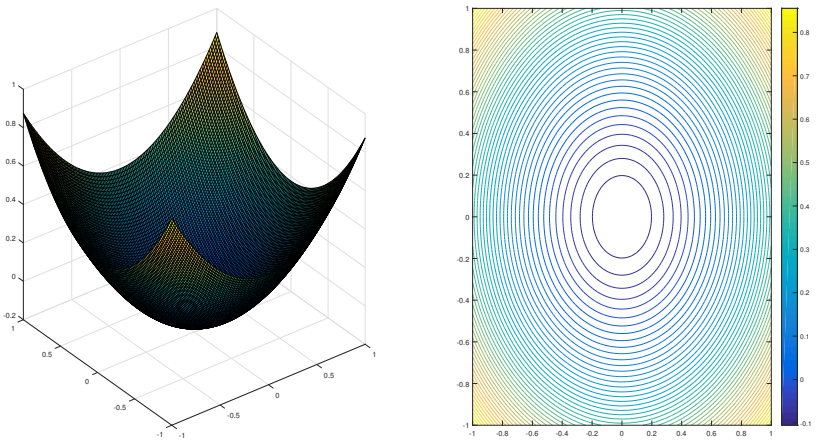

Figure 2: The initial function for Example 1: its graph (left) and contour (right)

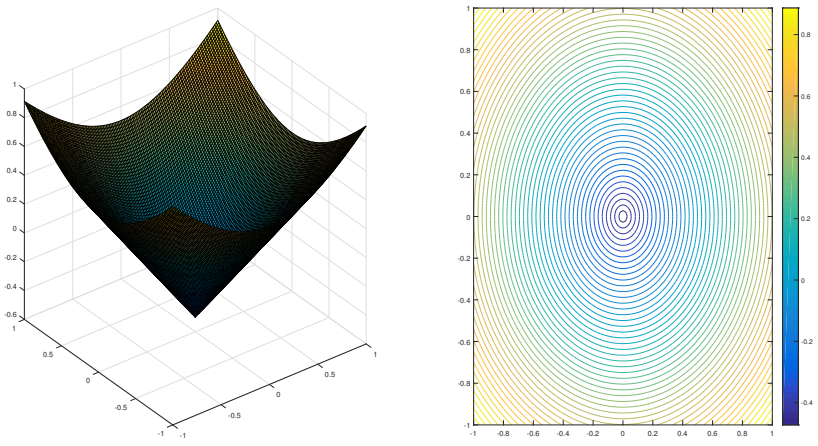

Figure 3: After redistancing the function in Figure 2 by our algorithm: its graph (left) and contour (right)

	Our algorithm	PDE-based method in [10]
Average computational time per point	1.6223e-05 (sec)	2.3943e-04 (sec)
Average computational time (128×128)	0.2699 (sec)	3.9844 (sec)

Table 1: Comparison of the computational times for our algorithm and the PDE-based method [10].

	$\|\kappa - \kappa_h\|_1$	order	$\|\kappa - \kappa_h\|_\infty$	order
16×16	3.48e-03		1.21e-03	
32×32	1.06e-04	5.04	7.06e-05	4.10
64×64	2.92e-06	5.18	3.48e-05	4.34
128×128	1.04e-07	4.81	2.05e-06	4.08

Table 2: Accuracy results in computing the interface's mean curvature

	$\|\overrightarrow{n} - \overrightarrow{n}_h\|_1$	order	$\|\overrightarrow{n} - \overrightarrow{n}_h\|_\infty$	order
16×16	6.29e-04		2.56e-03	
32×32	1.78e-05	5.14	1.06e-04	4.59
64×64	4.67e-07	5.26	5.88e-06	4.16
128×128	1.60e-08	4.86	3.54e-07	4.05

Table 3: Accuracy results in computing the interface's normal vector

	16×16	32×32	64×64	128×128
$\|\overrightarrow{n} - \overrightarrow{n}_h\|_\infty$	4.9985e-07	4.9985e-07	4.9985e-07	4.9985e-07

