Neurocomputing ∎ (∎∎∎∎) ∎∎∎–∎∎∎

Contents lists available at ScienceDirect

Neurocomputing

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

# A fast alternating time-splitting approach for learning partial differential equations

Zhenyu Zhao [a], Zhouchen Lin [b,c,*], Yi Wu [a]

[a] College of Science, National University of Defense Technology, PR China
[b] Key Lab. of Machine Perception (MOE), School of EECS, Peking University, PR China
[c] Cooperative Medianet Innovation Center, Shanghai Jiaotong University, PR China

ARTICLE INFO

*Article history:*
Received 3 October 2014
Received in revised form
13 May 2015
Accepted 20 October 2015
Communicated by T. Heskes

*Keywords:*
Learning-based PDEs
PDE constrained optimal control
FATSA
Computer vision
Image processing

ABSTRACT

Learning-based partial differential equations (PDEs), which combine fundamental differential invariants into a nonlinear regressor, have been successfully applied to several computer vision and image processing problems. However, the gradient descent method (GDM) for solving the linear combination coefficients among differential invariants is time-consuming. Moreover, when the regularization or constraints on the coefficients become more complex, it is troublesome or even impossible to deduce the gradients. In this paper, we propose a new algorithm, called fast alternating time-splitting approach (FATSA), to solve the linear combination coefficients. By minimizing the difference between the expected output and the actual output of PDEs *at each time step*, FATSA can solve the linear combination coefficients much faster than GDM. More complex regularization or constraints can also be easily incorporated. Extensive experiments demonstrate that our proposed FATSA outperform GDM in both speed and quality.



## 1. Introduction

Partial differential equations (PDEs) have been successfully used to solve many practical problems in computer vision and image processing [1–3], such as denoising [4,5], enhancement [6,7], inpainting [8], segmentation [9,10], and optical flow computation [11,12]. However, it is usually difficult to design a PDE system for a particular task which requires high mathematical skills and good insight into the problem. According to [13], the existing methods of designing PDEs can be mainly classified into two groups. The methods of the first group write down PDEs directly, which requires good mathematical understandings on the properties of the PDEs. The methods of second group first define an energy functional [14], which pursues the expected properties of the output image or video, and then derive evolution equations by computing the Euler–Lagrange variation of the energy functional. For example, the ROF model [15] and TV-$L_1$ [16] for image denoising are designed directly, while the Nambu model [17] and the PL model [18] for color image processing are designed in the variational way.

To reduce the difficulty in designing PDEs for complex vision problems, Liu and Lin et al. [13] proposed a framework that learns PDEs from training image pairs recently. They first considered learning PDEs for grayscale image restoration [19], which involve an anisotropic diffusion term. Then they generalized the idea significantly by linearly combining fundamental differential invariants that are invariant to translation and rotation. These differential invariants serve as "bases" of differential operators [13,20]. They utilized the gradient descent method (GDM) to solve the linear combination coefficients. The learnt PDEs have been successfully applied to various problems, such as image denoising, debluring, object detection, color to gray and demosaicking [13,21,22].

However, GDM has several drawbacks. First, the convergence speed of GDM is very slow due to the fact that objective functional is flat. Experiments show that the magnitude of gradient is usually at the order of $10^{-3}$ (Fig. 4), even at the beginning iterations. Therefore, the solution of GDM does not improve the initial value very much. Second, it needs to solve the adjoint PDEs to obtain the gradient, which is difficult to deduce and also time-consuming. Third, when the regularization or constraints on the linear combination coefficients become more complex, e.g., we use $L_1$ norm as the regularizer or add boundedness constraints, the deduction of gradient becomes very involved or even non-existent because of the non-differentiablity of the objective functional. Last, the quality of learnt PDEs is not very good. For example, the magnitudes of coefficients are unbalanced. We can see from Fig. 4(a) that most of $a_i(t)$'s are close to zeros while some jump to more than 20. This can cause numerical instability as the differential invariants

* Corresponding author.
*E-mail addresses:* dwightzzy@gmail.com (Z. Zhao), zlin@pku.edu.cn (Z. Lin), wuyi_work@sina.com (Y. Wu).

http://dx.doi.org/10.1016/j.neucom.2015.10.126
0925-2312/

involve multiplications of second order derivatives. As a result, blowup might occur when we apply the learnt PDEs to test images. Moreover, we can also see from Fig. 4(c) that $b_i(t)$'s are very close to zeros, which means that the indicator function is actually ineffective.

To overcome the above short-comings of GDM, we propose a new method, called fast alternating time-splitting approach (FATSA), to solve the linear combination coefficients. We first discretize the PDEs in time. Then we minimize the difference between the expected output (ground truth) and the actual output of the PDEs at each time step $n$, which is a nonlinear regression problem and can be solved by alternately minimizing $a_i(n\Delta t)$'s and $b_j((n-1)\Delta t)$'s. In such a greedy manner, the linear combination coefficients can be updated sequentially in time. Based on FATSA, it is convenient to add constrains and regularization on the coefficients, even when these constrains and regularization are non-differentiable. Moreover, we do not need to deduce and compute the adjoint PDEs any longer. Besides, compared with GDM, FATSA can greatly reduce the training time and the training error. For grayscale images, the speed of training is accelerated by ten times. For color images, the training time is cut by half. In summary, the contributions of this paper are summarized as follows:

- We propose a new fast alternating time-splitting approach (FATSA) to solve the PDE constrained optimal control problem, which not only speeds up the learning process, but also improves the results.
- Compared with GDM, FATSA is much simpler. It computes the linear combination coefficients in temporal order. It avoids to compute the adjoint PDEs for evaluating the Gâteaux derivatives [23] of the objective functional.
- FATSA is much more flexible than GDM. When we add more general regularizations (e.g., non-smooth regularization) and extra constraints on the linear combination coefficients, it can also improve the results.

The rest of the paper is organized as follows. First of all, we review the learning-based PDEs methods briefly in Section 2. Then we present the main idea of FATSA and the details of alternating minimization in Section 3. In Section 4, we discuss the complexity of FATSA and make a detailed comparison with GDM [13]. We also extend FATSA to solve learning-based PDEs for vector-valued image processing problems. Then in Section 5, we compare the performance of FATSA and GDM on some computer vision and image processing problems. Finally, we conclude our paper and discuss the future work in Section 6.

## 2. Learning-based PDEs

In this section, we briefly review the framework of learning-based PDEs for computer vision and image processing problems. More details can be found in [13,21,22].

### 2.1. Mathematical formulation

The current learning-based PDEs are grounded on the translational and rotational invariance of computer vision and image processing problems. Namely, when the input image is translated or rotated, the output image should be translated or rotated accordingly. Then it can be proven that the governing equations are functions of fundamental differential invariants, which form "bases" of all differential invariants that are invariant with respect to translation and rotation. We assume that the evolution of the image $u$ is guided by an indicator function $v$, which collects large scale information. As shown in Table 1, there are 17 fundamental differential invariants $\{\mathrm{inv}_i(u,v),\ i=0,\dots,16\}$ up to the second order. For brevity, we denote $\mathbf{inv}(u,v)=[\mathrm{inv}_0(u,v),\mathrm{inv}_1(u,v),\dots,\mathrm{inv}_{16}(u,v)]^T$, where $(\cdot)^T$ denoted the transpose of matrix (or vector).

**Table 1**
Fundamental differential invariants up to the second order, where tr is the trace operator and $\nabla f$ and $\mathbf{H}_f$ are the gradient and the Hessian matrix of function $f$, respectively.

| $j$ | $\mathrm{inv}_j(u,v)$ |
|---|---|
| 0,1,2 | $1, v, u$ |
| 3,4 | $\|\nabla v\|^2=v_x^2+v_y^2,\ \|\nabla u\|^2=u_x^2+u_y^2$ |
| 5 | $(\nabla v)^T\cdot\nabla u=v_xu_x+v_yu_y$ |
| 6,7 | $\mathrm{tr}(\mathbf{H}_v)=v_{xx}+v_{yy},\ tr(\mathbf{H}_u)=u_{xx}+u_{yy}$ |
| 8 | $(\nabla v)^T\cdot\mathbf{H}_v\cdot\nabla v=v_x^2v_{xx}+2v_xv_yv_{xy}+v_y^2v_{yy}$ |
| 9 | $(\nabla v)^T\cdot\mathbf{H}_u\cdot\nabla v=v_x^2u_{xx}+2v_xv_yu_{xy}+v_y^2u_{yy}$ |
| 10 | $(\nabla v)^T\cdot\mathbf{H}_v\cdot\nabla u=v_xu_xv_{xx}+(v_xu_y+u_xv_y)v_{xy}+v_yu_yv_{yy}$ |
| 11 | $(\nabla v)^T\cdot\mathbf{H}_u\cdot\nabla u=v_xu_xu_{xx}+(v_xu_y+u_xv_y)u_{xy}+v_yu_yu_{yy}$ |
| 12 | $(\nabla u)^T\cdot\mathbf{H}_v\cdot\nabla u=u_x^2v_{xx}+2u_xu_yv_{xy}+u_y^2v_{yy}$ |
| 13 | $(\nabla u)^T\cdot\mathbf{H}_u\cdot\nabla u=u_x^2u_{xx}+2u_xu_yu_{xy}+u_y^2u_{yy}$ |
| 14 | $\mathrm{tr}(\mathbf{H}_v^2)=v_{xx}^2+2v_{xy}^2+v_{yy}^2$ |
| 15 | $\mathrm{tr}(\mathbf{H}_v\cdot\mathbf{H}_u)=v_{xx}u_{xx}+2v_{xy}u_{xy}+v_{yy}u_{yy}$ |
| 16 | $\mathrm{tr}(\mathbf{H}_u^2)=u_{xx}^2+2u_{xy}^2+u_{yy}^2$ |

