Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Block Decomposition Methods for Total Variation by Primal-Dual Stitching

Chang-Ock Lee · Jong Ho Lee · Hyenkyun Woo · Sangwoon Yun

Received: date / Accepted: date

Abstract Due to the advance of image capturing devices, huge size of images are available in our daily life. As a consequence the processing of large scale image data is highly demanded. Since the total variation (TV) is kind of de facto standard in image processing, we consider block decomposition methods for TV based variational models to handle large scale images. Unfortunately, TV is non-separable and non-smooth and it thus is challenging to solve TV based variational models in a block decomposition. In this paper, we introduce a primal-dual stitching (PDS) method to efficiently process the TV based variational models in the block decomposition framework. To characterize TV in the block decomposition framework, we only focus on the proximal map of TV function. Empirically, we have observed that the proposed PDS based block decomposition framework outperforms other state-of-art methods such as Bregman operator splitting based approach in terms of computational speed.

Keywords Total variation · Block decomposition · Primal-dual stitching · Domain decomposition · Pseudo explicit method · Primal-dual optimization

Mathematics Subject Classification (2000) 49M27 · 65Y05 · 65K10 · 68U10

C.-O. Lee
Department of Mathematical Sciences, KAIST, Daejeon 305-701, Republic of Korea
E-mail: colee@kaist.edu

J.H. Lee
Samsung SDS
E-mail: jongho.lee84@gmail.com

H. Woo (Corresponding author)
Korea University of Technology & Education, Cheonan, Chungnam Province, 330-708, Republic of Korea
E-mail: hyenkyun@koreatech.ac.kr

S. Yun
Department of Mathematics Education, Sungkyunkwan University, Jongro-gu, Seoul 110-745, Republic of Korea
E-mail: yswmathedu@skku.edu

1 Introduction

Image restoration problems frequently appear in various applications such as remote sensing, medical image reconstruction, and computer vision. Due to the fast technological advance in image capturing devices, the real time processing of large scale image data is highly demanded. Nowadays, the total variation (TV) [23] is kind of de facto standard in image processing problems such as image deblurring, inpainting, denoising, and segmentation. Also, recent advances of optimization methodologies [1,7,10,16,29,31,34] reduce computational complexity for finding a solution of TV based variational model and we could thus find a solution in a reasonable CPU time. In modern multi-core CPU environment, real time processing of large scale image data is highly demanded. Therefore, it is natural to study block decomposition methods for TV based image restoration problems. However, due to the non-smoothness and non-separability of TV, it is not easy to directly apply the conventional block decomposition method [3] to the TV based models. Note that, if the non-differentiable function is separable, e.g., ℓ_1-norm minimization problem, then we can establish a global convergence of block decomposition methods [4,12] and a block coordinate method [24]. Due to the non-differentiability and non-separability of TV, theoretical global convergence is only available for overlapped domain decompositions [13,19]. Note that recently, dual based approaches [9,20] are also introduced to overcome the difficulties of TV in domain decomposition applications. Although the theoretical global convergence is not known, the block decomposition method is highly useful due to the simple structure of the method. In this paper, we introduce primal-dual stitching (PDS) based block decomposition methods. The proposed PDS can make us efficiently decompose the non-separable TV function. The proposed block decomposition framework gives a plausible answer in terms of efficient implementation of parallelism for TV based models.

Since we focus on the block decomposition framework for TV based models, we only treat the ROF model [23], which is also known as a proximal map of TV (proxTV) [2]. See also [5] for various features of the proximal map. Note that proxTV is the basic model for various image restoration problems such as deblurring [31] and denoising [21]. Now, let us assume that $b \in \mathbb{R}_+^{M\times N}$ (the set of $M\times N$ matrices whose entries are nonnegative real numbers) be given image data and the corresponding $M\times N$ image domain as Ω, then we get the following proxTV formulation:

$$\min_{u\in V} \quad J(u) = \|u-b\|_2^2 + \mu \mathrm{TV}(u) \tag{1}$$

where $\mu > 0$ and $V \subset {\mathbb{R}_+}^{M\times N}$ is a closed convex set. Note that, TV is defined as $\mathrm{TV}(u) = \|\nabla u\|_1 = \sum_{i,j} \|(\nabla u)_{i,j}\|_2$ with $(\nabla u)_{i,j} = ((\nabla u)^x_{i,j}, (\nabla u)^y_{i,j}) \in \mathbb{R}^2$ and $\|(\nabla u)_{i,j}\|_2 = \sqrt{((\nabla u)^x_{i,j})^2 + ((\nabla u)^y_{i,j})^2}$. Here the backward difference scheme is used for the discrete gradient operator ∇. Let Ω be an $M\times N$ image domain and we use the Neumann adiabatic boundary condition [8] on the boundary of the image domain Ω for the discrete gradient operator ∇ (i.e., $\nabla u|_{\partial\Omega} = 0$).

The paper is organized as follows. In section 2, we describe the general block decomposition framework. In section 3, we introduce a PDS method for the proximal map of TV. In section 4, we show the performance of the proposed PDS based block

decomposition method via numerical experiments on various test images. Finally, we give our conclusion in section 5.

2 Block Decomposition Method

In this section, we introduce a sequential block decomposition method (nonlinear block Gauss-Seidel method) and a parallel block decomposition method (nonlinear block Jacobi method) for proxTV (1). Note that the block decomposition methods we consider are nonoverlapping block decomposition methods [22]. For an overlapping block decomposition method, see [13,19].

Let $V = \cup_{i=1}^{I} V_i$ be a given domain such that $V_k \cap V_l = \emptyset$ for $k \neq l$. That is, $V = \oplus_{i=1}^{I} V_i$ is a direct sum of each subspace V_i. Then we get the following nonoverlapping sequential block decomposition framework (BD-S):

$$\begin{cases} \text{For } k = 1, ..., K \\ \text{For } i = 1, ..., I \\ \quad u_i^{(k+\frac{1}{2})} = \underset{u_i \in V_i}{\arg\min}\, J(u_1^{(k+1)} + ... + u_{i-1}^{(k+1)} + u_i + u_{i+1}^{(k)} + ... + u_I^{(k)}) \\ \quad u_i^{(k+1)} = \omega u_i^{(k+\frac{1}{2})} + (1-\omega) u_i^{(k)} \\ \text{End} \\ u^{(k+1)} = \sum_{i=1}^{I} u_i^{(k+1)} \\ \text{End} \end{cases} \tag{2}$$

where $\omega > 0$ is a tuning parameter. Note that when $\omega = 1$, BD-S becomes the conventional nonlinear block Gauss-Seidel method. If $0 < \omega < 1$ then the model is underrelaxed and if $\omega > 1$ then the model is overrelaxed. In fact, the best value for ω should be determined. Note that this block decomposition method is also known as a nonlinear block successive over relaxation (NBSOR) [17,27]. Since the structure of BD-S (2) is sequential, it requires elaborated effort to be implemented efficiently in a parallel computing machine. For instance, the coloring technique [30] is used for the block decomposition method of TV. To overcome this drawback, the following parallel nonoverlapping block decomposition method (or nonlinear block Jacobi) is commonly used.

$$\begin{cases} \text{For } k = 1, ..., K \\ \text{ParFor } i = 1, ..., I \\ \quad u_i^{(k+\frac{1}{2})} = \underset{u_i \in V_i}{\arg\min}\, J(u_1^{(k)} + ... + u_{i-1}^{(k)} + u_i + u_{i+1}^{(k)} + ... + u_I^{(k)}) \\ \text{End} \\ u^{(k+1)} = \sum_{i=1}^{I} (\omega u_i^{(k+\frac{1}{2})} + (1-\omega) u_i^{(k)}) \\ \text{End} \end{cases} \tag{3}$$

where $\omega > 0$. We call this model BD-P. Note that the BD-P method (3) only uses the previous k-th iteration result when i-th variable of the $(k+1)$-th iteration is updated. Hence the algorithm is highly parallelizable and easy to implement. Note that, to guarantee the monotonicity property of J, i.e. $J(u^{(k+1)}) \leq J(u^{(k)})$, the relaxation parameter $\omega = 1/I$ is used in [12,14].

Remark 1 In the image restoration problems such as image deblurring [31] and denoising [21] problems, we often solve

$$\min_{u\in V} f(u)+\mu \mathrm{TV}(u) \tag{4}$$

for a convex and smooth function f. Note that, for typical image restoration problems, $f(u)=\|Au-b\|_2^2$, where A depends on the type of the restoration model, for example, for the deblurring problem, A is a convolution operator and b is the given blurred image. For Poisson noise reduction problems, we have $f(u)=\langle u-b\log u, \mathbf{1}\rangle$. Since f in (4) is convex and smooth, a typical approach to find a solution of (4) is to use the proximal gradient method. That is, by using Taylor expansion of the fidelity term $f(u)$ at a, we introduce the following surrogate function for the fidelity term in (4) :

$$f^s(u;a) := f(a)+\langle \nabla f(a), u-a\rangle + \frac{1}{2}\|u-a\|_H^2,$$

where $\|u-a\|_H=\langle u-a, H(u-a)\rangle$. Note that H is a positive definite diagonal matrix which approximates $\nabla^2 f(a)$ and $H-\nabla^2 f(a)$ is a positive definite matrix. See also [21]. By using this surrogate function, we can easily find a solution of (4) with the following iterations:

$$u^{(r+1)} = \arg\min_{u\in V} f^s(u;u^{(r)})+\mu TV(u), \quad r=1,2,... \tag{5}$$

For more details on this proximal gradient method, see [1,2]. We note that, since the surrogate function $f^s(u,u^{(r)})$ is separable, it is naturally parallelizable for the block decomposition method. Therefore, in this paper, we focus on the proxTV (1).

2.1 Descent property of the block decomposition methods

In this section, we study the descent property of BD-S (2) when $0<\omega\le 1$ and BD-P (3) when $\omega = 1/I$.

The following lemma provides a key inequality for establishing the descent property of the BD-S method and the BD-P method.

Lemma 1 *For any $u=\sum_{i=1}^I u_i$, we have that*

$$J(\bar{u})-J(u)\le -\omega\|\bar{u}-u\|_2^2, \tag{6}$$

where $\bar{u}=\omega\tilde{u}(i)+(1-\omega)u$ with

$$\tilde{u}(i)=u_1+...+u_{i-1}+\tilde{u}_i+u_{i+1}+...+u_I$$

and

$$\tilde{u}_i = \arg\min_{u_i\in V_i} J(u_1+...+u_{i-1}+u_i+u_{i+1}+...+u_I)$$

for $1\le i\le I$.

