Pattern Recognition Letters 34 (2013) 219–225

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Fast gradient vector flow computation based on augmented Lagrangian method

Dongwei Ren [a], Wangmeng Zuo [a,*], Xiaofei Zhao [a], Zhouchen Lin [b], David Zhang [a,c]

[a] Biocomputing Research Centre, School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China
[b] Key Laboratory of Machine Perception (MOE), School of EECS, Peking University, Beijing, 100871, China
[c] Biometrics Research Centre, Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong

ARTICLE INFO

Article history:
Received 9 March 2012
Available online 4 October 2012

Communicated by G. Borgefors

Keywords:
Gradient vector flow
Convex optimization
Augmented Lagrangian method
Fast Fourier transform
Multiresolution method

ABSTRACT

Gradient vector flow (GVF) and generalized GVF (GGVF) have been widely applied in many image processing applications. The high cost of GVF/GGVF computation, however, has restricted their potential applications on images with large size. Motivated by progress in fast image restoration algorithms, we reformulate the GVF/GGVF computation problem using the convex optimization model with equality constraint, and solve it using the inexact augmented Lagrangian method (IALM). With fast Fourier transform (FFT), we provide two novel simple and efficient algorithms for GVF/GGVF computation, respectively. To further improve the computational efficiency, the multiresolution approach is adopted to perform the GVF/GGVF computation in a coarse-to-fine manner. Experimental results show that the proposed methods can improve the computational speed of the original GVF/GGVF by one or two order of magnitude, and are more efficient than the state-of-the-art methods for GVF/GGVF computation.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

Gradient vector flow (GVF) field (Xu and Prince, 1998a) was first introduced as a new external force to address the two key problems in parametric active contour model (ACM) (Kass et al., 1987): the small capture range of the external forces and difficulties of progressing into boundary concavities. Xu and Prince (1998b) further proposed a generalized GVF (GGVF) external field to improve the convergence to long thin boundary indentations by incorporating two spatially varying weights. Besides parametric ACM, GVF had also been adopted by Paragios et al. (2004) in geometric ACM for image segmentation.

Moreover, the applications of GVF can be extended to other image processing tasks, e.g., tracking, denoising, restoration, and skeletonization. In (Ray and Acton, 2004), by embedding the motion direction in the GVF energy functional using a regularized Heaviside function, Ray and Acton proposed a motion GVF external force for tracking rolling leukocyte. In (Yu and Chua, 2006), Yu and Chua introduced GVF field in the field of image restoration to reformulate several popular anisotropic diffusion models, e.g., shock filter, mean curvature flow, and Perona–Malik model, to obtain better robustness against noise and spurious edges with improved high-order derivative estimation. In (He et al., 2008), GVF was used as new intensity diffusion direction for better color photo denoising. In (Ghita and Whelan, 2010), Ghita and Whelan proposed a new GVF field formulation for adaptive denoising of mixed noise. In (Hassouna and Farag, 2009), Hassouna and Farag incorporated GVF in the variational skeleton model by introducing modified GVF medial function to improve the accuracy and robustness of the curve skeleton method.

Despite its success and popularity, the GVF method requires a high computational cost, which has restricted their potential applications to images with large sizes. One possible solution is to develop alternative external forces which can be efficiently computed. For example, Li and Acton (2007) proposed a vector field convolution (VFC) external force. Another solution is to design efficient numerical schemes for fast GVF computation. In (Ntalianis et al., 2001), Ntalianis et al. proposed a multiresolution implementation of GVF computation for video object segmentation. In (Boukerroui, 2009), Boukerroui compared several efficient numerical schemes for GVF computation, and showed that the alternating direction explicit scheme (ADES) may be a suitable alternative to the multigrid method. Recently, Han et al. (2007) proposed a multigrid GVF/GGVF (MGVF/MGGVF) algorithm, which can significantly improve the computational efficiency. To the best of our knowledge, MGVF/MGGVF are the most efficient schemes for GVF/GGVF computation.

In this paper, we propose a novel fast GVF/GGVF computation scheme based on the augmented Lagrangian method (ALM). We reformulate GVF as a constrained convex optimization problem, and use an efficient optimization scheme, i.e., ALM (Afonso et al., 2010; Wang et al., 2008) with variable splitting method (Liu et al., 2010), for fast GVF and GGVF computation. Part of the paper had been presented in (Li et al., 2011). In this paper, we further proposed different variable splitting methods for GVF and GGVF

* Corresponding author. Tel.: +86 451 86412871.
E-mail address: cswmzuo@gmail.com (W. Zuo).

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.09.017

computation, respectively. Furthermore, multiresolution scheme is adopted to enhance the efficiency. Experimental results indicate that, the proposed methods can significantly improve the computational speed, and are more efficient than MGVF and MGGVF. Finally, the contributions of this paper can be summarized as follows:

(1) We show that GVF/GGVF computation are convex optimization problems, and then reformulate them to the problems which can be efficiently solved using the augmented Lagrangian methods (ALM). To the best of our knowledge, ALM is used here for the first time for fast GVF/GGVF computation.
(2) We further combine the multiresolution approach with the proposed ALM solution, resulting in the MR-IALM algorithms. Comparative studies indicate that, MR-IALM is much faster than the multigrid method (Han et al., 2007).

The remainder of the paper is organized as follows. Section 2 introduces some background knowledge, including GVF, GGVF, and the augmented Lagrangian methods. Section 3 introduces the proposed GVF and GGVF computation methods. Section 4 first evaluates the efficiency of the proposed methods, and demonstrates the application of GVF for image segmentation and image restoration. Finally, Section 5 ends this paper with several concluding remarks.

2. Prerequisites and related work

In this section, we first describe the GVF and GGVF fields, and briefly review several fast computation schemes. Then we introduce several basic ingredients on convex optimization, i.e., variable splitting and augmented Lagrangian methods.

2.1. Gradient vector flow and generalized gradient vector flow

Given the edge map $f(x,y)$ derived from the image $I(x,y)$ using any edge detector, e.g., Canny or Sobel operator, GVF and GGVF fields (Xu and Prince, 1998a,b) can be defined as a vector field $\mathbf{w}(x,y) = [u(x,y), v(x,y)]$ that minimizes the following energy functional,

$$E(\boldsymbol{w}(x,y)) = \int\int g(|\nabla f|)|\nabla \boldsymbol{w}|^2 + h(|\nabla f|)|\boldsymbol{w} - \nabla f|^2 dxdy, \tag{1}$$

where $|\cdot|$ denotes the l_2 norm with $|\nabla \boldsymbol{w}|^2 = |u_x^2 + u_y^2 + v_x^2 + v_y^2|$. For GVF, we choose $g(|\nabla f|)$ be a constant μ and $h(|\nabla f|) = |\nabla f|^2$. For GGVF, we choose $g(|\nabla f|) = e^{-(|\nabla f|/K)}$ and $h(|\nabla f|) = 1 - g(|\nabla f|)$. Based on the calculus of variations, the GVF field can be obtained by solving the partial differential equation (PDE) problems, treating $\mathbf{u}$ and $\mathbf{v}$ as functions of time t:

$$u_t(x,y,t) = g(|\nabla f|)\nabla^2 u(x,y,t) + h(|\nabla f|)[u(x,y,t) - f_h(x,y)], \tag{2}$$

$$v_t(x,y,t) = g(|\nabla f|)\nabla^2 v(x,y,t) + h(|\nabla f|)[v(x,y,t) - f_v(x,y)], \tag{3}$$

where $u(x,y,t)$ and $v(x,y,t)$ can be obtained in parallel, where $f_h(x,y)$ and $f_v(x,y)$ are the partial derivatives of $f(x,y)$ in the horizontal and vertical directions, respectively. Xu and Prince (1998a) adopted an explicit difference scheme to obtain the solutions of (2) and (3). Boukerroui (2009) tested several other numerical schemes, including the alternating direction explicit scheme (ADES), the additive operating splitting (AOS), and the locally one dimensional (LOD) methods, and showed that ADES was more appropriate for fast GVF computation.

