A final version of this paper is published in the IEEE proceedings:
DOI: http://dx.doi.org/10.1109/ICSPCS.2011.6140840

A COMPARATIVE STUDY ON THE OPTIMIZATION OF GLOBAL OPTIMUM ACTIVE CONTOURS

Jonas De Vylder, Jan Aelterman, Wilfried Philips

Ghent University - IBBT - Dept. of Telecommunications and Information Processing
St.-Pietersnieuwstraat 41, B-9000 Ghent, Belgium
Jonas.deVylder@telin.ugent.be

ABSTRACT

The active contour framework is widely used for segmentation of images. Specially active contours with a convex energy gained a lot of interest, since they are independent of initialization. Several methods have been proposed to minimise the active contour energy, all claiming to be fast and efficient. In this paper we study and compare the convergence and constraints of these methods. We also propose a new optimization method specially designed for the optimization of convex energy active contours.

Index Terms— Active Contours, Image Segmentation, Convex optimization

1. INTRODUCTION

Since Kass et al. [1] introduced the active contours, the framework has become a constant recurring topic in segmentation literature [2, 3, 4, 5, 6]. In the active contour framework, an initial contour is moved and deformed in order to minimise a specific energy function. This energy function should be minimal when the contour is delineating the object of interest. Two main groups can be distinguished in the active contour framework: one group representing the active contour explicitly as a parameterized curve and a second group which represents the contour implicitly, e.g. using level-sets. In the first group, also called snakes, the contour commonly converges towards edges in the image [1, 4]. The second group generally has an energy function based on region properties, such as the variance of intensity of the enclosed segment [3, 7]. These level-set approaches have gained a lot of interest since they have some benefits over snakes. For example, they can easily change their topology, e.g. splitting a segment into multiple unconnected segments.

Recently an active contour model has been proposed with a convex energy function, making it possible to define fast global optimizers [5, 6]. These global active contours have the benefit that their result no longer depends on the initialization. In [8], Bresson et al. proposed an active contour with a convex energy function which combines edge information and region information. This method combines the original snake model [1] with the active contour model without edges [3].

Due to the convexity of the active contour energy function, several efficient optimizers have been proposed to calculate the active contours [5, 8, 6, 9, 10]. Although these methods all show fast convergence, they haven't been properly compared. In this paper we will discuss several of these state of the art optimization methods and experimentally compare their convergence. This paper also proposes a new optimizers which is a combination of two "simple" optimizers. The proposed method shows good convergence, while using little memory and having a low computational cost.

This paper is arranged as follows: the next section defines some notations used in this paper. The following section briefly summarizes the theory of the active contour framework. Section 4 discusses a range of optimizers which can be used for the active contour optimization. Then, section 5 experimentally tests and compares the optimization methods. Section 6 recapitulates and concludes.

2. NOTATIONS AND DEFINITIONS

In the remaining of this paper we will use specific notations. To make sure all notations are clear, we briefly summarize the notations and symbols used in this work.

We will refer to an image, F in its line scanning order, i.e. $\mathbf{f}(i*m+j) = F(i,j)$, where $m \times n$ are the dimensions of the image. In a similar way we will represent the active contour in vector format, $\mathbf{u}$. If a pixel $U(i,j)$ is part of the segment, it will have a value above a certain threshold, all background pixels will have a value lower than the given threshold. Note that this is similar to level-sets. The way these contours are optimized however is different than with classical level-set active contours, as is explained in the next section. We will use image operators, i.e. gradient, divergence and Laplacian in combination with this vector notation, however the semantics of the image operators remains the same as if it was used on

This research has been made possible by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT).

a matrix:

$$\begin{aligned}\nabla^+\mathbf{f}(t) &= \left(\nabla_x^+\mathbf{f}(t), \nabla_y^+\mathbf{f}(t)\right)\\ \nabla_\cdot^+\mathbf{f}(t) &= \nabla_x^+\mathbf{f}(t) + \nabla_y^+\mathbf{f}(t)\\ \nabla^2\mathbf{f}(t) &= \nabla_\cdot^+\nabla^+\mathbf{f}(t))\end{aligned}$$

where

$$\begin{aligned}\nabla_x^+\mathbf{f}(i*m+j) &= F(i+1,j) - F(i,j)\\ \nabla_y^+\mathbf{f}(i*m+j) &= F(i,j+1) - F(i,j)\end{aligned}$$

Each of these operators can also be defined using the backwards derivative operators:

$$\begin{aligned}\nabla_x^-\mathbf{f}(i*m+j) &= F(i,j) - F(i-1,j)\\ \nabla_y^-\mathbf{f}(i*m+j) &= F(i,j) - F(i,j-1)\end{aligned}$$

If no confusion is possible, we will use the forward operators and omit the $+$ sign for the sake of readability. Further we will use the following inner product and norm notations:

$$\begin{aligned}\langle\mathbf{f},\mathbf{g}\rangle &= \sum_{i=1}^{mn} f(i)g(i)\\ |\mathbf{f}|_\mathbf{g} &= \sum_{i=1}^{mn} g(i) \mid f(i) \mid\\ \|\mathbf{f}\|_2 &= \sqrt{\sum_{i=1}^{mn} f(i)^2}\end{aligned}$$

If the weights $g(i) = 1$ for all i, then we will omit $\mathbf{g}$, since we assume this will not cause confusion, but will increase readability.

