Fast implementation of piecewise constant level set methods *

Oddvar Christiansen[1] and Xue-Cheng Tai[2]

[1] Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5007 Bergen, Norway. oddvar@mi.uib.no
[2] Department of Mathematics, University of Bergen, Johannes Brunsgate 12, N-5007 Bergen, Norway. tai@mi.uib.no

Summary. Level set methods have been proven to be efficient tools for tracing interface problems. Recently, some variants of the Osher-Sethian level set methods, which is called the Piecewise Constant Level Set Methods (PCLSM), have been proposed for some interface problems. The methods need to minimize a smooth cost functional under some special constraints. A fast algorithm for image segmentation is proposed and tested. The algorithm uses an operator splitting scheme to deal with the gradient descent equation. A special technique is used to tackle the constraint for the PCLSM. By choosing the time step and the penalization parameter properly, the cost functional is minimized and the constraint is fulfilled. Experiments for image segmentation is given. The efficiency of the algorithm and the quality of the obtained images are demonstrated.

Key words: level set method; image segmentation; total variation regularization; operator splitting

1 Introduction

A function $u(\mathbf{x})$ defined on an open and bounded domain $\Omega \in R^d$ may have different properties in distinct regions of Ω. In many applications one wants to separate Ω into a union of these regions, i.e. $\Omega = \cup_{i=1}^{n} \Omega_i$. There are several approaches to accomplishing this segmentation, one is the successful level set method invented by Osher and Sethian [14].

In the standard level set method a distance function $\phi(\mathbf{x})$ is assigned to the function $u(\mathbf{x})$, and the interior and exterior of Ω are represented implicitly by the sign of $\phi(\mathbf{x})$. We have that the interior of Ω is represented by the points $\mathbf{x}$: $\phi(\mathbf{x}) > \mathbf{0}$ and the exterior of Ω is represented by the points $\mathbf{x}$: $\phi(\mathbf{x}) < \mathbf{0}$. The boundary is represented as the zero level set curve $\Gamma = \{\mathbf{x} \in \Omega,\ \phi(\mathbf{x}) = 0\}$.

*We acknowledge support from the Norwegian Research Council and IMS of the National University of Singapore.

The advantage of this representation is that rather than evolving the curve itself, we evolve the distance function $\phi(\mathbf{x})$. This makes topology changes such as merging and breaking an easy task. To divide Ω into more than two domains, one needs to use multiple level set functions [25, 16, 22].

Recently, Lie, Lysaker and Tai [8] presented the piecewise constant level set method (PCLSM) as an alternative approach to the multiple level set function. This method only requires one level set function to represent multiphase segmentation. For some shape identification problems, the PCLSM needs to minimize a smooth functional under some constraints. In [8], gradient method of Uzawa type has been used to solve the saddle point problem coming from the Euler-Lagrange equation for the constrained minimization. Such a method is rather stable, but often has slow convergence. In several recent works, fast algorithms have been proposed to solve this constrained minimization problem. In [19], the MBO projection of [12] has been used to deal with the constraint. The convergence is fast, but the time step needs to be chosen carefully. In [20], a quasi-Newton approach has been tested to solve the saddle point equations. Due to the special structure of the segmentation problem, the cost per Quasi-Newton updating is nearly the same as the gradient updating, but the convergence is much faster if we have good initial guesses. In this work, we try another technique to accelerate the convergence. Due to the special structure of the constraint, we are able to design a special procedure to deal with it. By choosing the penalization parameter and the time step in a proper manner, we are able to use Newton method to enforce the constraint in a rather cost efficient way. Numerical experiments show that this technique has fast convergence and it also has better stability properties. For most of the experiments we have done, we can use the same set of parameters and the algorithm is able to converge in about 40 iterations.

The PCLSM was intended as an alternative for the traditional level set idea of [14, 3]. The ideas are also somehow related to the phase field models for phase transition [17]. They also extend the models proposed in [18, 6]. In [5], the layer between the constant levels are used to distinguish the phases. Here, we use the constant levels. Recently, similar ideas have also been proposed in [9] for some complicated inverse scattering problems.

This work is organized in the following way. In Section 2, we outline the essential ideas of the PCLSM of [8]. In order to improve the efficiency of the algorithms, we will use some operator splitting methods for our computation. A general introduction about the operator splitting methods is given in Section 3. The essential ideas for our fast algorithm is presented in Section 4. Here all the details behind the algorithm are explained. The algorithm and its essential numerical features are exposed in Section 5. We report the numerical experiments in Section 6. The tests show both the quality and the speed of the proposed algorithm.

2 Piecewise Constant Level Set Formulation

First, we give a brief outline of the PCLSM [8]. Assume that we need to find N regions $\{\Omega_i\}_{i=1}^N$ which form a partition of Ω. In order to find the regions we try to find a piecewise constant function which takes values

$$\phi = i \text{ in } \Omega_i, \quad i = 1, 2, \ldots, N. \tag{1}$$

The discontinuities of ϕ give us the curves that separate the regions. Associated with ϕ we define the characteristic functions ψ_i for Ω_i as

$$\psi_i = \frac{1}{\alpha_i} \prod_{\substack{j=1 \\ j \neq i}}^{N} (\phi - j) \text{ and } \alpha_i = \prod_{\substack{k=1 \\ k \neq i}}^{N} (i - k). \tag{2}$$

Each ψ_i is expressed as a product of linear factors of the form $\phi - j$, with the ith factor omitted. Consequently the characteristic functions ψ_i will have the property

$$\psi_i(\mathbf{x}) = \begin{cases} 1 \text{ if } x \in \Omega_i \\ 0 \text{ elsewhere} \end{cases}, \tag{3}$$

as long as (1) holds.

From the characteristic functions we can easily calculate geometric properties like length and area. The length of the boundary of Ω_i is given by the relation

$$|\partial\Omega_i| = \int_{\Omega} |\nabla\psi_i| dx, \tag{4}$$

and the area inside Ω_i is given by the relation

$$|\Omega_i| = \int_{\Omega} \psi_i dx. \tag{5}$$

By linearly combining these characteristic functions we are able to build a cartoon or a piecewise constant image,

$$u = \sum_{i=1}^{n} c_i \psi_i. \tag{6}$$

This is a piecewise constant function and $u = c_i$ in Ω_i if ϕ is as given in (1).

In order to guarantee that the level set function ϕ takes the values as in (1) at convergence, we introduce the constraint function

$$K(\phi) = (\phi - 1)(\phi - 2) \cdots (\phi - N) = \prod_{i=1}^{N} (\phi - i). \tag{7}$$

Requiring

$$K(\phi) = 0 \tag{8}$$

at convergence ensures that ϕ only takes integer values, and that each point $\mathbf{x} \in \Omega$ belongs to one and only phase. This prevents vacuum and overlap between the different phases.

