Numer. Math. Theor. Meth. Appl.
doi: 10.4208/nmtma.2012.m1047

Vol. **5**, No. 2, pp. 260-277
May 2012

A Primal-Dual Hybrid Gradient Algorithm to Solve the LLT Model for Image Denoising

Chunxiao Liu[1,*], Dexing Kong[2] and Shengfeng Zhu[2]

[1] *Department of Mathematics, Hangzhou Normal University, Hangzhou 310036, China.*
[2] *Department of Mathematics, Zhejiang University, Hangzhou, China.*

Received 22 December 2010; Accepted (in revised version) 3 May 2011

Available online 27 March 2012

Abstract. We propose an efficient gradient-type algorithm to solve the fourth-order LLT denoising model for both gray-scale and vector-valued images. Based on the primal-dual formulation of the original nondifferentiable model, the new algorithm updates the primal and dual variables alternately using the gradient descent/ascent flows. Numerical examples are provided to demonstrate the superiority of our algorithm.

AMS subject classifications: 68U10, 65K10

Key words: LLT model, image denoising, primal-dual.

1. Introduction

Image denoising is one of the most fundamental tasks in image processing. For generality, let us consider a vectorial function $\mathbf{u}(x)$ defined on a bounded domain $\Omega \subset \mathbb{R}^2$:

$$\begin{aligned} &\mathbf{u}: \ \Omega \to \mathbb{R}^d \\ &x = (x_1, x_2) \mapsto \mathbf{u}(x) := (u_1(x), u_2(x), \cdots, u_d(x)). \end{aligned} \tag{1.1}$$

The observed noisy image $\mathbf{u}_0(x)$ is modeled as

$$\mathbf{u}_0 = \mathbf{u} + \mathbf{n}, \tag{1.2}$$

where $\mathbf{n}$ is a d-dimensional additive Gaussian noise with zero-mean. When $d = 1$ and 3, we get the model for gray-scale and color images, respectively. The task of image denoising is to recover the true image $\mathbf{u}$ from the given noisy data $\mathbf{u}_0$, which belongs to the inverse problems. The well-known ill-posedness of inverse problems is often handled by some regularization methods. Tikhonov type regularization is very popular and useful for many

*Corresponding author. *Email addresses:* xxliu198431@126.com (C. Liu), kong@cms.zju.edu.cn (D. Kong), shengfengzhu@zju.edu.cn (S. Zhu)

types of inverse problems. In the field of image denoising, one of the most famous and influential approaches is the ROF model, which is proposed by Rudin, Osher, and Fatemi in [15] originally for gray-scale image denoising. The ROF model aims to minimize the following energy functional:

$$\int_\Omega |\nabla u(x)| dx + \frac{\beta}{2} \int_\Omega |u - u_0|^2 dx, \tag{1.3}$$

where the first term is the total variation (TV) term and $\beta > 0$ is a parameter that controls the contribution of the fidelity term [5]. The novelty of the ROF model is that it allows discontinuous solutions and hence can preserve sharp edges while removing noise. The vectorial TV can be defined similarly and has been applied to some color image processing problems [1,2].

However, an undesired disadvantage of the TV based model is the "staircase" effects [12,13], as can be seen from images in [15,18], which is the result of piecewise constant solutions. A variety of higher order variational models (especially fourth-order) are introduced to alleviate the staircase effects [6,8,11,13,16,21]. In [13], a new fourth-order filter based variational model, which is referred as the LLT model, has been proposed. To better preserve edges and avoid the oversmoothing of the fourth order models simultaneously, some mixed models have been proposed [12,14]. Lysaker and Tai [14] devise an iterative scheme by a convex combination of the results of the second and the fourth order models. A variational approach combining a total variational filter and a fourth order filter is proposed by Li et al. [12].

In numerical implementation, both the TV-based and the LLT model suffer from the nondifferentiability. A direct method is to replace the original nondifferentiable L^1-norm by a modified version through introducing a small regularization parameter. However, there is a trade-off between the accuracy versus the speed of convergence. A variety of efficient algorithms are developed to deal with this problem. One class is based on variable splitting and constrained optimization [3,17]. In [17], a FFT based algorithm is proposed by virtue of variable splitting and the penalty method. The split Bregman algorithm [10] and the augmented Lagrangian method [19,20], which were demonstrated to be equivalent with each other, are also efficient algorithms. Another class uses the primal-dual formulation of the original problem. Many methods of this class focus either on the primal variable or on the dual variable [4,7]. Recently, Zhu and Chan [22] presented an efficient primal-dual hybrid gradient (PDHG) algorithm, which alternates between the primal and dual variable by gradient-type methods. This algorithm is both simple and fast. Furthermore, it can be applied to a large range of restoration problems. Motivated by [22], we propose an effective and fast primal-dual based gradient algorithm to solve the fourth-order LLT model.

The following part of the paper is organized as follows. In Section 2, we recall first the LLT model in a vectorial form and then review briefly the existing methods for solving the LLT model. In the next section, we present the PDHG algorithm for solving the LLT model. In Section 4, some numerical implementation details are given. Numerical examples and comparisons are shown in Section 5. Finally, we conclude the paper in Section 6.

2. LLT model and related works

We first recall the LLT model for gray-scale and vector-valued images. Then we briefly review three existing algorithms for solving this model.

2.1. LLT model

The LLT model was proposed in [13] for gray-scale image to alleviate the staircase effect introduced by the ROF model. The LLT model requires to solve the following minimization problem with respect to a scalar image u

$$\min_u \left\{ \int_\Omega |\nabla^2 u| dx + \frac{\beta}{2} \int_\Omega |u-u_0|^2 dx \right\}, \tag{2.1}$$

where

$$\nabla^2 u = \begin{pmatrix} \dfrac{\partial^2 u}{\partial x_1^2} & \dfrac{\partial^2 u}{\partial x_1 \partial x_2} \\ \dfrac{\partial^2 u}{\partial x_2 \partial x_1} & \dfrac{\partial^2 u}{\partial x_2^2} \end{pmatrix} \tag{2.2}$$

is the Hessian of u and

$$|\nabla^2 u| = \sqrt{\left(\frac{\partial^2 u}{\partial x_1^2}\right)^2 + \left(\frac{\partial^2 u}{\partial x_1 \partial x_2}\right)^2 + \left(\frac{\partial^2 u}{\partial x_2 \partial x_1}\right)^2 + \left(\frac{\partial^2 u}{\partial x_2^2}\right)^2}. \tag{2.3}$$

To solve this nondifferentiable problem, $|\nabla^2 u|$ is often replaced by $\sqrt{|\nabla^2 u|^2 + \varepsilon}$ in numerical implementation. Similar to the solution of the ROF model, a proper choice of ε is a difficult task. Moreover, the fourth-order filter imposes more rigid restrictions on the time step size of the discretized gradient flow, which leads to a slow convergence of the iterative process. Therefore, the dual method was employed to solve the fourth-order model [9] to improve accuracy and speed up convergence.

