# Fast Total-Variation Based Image Restoration Based on Derivative Alternated Direction Optimization Methods

Dongwei Ren[a], Hongzhi Zhang[a], David Zhang[b], Wangmeng Zuo[a]

[a] *School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China*

[b] *Department of Computing, The Hong Kong Polytechnic University, Kowloon, Hong Kong*

**Abstract**

The total variation (TV) model is one of the most successful methods for image restoration, as well as an ideal bed to develop optimization algorithms for solving sparse representation problems. Previous studies showed that derivative space formulation of the image restoration model is useful in improving the success rate in image recovery and kernel estimation performance in blind deconvolution. However, little attentions are paid on the model and algorithm for derivative space based image restoration. In this paper, we study the TV based image restoration (TVIR) by developing a novel derivative space - based reformulation together with an efficient derivative alternating direction method of multipliers (D-ADMM) algorithm. Thanks to the simplicity of the proposed derivative space reformulation, D-ADMM only requires four fast Fourier transform (FFT) operations per iteration, and is much efficient than the other augmented Lagrangian methods. Numerical experiments show that, D-ADMM can obtain satisfactory restoration result and is much faster than the state-of-the-art TVIR algorithms.

*Keywords:* total variation, image restoration, convex optimization, alternating direction method of multipliers, augmented Lagrangian method

## 1. Introduction

In many image processing applications, the image in hand is only a degraded observation $\mathbf{y}$ of the original image $\mathbf{x}$. In the linear degradation model, the procedure can be modeled as

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}, \tag{1}$$

where $\mathbf{A}$ is a linear operator and $\mathbf{e}$ is the additive Gaussian white noise (AGWN). Image restoration aims to estimate the clear image $\mathbf{x}$ from its degraded observation $\mathbf{y}$, and is well known as a typical linear inverse problem [1].

Since the linear operator $\mathbf{A}$ usually is ill-conditioned, the recovery of $\mathbf{x}$ from $\mathbf{y}$ is an ill-posed problem, and a typical image restoration models generally includes a fidelity term and a regularization term, resulting in the following optimization problem,

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2 + \tau R(\mathbf{x}), \tag{2}$$

where $R(\mathbf{x})$ is some regularizer on $\mathbf{x}$, and $\tau$ is the regularization parameter. By far, based on various models on image prior, a number of regularizers, e.g., total variation (TV) [2], gradient-based [3], wavelet-based [4], dictionary-based sparsity [5, 6, 7], and non-local models [8, 9, 10], have been developed for image restoration. Due to its simplicity and ability to preserve edges, the TV regularizer has been widely applied to various image restoration and recovery tasks, e.g., denoising [11, 12], deconvolution [13, 14, 15], and compressed sensing (CS) [16, 17].

The TV model is also an ideal bed to develop optimization algorithms for solving sparse representation problems. By far, a number of methods have been developed for TVIR. These algorithms, including split-Bregman [18], accelerated proximal gradient [19, 20, 21], and alternating direction method of multipliers [22, 23, 24], were applied to TVIR, and then were adopted for other image processing, computer vision, and machine learning tasks [3, 25, 26, 27, 28, 29].

### *1.1. Related work*

The TVIR model can be formulated as,

$$\min_{\mathbf{x}} F(\mathbf{x}) = \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2 + \tau \|\mathbf{Dx}\|, \tag{3}$$

where $\mathbf{D} = [\mathbf{D}_h^T, \mathbf{D}_v^T]^T$ is the discrete gradient operator, $\|\cdot\|$ denotes the norm in the gradient space (including both anisotropic and isotropic versions, and please refer to Section 2.1 for detailed definitions of TV regularizers), and $\tau$ is the regularization parameter.

The augmented Lagrangian methods (ALM) are one class of the most efficient among various TVIR algorithms. Because of the non-smoothness of the TV regularizer, variable splitting strategies usually are required in the ALM-based algorithms. By far, there are mainly two variable splitting strategies for ALM-based TVIR. In [23], an auxiliary variable $\mathbf{u}$ was introduced to substitute the variable $\mathbf{x}$ in the fidelity term, resulting in the following equality-constrained optimization problem,

$$\min_{\mathbf{x},\mathbf{u}} \frac{1}{2} \|\mathbf{A}\mathbf{u} - \mathbf{y}\|_2^2 + \tau \|\mathbf{D}\mathbf{x}\| \;\; s.t. \; \mathbf{u} = \mathbf{x}. \tag{4}$$

By incorporating with the efficient TV-based denoising algorithm [12], Afonso et al. developed a split augmented Lagrangian shrinkage algorithm (SALSA) for TVIR. In [22, 30], another variable splitting strategy was adopted in FTVd by introducing an auxiliary variable $\mathbf{d}$ to substitute $\mathbf{D}\mathbf{x}$ in the regularizer, resulting in the following equivalent formulation,

$$\min_{\mathbf{x},\mathbf{d}} \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \tau \|\mathbf{d}\| \;\; s.t. \; \mathbf{d} = \mathbf{D}\mathbf{x}, \tag{5}$$

where $\mathbf{d} = [\mathbf{d}_h^T, \mathbf{d}_v^T]^T$ with $\mathbf{d}_h = \mathbf{D}_h\mathbf{x}$, $\mathbf{d}_v = \mathbf{D}_v\mathbf{x}$. With this formulation, each subproblem of the ALM algorithms can be efficiently solved, making FTVd the state-of-the-art TVIR methods in term of computational efficiency.

Recently, derivative space formulation of the image restoration model had received considerable research interests and shown several unique advantages in compressed sensing [31, 32], image restoration [33], and blind deconvolution [34]. In compressed sensing, Patel et al. [31] proposed a GradientRec approach which first used the compressed sensing (CS) algorithm to recover the gradient images and then reconstructed the original image from the gradient images. Because the gradient images are much more sparse, it had been shown in [31] that derivative space based GradientRec could obtain higher success rate in image recovery.

In image restoration, Michailovich [33] also introduced a variable $\mathbf{d}$ to substitute $\mathbf{D}\mathbf{x}$ in the regularizer. By assuming that the image $\mathbf{x}$ has zero mean value, a left inverse operator $\mathcal{U}(\mathbf{d})$ [35] can be employed to recover the original image from the derivative space, i.e., $\mathbf{x} = \mathcal{U}(\mathbf{d})$. Thus, the variable $\mathbf{x}$ can be removed from the model in Eq. (3), and TVIR can be formulated in the derivative space,

$$\min_{\mathbf{d}} \frac{1}{2} \|\mathbf{A}\mathcal{U}\{\mathbf{d}\} - \mathbf{y}\|^2 + \tau \|\mathbf{d}\|. \tag{6}$$

Michailovich [33] proposed a TV-based iterative shrinkage (TVIS) algorithm for solving the model in Eq. (6).

In blind deconvolution, recent studies showed that, better kernel estimation performance can generally be obtained in the derivative space than in the image space [34, 36, 37, 38]. Cho and Lee [39] analyzed the condition numbers of the Hessians which indicated that the Hessian in the derivative space has a diagonally dominant structure and has a much smaller condition numbers than that in the image space.

Although previous studies had indicated the advantages of derivative space formulation, little attentions are paid on the proper modeling and efficient algorithms for derivative space based image restoration. For example, GradientRec only greedily solved the TV based CS problem and cannot guarantee the convergence to the solution of the original problem. The convergence rate of TVIS is $O\left(t^{-1}\right)$, which is much slower than the state-of-the-art TVIR algorithms.

In this paper, we study the derivative space TVIR problem by proposing a novel derivative space - based reformulation together with an efficient derivative alternating direction method of multipliers (D-ADMM) algorithm. This work is an extension of [40], based on which we deduce an explicit formulation of TVIR in the derivative space and propose two ADMM-based algorithms to solve it efficiently. First, by analyzing the connections of image space and derivative space, we introduce an explicit equality constraint on the gradients $\mathbf{d}$, and suggest a novel derivative space based reformulation of TVIR. Compared with the formulation in [33], the proposed formulation is more concise and much easier to be solved. Then, we adopt the alternating direction method of multipliers (ADMM) algorithm to solve the constrained optimization problem, resulting in the proposed derivative-space ADMM (D-ADMM) algorithm. D-ADMM only requires four fast Fourier transform (FFT) operations per iteration, and is much more efficient than the other TVIR methods. Finally, experimental results show that, D-ADMM can obtain satisfactory restoration results and is much faster than the state-of-the-art TVIR algorithms, e.g., FTVd and SALSA.

### *1.2. Organization*

This paper is organized as follows. Section 2 introduces some background knowledge related to this paper. Section 3 presents the derivative space based reformulation of TVIR, and Section 4 describes the proposed D-ADMM algorithms. Section 5 provides the experimental results by comparing D-ADMM with the state-of-the-art methods. Finally, Section 5 ends this paper with some concluding remarks.

## 2. Preliminaries

In this section, we first introduce the discrete TV operators with periodic boundary conditions, then summarize the related proximal operators used in this paper, and finally, briefly review the ADMM algorithm.

### *2.1. The discrete TV operators*

Analogous to [33], we assume that the image $\mathbf{x}$ should lie in the $\mathbb{R}^{m\times n}$ space $\mathbb{U}$ with zero mean value, i.e., $\mathbb{U} = \{\mathbf{x} \in \mathbb{R}^{m\times n} | mean\,(\mathbf{x}) = 0\}$. With the assumption of periodic boundary conditions, the gradient operator $\mathcal{D}$, also notated as $\nabla$, is defined as

$$\begin{array}{l} (\mathcal{D}_h \mathbf{x})_{k,l} = \mathbf{x}_{k,l} - \mathbf{x}_{k,l-1}, \text{with } \mathbf{x}_{k,-1} = \mathbf{x}_{k,n-1} \\ (\mathcal{D}_v \mathbf{x})_{k,l} = \mathbf{x}_{k,l} - \mathbf{x}_{k-1,l}, \text{with } \mathbf{x}_{-1,l} = \mathbf{x}_{m-1,l} \end{array}, \tag{7}$$

where $k = 0, 1, 2, \ldots, m-1$ and $l = 0, 1, 2, \ldots, n-1$. Thus, the anisotropic TV [19, 21] is defined as,

$$\mathrm{TV}_a(\mathbf{x}) = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} \left( \left| (\mathcal{D}_h \mathbf{x})_{k,l} \right| + \left| (\mathcal{D}_v \mathbf{x})_{k,l} \right| \right). \tag{8}$$

The isotropic TV [19, 21] is defined by

$$\mathrm{TV}_i(\mathbf{x}) = \sum_{k=0}^{m-1} \sum_{l=0}^{n-1} \sqrt{(\mathcal{D}_h \mathbf{x})_{k,l}^2 + (\mathcal{D}_v \mathbf{x})_{k,l}^2}. \tag{9}$$

The adjoint operators $\mathcal{D}_h^*$ and $\mathcal{D}_v^*$ of $\mathcal{D}_h$ and $\mathcal{D}_v$ can be defined by

$$\begin{array}{l} (\mathcal{D}_h^* \mathbf{x})_{k,l} = \mathbf{x}_{k,l} - \mathbf{x}_{k,l+1}, \text{with } \mathbf{x}_{k,n} = \mathbf{x}_{k,0} \\ (\mathcal{D}_v^* \mathbf{x})_{k,l} = \mathbf{x}_{k,l} - \mathbf{x}_{k+1,l}, \text{with } \mathbf{x}_{m,l} = \mathbf{x}_{0,l} \end{array}, \tag{10}$$

respectively.