Based on the above observations we propose to solve the following Mumford-Shah functional [13] to find a segmentation of a given image u_0:

$$\min_{\substack{\mathbf{c},\,\phi \\ K(\phi)=0}} \Big\{ F(\mathbf{c},\phi) = \int_\Omega |u-u_0|^2 dx + \beta \sum_{i=1}^{n} \int_\Omega |\nabla \psi_i| dx \Big\}. \tag{9}$$

In the above, β is a nonnegative parameter controlling the regularizing, u is a piecewise constant function depending on ϕ and $\boldsymbol{c}$, as in (6). The first term of (9) is a least square functional, measuring how well the piecewise constant image u approximates u_0. The second term is a regularizer measuring the length of the edges in the image u.

A more simplified cost functional can be achieved by regularizing ϕ directly in (9). The following relation

$$c_1(N) \int_\Omega |\nabla \phi| dx \le \sum_{i=1}^{N} \int_\Omega |\nabla \psi_i| dx \le c_2(N) \int_\Omega |\nabla \phi| dx, \tag{10}$$

where $c_1(N)$ and $c_2(N)$ only depends on N, gives the simplified minimization problem

$$\min_{\substack{\mathbf{c},\,\phi \\ K(\phi)=0}} \Big\{ F(\mathbf{c},\phi) = \int_\Omega |u-u_0|^2 dx + \beta \int_\Omega |\nabla \phi| dx \Big\}. \tag{11}$$

To deal with the constraint $K(\phi) = 0$ we use a penalization method. Defining $W(\phi) = |K(\phi)|^2$ we propose the following penalization functional:

$$\min_{\mathbf{c},\,\phi} \Big\{ F(\mathbf{c},\phi) = \int_\Omega |u-u_0|^2 dx + \beta \int_\Omega |\nabla \phi| dx + \frac{1}{\mu} \int_\Omega W(\phi) dx \Big\}. \tag{12}$$

To solve this minimization problem we propose to use an operator splitting scheme combined with Newton iteration. A similar minimization problem was solved in [8], using augmented Lagrangian Method. It has also been solved in [19] using a MBO approach and in [20] using a quasi-Newton approach.

3 Operator Splitting Scheme

In this section we try to explain the operator splitting scheme in a general setting. For a given function space V and an operator (linear or nonlinear) defined in V, we often need to solve the following time dependent equation:

$$\frac{\partial \phi}{\partial t} + A(\phi) = f(t), \quad t \in [0,T], \quad \phi(0) = \hat{\phi} \in V. \tag{13}$$

If the operator A and the function f can be split in the following way:

$$A = A_1 + A_2 + \cdots A_m, \quad f = f_1 + f_2 + \cdots f_m, \tag{14}$$

then splitting schemes can be used to approximate the solution of (13). Normally, the operators A_i are simpler and easier to solve. The first scheme is called the parallel splitting scheme or additive operator splitting (AOS) scheme. First we choose a time step τ and set $\phi^0 = \hat{\phi}$. At each time level $t_j = j\tau$, we compute $\phi^{j+\frac{i}{2m}}$ in parallel for $i = 1, 2, \cdots, m$ from:

$$\frac{\phi^{j+\frac{i}{2m}} - \phi^j}{m\tau} + A_i(\phi^{j+\frac{i}{2m}}) = f_i(t_j), \text{ and then set } \phi^{j+1} = \frac{1}{m}\sum_{i=1}^{m} \phi^{j+\frac{i}{2m}}. \tag{15}$$

This algorithm was first proposed in Lu, Neittaanmaki and Tai [10, 11]. It was discovered independently later in [23] and used in a different context for image processing [24, 2, 1].

The following sequential scheme, sometimes also called the multiplicative operator splitting (MOS) scheme can also be used to approximate the solution of (13):

$$\frac{\phi^{j+\frac{i}{m}} - \phi^{j+\frac{i-1}{m}}}{\tau} + A_i(\phi^{j+\frac{i}{m}}) = f_i(t_j), \quad i = 1, 2, \cdots, m. \tag{16}$$

We are able to combine the AOS and MOS schemes in different ways, and below we present a combined scheme which will be used for our simulations.

Split the operator A and the function f in the following way:

$$A = \underbrace{A_1 + A_2 + \cdots A_k} + \underbrace{A_{k+1} + \cdots + A_m}, \tag{17}$$

$$f = \underbrace{f_1 + f_2 + \cdots f_k} + \underbrace{f_{k+1} + \cdots + f_m}, \tag{18}$$

i.e. we have grouped A and f into two parts. We now can use the AOS scheme on the first k terms and then the MOS scheme on the remaining $m-k$ terms. This gives the following algorithm:

Algorithm 1 *(A General AOS-MOS scheme)*

- *Use the AOS scheme on the first k terms, i.e. solve* $\phi^{j+\frac{i}{2m}}$ *in parallel from*

$$\frac{\phi^{j+\frac{i}{2m}} - \phi^j}{k\tau} + A_i(\phi^{j+\frac{i}{2m}}) = f_i(t_j), \quad i = 1, 2, ..., k. \tag{19}$$

- *Set*

$$\phi^{j+\frac{k}{m}} = \frac{1}{k}\sum_{i=1}^{k} \phi^{j+\frac{i}{2m}}. \tag{20}$$

- *Use the MOS scheme for the remaining terms, i.e. solve $\phi^{j+\frac{i}{m}}$ sequentially from*

$$\frac{\phi^{j+\frac{i}{m}} - \phi^{j+\frac{i-1}{m}}}{\tau} + A_i(\phi^{j+\frac{i}{m}}) = f_i(t_j), \quad i = k+1, k+2, \cdots, m. \tag{21}$$

4 Operator splitting and Newton methods for image segmentation

In this section we show how the operator splitting idea can be used for the minimization problem (12). In order to solve (12) we need to find $\mathbf{c}$ and ϕ that satisfy

$$a)\ \frac{\partial F}{\partial \mathbf{c}} = 0, \qquad b)\ \frac{\partial F}{\partial \phi} = 0. \tag{22}$$

As u is linear with respect to the c_i values, we see that F is quadratic with respect to c_i. Thus the minimization of (12) with respect to $\mathbf{c}$ can be solved exactly. We have

$$\frac{\partial F}{\partial c_i} = \int_\Omega (u - u_0)\psi_i \, dx, \qquad \text{for } \ i = 1, 2, \cdots N. \tag{23}$$

Therefore, the minimizer of (12) with respect to $\mathbf{c}$ satisfies a linear system of equations $A\mathbf{c} = b$:

$$\sum_{j=1}^{n} \int_\Omega (\psi_i \psi_j) c_j \, dx = \int_\Omega u_0 \psi_i \, dx, \qquad \text{for } \ i = 1, 2, \cdots N. \tag{24}$$

This can be easily solved by forming the matrix A and the vector b and solve the equation $A\mathbf{c} = b$ by an exact solver. The size of the system is very small, i.e. A is a $N \times N$ matrix.