The simplest function of fundamental differential invariants is a linear combination of them. Therefore, learning the PDEs can be transformed into learning the linear combination coefficients among the fundamental differential invariants, which are functions of time $t$ only and independent of spatial variables [13,21,22]. To this end, one may prepare a number of input/output training image pairs. By minimizing the difference between the output of PDEs and the ground truth. We set the initial function as the input image. This results in a PDEs constrained optimal control problem:

$$\min_{\mathbf{a},\mathbf{b}} E(\mathbf{a}(t),\mathbf{b}(t))=\frac{1}{2}\sum_{m=1}^{M}\int_{\Omega}(O_m-u_m(x,y,T))^2\,\mathrm{d}\Omega + \lambda_1\sum_{i=0}^{16}\int_0^T a_i^2(t)\,\mathrm{d}t+\lambda_2\sum_{i=0}^{16}\int_0^T b_i^2(t)\,\mathrm{d}t, \tag{1}$$

$$\text{s.t.}\quad \begin{cases} \dfrac{\partial u_m}{\partial t}-\mathbf{inv}^T(u_m,v_m)\cdot\mathbf{a}(t)=0, & (x,y,t)\in Q,\\ u_m(x,y,t)=0, & (x,y,t)\in\Gamma,\\ u_m(x,y,0)=I_m, & (x,y)\in\Omega,\\ \dfrac{\partial v_m}{\partial t}-\mathbf{inv}^T(v_m,u_m)\cdot\mathbf{b}(t)=0, & (x,y,t)\in Q,\\ v_m(x,y,t)=0, & (x,y,t)\in\Gamma,\\ v_m(x,y,0)=I_m, & (x,y)\in\Omega, \end{cases} \tag{2}$$

where $\{(I_m,O_m),\ m=1,\dots,M\}$ denote the $M$ input/output training image pairs, $u_m(x,y,t)$ is the evolution image at time $t$ with respect to the input image $I_m$, $v_m(x,y,t)$ is the corresponding indicator function, $\Omega\subset\mathbb{R}^2$ is the (rectangular) region occupied by the image,[1] $T$ is the temporal span of evolution which can be normalized as 1, $Q=\Omega\times[0,T]$, $\Gamma=\partial\Omega\times[0,T]$, and $\partial\Omega$ denotes the boundary of $\Omega$. The last two terms in (1) are regularization terms on the coefficients $a_i(t)$ and $b_i(t)$. We denote it as $\mathbf{a}(t)=[a_0(t),a_1(t),\dots,a_{16}(t)]^T$ and $\mathbf{b}(t)=[b_0(t),b_1(t),\dots,b_{16}(t)]^T$ for brevity. For $\mathrm{inv}_i(v,u)$, it can be acquired by simply switching $u$ and $v$ in $\mathrm{inv}_i(u,v)$.

[1] The images are padded with zeros of several pixels width around them, so that the Dirichlet boundary conditions, $u_m(x,y,t)=0, v_m(x,y,t)=0, (x,y,t)\in\Gamma$, are naturally fulfilled.

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

### 2.2. Gradient descend method

Liu and Lin et al. [13,21,22] proposed a gradient descend method (GDM) to solve the optimal coefficients $a_i(t)$ and $b_i(t)$, where the "gradient" is actually the Gâteaux derivatives [23] of the objective functional $E$ with respect to the coefficient functions. The Gâteaux derivatives of $E$ with respect to $a_i(t)$ and $b_i(t)$ are as follows:

$$\begin{cases} \dfrac{DE}{Da_i} = \lambda_1 a_i - \sum_{m=1}^{M} \int_{\Omega} \varphi_m \mathrm{inv}_i(u_m, v_m)\, \mathrm{d}\Omega, \\ \dfrac{DE}{Db_i} = \lambda_2 b_i - \sum_{m=1}^{M} \int_{\Omega} \psi_m \mathrm{inv}_i(v_m, u_m)\, \mathrm{d}\Omega, \end{cases} \tag{3}$$

where $\varphi$ and $\psi$ are the solutions to the adjoint equations. Since the adjoint equations are very complex, we omit the details here.

Based on the Gâteaux derivatives, the local optimal solutions of $\mathbf{a}(t)$ and $\mathbf{b}(t)$ can be computed via linear searching along descent directions (e.g., conjugate gradient).

## 3. FATSA: fast alternating time-splitting approach

The Gâteaux derivatives shown above are only available for smooth regularizations, such as squared $L_2$ norms in (1), on the coefficients. However, when the regularizations are non-smooth or extra constraints are added to the coefficients, it is nearly impossible to deduce the Gâteaux derivatives. Besides, it is time-consuming to solve the adjoint equations. Moreover, as the objective functional is quite flat, the magnitudes of all the Gâteaux derivatives are very small. As a result, the speed of GDM is very slow and the obtained coefficients are not very far from their initial values. Consequently, the performance of learnt PDEs is not always satisfactory.

To tackle the above drawbacks of GDM, in this paper we propose a greedy strategy that minimizes the difference between the expected outputs (ground truth) and the actual outputs of the PDEs at *every time step*, rather than minimizing the difference between the expected outputs and the actual outputs of the PDEs at $T$ only. In view of the highly non-convex nature of the PDE constrained optimal control problem, it is very easy for GDM to be stuck at local minima during solving the optimal coefficients. Compared with GDM, our new greedy strategy is more effective, which can be demonstrated by experiments.

In the proposed approach, we first discretize the temporal variable $t$ with a step size $\Delta t$ and denote it as $t_i = i \cdot \Delta t$, $i = 0, \ldots, N$. At each time step $t_{n+1}$, we minimize

$$L(t_{n+1}) = \frac{1}{2} \sum_{m=1}^{M} \int_{\Omega} (O_m - u_m(x, y, t_{n+1}))^2\, \mathrm{d}\Omega, \tag{4}$$

where $u_m(x, y, t_{n+1})$ is the solution of PDEs (2) at time $t_{n+1}$. The motivation of minimizing (4) is to attract $u_m$ to the desired output $O_m$ as fast as possible, while GDM minimizes

$$E' = \frac{1}{2} \sum_{m=1}^{M} \int_{\Omega} (O_m - u_m(x, y, T))^2\, \mathrm{d}\Omega, \tag{5}$$

which is the difference between the expected output and the final outputs of the PDEs ($t = T$).

In the sequel, we simply use ${u_m}^n$ to denote $u_m(x, y, t_n)$. Other notations, such as ${v_m}^n$, $\mathbf{a}^n$, $\mathbf{b}^n$ and $L^n$, are denoted in the same way.

We use the following forward scheme to approximate the governing equations in (2),

$$\begin{cases} u_m^{n+1} = u_m^n + \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^n) \cdot \mathbf{a}^n, & n \geq 0, \\ v_m^n = v_m^{n-1} + \Delta t \cdot \mathbf{inv}^T(v_m^{n-1}, u_m^{n-1}) \cdot \mathbf{b}^{n-1}, & n \geq 1. \end{cases} \tag{6}$$

By combining (4) and (6), we can obtain that

$$L^{n+1} = \frac{1}{2} \sum_{m=1}^{M} \int_{\Omega} \left[ O_m - u_m^n - \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^n) \cdot \mathbf{a}^n \right]^2 \mathrm{d}\Omega. \tag{7}$$

Note that when $n \geq 1$, $L^{n+1}$ is dependent on $\mathbf{a}^n$ and $\mathbf{b}^{n-1}$, where the dependence on $\mathbf{b}^{n-1}$ is due to ${v_m}^n$ in $\mathbf{inv}^T(u_m^n, v_m^n)$, which can be computed by the second equation of (6). When $n = 0$, $L^1$ is only dependent on $\mathbf{a}^0$ as ${v_m}^0$ is known. So we can initialize $\mathbf{a}^0$ by minimizing $L^1$. Then $\mathbf{a}^n$ and $\mathbf{b}^{n-1}$ $(n \geq 1)$ can be computed sequentially by minimizing $L^{n+1}$ $(n \geq 1)$.