The images $\mathbf{x}$, $\mathbf{d}_h$ and $\mathbf{d}_v$ can be rearranged into the corresponding vectors, and vice versa. Thus, we use the same small bold notation to denote an image and its vectorization, and this should not cause ambiguity by referring to the context. Then the gradient operators $\mathcal{D}_h$ and $\mathcal{D}_v$ can be written as matrices $\mathbf{D}_h$ and $\mathbf{D}_v$ with $\mathbf{d}_h = \mathbf{D}_h\mathbf{x}$ and $\mathbf{d}_v = \mathbf{D}_v\mathbf{x}$, respectively, The corresponding adjoint operators $\mathcal{D}_h^*$ and $\mathcal{D}_v^*$ are associated with matrices $\mathbf{D}_h^T$ and $\mathbf{D}_v^T$, respectively.

### 2.2. Related proximal operators

Given a (nonsmooth) convex function $g(\mathbf{x})$ and a vector $\mathbf{z}$, the proximal operator with parameter $\lambda$ of $g$ is the function $\mathbf{prox}_{\lambda g}$ defined as,

$$\mathbf{prox}_{\lambda g}(\mathbf{z}) = \arg\min_{\mathbf{x}} \frac{1}{2}\left\|\mathbf{x}-\mathbf{z}\right\|^2 + \lambda g\left(\mathbf{x}\right). \tag{11}$$

Proximal operator is the basic ingredient of the proximal algorithms which can find many applications in optimization [41]. In the following, we only introduce several proximal operators used in this paper.

When g is the $l_1$ norm, i.e., $g(\mathbf{x}) = \left\|\mathbf{x}\right\|_{\mathbf{1}}$, the solution to Eq. (11) is

$$\mathbf{x} = \mathcal{T}_\lambda\left(\mathbf{z}\right), \tag{12}$$

where $\mathcal{T}_\lambda(z_i) = sgn(z_i)max(|z_i| - \lambda, 0)$ is the soft-thresholding operator [42] which is applied to elementary entry of $\mathbf{z}$.

When $g$ is the $l_{2,1}$ norm, i.e., $g(\mathbf{x}) = \left\|\mathbf{x}\right\|_{2,1} = \sum_i \left\|\mathbf{x}_i\right\|_2$ , then $i$-th column solution is

$$\mathbf{x}_i = \mathcal{G}_\lambda\left(\mathbf{z}_i\right) = \mathcal{T}_\lambda\left(\left\|\mathbf{z}_i\right\|_2\right)\frac{\mathbf{z}_i}{\left\|\mathbf{z}_i\right\|_2}, \tag{13}$$

where $\mathcal{G}_\lambda\left(\mathbf{z}\right)$ denotes the group shrinkage operator.

### 2.3. The left inverse operator

We introduce an operator to estimate image from its gradient vectors. For any image $\mathbf{x}$ in the space $\mathbb{U}$, it is straightforward to show that,

$$\mathcal{F}\left(\operatorname{div}\left(\nabla\mathbf{x}\right)\right) = \mathcal{F}\left(-\mathbf{D}_h^T\mathbf{D}_h - \mathbf{D}_v^T\mathbf{D}_v\right) \odot \mathcal{F}\left(\mathbf{x}\right), \tag{14}$$

where $\mathcal{F}$ is the Fourier transform, $\odot$ denotes the entry-wise product, and div is the divergence defined as,

$$\operatorname{div}\left(\mathbf{d}\right) = -\left(\mathbf{D}_h^T\mathbf{d}_h + \mathbf{D}_v^T\mathbf{d}_v\right), \tag{15}$$

We hereby use $\mathbf{W}$ to notate the Fourier transform of divergence in Eq. (14), and by the definition of Fourier transformation of gradient operators, $\mathbf{W}$ can be defined by,

$$\begin{array}{l} \mathbf{W}_{k,l} = \left(e^{j2\pi k/m} - 1\right)\left(1 - e^{-j2\pi k/m}\right) + \left(e^{j2\pi l/n} - 1\right)\left(1 - e^{-j2\pi l/n}\right) \\ \quad\quad = 2\cos\left(2\pi k/m\right) + 2\cos\left(2\pi l/n\right) - 4 \end{array}, \tag{16}$$

where $k = 0, 1, 2, \ldots, m-1$ and $l = 0, 1, 2, \ldots, n-1$. It is easy to point out that,

$$\mathbf{x} = \mathcal{U}(\mathbf{d}) = \mathcal{F}^{-1}\left(\mathcal{F}\left(\operatorname{div}(\mathbf{d})\right) \odot \mathbf{W}i\right), \tag{17}$$

where

$$(\mathbf{W}i)_{k,l} = (2\cos(2\pi k/m) + 2\cos(2\pi l/n) - 4)^{-1}, \tag{18}$$

for any $k + l > 0$, and $\mathbf{W}i(0,0)$ corresponds to that the mean value of $\mathbf{x}$ is 0.

The left inverse operator $\mathbf{x} = \mathcal{U}(\mathbf{d})$ actually requires $\mathbf{d}$ lying in the subspace of curl-free vector fields $\mathbb{V}$ [35]. Thus, given an arbitrary vector field $\mathbf{d} \in (\mathbb{R}^{m\times n})^2$, we need to first project it onto $\mathbb{V}$[35, 33].

### *2.4. ADMM*

Consider a general optimization problem,

$$\min_{\mathbf{u},\mathbf{v}} f(\mathbf{u}) + g(\mathbf{v}) \quad s.t. \quad \mathbf{Gu} + \mathbf{Hv} = \mathbf{c}, \tag{19}$$

where $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{G} \in \mathbb{R}^{d\times n}$, $\mathbf{H} \in \mathbb{R}^{d\times m}$ and $\mathbf{c} \in \mathbb{R}^d$.

In the augmented Lagrangian method (ALM), the augmented Lagrangian (AL) function of the problem in Eq. (19) is defined as,

$$\mathcal{L}(\mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}, \delta) = f(\mathbf{u}) + g(\mathbf{v}) + \boldsymbol{\lambda}^T(\mathbf{Gu} + \mathbf{Hv} - \mathbf{c}) + \frac{\delta}{2}\left\|\mathbf{Gu} + \mathbf{v} - \mathbf{c}\right\|^2, \tag{20}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^d$ is a vector of Lagrangian multiplier and $\delta > 0$ is the AL penalty parameter. For more concise expression, with minor algebra, the AL function can be rewritten as

$$\mathcal{L}(\mathbf{u}, \mathbf{v}, \mathbf{p}, \delta) = f(\mathbf{u}) + g(\mathbf{v}) + \frac{\delta}{2}\left\|\mathbf{Gu} + \mathbf{Hv} - \mathbf{c} + \mathbf{p}\right\|^2, \tag{21}$$

where $\mathbf{p} = \boldsymbol{\lambda}/\delta$.

For most problems, it is extremely difficult to solve $\mathbf{u}$ and $\mathbf{v}$ simultaneously. A natural strategy is to alternatively update $\mathbf{u}$ and $\mathbf{v}$, resulting in the alternating direction method of multipliers (ADMM) algorithm summarized in Algorithm 1. In conventional ADMM [23, 24, 43], the penalty parameter $\delta$ is fixed. In order to accelerate the convergence speed, Lin et al. [44] proposed an adaptive updating strategy for penalty parameter,

$$\delta^{(t+1)} = \min\left(\delta_{\max}, \rho\delta^{(t)}\right), \tag{22}$$

```
Algorithm 1: ADMM
1. Initilize $\mathbf{u}^{(0)}$, $\mathbf{v}^{(0)}$, $\mathbf{p}^{(0)}$, $t = 0$
2. while not converged
3.     $\mathbf{u}^{(t+1)} = \arg\min_{\mathbf{u}} \left\{ f(\mathbf{u}) + \frac{\delta^{(t)}}{2} \left\| \mathbf{G}\mathbf{u} + \mathbf{H}\mathbf{v}^{(t)} - \mathbf{c} + \mathbf{p}^{(t)} \right\|_2^2 \right\}$
4.     $\mathbf{v}^{(t+1)} = \arg\min_{\mathbf{v}} \left\{ g(\mathbf{v}) + \frac{\delta^{(t)}}{2} \left\| \mathbf{G}\mathbf{u}^{(t+1)} + \mathbf{H}\mathbf{v} - \mathbf{c} + \mathbf{p}^{(t)} \right\|_2^2 \right\}$
5.     $\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} + \left( \mathbf{G}\mathbf{u}^{(t+1)} + \mathbf{H}\mathbf{v}^{(t+1)} - \mathbf{c} \right)$
6.     Update $\delta^{(t)}$ to $\delta^{(t+1)}$
7.     $t = t + 1$
8. end while
```

with

$$\rho = \left\{ \begin{array}{l} \rho_0, \text{if } \delta \left\| \mathbf{H}\mathbf{v}^{(t+1)} - \mathbf{H}\mathbf{v}^{(t)} \right\| / \left\| \mathbf{G}\mathbf{u}^{(t+1)} \right\| < \varepsilon \\ 1, \text{ otherwise} \end{array} \right. . \tag{23}$$

where $\delta_{\max}$ is the upper bound of $\delta$ and $\rho_0$ is a positive scalar.

The convergence of ADMM has been presented in [45], where the Theorem 1 is proved.

**Theorem 1.** *If* (19) *satisfies the two assumptions,*

*(1) The functions* $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ *and* $g : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ *are closed, proper and convex,*

*(2) The unaugmented Lagrangian function of* (19) *has a saddle point, the ADMM iterates converge to the optimal solution of* (19).

## 3. Problem Formulation

In this section, we first present a novel derivative-based reformulation of TVIR by analyzing the connections of image space and derivative space, and then validate the rationale of the deduction of the proposed derivative space-based reformulation via numerical experiments.

### *3.1. Derivative space-based reformulation of TVIR*

Let's denote that a random variable $\mathbf{u} \in \mathbb{R}^{N\times 1}$ has a Gaussian distribution with the mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$ i.e., $\mathbf{u} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Given a matrix $\mathbf{L} \in \mathbb{R}^{M\times N}$, the random variable would have a Gaussian distribution with the mean $\mathbf{L}\boldsymbol{\mu}$ and the covariance $\mathbf{L}\boldsymbol{\Sigma}\mathbf{L}^T$, i.e., $\mathbf{L}\mathbf{u} \sim \mathcal{N}(\mathbf{L}\boldsymbol{\mu}, \mathbf{L}\boldsymbol{\Sigma}\mathbf{L}^T)$ [46].