Using the calculus of variations, the solution to (1) can be directly computed by seeking the solution to the following Euler–Lagrange equations,

$$0 = g(|\nabla f|)\nabla^2 u(x,y) + h(|\nabla f|)[u(x,y) - f_h(x,y)], \tag{4}$$

$$0 = g(|\nabla f|)\nabla^2 v(x,y) + h(|\nabla f|)[v(x,y) - f_v(x,y)]. \tag{5}$$

Here, $u(x,y)$ and $v(x,y)$ denote the final solution of $u(x,y,t)$ and $v(x,y,t)$, i.e., $u(x,y) = u(x,y,+)$ and $v(x,y) = v(x,y,+)$. In Han et al., 2007, Han et al. proposed a multigrid GVF computation scheme which adopted the full multigrid algorithm (FMG) framework Trottenberg et al., 2001 to solve the above equations.

2.2. Variable splitting and augmented Lagrangian methods

2.2.1. Variable splitting

Consider the following type of unconstrained optimization problem,

$$\min_{u\in\mathbb{R}^n} f(\boldsymbol{u}) + g(\boldsymbol{Gu}), \tag{6}$$

where $\boldsymbol{G} \in \mathbb{R}^{d\times n}$. Rather than directly solving the above problem, variable splitting reformulates the problem (6) as an equivalent constrained optimization problem,

$$\min_{\boldsymbol{u}\in\mathbb{R}^n, \boldsymbol{v}\in\mathbb{R}^d} f(\boldsymbol{u}) + g(\boldsymbol{v}), \quad \text{subject to } \boldsymbol{v} = \boldsymbol{Gu}, \tag{7}$$

by introducing an auxiliary variable $\boldsymbol{v}$. In several image processing applications (Afonso et al., 2010; Liu et al., 2010; Zuo and Lin, 2011), it is much easier to solve problem (7) than to solve the unconstrained problem (6).

2.2.2. Augmented Lagrangian method

Consider a convex optimization problem with equality constraints,

$$\min_{z\in\mathbb{R}^n} F(\boldsymbol{z}), \quad \text{subject to } \boldsymbol{Az} - \boldsymbol{b} = 0, \tag{8}$$

where $\boldsymbol{b} \in \mathbb{R}^p$ and $\boldsymbol{A} \in \mathbb{R}^{p\times n}$. The augmented Lagrangian (AL) function is then defined as,

$$L(\boldsymbol{z}, \boldsymbol{\lambda}, \sigma) = F(\boldsymbol{z}) + \boldsymbol{\lambda}^T(\boldsymbol{b} - \boldsymbol{Az}) + \frac{\sigma}{2}\|\boldsymbol{Az} - \boldsymbol{b}\|_2^2, \tag{9}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^p$ is a vector of Lagrangian multiplier and $\sigma > 0$ is the AL penalty parameter. As described in Algorithm 1, the augmented Lagrangian method (ALM) Ganesh et al., 2009, also known as the method of multipliers (MM), solves problem (9) by iteratively updating $\mathbf{z}$, λ, and σ until some convergence criterion is satisfied.

Algorithm 1. ALM/MM

1. Initialize $\boldsymbol{z}_0, \boldsymbol{\lambda}_0, \sigma_0 > 0, \rho > 0$
2. **while** not converged
3. $\quad \boldsymbol{z}_{k+1} = \arg\min_z\{\mathrm{L}(\boldsymbol{z}, \boldsymbol{\lambda}_k, \sigma_k)\}$
4. $\quad \boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \sigma_k(\boldsymbol{Az}_{k+1} - \boldsymbol{b})$
5. $\quad \sigma_{k+1} + \rho\sigma_k$
6. $\quad k \leftarrow k + 1$
7. **end while**

2.2.3. Inexact augmented Lagrangian method

One can use ALM to solve the problem (7) by defining $F(\mathbf{z}) = f(\mathbf{u}) + g(\mathbf{v})$ and choosing

$$\boldsymbol{z} = [\boldsymbol{u}^T, \boldsymbol{v}^T]^T, \quad \boldsymbol{b} = 0, \quad \boldsymbol{A} = [\boldsymbol{G}, -\boldsymbol{I}]. \tag{10}$$

Here Steps 3 and 4 of Algorithm 1 become,

$$(\boldsymbol{u}_{k+1}, \boldsymbol{v}_{k+1}) = \arg\min_{\boldsymbol{u},\boldsymbol{v}} \left\{ f(\boldsymbol{u}) + g(\boldsymbol{v}) + \frac{\sigma_{k+1}}{2}\|\boldsymbol{G}\boldsymbol{u} - \boldsymbol{v}\|_2^2 + \boldsymbol{\lambda}_k^T(\boldsymbol{v} - \boldsymbol{G}\boldsymbol{u}) \right\}, \tag{11}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \sigma_k(\boldsymbol{G}\boldsymbol{u}_{k+1} - \boldsymbol{v}_{k+1}). \tag{12}$$

For most problems, the solution to (11) is not trivial. Fortunately, we can use the inexact ALM (IALM) algorithm Wang et al., 2008; Ganesh et al., 2009, which is also called the alternating direction method, to iterate between updating $\mathbf{u}_{k+1}$ by keeping $\mathbf{v}$ fixed and updating $\mathbf{v}_{k+1}$ by keeping $\mathbf{u}$ fixed, and still guarantee the convergence and optimality.

To converge to an optimal solution, we should guarantee that σ_k is nondecreasing and $\sum_{k=1}^{+\infty}\sigma_k^{-1} = +\infty$. The conventional updating rule $\sigma_{k+1} = \rho\sigma_k$ cannot satisfy these constraints and we should requires σ_k be upper bounded, e.g. $\sigma_{k+1} = \min(\rho\sigma_k, \sigma_{max})$, and thus might result in slower convergence. In our implementation, we adopt another strategy in (Lin et al., 2009, 2011) for updating σ_k as follows:

$$\sigma^{(k+1)} = \begin{cases} \rho\sigma^{(k)}, & \text{if } \|\boldsymbol{v}_{k+1} - \boldsymbol{v}_k\|_2^2 / \|\boldsymbol{v}_{k+1}\|_2^2 < \varepsilon \\ \sigma_k, & \text{otherwise} \end{cases}, \tag{13}$$

The detail of IALM is described in Algorithm 2.