To compute $\frac{\partial F}{\partial \phi}$ we utilize the chain rule to get

$$\frac{\partial F}{\partial \phi} = -\beta \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) + (u(\phi, \boldsymbol{c}) - u_0)\frac{\partial u}{\partial \phi} + \frac{1}{\mu} W'(\phi). \tag{25}$$

The variational formulation also impose the following boundary condition for ϕ on Ω

$$\frac{\nabla \phi}{|\nabla \phi|} \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega. \tag{26}$$

Using a steepest descent method for the minimization of (12) with respect to ϕ we get the following equation for the level set function ϕ:

$$\phi_t = \beta \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right) - (u(\phi, \boldsymbol{c}) - u_0)\frac{\partial u}{\partial \phi} - \frac{1}{\mu} W'(\phi). \tag{27}$$

This is a partial differential equation which we solve using Algorithm 1. We split the right side of (27) into $d+1$ terms:

$$\phi_t = B_1(\phi) + B_2(\phi) + \cdots + B_d(\phi) + C(\phi), \tag{28}$$

where

$$B_i(\phi) = \beta D_i \cdot \left(\frac{D_i\phi}{|\nabla\phi|}\right) - \frac{1}{d}(u(\phi, \boldsymbol{c}) - u_0)\frac{\partial u}{\partial \phi}(\phi, \boldsymbol{c}), \tag{29}$$

and

$$C(\phi) = -\frac{1}{\mu}W'(\phi), \tag{30}$$

$D_i = \frac{\partial}{\partial x_i}$ and d is the spatial dimension. Thus, we have $m = d+1$. Applying Algorithm 1 directly on (28), it gives the following algorithm.

Algorithm 2 *(Mixed AOS-MOS scheme). For $n = 1, 2, ...$ until convergence.*

- *Use the AOS scheme on the first d terms*

$$\frac{\phi^{n+\frac{i}{2m}} - \phi^n}{\tau d} = \beta D_i \cdot \left(\frac{D_i\phi^{n+\frac{i}{2m}}}{|\nabla\phi^{n+\frac{i}{2m}}|}\right) - \frac{1}{d}(u(\phi^{n+\frac{i}{2m}}, \boldsymbol{c}) - u_0)\frac{\partial u}{\partial \phi}(\phi^{n+\frac{i}{2m}}, \boldsymbol{c}) \tag{31}$$

$$i = 1, 2, ..., d.$$

- *Set*

$$\phi^{n+\frac{1}{2}} = \frac{1}{d}\sum_{i=1}^{d}\phi^{n+\frac{i}{2m}} \tag{32}$$

- *Solve ϕ^{n+1} from*

$$\frac{\phi^{n+1} - \phi^{n+\frac{1}{2}}}{\tau} = -\frac{1}{\mu}W'(\phi^{n+1}). \tag{33}$$

In what follows we will show how to efficiently solve (31) and (33). The first of these equations, i.e. equation (31), is nonlinear and implicit. In order to solve it, we use the semi-implicit Picard iteration

$$\frac{\phi_i^{new} - \phi^n}{\tau d} = \beta D_i \cdot \left(\frac{D_i\phi_i^{new}}{|\nabla\phi_i^{old}|}\right) - \frac{1}{d}(u(\phi_i^{old}, \boldsymbol{c}) - u_0)\frac{\partial u}{\partial \phi}(\phi_i^{old}, \boldsymbol{c}). \tag{34}$$

For each i, we choose an initial value as ϕ_i^{old} and get a ϕ_i^{new} which is then taken to be ϕ_i^{old} and get another ϕ_i^{new}. This procedure is iterated until convergence. We then set

$$\phi^{n+\frac{1}{2}} = \frac{1}{d}\sum_{i=1}^{d}\phi_i^{new}. \tag{35}$$

We have chosen to use a semi-implicit scheme to improve the stability and reduce the computational time.

The reason for the dimensional splitting is that this leads to a system of equations which can be efficiently solved using direct solvers for tri-diagonal matrices. Rewrite (34) as

$$\phi_i^{new} - \tau\beta d D_i \cdot \left(\frac{D_i \phi_i^{new}}{|\nabla \phi_i^{old}|} \right) = \phi^n - \tau(u(\phi_i^{old}, \boldsymbol{c}) - u_0) \frac{\partial u}{\partial \phi}(\phi_i^{old}, \boldsymbol{c}) =: r_i, \quad (36)$$

and define the operator

$$A_i = D_i \cdot (a(\mathbf{x}) D_i), \quad (37)$$

where $a(\mathbf{x}) = \frac{1}{|\nabla \phi_i^{old}|}$. Using this we can write (34) as

$$(I - \tau\beta d A_i)\phi_i^{new} = r_i, \quad (38)$$

where I is the identity matrix. For each i, the matrix $(I - \tau\beta d A_i)$ is a tri-diagonal matrix on the mesh lines parallel to the x_i-axes. Thus the systems (38) can be solved fast using a tri-diagonal solver.

The second equation, (33), can be efficiently solved using the Newton iteration. Define:

$$G(\phi) = \phi + \frac{\tau}{\mu} W'(\phi) - \phi^{n+1/2}. \quad (39)$$

We see that (33) is the same as finding a root for G. This problem can be easily solved using the Newton iteration

$$\phi^{new} = \phi^{old} - \frac{G(\phi^{old})}{G'(\phi^{old})}. \quad (40)$$

There is however one problem to take into consideration. W' is a polynomial of degree $2N-1$, where N is the number of phases. Thus there are $2N-1$ roots. If no restriction is placed on τ and μ we can have more than one solution for the system and the Newton iteration can converge to any one of these solutions. Thus to ensure uniqueness and convergence of the Newton iteration we shall choose τ and μ so that $G' > 0$. This will ensure that G is strictly increasing and thus there is only one real root. The rest of the roots are complex. It is easy to see that

$$G'(\phi) = 1 + \frac{\tau}{\mu} |K'(\phi)|^2 + \frac{\tau}{\mu} K(\phi) K''(\phi). \quad (41)$$

Some simple calculation shows that $G' > 0$ will impose the following constrain on τ and μ

Table 1. Upper bounds σ_0 for $\frac{\tau}{\mu}$

N	$\tau/\mu <$
2	2
3	0.71
4	0.09
...	...