3. CONVEX ENERGY ACTIVE CONTOURS

In [5] an active contour model was proposed which has global minimisers. This active contour is calculated by minimizing the following convex energy:

$$E[\mathbf{u}] = |\nabla\mathbf{u}| + \gamma\langle\mathbf{u},\mathbf{r}\rangle \tag{1}$$

with

$$\mathbf{r}[x] = (\mu_f - \mathbf{f}[x])^2 - (\mu_b - \mathbf{f}[x])^2 \tag{2}$$

Here $\mathbf{f}$ represents the intensity values in the image, μ_f and μ_b are respectively the mean intensity of the segment and the mean intensity of the background, i.e. every pixel not belonging to the segment. Note that this energy is convex, only if μ_f and μ_b are constant. If these values are not known in advance, they can be approximated by alternating between the following two steps: first fix μ_f and μ_b and minimise eq. (1), secondly update μ_f and μ_b. Chan et al. found that the steady state of the gradient flow corresponding to this energy, i.e.

$$\frac{d\mathbf{u}}{dt} = \nabla\cdot\frac{\nabla\mathbf{u}}{|\nabla\mathbf{u}|} - \gamma\mathbf{r} \tag{3}$$

coincides with the steady state of the gradient flow of the original Chan-Vese active contours [5, 3]. So minimizing eq. (1) is equivalent to finding an optimal contour which optimizes the original Chan-Vese energy function. Although the energy in eq. 1 does not have a unique global minimiser, a well defined minimiser can be found within the interval $[0,1]^n$:

$$\mathbf{u}^* = \arg\min_{\mathbf{u}\in[0,1]^n} |\nabla\mathbf{u}| + \gamma\langle\mathbf{u},\mathbf{r}\rangle \tag{4}$$

Note that this results in a minimiser which values are between 0 and 1. It is however desirable to have a segmentation result where the values of a minimiser are constrained to $(0,1)$, i.e. a pixel belongs to a segment or not. Therefore $\mathbf{u}^*$ is tresholded, i.e.

$$\Phi(\mathbf{u}^*[t];\alpha) = \begin{cases}1 & \text{if } \mathbf{u}^*[t] > \alpha\\ 0 & \text{otherwise}\end{cases} \tag{5}$$

with a predefined $\alpha \in [0,1]$. In [11] it is shown that $\Phi(\mathbf{u}^*[\cdot];0.5)$ is a global minimiser for the energy in eq. (1) and by extension for the energy function of the original Chan-Vese active contour model [3]. In [8] the convex energy function in eq. (1) was generalized in order to incorporate edge information:

$$E[\mathbf{u}] = |\nabla\mathbf{u}|_\mathbf{g} + \gamma\langle\mathbf{u},\mathbf{r}\rangle \tag{6}$$

where $\mathbf{g}$ is the result of an edge detector, e.g. $\mathbf{g} = \frac{1}{1+|\nabla\mathbf{f}|}$. The active contour minimizing this energy function can be seen as a combination of edge based snake active contours [1] and the region based Chan-Vese active contours [3]. In Fig. 1 two examples of active contour segmentation are shown. The left shows the original images. The middle column shows the thresholded active contours, i.e. $\Phi(\mathbf{u}^*[\cdot];0.5)$. The segmentation result is clearer in the right column, where the found object boundaries are superimposed on the original image.

4. OPTIMIZATION

Due to the convexity of the energy function in eq. (6), a wide range of "fast" minimisers have been proposed to find an optimal contour $\mathbf{u}^*$ [8, 5, 6, 9, 10, 12]. In this section we will briefly summarize and explain some of these optimization methods. In the next section these methods will be experimentally compared to each other.

4.1. Gradient Descent

Probably the simplest optimization scheme is gradient descent which minimises eq. (6) by solving the following set of Euler-Lagrange equations:

$$\nabla\cdot\left(\frac{\mathbf{g}[t]\,\nabla\mathbf{u}[t]}{|\nabla\mathbf{u}[t]|}\right) - \gamma\mathbf{r}[t] = 0 \tag{7}$$