Proof Since $a^2-b^2 = 2\langle b, a-b\rangle + (a-b)^2$, we have that

$$\begin{aligned} J(\tilde{u}(i)) - J(u) &= \|\tilde{u}(i)-b\|_2^2 - \|u-b\|_2^2 + \mu\mathrm{TV}(\tilde{u}(i)) - \mu\mathrm{TV}(u) \\ &= 2\langle u-b, \tilde{u}(i)-u\rangle + \|\tilde{u}(i)-u\|_2^2 + \mu\mathrm{TV}(\tilde{u}(i)) - \mu\mathrm{TV}(u). \end{aligned} \quad (7)$$

For any $\omega \in (0,1)$, we have from the definition of $\tilde{u}(i)$, i.e., $\tilde{u}_i$, and the convexity of $\mu\mathrm{TV}(\cdot)$ that

$$\begin{aligned} & 2\langle u-b, \tilde{u}(i)-u\rangle + \|\tilde{u}(i)-u\|_2^2 + \mu\mathrm{TV}(\tilde{u}(i)) \\ &= \left(\|\tilde{u}(i)-b\|_2^2 - \|u-b\|_2^2\right) + \mu\mathrm{TV}(\tilde{u}(i)) \\ &\le \left(\|\bar{u}-b\|_2^2 - \|u-b\|_2^2\right) + \mu\mathrm{TV}(\bar{u}) \\ &= 2\langle u-b, \bar{u}-u\rangle + \|\bar{u}-u\|_2^2 + \mu\mathrm{TV}(\bar{u}) \\ &\le 2\omega\langle u-b, \tilde{u}(i)-u\rangle + \|\omega(\tilde{u}(i)-u)\|_2^2 + \omega\mu\mathrm{TV}(\tilde{u}(i)) + (1-\omega)\mu\mathrm{TV}(u), \end{aligned}$$

where the last inequality uses the definition of $\bar{u}$. Rearranging terms yields

$$2(1-\omega)\langle u-b, \tilde{u}(i)-u\rangle + (1-\omega)\left(\mu\mathrm{TV}(\tilde{u}(i)) - \mu\mathrm{TV}(u)\right) + (1-\omega^2)\|\tilde{u}(i)-u\|_2^2 \le 0.$$

Since $1-\omega^2 = (1-\omega)(1+\omega)$, dividing both sides by $1-\omega > 0$ and then taking $\omega \uparrow 1$ imply that

$$2\langle u-b, \tilde{u}(i)-u\rangle + (\mu\mathrm{TV}(\tilde{u}(i)) - \mu\mathrm{TV}(u)) + \|\tilde{u}(i)-u\|_2^2 \le -\|\tilde{u}(i)-u\|_2^2. \quad (8)$$

This together with (7) yields that

$$J(\tilde{u}(i)) - J(u) \le -\|\tilde{u}(i)-u\|_2^2. \quad (9)$$

For any $\omega \in (0,1]$, by the convexity of J and using (9), we have that

$$\begin{aligned} J(\bar{u}) - J(u) &\le \omega J(\tilde{u}(i)) + (1-\omega)J(u) - J(u) \\ &= -\omega\|\tilde{u}(i)-u\|_2^2 \\ &\le -\omega\|\bar{u}-u\|_2^2 \end{aligned}$$

which proves (6). ■

The next theorem establishes the descent property of the BD-S method.

Theorem 1 *Let $\{u^{(k)}\}$ be the sequences generated by the BD-S method (2) with $0 < \omega \le 1$. Then the following properties are hold.*

(a) $J(u^{(k)}) > J(u^{(k+1)})$ for all k unless $u^{(k)} = u^{(k+1)}$.
(b) $\lim_{k\to\infty} \|u^{(k+1)} - u^{(k)}\|_2 = 0$.

Proof (a) Let

$$\tilde{u}^{(k)}(i) = u_1^{(k+1)} + \ldots + u_{i-1}^{(k+1)} + u_i^{(k)} + u_{i+1}^{(k)} + \ldots + u_I^{(k)}$$

and

$$\tilde{u}^{(k+1)}(i) = u_1^{(k+1)} + \ldots + u_{i-1}^{(k+1)} + u_i^{(k+1)} + u_{i+1}^{(k)} + \ldots + u_I^{(k)}.$$

Then, by Lemma 1 with $\bar{u} = \tilde{u}^{(k+1)}(i)$ and $u = \tilde{u}^{(k)}(i)$, we obtain that, for all k and $i = 1, \dots, I$,

$$J(\tilde{u}^{(k)}(i)) - J(\tilde{u}^{(k+1)}(i)) \geq \omega \|\tilde{u}^{(k+1)}(i) - \tilde{u}^{(k)}(i)\|_2^2.$$

Summing the above inequality over $i = 1, \dots, I$, we have that

$$J(u^{(k)}) - J(u^{(k+1)}) \geq \omega \|u^{(k+1)} - u^{(k)}\|_2^2. \tag{10}$$

Hence if $u^{(k)} \neq u^{(k+1)}$, then $J(u^{(k)}) > J(u^{(k+1)})$.

(b) Summing up the inequalities (10) for all k implies that

$$J(u^{(1)}) - \lim_{k\to\infty} J(u^{(k)}) \geq \sum_{k=1}^{\infty} \omega \|u^{(k+1)} - u^{(k)}\|_2^2. \tag{11}$$

Since $J(u)$ is bounded below and the sequence $\{J(u^{(k)})\}$ is monotonically decreasing, (11) yields that $\lim_{k\to\infty} \|u^{(k+1)} - u^{(k)}\|_2 = 0$.

■

Remark 2 For the BD-S method, the coordinate block is cyclically selected, i.e., the Gauss-Seidel rule is applied; see [25] and references therein. Although the objective value decreases, it does not guarantee that the sequence generated by the method converges to the optimial solution. But if the Gauss-Southwell type rule [25] is applied, then it does guarantee that the sequence generated by the method converges to the optimal solution even if TV is nonseparable.

The next theorem establishes the descent property of the BD-P method.

Theorem 2 *Let $\{u^{(k)}\}$ be the sequences generated by the BD-P method (3) with $\omega = 1/I$. Then the following properties are hold.*

(a) $J(u^{(k)}) > J(u^{(k+1)})$ for all k unless $u^{(k)} = u^{(k+1)}$.
(b) $\lim_{k\to\infty} \|u^{(k+1)} - u^{(k)}\| = 0$.

Proof (a) Let

$$\hat{u}^{(k+\frac{1}{2})}(i) = u_1^{(k)} + \ldots + u_{i-1}^{(k)} + u_i^{(k+\frac{1}{2})} + u_{i+1}^{(k)} + \ldots + u_I^{(k)}.$$

Then, by Lemma 1 with $\bar{u} = \hat{u}^{(k+\frac{1}{2})}(i)$ and $u = u^{(k)}$, we obtain that, for all k and $i = 1, \dots, I$,

$$J(u^{(k)}) - J(\hat{u}^{(k+\frac{1}{2})}(i)) \geq \|\hat{u}^{(k+\frac{1}{2})}(i) - u^{(k)}\|_2^2. \tag{12}$$

Let

$$\hat{u}^{(k+1)}(i) = u_1^{(k)} + \ldots + u_{i-1}^{(k)} + u_i^{(k+1)} + u_{i+1}^{(k)} + \ldots + u_I^{(k)},$$

then, by using the inequality (12), we have that

$$J(u^{(k)}) - J(\hat{u}^{(k+\frac{1}{2})}(i)) \geq \omega^2 \|\hat{u}^{(k+1)}(i) - u^{(k)}\|_2^2. \tag{13}$$

Summing the above inequality over $i = 1, \ldots, I$ and dividing by I on both sides, we have that

$$J(u^{(k)}) - \omega \sum_{i=1}^{I} J(\hat{u}^{(k+\frac{1}{2})}(i)) \geq \omega^3 \|u^{(k+1)} - u^{(k)}\|_2^2. \tag{14}$$

By the definition of $\hat{u}^{(k+\frac{1}{2})}(i)$ and the convexity of J, we have that

$$\begin{aligned} J(u^{(k+1)}) &= J\left(\sum_{i=1}^{I} \omega u_i^{(k+\frac{1}{2})} + (1-\omega) u_i^{(k)}\right) \\ &= J\left(\omega\left(\sum_{i=1}^{I} (\hat{u}^{(k+\frac{1}{2})}(i)) - (I-1)u^{(k)}\right) + (1-\omega)\sum_{i=1}^{I} u_i^{(k)}\right) = J\left(\omega \sum_{i=1}^{I} (\hat{u}^{(k+\frac{1}{2})}(i))\right) \\ &\leq \omega \sum_{i=1}^{I} J(\hat{u}^{(k+\frac{1}{2})}(i)), \end{aligned} \tag{15}$$

where the third equality uses $u^{(k)} = \sum_{i=1}^{I} u_i^{(k)}$. This inequality together with (14) implies that

$$J(u^{(k)}) - J(u^{(k+1)}) \geq \omega^3 \|u^{(k+1)} - u^{(k)}\|_2^2. \tag{16}$$

Hence if $u^{(k)} \neq u^{(k+1)}$, then $J(u^{(k)}) > J(u^{(k+1)})$.

(b) Summing up the inequalities (16) for all k implies that

$$J(u^{(1)}) - \lim_{k\to\infty} J(u^{(k)}) \geq \sum_{k=1}^{\infty} \omega^3 \|u^{(k+1)} - u^{(k)}\|_2^2. \tag{17}$$

Since $J(u)$ is bounded below and the sequence $\{J(u^{(k)})\}$ is monotonically decreasing, (17) yields that $\lim_{k\to\infty} \|u^{(k+1)} - u^{(k)}\|_2 = 0$.

■

3 Primal-Dual Stitching

In this section, we introduce the PDS technique for the block decomposition (i.e. domain decomposition).

To find a solution of BD-S (2) and BD-P (3) for each block variable $u_i \in V_i$, we need to solve the following subproblems

$$u_i^{(k+1)} = \underset{u_i \in V_i}{\arg\min}\, J(u_i + U_i^{(k)}), \tag{18}$$

where $U_i^{(k)} = \sum_{j<i} u_j^{(k+1)} + \sum_{j>i} u_j^{(k)}$ for BD-S (2) and $U_i^{(k)} = \sum_{j\neq i} u_j^{(k)}$ for BD-P (3). Since TV is non-separable, we need to be careful when we find a solution of (18). One method is to reformulate it with a projection operator. The following is the reformulation of (18):

$$u_i^{(k+1)} = \underset{u \in V}{\arg\min} \left\{ J(u) \ : \ \pi_{V_i^c}(u) = U_i^{(k)} \right\}, \tag{19}$$

where $V_i^c = V_1 \oplus \ldots \oplus V_{i-1} \oplus V_{i+1} \oplus \ldots \oplus V_I$ and π_X is the orthogonal projection onto $X \subset V$. We can solve this problem by using a thin stripe line around the block u_i:

$$u_i^{(k+1)} = \underset{u_i \in V_i \oplus \tilde{V}_i}{\arg\min} \{\, J(u_i) \;:\; \pi_{\tilde{V}_i}(u_i) = \tilde{U}_i^{(k)} \,\}, \tag{20}$$

where $\tilde{U}_i^{(k)} = U_i^{(k)}|_{\tilde{V}_i}$. Note that Ω_i is a subdomain of Ω and $V_i = V|_{\Omega_i}$. The additional thin stripe line around Ω_i is denoted as $\tilde{\Omega}_i$, i.e., $\Omega \setminus \Omega_i \supset \tilde{\Omega}_i$ and $\partial(\Omega \setminus \Omega_i) \cap \partial\Omega_i = \partial\Omega_i \cap \partial\tilde{\Omega}_i$. Therefore, $\tilde{V}_i = V|_{\tilde{\Omega}_i}$. Langer et al. [22] solved the equality constraints in (20) with Bregmanized operator splitting (BOS) [32]; see Section 3.2 for Bregmanized Domain Decomposition (BDD) in more details. Although BDD [22] easily handles the constraints in (20), as shown in Fig. 4 and Fig. 5, the BDD algorithm for (20) has some drawbacks. It requires many iterations to satisfy the constraint condition in (20).

Now, we consider the following different reformulation of (18):

$$u_i^{(k+1)} = \underset{u_i \in V_i, z_i}{\arg\min} \{\, \|u_i - b_i\|_2^2 + \mu\|z_i\|_1 \;:\; z_i = \nabla(u_i + U_i^{(k)}) \,\}, \tag{21}$$

where $b_i = b|_{\Omega_i}$. A solution of (21) could be found via an augmented Lagrangian based optimization method, such as the proximal linearized alternating direction (PLAD) method [28]. For this, at each subdomain Ω_i, we exploit the additional information $U_i^{(k)}$ into the gradient operator ∇ and the corresponding divergence operator div under the following equality condition on the whole domain Ω:

$$\langle p, \nabla u\rangle_W = \langle -\operatorname{div} p, u\rangle_{V_i \oplus V_i^c}, \tag{22}$$

where $W = (\oplus_{i=1}^I V_i) \times (\oplus_{i=1}^I V_i)$. Hence, in the following Definition 1 and 2, we introduce the domain dependent gradient operator $\bar{\nabla}_i u_i$ for $u_i \in V_i$ and the modified domain dependent divergence operator $\overline{\operatorname{div}}_i p_i$ for $p_i \in W_i$. We call the optimization methodology using the modified operators as the PDS (Primal-Dual Stitching) based method[1].