Based on the degradation model defined in Eq. (1), one can easily see that the additive noise $\mathbf{e} = \mathbf{Ax} - \mathbf{y}$ has a Gaussian distribution, i.e., $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$. Thus, with the derivative operator $\mathbf{D}$, the random variable $\mathbf{De}$ has the Gaussian distribution $\mathbf{De} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{DD}^T)$. Based on the definition of covariance matrix, we can obtain the following,

$$E(\|\mathbf{e}\|_2^2) = mn\sigma^2, \tag{24}$$

$$E\left(\|\mathbf{De}\|_2^2\right) = \sigma^2 tr(\mathbf{DD}^T) = 4mn\sigma^2, \tag{25}$$

where $tr(\cdot)$ denotes the trace of a matrix. Note that $\mathbf{e}$ is an $m \times n$ image and each pixel of $\mathbf{e}$ is independent identically distributed (i.i.d.). When the image size is sufficiently high, it is natural to have

$$mn\sigma^2 = E(\|\mathbf{e}\|_2^2) = \sum\nolimits_{k,l} E\left(\|e_{k,l}\|_2^2\right) \approx \|\mathbf{e}\|_2^2, \tag{26}$$

$$4mn\sigma^2 = E(\|\mathbf{De}\|_2^2) = \sum\nolimits_{k,l} E\left(\left\|(\mathbf{De})_{k,l}\right\|_2^2\right) \approx \|\mathbf{De}\|_2^2. \tag{27}$$

Because of the inter-change property of convolution operators, $\|\mathbf{D}(\mathbf{Ax} - \mathbf{y})\|_2^2$ can be rewritten as $\|\mathbf{ADx} - \mathbf{Dy}\|_2^2$ . From Eqns. (26) and (27), we have

$$\|\mathbf{D}(\mathbf{Ax} - \mathbf{y})\|_2^2 = \|\mathbf{ADx} - \mathbf{Dy}\|_2^2 \approx 4\|\mathbf{Ax} - \mathbf{y}\|_2^2. \tag{28}$$

By replacing $\mathbf{Dx}$ with $\mathbf{d}$ and restricting $\mathbf{d} \in \mathbb{V}$, TVIR can be reformulated in the derivative space,

$$\begin{array}{c} \hat{\mathbf{d}} = \arg\min\limits_{\mathbf{d}} F(\mathbf{d}) = \frac{1}{2}\|\mathbf{Ad} - \mathbf{Dy}\|_2^2 + \mu\|\mathbf{d}\| \\ s.t. \quad \mathbf{d} \in \mathbb{V} \end{array}, \tag{29}$$

where we set $\mu = 4\tau$ based on Eq. (29).

Based on the definition of curl [47], the constraint $\mathbf{d} \in \mathbb{V}$ can be explicitly expressed as $\mathbf{D}_h^T \mathbf{d}_v = \mathbf{D}_v^T \mathbf{d}_h$. Thus, we use the notation $\mathbf{D}_c^T = \left[\mathbf{D}_v^T, -\mathbf{D}_h^T\right]$ and rewrite the equality constraint in more concise expression, i.e., $\mathbf{D}_c^T \mathbf{d} = \mathbf{0}$. The proposed derivative space-based reformulation of TVIR can then be written as,

$$\begin{array}{c} \hat{\mathbf{d}} = \arg\min\limits_{\mathbf{d}} \frac{1}{2}\|\mathbf{Ad} - \mathbf{Dy}\|_2^2 + \mu\|\mathbf{d}\| \\ s.t. \quad \mathbf{D}_c^T \mathbf{d} = \mathbf{0} \end{array}. \tag{30}$$

Compared with the formulation in [33], the proposed reformulation is much simpler and easier to be solved.

### *3.2. Numerical validation*

The noise $\mathbf{e} = \mathbf{Ax} - \mathbf{y}$ can be regarded as a set of $mn$ i.i.d. samples. Since the image size is high, it is reasonable to assume the approximation $\|\mathbf{ADx} - \mathbf{Dy}\|^2 \approx 4\|\mathbf{Ax} - \mathbf{y}\|^2$ would hold. To verify rationality of the assumption and the influence of image size, we use a set of natural images with different sizes, convolve them with different blur kernels, and further add different degrees of additive Gaussian white noise on the blurred images. For each noisy and blurred image, we calculate the ratio $\gamma = \|\mathbf{ADx} - \mathbf{Dy}\|^2 / \|\mathbf{Ax} - \mathbf{y}\|^2$ and then evaluate the distribution of $\gamma$ with respect to the image size.

**Table 1:** Statistical parameters of the $\gamma$ values

| Image size | mean/max/min | *std.* |
|---|---|---|
| 128 × 128 | 4.000/4.093/3.930 | 0.0221 |
| 256 × 256 | 3.999/4.036/3.960 | 0.0110 |
| 256 × 512 | 4.000/4.025/3.982 | 0.0056 |
| 1024 × 1024 | 4.000/4.009/3.991 | 0.0028 |

In our experiments, the distributions of $\gamma$ are evaluated with respect to four image sizes, i.e., 128 × 128, 256 × 256, 512 × 512, 1024 × 1024. For each image size, we adopt 20 natural images, six blur kernels (including four Gaussian blur kernels with = 3, 5, 7 and three camera motion kernels used in [38]), and ten degrees of Gaussian noise with the standard deviation ranged from 0.001 to 0.1. Thus, for each image size, we can obtain $20 \times 6 \times 10 = 1200$ $\gamma$ values.

Table 1 lists the mean, maximum, and minimum values, and the standard deviation of the $\gamma$ values with respect to each image size. In Fig. 1, we present the histograms of the $\gamma$ values for each image size, where red lines stand for the range of $[\mu - \sigma, \mu + \sigma]$. From Table 1 and Fig. 1, it is reasonable to assume $\gamma \approx 4$. Moreover, with the increasing of image size, the deviation of the $\gamma$ values to 4 would decrease. Thus, the problem formulation in Eq. (30) is really a good approximation of the original TVIR model, especially when the image size is sufficiently large.

## 4. The D-ADMM Algorithms

In this section, we propose two D-ADMM algorithms for solving the TVIR model in Eq. (30). First, we solve the problem in Eq. (30) using the

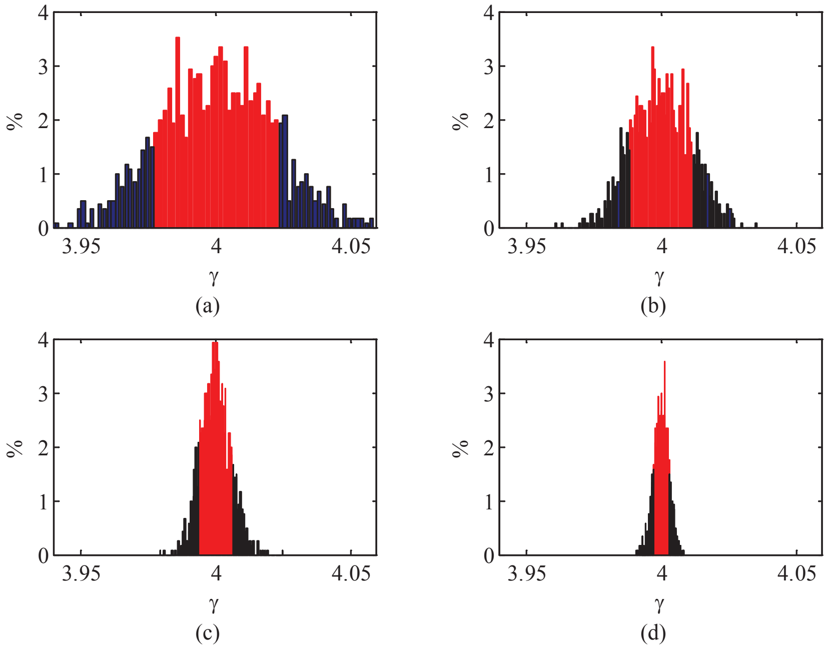

**Figure 1:** The histograms of the $\gamma$ values with respect to different image sizes: (a) 128×128, (b) 256×256, (c) 512×512, and (d) 1024×1024.

conventional ADMM algorithm, i.e., D-ADMM(C). Second, by combining both Lagrangian method and augmented Lagrangian method, we propose a hybrid ADMM algorithm, i.e., D-ADMM(H). which can further improve the efficiency of TVIR. Finally, we discuss some implementation issues of the proposed algorithms.

### *4.1. Conventional D-ADMM*

By introducing an auxiliary variable $\mathbf{f}$ with $\mathbf{f} = \mathbf{d}$, the problem in Eq. (30) can be equivalently formulated as,

$$\begin{array}{c} \mathbf{d} = \arg\min\limits_{\mathbf{d},\mathbf{f}} \frac{1}{2} \left\| \mathbf{A}\mathbf{d} - \mathbf{D}\mathbf{y} \right\|^2 + \mu \left\| \mathbf{f} \right\| \\ s.t. \quad \mathbf{D}_c^T \mathbf{d} = \mathbf{0}, \ \mathbf{d} = \mathbf{f} \end{array}. \tag{31}$$

The augmented Lagrangian function of Eq. (31) is defined as,

$$\begin{array}{l} \mathcal{L} = \mu \left\| \mathbf{f} \right\| + \frac{1}{2} \left\| \mathbf{A}\mathbf{d} - \mathbf{D}\mathbf{y} \right\|_2^2 + \frac{\delta_2}{2} \left\| \mathbf{d} - \mathbf{f} + \mathbf{q} \right\|^2 \\ \quad + \frac{\delta_1}{2} \left\| \mathbf{D}_h^T \mathbf{d}_v - \mathbf{D}_v^T \mathbf{d}_h + \mathbf{p} \right\|^2 \end{array}. \tag{32}$$

Like $\mathbf{d}$, either of $\mathbf{q}$, and $\mathbf{f}$ includes two parts, i.e., $\mathbf{q} = [\mathbf{q}_h^T, \mathbf{q}_v^T]^T$, and $\mathbf{f} = [\mathbf{f}_h^T, \mathbf{f}_v^T]^T$.

Given $\mathbf{f}$, $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{d}_v$, the $\mathbf{d}_h$ subproblem can be formulated as,

$$\begin{array}{c} \min\limits_{\mathbf{d}_h} \frac{1}{2} \left\| \mathbf{A}\mathbf{d}_h - \mathbf{D}_h\mathbf{y} \right\|^2 + \frac{\delta_2}{2} \left\| \mathbf{d}_h - \mathbf{f}_h + \mathbf{q}_h \right\|^2 \\ + \frac{\delta_1}{2} \left\| \mathbf{D}_h^T\mathbf{d}_v - \mathbf{D}_v^T\mathbf{d}_h + \mathbf{p} \right\|^2 \end{array}, \tag{33}$$

and the close-form solution of $\mathbf{d}_h$ is defined as,

$$\begin{array}{c} \mathbf{d}_h = \mathcal{F}^{-1} \left( \mathcal{F} \left( \mathbf{A}^T\mathbf{D}_h\mathbf{y} + \delta_1\mathbf{D}_v \left( \mathbf{D}_h^T\mathbf{d}_v + \mathbf{p} \right) \right. \right. \\ \left. \left. + \delta_2 \left( \mathbf{f}_h - \mathbf{q}_h \right) \right) \varnothing \mathcal{F}(\mathbf{B}_h) \right) \end{array}, \tag{34}$$

where $\mathbf{B}_h = \mathbf{A}^T\mathbf{A} + \delta_1\mathbf{D}_v\mathbf{D}_v^T + \delta_2\mathbf{I}$ and $\varnothing$ stands for the entry-wise division.

Similarly, given $\mathbf{f}$, $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{d}_h$, the close-form solution to $\mathbf{d}_v$ can be obtained by

$$\begin{array}{c} \mathbf{d}_v = \mathcal{F}^{-1} \left( \mathcal{F} \left( \mathbf{A}^T\mathbf{D}_v\mathbf{y} + \delta_1\mathbf{D}_h \left( \mathbf{D}_v^T\mathbf{d}_h - \mathbf{p} \right) \right. \right. \\ \left. \left. + \delta_2 \left( \mathbf{f}_v - \mathbf{q}_v \right) \right) \varnothing \mathcal{F}(\mathbf{B}_v) \right) \end{array}, \tag{35}$$

where $\mathbf{B}_v = \mathbf{A}^T\mathbf{A} + \delta_1\mathbf{D}_h\mathbf{D}_h^T + \delta_2\mathbf{I}$.