The bound depends on the number of phases N. This means that for a given μ we can easily calculate the time step τ to make $G' > 0$. In the next section we present the complete algorithm and show how to choose a proper value for the penalization parameter μ and the initila values for the constants $\mathbf{c}$.

Remark 1. We have used AOS for the B_i operators and MOS for the C operator. The reason to use AOS for B_i is to treat all the spatial variable x_i in a symmetrical way. This could avoid to turn symmetrical images to non-symmetrical images.

5 The Algorithm

The penalization parameter μ controls the effect of the constraint $K(\phi) = 0$. When μ is very small the constraint has large impact on (12), evolving the level set function quickly towards integer values. Whereas, when μ is large the regularizing and fidelity terms are more dominant, smoothing the image under the constraint that u is close to u_0. Our idea is that we start with a large μ ensuring that the regularizing- and fidelity terms are the dominant ones. We then slowly reduce μ towards zero. This will gradually increase the impact of the constraint, ensuring that the level set function ϕ converges towards a piecewise constant function with $\phi = i$ in Ω_i.

Numerical tests we have done show that starting with μ equal to 1000 and then setting $\mu_{new} = 0.75 \cdot \mu_{old}$ for every iteration give good results. This reduces μ from 1000 to $\sim$0.01 in 40 iterations, which is approximately the number of iterations necessary for convergence. Once μ is fixed we determine the value of τ according to Table 1. However, to ensure stability of the Picard iteration (34) we must require an upper bound of τ_{max}. In our numerical tests we have used $\tau_{max} = 0.5$. The number of Picard iterations in our algorithm is set to 1.

There is also a need for an initial approximation of the constants $\mathbf{c}$. When the image only has 2 phases the initial values for $\mathbf{c}$ is not important, the algorithms converge to the same solution even if the $\mathbf{c}$ values are far from the true ones. For more than 2 phases we need a good approximation for the initial

c values, and this is achieved using a simple isodata approach [4]. During the iterations we can update the **c** values using (24). However, this should not be done too early in the process. This is because (24) will give a poor estimate of the **c** values unless the level set function ϕ is close to piecewise constant.

This gives us the following algorithm:

Algorithm 3 *(MBO-Newton method)*

- *Find initial* $\mathbf{c^0}$ *values and set* $\tau = 0.5$, $\mu = 1000$.
  for $n = 1 : n_0$
  - *Solve*

$$\frac{\phi_i^{n+\frac{1}{2}} - \phi^n}{\tau d} = \beta D_i \cdot \left(\frac{D_i \phi_i^{n+\frac{1}{2}}}{|\nabla \phi_i^n|} \right) - \frac{1}{d}(u(\phi_i^n, \mathbf{c}^n) - u_0) \frac{\partial u}{\partial \phi}(\phi_i^n, \mathbf{c}^n). \quad (42)$$

  - *Set*

$$\phi^{n+\frac{1}{2}} = \frac{1}{d} \sum_{i=1}^{d} \phi_i^{n+\frac{1}{2}}. \quad (43)$$

  - *Solve* ϕ^{n+1} *from*

$$\phi^{n+1} - \phi^{n+1/2} = -\frac{\tau}{\mu} W'(\phi^{n+1}). \quad (44)$$

  - *update* $\mathbf{c}^n$ *according to (24)*
  - *Update* μ *and* τ *through* $\mu = 0.75 \cdot \mu$ *and* $\tau = \min(0.5, \mu\sigma_0)$.

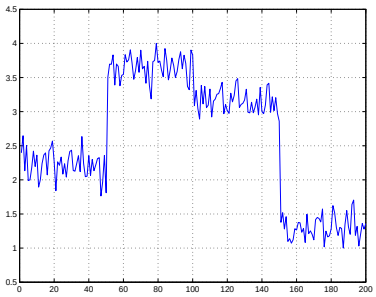
  end

Above σ_0 is the upper bound for τ/μ as given in Table 1 and the iteration number n_0 is chosen to be 40 for all the tests we have done. The cost for solving (44) is very cheap. It normally only takes three or four Newton iterations of (40) to get a rather accurate solution for it.

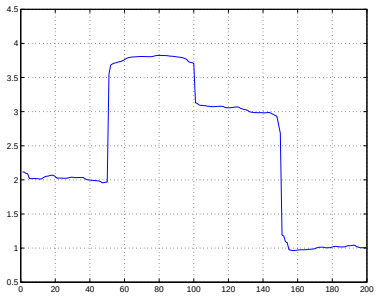
In the rest of this section we will try to explain how the algorithm works. That is, we will try to illustrate how (42) and (43) smooth the level set function ϕ under the restriction that u must be close to u_0. Whereas, (44) evolves ϕ towards integer values.

In order to show how (42) and (43) are evolving the level set function we will run the algorithm without (44). We also keep **c** fixed.

To make the example and visualization as simple as possible we have chosen to use a 1-dimensional signal. Thus instead of a complete image we look at a row of an image. As the initial signal u_0 and the initial level set ϕ^0 we take the noisy step function shown in Fig. 1a). Clearly we can see that this function contains 4 levels or phases. Thus we want the the level set function to converge towards a function which is close to u_0, but smoother due to the use of the total variation term. In Fig. 1b) we have shown the ϕ function after convergence. From the figure we clearly see that ϕ has converged into a smoothed function containing 4 different levels. It is important to notice that these levels are not 1,2,3 and 4 since we have removed (44).



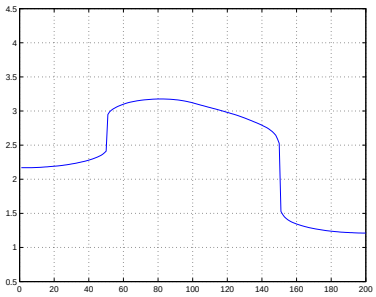
(a) Initial u_0 and ϕ^0.



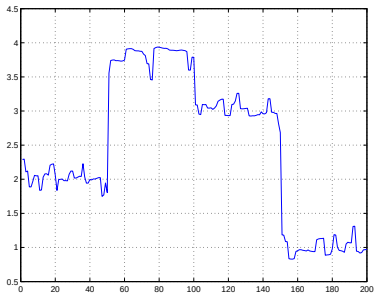
(b) ϕ after convergence

Fig. 1. (42) and (43) evolves the level set ϕ towards a smoothed function containing 4 different levels. β is here set to 0.2.

The parameter β controls the regularization. Thus if we choose β too big we will get a ϕ function that is too smooth, see Fig. 2a). On the other hand if we choose β too small we will get a ϕ function that is too noisy, see Fig. 2b).