Let us introduce some notations in the following. The TV^2 norm of u is defined in a more formalized form as

$$TV^2(u) = \int_\Omega |\nabla^2 u| dx = \sup_{p \in X} \int_\Omega \sum_{i,j=1}^{2} u \partial_i \partial_j p^{ij} dx,$$

where

$$X = \left\{ p = \begin{pmatrix} p^{11} & p^{12} \\ p^{21} & p^{22} \end{pmatrix} \in C_c^2(\Omega, \mathbb{R}^{2\times 2}), |p(x)| \leqslant 1 \text{ for all } x \in \Omega \right\} \tag{2.4}$$

with

$$|p(x)| = \sqrt{\sum_{i,j=1}^{2} (p^{ij})^2}.$$

For convenience, we denote the operator div^2 as

$$\mathrm{div}^2 p(x) = \sum_{i,j=1}^{2} \partial_i \partial_j p^{ij}.$$

For a vector-valued image $\mathbf{u} = (u_1, u_2, \cdots, u_d)$, the vectorial LLT model can be proposed similarly. Let $\langle \cdot, \cdot \rangle$ stand for the Euclidean scalar product defined by

$$\langle \mathbf{u}, \mathbf{v} \rangle := \sum_{i=1}^{d} \langle u_i, v_i \rangle, \qquad \forall\ \mathbf{v} = (v_1, v_2, \cdots, v_d) :\ \Omega \to \mathbb{R}^d.$$

Notice

$$\nabla^2 \mathbf{u} = (\nabla^2 u_1, \nabla^2 u_2, \cdots, \nabla^2 u_d).$$

Then

$$TV^2(\mathbf{u}) = \int_\Omega |\nabla^2 \mathbf{u}| dx = \sup_{\mathbf{p} \in X^d} \int_\Omega \langle \mathbf{u}, \mathrm{div}^2 \mathbf{p} \rangle dx,$$

where

$$\mathrm{div}^2\ \mathbf{p} = (\mathrm{div}^2 p_1, \mathrm{div}^2 p_2, \cdots, \mathrm{div}^2 p_d), \quad p_i = \begin{pmatrix} p_i^{11} & p_i^{12} \\ p_i^{21} & p_i^{22} \end{pmatrix}, \qquad \text{for } i \in \{1, 2 \cdots, d\},$$

$$X^d = \left\{ \mathbf{p} = \left(\begin{pmatrix} p_1^{11} & p_1^{12} \\ p_1^{21} & p_1^{22} \end{pmatrix}, \begin{pmatrix} p_2^{11} & p_2^{12} \\ p_2^{21} & p_2^{22} \end{pmatrix}, \cdots, \begin{pmatrix} p_d^{11} & p_d^{12} \\ p_d^{21} & p_d^{22} \end{pmatrix} \right), \ \ |\mathbf{p}(x)| \leqslant 1,\ \forall x \in \Omega \right\}$$

with

$$|\mathbf{p}(x)| = \sqrt{\sum_{i=1}^{d} \left((p_i^{11})^2 + (p_i^{12})^2 + (p_i^{21})^2 + (p_i^{22})^2 \right)}.$$

The d-dimensional vectorial LLT model reads

$$\min_{\mathbf{u}} \left\{ \int_\Omega |\nabla^2 \mathbf{u}| dx + \frac{\beta}{2} \int_\Omega |\mathbf{u} - \mathbf{u}_0|^2 dx \right\}, \tag{2.5}$$

including (2.1) as a special case $d = 1$.

2.2. Existing solvers

In this section, we briefly review three existing algorithms to solve the LLT model.

2.2.1. Gradient descent method

In [12, 13], the finite difference scheme was used to numerically discretize the LLT model (2.1). Regularization of the gradient flow was inevitable in their method even though a special difference scheme was employed. The convergence speed is slow and the accuracy is limited as demonstrated in [9].

2.2.2. Dual method for the LLT model

Motivated by Chambolle's dual algorithm for the ROF model [4], Chen et al. [9] proposed a dual method to solve the LLT model for gray-scale images. The general vectorial form can be given similarly. The solution of (2.5) is given by

$$\mathbf{u} = \mathbf{u}_0 - \frac{1}{\beta}\mathrm{div}^2\mathbf{p}, \tag{2.6}$$

where $\mathbf{p}$ is the converged value of the iterative sequence $\{\mathbf{p}^k\}$ generated by

$$\mathbf{p}^{k+1} = \frac{\mathbf{p}^k + \delta\nabla^2(\mathrm{div}^2\mathbf{p}^k - \beta\mathbf{u}_0)}{1 + \delta|\nabla^2(\mathrm{div}^2\mathbf{p}^k - \beta\mathbf{u}_0)|}, \qquad k = 0, 1, \cdots, \tag{2.7a}$$

$$\mathbf{p}^0 = 0, \tag{2.7b}$$

where the time step $\delta > 0$ should satisfy $\delta < 1/64$ to guarantee the convergence of the iterative scheme (2.7).