Given $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{d}$, the solutions to $\mathbf{f}$ can be obtained by solving the following subproblem,

$$\mathbf{f} = \arg\min_{\mathbf{f}} \frac{1}{2} \left\| \mathbf{f} - \mathbf{q} - \mathbf{d} \right\|_2^2 + \mu/\delta_2 \left\| \mathbf{f} \right\|, \tag{36}$$

For anisotropic TV, we have $\|\mathbf{f}\| = \|\mathbf{f}\|_1$, and we can use the soft-thresholding operator [42] introduced in Section 2.2 to update $\mathbf{f}$,

$$\mathbf{f} = \mathcal{T}_{\mu/\delta_2} \left( \mathbf{q} + \mathbf{d} \right). \tag{37}$$

For isotropic TV, we can simply let $(\mathbf{f})_{k,l} = [(\mathbf{f}_h)_{k,l}(\mathbf{f}_v)_{k,l}]^T$. Then $\mathbf{f}$ can be updated by using the group shrinkage operator introduced in Section 2.2,

$$(\mathbf{f})_{k,l} = \mathcal{G}_{\mu/\delta_2} \left( \left( \begin{array}{c} (\mathbf{q}_h)_{k,l} \\ (\mathbf{q}_v)_{k,l} \end{array} \right) + \left( \begin{array}{c} (\mathbf{d}_h)_{k,l} \\ (\mathbf{d}_v)_{k,l} \end{array} \right) \right). \tag{38}$$

Based on the ADMM algorithm, $\mathbf{p}$ and $\mathbf{q}$ can be updated as follows

$$\begin{array}{c} \mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} + \mathbf{D}_h^T\mathbf{d}_v^{(t+1)} - \mathbf{D}_v^T\mathbf{d}_h^{(t+1)} \\ \mathbf{q}^{(t+1)} = \mathbf{q}^{(t)} + \mathbf{d}^{(t+1)} - \mathbf{f}^{(t+1)} \end{array}. \tag{39}$$

For the updating of $\delta_1$ and $\delta_2$, we have

$$\begin{aligned} \delta_1^{(t+1)} &= \min\left(\delta_{\max}, \rho_1 \delta_1^{(t)}\right) \\ \delta_2^{(t+1)} &= \min\left(\delta_{\max}, \rho_2 \delta_2^{(t)}\right) \end{aligned}, \tag{40}$$

where $\delta_{\max}$ is upper bound of $\delta_1$ and $\delta_2$, and the values of $\rho_1$ and $\rho_2$ are defined according to Eq. (23), by substituting the corresponding variables of constraint.

Finally, Algorithm 2 summarizes the main steps of D-ADMM(C). One can see that only 4 FFT operations are required per iteration.

```
Algorithm 2: D-ADMM(C)
1. Preprocessing $\bar{\mathbf{y}} = \mathrm{mean}(\mathbf{y})$
2. Initilize $\mathbf{p}^{(0)}$, $\mathbf{q}^{(0)}$, $\mathbf{d}^{(0)}$, $\mathbf{f}^{(0)}$, $t = 0$
3. Precompute $\mathcal{F}(\mathbf{B}_h) = \mathcal{F}\left(\mathbf{A}^T\mathbf{A} + \delta_1 \mathbf{D}_v \mathbf{D}_v^T + \delta_2 \mathbf{I}\right)$
              $\mathcal{F}(\mathbf{B}_v) = \mathcal{F}\left(\mathbf{A}^T\mathbf{A} + \delta_1 \mathbf{D}_h \mathbf{D}_h^T + \delta_2 \mathbf{I}\right)$
4. while not converged
5.     $\mathbf{d}_h^{(t+1)} = \mathcal{F}^{-1}\left(\mathcal{F}\left(\mathbf{A}^T\mathbf{D}_h\mathbf{y} + \delta_1 \mathbf{D}_v\left(\mathbf{D}_h^T\mathbf{d}_v^{(t)} + \mathbf{p}^{(t)}\right)\right.\right.$
           $\left.\left. + \delta_2\left(\mathbf{f}_h^{(t)} - \mathbf{q}_h^{(t)}\right)\right) \oslash \mathcal{F}(\mathbf{B}_h)\right)$
6.     $\mathbf{d}_v^{(t+1)} = \mathcal{F}^{-1}\left(\mathcal{F}\left(\mathbf{A}^T\mathbf{D}_v\mathbf{y} + \delta_1 \mathbf{D}_h\left(\mathbf{D}_v^T\mathbf{d}_h^{(t)} - \mathbf{p}^{(t)}\right)\right.\right.$
           $\left.\left. + \delta_2\left(\mathbf{f}_v^{(t)} - \mathbf{q}_v^{(t)}\right)\right) \oslash \mathcal{F}(\mathbf{B}_v)\right)$
7.     $\mathbf{f}^{(t+1)} = \arg\min_{\mathbf{f}} \frac{1}{2}\left\|\mathbf{f} - \mathbf{q}^{(t)} - \mathbf{d}^{(t+1)}\right\|_2^2 + \mu/\delta_2 \|\mathbf{f}\|$
8.     $\mathbf{p}^{(t+1)} = \mathbf{p}^{(t)} + \mathbf{D}_h^T\mathbf{d}_v^{(t+1)} - \mathbf{D}_v^T\mathbf{d}_h^{(t+1)}$
9.     $\mathbf{q}^{(t+1)} = \mathbf{q}^{(t)} + \mathbf{d}^{(t+1)} - \mathbf{f}^{(t+1)}$
10.    Update $\delta_1^{(t+1)}$ and $\delta_2^{(t+1)}$
11.    $t = t + 1$
12. end while
13. $\mathbf{x} = \mathcal{U}\left(\mathbf{d}^{(t)}\right)$
14. $\mathbf{x} = \mathbf{x} + \bar{\mathbf{y}}$
```

### *4.2. Hybrid D-ADMM algorithm*

The TVIR model in Eq. (31) involves two equality constraints: $\mathbf{f} = \mathbf{d}$ and $\mathbf{D}_c^T\mathbf{d} = \mathbf{0}$. Hereby the augmented Lagrangian method is adopted to deal with the equality constraint $\mathbf{f} = \mathbf{d}$, and the conventional Lagrangian dual method is used to address the equality constraint $\mathbf{D}_c^T\mathbf{d} = \mathbf{0}$, resulting in a hybrid ADMM algorithm. In hybrid D-ADMM, we only consider the penalty on $\mathbf{f} = \mathbf{d}$, and define a partial augmented Lagrangian function as,

$$\begin{array}{c} \mathcal{L}_p = \mu \left\| \mathbf{f} \right\| + \frac{1}{2} \left\| \mathbf{A}\mathbf{d} - \mathbf{D}\mathbf{y} \right\|^2 + \frac{\delta}{2} \left\| \mathbf{d} - \mathbf{f} + \mathbf{q} \right\|^2 \\ \text{s.t.} \quad \mathbf{D}_c^T \mathbf{d} = \mathbf{0} \end{array}. \tag{41}$$

Then the ADMM algorithm is adopted to update $\mathbf{f}$, $\mathbf{d}$, and $\mathbf{q}$. Given $\mathbf{d}$ and $\mathbf{q}$, the $\mathbf{f}$ subproblem can be reformulated as,

$$\mathbf{f} = \arg \min_{\mathbf{f}} \frac{1}{2} \left\| \mathbf{f} - \mathbf{q} - \mathbf{d} \right\|_2^2 + \mu/\delta \left\| \mathbf{f} \right\|. \tag{42}$$

As discussed in Section 4.1, this subproblem can be easily solved by using the soft-thresholding or the group shrinkage operator.

Given $\mathbf{f}$ and $\mathbf{q}$, the $\mathbf{d}$ subproblem can be reformulated as,

$$\begin{array}{c} \mathbf{d} = \arg \min_{\mathbf{d}} \frac{1}{2} \left\| \mathbf{A}\mathbf{d} - \mathbf{D}\mathbf{y} \right\|^2 + \frac{\delta}{2} \left\| \mathbf{d} - \mathbf{f} + \mathbf{q} \right\|^2 \\ \text{s.t.} \quad \mathbf{D}_c^T \mathbf{d} = \mathbf{0} \end{array}. \tag{43}$$

To update $\mathbf{d}$, we first define the Lagrangian dual function of the $\mathbf{d}$ subproblem as,

$$\mathcal{L}_d = \frac{1}{2} \left\| \mathbf{A}\mathbf{d} - \mathbf{D}\mathbf{y} \right\|^2 + \frac{\delta}{2} \left\| \mathbf{d} - \mathbf{f} + \mathbf{q} \right\|^2 + \boldsymbol{\lambda}^T \mathbf{D}_c^T \mathbf{d}. \tag{44}$$

Thus, the KKT conditions can be written as,

$$\left\{ \begin{array}{l} \frac{\partial \mathcal{L}_d}{\partial \mathbf{d}} = \mathbf{A}^T \left( \mathbf{A}\mathbf{d} - \mathbf{D}\mathbf{y} \right) + \delta \left( \mathbf{d} - \mathbf{f} + \mathbf{q} \right) + \mathbf{D}_c \boldsymbol{\lambda} = \mathbf{0} \\ \frac{\partial \mathcal{L}_d}{\partial \boldsymbol{\lambda}} = \mathbf{D}_c^T \mathbf{d} = \mathbf{0} \end{array} \right. , \tag{45}$$

which yield the solutions,

$$\left\{ \begin{array}{l} \boldsymbol{\lambda} = \left( \mathbf{D}_c^T \mathbf{B}^{-1} \mathbf{D}_c \right)^{-1} \mathbf{D}_c^T \mathbf{B}^{-1} \left( \mathbf{A}^T \mathbf{D}\mathbf{y} + \delta \left( \mathbf{f} - \mathbf{q} \right) \right) \\ \mathbf{d} = \mathbf{B}^{-1} \left( \mathbf{A}^T \mathbf{D}\mathbf{y} + \delta \left( \mathbf{f} - \mathbf{q} \right) - \mathbf{D}_c \boldsymbol{\lambda} \right) \end{array} \right. , \tag{46}$$

where $\mathbf{B} = \mathbf{A}^T\mathbf{A} + \delta \mathbf{I}$. Fortunately, with the help of 2D discrete Fourier transform, all the matrix inverse operations in Eq. (46) can be efficiently computed in the Fourier domain. Let $\mathcal{F}(\boldsymbol{\lambda})$ be the Fourier transform of $\boldsymbol{\lambda}$, and $\mathcal{F}(\mathbf{D}_h)$, $\mathcal{F}(\mathbf{D}_v)$, $\mathcal{F}(\mathbf{D}_h^T)$ and $\mathcal{F}(\mathbf{D}_v^T)$ be the Fourier transform of $\mathbf{D}_h$, $\mathbf{D}_v$, $\mathbf{D}_h^T$ and $\mathbf{D}_v^T$, respectively. Similarly, we have $\mathcal{F}(\mathbf{a}) = \mathcal{F}\left(\mathbf{A}^T\mathbf{D}\mathbf{y}\right)$, $\mathcal{F}(\mathbf{B}) = \mathcal{F}\left(\mathbf{A}^T\mathbf{A}\right)$, $\mathcal{F}(\mathbf{D}_c^2) = \mathcal{F}\left(\mathbf{D}_h^T\mathbf{D}_h + \mathbf{D}_v^T\mathbf{D}_v\right)$. First, the solution to $\boldsymbol{\lambda}$ can be obtained in the Fourier domain,

$$\begin{array}{l} \mathcal{F}(\boldsymbol{\lambda}) = \big( \mathcal{F}(\mathbf{D}_v^T) \odot \left( \mathcal{F}(\mathbf{a}_h) + \delta \mathcal{F}(\mathbf{f}_h - \mathbf{q}_h) \right) \\ \quad - \mathcal{F}(\mathbf{D}_h^T) \odot \left( \mathcal{F}(\mathbf{a}_v) + \delta \mathcal{F}(\mathbf{f}_v - \mathbf{q}_v) \right) \big) \oslash \mathcal{F}(\mathbf{D}_c^2) \end{array}. \tag{47}$$