Algorithm 2. IALM

1. Initialize $\boldsymbol{z_0}, \lambda_0, \sigma_0 > 0, \rho > 0$
2. **while** not converged
3. $\boldsymbol{u}_{k+1} = \arg\min_u \left\{ f(\boldsymbol{u}) + \frac{\sigma_{k+1}}{2}\|\boldsymbol{G}\boldsymbol{u} - \boldsymbol{v}_k\|_2^2 - \boldsymbol{\lambda}_k^T(\boldsymbol{G}\boldsymbol{u}) \right\}$
4. $\boldsymbol{v}_{k+1} = \arg\min_v \left\{ g(\boldsymbol{v}) + \frac{\sigma_{k+1}}{2}\|\boldsymbol{G}\boldsymbol{u}_{k+1} - \boldsymbol{v}\|_2^2 - \boldsymbol{\lambda}_k^T\boldsymbol{v} \right\}$
5. $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \sigma_k(\boldsymbol{G}\boldsymbol{u}_{k+1} - \boldsymbol{v}_{k+1})$
6. Update σ_k *to* σ_{k+1}
7. $k \leftarrow k + 1$
8. **end while**

3. Fast GVF and GGVF computation

In this section, we present the proposed methods for fast GVF and GGVF computation by combining the augmented Lagrangian method and multiresolution scheme. For GVF and GGVF computation, we design different variable splitting methods, respectively. First, we provide a general outline of the proposed methods, then describe the augmented Lagrangian methods for GVF and GGVF computation, respectively, and finally present several remarks on the algorithm implementation.

3.1. Outline of the proposed methods

Generally, the discrete version of the GVF or GGVF energy functional can be rewritten as,

$$E(\boldsymbol{u}, \boldsymbol{v}) = (|\boldsymbol{G}\boldsymbol{D}_h\boldsymbol{u}|^2 + |\boldsymbol{G}\boldsymbol{D}_v\boldsymbol{u}|^2 + |\boldsymbol{G}\boldsymbol{D}_h\boldsymbol{v}|^2 + |\boldsymbol{G}\boldsymbol{D}_v\boldsymbol{v}|^2) + (\boldsymbol{u} - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u} - \boldsymbol{f}_h) + (\boldsymbol{v} - \boldsymbol{f}_v)^T\boldsymbol{M}(\boldsymbol{v} - \boldsymbol{f}_v), \tag{14}$$

where $\mathbf{D}_h$ and $\mathbf{D}_v$ are the gradient operator in horizontal and vertical directions, respectively, $\mathbf{G}$ is a diagonal weight matrix with $G(i,i) = (g(|\nabla f|))_i$,and $\mathbf{M}$ is a diagonal weight matrix with $M(i,i) = (h(|\nabla f|))_i$. In Eq. (14), all the terms are quadratic and the Hessian is nonnegative definite, thus the function $E(\mathbf{u}, \mathbf{v})$ is convex with respective to $\mathbf{u}$ and $\mathbf{v}$. Then, $\mathbf{u}$ and $\mathbf{v}$ can be solved individually by solving the following two unconstrained optimization problems:

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}}(\|\boldsymbol{G}\boldsymbol{D}_h\boldsymbol{u}\|^2 + \|\boldsymbol{G}\boldsymbol{D}_v\boldsymbol{u}\|^2) + (\boldsymbol{u} - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u} - \boldsymbol{f}_h), \tag{15}$$

$$\boldsymbol{v} = \arg\min_{\boldsymbol{v}}(\|\boldsymbol{G}\boldsymbol{D}_h\boldsymbol{v}\|^2 + \|\boldsymbol{G}\boldsymbol{D}_v\boldsymbol{v}\|^2) + (\boldsymbol{v} - \boldsymbol{f}_v)^T\boldsymbol{M}(\boldsymbol{v} - \boldsymbol{f}_v). \tag{16}$$

Since $\mathbf{u}$ and $\mathbf{v}$ can be computed in parallel by using the same algorithm, we only focus on the computation of $\mathbf{u}$ in the following.

Since the objective function defined in Eq. (15) is quadratic, we can directly obtain the optimal solution of $\mathbf{u}$ by solving the following system of linear equations,

$$(\boldsymbol{D}_h^T\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{D}_h + \boldsymbol{D}_v^T\boldsymbol{G}^T\boldsymbol{G}\boldsymbol{D}_v + \boldsymbol{M})\boldsymbol{u} = \boldsymbol{K}\boldsymbol{u} = \boldsymbol{M}\boldsymbol{f}_h. \tag{17}$$

However, the $mn \times mn$ matrix $\mathbf{K}$ is neither diagonal nor circular, making it is computationally expensive to obtain by direct solving the system of linear equations. Fortunately, as described in the remainder of this section, with the help of variable splitting, we can reformulate the problem in Eq. (15) to a convex problem with equality constraint, and then developed the IALM-based algorithms. Thanks to the fast convergence speed of ALM-based algorithm (Bertsekas, 1996), and all the subproblems can be efficiently solved, the proposed IALM-GVF/GGVF method is very efficient for GVF/GGVF computation. Moreover, we incorporate multiresolution scheme with IALM, resulting in the MR-IALM-GVF/GGVF algorithms.

The outline of the MR-IALM-GVF/GGVF algorithms are shown in Algorithm 3. First, we obtain the multiscale pyramid representation of the edge images. The size ratio of 2 is adopted between scales. Then we begin the GVF or GGVF computation (Please refer to Section 3.2 and Section 3.3) on the edge image of the coarsest scale, and then upsample the result as the initialization of the GVF or GGVF field on the finer scale. We continue this procedure to obtain the final GVF or GGVF field of the finest scale. In summary, we present the overall algorithm of the proposed method in Algorithm 3.

Algorithm 3. Overall Algorithm

Input: Edge image, parameter values, initial GVF or GGVF field in the coarsest level
Output: Final GVF or GGVF field

1. Construct coarse-to-fine pyramid of edge image
2. **Loop** over coarse-to-fine level
3. **Run** IALM-GVF (Section 3.2) or IALM-GGVF (Section 3.3)
4. Upsampling of the GVF or GGVF field
5. **Run** IALM-GVF (Section 3.2) or IALM-GGVF (Section 3.3) in the finest level

3.2. Inexact augmented Lagrangian method for GVF computation

For GVF, we have $g(|\nabla f|) = \mu$ and $h(|\nabla f|) = |\nabla f|^2$. Then Eq. (15) becomes,

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}} \mu(\|\boldsymbol{D}_h\boldsymbol{u}\|^2 + \|\boldsymbol{D}_v\boldsymbol{u}\|^2) + (\boldsymbol{u} - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u} - \boldsymbol{f}_h), \tag{18}$$

where $M(i,i) = f_h(i)^2 + f_v(i)^2$. Using variable splitting, the problem (18) can be reformulated as an equivalent constrained problem,

$$\begin{aligned} &\boldsymbol{u} = \arg\min_{\boldsymbol{u},\boldsymbol{u}'} \mu(\|\boldsymbol{D}_h\boldsymbol{u}\|^2 + \|\boldsymbol{D}_v\boldsymbol{u}\|^2) + (\boldsymbol{u}' - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u}' - \boldsymbol{f}_h) \\ &\text{subject to } \boldsymbol{u}' = \boldsymbol{u} \end{aligned}, \tag{19}$$

The augmented Lagrangian function of (19) is given by,

$$L(\boldsymbol{u}, \boldsymbol{u}', \boldsymbol{\lambda}) = \mu(|\boldsymbol{D}_h\boldsymbol{u}|^2 + |\boldsymbol{D}_v\boldsymbol{u}|^2) + (\boldsymbol{u}' - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u}' - \boldsymbol{f}_h) + \boldsymbol{\lambda}^T(\boldsymbol{u}' - \boldsymbol{u}) + \frac{\sigma}{2}\|\boldsymbol{u} - \boldsymbol{u}'\|_2^2. \tag{20}$$