2.2.3. Augmented Lagrangian method

Recently, the higher order models are solved using the variable splitting technique and the augmented Lagrangian method in [20]. They formulated the following constrained problem

$$\min_{\mathbf{u},\mathbf{p}} \left\{ \int_\Omega |\mathbf{p}| dx + \frac{\beta}{2}\int_\Omega |\mathbf{u} - \mathbf{u}_0|^2 dx \right\}$$
$$\text{subject to} \quad \mathbf{p} = \nabla^2\mathbf{u}.$$

The augmented Lagrangian method is employed to transform the above constrained problem into an unconstrained one:

$$\min_{\mathbf{u},\mathbf{p}} \left\{ \int_\Omega |\mathbf{p}| dx + \frac{\beta}{2}\int_\Omega |\mathbf{u} - \mathbf{u}_0|^2 dx + \int_\Omega \lambda(\mathbf{p} - \nabla^2\mathbf{u}) dx + \frac{\mu}{2}\int_\Omega |\mathbf{p} - \nabla^2\mathbf{u}| dx \right\},$$

where λ is the Lagrangian multiplier and $\mu > 0$ is the penalization parameter. Let $\mathscr{F}$ and $\mathscr{F}^{-1}$ denote the Fourier transform and inverse transform, respectively. Then minimizing $\mathbf{u}$ and $\mathbf{p}$ alternately gives the following algorithm [20].

Algorithm 1: Augmented Lagrangian method for the LLT model

- Step 1. Initialization: Set the Lagrangian multiplier $\lambda^0 = 0$, $\mathbf{u}^0$=0 and choose properly the penalization parameter μ.
- Step 2. Update alternately

$$\mathbf{u}^{k+1} = \mathscr{F}^{-1}\left(\frac{\beta\mathscr{F}(\mathbf{u}_0) + \mu\mathscr{F}(\mathrm{div}^2)\mathscr{F}(\mathbf{p}^k) + \mathscr{F}(\mathrm{div}^2)\mathscr{F}(\lambda^k)}{\beta + \mu\mathscr{F}(\mathrm{div}^2)\mathscr{F}(\nabla^2)} \right)$$

$$\mathbf{p}^{k+1} = \begin{cases} \left(|\mathbf{w}^k| - \dfrac{1}{\mu}\right)\dfrac{\mathbf{w}^k}{|\mathbf{w}^k|}, & |\mathbf{w}^k| > \dfrac{1}{\mu}, \\ 0, & |\mathbf{w}^k| \leqslant \dfrac{1}{\mu} \end{cases}$$

with

$$\mathbf{w}^k = \nabla^2 \mathbf{u}^{k+1} - \frac{\lambda^k}{\mu}.$$

- Step 3. Update the Lagrangian multiplier by

$$\lambda^{k+1} = \lambda^k + \mu(\mathbf{p}^{k+1} - \nabla^2 \mathbf{u}^{k+1}).$$

- Step 4. Check convergence. Iterate again if necessary; $k = k + 1$.

3. PDHG algorithm for LLT model

The existing approaches focus either on the primal or on the dual variable, while the PDHG algorithm alternates between the two variables and makes use of the information from each other. Therefore, the PDHG algorithm converges much faster. We try to efficiently solve the fourth-order nondifferentiable LLT model by a hybrid gradient algorithm.

We will present our primal-dual algorithm based on the vectorial LLT model (2.5). The LLT model for gray images can be solved using our algorithm as a special case $d = 1$. The primal-dual formulation of (2.5) can be written as

$$\min_{\mathbf{u}} \max_{\mathbf{p}\in X^d} \Phi(\mathbf{u}, \mathbf{p}), \tag{3.1}$$

where

$$\begin{aligned} \Phi(\mathbf{u}, \mathbf{p}) &= \int_\Omega \langle \mathbf{u},\ \mathrm{div}^2 \mathbf{p}\rangle dx + \frac{\beta}{2}\int_\Omega |\mathbf{u} - \mathbf{u}_0|^2 dx \\ &= \int_\Omega \langle \nabla^2 \mathbf{u},\ \mathbf{p}\rangle dx + \frac{\beta}{2}\int_\Omega |\mathbf{u} - \mathbf{u}_0|^2 dx. \end{aligned} \tag{3.2}$$

One can switch the minimization and the maximization to use equivalent formulations.

We use an alternating iterative algorithm to solve (3.1). From the initial guesses $\mathbf{u}^0$ and $\mathbf{p}^0$, we get $\mathbf{u}^{k+1}$ and $\mathbf{p}^{k+1}$ alternately from $\mathbf{u}^k$ and $\mathbf{p}^k$.

Firstly, for fixed $\mathbf{u}^k$, we update $\mathbf{p}$ by solving the following maximization problem with respect to $\mathbf{p}$:

$$\max_{\mathbf{p}\in X^d} \Phi(\mathbf{u}^k, \mathbf{p}). \tag{3.3}$$

The ascent direction of $\mathbf{p}$ is given by

$$\frac{\partial \Phi}{\partial \mathbf{p}}(\mathbf{u}^k, \mathbf{p}) = \nabla^2 \mathbf{u}^k.$$

The projected gradient ascent method for problem (3.3) reads

$$\mathbf{p}^{k+1} = P_{X^d}(\mathbf{p}^k + \tau^k \nabla^2 \mathbf{u}^k),$$

where $\tau^k > 0$ is the time step and the projection operator P_{X^d} is defined as

$$P_{X^d}(\mathbf{z}) = \arg\min_{\mathbf{y}\in X^d} \|\mathbf{z} - \mathbf{y}\|.$$

In our case, the projection operator P_{X^d} can be simply computed in a straightforward way as

$$P_{X^d}(\mathbf{x}) = \left(\frac{x_1}{\max\{|x_1|,\ 1\}},\ \frac{x_2}{\max\{|x_2|,\ 1\}}, \cdots,\ \frac{x_d}{\max\{|x_d|,\ 1\}}\right),$$

for $\mathbf{x} = (x_1, x_2, \cdots, x_d)$.