In order to show the flexibility of our new optimization method as well as ensure the computation stability, we further add boundedness constraints on $\mathbf{a}^n$ and $\mathbf{b}^n$. Then the problem of computing $\mathbf{a}^0$ is transformed into

$$\begin{aligned} &\min_{\mathbf{a}^0} L^1 = \frac{1}{2} \sum_{m=1}^{M} \int_{\Omega} \left[ O_m - u_m^0 - \Delta t \cdot \mathbf{inv}^T(u_m^0, v_m^0) \cdot \mathbf{a}^0 \right]^2 \mathrm{d}\Omega, \\ &\text{s.t. } \|\mathbf{a}^0\|_{\infty} \leq \eta_1, \end{aligned} \tag{8}$$

and the problem of solving $\mathbf{a}^n$ and $\mathbf{b}^{n-1}$ $(n \geq 1)$ is transformed into

$$\begin{aligned} &\min_{\mathbf{a}^n, \mathbf{b}^{n-1}} L^{n+1} = \frac{1}{2} \sum_{m=1}^{M} \int_{\Omega} \left[ O_m - u_m^n - \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^n) \right]^2 \mathrm{d}\Omega, \\ &\text{s.t.} \quad \|\mathbf{a}^n\|_{\infty} \leq \eta_1, \|\mathbf{b}^{n-1}\|_{\infty} \leq \eta_2. \end{aligned} \tag{9}$$

Problem (8) is simply a quadratic optimization problem with a boundedness constraint. As for problem (9), we can use the block coordinate descent method to update $\mathbf{a}^n$ and $\mathbf{b}^{n-1}$ alternately. Hence we call our new optimization method as fast alternating time-splitting approach (FATSA), which is summarized in Algorithm 1. We will present the details in the following subsections.

**Algorithm 1.** FATSA for training PDEs.

**Input** Training image pairs $\{(I_m, O_m)\}_{m=1}^{M}$.
**Initialize** $u_m^0 = I_m, v_m^0 = I_m, \Delta t = 0.05, \varepsilon = 10^{-3}, N = 100$.
**Step 0** Compute $\mathbf{a}^0$ by solving problem (8).
**while** not converged **do**
1. Compute $\mathbf{a}^n$ and $\mathbf{b}^{n-1}$ by solving problem (9),
2. Compute $u_m^{n+1}$ and $v_m^n$ by using (6),
3. Check the convergence conditions:
   $|L^{n+1} - L^n| / L^n < \varepsilon$ or $n > N$,
4. $n \leftarrow n + 1$.

**end while**

### 3.1. Alternating minimization

In this subsection, we focus on solving (9) by alternating minimization. As for (8), it is easy to see that this problem is same as the one when we update $\mathbf{a}^n$ with fixed $\mathbf{b}^{n-1}$ in (9) (see (10) below). So we skip the details of solving (8).

By alternating minimization, (9) reduces to the following two subproblems. While $\mathbf{b}^{n-1}$ is fixed, we can get $v_m^n = v_m^{n-1} + \Delta t \cdot \mathbf{inv}^T(v_m^{n-1}, u_m^{n-1}) \cdot \mathbf{b}^{n-1}$ and $\mathbf{inv}(u_m^n, v_m^n)$. Then the problem for updating $\mathbf{a}^n$ is transformed into

$$\min_{\mathbf{a}^n} L_a^{n+1} = \frac{1}{2} \sum_{m=1}^{M} \int_{\Omega} \left[ O_m - u_m^n - \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^n) \cdot \mathbf{a}^n \right]^2 \mathrm{d}\Omega,$$

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

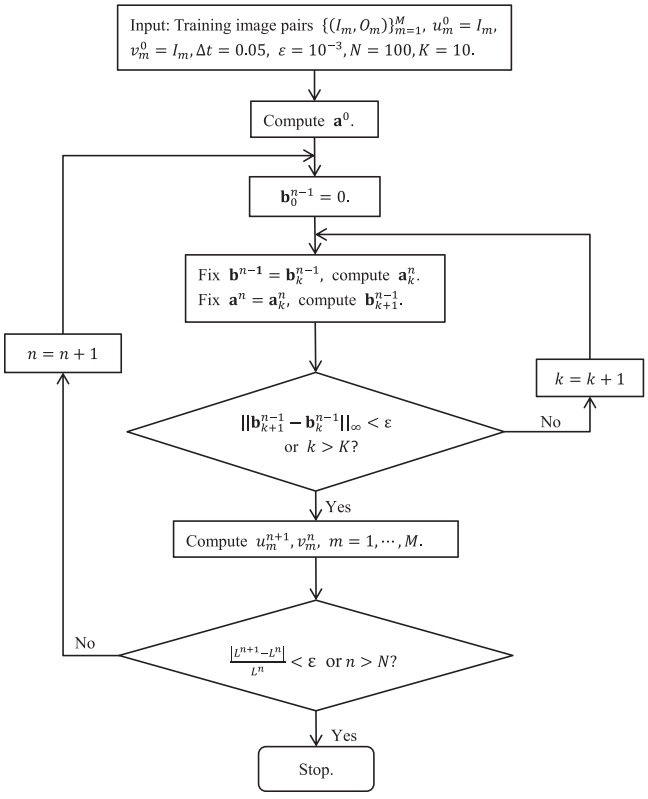


**Fig. 1.** Flowchart of the proposed FATSA.

$$\text{s.t. } \|\mathbf{a}^n\|_\infty \le \eta_1. \tag{10}$$

When $\mathbf{a}^n$ is fixed, problem (9) is simplified into the following problem:

$$\min_{\mathbf{b}^{n-1}} L_b^{n+1} = \frac{1}{2}\sum_{m=1}^{M}\int_\Omega \Big[O_m - u_m^n - \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^{n-1} + \Delta t \cdot \mathbf{inv}^T(v_m^{n-1}, u_m^{n-1}) \cdot \mathbf{b}^{n-1}) \cdot \mathbf{a}^n\Big]^2 \, d\Omega,$$

$$\text{s.t. } \|\mathbf{b}^{n-1}\|_\infty \le \eta_2. \tag{11}$$

We initialize $\mathbf{b}^{n-1}$ as $\mathbf{0}$ and summarize the process of solving problem (9) in Algorithm 2. The flowchart to solve the whole problem is presented in Fig. 1. In the following subsections, we present how to solve (10) and (11).

**Algorithm 2.** Alternating minimization for solving problem (9).

**Input** $\{(u_m^n, u_m^{n-1}, v_m^{n-1}, O_m)\}_{m=1}^M, K = 10.$
**Initialize** $\varepsilon = 10^{-3}$, $k = 0$.
**Step 0** $\mathbf{b}_0^{n-1} = \mathbf{0}$.
**while** not converged **do**
1. Fix $\mathbf{b}^{n-1} = \mathbf{b}_k^{n-1}$ and update $\mathbf{a}_k^n$ by solving subproblem (10),
2. Fix $\mathbf{a}^n = \mathbf{a}_k^n$ and update $\mathbf{b}_{k+1}^{n-1}$ by solving subproblem (11),
3. Check the convergence conditions:
$\|\mathbf{b}_{k+1}^{n-1} - \mathbf{b}_k^{n-1}\|_\infty < \varepsilon$ or $k > K$,
4. $k \leftarrow k+1$.
**end while**

### 3.1.1. Solving subproblem (10)

In this subsection, we present the details of solving subproblem (10). From (10),

$$\begin{aligned} L_a^{n+1} &= \frac{1}{2}\sum_{m=1}^{M}\int_\Omega \Big[O_m - u_m^n - \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^n) \cdot \mathbf{a}^n\Big]^2 \, d\Omega \\ &= \frac{1}{2}\sum_{m=1}^{M}\int_\Omega (\Delta t)^2 \cdot (\mathbf{a}^n)^T \cdot \mathbf{inv}(u_m^n, v_m^n) \cdot \mathbf{inv}^T(u_m^n, v_m^n) \cdot \mathbf{a}^n \, d\Omega \\ &\quad - \sum_{m=1}^{M}\int_\Omega \Delta t \cdot (O_m - u_m^n) \cdot \mathbf{inv}^T(u_m^n, v_m^n) \cdot \mathbf{a}^n \, d\Omega \\ &\quad + \frac{1}{2}\sum_{m=1}^{M}\int_\Omega (O_m - u_m^n)^2 \, d\Omega. \end{aligned}$$

Note that $\mathbf{a}^n$ is independent of $(x,y)$. Denote

$$\begin{cases} \mathbf{g}_1 = \Delta t \cdot \sum_{m=1}^{M}\int_\Omega (O_m - u_m^n) \cdot \mathbf{inv}(u_m^n, v_m^n) \, d\Omega, \\ \mathbf{G}_1 = (\Delta t)^2 \cdot \sum_{m=1}^{M}\int_\Omega \mathbf{inv}(u_m^n, v_m^n) \cdot \mathbf{inv}^T(u_m^n, v_m^n) \, d\Omega. \end{cases} \tag{12}$$

Then subproblem (10) can be rewritten as follows:

$$\min_{\mathbf{a}^n} \tfrac{1}{2}(\mathbf{a}^n)^T \cdot \mathbf{G}_1 \cdot \mathbf{a}^n - \mathbf{g}_1^T \cdot \mathbf{a}^n, \quad \text{s.t. } \|\mathbf{a}^n\|_\infty \le \eta_1. \tag{13}$$

It is a quadratic optimization problem with boundedness constraint, which can be conveniently solved by traditional optimization methods, e.g., the trust region reflective method [24].

### 3.1.2. Solving subproblem (11)

Subproblem (11) is not a least squares problem and does not have close-form solution. We adopt the Gauss–Newton method [25] to solve it iteratively. In each iteration, we linearize the term in $\int_\Omega (\cdot)^2 \, d\Omega$ of (11) locally and relax (11) to a least squares problem.