Here Step 3 of Algorithm 2 becomes

$$\boldsymbol{u}^{(k+1)} = \arg\min_{\boldsymbol{u}} \mu \boldsymbol{u}^T(\boldsymbol{D}_h^T\boldsymbol{D}_h + \boldsymbol{D}_v^T\boldsymbol{D}_v)\boldsymbol{u} - \boldsymbol{\lambda}^{(k)T}\boldsymbol{u} + \frac{\sigma^{(k)}}{2}\|\boldsymbol{u} - \boldsymbol{u}'^{(k)}\|_2^2. \tag{21}$$

With the help of fast Fourier transform (FFT), we can derive the closed-form solution of problem (21),

$$\boldsymbol{u}^{(k+1)} = FFT^{-1}\{(FFT(\boldsymbol{\lambda}^{(k)} + \sigma^{(k)}\boldsymbol{u}'^{(k)}))\oslash(2\mu FFT(\boldsymbol{D}_h^T\boldsymbol{D}_h + \boldsymbol{D}_v^T\boldsymbol{D}_v) + \sigma^{(k)}\boldsymbol{I})\}, \tag{22}$$

where $\mathbf{I}$ is the identity matrix, $\oslash$ is the entry-wise division, and FFT^{-1} denotes the inverse fast Fourier transform. Step 4 of Algorithm 2 becomes

$$\boldsymbol{u}'^{(k+1)} = \arg\min_{\boldsymbol{u}'}(\boldsymbol{u}' - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u}' - \boldsymbol{f}_h) + \boldsymbol{\lambda}^{(k)T}\boldsymbol{u}' + \frac{\sigma^{(k)}}{2}\|\boldsymbol{u}^{(k)} - \boldsymbol{u}'\|_2^2, \tag{23}$$

and the closed-form solution of $\boldsymbol{u}'_{k+1}$ is,

$$\boldsymbol{u}'^{(k+1)} = (2\boldsymbol{M} + \sigma^{(k)}\boldsymbol{I})^{-1}(2\boldsymbol{M}\boldsymbol{f}_h - \boldsymbol{\lambda}^{(k)} + \sigma^{(k)}\boldsymbol{u}^{(k+1)}). \tag{24}$$

Similar with Algorithm 2, we update the penalty parameter as follows,

$$\sigma^{(k+1)} = \begin{cases} \rho\sigma^{(k)}, & if \|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\|_2^2/\|\boldsymbol{u}_{k+1}\|_2^2 < \varepsilon \\ \sigma^{(k)}, & otherwise \end{cases}. \tag{25}$$

Finally, we summarize the IALM-based algorithm for computing $\mathbf{u}$ in Algorithm 4.

Algorithm 4. IALM-GVF

Input: $\boldsymbol{f}_h$, μ, $\boldsymbol{M}$
Output: $\boldsymbol{u}$
1. Initialize $k = 0$, $\boldsymbol{u}^{(k)}$, $\boldsymbol{u}'^{(k)}$,$\boldsymbol{\lambda}^{(k)}$, $\sigma^{(k)} > 0$, $\rho > 1$
2. **while** not converged
3. $\mathbf{u}^{(k+1)} = FFT^{-1}\{(FFT(\boldsymbol{\lambda}^{(k)} + \sigma^{(k)}\mathbf{u}'^{(k)}))\oslash(2\ \mu\ FFT(\boldsymbol{D}_h^T\boldsymbol{D}_h + \boldsymbol{D}_v^T\boldsymbol{D}_v) + \sigma^{(k)}\boldsymbol{I})\}$
4. $\boldsymbol{u}'^{(k+1)} = (2\boldsymbol{M} + \sigma^{(k)}\boldsymbol{I})^{-1}(2\boldsymbol{M}\boldsymbol{f}_h - \boldsymbol{\lambda}^{(k)} + \sigma^{(k)}\ \boldsymbol{u}^{(k+1)})$
5. $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \sigma^{(k+1)}(\boldsymbol{u}'^{(k+1)} - \boldsymbol{u}^{(k+1)})$
6. Update $\sigma^{(k)}$ to $\sigma^{(k+1)}$
7. $k \leftarrow k + 1$
8. **end while**
9. $\boldsymbol{u} = \boldsymbol{u}^{(k)}$

3.3. Inexact augmented Lagrangian method for GGVF computation

For GGVF, we have $g(|\nabla f|) = e^{-(|\nabla f|/K)}$ and $h(|\nabla f|) = 1 - g(|\nabla f|)$. Then Eq. (15) becomes,

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}}(\|\boldsymbol{G}\boldsymbol{D}_h\boldsymbol{u}\|^2 + \|\boldsymbol{G}\boldsymbol{D}_v\boldsymbol{u}\|^2) + (\boldsymbol{u} - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u} - \boldsymbol{f}_h), \tag{26}$$

where $\mathbf{G}$ is a diagonal matrix with $G(i,i) = (g(|\nabla f|))_i$, and $\mathbf{M}$ is a diagonal matrix with $M(i,i) = (h(|\nabla f|))_i$. Since $\boldsymbol{G}^T\boldsymbol{G}$ is not equal to $\mu\boldsymbol{I}$, the use of the variable splitting strategy in Section 3.2 cannot result in efficient solution. So we adopt a new variable splitting method by introducing three auxiliary variables, $\boldsymbol{u}'$, $\boldsymbol{d}_h$, and $\boldsymbol{d}_v$, with the constraints, $\boldsymbol{u}' = \boldsymbol{u}, \boldsymbol{d}_h = \boldsymbol{D}_h\boldsymbol{u}, \boldsymbol{d}_v = \boldsymbol{D}_v\boldsymbol{u}$. Using variable splitting, the problem (26) can be formulated as an equivalent constrained problem,

$$\begin{aligned} \boldsymbol{u} = \arg\min_{\boldsymbol{u},\boldsymbol{u}',\boldsymbol{d}_h,\boldsymbol{d}_v} & \boldsymbol{d}_h^T\boldsymbol{H}\boldsymbol{d}_h + \boldsymbol{d}_v^T\boldsymbol{H}\boldsymbol{d}_v + (\boldsymbol{u}' - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u}' - \boldsymbol{f}_h) \\ \text{subject to} & \ \boldsymbol{u}' = \boldsymbol{u} \\ & \ \boldsymbol{d}_h = \boldsymbol{D}_h\boldsymbol{u} \\ & \ \boldsymbol{d}_v = \boldsymbol{D}_v\boldsymbol{u} \end{aligned}, \tag{27}$$

where $\boldsymbol{H} = \boldsymbol{G}^T\boldsymbol{G}$ is a diagonal matrix. The augmented Lagrangian function of (27) is given by,

$$\begin{aligned} L(\boldsymbol{u},\boldsymbol{u}',\boldsymbol{d}_h,\boldsymbol{d}_v,\boldsymbol{\lambda}_1,\boldsymbol{\lambda}_2,\boldsymbol{\lambda}_3) = {} & \boldsymbol{d}_h^T\boldsymbol{H}\boldsymbol{d}_h + \boldsymbol{d}_v^T\boldsymbol{H}\boldsymbol{d}_v + (\boldsymbol{u}' - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u}' - \boldsymbol{f}_h) + \boldsymbol{\lambda}_1^T(\boldsymbol{u}' - \boldsymbol{u}) + \frac{\sigma_1}{2}\|\boldsymbol{u} - \boldsymbol{u}'\|_2^2 \\ & + \boldsymbol{\lambda}_2^T(\boldsymbol{d}_h - \boldsymbol{D}_h\boldsymbol{u}) + \frac{\sigma}{2}\|\boldsymbol{d}_h - \boldsymbol{D}_h\boldsymbol{u}\|_2^2 \\ & + \boldsymbol{\lambda}_3^T(\boldsymbol{d}_v - \boldsymbol{D}_v\boldsymbol{u}) + \frac{\sigma}{2}\|\boldsymbol{d}_v - \boldsymbol{D}_v\boldsymbol{u}\|_2^2. \end{aligned} \tag{28}$$

Note that we use the penalty parameter σ_1 for $\|\boldsymbol{u} - \boldsymbol{u}'\|_2^2$ and σ for $\|\boldsymbol{d}_h - \boldsymbol{D}_h\boldsymbol{u}\|_2^2$ and $\|\boldsymbol{d}_v - \boldsymbol{D}_v\boldsymbol{u}\|_2^2$.