Secondly, for fixed $\mathbf{p}^{k+1}$, we solve the following minimization problem with respect to $\mathbf{u}$ to update $\mathbf{u}$:

$$\min_{\mathbf{u}}\ \Phi(\mathbf{u}, \mathbf{p}^{k+1}). \tag{3.4}$$

The gradient descent direction for (3.4) is

$$-\frac{\partial \Phi}{\partial \mathbf{u}}(\mathbf{u}, \mathbf{p}^{k+1}) = -\Big(\mathrm{div}^2 \mathbf{p}^{k+1} + \beta(\mathbf{u} - \mathbf{u}_0)\Big). \tag{3.5}$$

Scaling the descent direction by $1/\beta$ and employing the gradient descent flow by the explicit Euler scheme with a time step $\theta^k > 0$, we can therefore update $\mathbf{u}$ by

$$\frac{\mathbf{u}^{k+1} - \mathbf{u}^k}{\theta^k} = -\left(\frac{1}{\beta}\mathrm{div}^2 \mathbf{p}^{k+1} + \mathbf{u}^k - \mathbf{u}_0\right) \tag{3.6}$$

or equivalently,

$$\mathbf{u}^{k+1} = (1 - \theta^k)\mathbf{u}^k + \theta^k\left(\mathbf{u}_0 - \frac{1}{\beta}\big(\mathrm{div}^2 \mathbf{p}^{k+1}\big)\right). \tag{3.7}$$

Unifying all the schemes together, we obtain the following algorithm.

Algorithm 2: PDHG algorithm for the LLT model

Step 1. Initialization: $\mathbf{u}^0 = \mathbf{u}_0$, $\mathbf{p}^0 = 1$, $k = 0$.

Step 2. Choose proper time steps τ^k and θ^k.

Step 3. Update alternatively

$$\mathbf{p}^{k+1} = P_{X^d}(\mathbf{p}^k + \tau^k \nabla^2 \mathbf{u}^k)$$
$$\mathbf{u}^{k+1} = (1 - \theta^k)\mathbf{u}^k + \theta^k\Big(\mathbf{u}_0 - \frac{1}{\beta}\big(\mathrm{div}^2 \mathbf{p}^{k+1}\big)\Big).$$

Step 4. Check convergence. Iterate again if necessary; $k = k + 1$.

Remark 3.1. The updating of $\mathbf{u}$ in (3.7) can actually be seen as a relaxation method making use of the information from the former step. From (3.5), we can derive the explicit expression $\mathbf{u}^{k+1} = \mathbf{u}_0 + \frac{1}{\beta}\mathrm{div}^2\mathbf{p}^{k+1}$. Then, (3.7) is the convex combination of the obtained result and $\mathbf{u}^k$, which is a relaxation strategy.

4. Implementation issues

In numerical implementation, we represent a gray-scale image as an $M \times N$ matrix. We first give the discretizations of the derivatives for the gray-scale image and that for the vector-valued image can be defined channel by channel. We define $D^+_{x_1}$, $D^+_{x_2}$ (resp. $D^-_{x_1}$, $D^-_{x_2}$) as the forward (resp. backward) difference operators with the periodic boundary condition. Denote

$$(D^+_{x_1}u)_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j}, & 1 \leqslant i \leqslant M-1, \\ u_{1,j} - u_{M,j}, & i = M, \end{cases}$$
$$(D^+_{x_2}u)_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j}, & 1 \leqslant j \leqslant N-1, \\ u_{i,1} - u_{i,N}, & j = N. \end{cases}$$

Then $D^-_{x_1}$, $D^-_{x_2}$ can be defined similarly. For the operators ∇^2 and div^2, we have

$$(\nabla^2 u)_{i,j} = \begin{pmatrix} (D^{-+}_{x_1x_1}u)_{i,j} & (D^{++}_{x_1x_2}u)_{i,j} \\ (D^{++}_{x_2x_1}u)_{i,j} & (D^{-+}_{x_2x_2}u)_{i,j} \end{pmatrix}$$

and

$$(\mathrm{div}^2 p)_{i,j} = (D^{+-}_{x_1x_1}p^{11})_{i,j} + (D^{--}_{x_2x_1}p^{12})_{i,j} + (D^{--}_{x_1x_2}p^{21})_{i,j} + (D^{+-}_{x_2x_2}p^{22})_{i,j},$$
$$p = \begin{pmatrix} p^{11} & p^{12} \\ p^{21} & p^{22} \end{pmatrix},$$

where

$$(D^{\pm\mp}_{x_1x_1}u)_{i,j} := (D^{\pm}_{x_1}(D^{\mp}_{x_1}u))_{i,j},$$
$$(D^{\pm\mp}_{x_2x_2}u)_{i,j} := (D^{\pm}_{x_2}(D^{\mp}_{x_2}u))_{i,j},$$
$$(D^{\pm\pm}_{x_1x_2}u)_{i,j} := (D^{\pm}_{x_1}(D^{\pm}_{x_2}u))_{i,j},$$
$$(D^{\pm\pm}_{x_2x_1}u)_{i,j} := (D^{\pm}_{x_2}(D^{\pm}_{x_1}u))_{i,j}.$$

Then for a vector-valued image $\mathbf{u} = (u_1, u_2, \cdots, u_d)$ and $\mathbf{p} = (p_1, p_2, \cdots, p_d) \in X^d$, we obtain

$$(\nabla^2\mathbf{u})_{i,j} = \Big((\nabla^2 u_1)_{i,j}, (\nabla^2 u_2)_{i,j}, \cdots, (\nabla^2 u_d)_{i,j}\Big),$$
$$(\mathrm{div}^2\mathbf{p})_{i,j} = \Big((\mathrm{div}^2 p_1)_{i,j}, (\mathrm{div}^2 p_2)_{i,j}, \cdots, (\mathrm{div}^2 p_d)_{i,j}\Big).$$

All these operators can be computed simply and fast using matrix difference in MATLAB.

We terminate the iteration if the following stopping condition is satisfied for some prescribed tolerance TOL> 0

$$\frac{\|\mathbf{u}^{k+1}-\mathbf{u}^k\|_2}{\|\mathbf{u}^k\|_2} < \text{TOL}. \tag{4.1}$$

The MATLAB function `imnoise` is used to add Gaussian noise with zero-mean and deviation σ to various images. For different values of TOL, we compute the Peak Signal-to-Noise Ratio (PSNR) of the restored image to quantify the performance of the algorithms. If the true image is $\mathbf{u}$ and $\mathbf{u}_0$ is its noisy perturbation, the PSNR of $\mathbf{u}$ and $\mathbf{u}_0$ is defined by

$$\text{PSNR}(\mathbf{u},\mathbf{u}_0) = 20\log_{10}\left(\frac{C}{E}\right), \tag{4.2}$$

where C is the maximum possible pixel value of the image and E is the mean squared error given by $\|\mathbf{u}-\mathbf{u}_0\|_2/(MN)$. In our experiments, all image intensities have been scaled between 0 and 255. We therefore have $C = 255$.