Then, the solutions to $\mathbf{d}_h$ and $\mathbf{d}_v$ can also be obtained in the Fourier domain,

$$\left\{ \begin{array}{l} \mathbf{d}_h = \mathcal{F}^{-1} \left( \left( F(\mathbf{a}_h) + \delta \mathcal{F}(\mathbf{f}_h - \mathbf{q}_h) \right) \ -\mathcal{F}(\mathbf{D}_v) \odot \mathcal{F}(\boldsymbol{\lambda}) \right) \varnothing \mathcal{F}(\mathbf{B}) ) \\ \mathbf{d}_v = \mathcal{F}^{-1} \left( \left( \mathcal{F}(\mathbf{a}_v) + \delta \mathcal{F}(\mathbf{f}_v - \mathbf{q}_v) \right) \ +\mathcal{F}(\mathbf{D}_h) \odot \mathcal{F}(\boldsymbol{\lambda}) \right) \varnothing \mathcal{F}(\mathbf{B}) ) \end{array} \right. . \tag{48}$$

Given $\mathbf{d}$ and $\mathbf{f}$, we update $\mathbf{q}$ using the standard strategy adopted in ADMM,

$$\mathbf{q}^{(t+1)} = \mathbf{q}^{(t)} + \mathbf{d}^{(t+1)} - \mathbf{f}^{(t+1)}. \tag{49}$$

For the updating of $\delta$, we adopt

$$\delta^{(t+1)} = \min \left( \delta_{\max}, \rho \delta^{(t)} \right), \tag{50}$$

where $\delta_{\max}$ is upper bound of $\delta$, and the values of $\rho$ is defined according to Eq. (23), by substituting the corresponding variables of constraint.

```
Algorithm 3: D-ADMM(H)
Preprocessing $\bar{\mathbf{y}} = \mathrm{mean}(\mathbf{y})$
Initilize $\mathbf{p}^{(0)}$, $\mathbf{d}^{(0)}$, $\mathbf{f}^{(0)}$, $t = 0$
Precompute $\mathcal{F}(\mathbf{B}) = \mathcal{F}\left(\mathbf{A}^T\mathbf{A} + \delta\mathbf{I}\right)$, $\mathcal{F}(\mathbf{a}) = \mathcal{F}\left(\mathbf{A}^T\mathbf{D}\mathbf{y}\right)$
           $\mathcal{F}\left(\mathbf{D}_c^2\right) = \mathcal{F}\left(\mathbf{D}_h^T\mathbf{D}_h + \mathbf{D}_v^T\mathbf{D}_v\right)$
while not converged
    $\mathcal{F}(\mathbf{s}_h) = \mathcal{F}\left(\mathbf{f}_h^{(t)} - \mathbf{q}_h^{(t)}\right)$, $\mathcal{F}(\mathbf{s}_v) = \mathcal{F}\left(\mathbf{f}_v^{(t)} - \mathbf{q}_v^{(t)}\right)$
    $\mathcal{F}(\boldsymbol{\lambda}) = \big(\mathcal{F}(\mathbf{D}_v^T) \odot (\mathcal{F}(\mathbf{a}_h) + \delta\mathcal{F}(\mathbf{s}_h))$
        $-\mathcal{F}(\mathbf{D}_h^T) \odot (\mathcal{F}(\mathbf{a}_v) + \delta\mathcal{F}(\mathbf{s}_v))\big) \varnothing \mathcal{F}(\mathbf{D}_c^2)$
    $\mathbf{d}_h^{(t+1)} = \mathcal{F}^{-1}\left(\left(\mathcal{F}(\mathbf{a}_h) + \delta\mathcal{F}(\mathbf{s}_h) - \mathcal{F}(\mathbf{D}_v) \odot \mathcal{F}(\boldsymbol{\lambda})\right) \varnothing \mathcal{F}(\mathbf{B})\right)$
    $\mathbf{d}_v^{(t+1)} = \mathcal{F}^{-1}\left(\left(\mathcal{F}(\mathbf{a}_v) + \delta\mathcal{F}(\mathbf{s}_v) - \mathcal{F}(\mathbf{D}_h) \odot \mathcal{F}(\boldsymbol{\lambda})\right) \varnothing \mathcal{F}(\mathbf{B})\right)$
    $\mathbf{f}^{(t+1)} = \arg\min_{\mathbf{f}} \frac{1}{2} \left\| \mathbf{f} - \mathbf{q}^{(t)} - \mathbf{d}^{(t+1)} \right\|_2^2 + \mu/\delta_2 \|\mathbf{f}\|$
    $\mathbf{q}^{(t+1)} = \mathbf{q}^{(t)} + \mathbf{d}^{(t+1)} - \mathbf{f}^{(t+1)}$
    Update $\delta^{(t+1)}$
    $t = t + 1$
end while
$\mathbf{x} = \mathcal{U}\left(\mathbf{d}^{(t)}\right)$
$\mathbf{x} = \mathbf{x} + \bar{\mathbf{y}}$
```

Finally, we summarize D-ADMM(H) in Algorithm 3.

### *4.3. Convergence and complexity*

We first prove the convergence of the D-ADMM algorithms.

**Proposition 1.** *For the reformulated derivative problem* (31)*, the iterates* $\{\mathbf{d}^{(t)}\}$ *generated by the D-ADMM algorithms converge to the optimal solution* $\mathbf{d}^*$*.*

*Proof*: To prove the convergence of D-ADMM, we simply need to demonstrate that it satisfies the conditions of Theorem 1.

The constraints of (31) can be rearranged as $[\mathbf{I}, \mathbf{D}_c]^T\mathbf{d} + [-\mathbf{I}, \mathbf{0}]^T\mathbf{f} = \mathbf{0}$, and then Problem (31) has the form (19), with $f(\mathbf{d}) = \frac{1}{2}\|\mathbf{Ad} - \mathbf{Dy}\|_2^2$ , $g(\mathbf{f}) = \mu\|\mathbf{f}\|$ , $\mathbf{G} = [\mathbf{I}, \mathbf{D}_c]^T$, $\mathbf{H} = [-\mathbf{I}, \mathbf{0}]^T$ and $\mathbf{c} = \mathbf{0}$. First, the functions $f$ and $g$ simply are quadratic function and TV function, respectively, which are obviously proper, closed and convex. Then, D-ADMM(C) and D-ADMM(H) share the same unaugmented Lagrangian function, i.e.,

$$\mathcal{L} = \frac{1}{2}\|\mathbf{Ad} - \mathbf{Dy}\|_2^2 + \mu\|\mathbf{f}\| + \boldsymbol{\lambda}^T(\mathbf{Gd} + \mathbf{Hf}). \tag{51}$$

For there only exists linear equality constraints in (31), its strong duality holds, leading to that the unaugmented Lagrangian function (51) has a saddle point [48].

Thus, D-ADMM(C) and D-ADMM(H) can both converge to the optimal solution $\mathbf{d}^*$. □

For the computational complexity, since $\mathcal{F}(\mathbf{D}_h)$, $\mathcal{F}, (\mathbf{D}_v)$ $\mathcal{F}(\mathbf{D}_h^T)$, $\mathcal{F}(\mathbf{D}_v^T)$, $\mathcal{F}(\mathbf{a})$, $\mathcal{F}(\mathbf{B})$, and $\mathcal{F}(\mathbf{D}_c^2)$ can be pre-computed in advance, D-ADMM(H) only requires four FFT operations per iteration. D-ADMM(C) also only requires four FFT operations per iteration. Compared with D-ADMM(C), D-ADMM(H) does not need to introduce the variable $\mathbf{p}$, and uses the Lagrangian dual method to obtain the closed form solution to $\mathbf{d}$. Thus, D-ADMM(H) would be better than D-ADMM(C) in terms of convergence speed and efficiency.

### *4.4. Implementation issues*

For D-ADMM(C) and D-ADMM(H), we adopt the following stopping criteria by checking whether the difference of variables between the subsequent iterations is below a sufficient small positive value $\varepsilon$,

$$\max\left(\|\mathbf{d}^{(t+1)} - \mathbf{d}^{(t)}\|/\|\mathbf{d}^{(t)}\|, \|\mathbf{f}^{(t+1)} - \mathbf{f}^{(t)}\|/\|\mathbf{f}^{(t)}\|\right) \le \varepsilon. \tag{52}$$

The iterative algorithm would stop once the difference is below a predefined threshold $\varepsilon$. The same initialization strategy is employed in both D-ADMM(C) and D-ADMM(H). $\mathbf{d}^{(0)}$ and $\mathbf{f}^{(0)}$ are simply initialized to be $\mathbf{Dy}$, and both $\mathbf{p}^{(0)}$ and $\mathbf{q}^{(0)}$ are initialized to be zero.

The D-ADMM(C) and D-ADMM(H) algorithms involve several parameters, i.e., $\delta_1^{(0)}$, $\delta_2^{(0)}$, $\delta_{max}$ for D-ADMM(C), and $\delta^{(0)}$, $\delta_{max}$ for D-ADMM(H). Although the algorithms converge to the optimal solution for any and , the values of these parameters do affect the convergence rate of algorithm. We empirically give the following recommendation: for D-ADMM(C) we have $\delta_1^{(0)} = \delta_2^{(0)} = 10^{-4}$, $\delta_{max} = 100$, while for D-ADMM(H) we have $\delta^{(0)} = 10^{-4}$, $\delta_{max} = 100$.

## 5. Experimental Results

In this section, we report the experimental results of the proposed algorithms for image restoration. In our experiments, we first compare the proposed D-ADMM algorithms with another derivative space based TVIR algorithm, TVIS [33]. To further verify the efficiency and effectiveness of the proposed methods, we further compare the restoration performance and running time of ADMM with those of the state-of-the-art ALM-based algorithms, i.e., SALSA [23] and FTVd [22].

The programs in our experiments are all coded in MATLAB and ran on a 2.40GHz Core(TM)i7-4700MQ laptop. We independently implement the codes of TVIS according to Michailovich's work [33]. The source codes of SALSA [1] and FTVd [2] are downloaded from the websites [3]. In our experiments, we only report the results of D-ADMM and the competing algorithms for solving the isotropic TVIR problem.

### *5.1. Comparison with TVIS*

In [33], Michailovich first presented a TVIS algorithm for solving the anisotropic TVIR problem, and then found a connection between $\mathrm{TV}_a$ and $\mathrm{TV}_i$ via multidirectional gradient, where $\mathrm{TV}_i$ can be approximated by $\mathrm{TV}_L$ with the following property,

$$\mathrm{TV}_L = \left\{ \begin{array}{l} \mathrm{TV}_a, \;\; L = 1 \\ \mathrm{TV}_i, \; L \to \infty \end{array} \right. . \tag{53}$$

[1] `http://cascais.lx.it.pt/~mafonso/SALSA_v2.0.zip`

[2] `http://www.caam.rice.edu/~optimization/L1/ftvd/v4.1/FTVd_v4.1.zip`

[3] We will also make our source code available online after the paper is accepted.