Following the procedure of IALM, we first update $\mathbf{d}_h$ and $\mathbf{d}_v$ by solving,

$$\boldsymbol{d}_h^{(k+1)} = \arg\min_{d_h} \boldsymbol{d}_h^T\boldsymbol{H}\boldsymbol{d}_h + \boldsymbol{\lambda}_2^{(k)T}(\boldsymbol{d}_h - \boldsymbol{D}_h\boldsymbol{u}^{(k)}) + \frac{\sigma^{(k)}}{2}\|\boldsymbol{d}_h - \boldsymbol{D}_h\boldsymbol{u}^{(k)}\|_2^2, \tag{29}$$

$$\boldsymbol{d}_v^{(k+1)} = \arg\min_{d_v} \boldsymbol{d}_v^T\boldsymbol{H}\boldsymbol{d}_v + \boldsymbol{\lambda}_3^{(k)T}(\boldsymbol{d}_v - \boldsymbol{D}_v\boldsymbol{u}^{(k)}) + \frac{\sigma^{(k)}}{2}\|\boldsymbol{d}_v - \boldsymbol{D}_v\boldsymbol{u}^{(k)}\|_2^2. \tag{30}$$

By making derivatives of Eqs. (29) and (30) be zero with respect to $\mathbf{d}_h$ and $\mathbf{d}_v$, respectively, $\boldsymbol{d}_h^{(k+1)}$ and $\boldsymbol{d}_v^{(k+1)}$ can be obtained by,

$$\boldsymbol{d}_h^{(k+1)} = (2\boldsymbol{H} + \sigma^{(k)}\boldsymbol{I})^{-1}(\sigma^{(k)}\boldsymbol{D}_h\boldsymbol{u}^{(k)} + \lambda_2^{(k)}), \tag{31}$$

$$\boldsymbol{d}_v^{(k+1)} = (2\boldsymbol{H} + \sigma^{(k)}\boldsymbol{I})^{-1}(\sigma^{(k)}\boldsymbol{D}_v\boldsymbol{u}^{(k)} + \lambda_3^{(k)}), \tag{32}$$

where $2\boldsymbol{H} + \sigma^{(k)}\boldsymbol{I}$ is a diagonal matrix, so the corresponding inverse matrix can be efficiently obtained.

Second, $\boldsymbol{u}'^{(k+1)}$ can be obtained as follows,

$$\boldsymbol{u}'^{(k+1)} = \arg\min_{\boldsymbol{u}'}(\boldsymbol{u}' - \boldsymbol{f}_h)^T\boldsymbol{M}(\boldsymbol{u}' - \boldsymbol{f}_h) + \boldsymbol{\lambda}_1^{(k)T}\boldsymbol{u}' + \frac{\sigma_1^{(k)}}{2}\|\boldsymbol{u}^{(k)} - \boldsymbol{u}'\|_2^2, \tag{33}$$

and the closed-form solution of $\boldsymbol{u}'^{(k+1)}$ is,

$$\boldsymbol{u}'^{(k+1)} = (2\boldsymbol{M} + \sigma_1^{(k)}\boldsymbol{I})^{-1}(2\boldsymbol{M}\boldsymbol{f}_h - \boldsymbol{\lambda}_1^{(k)} + \sigma_1^{(k)}\boldsymbol{u}^{(k)}), \tag{34}$$

where $2\boldsymbol{M} + \sigma_1^{(k)}\boldsymbol{I}$ is a diagonal matrix, so the corresponding inverse matrix can be efficiently obtained.

Finally, $\boldsymbol{u}^{(k+1)}$ can be updated by solving following subproblem

$$\begin{aligned} \boldsymbol{u}^{(k+1)} = \arg\min_{\boldsymbol{u}} {} & \boldsymbol{\lambda}_1^{(k)T}(\boldsymbol{u}'^{(k+1)} - \boldsymbol{u}) + \frac{\sigma_1^{(k)}}{2}\|\boldsymbol{u} - \boldsymbol{u}'^{(k+1)}\|_2^2 \\ & + \boldsymbol{\lambda}_2^{(k)T}(\boldsymbol{d}_h^{(k+1)} - \boldsymbol{D}_h\boldsymbol{u}) + \frac{\sigma^{(k)}}{2}\|\boldsymbol{d}_h^{(k+1)} - \boldsymbol{D}_h\boldsymbol{u}\|_2^2 \\ & + \boldsymbol{\lambda}_3^{(k)T}(\boldsymbol{d}_h^{(k+1)} - \boldsymbol{D}_v\boldsymbol{u}) + \frac{\sigma^{(k)}}{2}\|\boldsymbol{d}_v^{(k+1)} - \boldsymbol{D}_v\boldsymbol{u}\|_2^2, \end{aligned} \tag{35}$$

and the closed-form solution of $\boldsymbol{u}^{(k+1)}$ is,

$$\boldsymbol{u}^{(k+1)} = FFT^{-1}\{(FFT(\boldsymbol{u}_0))\oslash(\sigma^{(k)}FFT(\boldsymbol{D}_h^T\boldsymbol{D}_h + \boldsymbol{D}_v^T\boldsymbol{D}_v) + \sigma_1^{(k)}\boldsymbol{I})\}, \tag{36}$$

where

$$\boldsymbol{u}_0 = \sigma_1^{(k)}\boldsymbol{u}'^{(k+1)} - \boldsymbol{\lambda}_1^{(k)} - \boldsymbol{D}_h^T\boldsymbol{\lambda}_2^{(k)} + \sigma^{(k)}\boldsymbol{D}_h^T\boldsymbol{d}_h^{(k+1)} + \sigma^{(k)}\boldsymbol{D}_v^T\boldsymbol{d}_v^{(k+1)} - \boldsymbol{D}_v^T\boldsymbol{\lambda}_3^{(k)}, \tag{37}$$

and $\oslash$ is the entry-wise division, and FFT^{-1} denotes the inverse fast Fourier transform.

For updating σ_1, we have

$$\sigma_1^{(k+1)} = \begin{cases} \rho\sigma_1^{(k)}, & if\ \left\|\boldsymbol{u}_{k+1} - \boldsymbol{u}'_{k+1}\right\|_2^2/\|\boldsymbol{u}_{k+1}\|_2^2 < \varepsilon_1 \\ \sigma_1^{(k)}, & otherwise \end{cases} \tag{38}$$

For updating σ, we have

$$\sigma^{k+1} = \begin{cases} \rho\sigma^{(k)}, & \text{if } R_{\min} < \varepsilon_2 \\ \sigma^{(k)} & \text{otherwise} \end{cases}, \tag{39}$$

where $R_{\min} = \min(\|\boldsymbol{d}_h^{(k+1)} - \boldsymbol{D}_h\boldsymbol{u}^{(k+1)}\|_2^2/\|\boldsymbol{d}_h^{(k+1)}\|_2^2, \|\boldsymbol{d}_v^{(k+1)} - \boldsymbol{D}_v\boldsymbol{u}^{(k+1)}\|_2^2/\|\boldsymbol{d}_v^{(k+1)}\|_2^2)$.