The choice of suitable time steps is critical for the convergence behavior and speed of the algorithm. We can use fixed time steps or similar adaptive time steps as in [22]. We use the following strategy to determine the time steps:

$$\tau^k = \frac{(0.1+0.05*k)}{4} \quad \text{and} \quad \theta^k = \frac{\left(0.2-\frac{1}{10+k}\right)}{(4\tau^k)}. \tag{4.3}$$

5. Numerical experiments

In this section, we perform some numerical experiments and make comparisons between three different algorithms introduced in Section 2.2 for both gray and color image denoising. We test our algorithm by some well-known images including Lena, Man and Rose. All numerical experiments are implemented within the software MATLAB on a PC with a 2.33GHz CPU and 2.0GB memory. The CPU time (in seconds) is reported. The dual algorithm for the LLT model has been demonstrated to be more efficient than the gradient descent method [9]. Here we focus on comparing among the primal-dual hybrid gradient algorithm for the LLT model, the dual algorithm for the LLT model and the augmented Lagrangian method for the LLT model, which we refer respectively as PDHG-LLT, D-LLT and AL-LLT in the following.

5.1. Gray-scale image: $d = 1$

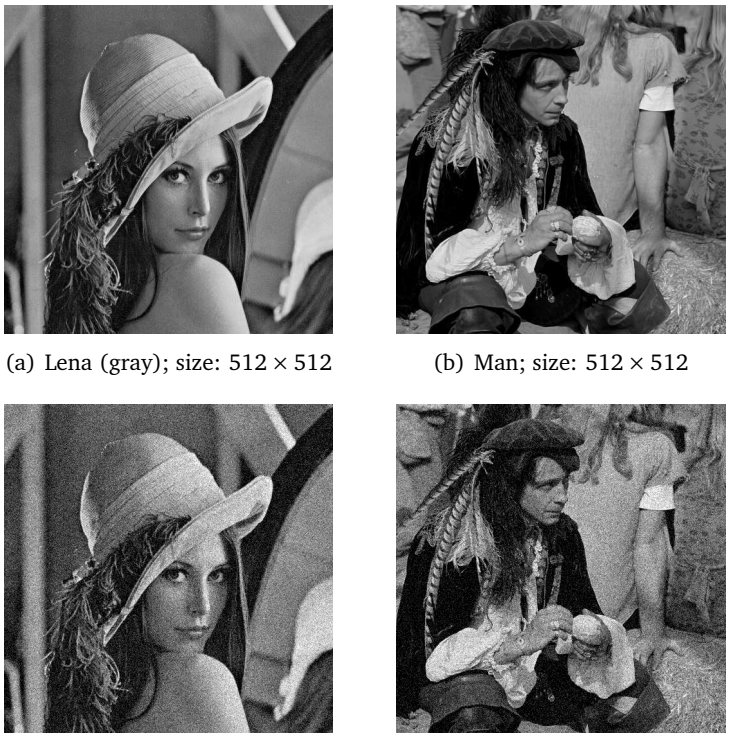
We use the gray images shown in Fig. 1 to test our algorithm corresponding to $d = 1$.

First, the noisy Lena image Fig. 1(c) is restored by different methods for different stopping tolerance values TOL=10^{-4} and 10^{-6} to evaluate the performance of our algorithm. When TOL=10^{-4}, we can observe from Fig. 2(a), Fig. 2(b) and Fig. 2(c)that the PDHG-LLT algorithm and the AL-LLT algorithm give more visually satisfactory solution than the D-LLT algorithm. In order to better illustrate this difference, we provide the zoomed-in

figures in Fig. 2(d), Fig. 2(e) and Fig. 2(f) by extracting a small portion from the restored images. From the local regions, we can clearly see that the PDHG-LLT algorithm and the AL-LLT algorithm obviously perform better than the D-LLT algorithm for the same tolerance. There are small differences on the PSNR values of the three algorithms as reported in Table 1. More accurate restoration results with TOL=10^{-6} are displayed in the bottom row of Fig. 2.

A qualitative comparison is given in Fig. 3 by potting the convergence history of the relative L^2-Error versus the CPU time of the three algorithms. To see the convergence behavior more clearly, we provide the zoomed-in plot at the crossing point in the first few iterations in Fig. 3. From Table 1 and Fig. 3, we can see that when the tolerance value changes from 10^{-4} to 10^{-6}, our method gives stable and satisfactory results, while D-LLT has much improvement both in the visual effect and the PSNR value. Therefore, the D-LLT algorithm requires higher accuracy hence much more iterations and time to achieve the same solution as the PDHG-LLT algorithm. Moreover, the PDHG-LLT algorithm is slightly slower than the AL-LLT algorithm when the tolerance is low but faster when the tolerance is higher, which can be seen from the zoomed-in plot in Fig. 3.