Table 4: Accuracy results in computing the interface's normal vector

4.1. Example 1 - Circle

In this example, we consider the reinitialization of the initial function $\phi(x, 0) = J(x) = \frac{1}{2}(\sum_{i=1}^{n} x_i^2 - (\frac{1}{2})^2)$ whose zero level set is a circle with a radius of $1/2$. Figure 2 shows the graph and contours of the initial function $J(x)$. From the considerations in Section 3.2.1, we define $\tilde{J}(x) = -J(x)$ when $J(x) < 0$. Note that $\tilde{J}(x)$ is nonconvex in the inside region of the front. We choose $\lambda = 5$ for this example. In addition, we set the initial condition $v_0 = b_0 = (0, 0)$, and $d_0 = (0, 0)$ in the exterior region of the front and $d_0 = x - \frac{\nabla_x \tilde{\phi}}{\|\nabla_x \tilde{\phi}\|_2} t$, in the interior region, for the split Bregman iteration in (20)–(22). Here, $\tilde{\phi}$ is a finite difference approximated solution of (7). The numerical results are shown in Figure 3 for the graph and contours. We can see that the results agree with the desired signed function with a high accuracy as an "exact" solution. For ensuring the accuracy of our approach, we compute the interface's mean curvature and

normal, which are closely related with the smoothness of the objective function. In the recent work in [30], they proposed a high-order reinitalization scheme to get an accurate computation of geometric quantities such as curvature and normal to the front. Their method can compute the curvature with second-order accuracy using a standard central differencing and compute normal with third-order accuracy with a standard fourth-order approximation. In this example, we present an accuracy test on computing the geometric quantities as in in [30] but using higher-order approximations to emphasize that our method computes the exact signed distance function. In order to obtain higher-order accurate results, all geometric quantities of the interface are approximated with a standard fourth-order accurate central difference formula. The errors of computing the normal and curvature are given at Table 2 and 3. This result demonstrates that our method gives fourth-order accuracy for the computation of the interface's curvature and normal. In addition, we present numerical results for computing the normal based on formula (16) in Table 4. This shows that our method can compute the normal very accurately up to the errors of order 10^{-7}.
Table 1 summarizes the computational time per point and a comparison with the results from the PDE-based method in [10]. Even if our method computes the "exact" signed distance function, the results in Table 1 indicate that our approach is fifteen times faster than the PDE-based method [10].

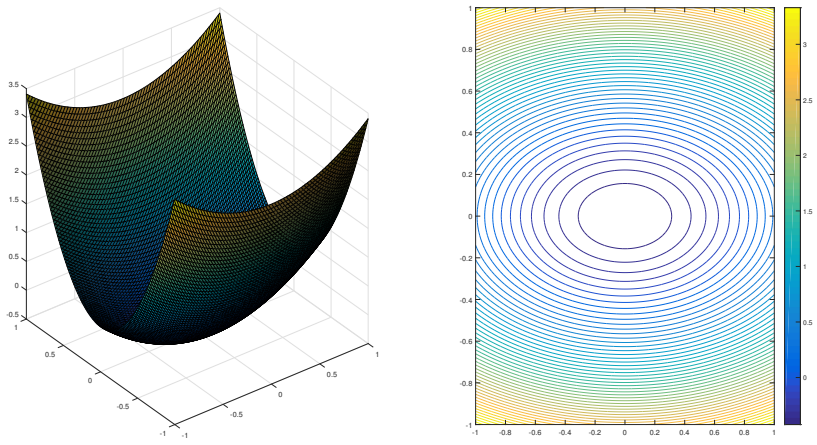

Figure 4: The initial function for Example 2: its graph (left) and contour (right).