In summary, we describe the IALM-based algorithm for GGVF computation in Algorithm 5.

Algorithm 5. IALM-GGVF

Input: $\mathbf{f}_h$, **H**, **M**
Output: **u**

1. Initialize $k = 0$, $\mathbf{u}^{(k)}$, $\mathbf{u}'^{(k)}$, $\boldsymbol{d}_h^k, \boldsymbol{d}_v^k, \boldsymbol{\lambda}_1^k, \boldsymbol{\lambda}_2^k, \boldsymbol{\lambda}_3^k, \sigma_1^k > 0, \sigma^{(k)} > 0, \rho > 1$
2. **while** not converged
3. $\boldsymbol{d}_h^{(k+1)} = (2\boldsymbol{H} + \sigma^{(k)}\boldsymbol{I})^{-1}(\sigma^{(k)}\boldsymbol{D}_h\boldsymbol{u}^{(k)} + \boldsymbol{\lambda}_2^{(k)})$
4. $\boldsymbol{d}_v^{(k+1)} = (2\boldsymbol{H} + \sigma^{(k)}\boldsymbol{I})^{-1}(\sigma^{(k)}\boldsymbol{D}_v\boldsymbol{u}^{(k)} + \boldsymbol{\lambda}_3^{(k)})$
5. $\boldsymbol{u}'^{(k+1)} = (2\boldsymbol{M} + \sigma_1^{(k)}\boldsymbol{I})^{-1}(2\boldsymbol{M}\boldsymbol{f}_h - \boldsymbol{\lambda}_1^{(k)} + \sigma_1^{(k)}\boldsymbol{u}^{(k)})$
6. $\boldsymbol{u}_0 = \sigma_1^{(k)}\boldsymbol{u}'^{(k+1)} - \boldsymbol{\lambda}_1^{(k)} - \boldsymbol{D}_h^T\boldsymbol{\lambda}_2^{(k)} + \sigma^{(k)}\boldsymbol{D}_h^T\boldsymbol{d}_h^{(k+1)} + \sigma^{(k)}\boldsymbol{D}_v^T\boldsymbol{d}_v^{(k+1)} - \boldsymbol{D}_v^T\boldsymbol{\lambda}_3^{(k)}$
7. $\boldsymbol{u}^{(k+1)} = FFT^{-1}\left\{(FFT(\boldsymbol{u}_0))\oslash\left(\sigma^{(k)}FFT(\boldsymbol{D}_h^T\boldsymbol{D}_h + \boldsymbol{D}_v^T\boldsymbol{D}_v) + \sigma_1^{(k)}\boldsymbol{I}\right)\right\}$
8. $\boldsymbol{\lambda}_1^{(k+1)} = \boldsymbol{\lambda}_1^{(k)} + \sigma_1^{(k)}(\boldsymbol{u}'^{(k+1)} - \boldsymbol{u}^{(k+1)})$
9. $\boldsymbol{\lambda}_2^{(k+1)} = \boldsymbol{\lambda}_2^{(k)} + \sigma^{(k)}(\boldsymbol{d}_h^{(k+1)} - \boldsymbol{D}_h\boldsymbol{u}^{(k+1)})$
10. $\boldsymbol{\lambda}_3^{(k+1)} = \boldsymbol{\lambda}_3^{(k)} + \sigma^{(k)}(\boldsymbol{d}_v^{(k+1)} - \boldsymbol{D}_v\boldsymbol{u}^{(k+1)})$
11. Update $\sigma_1^{(k)}$ *to* $\sigma_1^{(k+1)}$
12. Update $\sigma^{(k)}$ *to* $\sigma^{(k+1)}$
13. $k \leftarrow k + 1$
14. **end while**
15. $\mathbf{u} = \mathbf{u}^{(k)}$

3.4. Algorithm implementation and discussion

We use the similar initialization scheme for IALM-GVF and IALM-GGVF. Both $\boldsymbol{u}^{(0)}$ and $\boldsymbol{u}'^{(0)}$ are initialized to $\boldsymbol{f}_h$. $\boldsymbol{\lambda}^{(k)}$, $\boldsymbol{\lambda}_1^{(k)}$, $\boldsymbol{\lambda}_2^{(k)}$, and $\boldsymbol{\lambda}_3^{(k)}$ are all initialized to be zero. For IALM-GGVF, we choose $\boldsymbol{d}_h^{(0)} = \boldsymbol{D}_h\boldsymbol{u}^{(0)}$ and $\boldsymbol{d}_v^{(0)} = \boldsymbol{D}_v\boldsymbol{u}^{(0)}$.

Although the IALM-GVF algorithm converges for any $\mu_0 > 0$, $\rho > 0$, the values of μ_0 and ρ do affect the convergence speed of the algorithm seriously. In our work, for an $m \times m$ image, we empirically set $\sigma^{(0)} = 0.5/m$, $\varepsilon = 10^{-3}$ and $\rho = 3$ for IALM-GVF, and set $\sigma^{(0)} = 0.1/m$, $\sigma_1^{(0)} = \max(1, (m-60)/6)\sigma^{(0)}$, $\varepsilon_1 = \varepsilon_2 = 10^{-3}$ and $\rho = 3$ for IALM-GGVF.

Fast Fourier transform usually involves some assumptions on boundary conditions, e.g., periodic or reflective. To alleviate the adverse influence of boundary condition, we first make a larger zero image **g** with the size of $1.14m \times 1.14m$, and then put the edge map **f** at the centre of the image **g**. Then we use IALM-GVF or IALM-GGVF on the larger image **g** to compute the field $\mathbf{w}(x,y) = [u(x,y), v(x,y)]$ and crop the central part as the final result.

There are several possible choices of the stopping criteria of IALM-GVF and IAML-GGVF. One may check whether the difference in the objective function value or the iterative solution is below a sufficient small positive value. Thanks to the multiresolution scheme, one can also simply run a fixed number of iterations N (= 1 or 2) to stop the algorithm.

Finally, we discuss the potential advantages of the proposed IALM and MR-IALM methods. First, IALM is a convex optimization algorithm, and it is guaranteed to converge to the global optimal solution. Second, the proposed IALM and MR-IALM methods only involve several FFT and matrix operations which have been included in most image processing libraries, and thus it is easy to implement them in C/C++ or matlab. Besides, similar optimization problems and algorithms have been extensively investigated in image restoration (Afonso et al., 2010; Zuo and Lin, 2011), compressed sensing (Afonso et al., 2011), and robust principal component analysis (Wang et al., 2008; Ganesh et al., 2009). With the rapid progress in convex optimization algorithms and widely applications of ALM, we hope more efficient GVF/GGVF computation algorithms would be developed in this context.