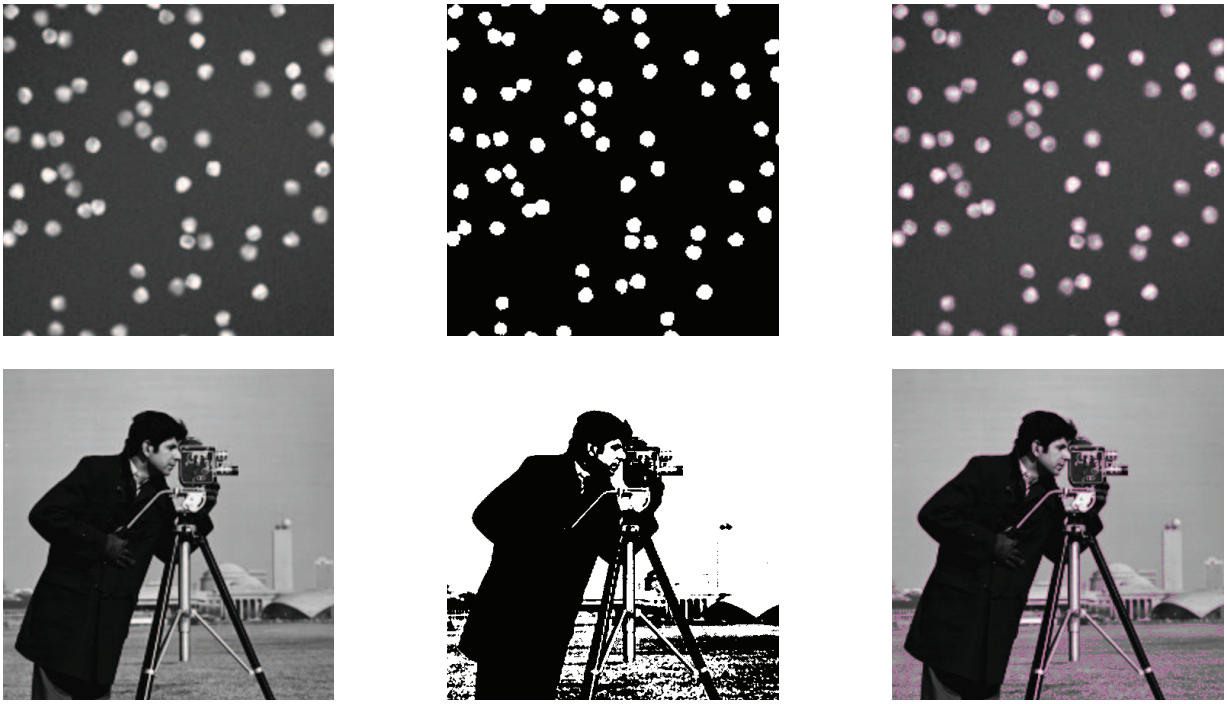

Fig. 1. Two examples of active contour segmentation. In the left column, the original images, the middle column depicts the binary result of the segmentation and the right column superimposes the boundary of the segments onto the image.

In order to constrain the solution of this equation to the interval $[0, 1]$, a convex barrier or potential function, $\Gamma(.)$ is added as an extra term to eq. (6), i.e.

$$E[\mathbf{u}] = |\nabla \mathbf{u}|_{\mathbf{g}} + \gamma \langle \mathbf{u}, \mathbf{r} \rangle + \Gamma(\mathbf{u}) \tag{8}$$

A wide variety of barrier functions have been proposed in literature, e.g. quadratic, linear, Huber, Lipschitz functions etc. In Fig. 2 some different barrier functions for the interval $[0, 1]$ are shown. For a more extensive review on different barrier functions, we refer to [13]. For this specific application, we use the piecewise linear barrier function, since it was proven in [5] that this function forces $\mathbf{u} \in [0, 1]^n$ exactly. And as can be seen in Fig. 2 ,this function does not penalize or favour any value within $[0, 1]$. This piecewise linear barrier function can be defined by

$$\Gamma(t) = \max\left(0, 2\beta\left|t - \frac{1}{2}\right| - 1\right) \tag{9}$$

where β is the maximum intensity of the image. Adding this penalty function to eq. (7) results in the following gradient descent optimization step:

$$\mathbf{u_{k+1}}[t] = \mathbf{u_k}[t] + \delta t \Psi_{gd}(\mathbf{u_k}[t]) - \delta t \frac{d\,\Gamma(\mathbf{u_k}[t])}{d\mathbf{u_k}[t]} \tag{10}$$

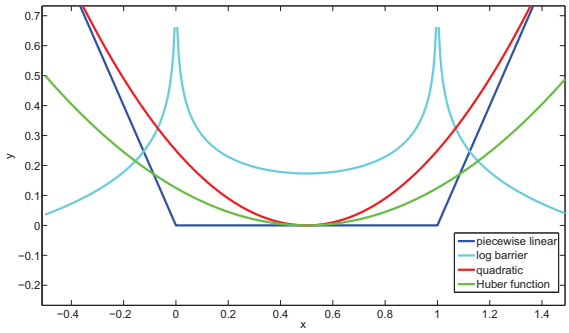

Fig. 2. Four different examples of a barrier function which can be used to keep optimizers within the interval $[0, 1]$

where

$$\Psi_{gd}(\mathbf{u}[t]) = \nabla_x^-\left(\frac{\mathbf{g}[t]\ \nabla_x^+ \mathbf{u_k}[t]}{\|\nabla^+ \mathbf{u_k}[t]\|_2}\right) + \nabla_y^-\left(\frac{\mathbf{g}[t]\ \nabla_y^+ \mathbf{u_k}[t]}{\|\nabla^+ \mathbf{u_k}[t]\|_2}\right) - \gamma \mathbf{r}[t] \tag{11}$$

4.2. Newton-Raphson

A similar approach to solve the Euler-Lagrange equations in eq. (7) is done by adding an update term according to the

Newton-Raphson optimization scheme, i.e.

$$\mathbf{u_{k+1}}[t] = \mathbf{u_k}[t] + \frac{\Psi_{gd}(\mathbf{u_k}[t])}{\Psi_{nr}(\mathbf{u_k}[t])} \tag{12}$$

where $\Psi_{nr}(.)$ is the derivative of $\Psi_{gd}(.)$, i.e.

$$\begin{aligned}
\Psi_{nr}(\mathbf{u}[t]) &= \frac{d\,\Psi_{gd}(\mathbf{u}[t])}{d\,\mathbf{u}[t]} \qquad (13)\\
&= \frac{-2\mathbf{g}[t]}{\|\nabla^+\mathbf{u_k}[t]\|_2} - \frac{\mathbf{g}[t_1]\ + \mathbf{g}[t_2]}{\|\nabla^-\mathbf{u_k}[t]\|_2}\\
&\quad + \frac{\mathbf{g}[t]\,(\nabla_x^+\mathbf{u_k}[t])\,(\nabla_\cdot^+\mathbf{u_k}[t])}{\|\nabla^+\mathbf{u_k}[t]\|_2^3}\\
&\quad + \frac{\mathbf{g}[t_1]\,(\nabla_x^-\mathbf{u_k}[t])\,(\nabla_\cdot^-\mathbf{u_k}[t])}{\|\nabla^-\mathbf{u_k}[t]\|_2^3}\\
&\quad + \frac{\mathbf{g}[t]\,(\nabla_y^+\mathbf{u_k}[t])\,(\nabla_\cdot^+\mathbf{u_k}[t])}{\|\nabla^+\mathbf{u_k}[t]\|_2^3}\\
&\quad + \frac{\mathbf{g}[t_2]\,(\nabla_y^-\mathbf{u_k}[t])\,(\nabla_\cdot^-\mathbf{u_k}[t])}{\|\nabla^-\mathbf{u_k}[t]\|_2^3}
\end{aligned}$$

Where t_1 and t_2 are the left and top neighbour of pixel t. The Newton-Raphson theoretically has a quadratic convergence, the real convergence rate however might be significantly lower due to the non differentiability of eq. (7) at places where $\nabla\mathbf{u}[t] = 0$. In section 5 the real convergence speed is experimentally tested.