Suppose that $\mathbf{b}_k^{n-1}$ has been obtained at the last iteration and denote $h(\mathbf{b}^{n-1}) = O_m - u_m^n - \Delta t \cdot \mathbf{inv}^T(u_m^n, v_m^{n-1} + \Delta t \cdot \mathbf{inv}^T(v_m^{n-1}, u_m^{n-1}) \cdot \mathbf{b}^{n-1}) \cdot \mathbf{a}^n$. Then the linear approximation of $h$ at $\mathbf{b}_k^{n-1}$ is

$$h(\mathbf{b}^{n-1}) = h(\mathbf{b}_k^{n-1}) + \left(\frac{Dh}{D\mathbf{b}_k^{n-1}}\right)^T \cdot (\mathbf{b}^{n-1} - \mathbf{b}_k^{n-1}), \tag{14}$$

where $\frac{Dh}{D\mathbf{b}_k^{n-1}}$ is the Gâteaux derivatives of $h$ with respect to $\mathbf{b}^{n-1}$. It can be computed as follows:

$$\frac{Dh}{D\mathbf{b}^{n-1}} = -(\Delta t)^2 \cdot \sum_{(p,q)\in\mathcal{P}} \sigma_{pq}(v_m^n) \cdot \mathbf{inv}(u_m^{n-1}, v_m^{n-1}), \tag{15}$$

where

$$\begin{cases} \sigma_{pq}(v_m^n) = \dfrac{\partial \mathbf{inv}^T(u_m^n, v_m^n)}{\partial (v_m^n)_{pq}} \cdot \mathbf{a}^n = \sum_{j=0}^{16} a_j^n \dfrac{\partial \mathrm{inv}_j(u_m^n, v_m^n)}{\partial (v_m^n)_{pq}}, \\ (v_m^n)_{pq} = \dfrac{\partial^{p+q}(v_m^n)}{\partial x^p \partial y^q}, \end{cases} \tag{16}$$

and $\mathcal{P} = \{(0,0),(0,1),(1,0),(2,0),(1,1),(0,2)\}$ is the index set for partial differentiation.

Then we obtain a relaxed problem

$$\min_{\mathbf{b}^{n-1}} \frac{1}{2}\sum_{m=1}^{M}\int_\Omega \left[h(\mathbf{b}_k^{n-1}) + \left(\frac{Dh}{D\mathbf{b}_k^{n-1}}\right)^T \cdot (\mathbf{b}^{n-1} - \mathbf{b}_k^{n-1})\right]^2 d\Omega,$$

$$\text{s.t. } \|\mathbf{b}^{n-1}\|_\infty \le \eta_2. \tag{17}$$

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

Note that $\mathbf{b}^{n-1}$ is independent of $(x,y)$. Denote

$$\begin{cases} \mathbf{g}_2 = \sum_{m=1}^{M} \int_{\Omega} \frac{Dh}{D\mathbf{b}_k^{n-1}} \cdot \left[ h(\mathbf{b}_k^{n-1}) - \left( \frac{Dh}{D\mathbf{b}_k^{n-1}} \right)^T \cdot \mathbf{b}_k^{n-1} \right] \mathrm{d}\Omega, \\ \mathbf{G}_2 = \sum_{m=1}^{M} \int_{\Omega} \frac{Dh}{D\mathbf{b}_k^{n-1}} \cdot \left( \frac{Dh}{D\mathbf{b}_k^{n-1}} \right)^T \mathrm{d}\Omega. \end{cases} \tag{18}$$

Then problem (17) can be rewritten as follows:

$$\min_{\mathbf{b}^{n-1}} \tfrac{1}{2} (\mathbf{b}^{n-1})^T \cdot \mathbf{G}_2 \cdot \mathbf{b}^{n-1} + \mathbf{g}_2^T \cdot \mathbf{b}^{n-1}, \quad \text{s.t. } \|\mathbf{b}^{n-1}\|_\infty \le \eta_2. \tag{19}$$

It is also a quadratic optimization problem with boundedness constraint, which can be conveniently solved by the trust region reflective method [24]. The iteration terminates when the difference between $\mathbf{b}_k^{n-1}$ and $\mathbf{b}_{k-1}^{n-1}$ is sufficiently small.

## 4. Discussions

In this section, we first discuss the complexity of the proposed approach and then make a comparison between FATSA and GDM [13]. Finally, we extend FATSA to handle vector-valued processing problems.

### 4.1. Computational complexity

In this subsection, we discuss the computational complexity of the proposed FATSA. As we set the maximum time $T = N\Delta t$, we can compute an asymptotic worst-case bound for the time complexity. We denote $n_{\mathbf{inv}}$ as the number of invariants (for gray image $n_{\mathbf{inv}} = 17$, for color image $n_{\mathbf{inv}} = 69$) and $|\Omega|$ as the number of pixels in $\Omega$. Problems (13) and (19) are convex quadratic optimization problems for the positive definite $\mathbf{G}_1$ and $\mathbf{G}_2$, respectively. The cubic time complexity $O(n_{\mathbf{inv}}^3)$ is an asymptotic worst-case bound. For each iteration in Algorithm 2, we need to update ${v_m}^n$. So the time complexity is $O(|\Omega|)$. When we have $M$ image training pairs and the maximum number of iteration is $K$, $O(K(M|\Omega| + n_{\mathbf{inv}}^3))$ is the worst-case bound for Algorithm 2. As the terminal time $T = N\Delta t$, the worst-case bound for Algorithm 1 is $O(KN(M|\Omega| + n_{\mathbf{inv}}^3))$.

### 4.2. Comparison between FATSA and GDM

In this subsection, we make a comparison between FATSA and GDM [13]. For the training speed, we compare the training time and errors between FATSA and GDM in Section 5.5. It can be seen from Table 4 that the training time is greatly reduced by FATSA, and the training errors of FATSA are obviously lower than those of GDM in all datasets. In this case, FATSA is much faster and more effective than GDM. For the simplicity, we have shown that FATSA minimizes (4) at each time step, while GDM minimizes (5) at $t=T$ only. This makes the learning of PDEs much simpler as it is unnecessary to compute the adjoint PDEs for evaluating the Gâteaux derivatives [23] of the objective functional. For the flexibility, when minimizing (4) at each time step, it can fit for more general regulations (e.g., non-smooth regularization) and extra constraints on the linear combination coefficients. Moreover, we can still run FATSA when $t$ reaches $T$ until the objective function value of (4) does not decrease sufficiently. For GDM, it is impossible to do so because the dimension of coefficients after temporal discretization is fixed. Although there are more time steps, FATSA is still much faster than GDM (see Table 4).

### 4.3. Handling vector-valued images

In practice, images in many vision tasks are often vector-valued, such as RGB color images, multi-spectral satellite images and multimodal medical images. FATSA can also work for learning PDEs for vector-valued images. In this case, the optimization problem at time $t_n$ is

$$\min_{\mathbf{a}^n, \mathbf{b}^{n-1}} L^{n+1} = \frac{1}{2} \sum_{m=1}^{M} \sum_{c=1}^{C} \int_{\Omega} \left[ O_m^c - u_m^c(t_{n+1}) \right]^2 \mathrm{d}\Omega,$$
$$\text{s.t.} \quad \|\mathbf{a}_c^n\|_\infty < \eta_1, c = 1, \ldots, C, \quad \text{and} \quad \|\mathbf{b}^{n-1}\|_\infty < \eta_2, \tag{20}$$

where $\mathbf{u}_m = \{u_m^c, c = 1, \ldots, C\} : \Omega \to \mathbb{R}^C$ are the evolutionary vector-valued images for input image $\mathbf{I}_m$ and expected output images $\mathbf{O}_m = \{O_m^c, c = 1, \ldots, C\} : \Omega \to \mathbb{R}^C$. We simply take the luminance of the input image as the initial function of the indicator function $v$. So the PDEs system consists of $C+1$ evolutionary PDEs. Accordingly, there are much more fundamental differential invariants, which are invariant to translation and rotation. The set of such invariants up to second order is

$$\Big\{ 1, f_r, (\nabla f_r)^T \cdot \nabla f_s, (\nabla f_r)^T \cdot \mathbf{H}_{f_m} \cdot \nabla f_s, tr(\mathbf{H}_{f_r}), tr(\mathbf{H}_{f_r} \cdot \mathbf{H}_{f_s}) | f_r, f_s, f_m \in \{u^1, \ldots, u^C, v\} \Big\}.$$

If $C=3$, there are 69 fundamental differential invariants up to the second order. For more details, refer to [13,21,22]. Obviously, $\{\mathbf{a}_c | c = 1, \ldots, C\}$ are not coupled with each other. So we can still use alternating minimization to solve (20).

## 5. Numerical experiments

In this section, we first give more details on implementation. Then we conduct extensive experiments on various datasets to test the performance of the proposed FATSA. *Note that this paper targets on solving the PDE constrained optimal control problem better. So we mainly focus on comparing FATSA with GDM, rather than comparing the learnt PDEs with state-of-the-art methods for each vision task,* which has been shown in [13,21,22].

### 5.1. Implementation

To compute the spatial derivatives and integrations, we need to do spatial discretization. We use central difference to approximate the derivatives:

$$\begin{cases} \frac{\partial f}{\partial x} = \frac{f(x+1) - f(x-1)}{2}, \\ \frac{\partial^2 f}{\partial x^2} = f(x+1) - 2f(x) + f(x-1). \end{cases} \tag{21}$$

The discrete forms of $\frac{\partial f}{\partial y}$, $\frac{\partial^2 f}{\partial y}$ and $\frac{\partial^2 f}{\partial x \partial y}$ can be defined similarly. In addition, we discretize the integrations as

$$\int_{\Omega} f(x,y)\, \mathrm{d}\Omega = \frac{1}{|\Omega|} \sum_{(x,y) \in \Omega} f(x,y), \tag{22}$$

where $|\Omega|$ is the number of pixels in $\Omega$.