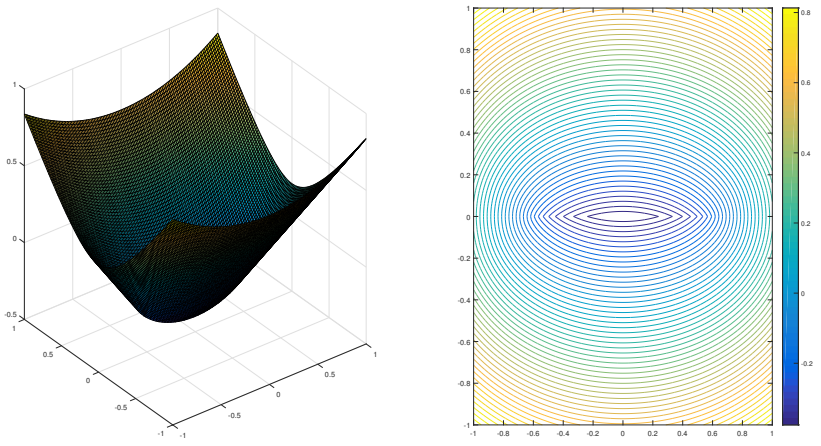

Figure 5: After redistancing the function in Figure 4 by our algorithm. Its graph (left) and contour (right).

	Our algorithm	PDE-based method in [10]
Average computational time per point	2.2535e-05 (sec)	2.3862e-04 (sec)
Average comtpuational time (128×128)	0.3750 (sec)	3.971 (sec)

Table 5: Comparison of the computational times for our algorithm and the PDE-based method [10].

4.2. Example 2 - Ellipse

In this example, we perform redistancing of the initial $\phi(x,0) = J(x) = \frac{1}{2}(\sum_{i=1}^{n} \frac{x_i^2}{a_i^2} - 1)$, whose zero level set is an ellipse. We set $a_1 = 0.8$ and $a_2 = 0.4$ for this example as in Figure 4. Owing to the properties of an ellipse, it is expected that the signed distance function of the ellipse has a segment with multiple gradient values. This singularity makes it more difficult to compute the signed distance function of the ellipse on the basis of an optimization technique since the split Bregman procedure shares the solution's gradient information, which leads to converge to the wrong solution. In contrast, our algorithm is not affected by the singularity in the solution owing to the Hopf–Lax formula in (11). This can be verified by the numerical results in Figure 5. We set the initial values v_0, d_0, and b_0 to be same as those in Example 4.1, and $\lambda = 30$ so that the objective function in (20) is convex. We can see a good result from Table 5, which indicates that our algorithm is about 10 times as fast as its comparison data. Figure 5 shows the graph and contour after redistancing.

4.3. Example 3 - Union of circles

The third example is the reinitialization of a more complex shape that can be represented as the union of circles. We note that there are many points having nonsingularities in the gradients of the initial function $J(y)$ from Figure 6. In this example, we set $\lambda = 5$ and the initial values v_0, d_0, and b_0 are chosen as same as the previous examples. Figure 7 shows the contour and graph after reinitialization, and Table 6 summarizes the computational efficiency, which is almost 12 times faster than the comparison data. We can see that our algorithm works very well with the initial function with nonsingular gradients.

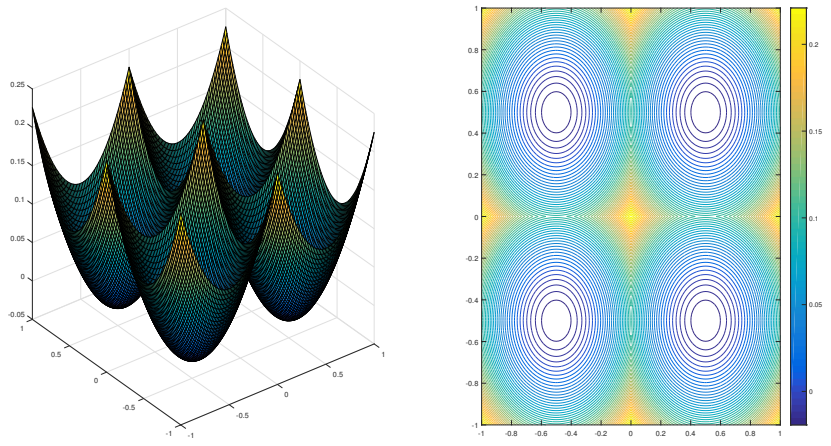

Figure 6: The initial function for Example 3: its graph(left) and contour(right).