Definition 1 Let $\Omega = \cup_{i=1}^I \Omega_i$, where if $k \neq l$ then $\Omega_k \cap \Omega_l = \emptyset$, and $V_i = [0,255]^{|\Omega_i|}$ with $|\Omega_i| = M_i \times N_i$. Let us define the domain dependent gradient operator

$$\bar{\nabla}_i : V_i \to W_i, \tag{23}$$

as follows;

$$(\bar{\nabla}_i u_i)_{m,n} = ((\bar{\nabla}_i u_i)^x_{m,n}, (\bar{\nabla}_i u_i)^y_{m,n}), \tag{24}$$

where $W_i = V_i \times V_i$ and

$$(\bar{\nabla}_i u_i)^x_{m,n} = \begin{cases} u_i(m,n) - u_i(m-1,n) & \text{if } 1 < m \le M_i \\ u_i(m,n) - c_i^x(1,n) & \text{if } m = 1, \end{cases}$$

$$(\bar{\nabla}_i u_i)^y_{m,n} = \begin{cases} u_i(m,n) - u_i(m,n-1) & \text{if } 1 < n \le N_i \\ u_i(m,n) - c_i^y(m,1) & \text{if } n = 1. \end{cases}$$

[1] It is inspired from an image stitching method in [26].

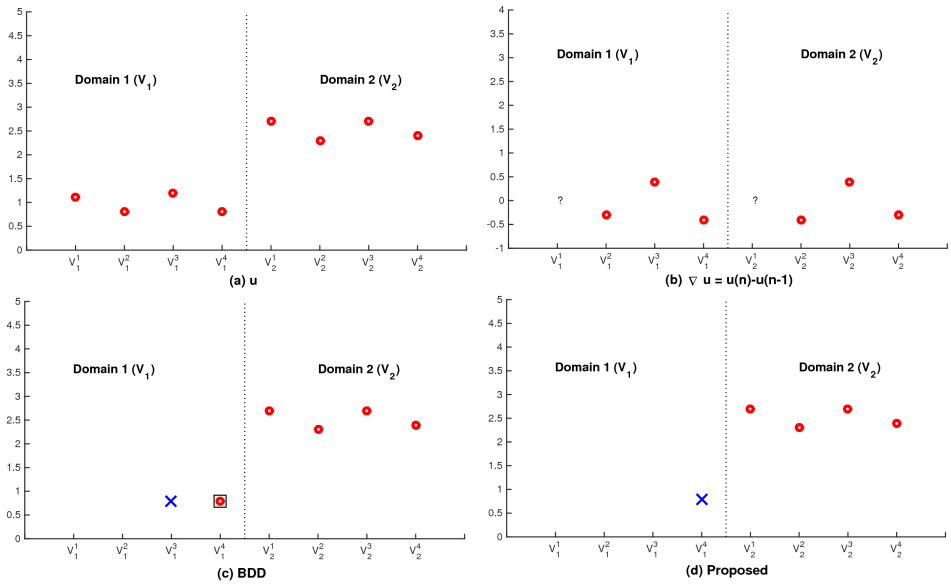

Fig. 1: Primal stitching of the backward difference gradient operator. (a) is the given data u. (b) is the backward gradient operator $\nabla u = u(n) - u(n-1)$. Here, $\nabla u = 0$ at V_1^1, due to the Neumann adiabatic boundary condition of the whole domain. However, we need to define ∇u at V_2^1. (c) is BDD (44) for primal variables. In BDD, ∇u at V_2^1 is naturally defined with unknown u at V_1^4 with constraint $\pi_{V_1^4}(u) = \square$. Also, the Neumann adiabatic boundary condition is used at V_1^4. Note that this additional pixel V_1^4 (i.e., $\pi_{V_1^4}(u) = \square$) can be extended to several pixels [22]. (d) is the proposed PDS for primal variables. In PDS, we simply use u at V_1^4 as a boundary data for the domain V_2. That is, we use a local Dirichlet boundary condition at V_2^1 with global Neumann adiabatic boundary condition at V_1^1 for the modified gradient in PDS.

We note that $c_i^x(1,n)$ is the adjacent pixel value in the negative x-direction on the location $(1,n)$ of i-th block. If the location $(1,n)$ of i-th block is the first pixel (in x-axis) of the given image domain Ω, then $c_i^x(1,n) = u_i(1,n)$ and thus $(\bar{\nabla}_i u_i)_{m,n}^x = 0$. Also, $c_i^y(m,1)$ is the adjacent pixel value in the negative y-direction on the location $(m,1)$ of i-th block. If the location $(m,1)$ of i-th block is the first pixel (in y-axis) of the given image domain Ω, then $c_i^y(m,1) = u_i(m,1)$ and thus $(\bar{\nabla}_i u_i)_{m,n}^y = 0$. That is, we use local Dirichlet boundary condition on each subdomain Ω_i and the global Neumann adiabatic boundary condition on the whole domain Ω; see also Fig. 1. Note that, for convenience, we let $\bar{\nabla} = [\bar{\nabla}_1, ..., \bar{\nabla}_I]$.

Now, we also modify the divergence operator $\mathrm{div} : W \to V$ to be the negative adjoint operator of the gradient operator $\bar{\nabla}$ on the whole domain Ω and thus (22) is preserved.

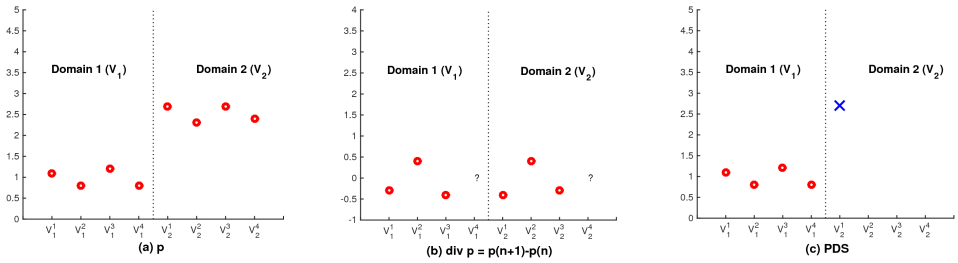

Fig. 2: Dual stitching of the proposed PDS for the forward difference divergence operator. In the TV based optimization algorithm, we are naturally required to calculate the forward difference operator $\operatorname{div} p = p(n+1) - p(n)$. Since div is the forward difference operator, we only need to know boundary information of the domain V_1. Note that as observed in Fig. 1, for ∇, we need to know boundary information of the domain V_2.

Definition 2 Let $\Omega = \cup_{i=1}^{I} \Omega_i$, where if $k \neq l$ then $\Omega_k \cap \Omega_l = \emptyset$, and $V_i = [0,255]^{|\Omega_i|}$ with $|\Omega_i| = M_i \times N_i$. Let us define the domain dependent divergence operator

$$\overline{\operatorname{div}}_i : W_i \to V_i \tag{25}$$

as follows;

$$\begin{aligned}(\overline{\operatorname{div}}_i p_i)_{m,n} = &\begin{cases} p_i^x(m+1,n) - d_i^x(1,n) & \text{if } m = 1 \\ p_i^x(m+1,n) - p_i^x(m,n) & \text{if } 1 < m < M_i \\ e_i^x(M_i,n) - p_i^x(m,n) & \text{if } m = M_i \end{cases} \\ + &\begin{cases} p_i^y(m,n+1) - d_i^y(m,1) & \text{if } n = 1 \\ p_i^y(m,n+1) - p_i^y(m,n) & \text{if } 1 < n < N_i \\ e_i^y(m,N_i) - p_i^y(m,n) & \text{if } n = N_i, \end{cases}\end{aligned} \tag{26}$$

where we set $d_i^x(1,n) = 0$ if $(1,n)$ of i-th block is on the boundary of the whole domain Ω otherwise $d_i^x(1,n) = p_i^x(1,n)$. Also, we set $d_i^y(m,1) = 0$ if $(m,1)$ of i-th block is on the boundary of the whole domain Ω otherwise $d_i^y(m,1) = p_i^y(m,1)$. Note that $e_i^x(M_i,n)$ is the adjacent p^x value in the positive x-direction on the location (M_i,n) of i-th block if (M_i,n) is not on the boundary of the whole domain Ω, otherwise $e_i^x(M_i,n) = 0$. Also, e_m^y is the adjacent p^y value in the positive y-direction on the location (m,N_i) if (m,N_i) is not on the boundary of the whole domain Ω, otherwise $e_i^y(m,N_i) = 0$. See also Fig. 2. For convenience, we let $\overline{\operatorname{div}} = [\overline{\operatorname{div}}_1, ..., \overline{\operatorname{div}}_I]$.

Although we define $\bar{\nabla} = [\bar{\nabla}_1, ..., \bar{\nabla}_I]$ and $\overline{\operatorname{div}} = [\overline{\operatorname{div}}_1, ..., \overline{\operatorname{div}}_I]$ to satisfy (22) on the whole domain Ω, the following Lemma shows that (22) is approximately satisfied on each subdomain Ω_i. Note that in Section 3.1, we show when the effects of this approximation is negligible.

Lemma 2 *Let $V = V_i \oplus V_i^c$ and $W = W_i \oplus W_i^c$, where $W_i = V_i \times V_i$, $V_i = V|_{\Omega_i}$, $\Omega_i = M_i \times N_i$, and $\Omega = \cup_{i=1}^{I} \Omega_i$. Then, (22) is approximately satisfied on each subdomain Ω_i:*

$$\langle p_i, \bar{\nabla}_i u_i \rangle_{W_i} = \langle -\overline{\operatorname{div}}_i p_i, u_i \rangle_{V_i} + X_i, \tag{27}$$

where X_i is the distortion terms on $\partial\Omega_i$.

Proof For simplicity, we characterize X_i in one dimension (x-axis):

$$\begin{aligned}\langle p_i, \bar{\nabla}_i u_i\rangle_{W_i} &= \sum_{m_i=2}^{M_i} p_i(m_i)[u_i(m_i)-u_i(m_i-1)] + p_i(1)[u_i(1)-c_i(1)] \\ &= p_i(M_i)u_i(M_i) - [p_i(M_i)-p_i(M_i-1)]u_i(M_i-1)\ldots - [p_i(2)-p_i(1)]u_i(1) - p_i(1)c_i(1) \\ &= \{p_i(M_i)u_i(M_i) - \sum_{m_i=2}^{M_i-1}[p_i(m_i+1)-p_i(m_i)]u_i(m_i) - p_i(2)u_i(1)\} + p_i(1)u_i(1) - p_i(1)c_i(1) \\ &= -\langle\overline{\mathrm{div}}_i p_i, u_i\rangle_{V_i} + e_i(M_i)u_i(M_i) - d_i(1)u_i(1) + p_i(1)u_i(1) - p_i(1)c_i(1)\end{aligned}$$

Now, we have characterize X_i for three different cases:

(a) $\partial\Omega_i = \partial\Omega$: $(c_i(1), d_i(1), e_i(M_i)) = (u_i(1), 0, 0)$

$$\langle p_i, \bar{\nabla}_i u_i\rangle_{W_i} = -\langle\overline{\mathrm{div}}_i p_i, u_i\rangle_{V_i}$$

(b) $\partial\Omega_i \cap \partial\Omega = \emptyset$: $(c_i(1), d_i(1), e_i(M_i)) = (u_{i-1}(M_{i-1}), p_i(1), p_{i+1}(1))$

$$\langle p_i, \bar{\nabla}_i u_i\rangle_{W_i} = -\langle\overline{\mathrm{div}}_i p_i, u_i\rangle_{V_i} + p_{i+1}(1)u_i(M_i) - p_i(1)u_{i-1}(M_{i-1})$$