In the first two experiments, the numbers $M$ of training image pairs are both 60. For GDM, $\lambda_1 = \lambda_2 = 10^{-7}$, $T=1$ and $\Delta t = 0.05$. For FATSA, $\eta_1 = \eta_2 = 10$ and $\Delta t = 0.05$, but $T$ is not limited to 1. Rather, we run FATSA until the difference between successive objective function value is below $10^{-6}$ or the number of iteration reached 100 (see Algorithm 1).

### 5.2. Image deblurring

We first compare the characteristics of FATSA and GDM [13] with a fundamental vision problem: image deblurring. For this task, we take the Berkeley image database [26] and generate the input images by blurring images using a $5 \times 5$ Gaussian kernel

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

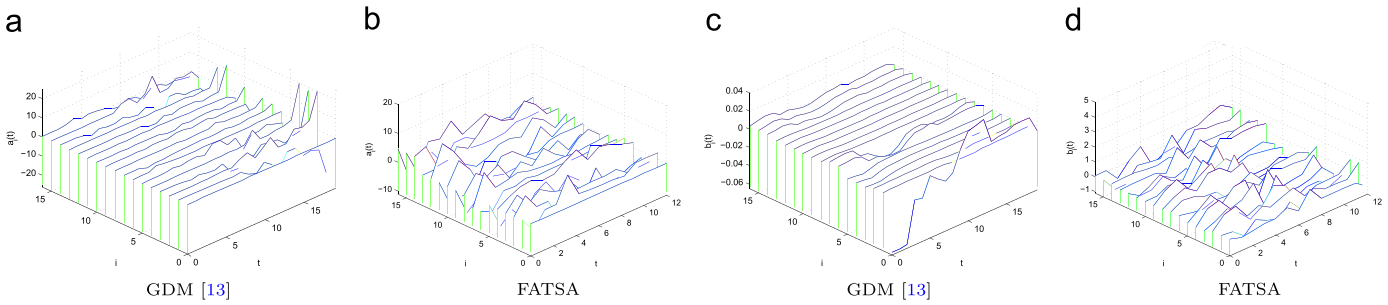


**Fig. 2.** The coefficients $a_i$'s ((a), (b)) and $b_i$'s ((c), (d)), $i=0,\ldots,16$, learnt of GDM [13] and FATSA for debluring.

with $\sigma=1$. The original images are used as the expected output images.

The coefficients learnt by GDM and FATSA for image debluring are shown in Fig. 2. We can observe that the coefficients $a_i$'s learnt by GDM (Fig. 2(a)) have a large variance in magnitude and some $a_i$'s increase abruptly to 20 at the last time step. This may cause numerical instability, which we indeed have encountered. In comparison, the coefficients $a_i$'s learnt by FATSA (Fig. 4(b)) are much more balanced, which makes the learnt PDEs more numerically stable. The coefficients $b_i$'s learnt by GDM (Fig. 4(c)) are very small (at a magnitude of $10^{-2}$), which implies that the evolution of indicator function is not effective. In comparison, the magnitudes of coefficients $b_i$'s learnt by FATSA (Fig. 4(d)) are much larger, which shows that the indicator function evolves effectively over time.

As for the performance of deblurring, the mean PSNR of FATSA over 200 test images is 32.60 dB, while that of GDM is 31.98 dB. Some deblurring results are shown in Fig. 3. We can see that FATSA restores more edges and textures. It is clear that the images processed by FATSA are sharper than those processed by GDM.

### 5.3. Natural image denoising

We then apply FATSA to learn PDEs for another fundamental vision problem: image denoising. For this task, we use the images tested in [13] with unknown natural noises. There are 240 images, each with a size of $150\times150$ pixels, of 11 objects taken by a Canon 30D digital camera, whose ISO is 1600. For each object, 30 images are taken without changing the camera settings (by fixing the focus, aperture and exposure time) and without moving the camera position. The average image of them can be regraded as the ground truth noiseless image, which serve as output training images.

We first compare the coefficients learnt by FATSA and those by GDM [13]. From Fig. 4, we can observe the similar phenomenon as in image deblurring. Most of the coefficients $a_i$'s learnt by GDM (Fig. 4(a)) are zeros and some jump to more than 20, which may result in numerical instability. The coefficients $a_i$'s learnt by FATSA (Fig. 4(b)) are much smoother. Their magnitudes are also much more balanced. Moreover, the coefficients $b_i$'s learnt by GDM (Fig. 4(c)) are very small (at the scale of $10^{-4}$), which implies that the indicator function actually does not change much over time. In comparison, the magnitudes of coefficients $b_i$'s learnt by FATSA (Fig. 4(d)) are much larger.

Next, we compare the denoising results on the test images between GDM [13] and FATSA. Some results are given in Fig. 5. We can see that both GDM and FATSA can remove noise effectively, but FATSA gives the sharper results. Moreover, the PSNRs of FATSA are higher than those of GDM.

### 5.4. Color image demosaicking

In this subsection, we consider a color image processing problem: image demosaicking. Demosaicking is to infer the two missing color components for every pixel from a raw image, which only captures one color component at every pixel due to the filtering of color filter arrays (CFAs) [27,28]. The most commonly used CFA is the Bayer CFA. For demosaicking, we test our method on both clean and noisy images. For the clean images demosaicking, we choose the Kodak image database [29]. For the noisy images demosaicking, we choose the publicly available noisy dataset [30].

#### 5.4.1. Demosaicking on the clean images

First, we take the Kodak image database [29] for clean images demosaicking. Images $1\sim12$ are used for training and images $13\sim24$ are used for testing. As for saving time and memory cost of training, we divide each $512\times768$ image into 12 non-overlapping $150\times150$ patches and choose the first 60 patches with the richest texture, which is measured by their variances. After that we downsample the patches into Bayer CFA raw image. Then we bilinearly interpolate the CFA raw data (i.e., for each channel the missing values are bilinearly interpolated from their nearest four available values) into full-color images and use them as the input images of the training pairs. Note that bilinear interpolation (BI) is the naivest way of inferring the missing colors and many artifacts can occur. We use the original full color clean images as the output images of the training pairs. We compare the results of FATSA with those of GDM [13] on the test images.

Fig. 6 shows some demosaciking results. We can see that FATSA is better than GDM in all PSNR value, noise reduction, and color fidelity. GDM [13] gives blurry results with many color artifacts, while FATSA is capable of eliminating most of the noises. Table 2 shows the comparison on the mean PSNRs of 12 test images. It can be seen that the mean PSNRs of FATSA are much higher than those of GDM.

#### 5.4.2. Demosaicking on the noisy images

Next, we want to show that FATSA can also work very well for noisy images demosaicking. For better comparison, we choose the publicly available Image Denoising Benchmark dataset [30], which is generated from the Berkeley Segmentation Database and images are degraded by additive, uncorrelated Gaussian noise with standard deviations (std) of 5, 10, 15, 25 and 35. We divide each $320\times480$ image into 6 non-overlapping $150\times150$ patches and select the first 60 patches with the richest texture, which is measured by their variances. Then we downsample the patches into Bayer CFA raw data and use bilinearly interpolated results as the input images of the training pairs.