4. Experimental results

In this section, we first evaluate the computational speed of the proposed IALM-GVF and IALM-GGVF methods[1], and compare the proposed methods with the original methods in (Xu and Prince, 1998b), the multiresolution method (MR-GVF/GGVF) in (Ntalianis et al., 2001), and the multigrid methods (MGVF and MGGVF) in (Han et al., 2007). Then, we verify the GVF fields computed by the proposed methods by applying them on two GVF-based image processing tasks: GVF snake and GVF-based anisotropic diffusion based on Perona–Malik equation. Note that the programs are all coded in C/C++ and ran on a 2.30 GHz Core(TM)i5–2410 M laptop PC with a Windows 7 operating system. The implementation of these methods are summarized as follows:

(a) The codes of the original GVF/GGVF and the multigrid GVF are downloaded from the webpage (http://www.iacl.ece.jhu.edu/static/gvf/). The codes of GVF/GGVF are implemented in matlab. For the sake of fairness, we have rewritten it in C/C++. The code of multigrid GVF is implemented in C/C++, but the website does not provide the MGGVF code. So we implement the multigrid GGVF code in C/C++ by modifying the definition of $h(f)$ and $g(f)$.
(b) Based on Ntalianis et al., 2001, we implement the multiresolution GVF/GGVF methods in C/C++. Finally, we implement the proposed IALM-GVF/GGVF and MR-IALM-GVF/GGVF methods in C/C++.

4.1. Evaluation on computational efficiency

In this section, we use a set of human lung CT images to evaluate the speed of original GVF/GGVF, MGVF/MGGVF, and IALM-GVF/GGVF methods. The sizes of images vary from 256×256 to 1024×1024. Fig. 1(a) shows one image from the set. For the sake of fairness, all the algorithms are implemented in C/C++ and test them using the same environment.

Table 1 lists the run time of different GVF algorithms on the test images with different sizes. One can see that the run time of all algorithms increases with the increasing of image size. Compared with the original GVF and GGVF, IALM-GVF and MR-IALM-GVF can significantly improve the computational speed by one or several orders of magnitude, and would be more efficient when the size of image increases. Moreover both IALM-GVF and MR-IALM-GVF can achieve faster speed than MR-GVF and MGVF.

Table 2 lists the run time of different GGVF algorithms on the test images with different sizes. Analogously, IALM-GGVF is more efficient than GGVF and MR-GGVF, and MR-IALM-GGVF can achieve much faster speed than all the other methods. So we can conclude that the efficiency of MR-IALM-GGVF should be contributed to both the IALM algorithm and the multiresolution approach.

[1] https://sites.google.com/site/cswmzuo/IALM-GVF_GGVF.rar.

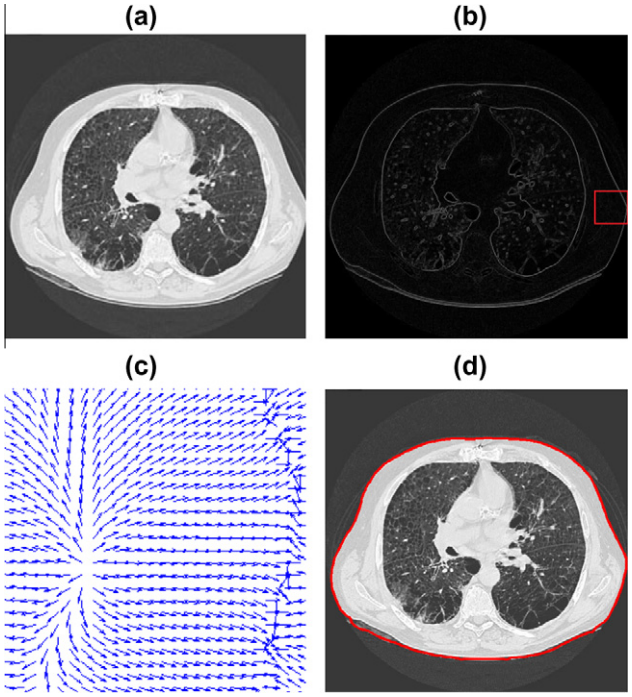

Fig. 1. Example of a 256 × 256 CT lung image: (a) original image, (b) edge map, (c) details of partial GGVF field, and (d) segmentation result.

Table 1
Run time (*s*) of different GVF computation algorithms.

Image size	256 × 256	512 × 512	1024 × 1024
GVF	0.919	14.389	194.354
MR-GVF	0.246	0.640	1.386
MGVF	0.095	0.437	1.731
IALM-GVF	0.039	0.172	0.785
MR-IALM-GVF	0.051	0.219	0.999

Table 2
Run time (*s*) of different GGVF computation algorithms.

Image size	256 × 256	512 × 512	1024 × 1024
GGVF	1.191	15.323	206.663
MR-GGVF	0.396	0.997	2.242
MGGVF	0.509	2.100	7.868
IALM-GGVF	0.102	0.507	3.008
MR-IALM-GGVF	0.078	0.390	1.732

In summary, by combining the advantages of IALM and multiresolution analysis, MR-IALM-GVF/GGVF can achieve much faster speed than the state-of-the-art multigrid GVF/GGVF methods.

4.2. GVF-based image processing

In this subsection, we apply the GVF field on two GVF-based image processing tasks to illustrate the effectiveness of the proposed methods, i.e. image segmentation using GVF snake and GVF-based anisotropic diffusion model.

4.2.1. GVF snake

An active model (or snake) Kass et al., 1987 is a curve $\boldsymbol{x}(s) = [x(s), y(s)]$, $s \in [0, 1]$, that moves through the spatial domain of an image to minimize the following energy functional,

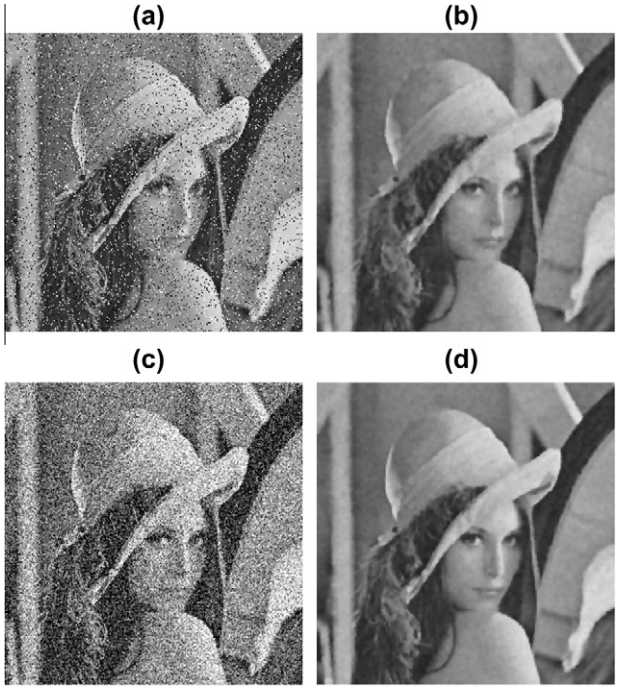

Fig. 2. Results of anisotropic diffusion model using Perona–Malik equation: (a) Lena image with salt and pepper noise, (b) denoising result of (a), (c) Lena image with Gaussian noise, and (d) denoising result of (c).

$$E = \int_0^1 \frac{1}{2}\left[\alpha|\boldsymbol{x}'(s)|^2 + \beta|\boldsymbol{x}''(s)|^2\right] + E_{ext}(\boldsymbol{x}(s))ds, \tag{40}$$

where α and β are weighting parameters that control the snake's tension and rigidity, respectively, and $\boldsymbol{x}'(s)$ and $\boldsymbol{x}''(s)$ denote the first and second derivatives of $\boldsymbol{x}(s)$ with respect to s. For traditional snake, E_{ext} is defined as $-|\nabla I(x,y)|^2$. But for GVF snake, we replace ∇E_{ext} with the GVF or GGVF field $\boldsymbol{w}(x,y)$.