(c) $\partial\Omega_i \cap \partial\Omega \neq \emptyset$ and $\partial\Omega_i \neq \partial\Omega$

(c-1) Left side of $\partial\Omega_i$ = Left side of $\partial\Omega$: $(c_i(1), d_i(1), e_i(M_i)) = (u_i(1), 0, p_{i+1}(1))$

$$\langle p_i, \bar{\nabla}_i u_i\rangle_{W_i} = -\langle\overline{\mathrm{div}}_i p_i, u_i\rangle_{V_i} + p_{i+1}(1)u_i(M_i)$$

(c-2) Right side of $\partial\Omega_i$ = Right side of $\partial\Omega$: $(c_i(1), d_i(1), e_i(M_i)) = (u_{i-1}(M_{i-1}), p_i(1), 0)$

$$\langle p_i, \bar{\nabla}_i u_i\rangle_{W_i} = -\langle\overline{\mathrm{div}}_i p_i, u_i\rangle_{V_i} - p_i(1)u_{i-1}(M_{i-1})$$

Note that when $\partial\Omega_i \cap \partial\Omega = \emptyset$, we get the worst boundary distortion $X_i = p_{i+1}(1)u_i(M_i) - p_i(1)u_{i-1}(M_{i-1})$. See also Example 1. ■

To understand clearly the proposed PDS method, we introduce one-dimensional example in the following:

Example 1 Let $u = [u(1), u(2), u(3), u(4), u(5)] \in V \subset \mathbb{R}^5$ and $p = [p(1), p(2), p(3), p(4), p(5)] \in W \subset \mathbb{R}^5$. $V = V_1 \oplus V_2$ where $u_1 = [u_1(1), u_1(2), u_1(3)] = [u(1), u(2), u(3)] \in V_1$, $u_2 = [u_2(1), u_2(2)] = [u(4), u(5)] \in V_2$. $W = W_1 \oplus W_2$ where $p_1 = [p_1(1), p_1(2), p_1(3)] = [p(1), p(2), p(3)] \in W_1$, $p_2 = [p_2(1), p_2(2)] = [p(4), p(5)] \in W_2$. Then we get the following in each subdomain:

$$\begin{aligned}\bar{\nabla}_1 u_1 &= [0, u_1(2)-u_1(1), u_1(3)-u_1(2)] = [0, u(2)-u(1), u(3)-u(2)] \in V_1, \\ \bar{\nabla}_2 u_2 &= [u_2(1)-\mathbf{u_1}(\mathbf{3}), u_2(2)-u_2(1)] = [u(4)-\mathbf{u}(\mathbf{3}), u(5)-u(4)] \in V_2 \\ \overline{\mathrm{div}}_1 p_1 &= [p_1(2) \quad , p_1(3)-p_1(2), \mathbf{p_2}(\mathbf{1})-p_1(3)] = [p(2) \quad , p(3)-p(2), \mathbf{p}(\mathbf{4})-p(3)] \in W_1, \\ \overline{\mathrm{div}}_2 p_2 &= [p_2(2)-p_2(1), \quad -p_2(2)] = [p(5)-p(4), \quad -p(5)] \in W_2\end{aligned}$$

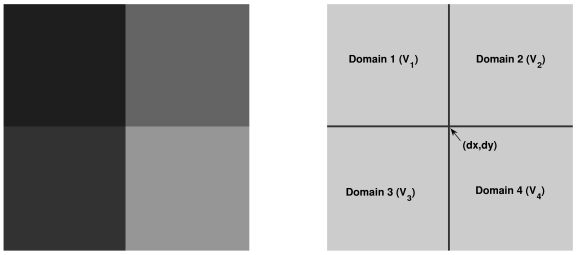

Fig. 3: For performance comparison of PDS and BDD for 2×2 block decomposition, we use 256×256 synthetic image (Left). The center of 2×2 block boundary $(128, 128)$ is shifted by (dx, dy), where $-10 \leq dx \leq 10$ and $-10 \leq dy \leq 10$ (Right). See Fig. 4, Fig. 5, and Table 1 for test results.

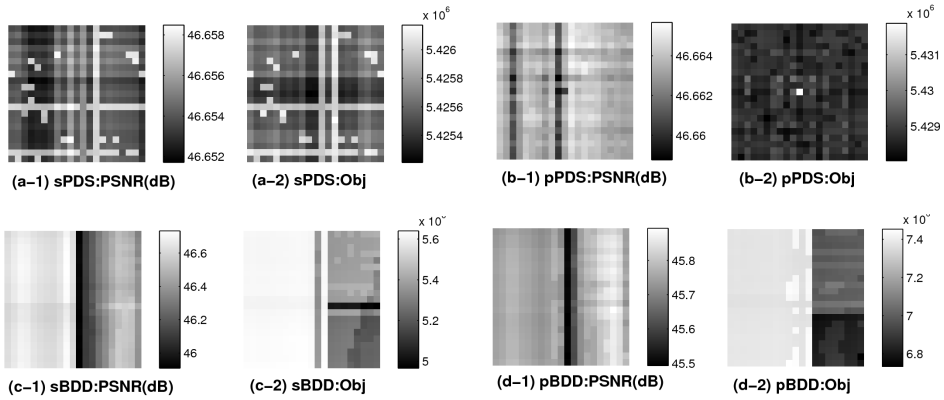

Fig. 4: Performance comparison of PDS and BDD for 2×2 block decomposition (see Fig. 3). The PSNR and Obj values at each pixel location corresponds to the shifted center (dx, dy) with $-10 \leq dx \leq 10$ and $-10 \leq dy \leq 10$. For BDD, we use one BOS iteration. The PDS based model obtains smaller variation in PSNR and Obj value than the BOS based method.

Also, we get the following on the whole domain V and W:

$$\begin{aligned} \bar{\nabla} u &= [\bar{\nabla}_1 u_1, \bar{\nabla}_2 u_2] \quad = [0 \qquad , u(2)-u(1), u(3)-u(2), u(4)-u(3), u(5)-u(4)] \\ \overline{\mathrm{div}} p &= [\overline{\mathrm{div}} p_1, \overline{\mathrm{div}} p_2] = [p(2) \quad , p(3)-p(2), p(4)-p(3), p(5)-p(4), \quad -p(5)] \end{aligned}$$

which satisfy $\langle p, \bar{\nabla} u \rangle_W = \langle -\overline{\mathrm{div}} p, u \rangle_V$. Note that we get $\langle p_i, \bar{\nabla}_i u_i \rangle_{W_i} \approx \langle -\overline{\mathrm{div}}_i p_i, u_i \rangle_{V_i}$. For instance, we have

$$\langle p_1, \bar{\nabla}_1 u_1 \rangle_{W_1} = \langle -\overline{\mathrm{div}}_1 p_1, u_1 \rangle_{V_1} - \mathbf{p}(\mathbf{4})\mathbf{u}(\mathbf{3}).$$

As shown in Fig. 1, the main difference between BDD [22] and the proposed PDS method is that BDD need additional equality constraints on the outside of the domain

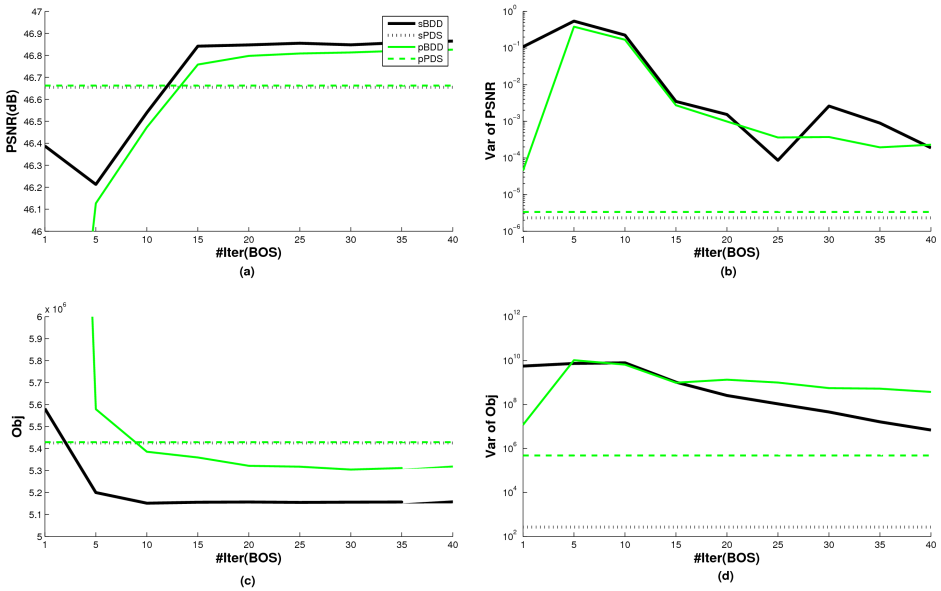

Fig. 5: Performance of BDD vs. the number of BOS iterations. (a) and (c) are the averaged value of 25 different cases (i.e., $-2 \leq dx \leq 2$ and $-2 \leq dy \leq 2$) in Fig. 3. (b) and (d) are the variance of each case. As we increase the number of BOS iterations, the PSNR and Objective value are getting better. To obtain comparable performance to the proposed PDS, we need more than 15 BOS iterations. However, as observed in (b) and (d), the variance is still larger than the PDS based model.

with the adiabatic Neumann boundary condition (Fig. 1 (c)). However, the proposed PDS uses information outside of the domain into the boundary condition and defines new gradient operator $\bar{\nabla}$ (24) and divergence operator $\overline{\text{div}}$ (26) (Fig. 1 (d) and Fig. 2 (c)). Therefore, the proposed PDS could remove the additional iterations to satisfy the equality constraint in (20). Due to this simplification process, as observed in Fig. 4, the proposed PDS method is more consistent with respect to changes of the domain in Fig. 3 when compared with the BDD method [22]. Note that Fig. 3 shows the experimental setting for Fig. 4, Fig. 5, and Table 1. In Fig. 5, we show the performance of BDD versus the number of BOS iterations. For comparable performance to the proposed PDS based model, BDD is required more than 15 BOS iterations. Although BDD obtains higher PSNR and lower Objective value, the BOS based model shows relatively large variance in PSNR and Objective value than the proposed PDS based model. In addition, Table 1[2] reports that the PDS based approach shows consistent performance irrespective of the number of iterations for the inner solver. Note that BDD also shows consistent performance between sequential method and parallel

[2] As shown in Theorem 3, the sequential method (i.e., BD-S) with $subMaxIt = 1$ and $\omega = 1$ corresponds to the method (i.e. PLAD) without using block decomposition.

Method	subMaxIt	Sequential	Parallel
	S in (43)	PSNR(dB)/Obj	PSNR(dB)/Obj
PDS	1	$46.7976/5.4283\times10^{6}$	$46.7976/5.4283\times10^{6}$
	10	$46.7977/5.4282\times10^{6}$	$46.7978/5.4282\times10^{6}$
	20	$46.7978/5.4283\times10^{6}$	$46.7986/5.4283\times10^{6}$
BDD	10	$46.9772/5.1602\times10^{6}$	$46.9773/5.1602\times10^{6}$

Table 1: Comparison of consistency of the PDS based method and the BDD method for the synthetic data in Fig. 3. Here, we set $\omega = 1$ for sequential algorithm, $\omega = 1/4$ for parallel algorithm, and $(dx, dy) = (0,0)$. Also we run 1000 iterations for all algorithms with different number of sub-iterations. The PDS based method is consistent regardless of choice of the number of iterations for the inner solver and the methodology (sequential vs. parallel). Here, the case with $subMaxIT = 1$ (i.e., $S = 1$ in (43)) corresponds to the case without using block decomposition (Theorem 3). Note that BDD shows consistent performance irrespective of the methodology. However, PDS and BDD show different results due to the different choice of parameters and different approach for block decomposition.

method. However, PDS and BDD obtain different results due to the different choice of parameters and different approach for block decomposition.

