A multiplicative iterative algorithm for box-constrained penalized likelihood image restoration

Raymond H. Chan and Jun Ma

Abstract—**Image restoration is a computationally intensive problem as a large number of pixel values have to be determined. Since the pixel values of digital images can attain only a finite number of values (e.g. 8-bit images can have only 256 gray levels), one would like to recover an image within some dynamic range. This leads to the imposition of box constraints on the pixel values. The traditional gradient projection methods for constrained optimization can be used to impose box constraints, but they may suffer from either slow convergence or repeated searching for active sets in each iteration. In this paper, we develop a new box-constrained multiplicative iterative (BCMI) algorithm for box-constrained image restoration. The BCMI algorithm just requires pixel-wise updates in each iteration, and there is no need to invert any matrices. We give the convergence proof of this algorithm and apply it to TV image restoration problems where the observed blurry images contain Poisson, Gaussian or salt-and-pepper noises.**

Keywords: Box constraints, image restoration, penalized likelihood optimization, box-constrained multiplicative iterative algorithm, global convergence.

I. Introduction

Statistical image restoration methods provide effective approaches to denoise and deblur images contaminated with noise and blur as they can accommodate accurate noise models and blurring schemes into the restoration. Commonly used noise models include Gaussian noise, Poisson noise and salt-and-pepper (or impulsive) noise [1], where the salt-and-pepper noise can be modeled by the Laplace distribution (e.g. [2]).

Since the pixel values of digital images can attain only a finite number of values (e.g. 8-bit images can have only 256 gray levels), one would like to restore an image within some dynamic range, e.g. in [0, 255]. This leads to the imposition of box constraints. We will see later in the numerical section that the box-constrained restored images can have as high as 1.1dB improvement in PSNR (see (27) for definition) over the restorations obtained by simply projecting non-constrained restorations onto the dynamic range. This paper considers the box-constrained penalized likelihood (PL) (or, equivalently, maximum a posteriori (MAP)) image restoration methods. We will develop an efficient multiplicative iterative (MI) algorithm for box-constrained PL image restoration.

R. H. Chan is with the Department of Mathematics, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong. The research was supported in part by HKRGC Grant CUHK400510 and DAG Grant 2060408. Email: rchan@math.cuhk.edu.hk

J. Ma is with the Department of Statistics, Macquarie University, Australia. Email: jun.ma@mq.edu.au

Importance of box-constraints is also demonstrated in [3] where the improvement in PSNR can be as high as 2dB. Of course, the gain depends on the image: the more pixels with values that are close to 0 or 255, the more gain will be in signal-to-noise ratio. In this paper, we will demonstrate that our constrained image restoration method has fast convergence when compared with some of the existing solvers.

Box constraints have been considered for image restoration by many authors, such as [3], [4] and [5], and the references therein. However, most of these methods have the following two major weaknesses: (i) they mostly consider the Gaussian noise (equivalently least-squares restoration), and (ii) they are developed based on gradient projection algorithms, which can be either slow converging or require identification of the active sets in every iteration. Identifying the active set in each iteration can be a significant computational burden for image processing. In contrast, the method developed in this paper completely avoids active sets and is remarkably easy to implement.

Our method is formulated from the general principle of PL, where the unknown true image is estimated by minimizing the penalized negative log-likelihood function. The penalty function (equivalent to the negative log prior density function in the context of MAP) is used to restrict the restored image such that it satisfies certain local smoothness conditions. The box-constrained problem is solved by a multiplicative iterative (MI) algorithm extended from the MI algorithm in [6] which has been applied successfully to positively constrained tomographic reconstruction [7]. This new MI algorithm can handle a general noise distribution such as Gaussian, Poisson and Laplace, and any penalty function as long as its first derivative is available, making it feasible for many well known penalty functions in image processing. Note that, however, for some edge preserving penalties, such as the TV [8] and Gaussian Markov random field (GMRF) [9] penalties, smoothing of their first derivatives is necessary as they are not differentiable at some points; see Section IV for TV regularization examples.

The new MI algorithm requires just pixel-wise updates in each iteration, making it extremely easy to implement. There is no need to invert any matrices, and hence there is no need to impose any boundary conditions on the observed images. Boundary conditions are necessitated when fast Fourier transformations or discrete cosine transforms are involved in inverting matrices, such as in [10], [11], [5]. We will prove that this algorithm is globally convergent under certain regular conditions. We will demonstrate through a simulation study that this method is efficient for TV regularized problems under the Gaussian, Poisson and slat-and-pepper noise models, and is capable of producing better reconstructions than some existing TV regularized image restoration algorithms.

The rest of this paper is organized as follows. Section II develops the MI algorithm for box-constrained PL image processing. Section III provides the convergence property of this MI algorithm. Results of a simulation study are reported in Section V and conclusions are included in Section VI.

II. The method

To simplify discussions, all images will be lexicographically ordered into vectors. Thus any $p \times q$ image will be represented as a column vector in $\mathbb{R}^n$ where $n = pq$. The true image $\mathbf{x} = (x_1, \cdots, x_n)^\top$ and the blurred and noisy observed image $\mathbf{y} = (y_1, \cdots, y_n)^\top$ are related via

$$\mathbb{E}(\mathbf{y}) = A\mathbf{x}, \tag{1}$$

where A is the blurring matrix of dimension $n \times n$, $\mathbb{E}$ is the expectation operator and $\top$ is the matrix transpose operator. Matrix A is, in fact, the discretized point spread function, which represents the response of the image system to a point source.