4.3. Line Search

In the gradient descent method the variable $\mathbf{u}[t]$ is updated according to the gradient direction with a constant step size δt. A variation on this optimization scheme is given by a line search algorithm, where the updating direction is the same as with gradient descent, but where the step size is variable. We propose a new line search algorithm, where the step size is chosen such that the value of $\mathbf{u}[t] \in (0, 1)$. This is a reasonable assumption, since a thresholded version of an optimal $\mathbf{u}^*$ is also optimal (see section 3 or [11]). So instead of looking for an optimal $\mathbf{u}^*$ and then threshold it, we will immediately search for a binary optimum. This is achieved by the following updating scheme:

$$\mathbf{u_{k+1}}[t] = \mathbf{u_k}[t] + \Psi_{ls}(\mathbf{u_k}[t])\,\mathrm{sgn}(\Psi_{gd}(\mathbf{u_k}[t])) \tag{14}$$

where $\Psi_{ls}(.)$ represents the step size, i.e.

$$\Psi_{ls}(\mathbf{u}[t]) = \begin{cases} 0 & \text{if } \mathbf{u}[t] = 0 \text{ and} \Psi_{gd}(\mathbf{u}[t]) < 0 \\ 0 & \text{if } \mathbf{u}[t] = 1 \text{ and} \Psi_{gd}(\mathbf{u}[t]) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

This step size results in an optimization step where $\mathbf{u}[t]$ either keeps the same value or where it switches its value. Assume for example that $\mathbf{u_k}[t] = 0$ and that the gradient of eq. (7) points in the direction of $-\infty$, then $\mathbf{u_{k+1}}[t]$ will remain zero. If the gradient pointed in the direction of $+\infty$ however, $\mathbf{u_k}[t]$ would switch to the value one. Note that this method is more a heuristic than a real optimization technique, i.e. there is no proof of convergence. In section 5 the convergence results of this method will be tested and discussed.

4.4. Dual formulation

A different approach is proposed in [8, 9]. In this approach an optimization scheme is calculated based on the dual formulation of the total variation norm. Instead of optimizing eq. (6) directly for $\mathbf{u}$, an extra variable $\mathbf{v}$ is introduced in the energy term, i.e. the following energy term is optimized for $\mathbf{u}$ and $\mathbf{v}$:

$$E[\mathbf{u}, \mathbf{v}] = |\nabla\mathbf{u}|_\mathbf{g} + \gamma\langle\mathbf{v}, \mathbf{r}\rangle + \frac{\lambda}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 + \Gamma(\mathbf{u}) \tag{16}$$

where $\Gamma(.)$ is the same barrier function as in subsection 4.1 and λ is a weighting parameter, defining the influence of the similarity between $\mathbf{u}$ and $\mathbf{v}$. Due to the convexity of the energy function, we can optimize $\mathbf{u}$ and $\mathbf{v}$ independently. This results in alternating between the following two optimization problems:

$$\arg\min_{\mathbf{u}} \qquad |\nabla\mathbf{u}|_\mathbf{g} + \frac{\lambda}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 \tag{17}$$

$$\arg\min_{\mathbf{v}} \qquad \frac{\lambda}{2}\|\mathbf{u} - \mathbf{v}\|_2^2 + \gamma\langle\mathbf{v}, \mathbf{r}\rangle + \Gamma(\mathbf{u}) \tag{18}$$

In [8] it was proven that the optimal $\mathbf{u}$ for eq. (17) is given by:

$$\mathbf{u_{k+1}}[\mathbf{t}] = \mathbf{v_k}[t] - \frac{1}{\lambda}\nabla_\cdot^+\mathbf{p}[t] \tag{19}$$

where $\mathbf{p}$ satisfies

$$\mathbf{g}[t]\,\nabla\big(\frac{1}{\lambda}\nabla\cdot\mathbf{p}[t] - \mathbf{v}[t]\big) - \Big|\nabla\big(\frac{1}{\lambda}\nabla\cdot\mathbf{p}[t] - \mathbf{v}[t]\big)\Big|\mathbf{p}[t] = 0 \tag{20}$$

Given this set of equations, $\mathbf{p}$ can be calculated using the fixed point algorithm described in Algorithm 1. experiments show

Algorithm 1: A fixed point optimization algorithm which can be used to calculate a proper $\mathbf{p}$ for eq. 20.

1 $\mathbf{p_0}[\mathbf{t}] = (0, 0)$

2 $\mathbf{p_{k+1}}[\mathbf{t}] = \frac{\mathbf{p_k}[\mathbf{t}] + \delta t\ \nabla^-(\nabla_\cdot^+\mathbf{p_k}[\mathbf{t}] - \lambda\mathbf{v_k}[t])}{1 + \delta t\ |\nabla^-(\nabla_\cdot^+\mathbf{p_k}[\mathbf{t}] - \lambda\mathbf{v_k}[t])|}$

that a single run of this fixed point method, already results in good convergence properties for the optimization of eq. (17) and by extension for the optimization of eq. (16). The optimization of $\mathbf{v}$ in eq. (18) is give by the following closed solution:

$$\mathbf{v_{k+1}} = \min\Big(\max\big(\mathbf{u_{k+1}} - \frac{\mu}{\lambda}\mathbf{c}, 0\big), 1\Big) \tag{21}$$

For the proof of this closed solution, we refer to [8].