3.1 Block proxTV solver

There are various optimization methods, e.g. penalty method, alternating minimization algorithm (AMA), alternating direction method of multiplier (ADMM), proximal linearized alternating direction (PLAD) to implement block decomposition method proposed in Section 2. In this section, we introduce the PLAD method under the block decomposition framework for proxTV (1).

Let us start with the following reformulated proxTV (1):

$$\min_{u\in V,z} \{\|u-b\|_2^2 + \mu\|z\|_1 \ : \ \nabla u = z\}. \tag{28}$$

To find a solution of (28), we introduce Lagrangian based formulations, augmented Lagrangian and its linearized version. First, the augmented Lagrangian function for (28) is

$$\mathscr{L}_\alpha(u,z,p) := \|u-b\|_2^2 + \mu\|z\|_1 + \langle p, z-\nabla u\rangle + \frac{\alpha}{2}\|z-\nabla u\|_2^2, \tag{29}$$

where α is a nonnegative constant and when $\alpha = 0$, we get the Lagrangian function $\mathscr{L}_0(u,z,p)$ of (28). By using the Taylor expansion of $\mathscr{F}(u) := \frac{\alpha}{2}\|z-\nabla u\|_2^2$ of the augmented Lagrangian function (29) at $\breve{u}$, we have that

$$\mathscr{F}(u) = \mathscr{F}(\breve{u}) + \langle \nabla_u \mathscr{F}(\breve{u}), u-\breve{u}\rangle + \frac{1}{2}(u-\breve{u})^T \nabla_u^2 \mathscr{F}(\breve{u})(u-\breve{u}).$$

We replace $\mathscr{F}(u)$ of the augmented Lagrangian function $\mathscr{L}_\alpha(u,z,p)$ by the above second-order approximation, with D being a positive definite diagonal matrix approximating the Hessian matrix $\nabla_u^2\mathscr{F}(u)$, then we obtain

$$\begin{aligned}\mathscr{LL}_\alpha(u,z,p;\check{u},D) := &\|u-b\|_2^2+\mu\|z\|_1+\langle p, z-\nabla u\rangle \\ &+ \mathscr{F}(\check{u})+\langle\nabla_u\mathscr{F}(\check{u}), u-\check{u}\rangle+\frac{1}{2}(u-\check{u})^T D(u-\check{u}).\end{aligned} \tag{30}$$

We call $\mathscr{LL}_\alpha(u,z,p;\check{u},D)$ as the linearized augmented Lagrangian function with the generalized proximal function.

Now, let us start with the following min-max problem with the Lagrangian function of (28):

$$\max_p \min_{u\in V,z} \mathscr{L}_0(u,z,p), \tag{31}$$

where each term is separable except $\langle p,\nabla u\rangle$. However, by using the PDS method with $\sum_i\langle p_i,\bar{\nabla}_i u_i\rangle = \langle p,\bar{\nabla}u\rangle = \langle p,\nabla u\rangle$, we could decompose the Lagrangian function into small blocks:

$$\mathscr{L}_0(u,z,p)=\sum_i \mathscr{L}_0^i(u_i,z_i,p_i), \tag{32}$$

where

$$\mathscr{L}_0^i(u_i,z_i,p_i)=\|u_i-b_i\|_2^2+\mu\|z_i\|_1+\langle p_i, z_i-\bar{\nabla}_i u_i\rangle.$$

Note that, since ∇ in the i-th local block V_i has the Neumann adiabatic boundary condition, we could not replace $\bar{\nabla}_i$ with ∇. For that reason, it is more natural to use PDS for primal-dual block decomposition.

To analyze the structure of primal-dual block decomposition, we first consider the following unblocked virtual primal-dual optimization to find a saddle point of (31):

$$\begin{cases}\text{For } k=1,...,K\\ \quad (u^{(k+1)},z^{(k+1)},p^{(k+1)})=Opt(u^{(k)},z^{(k)},p^{(k)})\\ \text{End}\end{cases} \tag{33}$$

Actually, we could use simple sequential primal-dual block decomposition (i.e., BDS (2) with $\omega=1$) for (33):

$$\begin{cases}\text{For } k=1,...,K\\ \text{For } i=1,...,I\\ \quad (u_i^{(k+1)},z_i^{(k+1)},p_i^{(k+1)})=Opt_i(U_i^{(k)},Z_i^{(k)},P_i^{(k)})\\ \text{End}\\ u^{(k+1)}=\sum_{i=1}^I u_i^{(k+1)}\\ \text{End}\end{cases} \tag{34}$$

where $I\geq 2$ and

$$(U_i^{(k)},Z_i^{(k)},P_i^{(k)})=\sum_{j<i}(u_j^{(k+1)},z_j^{(k+1)},p_j^{(k+1)})+\sum_{j>i}(u_j^{(k)},z_j^{(k)},p_j^{(k)}).$$

Note that Opt_i is a virtual optimization method with PDS on the i-th block V_i. It is not easy to analyze the relation between unblocked optimization (33) and multi-blocked optimization (34). However, in the following Lemma, we show that when

Opt is carefully designed to satisfy some specific conditions then (33) is equal to the multi-blocked primal-dual optimization (34).

Lemma 3 (Pseudo Explicit Method: Dependency on previous information) *For any $u(m_i,n_i) \in V_i$ and $u(m_j,n_j) \in V_j$, let us assume that block index i is a non-decreasing sequence in the forward direction at each pixel location (m_i,n_i). That is, if $m_i \le m_j$ and $n_i \le n_j$ then $i \le j$. For simplicity, let u,z,p at (m,n) be ordered as $[a(1,m,n),a(2,m,n),a(3,m,n)] = [u(m,n),z(m,n),p(m,n)]$. We also update the variable a (i.e. u,z,p) in lexicographic order:*

$$
\begin{aligned}
&[a^{(k+1)}(1,1,1) \to a^{(k+1)}(2,1,1) \to a^{(k+1)}(3,1,1)] \to \\
&[a^{(k+1)}(1,2,1) \to a^{(k+1)}(2,2,1) \to a^{(k+1)}(3,2,1)] \to \\
&\dots \\
&[a^{(k+1)}(l-1,m,n) \to \mathbf{a}(\mathbf{l},\mathbf{m},\mathbf{n}) \to a^{(k)}(l+1,m,n)] \to \\
&[a^{(k)}(1,m+1,n) \to a^{(k)}(2,m+1,n) \to a^{(k)}(3,m+1,n)] \to \\
&\dots
\end{aligned}
$$

Now we say that $a(l,m,n)$ depends only on previous information when $a^{(k+1)}(l,m,n)$ is updated by using $a^{(k)}$ and $a^{(k+1)}(l_g,m_g,n_g)$ where $(l_g,m_g,n_g) < (l,m,n)$. Let us assume that Opt_i and Opt have one iteration[3]. Then unblocked virtual optimization Opt (33) can be replaced with multi-block decomposition method Opt_i in (34).

Proof In min-max problem (31), the only non-separable part is the following function:

$$\langle p, \nabla u \rangle = \langle -\operatorname{div} p, u \rangle,$$

where ∇ is the backward difference operator and div is the forward difference operator. Due to the dependency on previous information condition, *Opt* should be designed to use $a^{(k)}$ for the forward difference operator div. Also, since the block index *i* increases in forward direction (i.e., matched with the update direction of each pixel location), the information is always available for the backward difference operator ∇. In this case, we could replace ∇ and div in *Opt* with $\bar{\nabla}_i$ and $\overline{\text{div}}_i$ in Opt_i. That is, PDS is naturally processed under the previous information dependency condition. Now, we can replace *Opt* in (33) with $[Opt_1,...,Opt_I]$ in (34). ■

At this point, the question is which optimization method is transparent under the block decomposition in the sense of Lemma 3. Let us start with the well-known ADMM [10] for the virtual optimization algorithm $Opt(\cdot)$ in (33):

$$
ADMM(u^{(k)},z^{(k)},p^{(k)}) := \begin{cases} u^{(k+1)} = \arg\min\limits_{u \in V} \mathscr{L}_\alpha(u,z^{(k)},p^{(k)}) \\ z^{(k+1)} = \arg\min \mathscr{L}_\alpha(u^{(k+1)},z,p^{(k)}) \\ p^{(k+1)} = p^{(k)} + \alpha(z^{(k+1)} - \nabla u^{(k+1)}). \end{cases} \tag{35}
$$

[3] Note that one iteration condition is used in coordinate optimization only with primal variable to find a solution of fused Lasso problem [11].

In the first step of ADMM (35), we need to solve the following system equation to update $u^{(k+1)}$.

$$2(u-b)+\alpha\,\mathrm{div}(\nabla u)+\mathrm{div}(p^{(k)}+\alpha z^{(k)})=0. \tag{36}$$

Here, when we try to update $u^{(k+1)}(m,n)$ (i.e., u at (m,n)), we need $u^{(k+1)}(m+1,n)$ and $u^{(k+1)}(m,n+1)$ information in advance. It violates the dependency of previous information in Lemma 3 and thus we cannot replace div with $\overline{\mathrm{div}}_i$. Note that the first step of ADMM (35) is a typical implicit method, which does not have a closed form solution.

The interesting part is the second step of ADMM (35). This step is also an implicit method in general. However, it has a closed form solution for the TV based model:

$$z^{(k+1)}=\mathrm{shrink}(\nabla u^{(k+1)}-\frac{p^{(k)}}{\alpha},\frac{\mu}{\alpha}). \tag{37}$$

Here, the shrink operator has a separable structure and is defined as

$$\mathrm{shrink}(a,c)=\max(\|a\|_2-c,0)\frac{a}{\|a\|_2},$$

where $a=(a^x,a^y)\in\mathbb{R}^2$ and $\|a\|_2=\sqrt{(a^x)^2+(a^y)^2}$. Note that ∇ is the backward difference operator and satisfy the dependency of previous information property in Lemma 3. Therefore we could use block decomposition for (37), though the second step of ADMM (35) is an implicit scheme:

$$z_i^{(k+1)}=\mathrm{shrink}(\bar{\nabla}_i u_i^{(k+1)}-\frac{p_i^{(k)}}{\alpha},\frac{\mu}{\alpha}). \tag{38}$$

The third step of ADMM (35) is also satisfy Lemma 3, since ∇ is the backward difference operator and it can be replaced with $\bar{\nabla}_i$ without any harm in ADMM algorithm:

$$p_i^{(k+1)}=p_i^{(k)}+\alpha(z_i^{(k+1)}-\bar{\nabla}_i u_i^{(k+1)}).$$

Now, we need to replace the first step of ADMM with an explicit method to satisfy the dependency condition of previous information in Lemma 3. For this, we use linearized augmented Lagrangian function (30) for the first step of ADMM:

$$u^{(k+1)}=\underset{u\in V}{\arg\min}\,\mathscr{LL}_\alpha(u,z^{(k)},p^{(k)};u^{(k)},\frac{1}{\delta}I), \tag{39}$$

Since we use linearized augmented Lagrangian function (30), $u^{(k+1)}$ in (39) has a closed form solution and also satisfies the condition in Lemma 3:

$$u^{(k+1)}=\left(2+\frac{1}{\delta}\right)^{-1}(2b+\frac{1}{\delta}u^{(k)}-\mathrm{div}\,p^{(k)}-\alpha\,\mathrm{div}(z^{(k)}-\nabla u^{(k)})),$$

where the forward difference operator div is applied to the previous iteration $(u^{(k)}, z^{(k)}, p^{(k)})$. This optimization method is called the PLAD[4] algorithm [28]:

$$PLAD(u^{(k)}, z^{(k)}, p^{(k)}) := \begin{cases} u^{(k+1)} = \arg\min_{u \in V} \mathscr{L}\mathscr{L}_{\alpha}(u, z^{(k)}, p^{(k)}; u^{(k)}, \frac{1}{\delta} I) \\ z^{(k+1)} = \arg\min_{z} \mathscr{L}_{\alpha}(u^{(k+1)}, z, p^{(k)}) \\ p^{(k+1)} = p^{(k)} + \alpha(z^{(k+1)} - \nabla u^{(k+1)}). \end{cases} \tag{40}$$

We summarize the usefulness of PLAD (40) for block decomposition in the following theorem.