(a) $\beta = 5$.

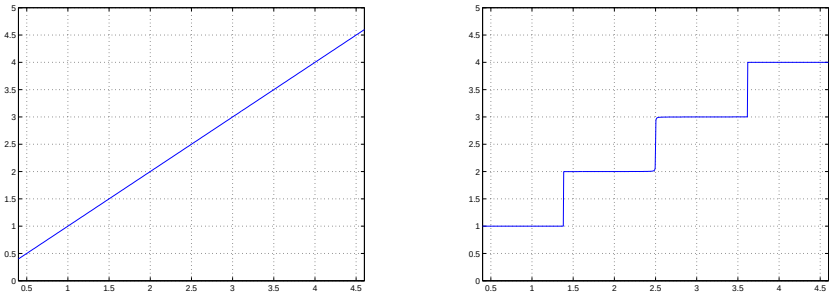


(b) $\beta = 0.05$

Fig. 2. If we increase β too much we will get a ϕ function which is too smooth. On the other hand if we reduce β too much we will get a ϕ function which is too noisy.

In order to show how (44) is evolving the level set function towards integer values we will run the algorithm without (42) and (43). We have chosen a linear function as shown in Fig. 3a) as the initial level set function ϕ^0. Since we want (44) to force the function values towards the nearest integer we want

ϕ to converge into a staircase function, and as Fig. 3b) shows, this is exactly what happens.



(a) Initial ϕ^0 (b) ϕ after convergence

Fig. 3. (44) forces the function values towards the integers and ϕ to converge to a stair function.

We now run the entire algorithm, i.e. we combine (42), (43) and (44). We want the level set function ϕ to converge towards $\phi = i$ in Ω_i, $i = 1, 2, ..., N$, and as Fig. 4 shows, this is exactly what happens.

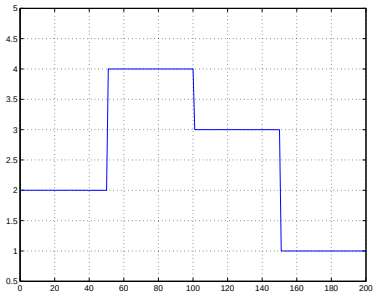


Fig. 4. When the entire algorithm is applied, the level set function ϕ converge towards $\phi = i$ in Ω_i, $i = 1, 2, ..., N$.

Remark 2. If we take $u = \phi, u_0 = \phi_0$ and only iterate between (42) and (43), then this gives a fast implementation for the ROF total variation denoising algorithms of [15].

Remark 3. Compared with the algorithm of [19], we have replaced the MBO-projection [12] by the solving of (44). The MBO-project is enforcing the constrain so "brutally" that the final results depends on the time step size used for (42)-(43). When it is replaced by (44), the cost for the computation is not increased much and the constraint is also enforced properly by reducing the penalization parameter and the time step according to Table 1.

6 Numerical experiments

In this section we validate our algorithm with numerical experiments for real applications. We consider only two-dimensional cases and restrict ourself to gray-scale images, but the schemes can handle any dimension and can be extended to vector-valued images as well. Synthesized images, natural images and an MR image are evaluated.

The algorithm is as described in section 5 and the advantage of this algorithm is that the only parameter which has to be chosen is the regularizer β. This means that for all images presented in this section we have set the initial $\tau = 0.5$ and the reduction factor $\mu = 0.75$. It might be possible that other images requires different values for τ and μ, but we have not experienced this.

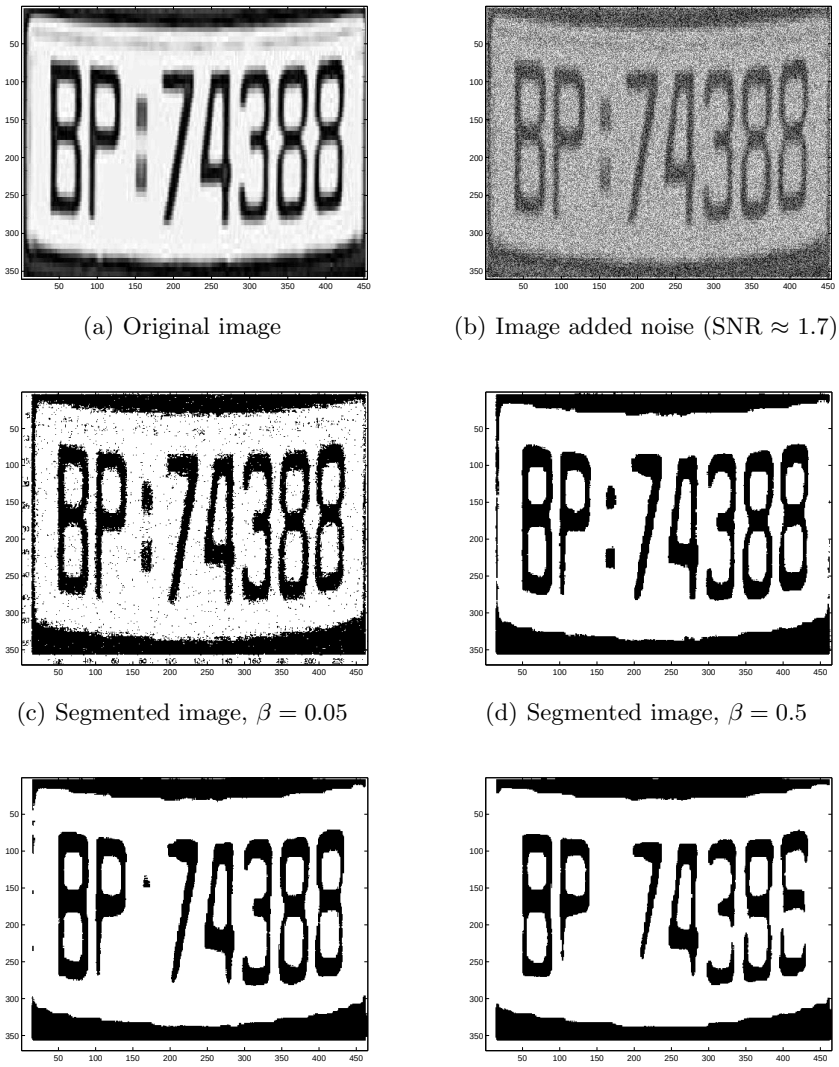
All implementations are done in Matlab, and as the initial ϕ function we use the input image u_0, scaled between one and the number of phases:

$$\phi_0(x) = 1 + \frac{u_0(x) - \min u_0}{\max u_0 - \min u_0}(N-1), \tag{45}$$

where N is the number of phases. All tests are run on on a 2.8GHz Pentium 4 processor.