4.5. Split Bregman

A different optimization method is Split Bregman optimization, which is similar to the augmented Lagrangian method and is considered to be an efficient optimization technique for solving L1 regularized problems [14, 6, 15]. Similar to the optimization based on the dual formulation, the split Bregman method will "de-couple" the L1 norm and the inner product, by introducing a new variable $\mathbf{d}$ and by putting constraints on this new variable. This results in the following optimization problem:

$$E[\mathbf{u}, \mathbf{d}] = |\mathbf{d}|_{\mathbf{g}} + \gamma\langle \mathbf{u}, \mathbf{p}\rangle \text{ such that } \mathbf{d} = \nabla\mathbf{u} \tag{22}$$

This optimization problem can be converted to an unconstrained problem by adding a quadratic penalty function, i.e.

$$E[\mathbf{u}, \mathbf{d}] = |\mathbf{d}|_{\mathbf{g}} + \gamma\langle \mathbf{u}, \mathbf{p}\rangle + \frac{\lambda}{2}|\mathbf{d} - \nabla\mathbf{u}|_2^2 \tag{23}$$

Where λ is a weighting parameter. If γ is large, $\mathbf{d} = \nabla\mathbf{u}$. However setting γ high introduces numerical instability. Note that that when λ is not high, the quadratic penalty function only approximates the constraint $\mathbf{d} = \nabla\mathbf{u}$. By using a Bregman iteration technique [14], this constraint can be enforced exactly, in an efficient way. In the Bregman iteration technique an extra vector, $\mathbf{b_k}$ is subtracted from the penalty function. This results in the following two unconstrained steps.

$$\begin{aligned}(\mathbf{u_{k+1}}, \mathbf{d_{k+1}}) &= \arg\min_{\mathbf{u_k}, \mathbf{d_k}} |\mathbf{d_k}|_{\mathbf{g}} + \gamma\langle \mathbf{u_k}, \mathbf{p}\rangle \\ &\quad + \frac{\lambda}{2}|\mathbf{d_k} - \nabla\mathbf{u_k} - \mathbf{b_k}|_2^2 \qquad (24)\\ \mathbf{b_{k+1}} &= \mathbf{b_k} + \nabla\mathbf{u_{k+1}} - \mathbf{d_{k+1}} \qquad (25)\end{aligned}$$

The first step requires optimizing for two different vectors, $\mathbf{u}$ and $\mathbf{d}$. Since the constrained optimization step is convex, these optimal vectors can be calculated by alternating between optimizing eq. (24) for $\mathbf{u}$ and optimizing for $\mathbf{d}$:

$$\begin{aligned}\mathbf{u_{k+1}} &= \arg\min_{\mathbf{u_k}} \gamma\langle \mathbf{u_k}, \mathbf{p}\rangle + \frac{\lambda}{2}|\mathbf{d_k} - \nabla\mathbf{u_k} - \mathbf{b_k}|_2^2 \\ \mathbf{d_{k+1}} &= \arg\min_{\mathbf{d_k}} |\mathbf{d_k}|_{\mathbf{g}} + \frac{\lambda}{2}|\mathbf{d_k} - \nabla\mathbf{u_{k+1}} - \mathbf{b_k}|_2^2\end{aligned} \tag{26}$$

The first problem can be optimized by solving a set of Euler-Lagrange equations. For each element $u(i)$ of the optimal $\mathbf{u}$ the following optimality condition should be satisfied:

$$\nabla^2\mathbf{u}[t] = \frac{\gamma}{\lambda}\mathbf{c}[t] + \nabla\cdot(\mathbf{d}[t] - \mathbf{b}[t]) \tag{27}$$

Note that this system of equations can be written as $A\mathbf{u} = \mathbf{w}$. In [6] they proposed to solve this linear system using the iterative Gauss-Seidel method. However, in order to guarantee the convergence of the Gauss-Seidel method, A should be strictly diagonally dominant or should be positive semi definite. Unfortunately A is neither. Therefore we will optimize eq. (27) using the iterative conjugate residual method, which is a Krylov subspace method for which convergence is guaranteed if A is Hermitian [16]. Experiments showed that due to the fact that this system is not constrained to $[0, 1]$, too many iterations of the conjugate residual method results in numerical unstable solutions. In [6] it was observed that the split Bregman optimization converges even with just an approximation of the solution of the linear system in eq. (27). Therefore, we propose to use only one iteration of the conjugate residual method. Using just one iteration enables to get a reduced algorithm which needs less calculations and less memory. This reduced conjugate residual algorithm is shown in Algorithm 2.

Algorithm 2: The reduced conjugate residual step for a linear system

1 $\mathbf{r} = \mathbf{w} - A\mathbf{u}$
2 $\alpha = \frac{\langle \mathbf{r}, A\mathbf{r}\rangle}{\langle A\mathbf{r}, A\mathbf{r}\rangle}$
3 $\mathbf{u} = \mathbf{u} - \alpha\mathbf{r}$

The solution of eq. (27) is unconstrained, i.e. $\mathbf{u}[t]$ does not have to lie in the interval $[0, 1]$. Note that minimizing eq. (26) for $\mathbf{u}[t]$, i.e. all other elements of $\mathbf{u}$ remain constant, is equivalent to minimise a quadratic function. If $\mathbf{u}[t] \notin [0, 1]$ then the constrained optimum is either 0 or 1, since a quadratic function is monotonic in an interval which does not contain its extremum. So the constrained optimum is given by:

$$\mathbf{u}^*[t] = \max\Big(\min\big(\mathbf{u}[t], 1\big), 0\Big) \tag{28}$$

In order to calculate an optimal $\mathbf{d_k}$ in eq. (26), a closed form solution can be calculated using the shrinking operator, i.e.