Theorem 3 *The PLAD algorithm* (40) *can be used for the virtual optimization Opt* (33) *with PLAD$_i$ (i.e., PLAD with PDS on the i-th block V_i) for block decomposition Opt$_i$* (34)*. That is, with the one iteration condition, PLAD is equal to the sequential block decomposition method with PLAD$_i$.*

Proof By (37), (38), and (39), the PLAD algorithm (40) satisfies the dependency of previous information condition in Lemma 3. Therefore, we could replace unblocked *PLAD* with multi-blocked *PLAD$_i$*. ■

In general, PLAD (40) is more appropriate for primal-dual block decomposition method (34) compared to the ADMM method. Based on Theorem 3, we introduce PDS based sequential primal-dual block decomposition with PLAD:

$$\begin{cases} \text{For } k = 1, ..., K \\ \text{For } i = 1, ..., I \\ (u_i^{(k+\frac{1}{2})}, z_i^{(k+1)}, p_i^{(k+1)}) = \text{PLAD}_i(U_i^{(k)}, Z_i^{(k)}, P_i^{(k)}) \\ \quad u_i^{(k+1)} = \omega u_i^{(k+\frac{1}{2})} + (1-\omega) u_i^{(k)} \\ \text{End} \\ \quad u^{(k+1)} = \sum_{i=1}^{I} u_i^{(k+1)} \\ \text{End} \end{cases} \tag{41}$$

where $\omega > 0$. We call this framework sPDS. Also in the same way we get the PDS based parallel primal-dual block decomposition framework with PLAD:

$$\begin{cases} \text{For } k = 1, ..., K \\ \text{ParFor } i = 1, ..., I \\ \quad (u_i^{(k+\frac{1}{2})}, z_i^{(k+1)}, p_i^{(k+1)}) = \text{PLAD}_i(U_i^{(k)}, Z_i^{(k)}, P_i^{(k)}) \\ \text{End} \\ u^{(k+1)} = \sum_{i=1}^{I} \omega u_i^{(k+\frac{1}{2})} + (1-\omega) u_i^{(k)} \\ \text{End} \end{cases} \tag{42}$$

where $\omega > 0$. We call this framework pPDS. Note that $(U_i^{(k)}, Z_i^{(k)}, P_i^{(k)})$ is the available previous information while we update (u_i, z_i, p_i). Note that the PDS based block optimization algorithm *PLAD$_i$* in (41) and (42) can have multiple iterations. See also Remark 4.

[4] Note that the original PLAD algorithm in [28] linearizes fidelity term and augmented term at the same time. However, since the fidelity term of the proxTV model (1) is a simple proximal term, the original PLAD method is up to constant equal to (40), which we called PLAD in this paper.

Remark 3 The global convergence to the saddle point in the min-max problem (28) of the PLAD algorithm (40) can be derived with the analysis in [18, Theorem 1] under the condition $0 < \delta < \frac{1}{\alpha\|\Delta\|_2}$. See also [33].

Remark 4 Note that, in general, $PLAD_i$ have multi iteration (i.e., $S > 1$):

$$PLAD_i(U_i^{(k)}, Z_i^{(k)}, P_i^{(k)}) := \lim_{s=1\to S} \begin{cases} u_i^{(s+1)} = \arg\min_{u_i \in V_i} \mathscr{L}\mathscr{L}_\alpha^i(u_i, z_i^{(s)}, p_i^{(s)}; u_i^{(s)}, \frac{1}{\delta} I) \\ z_i^{(s+1)} = \arg\min_{z_i} \mathscr{L}_\alpha^i(u_i^{(s+1)}, z_i, p_i^{(s)}) \\ p_i^{(s+1)} = p_i^{(s)} + \alpha(z_i^{(s+1)} - \bar{\nabla}_i u_i^{(s+1)}). \end{cases} \tag{43}$$

When $S > 1$, the convergence of PDS based block decomposition with PLAD (i.e., sPDS (41) and pPDS (42)) is unknown.

3.2 Bregmanized Domain Decomposition

In this section, we briefly describe the Bregmanized domain decomposition (BDD) method [22] to find a solution of (20).

By using Bregman operator splitting [32], we can handle the linear constraint $\pi_{\tilde{V}_i}(u_i) = \tilde{U}_i^{(k)}$ in (20) as follows:

$$\begin{aligned} u_i^{(s+1)} &= \arg\min_{u_i \in V_i \oplus \tilde{V}_i} J(u_i) + \frac{1}{2\delta}\|u_i - \{u_i^{(s)} - \delta\pi_{\tilde{V}_i}^T(\pi_{\tilde{V}_i}(u_i^{(s)}) - f_i^{(s)})\}\|_2^2 \\ f_i^{(s+1)} &= f_i^{(s)} + \tilde{U}_i^{(k)} - \pi_{\tilde{V}_i}(u_i^{(s+1)}) \end{aligned} \tag{44}$$

where $f_i^{(1)} = \tilde{U}_i^{(k)}$. For each BOS iteration, we need to solve the following reformulated problems:

$$\min_{u_i \in V_i \oplus \tilde{V}_i, d_i} \{X(u_i, d_i) \mid d_i = \nabla u_i\},$$

where

$$X(u_i, d_i) = \|u_i - b_i\|_2^2 + \mu\|d\|_1 + \frac{1}{2\delta}\|u_i - \{u_i^{(s)} - \delta\pi_{\tilde{V}_i}^T(\pi_{\tilde{V}_i}u_i^{(s)} - f_i^{(s)})\}\|_2^2.$$

Note that they use split Bregman with the Neumann adiabatic boundary condition for all sub-block Ω_i. The following is the split Bregman iteration to find a solution of (44):

$$\begin{cases} u_i^{(t+1)} = \arg\min_{u_i \in V_i \oplus \tilde{V}_i} X(u_i, d_i^{(t)}) + \frac{\alpha}{2}\|d_i^{(t)} - \nabla u_i - b_i^{(t)}\|_2^2 \\ d_i^{(t+1)} = \arg\min_{d_i} X(u_i^{(t+1)}, d_i) + \frac{\alpha}{2}\|d_i - \nabla u_i^{(t+1)} - b_i^{(t)}\|_2^2 \\ b_i^{(t+1)} = b_i^{(t)} + \nabla u_i^{(t+1)} - d_i^{(t+1)}. \end{cases} \tag{45}$$

As recommended in [22], when we apply the above BOS with split Bregman to BD-S (2) with $\omega = 1$, we get the sequential BDD. We call it sBDD. When we apply to BD-S (3) with $\omega = 1/I$, we get the parallel BDD. We call it pBDD.

As observed in Fig. 4 and Fig. 5, this BDD framework shows better performance in terms of PSNR and Objective values when we run sufficiently large number of BOS

Fig. 6: Dataset for the numerical experiments - seven different real images. The size of images varies from 256×256 to 6048×4032. The grayscale of images is $[0, 255]$.

iterations. Although we could obtain meaningful performance in terms of PSNR, the computational time is significantly increased as we increase BOS iterations. This is the main drawback of the BDD framework.

4 Numerical Experiments

In this section, we report numerical experiments for the block decomposition of proxTV (1). We compare sPDS (41) and pPDS (42) with the BOS based block decomposition method; sBDD and pBDD in Section 3.2. Note that all algorithms are implemented in MATLAB (version 8.3.0.532 (R2014a)). All runs are performed on a laptop with an Intel i7 CPU (2.9GHz) and 8GB memory. The operating system is 64-bit OS X. We use seven test images in Fig. 6. The image scale is from 256×256 to 6048×4032. Note that we add Gaussian additive noise with $\sigma = 20$ to all test images and set the TV regularization parameter $\mu = 10$ for all experiments. We terminate all the algorithms by the relative error stopping condition:

$$\|u^{(k+1)} - u^{(k)}\|_2 \leq \tau \|u^{(k)}\|_2 \tag{46}$$

with a maximum iteration $MaxIt = 300$ and $\tau = 10^{-4}$. For each denoising subproblem, PLAD (43) for PDS and split Bregmann (45) for BDD are applied and we use relative error stopping condition (46) with $\tau = 10^{-3}$ and $subMaxIt = 10$. In case of BDD, we need to set the number of Bregman iterations. As noticed in Fig. 4 and Fig. 5, when we choose small number of Bregman iterations, the linear constrains in (20) is not well satisfied. It causes artifacts on the boundary of each block and degrades the performance of BDD. Therefore, based on Fig. 5, we set 15 Bregmann iterations. We note that PSNR is defined by

$$10 \log_{10} \left(\frac{255^2 MN}{\|b - u^*\|_2^2} \right),$$

where MN is the size of $M \times N$ image, b is the given original image, and u^* is the recovered image. Other parameters are tuned for the best performance. For instance, in PLAD (43), we set $\alpha = 0.3$ and $\delta = 0.2$. For BDD, we set $\delta = 0.1$ in BOS (44) and $\alpha = 1.0$ in split Bregmann (45).

In Table 2, we report various block decompositions (from 1×1 to 16×16) for 'Cameraman' test image in Fig. 6. For pPDS and pBDD, we set $\omega = 1/I$ ($I = \#Block$) to guarantee the theoretical convergence. However, when we choose large number of blocks I, it causes extremely slow convergence or unusual behavior such as early termination with respect to relative error stopping condition. Therefore, we only use maximum iteration stopping condition for worst 16×16 case in parallel methods. In general, PSNRs of all methods are consistent irrespective of various block decompositions, except the pBDD method.

In Table 3, we report the overall performance of PDS for various acceleration rates $\omega \in [0.5, 1.5]$. Note that sPDS shows the best performance when $\omega \in [1.2, 1.3]$ and pPDS shows the best performance when $\omega = 1$, irrespective of domain decomposition method (e.g., 4×4 vs. 8×8). Although we don't have any theoretical guarantee for convergence when we choose large ω, we empirically observed that the proposed PDS based block decomposition algorithm converges well for the proxTV problem (1).

In Figure 7, we compare performance for various acceleration rates ($0.7 \leq \omega \leq 1.3$). In case of sPDS (41), we have empirically observed that, as we increase ω in limited range $[0.7, 1.3]$, the number of iterations is decreasing. We use 'Lena' image and 4×4 block decomposition scheme. Note that when we use Jacobi-type iteration (i.e., $\omega = 1.0$) for pPDS (42), we obtain the best performance, although we can not guarantee the convergence for this chosen parameter.

In Figures 8-13, we use 'Cameraman','Boat', 'House','Lena', 'Man', and 'Building' test images for 8×8, 4×4, 2×2, 8×2, 8×8, 4×4 block decompositions. The proposed PDS based approach outperforms BDD based schemes in terms of CPU time.

5 Conclusion

In this paper, we have introduced primal-dual block decomposition method with the primal-dual stitching. The proposed PDS based block decomposition method shows consistent performance irrespective of domain decomposition type and image type. Note that, although we do not know the convergence of parallel block decomposition method BD-P (3) when $\omega > 1/I$, it has a strong advantage in parallel processing of large scale problems. The future work is to implement the proposed PDS based BD-P method on parallel processing machine.