In the first example we illustrate a 2-phase segmentation on a real car plate. The size of the image is 370×465 pixels, and the CPU time of the segmentation is 26 s. To challenge the segmentation we add Gaussian distributed noise to the real image and use the polluted image in Fig. 5a) as the input data.

To demonstrate the effect of the regularization parameter β we show a number of segmentations with different β values. We see that for $\beta = 0.5$ we have a very good segmentation of the noisy car plate. For smaller β values we are not able to remove the noise, and for larger β values we regularize too much and remove details from the image.



(a) Original image

(b) Image added noise (SNR ≈ 1.7)

(c) Segmented image, $\beta = 0.05$

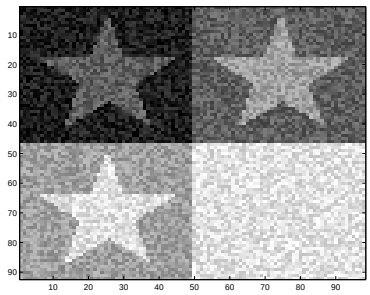
(d) Segmented image, $\beta = 0.5$

(e) Segmented image, $\beta = 1$

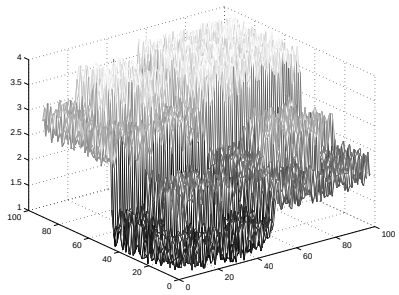
(f) Segmented image, $\beta = 2$

Fig. 5. The regularization parameter β controls the length of the boundary. For $\beta = 0.5$ we have a good segmentation of the car plate. For smaller β we are not able to remove the noise, and for larger β we regularize too much and remove details from the image.

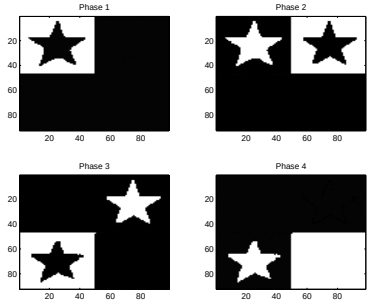
In the next example we show a segmentation of the noisy star image in Fig. 6a). The size of the image is 92×98 pixels, and the CPU time for the segmentation is 2.1 s. The star image consists of four different phases, and as Fig. 6c) shows the algorithm separates these phases very well. We have also shown the initial and final ϕ function. As Fig. 6b) shows, the initial ϕ function is the image scaled between one and four, i.e. the number of phases. After the algorithm has converged the ϕ function only contains four levels, see Fig. 6d).



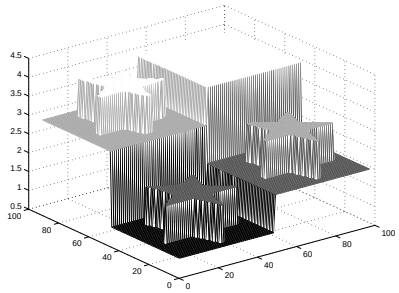
(a) Star image added noise (SNR ≈ 15.7)



(b) Initial level set function ϕ



(c) Segmented image, $\beta = 0.25$



(d) Converged level set function ϕ

Fig. 6. 4 phase segmentation of a noisy star image.

In our next example segmentation of a MR image is demonstrated. The size of image is 296×400 pixels, and the CPU time for the segmentation is 35 s. The image in Fig. 7 is available at `http://www.bic.mni.mcgill.ca/brainweb/`. These realistic MRI data are used by the neuro imaging community to evaluate the performance of various image analysis methods in a setting where the truth is known. For the image used in this test the noise level is 7% and the non-uniformity intensity level of the RF-puls is 20%.

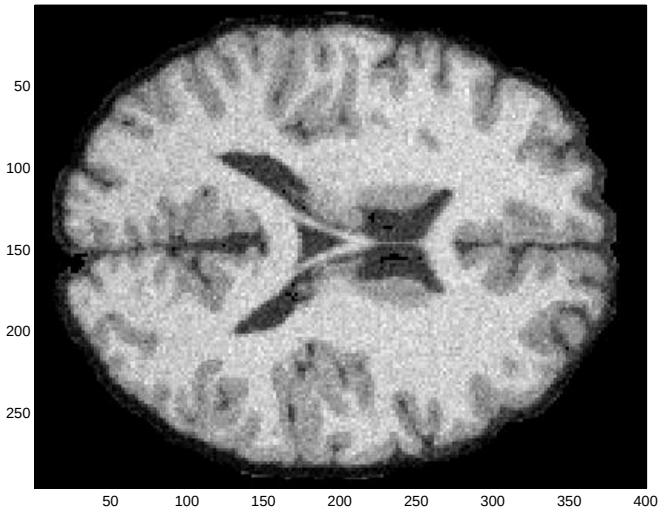
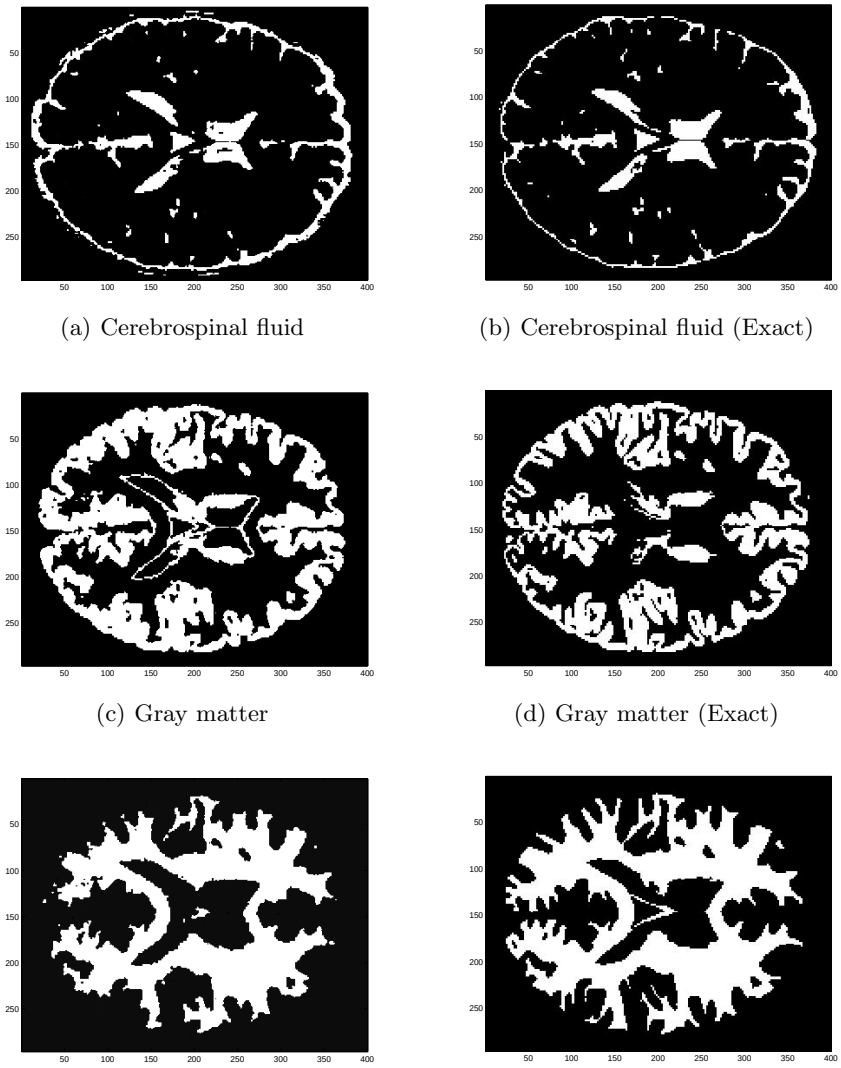