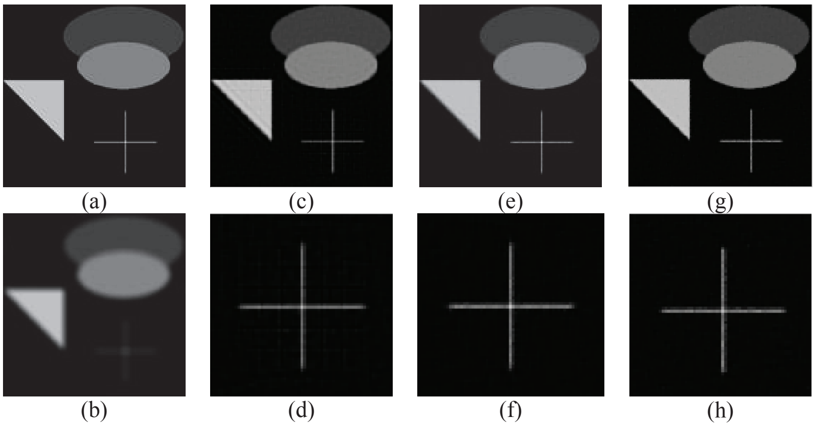

**Figure 2:** Restoration of a synthesis image. (a) the original image, (b) the degraded image, (c), (e) and (g) are the restoration results of TVIS-3, D-ADMM(C) and D-ADMM(H), respectively, (d), (f) and (h) are restoration details of (c), (e) and (g), respectively.

As suggested in [33], $\mathrm{TV}_i$ can be well approximated by $L = 3$. In our experiments, we set $L = 3$ for isotropic TVIR, and notate the TVIS method as TVIS-3.

We first use a $256 \times 256$ synthesis image shown in Fig. 2(a) to evaluate D-ADMM(C), D-ADMM(H) and TVIS-3 for isotropic TVIR. The synthesis image is blurred by the $7 \times 7$ Gaussian kernel with the mean of zero and the standard deviation of 3, and then Gaussian white noise with the standard deviation of $5 \times 10^{-3}$ is added to the blurred image to generate the degraded image. In the experiments, we set the regularization parameters $\mu = 4 \times 10^{-4}$ for D-ADMM(C) and D-ADMM(H) and $\tau = \mu/4$ for TVIS-3. Fig. 2 shows the restoration results of D-ADMM(C), D-ADMM(H) and TVIS-3. From Fig. 2, one can see that D-ADMM(C) and D-ADMM(H) are more effective for the small details like edges and lines.

For comprehensive evaluation, we further use ten $256 \times 256$ images to evaluate D-ADMM(C), D-ADMM(H) and TVIS-3. The ten images are shown in Fig. 3, and we use the same procedure for the synthesis image to generate the degraded images. Table 2 [4] and Table 3 list the PSNR and SSIM values of D-ADMM(C), D-ADMM(H), and TVIS-3 on these ten images. One can see that, D-ADMM(C) and D-ADMM(H) can achieve comparable or better

[4] In the following tables, the highest PSNR, SSIM or the fastest running time for each case is highlighted.

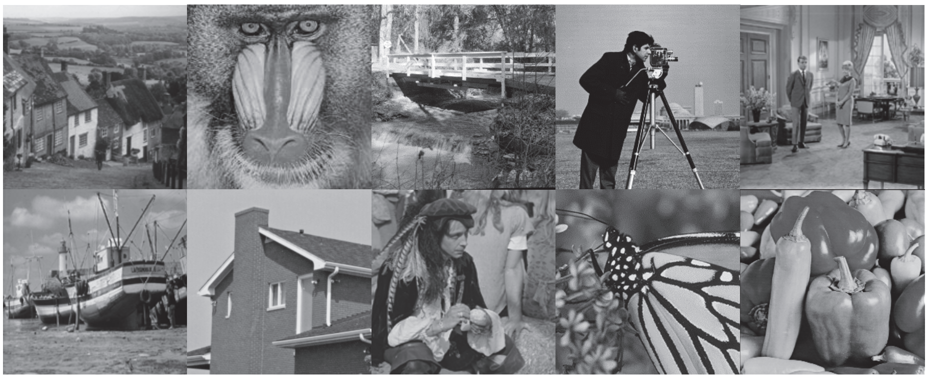

**Figure 3:** The ten images used in our experiments, hereafter we numbered them as 1 to 10 from left to right and top to down.

restoration quality than TVIS-3 in terms of both PSNR and SSIM [49] values. Since the D-ADMM algorithms inherit the second order convergence rate of ADMM, D-ADMM(C) and D-ADMM(H) are both much faster than TVIS [5], shown in Table 4.

### *5.2. Comparison with the state-of-the-art ALM algorithms*

In this section, we compare D-ADMM with two state-of-the-art ALM-based algorithms, i.e., SALSA and FTVd. To the best of our knowledge, FTVd is the fastest algorithm for TVIR. The ten images shown in Fig. 3 are adopted to test the TVIR algorithms. We conducted the deblurring experiments on different kernels and noise levels. We used two camera shake kernels, $\mathbf{k}1$ and $\mathbf{k}2$, used in [38], and the standard deviation (*std.*) of Gaussian white noise was set ranged from $10^{-3}$ to $10^{-2}$, which has been widely adopted in image deblurring literatures [14, 22, 24, 30, 32]. The degraded image is generated by convolving with two camera shake kernels, and we then add Gaussian white noise with zero mean value and different *std.*, i.e., $1 \times 10^{-3}$, $5 \times 10^{-3}$ and $1 \times 10^{-2}$. Corresponding to the noise levels, the trade-off parameter $\mu$ is set as $3 \times 10^{-5}$, $4 \times 10^{-4}$ and $2 \times 10^{-3}$, respectively, and $\tau = \mu/4$.

[5] In [33], Michailovich presents several ways to improve convergence rate of TVIS, which, however, are not adopted in our implementation. Even cooperated with accelerating approaches, TVIS is believed to be much less efficient than D-ADMM, for its first order convergence rate.

**Table 2:** PSNR comparison of TVIS and D-ADMM for isotropic TVIR

| Method | TVIS-3 | D-ADMM(C) | D-ADMM(H) |
|---|---|---|---|
| 1 | 29.76 | 29.78 | **29.80** |
| 2 | **24.88** | 24.86 | 24.85 |
| 3 | 25.41 | **25.61** | 25.58 |
| 4 | 27.20 | 27.26 | **27.30** |
| 5 | 28.13 | **28.26** | 28.19 |
| 6 | 27.49 | **28.03** | 27.99 |
| 7 | 31.31 | **32.56** | 32.51 |
| 8 | 28.54 | 28.64 | **28.65** |
| 9 | 26.96 | 27.80 | **27.82** |
| 10 | 29.53 | 29.90 | **29.95** |
| **Avg** | 27.92 | **28.27** | 28.26 |

Fig. 4 shows the restoration results of Image 7 obtained using D-ADMM(C), D-ADMM(H), SALSA, and FTVd. One can easily see that D-ADMM(C) and D-ADMM(H) can obtain similar or better restoration results while compared with SALSA and FTVd. We also investigate the influence of iteration number on PSNR and SSIM, in which the image is blurred by kernel 1 and zero-mean Gaussian white noise with *std.* $5 \times 10^{-3}$. We hereby stop all the algorithms at 100 iteration, and the PSNR, SSIM and objective function, i.e., $F(\mathbf{x})$ or $F(\mathbf{d})$, versus iterations are shown in Fig. 5. One can see that, all the four algorithms converge to the almost same objective function loss [6], and the two D-ADMM algorithms can converge to better results than SALSA and FTVd in terms of PSNR and SSIM. Also, D-ADMM(H) converges much faster than D-ADMM(C) and SALSA.

We further compare D-ADMM with the competing algorithms based on three performance indicators, PSNR, SSIM, and running time. From Table 5 and Table 6, one can see that the D-ADMM algorithms generally are superior or comparable with the competing methods in terms of restoration quality, especially in terms of SSIM values, for all noise levels. When the noise level is low, the D-ADMM algorithms perform better. When the noise level is high, the D-ADMM algorithms are lower than SALSA and FTVd in terms of PSNR, but are superior in terms of SSIM.

[6] Please note that $F(\mathbf{d})$ is about 4 times of $F(\mathbf{x})$.

**Table 3:** SSIM comparison of TVIS and D-ADMM for isotropic TVIR

| Method | TVIS-3 | D-ADMM(C) | D-ADMM(H) |
|---|---|---|---|
| 1 | 0.809 | 0.814 | **0.815** |
| 2 | **0.650** | 0.649 | 0.648 |
| 3 | 0.701 | **0.717** | 0.715 |
| 4 | 0.826 | **0.844** | 0.843 |
| 5 | 0.812 | **0.817** | 0.816 |
| 6 | 0.815 | **0.820** | 0.818 |
| 7 | 0.850 | **0.868** | 0.865 |
| 8 | 0.808 | **0.812** | 0.811 |
| 9 | 0.788 | 0.884 | **0.886** |
| 10 | 0.865 | **0.880** | 0.879 |
| **Avg** | 0.792 | **0.811** | 0.810 |

Note that SALSA, FTVd and the D-ADMM algorithms are developed to solve the same convex optimization problem and all can converge to the global optimal solution. Thus, efficiency is a more critical indicator for evaluating different algorithms. From Table 7, D-ADMM(C) is much faster than SALSA and is slower than FTVd. D-ADMM(H) can be faster than both SALSA and FTVd. D-ADMM(H) is at least 2 times faster than FTVd, and is at least 40 times faster than SALSA.

## 6. Conclusion

In this paper, we studied the derivative space based reformulation of the TVIR problem, and proposed two derivative augmented Lagrangian methods, i.e., D-ADMM(C) and D-ADMM(H). Unlike previous TVIR formulation, the proposed derivative space based reformulation provides a reasonable approximation of TVIR, and introduces an explicit equality constraint on the gradients d, which makes the proposed reformulation more concise and much easier to be solved. Thanks to the simplicity of the derivative space based TVIR formulation, D-ADMM(C) and D-ADMM(H) only require four FFT operations per iteration, and thus are efficient for TVIR. Experimental results show that, compared with the state-of-the-art TVIR algorithms, D-ADMM(H) can obtain satisfactory restoration result and is much faster. Besides, the proposed algorithms can also be extended for solving other image restoration applications, e.g., image inpainting, compressed sensing, and

**Table 4:** Running time comparison of TVIS and D-ADMM for isotropic TVIR

| Method | TVIS-3<br>($10^4$ *sec.*) | D-ADMM(C)<br>(*sec.*) | D-ADMM(H)<br>(*sec.*) |
|---|---|---|---|
| 1 | 5.272 | 4.656 | **0.208** |
| 2 | 5.298 | 5.341 | **0.183** |
| 3 | 5.281 | 4.447 | **0.115** |
| 4 | 5.306 | 4.151 | **0.224** |
| 5 | 5.305 | 4.274 | **0.241** |
| 6 | 5.256 | 4.274 | **0.221** |
| 7 | 5.260 | 5.705 | **0.353** |
| 8 | 5.342 | 5.681 | **0.274** |
| 9 | 5.371 | 4.391 | **0.159** |
| 10 | 5.340 | 3.765 | **0.114** |
| **Avg** | 5.301 | 4.669 | **0.209** |

even blind deconvolution.

## Acknowledgment

The work is partially supported by the NSFC funds of China (Grant No.s: 61271093 and 61001037), and the program of Ministry of Education for new century excellent talents under Grant No. NCET-12-0150.

## References

[1] H. Andrews, B. Hunt, Digital Image Restoration, Englewood Cliffs, HJ: Prentice-Hall, 1977.

[2] L. I. Rudin, S. Osher, E. Fatemi, Nonlinear total variation based noise removal algorithms, Physica D: Nonlinear Phenomena 60 (1) (1992) 259–268.

[3] D. Krishnan, R. Fergus, Fast image deconvolution using hyper-laplacian priors, in: Advances in Neural Information Processing Systems, 2009, pp. 1033–1041.