Acknowledgements The authors would like to thank the anonymous referees for their detailed comments to improve this paper. Chang-Ock Lee was supported by the National Research Foundation of Korea (NRF-2011-0015399). Hyenkyun Woo was supported by the New Professor Research Program of KOREATECH and Basic Science Program through the NRF of Korea funded by the Ministry of Education (NRF-2015R101A1A01061261). Sangwoon Yun was supported by the TJ Park Science Fellowship of

Domain	sPDS	pPDS	sBDD	pBDD
	PSNR/TIME/ITER	PSNR/TIME/ITER	PSNR/TIME/ITER	PSNR/TIME/ITER
1x1	44.41/1.1/10	-	-	-
1x2	44.41/0.8/10	44.42/1.7/20	44.38/10.9/ 5	44.39/17.3/ 8
1x4	44.41/0.8/10	44.42/3.2/39	44.34/8.8/ 5	44.36/26.2/15
1x8	44.41/0.8/ 9	44.42/6.6/75	44.39/9.6/ 5	44.44/45.9/24
1x16	44.41/0.9/ 9	44.42/15.4/145	44.39/13.6/ 5	44.48/99.4/37
2x1	44.41/0.9/10	44.42/1.9/20	44.28/13.4/ 6	44.29/19.7/ 9
2x2	44.24/1.0/10	44.26/4.0/39	44.32/15.2/ 5	44.35/42.5/14
2x4	44.41/1.0/ 9	44.42/8.0/76	44.43/13.0/ 5	44.48/60.7/24
2x8	44.41/1.0/ 9	44.42/16.9/147	44.41/22.9/ 7	44.51/118.0/37
2x16	44.41/1.0/ 9	44.42/34.4/286	44.39/22.1/ 5	44.46/223.0/52
4x1	44.41/0.9/10	44.42/3.6/38	44.40/10.2/ 5	44.42/30.6/15
4x2	44.21/1.0/ 9	44.21/8.5/76	44.33/14.6/ 5	44.38/70.1/24
4x4	44.34/1.2/ 9	44.35/20.0/148	44.41/15.8/ 5	44.49/114.1/37
4x8	44.38/1.2/ 9	44.39/41.0/289	44.40/19.3/ 5	44.44/195.3/52
4x16	44.40/1.5/ 9	44.44/62.3/300	44.41/26.3/ 5	43.75/301.6/58
8x1	44.41/0.9/ 9	44.42/7.5/74	44.45/10.8/ 5	44.50/51.3/24
8x2	44.42/1.1/ 9	44.43/18.8/146	44.46/20.5/ 6	44.55/123.2/37
8x4	44.41/1.3/ 9	44.43/44.3/288	44.47/24.4/ 6	44.47/208.4/52
8x8	44.41/1.8/ 9	44.45/73.5/300	44.45/31.4/ 6	43.77/298.2/58
8x16	44.41/2.5/ 9	44.44/129.7/300	44.42/40.1/ 5	41.39/205.2/26
16x1	44.41/1.0/ 9	44.42/17.1/143	44.41/14.6/ 5	44.50/106.0/37
16x2	44.39/1.4/11	44.41/38.2/286	44.39/22.6/ 5	44.44/228.5/52
16x4	44.41/1.6/ 9	44.45/65.6/300	44.39/27.0/ 5	43.76/304.3/58
16x8	44.41/2.5/ 9	44.43/133.8/300	44.42/40.7/ 5	41.38/217.6/26
16x16	44.41/4.1/ 9	44.40/253.2/300	44.41/72.7/ 6	42.31/2326/ 300

Table 2: Performance comparison of PDS and BDD for various block decompositions. Note that we decompose from 1×1 to 16×16. The 'Cameraman' image in Fig. 6 is used. For parallel version, we use $\omega = 1/I$ and for sequential version, we use $\omega = 1$. Here, the maximum iteration is set as 300. The BOS iteration for BDD (44) is 15. Both methods, PDS and BDD, show consistent PSNR, except pBDD.

POSCO TJ Park Foundation and Basic Science Research Program through NRF funded by the Ministry of Science, ICT & Future Planning (2012R1A1A1006406,2014R1A1A2056038).

References

1. A. BECK AND M. TEBOULLE, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, SIAM J. Imaging Sciences, **2**, 183-202 (2009).
2. A. BECK AND M. TEBOULLE, Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems, IEEE Trans. on Image Processing, **18**, 2419-2434 (2009).
3. D. P. BERTSEKAS AND J. N. TSITSIKLIS, Parallel and distributed computation, Prentice Hall (1989).
4. C. CARSTENSEN, Domain decomposition for a non-smooth convex minimization problems and its application to plasticity, Numerical Linear Algebra with Applications, **4**, 177-190 (1998).
5. P. COMBETTES AND W. WAJS, Signal recovery by proximal forward-backward splitting, Multiscale Model. Simul., **4**, 1168-1200 (2005).
6. A. CHAMBOLLE, An algorithm for total variation minimization and applications, J. Math. Imaging and Vision, **20**, 89-97 (2004).
7. A. CHAMBOLLE AND T. POCK, A first-order primal-dual algorithm for convex problems with applications to imaging, J. Math. Imaging and Vision, **40**, 120-145 (2011).
8. T. F. CHAN AND J. SHEN, Image Processing and Analysis, SIAM (2005).
9. H. CHANG, X.-C. TAI, L.-L. WANG, AND D. YANG, Convergence rate of overlapping domain decomposition methods for the Rudin-Osher-Fatemi model based on a dual formulation, SIAM J. Imaging Sciences, **8**, 564-591, (2015).

#Block	Method	ω	Cameraman	Boat	House	Lena	Man	Peppers
			PSNR/Time/It	PSNR/Time/It	PSNR/Time/It	PSNR/Time/It	PSNR/Time/It	PSNR/Time/It
4x4	sPDS	1.0	44.38/1.2/ 9	47.75/3.8/ 8	45.21/1.2/ 9	48.60/3.7/ 8	51.25/21.5/ 8	44.76/1.2/ 9
		1.1	44.38/1.1/ 8	47.76/3.7/ 8	45.21/1.1/ 8	48.62/3.7/ 8	51.27/21.6/ 8	44.75/1.1/ 8
		1.2	44.38/1.1/ 8	47.76/3.3/ 7	45.22/1.0/ 7	48.63/3.3/ 7	51.27/19.4/ 7	44.75/1.1/ 8
		1.3	44.38/1.2/ 8	47.77/3.4/ 7	45.22/1.0/ 7	48.63/3.5/ 7	51.28/19.4/ 7	44.76/1.1/ 8
		1.4	44.38/1.3/ 8	47.77/3.5/ 7	45.23/1.2/ 8	48.64/3.6/ 7	51.29/19.7/ 7	44.75/1.2/ 8
		1.5	44.38/1.6/10	47.77/3.7/ 7	45.23/1.4/ 9	48.64/4.2/ 8	51.29/20.5/ 7	44.76/1.3/ 8
	sBDD	1.0	44.40/18.9/ 5	47.88/46.4/ 5	45.36/17.9/ 5	48.95/43.4/ 5	51.52/167.3/ 5	44.74/18.3/ 5
	pPDS	0.5	44.39/2.5/18	47.75/8.3/17	45.21/2.4/18	48.59/8.1/17	51.25/46.6/17	44.76/2.5/18
		0.6	44.39/2.1/15	47.75/6.9/14	45.21/2.1/15	48.59/6.6/14	51.24/38.1/14	44.76/2.1/15
		0.7	44.39/1.8/13	47.75/5.8/12	45.21/1.7/13	48.59/5.8/12	51.24/33.0/12	44.76/1.8/13
		0.8	44.39/1.6/12	47.75/5.3/11	45.20/1.5/11	48.60/5.3/11	51.25/29.9/11	44.76/1.5/11
		0.9	44.39/1.4/10	47.75/4.4/ 9	45.21/1.4/10	48.61/4.7/10	51.24/25.0/ 9	44.76/1.4/10
		1.0	44.38/1.3/ 9	47.75/4.0/ 8	45.21/1.3/ 9	48.60/3.9/ 8	51.25/22.2/ 8	44.76/1.2/ 9
	pBDD	-	43.84/110.5/30	46.85/263.4/30	44.10/105.3/30	47.33/249.9/30	50.25/969.1/30	43.82/108.1/30
8x8	sPDS	1.0	44.41/1.7/ 9	47.76/4.7/ 8	45.22/1.7/ 9	48.60/4.6/ 8	51.26/25.2/ 8	44.76/1.7/ 9
		1.1	44.41/1.6/ 8	47.76/4.2/ 7	45.23/1.6/ 8	48.62/4.7/ 8	51.26/21.1/ 7	44.76/1.6/ 8
		1.2	44.41/1.5/ 7	47.77/4.3/ 7	45.23/1.4/ 7	48.63/4.3/ 7	51.28/21.3/ 7	44.76/1.4/ 7
		1.3	44.41/1.6/ 7	47.77/3.9/ 6	45.24/1.5/ 7	48.64/4.4/ 7	51.29/21.8/ 7	44.76/1.5/ 7
		1.4	44.42/1.8/ 8	47.77/4.6/ 7	45.25/1.7/ 8	48.64/4.6/ 7	51.29/22.5/ 7	44.76/1.8/ 8
		1.5	44.42/2.5/11	47.77/5.3/ 8	45.25/2.5/12	48.64/6.3/10	51.30/26.1/ 8	44.76/2.1/ 9
	sBDD	1.0	44.45/31.7/ 6	47.87/68.8/ 6	45.32/24.1/ 5	48.84/54.9/ 5	51.48/235.9/ 6	44.72/24.8/ 5
	pPDS	0.5	44.43/3.6/18	47.76/11.0/17	45.22/3.4/17	48.59/10.1/17	51.24/51.8/16	44.76/3.3/17
		0.6	44.42/3.1/15	47.76/9.0/14	45.22/3.0/15	48.59/8.4/14	51.25/45.2/14	44.76/2.9/15
		0.7	44.42/2.6/13	47.76/7.1/12	45.22/2.5/13	48.59/7.2/12	51.25/38.7/12	44.76/2.5/13
		0.8	44.42/2.2/11	47.76/6.0/10	45.22/2.1/11	48.59/6.0/10	51.25/32.2/10	44.76/2.1/11
		0.9	44.42/2.0/10	47.76/5.4/ 9	45.22/2.0/10	48.59/5.4/ 9	51.25/29.1/ 9	44.77/1.9/10
		1.0	44.42/1.8/ 9	47.76/4.8/ 8	45.22/1.8/ 9	48.60/4.9/ 8	51.26/25.9/ 8	44.76/1.8/ 9
	pBDD	-	42.56/154.6/30	45.55/338.5/30	42.65/141.3/30	45.74/323.7/30	48.66/1146.2/30	42.52/146.7/30

Table 3: Performance comparison for various image data and various relaxation parameters ω. For sPDS, we test for $\omega \in [1.0, 1.5]$ and for pPDS, we test for $\omega \in [0.5, 1.0]$. For pBDD, we set $\omega = 1/I$, i.e., $1/16$ for 4×4 blocks and $1/64$ for 8×8 blocks. Note that sPDS shows the best performance when $\omega \in [1.2, 1.3]$ and pPDS shows the best performance when $\omega = 1$. Regarding PDS, although we don't have any theoretical guarantee for convergence, we empirically observed that the SOR-type iteration of sPDS and Jacobi-type iteration of pPDS converge well for the proxTV problem (1).

10. E. ESSER, "Applications of Lagrangian-based alternating direction methods and connections to split Bregman", Preprint, 2009.
11. J. FRIEDMAN, T. HASTIE, H. HÖFLING, AND R. TIBSHIRANI, Pathwise coordinate optimization, The Annals of Applied Statistics, **1**, 302-332, (2007).
12. M. FORNASIER, Domain decomposition methods for linear inverse problems with sparsity constraints, Inverse Problems, **23**, 2505-2526, (2007).
13. M. FORNASIER, A. LANGER, AND C.-B. SCHÖNLIEB, A convergent overlapping domain decomposition method for total variation minimization, Numerische Mathematik, **116**, 645-685 (2010).
14. M. FORNASIER AND C.-B. SCHÖNLIEB, Subspace correction methods for total variation and l1-minimization, SIAM J. Numer. Anal., **47**, 3397-3428 (2009).
15. D. GABAY AND B. MERCIER, A dual problem for the solution of nonlinear variational problems via finite element approximations, Computers and Mathematics with Applications, **2**, 17-40 (1976).
16. T. GOLDSTEIN, AND S. OSHER, The split Bregman method for l1 regularized problems, SIAM J. Imaging Sci., **2**, 323-343 (2009).
17. L. A. HAGEMAN AND T. A. PORSCHING, Aspects of nonlinear block successive overrelaxation, SIAM J. on Numerical Analysis, **2**, 316-335 (1975).