Fig. 7. MRI image with a change in the intensity values going from left to right caused by the non-uniform RF-puls.

There are three tissue classes that should be identified; phase 1: cerebrospinal fluid, phase 2: gray matter, phase 3: white matter, and in Fig. 8 we have compared the results from our algorithm with the exact phases. We have not depicted the background phase here. We see that we have lost some details, due to the presence of noise.



(a) Cerebrospinal fluid (b) Cerebrospinal fluid (Exact)

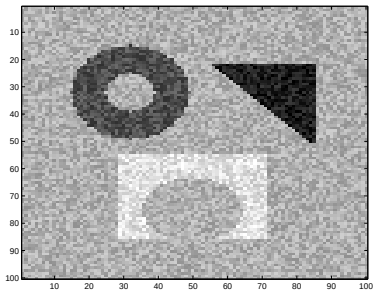
(c) Gray matter (d) Gray matter (Exact)

(e) White matter (f) White matter (Exact)

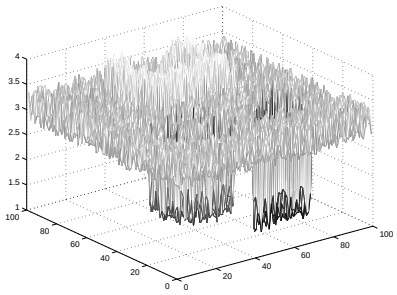
Fig. 8. Comparison of the exact segmentation of a MRI phantom and the results using our algorithm ($\beta = 0.14$).

In Fig. 9 we show the results from a 4-phase segmentation of a noisy synthetic image containing 3 objects. The size of the image is 100×100 pixels, and the CPU time for the segmentation is 2.5 s. This is the same image as Chan

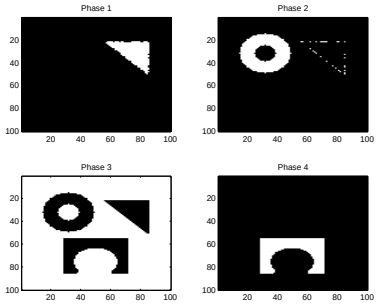
and Vese used to examine their multiphase algorithm [3, 22]. We see that the algorithm captures the circle and the curved object perfectly, however there is a problem with the triangle. We have a misclassification of the boundary of the triangle. This is probably due to the fact that we regularize directly on the ϕ function in (12). The jump on the boundary of the triangle is twice of the jump on the boundary of the other two objects, see Fig. 9 d), and the regularization probably "punishes" this jump too hard. Thus in future work we might have to consider to regularize directly on the characteristic functions, in the same manner as in [8].



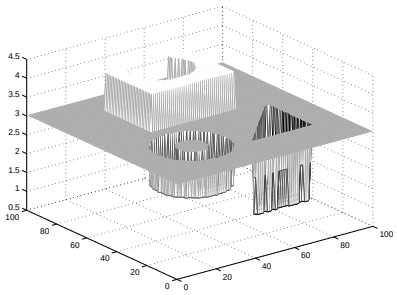
(a) Noisy image containing 3 objects (SNR ≈ 6.1)



(b) Initial level set function ϕ



(c) Different phases, $\beta = 0.12$

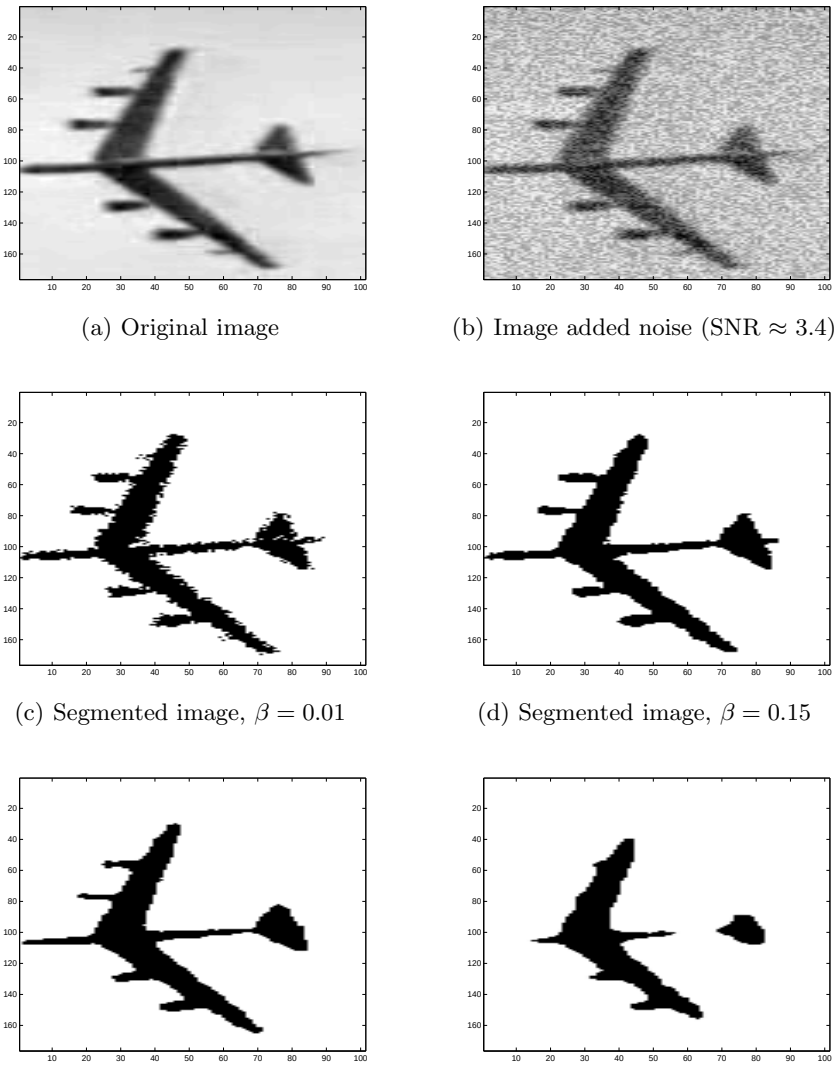


(d) Converged level set function ϕ

Fig. 9. 4-phase segmentation of a noisy synthetic image containing 3 objects.