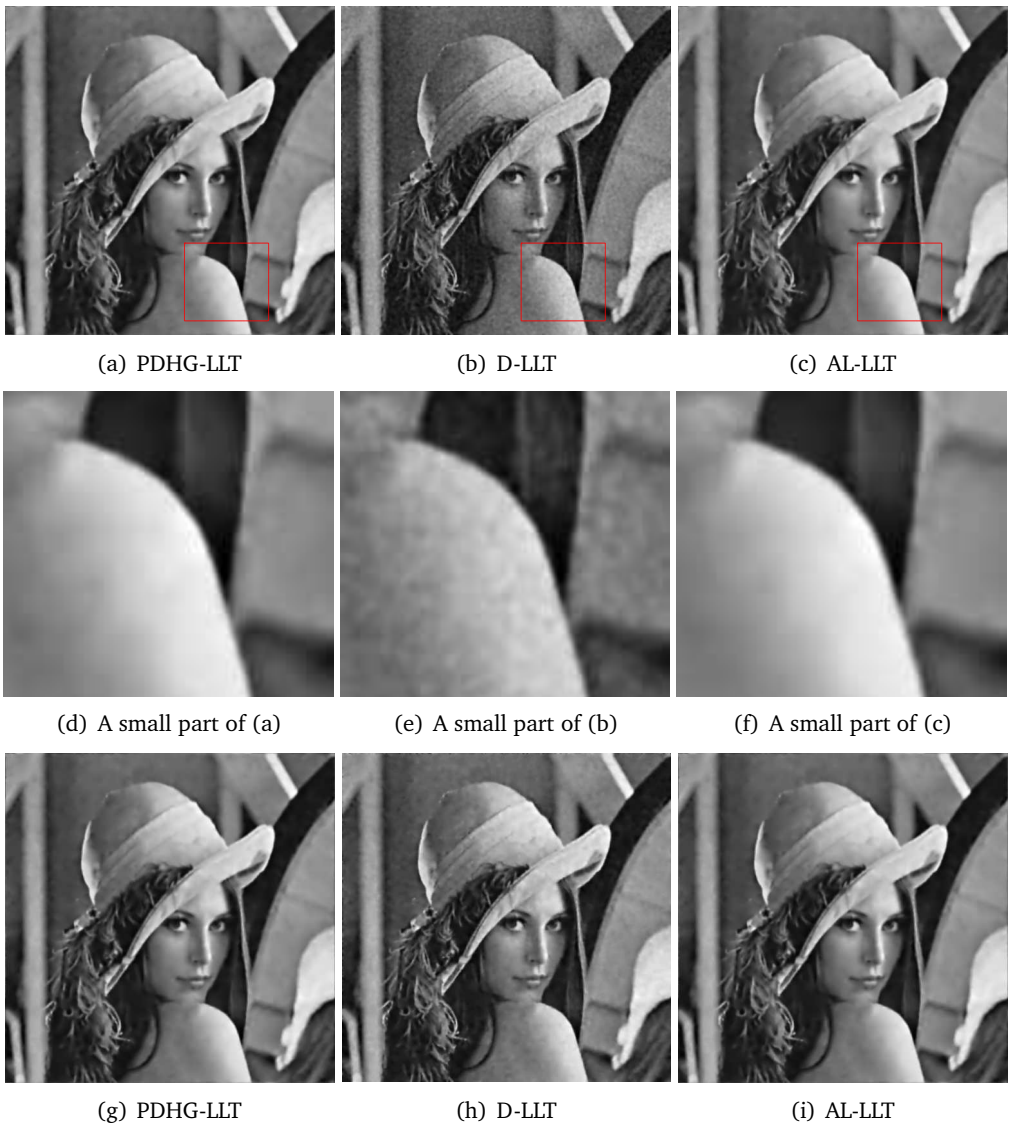
Then Fig. 4, Fig. 5 and Table 2 show results of comparisons between the three algorithms for the noisy Man image in Fig. 1(d). We can see that the PDHG-LLT algorithm is



(a) Lena (gray); size: 512×512 (b) Man; size: 512×512

(c) Noisy image; σ=0.01; PSNR=20.25 (d) Noisy image; σ=0.01; PSNR=20.50

Figure 1: Original gray images and the degraded ones.



(a) PDHG-LLT (b) D-LLT (c) AL-LLT

(d) A small part of (a) (e) A small part of (b) (f) A small part of (c)

(g) PDHG-LLT (h) D-LLT (i) AL-LLT

Figure 2: Restored images for different TOL values by the PDHG-LLT algorithm, the D-LLT algorithm and the AL-LLT algorithm, respectively. Top row: restored images for TOL=10^{-4}. Middle row: a small portion of the restored images in the top row. Bottom row: restored images for TOL=10^{-6}.

significantly faster than the D-LLT algorithm for various tolerance values. Furthermore, the PDHG-LLT algorithm also converges more slowly in a low level of accuracy and becomes faster than the AL-LLT algorithm as the level of accuracy increases.

5.2. Color image: $d = 3$

In this subsection, we provide some examples for color image denosing. The original color images and their degraded images are listed in Fig. 6.

Firstly, in Fig. 7, Fig. 8 and Table 3, we present the results of the PDHG-LLT, the D-LLT and the AL-LLT algorithm for the color Rose image in Fig. 6(a). From the figures and data, we conclude that our PDHG-LLT algorithm and the AL-LLT algorithm are much faster

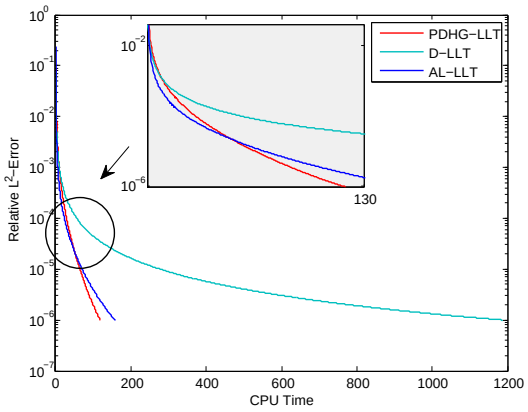


Figure 3: Plots of relative L^2-Error vs. CPU time for the gray Lena image by different algorithms (TOL=10^{-6}).

(a) PDHG-LLT (b) D-LLT (c) AL-LLT

(d) PDHG-LLT (e) D-LLT (f) AL-LLT

Figure 4: Restored images for different TOL values by the PDHG-LLT algorithm, the D-LLT algorithm and the AL-LLT algorithm, respectively. The first row: restored images for TOL=10^{-4}. The second row: restored images for TOL=10^{-6}.

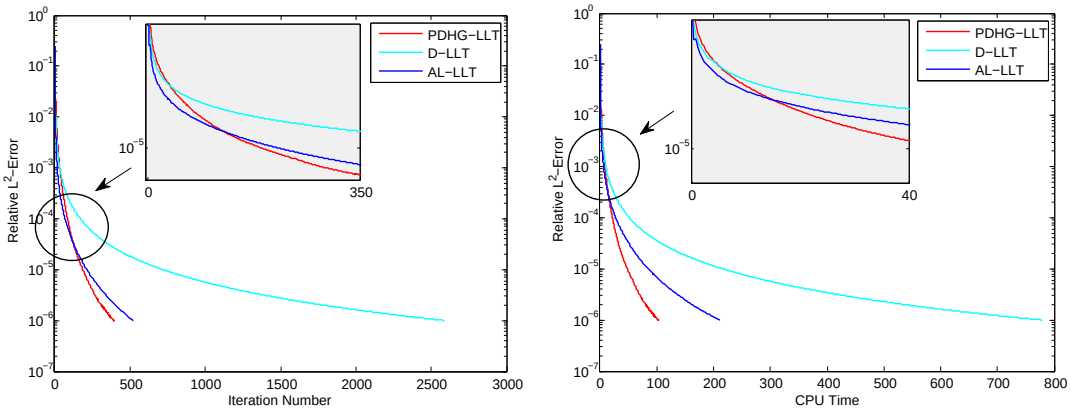


Figure 5: Plots of relative L^2-Error vs. iteration number and CPU time for the Man image by different algorithms (TOL=10^{-6}). Left: L^2-Error vs. iteration number. Right: L^2-Error vs. CPU time.