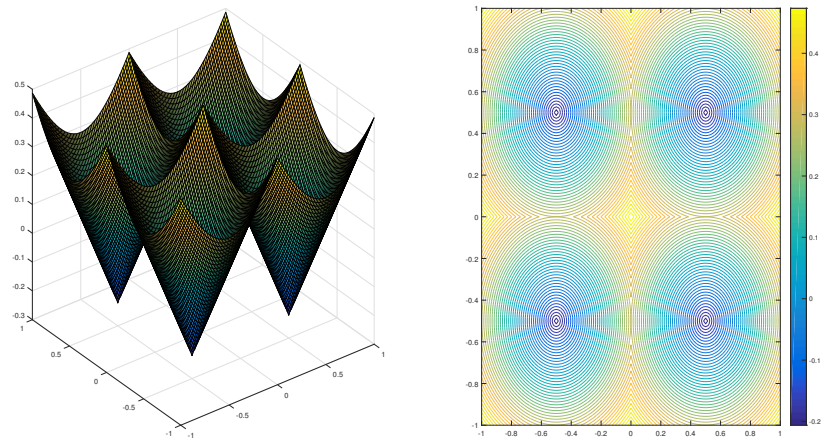

Figure 7: After redistancing the function in Figure 6 by our algorithm: its graph(left) and contour(right).

	Our algorithm	PDE-based method in [10]
Average computational time per point	1.9764e-05 (sec)	2.3790e-04 (sec)
Average comtpuational time (128×128)	0.3289 (sec)	3.959 (sec)

Table 6: Comparison of the computational times of our algorithm and the PDE-based method [10].

4.4. Example 4 - 3D example : sphere

In this example, we are presenting the redistancing of the initial function $\phi(x,0) = J(x) = \frac{1}{2}(\sum_{i=1}^{n} x_i^2 - (0.7)^2)$ whose zero level set is a sphere with radius 0.7. We set $\lambda = 5$ and the initial values v_0, d_0, and b_0 are chosen as similar as the ones of 2D examples. Note that coarsening approximated solver of (7) is relatively more efficient in 3D cases than in 2D cases. The cross-sectional($z = 0$) contours and graphs of $J(x)$ and after redistancing are given by Figure 8, 9, respectively. Table 7 indicates that our algorithm is 40 times as fast as its comparison data. As in

example 1, accuracy results in computing the interface's geometric quantities are given in Table 8 and 9. These results demonstrate that the computed signed distance function from our algorithm is very accurate. Also, we can deduce that extending dimension seldom affect the point-wise update nature of the split bregman while the PDE-based method based on the finite diffence approximation does.

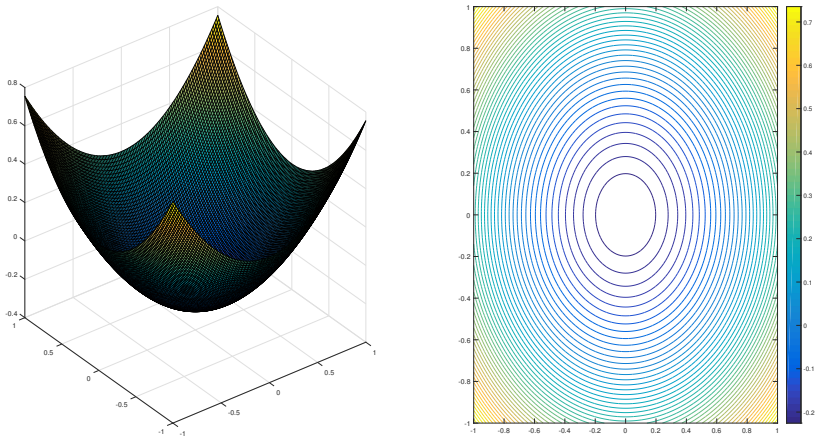

Figure 8: The initial function for Example 4: its graph(left) and contour(right).