$$\mathbf{d_{k+1}}[t] = \Upsilon\Big(\nabla\mathbf{u}[t] + \mathbf{b_k}, g[t], \lambda\Big) \tag{29}$$

where

$$\Upsilon(\tau, \theta, \lambda) = \begin{cases} 0 & \text{if } |\tau| \le \frac{\theta}{\lambda} \\ \tau - \frac{\theta}{\lambda}\operatorname{sgn}(\tau) & \text{otherwise}\end{cases} \tag{30}$$

5. RESULTS

For the validation of the segmentation, a dataset of 7 pictures was assembled. These pictures come from different image modalities such as microscopy, photos, ultrasound and MRI-scans. All these images have a dimension of 512×512 pixels.

To compare the active contour result with ground truth segmentation, the Dice coefficient is used. If S is the resulting segment from the active contour, i.e. $\phi(\mathbf{u}^*, 0.5)$, and GT the

ground truth segment, then the Dice coefficient between S and GT is defined as:

$$d(S, GT) = \frac{2 \text{ Area}(S \wedge GT)}{\text{Area(S)} + \text{Area}(GT)} \tag{31}$$

where $S \wedge GT$ consist of all pixels which both belong to the detected segment as well as to the ground truth segment. If S and GT are equal, the Dice coefficient is equal to one. The Dice coefficient will approach zero if the regions hardly overlap. Note that ground truth is a relative concept, what is considered to be a foreground segment for one application might be clutter for a different application. Instead we will compare the calculated active contour with the optimal contour, i.e. the contour with minimal energy, $\mathbf{u}^*$. Therefore, the active contour is calculated using all methods described in the previous section. The active contour which had the lowest energy after 300 iteration is considered to be the ground truth.

All methods were implemented in C and run on a computer with an Intel i7 Q720 1.6 GHz CPU with 4GB RAM.

5.1. Convergence

As a first experiment, each method runs 300 iterations on the test dataset. After each iteration, the Dice coefficient between the thresholded active contour and the ground truth was calculated. In the top of Fig. 3 the average Dice coefficient after each iteration is plotted for each method. As can be seen is the gradient descent method the slowest. All other methods converge fast at the first few iterations, but then slows down. In the bottom part of Fig. 3 a close up of the top plot is shown. The line search converges much faster during the first iteration then other methods, but then starts oscillating, and never really converges. Optimization based on the dual formulation of the TV-norm is the second fastest method and does converge to a global optimum. Both the Newton-Raphson as the split Bregman method converge slower than previous methods. Neither do they reach the real optimum due to numerical errors. In order to benefit from the fast start from the line search optimization, we propose to combine the line search scheme and the Newton-Raphson optimization scheme. We apply a single updating step based on line search and for all other iterations we apply the Newton-Raphson updating step. This combined optimization reaches the real optimum, while converging faster than all other methods. Since the rate of convergence does not say everything, we summarize some extra measurement in Table 1. The second column of this table contains the theoretical time complexity of one iteration of each method. For the primal-dual and the split Bregman method we explicitly mentioned the amount that a matrix of dimension of $n \times m$ has to be scanned. For the primal-dual method consists this of one scan for the calculation of $\mathbf{p}$, one run for the calculation of $\mathbf{u}$ and a final scan for $\mathbf{v}$. The split Bregman method needs 9 runs in total: 7 runs needed in the conjugate residual optimization and two runs to update the vectors $\mathbf{d}$ and $\mathbf{b}$. The conjugate residual needs a single run to calculate the right side of the Euler-Lagrange equations, two runs to calculate a Laplacian, one run to calculate the residuals, two runs to calculate an inner product and finally a single run to update $\mathbf{u}$.

The third column of Table 1 contains the average time to calculate a single iteration of a 512×512 image. The split Bregman has the highest computational cost, as was expected by the theoretical complexity of the methods. All other methods result in a similar computational time, except for the line search method which is the fastest to calculate. The last column summarizes the memory required for each method. We have omitted the memory needed to store $\mathbf{u_k}$ and $\mathbf{u_{k+1}}$ since this is the same for each method. Only the primal-dual method and the split Bregman method require more memory then to store a temporal variable. The primal-dual method needs to store $\mathbf{v}$ and $\mathbf{p}$, $\mathbf{v}$ has the same size as the image, whereas $\mathbf{p}$ is twice as big as the image. For the split Bregman the extra vectors $\mathbf{d}$ and $\mathbf{b}$ have to be stored, each twice the size of the image. The conjugate residual method, needed for the optimization of the split Bregman method, requires three times the amount of memory to store the image. This is needed to store the residuals, the Laplacian of the contour itself and the Laplacian of the residuals. The Laplacians could be recalculated, instead of stored, but this would lead to slower iterations.

5.2. Scalability

To test the scalability of the different optimization algorithms, the average calculation times of a single iteration was measured for images of different sizes. These images have a dimension of $N \times N$ where $N = 2^7, 2^8, ..., 2^{12}$. The results of these time measurements are shown in Fig. 4. As can be seen does the split Bregman method slow down much faster than the other methods. The line search method remains the fastest to calculate independent of the image size. The combination of the line search method and the Newton-Raphson method is slower than optimization only based on the line search scheme, but is still faster to calculate than the other proposed methods. A side from computational speed, is the memory requirements an extra factor determining the scalability. Although the primal-dual method is reasonably fast for large 2D images, e.g. images of 4096×4096 pixels, it does require three times the amount of memory needed by the gradient based methods. This memory constraint will hamper the method for even bigger datasets, e.g. for segmentation of 3D images.