Table 1: Computational efforts for the gray Lena image: β =0.04.

	TOL=10^{-4}			TOL=10^{-6}			
Algorithm	Iter	CPU	PSNR	Iter	CPU	T-One[a]	PSNR
PDHG-LLT	116	28.56	28.43	468	118.77	0.254	28.52
D-LLT	186	55.97	28.44	3869	1118	0.307	28.52
AL-LLT	60	23.8	28.38	394	158.3	0.402	28.45

[a] The average CPU time of one iteration.

Table 2: Computational efforts for the Man image: β =0.05.

	TOL=10^{-3}			TOL=10^{-4}			TOL=10^{-6}		
Algorithm	Iter	CPU	PSNR	Iter	CPU	PSNR	Iter	CPU	PSNR
PDHG-LLT	35	8.59	26.34	85	21.94	26.36	396	103.36	26.36
D-LLT	33	10.03	26.67	171	52.66	26.43	2589	778.89	26.35
AL-LLT	18	6.86	25.41	71	28.36	25.92	523	211.6	26.05

Table 3: Computational efforts for the Rose image: β =0.07.

	TOL=10^{-4}			TOL=10^{-6}			
Algorithm	Iter	CPU	PSNR	Iter	CPU	T-One[a]	PSNR
PDHG-LLT	38	6.12	22.14	91	15.76	0.173	22.15
D-LLT	98	23.18	21.99	802	192.34	0.240	22.13
AL-LLT	33	14.92	21.49	194	94.76	0.489	21.52

[a] The average CPU time of one iteration.

than D-LLT algorithm in the vectorial case. Furthermore, the superiority of our algorithm is more and more obvious as the tolerance level becomes smaller.

Then we use the color Lena image to evaluate again the performance of the algorithms by increasing the noise level to $\sigma = 0.02$ as shown in Fig. 6(d). For the same tolerance level, the results of our PDHG-LLT is better than D-LLT algorithm and the convergence

(a) Rose; size: 250×303 (b) Lena (color); size: 512×512

(c) Noisy image; σ=0.01; PSNR=16.06 (d) Noisy image; σ=0.02; PSNR=12.61

Figure 6: Original color images and the degraded ones.

(a) PDHG-LLT (b) D-LLT (c) AL-LLT

Figure 7: Restored images for TOL=10^{-6} by the PDHG-LLT algorithm, the D-LLT algorithm and the AL-LLT algorithm, respectively.

Table 4: Computational efforts for the color Lena image: β =0.04.

	TOL=10^{-3}			TOL=10^{-4}			TOL=10^{-6}		
Algorithm	Iter	CPU	PSNR	Iter	CPU	PSNR	Iter	CPU	PSNR
PDHG-LLT	29	20.24	23.49	51	34.38	23.67	108	72.20	23.71
D-LLT	31	29.00	21.47	142	135.17	22.58	1280	1216.9	23.66
AL-LLT	10	11.87	23.51	25	32.47	23.58	189	248.9	23.59

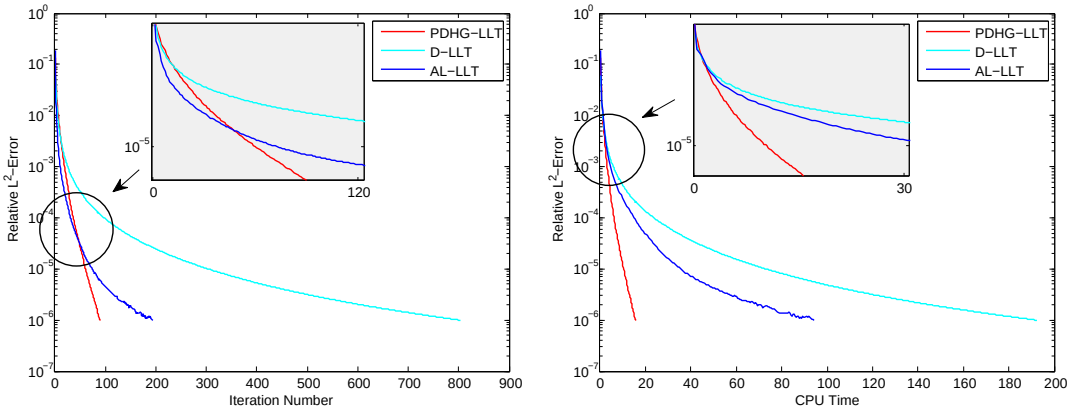


Figure 8: Plots of L^2-Error vs. iteration number and CPU time for the Rose image (TOL=10^{-6}). Left: L^2-Error vs. iteration number. Right: L^2-Error vs. CPU time.

is faster as we can see from Fig. 9 and Table 4. Furthermore, compared to the D-LLT algorithm, the saved time of our algorithm raises rapidly as the dimension d increases as can be seen from Fig. 3 and Fig. 10. As the gray-scale case, the PDHG-LLT algorithm is the fastest algorithm among the three algorithms in a high level of accuracy in the vectorial case.

Obviously, the increase of the size or the dimension d of the noisy image can increase the computational efforts of each algorithm. As demonstrated above, the total cost saving by our algorithm is rather considerable compared with the D-LLT algorithm. The reduction of the computational time mainly comes from both the huge decrease of the total iteration number and the reduction of the cost time of each iteration. For the latter point, we can see from Table 1 and Table 3 that the average cost saving for one iteration of our algorithm compared with D-LLT algorithm is about 30%.