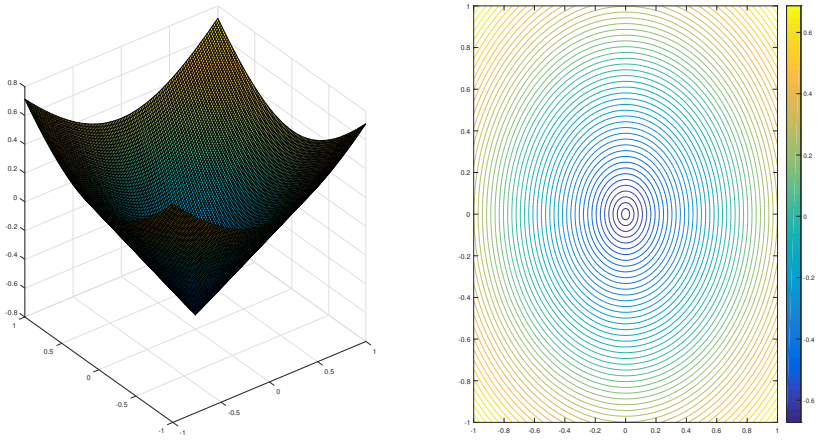

Figure 9: After redistancing the function in Figure 8 by our algorithm (cross section at $z = 0$): its graph(left) and contour(right).

	Our algorithm	PDE-based method in [10]
Average computational time per point	1.342e-05 (sec)	5.6548e-04 (sec)
Average comtpuational time (128×128×128)	28.8093 (sec)	1213.924 (sec)

Table 7: Comparison of the computational times of our algorithm and the PDE-based method [10].

	$\|\kappa - \kappa_h\|_1$	order	$\|\kappa - \kappa_h\|_\infty$	order
16^3	8.33e-04		2.27e-03	
32^3	3.21e-05	4.70	1.61e-04	3.82
64^3	1.99e-06	4.01	9.97e-06	4.02
128^3	1.25e-07	3.99	6.31e-07	3.98

Table 8: Accuracy results in computing the interface's mean curvature

	$\|\overrightarrow{n} - \overrightarrow{n}_h\|_1$	order	$\|\overrightarrow{n} - \overrightarrow{n}_h\|_\infty$	order
16^3	2.64e-04		5.92e-04	
32^3	9.23e-06	4.84	2.85e-05	4.37
64^3	3.03e-07	4.93	1.73e-06	4.04
128^3	1.90e-08	3.99	1.10e-07	3.98

Table 9: Accuracy results in computing the interface's normal vector

4.5. Example 5 - 3D example : ellipsoid

Our final example is an ellipsoid given $\phi(x, 0) = J(x) = \frac{1}{2}\left(\sum_{i=1}^{n} \frac{x_i^2}{a_i^2} - 1\right)$ where $a_1 = 0.9, a_2 = 0.4, a_3 = 0.5$. Figure 10 shows its cross-sectional($z = 0$) contour and graphs. λ is set to be 30 for this example and the initial values v_0, d_0 and b_0 are choosed to be same as Example 4.4. Numerical results after reinitializing are given in Figure 11. We can verify that our algorithm is about 20 times as fast as the PDE-based method from Table 10.

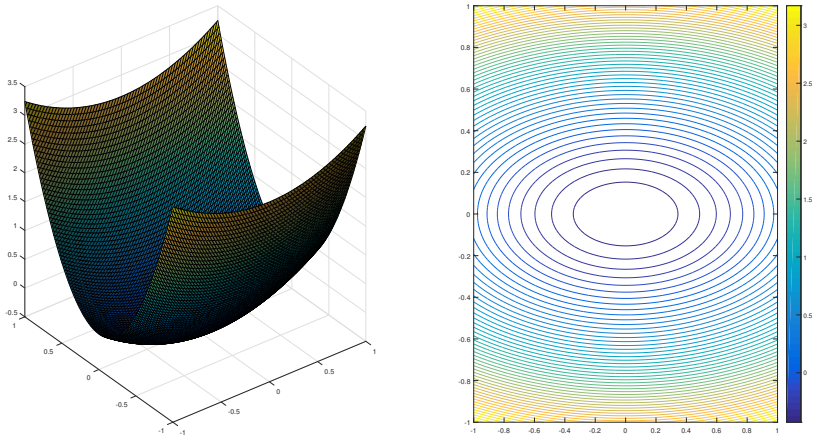

Figure 10: The initial function for Example 5: its graph(left) and contour(right).