5.3. Robustness

The experiments done in subsection 5.2 are all done on using the same parameters. Although the measurements are done on different types of images, e.g. microscopy, photos, etc., these measurements say little of the influence of noise or on

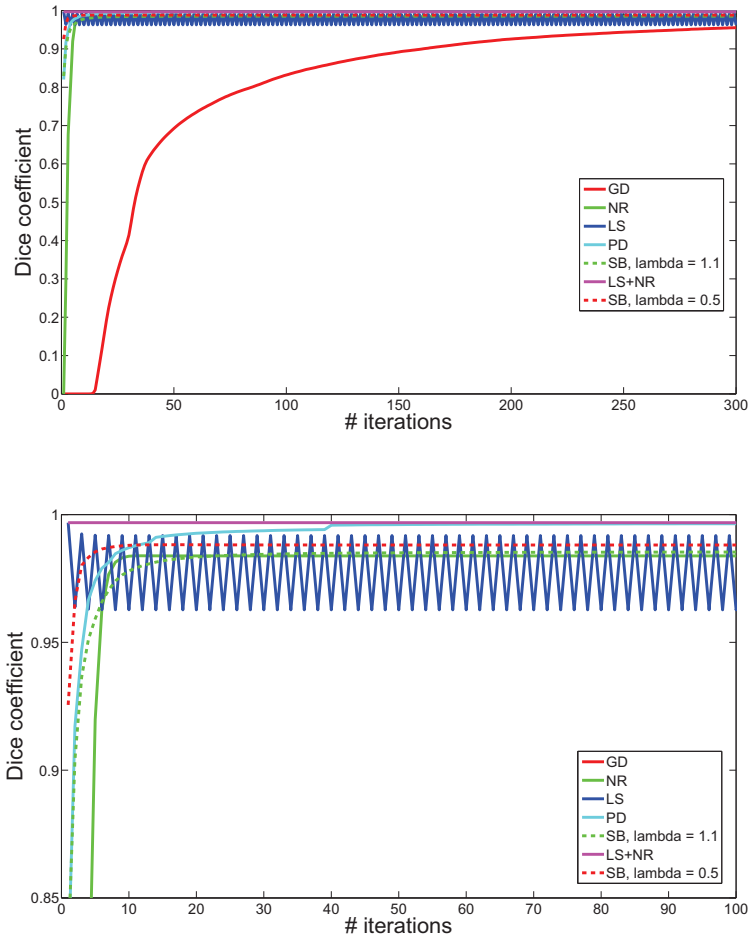

Fig. 3. A comparison of the convergence of the Dice coefficient between ground truth and the active contours, calculated using different methods. The methods used for the active contour optimization are: gradient descent, Newton-Raphson, line search, primal-dual, split Bregman and a combination of line search and Newton-Raphson. The top plot shows an overview whereas the bottom plot shows a close up of the same plot.

method	Time complexity	Time measurement (ms)	Memory
Gradient descent	$O(n \times m)$	30.83	$O(1)$
Newton-Raphson	$O(n \times m)$	33.91	$O(1)$
Line Search	$O(n \times m)$	25.79	$O(1)$
Primal-Dual	$O(3 \times n \times m)$	33.8	$O(3 \times n \times m)$
Split Bregman	$O(9 \times n \times m)$	60.09	$O(7 \times n \times m)$
Line search + Newton-Raphson	$O(n \times m)$	32.49	$O(1)$

Table 1. A Comparison of the different methods

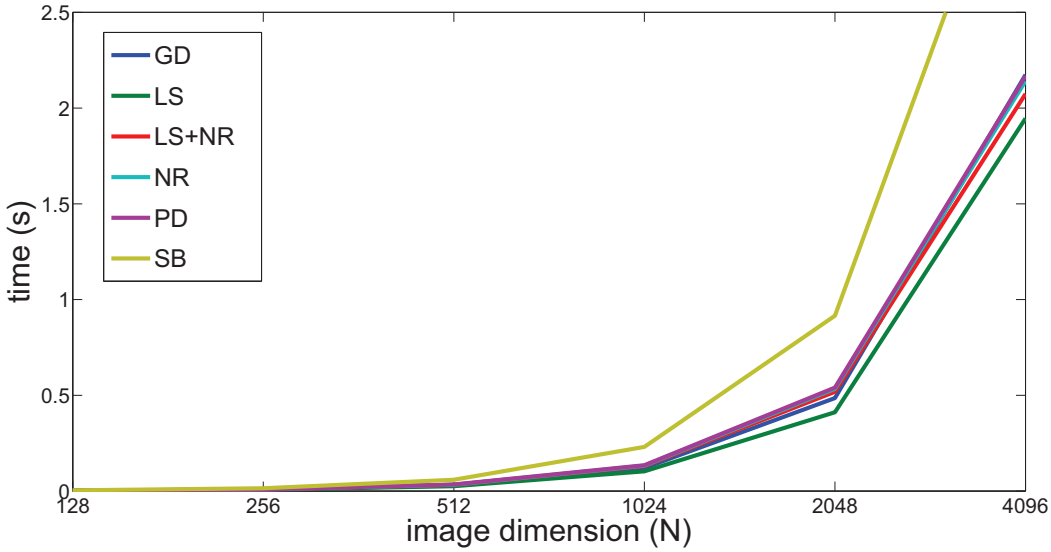

Fig. 4. The time needed to calculate a single optimization iteration, depending on the image size. The images have dimension $N \times N$.