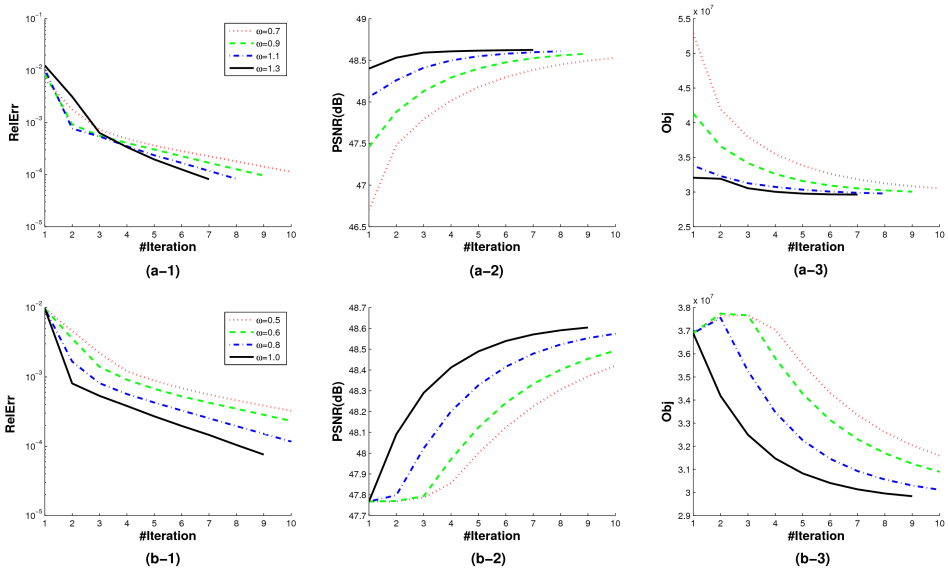

Fig. 7: Performance comparison with respect to relaxation parameter ω. The 'Lena' image with 4×4 block decomposition is used. (a) is for sPDS and (b) is for pPDS. We observed that, as we increase $\omega > 1$, the convergence rate becomes faster.

18. B. He, L. Z. Liao, D. Han and H. Yang, A new inexact alternating directions method for monotone variational inequalities, Math. Program., 92 (2002), 103-118.
19. M. Hintermüller and A. Langer, Subspace correction methods for a class of non-smooth and non-additive convex variational problems with mixed L^1/L^2 data-fidelity in image processing, SIAM J. Imaging Sciences, **6**, 2134-2173 (2013).
20. M. Hintermüller and A. Langer, Non-overlapping domain decomposition methods for dual total variation based image denoising, **62**, 456-481 (2015).
    Subspace correction methods for a class of non-smooth and non-additive convex variational problems with mixed L^1/L^2 data-fidelity in image processing, SIAM J. Imaging Sciences, **6**, 2134-2173 (2013).
21. M. Kang, S. Yun, and H. Woo, Two-level convex relaxed variational model for multiplicative denoising, SIAM J. Imaging Sciences, **6**, 875-903 (2013).
22. A. Langer, S. Osher, and C.-B. Schönlieb, Bregmanized domain decomposition for image restoration, J. Sci. Comp., **54**, 549-576 (2013).
23. L. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation based noise removal algorithms. Physica D, **60**, 259-268 (1992).
24. P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, J. Optim. Theory Appl., **109**, 475-494 (2001).
25. P. Tseng and S. Yun, A coordinate gradient descent method for nonsmooth separable minimization, Math. Prog. (Ser. B), **117**, 387-423 (2009).
26. W. Wang and M.K. Ng, A Variational approach for image stitching I, SIAM J. Imaging Sci., **6**, 1318-1344, (2013).
27. Z. Wen, W. Yin, and Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, Math. Prog. Comp., **4**, 333-361 (2012).
28. H. Woo and S. Yun, Proximal linearized alternating direction method for multiplicative denoising, SIAM J. Sci. Comp., **35**, B336-B358 (2013).

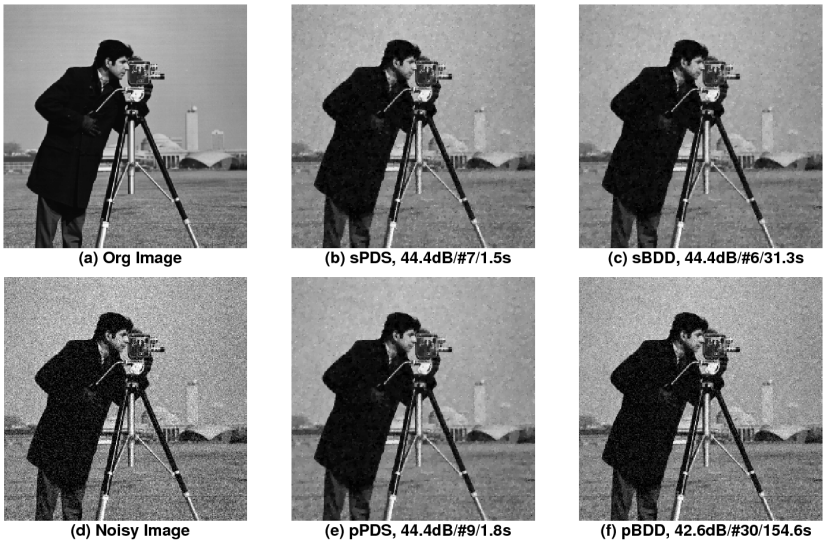

Fig. 8: Performance comparison between PDS and BDD. The 'Cameraman' image is used and 8×8 block decomposition is used. For sPDS, we set $\omega = 1.2$ and, for pPDS, $\omega = 1.0$. Note that for pBDD, we set $\omega = 1/64$. The proposed PDS method requires small number of iterations and computation time.

29. C. WU AND X.-C. TAI, Augmented Lagrangian method, Dual methods and Split-Bregman Iterations for ROF, vectorial TV and higher order models, SIAM J. Imaging Science, **3**, 300-339 (2010).
30. J. XU, X.-C. TAI AND L.-L. WANG, A two-level domain decomposition method for image restoration, Inverse Problems and Imaging, **4** 523-545 (2010).
31. S. YUN AND H. WOO, Linearized proximal alternating minimization algorithm for motion deblurring by nonlocal regularization, Pattern Recognition, **44**, 1312-1326 (2011).
32. X. ZHANG, M. BURGER, X. BRESSON, AND S. OSHER, Bregmanized nonlocal regularization for deconvolution and sparse reconstruction, SIAM J. Img. Sci., **3** 253-276 (2010).
33. X. ZHANG, M. BURGER, AND S. OSHER, A unified primal-dual algorithm framework based on Bregman iteration, J. Sci. Comput., 46 (2011), 20-46.
34. M. ZHU AND T. CHAN, An efficient primal-dual hybrid gradient algorithm for total variation image restoration, UCLA CAM-Report, (2008).

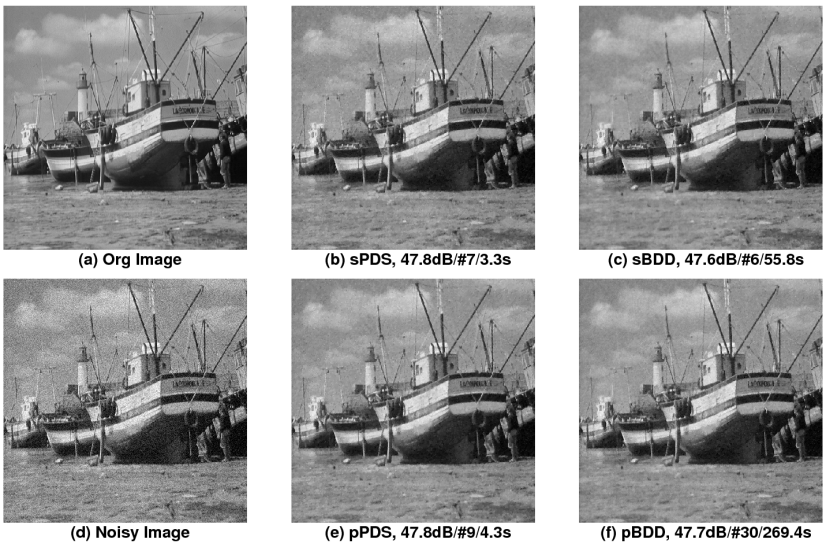

Fig. 9: Performance comparison between PDS and BDD. The 'Boat' image is used and 4×4 block decomposition is used. For sPDS, we set $\omega = 1.2$ and, for pPDS, $\omega = 1.0$. Note that for pBDD, we set $\omega = 1/16$. The proposed PDS method requires small number of iterations and computation time.

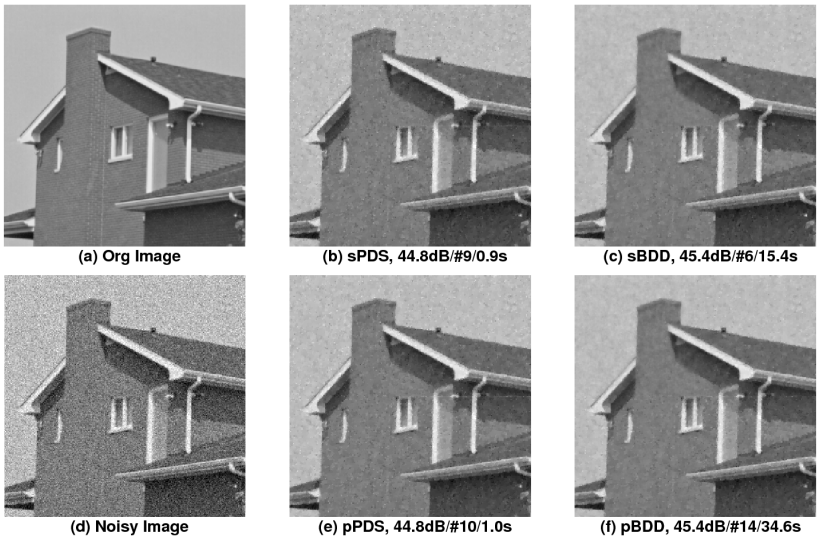

Fig. 10: Performance comparison between PDS and BDD. The 'House' image is used and 2×2 block decomposition is used. For sPDS, we set $\omega = 1.2$ and, for pPDS, $\omega = 1.0$. Note that for pBDD, we set $\omega = 1/4$. The proposed PDS method requires small number of iterations and computation time.

Fig. 11: Performance comparison between PDS and BDD. The 'Lena' image is used and 8×2 block decomposition is used. For sPDS, we set $\omega = 1.2$ and, for pPDS, $\omega = 1.0$. Note that for pBDD, we set $\omega = 1/16$. The proposed PDS method requires small number of iterations and computation time.

Fig. 12: Performance comparison between PDS and BDD. The 'Man' image is used and 8×8 block decomposition is used. For sPDS, we set $\omega = 1.2$ and, for pPDS, $\omega = 1.0$. Note that for pBDD, we set $\omega = 1/64$. The proposed PDS method requires small number of iterations and computation time.

Fig. 13: Performance comparison between PDS and BDD. The 'Building' image is used and 4×4 block decomposition is used. Note that for pBDD, we set $\omega = 1/16$ and for others, we set $\omega = 1.0$. The proposed PDS method requires small number of iterations and computation time.