6. Conclusion

High order models usually impose severe restrictions on the time step size. In this paper, we have presented an efficient primal-dual based gradient method to solve the fourth-order LLT model for both gray-scale and vectorial image denoising. The proposed algorithm avoids the regularization parameter for the nondifferentiable term by employing the primal-dual formulation. Gradient-type methods are used to update the primal and dual variables alternately. In each iteration, the primal variable and the dual variable use

the updated information from each other, which makes the convergence very fast. We compare our method with the existing methods for the LLT model. Various numerical comparisons for gray and color image restoration show that our algorithm is much more efficient than the dual algorithm due to the huge reduction in the total iteration number and cost saving at each iteration. Comparing with the augmented Lagrangian algorithm, the convergence of the proposed primal-dual algorithm is slightly slower at a low level of accuracy but becomes faster at a relatively high level of accuracy.

(a) PDHG-LLT (b) D-LLT (c) AL-LLT

(d) A small portion of (a) (e) A small portion of (b) (f) A small portion of (c)

(g) PDHG-LLT (h) D-LLT (i) AL-LLT

Figure 9: Restored images for different TOL values. Top row: restored images for TOL=10^{-4} by the PDHG-LLT algorithm, the D-LLT algorithm and the AL-LLT algorithm, respectively. Middle row: a small portion of the restored images in the top row. Bottom row: restored images for TOL=10^{-6} by different methods.

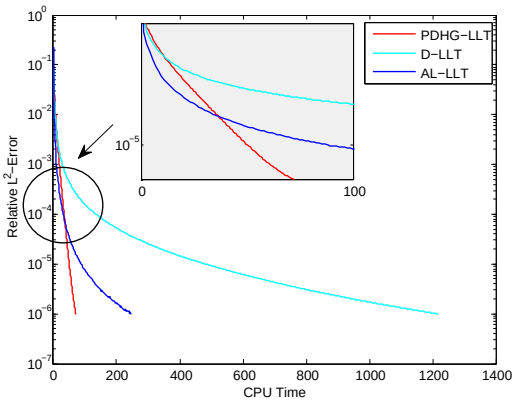


Figure 10: Plots of L^2-Error vs. CPU time for the color Lena image (TOL=10^{-6}).

Acknowledgments The authors express sincerely thanks to Professor Xuecheng Tai and Chunlin Wu for providing the code for the augmented Lagrangian method. The first author would like to thank Jiangli Shi and Xiaojing Ye for their help in the FFT algorithm.

References

[1] P. Blomgren and T.F. Chan, *Color TV: total variation methods for restoration of vector valued images*, IEEE Trans. Image Process., 7 (1996), pp. 304–309.

[2] X. Bresson and T.F. Chan, *Fast dual minimization of the vectorial total variation norm and applications to color image processing*, Inverse Probl. Imag., 2 (2008), pp. 455–48.

[3] X. Bresson, S. Esedoglu, P. Vandergheynst, J. Thiran and S. Osher, *Fast global minimization of the active contour/snake model*, J. Math. Imaging Vis., 28 (2007), pp. 151–167.

[4] A. Chambolle, *An algorithm for total variation minimization and applications*, J. Math. Imaging Vis., 20 (2004), pp. 89–97.

[5] A. Chambolle and P. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.

[6] T.F. Chan, S. Esedoglu and F.E. Park, *A fourth order dual method for staircase reduction in texture extraction and image restoration problems*, Technical Report 05-28, UCLA CAM Reports, 2005.

[7] T.F. Chan, G.H. Golub and P. Mulet, *A nonlinear primal dual method for total variation based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[8] T.F. Chan, A. Marquina and P. Mulet, *High-order total variation-based image restoration*, SIAM J. Sci. Comput., 22 (2000), pp. 503–516.

[9] H. Chen, J. Song and X.-C. Tai, *A dual algorithm for minimization of the LLT model*, Adv. Comput. Math., 31 (2009), pp. 115–130.

[10] T. Goldstein and S. Osher, *The split Bregman method for L1-regularized problems*, SIAM J. Imaging Sci., 2 (2009), pp. 323–343.

[11] M.R. Hajiaboli, *An anisotropic fourth-order diffusion filter for image noise removal*, Int. J. Comput. Vis., 92 (2011), pp. 177–191.

[12] F. Li, C. Shen, J. Fan and C. Shen, *Image restoration combining a total variational filter and a fourth-order filter*, J. Vis. Commun. Image R., 18 (2007), pp. 322–330.

[13] M. Lysaker, A. Lundervold and X.-C. Tai, *Noise removal using fourth-order partial differential equation with applications to medical magnetic resonance images in space and time*, IEEE Trans. Image Process., 12 (2003), pp. 1579–1590.
[14] M. Lysaker and X.-C. Tai, *Iterative image restoration combining total variation minimization and a second-order functional*, Int. J. Comput. Vis., 66 (2006), pp. 5–18.
[15] L. Rudin, S. Osher and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, 60 (1992), pp. 259–268.
[16] G. Steidl, *A note on the dual treatment of higher-order regularization functionals*, Computing, 76 (2006), pp. 135–148.
[17] Y. Wang, W. Yin and Y. Zhang, *A fast algorithm for image deblurring with total variation regularization*, Rice University CAAM Technical Report TR07-10, 2007.
[18] R.T. Whitaker and S.M. Pizer, *A multi-scale approach to nonuniform diffusion*, Comput. Vis. Graph. Image Process.: Image Understand., 57 (1993), pp. 99–110.
[19] C. Wu and X.-C. Tai, *Augmented Lagrangian method, dual methods, and split bregman iteration for ROF model*, SSVM 2009: pp. 502-513.
[20] C. Wu and X.-C. Tai, *Augmented Lagrangian method, dual methods, and split Bregman iteration for ROF, vectorial TV, and high order models*, SIAM J. Imaging Sci., 3 (2010), pp. 300–339.
[21] Y.-L. You and M. Kaveh, *Fourth-order partial differential equations for noise removal*, IEEE Trans. Image Process., 9 (2000), pp. 1723–1730.
[22] M. Zhu and T.F. Chan, *An efficient primal-dual hybrid gradient algorithm for total variation image restoration*, Technical Report 08-34, UCLA CAM Reports, 2008.