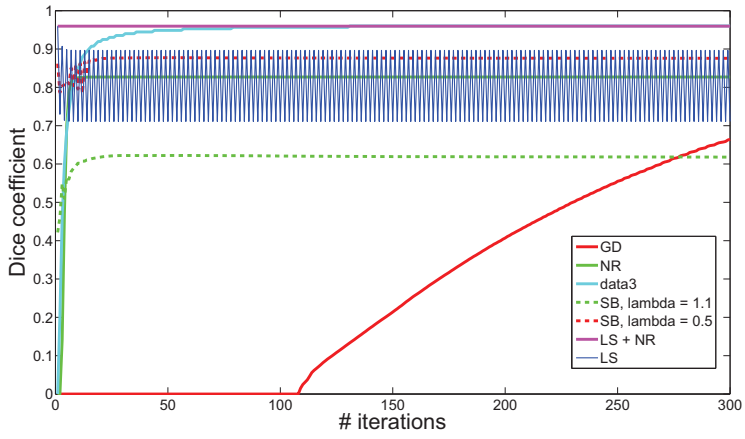

Fig. 5. A comparison of the convergence of the active contours for noisy images with stronger regularization. The Dice coefficient between ground truth and the active contours are plotted for each iterations step.. The methods used for the active contour optimization are: gradient descent, Newton-Raphson, line search, primal-dual, split Bregman and a combination of line search and Newton-Raphson.

the influence of the regularization term. To test these influences, we added white Gaussian noise to the images. The active contours where optimized with a small weight for the data fit term, i.e. $\gamma = 0.0001$, thus emphasizing the influence of the total variation regularization term. A new set of ground truth images was calculated for these noisy images in the same way as described in subsection 5.1. Fig. 5 shows the results of the convergence measured with the Dice coefficient. Just as with the non noisy images, do both the primal-dual optimization as the combination of line search and Newton-Raphson result in the best convergence. The convergence of the split Bregman method clearly varies depending on the optimization parameter λ. Note that the both λ values resulted in reasonable convergence in subsection 5.2, while $\lambda = 1.1$ results in poor convergence on noisy images. So in order to have good convergence, the split Bregman optimization needs proper parameter tuning for each different application or image dataset.

6. CONCLUSION

This paper compared several optimization methods for the optimization of convex energy active contours. A side from state of the art optimizers, new optimization method based on a combination of a line search and a Newton-Raphson was proposed. All methods where compared for the segmentation of both regular as noisy images. The proposed combination method converges significantly faster than other methods discussed, while taking less time to calculate a single iteration than most methods, e.g. the split Bregman optimization. The proposed method also needs significantly less memory than other state of the art methods, e.g. the primal-dual method.

7. ACKNOWLEDGEMENTS

This research has been made possible by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT).

8. REFERENCES

[1] M. Kass, A Witkin, and D. Terzopoulos, "Snakes: active contour models," *International journal of computer vision*, pp. 321–331, 1988.

[2] M. Isard and A. Blake, *Active contours*, Springer, 1998.

[3] T. Chan and L. Vese, "An active contour model without edges," *Scale-Space Theories in Computer Vision*, vol. 1682, pp. 141–151, 1999.

[4] M.-A. Charmi, S. Derrode, and S. Ghorbel, "Fourier-based geometric shape prior for snakes," *Pattern Recognition Letters*, vol. 29, pp. 897–904, 2008.

[5] T. F. Chan, S. Esedoglu, and M. Nikolova, "Algorithms for finding global minimizers of image segmentation and denoising models," *Siam Journal on Applied Mathematics*, vol. 66, no. 5, pp. 1632–1648, 2006.

[6] T. Goldstein, X. Bresson, and S. Osher, "Geometric applications of the split bregman method: Segmentation and surface reconstruction," *Journal of Scientific Computing*, vol. 45, no. 1-3, pp. 272–293, 2010.

[7] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, "Fast geodesic active contours," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1467–1475, 2001.

[8] X. Bresson, S. Esedoglu, P. Vandergheynst, J. P. Thiran, and S. Osher, "Fast global minimization of the active contour/snake model," *Journal of Mathematical Imaging and Vision*, vol. 28, no. 2, pp. 151–167, 2007.

[9] A. M. Yip, D. Krishnan, and Q. V. Pham, "A primal-dual active-set algorithm for bilaterally constrained total variation deblurring and piecewise constant mumford-shah segmentation problems," *Advances in Computational Mathematics*, vol. 31, no. 1-3, pp. 237–266, 2009.

[10] Dai-Qiang Chen, Hui Zhang, and Li-Zhi Cheng, "A fast fixed point algorithm for total variation deblurring and segmentation," *Journal of Mathematical Imaging and Vision*, pp. 1–13, 2011.

[11] X. Bresson and T. F. Chan, "Active contours based on chambolle's mean curvature motion," *2007 IEEE International Conference on Image Processing, Vols 1-7*, pp. 33–363371, 2007.

[12] Stephen P. Boyd and Lieven Vandenberghe, *Convex optimization*, Cambridge University Press, Cambridge, UK ; New York, 2004.

[13] T. Pock, D. Cremers, H. Bischof, and A. Chambolle, "Global solutions of variational models with convex regularization," *Siam Journal on Imaging Sciences*, vol. 3, no. 4, pp. 1122–1145, 2010.

[14] T. Goldstein and S. Osher, "The split bregman method for l1-regularized problems," *Siam Journal on Imaging Sciences*, vol. 2, no. 2, pp. 323–343, 2009.

[15] H. Mao, H. Liu, and P. Shi, "A convex neighbor-constrained active contour model for image segmentation," in *IEEE International Conference on Image Processing*, 2010.

[16] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, Philadelphia, 2nd edition, 2003.