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

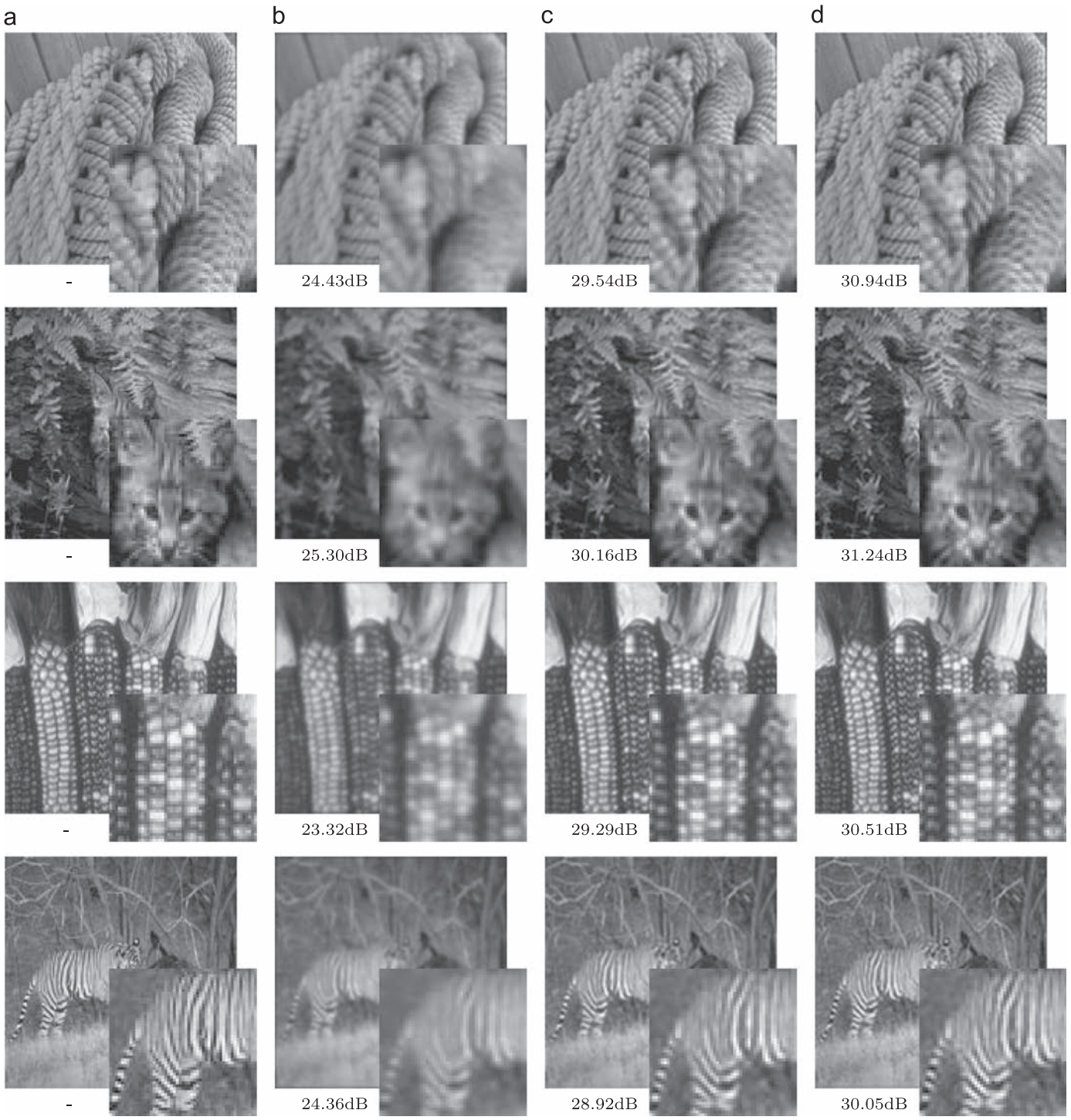


**Fig. 3.** The results of image deblurring. (a) Ground truth (GT) images. (b) The images blurred by a Gaussian kernel. (c, d) The deblurring results of GDM [13] and FATSA, respectively. The PSNRs are presented below each image.

Some results are given in Fig. 7. It shows the demosaicking results under different noise levels. We can see that the proposed FATSA can be used to fully recover Bayer CFA images with low noise levels and it preserves more texture information than GDM [13]. It is clear that the proposed FATSA gives sharper images and less color artifacts in high noise levels. Also, we compute the mean PSNRs on this dataset and show them in Table 3. We can see that the proposed FATSA performs better than GDM [13] in all noise levels.

### 5.5. Comparison of training error and time with GDM

Finally, we compare the training error and training time between GDM [13] and FATSA. The training error is measured by

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

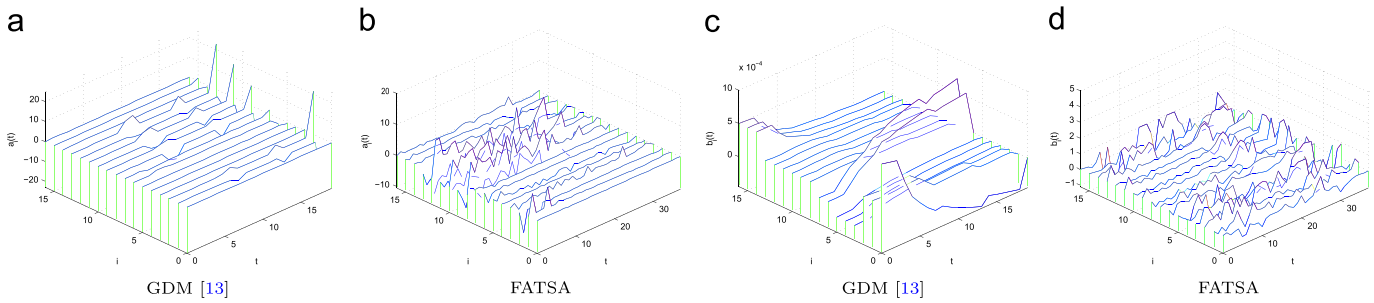


**Fig. 4.** The coefficients $a_i$'s ((a), (b)) and $b_i$'s ((c), (d)), $i=0,\ldots,16$, learnt by GDM [13] and FATSA in image denoising.

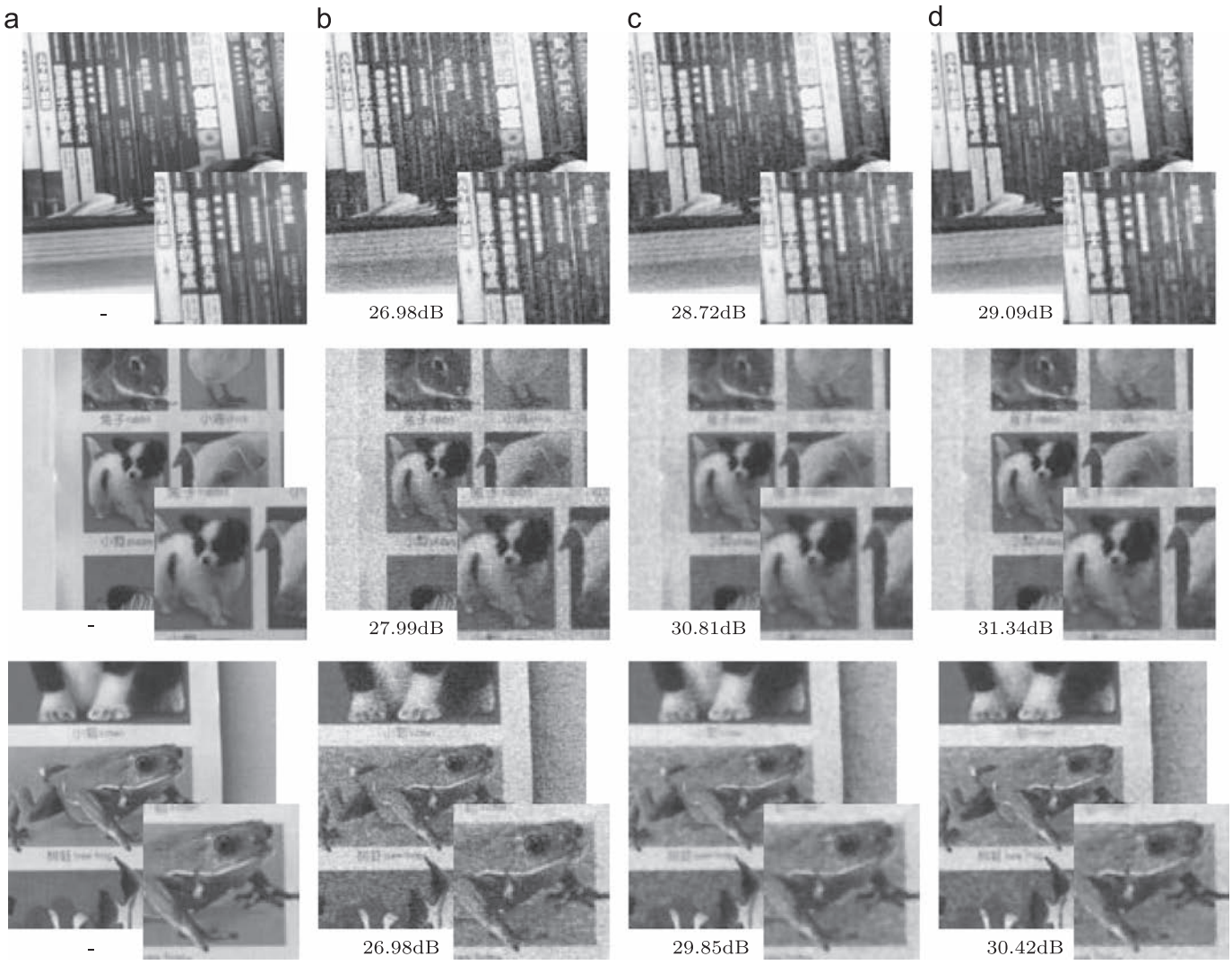


**Fig. 5.** The results of denoising images with natural noise. (a) Ground truth (GT) images. (b) Images with natural noise. (c, d) Denoised images using GDM [13] and the proposed FATSA, respectively. The PSNRs are presented below each image.

(5), where $M=60$ for image deblurring and noising and $M=12$ for image demosaicking. The Matlab implementation of GDM and FATSA are run on a PC equipped with a dual 3.4 GHz CPU and 8 GB memory.

The training error and time of GDM and FATSA are shown in Table 4. Obviously, the training time is greatly reduced by FATSA. For grayscale images, the training speed is accelerated by ten times. For color images, the training time is cut by more than half. Moreover, the training errors of FATSA are lower than those of GDM. So we can conclude that FATSA is much more effective than GDM on learning PDEs.