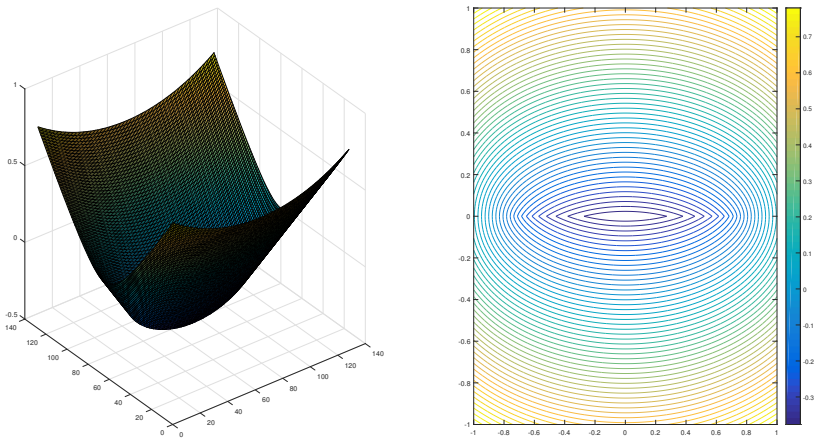

Figure 11: After redistancing the function in Figure 10 by our algorithm (cross section at $z = 0$): its graph(left) and contour(right).

	Our algorithm	PDE-based method in [10]
Average computational time per point	2.7057e-05 (sec)	5.7042e-04 (sec)
Average comtpuational time (128×128×128)	58.0833 (sec)	1224.528 (sec)

Table 10: Comparison of the computational times of our algorithm and the PDE-based method [10].

5. Conclusion and Future Works

In this work, we have developed a fast, accurate algorithm for redistancing the objective function based on the Hopf-Lax formula. The numerical results showed that our algorithm can compute the signed distance function of the given initial data fast and 'exact' way in terms of not using any finite difference approximation. Also, our new algorithm is seldom affected on the curse of dimensionality from its pointwise update nature. Lastly, using the split Bregman and secant method for redistancing enables us to apply parallel computing without any efforts.

However, although we can choose good initial guess for the split bregman method when $J(x)$ is grid based data, it still remains to construct resonable initial $J(x)$ for non-formulated data with complex shape in general situation. In future work, we will consider applying this algorithm to the problem with non-formulated data and modify our algorithm more efficiently, for example, combined with the localization technique in [31] for the practical usage.

Acknowledgement

Jérôme Darbon and Stanley Osher are supported by ONR N000141612157 and DOE DE SC00183838. Myungjoo Kang is supported by Basic Sciences Research Program through the NRF of Korea funded by the Ministry of Science, ICT and Future Planning (2014R1A2A1A10050531 and 2015R1A5A1009350) and MOTIE (10048720).

References

[1] L. C. Evans, Partial differential equations, Graduate Studies in Mathematics 19.

[2] J. Darbon, S. Osher, Algorithms for overcoming the curse of dimensionality for certain hamilton-jacobi equations arising in control theory and elsewhere, CAM report cam15-50.

[3] S. Osher, J. A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations, Journal of computational physics 79 (1) (1988) 12–49.

[4] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational physics 114 (1) (1994) 146–159.