We perform GVF snake based on the GGVF field computed by IALM-GGVF. Fig. 1(c) shows part of the GGVF field and Fig. 1(d) shows the segmentation result on a 256 × 256 CT lung image. One can see that MR-IALM-GGVF can obtain satisfactory GVF field to make GVF snake converge to correct image boundary. Besides, we note that one can incorporate MR-IALM-GGVF scheme with multiresolution GVF snake. In this way, the segmentation of Fig. 1(a) can be finished within 0.096 s.

4.2.2. GVF-based anisotropic diffusion model

We implemented a GVF-based anisotropic diffusion model based on Perona–Malik equation (Yu and Chua, 2006), where the GVF field is computed in advance using the proposed MR-IALM-GVF method. The GVF-based Perona–Malik equation is written as

$$I_t = -\boldsymbol{w} \cdot \nabla I + g\Delta I, \tag{41}$$

where $\mathbf{w}$ is the GVF field, $g(|\nabla I|) = (1/\sqrt{2\pi}\sigma_E)\exp(-|\nabla I|^2/2\sigma_E^2)$. We use a 512 × 512 Lena image to evaluate the denoising performance of the GVF-based Perona–Malik equation. Two kinds of noise was added to the Lena image: 10% percent of salt and pepper noise (as shown in Fig. 2(a)) and Gaussian noise with mean of zero and variance of 0.04 (as shown in Fig. 2(c)). As shown in Fig. 2(b) and (d), satisfactory results can be obtained by GVF-based anisotropic diffusion model with the GVF field obtained by MR-IALM-GVF.

5. Conclusion

In this paper, by reformulating GVF or GGVF computation as constrained convex optimization problems and using the variable

splitting and augmented Lagrangian methods, we proposed two algorithms, IALM-GVF and IALM-GGVF, for fast GVF and GGVF computation, respectively. The proposed methods can be further incorporated in the multiresolution scheme for efficient GVF/GGVF field computation (MR-IALM-GVF/GGVF). Compared with the original GVF and GGVF methods, the proposed methods can improve the computational speed by one or two order of magnitude, and are even more efficient for images with large size. Moreover, MR-IALM-GVF/GGVF are also much faster than multigrid GVF/GGVF in terms of computational efficiency, and are easy to be implemented. Finally, we implemented GVF snake for image segmentation and GVF-based anisotropic diffusion model for image denoising to confirm the validity of our new methods.

Besides, this paper suggested a viewpoint of convex optimization for fast GVF or GGVF computation. Recently, convex optimization algorithms have received increasing interests and been widely applied in many applications, e.g., image restoration, and compressed sensing, and matrix completion, In the future, we will investigate more efficient scheme for GVF/GGVF computation by referring to progress in related optimization algorithms.

Acknowledgements

The work is partially supported by the GRF fund from the HKSAR Government, the central fund from the Hong Kong Polytechnic University, the NSFC funds of China (Grant Nos: 60902099, 61071179, 61272341, 61231002, and 61001037), the Hong Kong Scholar Program, and the Fundamental Research Funds for the Central Universities (Grant No. HIT.NSRIF.2010051).

References

Afonso, M.V., Bioucas-Dias, J.M., Figueiredo, M.A.T., 2010. Fast image recovery using variable splitting and constrained optimization. IEEE Trans. Image Process. 19 (9), 2345–2356.

Afonso, M.V., Bioucas-Dias, J.M., Figueiredo, M.A.T., 2011. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. IEEE Trans. Image Process. 20 (3), 681–695.

Bertsekas, D.P., 1996. Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific.

Boukerroui, D., 2009. Efficient numerical schemes for gradient vector flow. In: 2009 Sixteenth IEEE Internat. Conf. on Image Process. (ICIP 2009), pp. 4057–4060.

Ganesh, A., Lin, Z., Wright, J., Wu, L., Chen, M., Ma, Y., 2009. Fast algorithms for recovering a corrupted low-rank matrix. In: Internat. Workshop on Computational Advances in Multi-Sensor Adaptive Process.

Ghita, O., Whelan, P.F., 2010. A new GVF-based image enhancement formulation for use in the presence of mixed noise. Pattern Recognition 43 (8), 2646–2658.

Han, X., Xu, C., Prince, J.L., 2007. Fast numerical scheme for gradient vector flow computation using a multigrid method. IET Image Process. 1 (1), 48–55.

Hassouna, M.S., Farag, A.A., 2009. Variational curve skeletons using gradient vector flow. IEEE Trans. Pattern Anal. Machine Intell. 31 (12), 2257–2274.

He, L., Li, C., Xu, C., 2008. Intensity statistics-based HSI diffusion for color photo denoising. In: IEEE Internat. Conf. on Computer Vision and Pattern Recognition, Anchorage, AK.

Kass, M., Witkin, A., Terzopoulos, D., 1987. Snakes: active contour models. Internat. J. Comput. Vision 1 (4), 321–331.

Li, B., Acton, S.T., 2007. Active contour external force using vector field convolution for image segmentation. IEEE Trans. Image Process. 16 (8), 2096–2106.

Li, J., Zuo, W., Zhao, X., Zhang, D., 2011. An augmented Lagrangian method for fast gradient vector flow computation. In: Eighteenth IEEE Internat. Conf. on Image Process. (ICIP 2011), pp. 1557–1560.

Lin, Z., Chen, M., Ma, Y., 2009. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical Report UILU-ENG-09-2215, UIUC. Available from: <arxiv:1009.5055>.

Lin, Z., Liu, R., Su, Z., 2011. Linearized alternating direction method with adaptive penalty for low rank representation. Neural Info. Process. Systems, NIPS 2011.

Liu, G., Lin, Z., Yu, Y., 2010. Robust subspace segmentation by low-rank representation. In: Internat. Conf. on Machine Learning, Haifa, Israel.

Ntalianis, K., Doulamis, N., Doulamis, A., Kollias, S., 2001. Multiresolution gradient vector flow field: a fast implementation towards video object plane segmentation. In: IEEE Internat. Conf. on Multimedia and Expo (ICME'01).

Paragios, N., Mellina-Gottardo, O., Ramesh, V., 2004. Gradient vector flow fast geometric active contours. IEEE Trans. Pattern Anal. Machine Intell. 26 (3), 402–407.

Ray, N., Acton, S.T., 2004. Motion gradient vector flow: an external force for tracking rolling leukocytes with shape and size constrained active contours. IEEE Trans. Med. Imaging 23 (12), 1466–1478.

Trottenberg, U., OOsterlee, C., Schüller, A., 2001. Multigrid. Academic Press.

Wang, Y., Yang, J., Yin, W., Zhang, Y., 2008. A new alternating minimization algorithm for total variation image reconstruction. SIAM J. Image Sci. 1, 248–272.

Xu, C., Prince, J.L., 1998a. Snakes, shapes, and gradient vector flow. IEEE Trans. Image Process. 7 (3), 359–369.

Xu, C., Prince, J.L., 1998b. Generalized gradient vector flow external forces for active contours. Signal Process. 71 (2), 131–139.

Yu, H., Chua, C.-S., 2006. GVF-based anisotropic diffusion models. IEEE Trans. Image Process. 15 (6), 1517–1524.

Zuo, W., Lin, Z., 2011. A generalized accelerated proximal gradient approach for total-variation-based image restoration. IEEE Trans. Image Process. 20 (10), 2748–2759.