## 6. Conclusion

In this paper, we propose a new fast alternating time-splitting approach (FATSA) to efficiently learn PDEs for different visual tasks. We aim at minimizing the difference between the expected

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

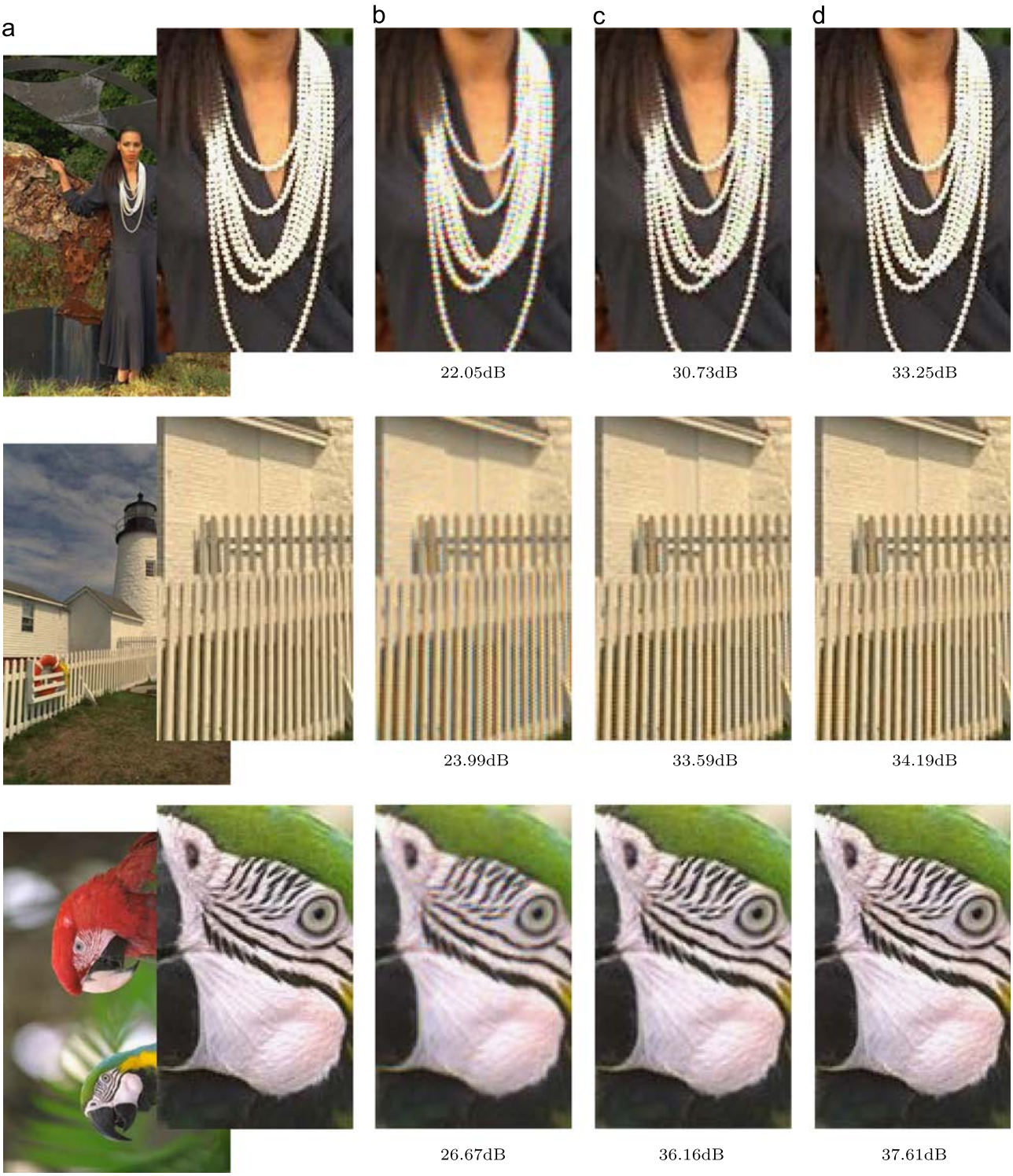


**Fig. 6.** The demosaicking results for clean images. (a) Ground truth (GT) images. (b–d) The demosaicking results of bilinearly interpolation (BI), GDM [13] and FATSA, respectively. The PSNRs are presented below each image.

output and the actual output of PDEs at every time step, rather than at the final step only as GDM does. FATSA is much faster than GDM and also much simpler as it does not require deducing and solving the adjoint equations. Moreover, it is more flexible than GDM in incorporating more complex regularizations or constraints on the linear combination coefficients. Experiments on typical image processing problems show that FATSA outperforms GDM.

In the future, we plan to improve and enrich our work in the following aspects. First, we want to carry out theoretical analysis on the convergence behavior of FATSA and the properties of learnt PDEs. Second, we would like to combine other regularizations and constraints to improve the performance of learnt PDEs. Finally,

**Table 2**
Comparison on the mean PSNRs on 12 test images in the Kodak image database.

| Input (BI) | GDM [13] | FATSA |
|---|---|---|
| 31.42 ± 5.58 dB | 37.96 ± 3.75 dB | **38.71** ± 3.61 dB |

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

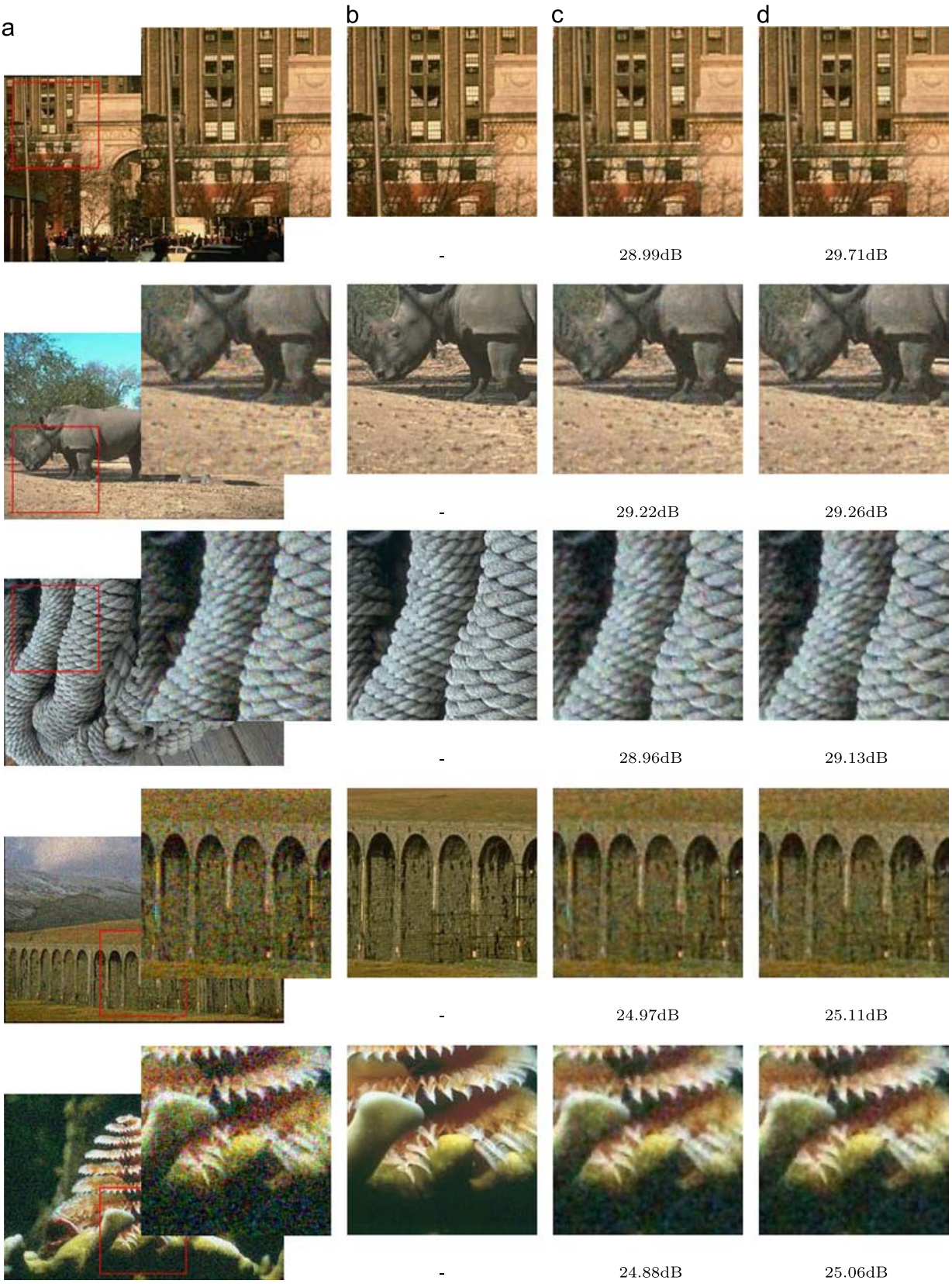


**Fig. 7.** The demosaicking results for noisy images (From the first row to the last row, images are degraded by Gaussian noise with standard deviations (std) of 5, 10, 15, 25 and 35, respectively). (a) Noisy images with different noise level. (b) Ground truth (GT) images. (c, d) The demosaicking results of GDM [13] and the proposed FATSA.