In our final example we present a two phase segmentation of a real picture of a plane, Fig. 10a). The size of the image is 176×101 pixels, and the CPU time for the segmentation is 2.6 s. As before we have added noise to challenge the segmentation, see Fig. 10b). We show a number of segmentations with different β values. We see that for $\beta = 0.15$ we have a very good segmentation

of the plane. For smaller β values the edges are too noisy, and for larger β values we regularize too much and remove details from the image.



(a) Original image

(b) Image added noise (SNR ≈ 3.4)

(c) Segmented image, $\beta = 0.01$

(d) Segmented image, $\beta = 0.15$

(e) Segmented image, $\beta = 0.5$

(f) Segmented image, $\beta = 1$

Fig. 10. Different segmentations of a noisy plane image. For $\beta = 0.5$ we have a good segmentation of the plane. For smaller β values the edges are too noisy, and for larger β values we regularize too much and remove details from the image.

7 Conclusion

Due to the special structure of the PCLSM, we propose a special method to deal with the constraint. In order to have this method to work, we need to choose the time step τ and the penalization parameter μ to satisfy some inequality. By doing this, we have a very cost efficient way to enforce the constraint. Application to image segmentation is tested in this work. The convergence is fast. Compared with the other fast methods of [19, 20], we do not need good initial values and the algorithm is nearly parameter "free". It is easy to find values for τ and μ that are good for most of the tested images.

Applications of this idea for PCLSM to inverse problems and multiphase motion problems have also been tested in [21, 7]. Those results show the applicability of the PCLSM for a class of shape identification problems.

References

1. Danny Barash. Nonlinear diffusion filtering on extended neighborhood. *Appl. Numer. Math.*, 52(1):1–11, 2005.
2. Danny Barash, Tamar Schlick, Moshe Israeli, and Ron Kimmel. Multiplicative operator splittings in nonlinear diffusion: from spatial splitting to multiple timesteps. *J. Math. Imaging Vision*, 19(1):33–48, 2003.
3. Tony F. Chan and Luminita A. Vese. Image segmentation using level sets and the piecewise constant mumford-shah model. Technical report, CAM Report 00-14, UCLA, Math. Depart., April 2000. revised December 2000.
4. Velasco Dias and Flavio R. Thresholding using the ISODATA clustering algorithm. *IEEE Trans. Systems Man Cybernet.*, 10(11):771–774, 1980.
5. Chung G. and L. A. Vese. Energy minimization based segmentation and denoising using a multilayer level set approach. In *Energy minimization methods in computer vision and pattern recognition, proceedings lecture notes in computer science, vol. 3757*, pages 439–455, Springer-Verlag, Berlin, 2005.
6. Frédéric Gibou and Ronald Fedkiw. A fast hybrid k-means level set algorithm for segmentation. Technical report, Stanford Technical Report, 2002.
7. Hongwei Li and Xue-Cheng Tai. Piecewise constant level set methods (PCLSM) for interface problems. Technical report, UCLA, Applied Mathematics, 2006.
8. Johan Lie, Marius Lysaker, and Xue-Cheng Tai. A variant of the levelset method and applications to image segmentation. *UCLA CAM03-50*, 2003 (to appear in Math. Comp.).
9. A. Litman. Reconstruction by level sets of n-ary scattering obstacles. *Inverse Problems*, 21:131–152, 2005.
10. T. Lu, P. Neittaanmaki, and X.-C. Tai. A parallel splitting up method and its application to Navier-Stoke equations. *Applied Mathematics Letters*, 4:25–29, 1991.
11. T. Lu, P. Neittaanmaki, and X.-C. Tai. A parallel splitting up method for partial differential equations and its application to Navier-Stokes equations. *RAIRO Math. Model. and Numer. Anal.*, 26:673–708, 1992.
12. Barry Merriman, James K. Bence, and Stanley J. Osher. Motion of multiple functions: a level set approach. *J. Comput. Phys.*, 112(2):334–363, 1994.

13. D. Mumford and J. Shah. Optimal approximation by piecewise smooth functions and assosiated variational problems. *Comm. Pure Appl. Math*, 42:577–685, 1989.
14. S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: Algorithms based on hamilton-jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.
15. L.I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.
16. C. Samson, L. Blanc-Féraud, G. Aubert, and J. Zerubia. A level set model for image classification. *IJCV*, 40(3):187–197, 2000.
17. J. Shen. Gamma-convergence approximation to piecewise constant Mumford-Shah segmentation. Tech. Rep. CAM05-16, UCLA Dep. Math, 2005.
18. B. Song and T.F. Chan. Fast algorithm for level set segmentation. *UCLA CAM report 02-68*, 2002.
19. X.-C. Tai, O. Christiansen, P. Lin, and I. Skjælaaen. Fast implementation of piecewise constant level set methods. *International Journal of Computer Vision*, to appear.
20. X.-C. Tai and C. Yao. Fast piecewise constant level set methods (PCLSM) with Newton updating. *UCLA CAM 05-52*, 2005.
21. Xue-Cheng Tai and Hongwei Li. Piecewise constant level set methods (PCLSM) for elliptic inverse problems. Cam-report-05-59, UCLA, Applied Mathematics, 2005.
22. Luminita A. Vese and Tony F. Chan. A new multiphase level set framework for image segmentation via the Mumford and Shah model. *International Journal of Computer Vision*, 50:271–293, 2002.
23. J. Weickert, B.H. Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE TRans. Image Process.*, 7:398–409, 1998.
24. Joachim Weickert and Gerald Kühne. Fast methods for implicit active contour models. In *Geometric level set methods in imaging, vision, and graphics*, pages 43–57. Springer, New York, 2003.
25. Hong-Kai Zhao, T. Chan, B. Merriman, and S. Osher. A variational level set approach to multiphase motion. *J. Comput. Phys.*, 127(1):179–195, 1996.