[4] J. Portilla, V. Strela, M. J. Wainwright, E. P. Simoncelli, Image denoising using scale mixtures of gaussians in the wavelet domain, IEEE Trans. Image Process. 12 (11) (2003) 1338–1351.

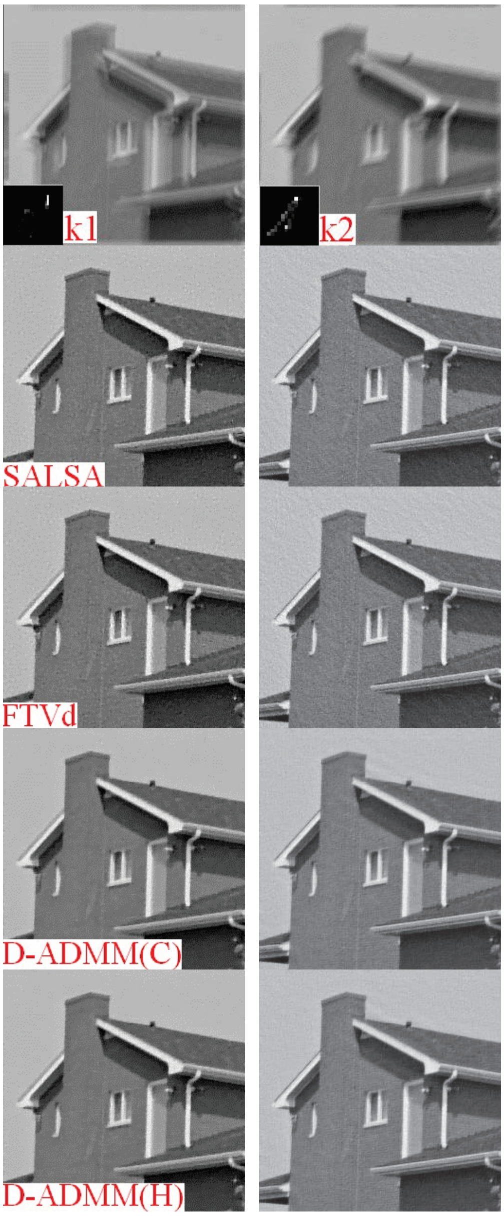

**Figure 4:** Restoration results of Image 7 obtained using different TVIR algorithms.

[5] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans. Image Process. 15 (12) (2006) 3736–3745.

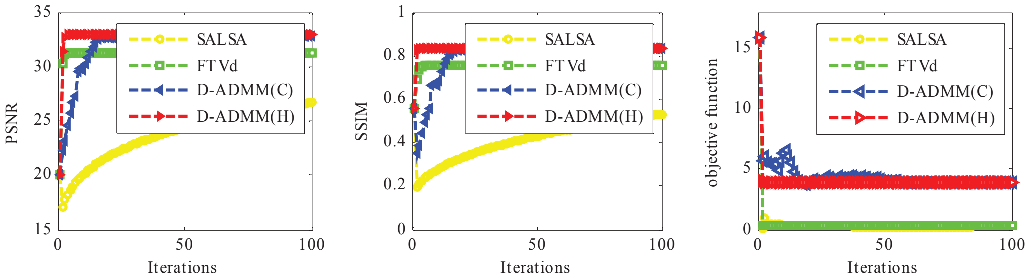

**Figure 5:** Iteration-varying of PSNR and SSIM.

[6] W. Dong, L. Zhang, G. Shi, X. Wu, Image deblurring and super-resolution by adaptive sparse domain selection and adaptive regularization, IEEE Trans. Image Process. 20 (7) (2011) 1838–1857.

[7] H. Lai, Y. Pan, C. Liu, L. Lin, J. Wu, Sparse learning-to-rank via an efficient primal-dual algorithm, IEEE Transactions on Computers 62 (6) (2013) 1221–1233.

[8] A. Buades, B. Coll, J. Morel, A review of image denoising algorithms, with a new one, Multiscale Modeling & Simulation 4 (2) (2005) 490–530.

[9] K. Dabov, A. Foi, V. Katkovnik, K. Egiazarian, Image denoising by sparse 3-d transform-domain collaborative filtering, IEEE Trans. Image Process. 16 (8) (2007) 2080–2095.

[10] W. Dong, L. Zhang, G. Shi, X. Li, Nonlocally centralized sparse representation for image restoration, IEEE Trans. Image Process. 22 (4) (2013) 1620–1630.

[11] A. Chambolle, An algorithm for total variation minimization and applications, Journal of Mathematical imaging and vision 20 (1-2) (2004) 89–97.

[12] Q. Chang, I.-L.Chern, Acceleration methods for total variation-based image denoising, SIAM Journal on Scientific Computing 25 (3) (2003) 982–994.

[13] T. F. Chan, C. K. Wong, Total variation blind deconvolution, IEEE Trans. Image Process. 7 (3) (1998) 370–375.

[14] S. H. Chan, R. Khoshabeh, K. B. Gibson, P. E. Gill, T. Q. Nguyen, An augmented lagrangian method for total variation video restoration, IEEE Trans. Image Process. 20 (11) (2011) 3097–3111.

[15] Y.-W. Wen, R. H. Chan, Parameter selection for total-variation-based image restoration using discrepancy principle, IEEE Trans. Image Process. 21 (4) (2012) 1770–1781.

[16] S. Ma, W. Yin, Y. Zhang, A. Chakraborty, An efficient algorithm for compressed mr imaging using total variation and wavelets, in: Computer Vision and Pattern Recognition, 2008, pp. 1–8.

[17] N. D, W. R, Stable image reconstruction using total variation minimization, SIAM Journal on Imaging Sciences 6 (2) (2013) 1035–1058.

[18] T. Goldstein, S. Osher, The split bregman method for l1-regularized problems, SIAM Journal on Imaging Sciences 2 (2) (2009) 323–343.

[19] A. Beck, M. Teboulle, Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems, IEEE Trans. Image Process. 18 (11) (2009) 2419–2434.

[20] J. Fadili, G. Peyr, Total variation projection with first order schemes, IEEE Trans. Image Process. 20 (3) (2011) 657–669.

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

[22] Y. Wang, J. Yang, W. Yin, Y. Zhang, A new alternating minimization algorithm for total variation image reconstruction, SIAM Journal on Imaging Sciences 1 (3) (2008) 248–272.

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

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

[25] L. Xu, C. Lu, Y. Xu, J. Jia, Image smoothing via l0 gradient minimization, ACM Transactions on Graphics (SIGGRAPH Asia) 30 (6) (2011) 681–695.

[26] S. Ramani, J. A. Fessler, A splitting-based iterative algorithm for accelerated statistical x-ray ct reconstruction, IEEE Trans. Image Process. 31 (3) (2012) 677–688.

[27] M. Tao, X. Yuan, Recovering low-rank and sparse components of matrices from incomplete and noisy observations, SIAM Journal on Optimization 21 (1) (2011) 57–81.

[28] A. Ganesh, Z. Lin, J. Wright, L. Wu, M. Chen, Y. Ma, Fast algorithms for recovering a corrupted low-rank matrix, in: IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009, pp. 213–216.

[29] J. Yang, Y. Zhang, Alternating direction algorithms for l1-problems in compressive sensing, SIAM Journal on Scientific Computing 33 (1) (2009) 250–278.

[30] M. Tao, J. Yang, B. He, Alternating direction algorithms for total variation deconvolution in image reconstruction, TR0918, Department of Mathematics, Nanjing University.

[31] V. M. Patel, R. Maleh, A. C. Gilbert, R. Chellappa, Gradient-based image recovery methods from incomplete fourier measurements, IEEE Trans. Image Process. 21 (1) (2012) 94–105.

[32] M. Rostami, O. V. Michailovich, Z. Wang, Image deblurring using derivative compressed sensing for optical imaging application, IEEE Trans. Image Process. 21 (7) (2012) 3139–3149.

[33] O. V. Michailovich, An iterative shrinkage approach to total-variation image restoration, IEEE Trans. Image Process. 20 (5) (2011) 1281–1299.

[34] R. Fergus, B. Singh, A. Hertzmann, S. T. Roweis, W. T. Freeman, Removing camera shake from a single photograph, ACM Transactions on Graphics (TOG) 25 (3) (2006) 787–794.

[35] K. Rektorys, Variational methods in mathematics, science and engineering, D. Reidel Pub. Co., 1977.

[36] D. Krishnan, T. Tay, R. Fergus, Blind deconvolution using a normalized sparsity measure, in: Computer Vision and Pattern Recognition, 2011, pp. 233–240.

[37] A. Levin, Y. Weiss, F. Durand, W. T. Freeman, Efficient marginal likelihood optimization in blind deconvolution, in: Computer Vision and Pattern Recognition, 2011, pp. 2657–2664.

[38] A. Levin, Y. Weiss, F. Durand, W. T. Freeman, Understanding and evaluating blind deconvolution algorithms, in: Computer Vision and Pattern Recognition, 2009, pp. 1964–1971.

[39] S. Cho, S. Lee, Fast motion deblurring, ACM Transactions on Graphics (SIGGRAPH Asia) 28 (5) (2009) 1–8.

[40] D. Ren, W. Zuo, H. Zhang, D. Zhang, A derivative augmented lagrangian method for fast total variation based image restoration, in: Intelligence Science and Big Data Engineering, Springer, 2013, pp. 287–294.

[41] N. Parikh, S. Boyd, Proximal algorithms, Foundations and Trends in Optimization 1 (3) (2013) 123–231.

[42] D. L. Donoho, J. M. Johnstone, Ideal spatial adaptation via wavelet shrinkage, Biometrika 81 (3) (1994) 425–455.

[43] R. H. Chan, J. Yang, X. Yuan, Alternating direction method for image inpainting in wavelet domains, SIAM Journal on Imaging Sciences 4 (3) (2011) 807–826.

[44] Z. Lin, R. Liu, Z. Su, Linearized alternating direction method with adaptive penalty for low-rank representation, in: NIPS, 2011.

[45] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends® in Machine Learning 3 (1) (2011) 1–122.

[46] C. M. Bishop, Pattern recognition and machine learning, Springer, 2006.

[47] G. B. Arfken, H. J. Weber, Mathematical methods for physicists, Vol. 6, San Diego, CA: Academic, 1995.

[48] S. P. Boyd, L. Vandenberghe, Convex optimization, Cambridge university press, 2004.

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Process. 13 (4) (2004) 600–612.