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126

**Table 3**
Comparison on the mean PSNRs for increasing noise level of different algorithms.

| Noise (std) | Input (BI) | GDM [13] | FATSA |
|---|---|---|---|
| 5 | 28.63 dB | 31.46 dB | **31.60** dB |
| 10 | 27.64 dB | 29.37 dB | **29.39** dB |
| 15 | 24.47 dB | 27.88 dB | **27.92** dB |
| 25 | 20.39 dB | 25.93 dB | **26.01** dB |
| 35 | 17.73 dB | 24.52 dB | **24.83** dB |

**Table 4**
Comparison of training error and time of GDM and FATSA.

| | Error | | Time (s) | |
|---|---|---|---|---|
| | GDM [13] | FATSA | GDM [13] | FATSA |
| Debluring | 1516 | **1180** | 4769 | **491** |
| Denoising | 1602 | **1521** | 4729 | **473** |
| Demosaicking, clean | 2630 | **2548** | 13349 | **6210** |
| Noisy, std = 5 | 4827 | **4649** | 10212 | **2869** |
| std = 10 | 7147 | **7035** | 11333 | **3499** |
| std = 15 | 9843 | **9539** | 10293 | **3759** |
| std = 25 | 15056 | **14923** | 13325 | **5336** |
| std = 35 | 21087 | **20261** | 12263 | **5021** |

beyond learning-based PDEs, we will apply FATSA to solve other PDE constrained optimal control problems in computer vision [20], such as optical flow estimation [12,31], tracking [32,33] and image sequence interpolation models [34,35].

## Acknowledgment

Zhenyu Zhao is supported by NSF China (Grant no. 61473302). Zhouchen Lin is supported by National Basic Research Program of China (973 Program) (Grant no. 2015CB352502), National Natural Science Foundation (NSF) of China (Grant nos. 61272341 and 61231002), Microsoft Research Asia Collaborative Research Program, and Okawa Foundation. The authors thank Jianlong Wu for helping on revising and proofreading the paper.

## References

[1] Gilles Aubert, Pierre Kornprobst, Mathematical Problems in Image Processing: Partial Differential Equations and the Calculus of Variations, vol. 147, Springer, New York, USA, 2006.
[2] Vicent Caselles, Jean-Michel Morel, Guillermo Sapiro, Allen R. Tannenbaum, Introduction to the special issue on partial differential equations and geometry-driven diffusion in image processing and analysis, IEEE Trans. Image Process. 6 (3) (1998) 269–273.
[3] Guillermo Sapiro, Geometric Partial Differential Equations and Image Analysis, Cambridge University Press, Cambridge, England, 2006.
[4] Pietro Perona, Jitendra Malik, Scale-space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intell. 12 (7) (1990) 629–639.
[5] Min Yang, Jingkun Liang, Jianhai Zhang, Haidong Gao, Fanyong Meng, Li Xingdong, Sung-Jin Song, Non-local means theory based Perona–Malik model for image denosing, Neurocomputing 120 (2013) 262–267.
[6] Stanley Osher, Leonid I. Rudin, Feature-oriented image enhancement using shock filters, SIAM J. Numer. Anal. 27 (4) (1990) 919–940.
[7] Faming Fang, Guixu Zhang, Fang Li, Chaomin Shen, Framelet based pan-sharpening via a variational method, Neurocomputing 129 (2014) 362–377.
[8] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, Coloma Ballester, Image inpainting, In: SIGGRAPH, 2000, pp. 417–424.
[9] Chunming Li, Chenyang Xu, Changfeng Gui, Martin D. Fox, Level set evolution without re-initialization: a new variational formulation, in: CVPR, vol. 1, 2005, pp. 430–436.
[10] Xuchu Wang, Jinxiao Shan, Yanmin Niu, Liwen Tan, Shao-Xiang Zhang, Enhanced distance regularization for re-initialization free level set evolution with application to image segmentation, Neurocomputing 141 (2014) 223–235.
[11] Steven S. Beauchemin, John L. Barron, The computation of optical flow, ACM Comput. Surv. 27 (3) (1995) 433–466.
[12] Qiao Cai, Yafeng Yin, Hong Man, DSPM: dynamic structure preserving map for action recognition, In: ICME, IEEE, San Jose, California, USA, 2013, pp. 1–6.
[13] Risheng Liu, Zhouchen Lin, Wei Zhang, Kewei Tang, Zhixun Su, Toward designing intelligent PDEs for computer vision: an optimal control approach, Image Vis. Comput. 31 (2013) 43–56.
[14] Burger Martin, C.G. Mennucci Andrea, Osher Stanley, Rumpf Martin, Level Set and PDE Based Reconstruction Methods in Imaging, Springer, Cetraro, Italy, 2013.
[15] Leonid I. Rudin, Stanley Osher, Emad Fatemi, Nonlinear total variation based noise removal algorithms, Physica D: Nonlinear Phenom. 60 (1) (1992) 259–268.
[16] Tony F. Chan, Selim Esedoglu, Aspects of total variation regularized $L^1$ function approximation, SIAM J. Appl. Math. 65 (5) (2005) 1817–1837.
[17] Nir Sochen, Ron Kimmel, Ravi Malladi, A general framework for low level vision, IEEE Trans. Image Process. 7 (3) (1998) 310–318.
[18] M. Ehrhardt, Simon R. Arridge, Vector-valued image processing by parallel level sets, IEEE Trans. Image Process. 23 (1) (2014) 9–18.
[19] Risheng Liu, Zhouchen Lin, Wei Zhang, Zhixun Su, Learning PDEs for image restoration via optimal control, in: ECCV, 2010.
[20] Kimia Benjamin, Tannenbaum Allen, Zucker Steven, On optimal control methods in computer vision and image processing, Geom.-Driven Diff. Comput. Vis. 1 (1994) 307–338.
[21] Zhouchen Lin, Wei Zhang, Xiaoou Tang, Learning Partial Differential Equations for Computer Vision, Technical Report, Microsoft Research, MSR-TR-2008-189, 2008.
[22] Zhouchen Lin, Wei Zhang, Xiaoou Tang, Designing Partial Differential Equations for Image Processing by Combining Differential Invariants, Technical Report, Microsoft Research, MSR-TR-2009-192, 2009.
[23] E. Zeidler, Nonlinear Functional Analysis and its Applications, Springer-Verlag, New York, USA, 1985.
[24] Thomas F. Coleman, Yuying Li, A reflective Newton method for minimizing a quadratic function subject to bounds on some of the variables, SIAM J. Optim. 6 (4) (1996) 1040–1058.
[25] Ake Björck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, Pennsylvania, USA, 1996.
[26] P. Arbelaez, C. Fowlkes, D. Martin, The Berkeley segmentation dataset [Online]. Available: ⟨http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/⟩, 2013.
[27] B.K. Gunturk, Y. Altunbasak, R.M. Mersereau, Color plane interpolation using alternating projections, IEEE Trans. Image Process. 11 (9) (2002) 997–1013.
[28] B. Leung, G. Jeon, E. Dubois, Least-squares luma-chroma demultiplexing algorithm for Bayer demosaicking, IEEE Trans. Image Process. 20 (7) (2011) 1885–1894.
[29] True color Kodak images [Online]. Available: ⟨http://r0k.us/graphics/kodak/⟩, 2013.
[30] F.J. Estrada, Image denoising benchmark [online]. Available: ⟨http://www.cs.utoronto.ca/~strider/Denoise/Benchmark/⟩, 2010.
[31] Nicolas Papadakis, Thomas Corpetti, Etienne Mémin, Dynamically consistent optical flow estimation, in: ICCV, 2007.
[32] Nicolas Papadakis, Etienne Mémin, Variational optimal control technique for the tracking of deformable objects, In: ICCV, 2007.
[33] Jianwu Fang, Qi Wang, Yuan Yuan, Multi-cue based tracking, Neurocomputing 131 (2014) 227–236.
[34] Kanglin Chen, Dirk A. Lorenz, Image sequence interpolation using optimal control, J. Math. Imaging Vis. 41 (3) (2011) 222–238.
[35] Kanglin Chen, Dirk A. Lorenz, Image sequence interpolation based on optical flow, segmentation, and optimal control, IEEE Trans. Image Process. 21 (3) (2012) 1020–1030.

**Zhenyu Zhao** received the B.S. degree in mathematics from University of Science and Technology in 2009, and the M.S. degree in system science from National University of Defense and Technology in 2011. He is currently pursuing the Ph.D. degree in applied mathematics, National University of Defense and Technology. His research interests include computer vision, pattern recognition and machine learning.

**Zhouchen Lin** received the Ph.D. degree in applied mathematics from Peking University in 2000. Currently, he is a professor at the Key Laboratory of Machine Perception (MOE), School of Electronics Engineering and Computer Science, Peking University. He is also a chair professor at Northeast Normal University. He was a guest professor at Shanghai Jiaotong University, Beijing Jiaotong University, and Southeast University. He was also a guest researcher at the Institute of Computing Technology, Chinese Academic of Sciences. His research interests include computer vision, image processing, machine learning, pattern recognition, and numerical optimization. He is an area chair of CVPR 2014, ICCV

2015, NIPS 2015, AAAI 2016, CVPR 2016 and IJCAI 2016. He is an associate editor of IEEE T. Pattern Analysis and Machine Intelligence and International J. Computer Vision and a senior member of the IEEE.

**Yi Wu** is a professor in the College of Science at the National University of Defense Technology in Changsha, China. He earned bachelor's and master's degrees in applied mathematics at the National University of Defense Technology in 1981 and 1988. He worked as a visiting researcher at New York State University in 1999. His research interests include applied mathematics, statistics, and data processing.

Please cite this article as: Z. Zhao, et al., A fast alternating time-splitting approach for learning partial differential equations, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2015.10.126