[5] M. Kang, R. P. Fedkiw, X.-D. Liu, A boundary condition capturing method for multiphase incompressible flow, Journal of Scientific Computing 15 (3) (2000) 323–360.
[6] B. Lee, M. Kang, Full 3d simulations of two-phase core–annular flow in horizontal pipe using level set method, Journal of Scientific Computing 66 (3) (2015) 1025–1051.
[7] H.-K. Zhao, S. Osher, B. Merriman, M. Kang, Implicit and nonparametric shape reconstruction from unorganized data using a variational level set method, Computer Vision and Image Understanding 80 (3) (2000) 295–314.
[8] S. Setzer, G. Steidl, T. Teuber, Deblurring poissonian images by split bregman techniques, Journal of Visual Communication and Image Representation 21 (3) (2010) 193–199.
[9] M. Jung, M. Kang, M. Kang, Variational image segmentation models involving non-smooth data-fidelity terms, Journal of Scientific Computing 59 (2) (2014) 277–308.
[10] G. Russo, P. Smereka, A remark on computing distance functions, Journal of Computational Physics 163 (1) (2000) 51–67.
[11] M. G. Crandall, P.-L. Lions, Two approximations of solutions of hamilton-jacobi equations, Mathematics of Computation 43 (167) (1984) 1–19.
[12] J. N. Tsitsiklis, Efficient algorithms for globally optimal trajectories, IEEE Transactions on Automatic Control 40 (9) (1995) 1528–1538.
[13] H. Zhao, A fast sweeping method for eikonal equations, Mathematics of computation 74 (250) (2005) 603–627.
[14] M. Bardi, S. Osher, The nonconvex multidimensional riemann problem for hamilton-jacobi equations, SIAM Journal on Mathematical Analysis 22 (2) (1991) 344–351.
[15] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, Journal of Computational Physics 77 (2) (1988) 439–471.
[16] E. Rouy, A. Tourin, A viscosity solutions approach to shape-from-shading, SIAM Journal on Numerical Analysis 29 (3) (1992) 867–884.
[17] L.-T. Cheng, Y.-H. Tsai, Redistancing by flow of time dependent eikonal equation, Journal of Computational Physics 227 (8) (2008) 4002–4017.
[18] J. Darbon, On convex finite-dimensional variational methods in imaging sciences and hamilton–jacobi equations, SIAM J. Imaging Sciences 8 (4) (2015) 2268–2293.
[19] R. T. Rockafellar, Convex analysis, Princeton university press, 2015.
[20] T. Goldstein, S. Osher, The split bregman method for l1-regularized problems, SIAM Journal on Imaging Sciences 2 (2) (2009) 323–343.
[21] M. Hong, Z.-Q. Luo, M. Razaviyayn, Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems, in: Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on, IEEE, 2015, pp. 3836–3840.
[22] G. Li, T. K. Pong, Global convergence of splitting methods for nonconvex composite optimization, SIAM Journal on Optimization 25 (4) (2015) 2434–2460.
[23] Y. Wang, W. Yin, J. Zeng, Global convergence of admm in nonconvex nonsmooth optimization, arXiv preprint arXiv:1511.06324.
[24] Z. Wen, C. Yang, X. Liu, S. Marchesini, Alternating direction methods for classical and ptychographic phase retrieval, Inverse Problems 28 (11) (2012) 115010.
[25] Y. Shen, Z. Wen, Y. Zhang, Augmented lagrangian alternating direction method for matrix separation based on low-rank factorization, Optimization Methods and Software 29 (2) (2014) 239–263.
[26] Y. Xu, W. Yin, Z. Wen, Y. Zhang, An alternating direction algorithm for matrix completion with nonnegative factors, Frontiers of Mathematics in China 7 (2) (2012) 365–384.
[27] M. Storath, A. Weinmann, L. Demaret, Jump-sparse and sparse recovery using potts functionals, Signal Processing, IEEE Transactions on 62 (14) (2014) 3654–3666.
[28] S. Osher, B. Merriman, The wulff shape as the asymptotic limit of a growing crystalline interface, Asian Journal of Mathematics 1 (1997) 560–571.
[29] G.-S. Jiang, D. Peng, Weighted eno schemes for hamilton–jacobi equations, SIAM Journal on Scientific computing 21 (6) (2000) 2126–2143.
[30] A. du Chéné, C. Min, F. Gibou, Second-order accurate computation of curvatures in a level set framework using novel high-order reinitialization schemes, Journal of Scientific Computing 35 (2-3) (2008) 114–131.
[31] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A pde-based fast local level set method, Journal of computational physics 155 (2) (1999) 410–438.