**Table 5:** PSNR comparison of different algorithms. In each cell, 4 values at Top-Left, Top-Right, Bottom-Left and Bottom-Right represent PSNR obtained by SALSA, FTVd, D-ADMM(C), D-ADMM(H), respectively.

| noise | $1 \times 10^{-3}$ | | $5 \times 10^{-3}$ | | $1 \times 10^{-2}$ | |
|---|---|---|---|---|---|---|
| kernel | $\mathbf{k_1}$ | $\mathbf{k_2}$ | $\mathbf{k_1}$ | $\mathbf{k_2}$ | $\mathbf{k_1}$ | $\mathbf{k_2}$ |
| 1 | **35.90** 35.88 | 35.15 35.19 | 30.93 30.41 | 30.82 31.14 | 29.80 **29.81** | 30.29 30.36 |
| | 35.53 35.74 | 36.64 **36.44** | 31.36 **31.54** | 32.09 **32.35** | 28.74 28.77 | 30.58 **30.66** |
| 2 | **31.28** 30.87 | **32.57** 32.22 | **28.33** 28.05 | 29.01 **29.07** | **26.41** 26.35 | **27.59** 27.50 |
| | 29.58 30.74 | 30.49 31.07 | 27.45 27.49 | 28.26 28.39 | 25.38 25.38 | 26.60 26.62 |
| 3 | 30.33 30.94 | 31.65 31.28 | 28.39 **28.41** | 29.17 **29.26** | **26.98** 26.88 | **27.71** 27.66 |
| | 30.54 **31.90** | 31.28 **32.27** | 28.14 28.12 | 28.87 28.95 | 25.59 25.60 | 27.00 27.03 |
| 4 | 33.03 **33.82** | **33.46** 33.21 | 30.03 **30.22** | 30.53 **30.85** | 29.02 **29.11** | 29.5 **29.63** |
| | 31.88 32.73 | 32.01 33.06 | 29.87 29.87 | 30.12 30.01 | 28.13 28.14 | 28.85 28.87 |
| 5 | 34.40 34.16 | 34.95 34.03 | 30.90 30.05 | 30.38 30.65 | **29.1829.18** | 29.56 **29.61** |
| | 34.58 **34.83** | 35.22 **35.32** | 30.72 **30.96** | 31.25 **31.51** | 28.26 28.32 | 29.53 29.59 |
| 6 | 33.89 33.80 | 34.14 34.08 | 30.06 30.24 | 30.54 30.84 | 29.42 **29.46** | 29.83 **29.92** |
| | **33.96** 34.66 | 35.08 **35.19** | 30.76 **31.03** | 31.29 **31.53** | 28.58 28.60 | 29.82 29.88 |
| 7 | 36.60 36.4 | 36.83 36.43 | 31.13 31.38 | 26.10 31.89 | 31.00 **31.13** | 31.37 31.58 |
| | **37.54** 37.31 | **38.75** 38.15 | 32.78 **33.00** | 33.31 **33.60** | 30.77 30.83 | 32.66 **32.78** |
| 8 | **33.77** 33.75 | 33.12 34.10 | 30.03 30.21 | 25.14 30.97 | 29.58 **29.60** | 29.94 **30.03** |
| | 33.73 34.2 | 34.55 **34.67** | 30.66 **30.91** | 31.19 **31.41** | 28.78 28.85 | 29.91 29.97 |
| 9 | 32.14 **32.88** | **32.53** 32.25 | 30.20 **30.43** | 30.81 **31.20** | 29.60 **29.69** | 30.20 **30.35** |
| | 30.62 31.49 | 31.61 31.76 | 29.07 29.16 | 29.65 29.86 | 26.99 27.02 | 28.72 28.74 |
| 10 | **35.19** 35.14 | 32.53 32.29 | 30.51 30.69 | 30.93 31.22 | **30.06** 29.88 | **30.41** 30.27 |
| | 33.70 35.17 | 33.44 **35.73** | **31.38** 31.2 | **31.77** 31.35 | 28.94 28.94 | 30.32 30.38 |
| **Avg.** | 33.65 33.76 | 33.69 33.43 | 30.05 30.01 | 29.34 30.71 | **29.1129.11** | 29.64 **29.69** |
| | 33.17 **33.88** | 33.91 **34.37** | 30.22 **30.33** | 30.78 **30.90** | 28.02 28.04 | 29.40 29.45 |

**Table 6:** SSIM comparison of different algorithms. In each cell, 4 values at Top-Left, Top-Right, Bottom-Left and Bottom-Right represent SSIM obtained by SALSA, FTVd, D-ADMM(C), D-ADMM(H), respectively.

| noise | $1 \times 10^{-3}$ | | $5 \times 10^{-3}$ | | $1 \times 10^{-2}$ | |
|---|---|---|---|---|---|---|
| kernel | $\mathbf{k_1}$ | $\mathbf{k_2}$ | $\mathbf{k_1}$ | $\mathbf{k_2}$ | $\mathbf{k_1}$ | $\mathbf{k_2}$ |
| 1 | 0.936 0.942<br>0.944 **0.948** | 0.952 0.953<br>0.954 **0.959** | 0.849 0.830<br>0.857 **0.861** | 0.838 0.850<br>0.874 **0.880** | 0.808 **0.812**<br>0.790 0.792 | 0.827 0.832<br>0.842 **0.845** |
| 2 | **0.954** 0.951<br>0.918 0.919 | 0.966 **0.971**<br>0.926 0.920 | 0.830 **0.834**<br>0.813 0.815 | 0.861 **0.866**<br>0.844 0.848 | 0.765 **0.766**<br>0.702 0.702 | **0.817** **0.817**<br>0.771 0.772 |
| 3 | 0.932 0.917<br>**0.941** 0.924 | 0.972 **0.975**<br>0.947 0.938 | 0.860 **0.862**<br>0.853 0.855 | 0.881 0.884<br>0.874 **0.877** | **0.811** 0.810<br>0.755 0.755 | **0.837** **0.837**<br>0.804 0.806 |
| 4 | 0.893 0.926<br>0.925 **0.934** | 0.918 0.925<br>0.940 **0.946** | 0.761 0.775<br>0.832 **0.835** | 0.769 0.791<br>0.832 **0.836** | 0.774 0.786<br>0.831 **0.832** | 0.776 0.792<br>0.847 **0.849** |
| 5 | 0.940 0.955<br>0.952 **0.959** | 0.954 0.955<br>0.960 **0.966** | 0.837 0.844<br>0.876 **0.882** | 0.846 0.857<br>0.884 **0.889** | 0.830 **0.833**<br>0.828 0.831 | 0.833 0.838<br>0.853 **0.855** |
| 6 | 0.930 0.950<br>0.946 **0.951** | 0.945 0.951<br>0.956 **0.963** | 0.820 0.829<br>0.862 **0.868** | 0.831 0.844<br>0.875 **0.879** | 0.819 **0.825**<br>0.824 **0.825** | 0.825 0.833<br>0.855 **0.858** |
| 7 | 0.810 0.922<br>0.931 **0.950** | 0.866 0.939<br>0.946 **0.962** | 0.750 0.764<br>0.831 **0.836** | 0.764 0.776<br>0.834 **0.840** | 0.767 0.779<br>0.835 **0.837** | 0.766 0.781<br>0.850 **0.854** |
| 8 | 0.933 **0.951**<br>0.940 0.946 | 0.949 0.943<br>0.949 **0.957** | 0.825 0.833<br>0.863 **0.869** | 0.864 0.851<br>0.872 **0.877** | 0.824 0.827<br>0.830 **0.833** | 0.828 0.835<br>0.851 **0.853** |
| 9 | 0.929 0.931<br>0.939 **0.942** | 0.946 0.943<br>0.950 **0.961** | 0.857 0.848<br>0.877 **0.880** | 0.856 0.862<br>0.882 **0.887** | 0.853 0.862<br>0.870 **0.871** | 0.852 0.862<br>0.891 **0.892** |
| 10 | 0.914 0.938<br>0.939 **0.945** | 0.935 0.933<br>**0.952** 0.95 | 0.802 0.815<br>0.866 **0.868** | 0.806 0.824<br>0.865 **0.867** | 0.819 0.828<br>0.862 **0.863** | 0.815 0.827<br>0.876 **0.879** |
| **Avg.** | 0.917 0.938<br>0.937 **0.941** | 0.940 0.948<br>0.948 **0.952** | 0.819 0.823<br>0.853 **0.857** | 0.780 0.841<br>0.864 **0.868** | 0.807 0.813<br>0.813 **0.814** | 0.817 0.825<br>0.844 **0.846** |

**Table 7:** Running time (*sec.*) comparison of different algorithms. In each cell, 4 values at Top-Left, Top-Right, Bottom-Left and Bottom-Right represent running time of SALSA, FTVd, D-ADMM(C), D-ADMM(H), respectively.

| noise | $1 \times 10^{-3}$ | | $5 \times 10^{-3}$ | | $1 \times 10^{-2}$ | |
|---|---|---|---|---|---|---|
| kernel | $\mathbf{k_1}$ | $\mathbf{k_2}$ | $\mathbf{k_1}$ | $\mathbf{k_2}$ | $\mathbf{k_1}$ | $\mathbf{k_2}$ |
| 1 | 21.47 0.473<br>1.307 **0.151** | 25.52 0.494<br>2.459 **0.178** | 24.64 0.803<br>2.296 **0.279** | 27.74 0.832<br>2.715 **0.180** | 24.09 1.252<br>4.354 **0.788** | 25.15 1.072<br>4.544 **0.602** |
| 2 | 24.49 0.699<br>3.450 **0.197** | 26.26 0.632<br>3.522 **0.249** | 25.49 1.032<br>3.309 **0.243** | 24.28 1.004<br>3.481 **0.228** | 23.69 1.168<br>4.109 **0.595** | 23.06 1.025<br>4.071 **0.531** |
| 3 | 24.93 0.724<br>3.326 **0.239** | 23.86 0.677<br>3.625 **0.242** | 24.64 1.004<br>3.330 **0.240** | 24.22 0.888<br>3.667 **0.226** | 22.69 1.212<br>3.821 **0.562** | 22.28 1.057<br>4.389 **0.521** |
| 4 | 23.27 0.763<br>3.739 **0.237** | 23.04 0.720<br>3.879 **0.215** | 23.32 1.045<br>3.724 **0.308** | 22.77 0.924<br>3.994 **0.227** | 20.25 1.411<br>4.312 **0.597** | 26.09 1.258<br>4.445 **0.522** |
| 5 | 22.95 0.744<br>3.510 **0.239** | 21.35 0.630<br>3.773 **0.210** | 21.14 1.020<br>3.142 **0.320** | 20.28 0.913<br>3.377 **0.217** | 24.15 1.420<br>3.798 **0.557** | 21.56 0.999<br>4.309 **0.551** |
| 6 | 27.72 0.795<br>3.548 **0.217** | 22.78 0.701<br>4.028 **0.367** | 27.83 1.031<br>3.407 **0.446** | 29.40 2.107<br>4.602 **0.235** | 23.10 1.155<br>3.544 **0.543** | 22.79 0.969<br>4.059 **0.567** |
| 7 | 16.43 0.709<br>2.879 **0.222** | 13.59 0.657<br>3.714 **0.196** | 21.65 0.941<br>4.406 **0.500** | 19.36 0.839<br>3.582 **0.290** | 25.49 1.103<br>4.269 **0.641** | 26.78 1.713<br>6.348 **0.717** |
| 8 | 13.96 0.735<br>3.284 **0.196** | 14.05 0.673<br>4.079 **0.209** | 23.91 0.996<br>3.341 **0.289** | 18.65 0.892<br>4.039 **0.228** | 26.93 1.172<br>6.132 **0.566** | 21.87 1.002<br>4.651 **0.528** |
| 9 | 24.45 0.720<br>3.660 **0.190** | 20.94 0.669<br>5.476 **0.222** | 20.62 0.995<br>3.942 **0.290** | 18.89 2.247<br>5.434 **0.224** | 24.29 0.618<br>1.877 **0.245** | 24.75 0.540<br>2.151 **0.243** |
| 10 | 25.24 0.729<br>3.981 **0.449** | 21.61 0.681<br>4.032 **0.220** | 23.85 1.138<br>3.804 **0.324** | 23.20 2.153<br>3.901 **0.245** | 24.58 0.585<br>1.579 **0.253** | 24.07 0.494<br>1.975 **0.236** |
| **Avg.** | 24.49 0.709<br>3.268 **0.234** | 21.30 0.653<br>3.859 **0.231** | 23.71 1.001<br>3.470 **0.324** | 22.88 1.280<br>3.879 **0.230** | 23.93 1.110<br>3.779 **0.535** | 23.84 1.013<br>4.094 **0.502** |