For any true digital image $\mathbf{x}$, its pixel can attain only a finite number of values. Hence, it is natural to require all pixel values of the restored image to lie in a certain interval $[0, b]$. Such a constraint is called the *box constraint*. For statistical image restoration using PL and the box constraint, our aim is to recover $\mathbf{x}$ by minimizing the following constrained optimization problem:

$$\min_{0 \le \mathbf{x} \le b} \Phi(\mathbf{x}) \equiv \min_{0 \le \mathbf{x} \le b} \left\{ -\sum_{i=1}^{n} l_i(\mu_i) + \lambda \mathcal{J}(\mathbf{x}) \right\}. \tag{2}$$

Here $l_i(\mu_i)$ denotes the log-likelihood function (as a function of $\mu_i = \mathbb{E}(y_i)$) corresponding to observation y_i, $\lambda \ge 0$ is the smoothing parameter, $\mathcal{J}(\mathbf{x})$ is the penalty function, and the inequality on $\mathbf{x}$ is to be interpreted componentwise. Note that $\mathcal{J}(\mathbf{x})$ is usually specified in a way so that it can describe the underlying spatial structure of image $\mathbf{x}$. Parameter λ is used to balance two possibly conflicting image quality measurements: data mismatch measured by the negative log-likelihood and spatial dependence measured by the penalty function. Note that from (1), $\mu_i = A_i\mathbf{x}$, where A_i is the ith row of matrix A.

Let $\Phi'(\mathbf{x})$ denote the derivative of Φ with respect to $\mathbf{x}$, and $\Phi'_i(\mathbf{x})$ the derivative of Φ with respect to x_i. The Karush-Kuhn-Tucker (KKT) condition defining the solution of (2) is:

$$\begin{cases} \Phi'_i(\mathbf{x}) = 0 & \text{if } 0 < x_i < b, \\ \Phi'_i(\mathbf{x}) > 0 & \text{if } x_i = 0, \qquad i = 1, \ldots, n. \\ \Phi'_i(\mathbf{x}) < 0 & \text{if } x_i = b, \end{cases} \tag{3}$$

These conditions lead to solving the equations

$$x_i \Phi'_i(\mathbf{x}) = 0, \quad i = 1, \ldots, n, \tag{4}$$

when considering the $x_i \ge 0$ constraint, and solving the equations

$$(b - x_i)\Phi'_i(\mathbf{x}) = 0, \quad i = 1, \ldots, n, \tag{5}$$

when considering the $x_i \le b$ constraint. We propose to solve equations (4) and (5) iteratively by adjusting the multiplicative iterative (MI) algorithm developed in [6]. In each iteration, our strategy is to first update *all* the x_i's using (4), ensuring that all $x_i \ge 0$. Then for those $x_i > b$, they are re-calculated using (5) so that all $x_i \le b$. The MI algorithm has been successfully implemented to positively constrained tomographic reconstructions [7] for Poisson noise and to positively constrained total variation penalized image restorations [12] for different noises. In this paper, we extend this algorithm to image restoration problems with box constraints under different noise models.

We first introduce some notations needed for the MI algorithm. For a constant c, let $[c]^+ = \max\{0, c\}$ and $[c]^- = \min\{0, c\}$, so that $c = [c]^+ + [c]^-$. Similarly for a function $a(x)$, let $a(x)^+$ and $a(x)^-$ be its positive and negative components respectively, such that $a(x) = a(x)^+ + a(x)^-$. Rearranging the terms in (4) such that both sides are non-negative, we have

$$\begin{aligned} & x_i \left(-\sum_{j=1}^{n} a_{ji} l'_j(\mu_j)^- + \lambda [\mathcal{J}'_i(\mathbf{x})]^+ \right) \\ & = x_i \left(\sum_{j=1}^{n} a_{ji} l'_j(\mu_j)^+ - \lambda [\mathcal{J}'_i(\mathbf{x})]^- \right). \end{aligned} \tag{6}$$

This equation immediately provides an updating scheme for $\mathbf{x}$:

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} \frac{\delta_{1i}^{(k)}}{\delta_{2i}^{(k)}}, \quad i = 1, \ldots, n. \tag{7}$$

In (7),

$$\delta_{1i}^{(k)} = \sum_{j=1}^{n} a_{ji} l'_j(\mu_j^{(k)})^+ - \lambda [\mathcal{J}'_i(\mathbf{x}^{(k)})]^- + \varepsilon \equiv g_{1i}^{(k)} + \varepsilon, \tag{8}$$

$$\delta_{2i}^{(k)} = -\sum_{j=1}^{n} a_{ji} l'_j(\mu_j^{(k)})^- + \lambda [\mathcal{J}'_i(\mathbf{x}^{(k)})]^+ + \varepsilon \equiv g_{2i}^{(k)} + \varepsilon, \tag{9}$$

where $\mu_j^{(k)} = A_j \mathbf{x}^{(k)}$ and the constants $\varepsilon \ge 0$ which is included to avoid possible zero denominators in (7) or (11). It will become clear later that the values for ε will not alter where the algorithm will converge to, but they will change its speed of convergence.

Clearly, if $x_i^{(k)} \ge 0$ then $x_i^{(k+\frac{1}{2})}$ given by (7) satisfies $x_i^{(k+\frac{1}{2})} \ge 0$. If all of these $x_i^{(k+\frac{1}{2})}$ are not greater than b, then we move to the line search step given by (18); otherwise, those $x_i^{(k+\frac{1}{2})}$, whose values are greater than b, are re-computed using (11) below. In fact, similar to (6), we can rearrange (5) as

$$\begin{aligned} & (b - x_i) \left(\sum_{j=1}^{n} a_{ji} l'_j(\mu_j)^+ - \lambda [\mathcal{J}'_i(\mathbf{x})]^- \right) \\ & = (b - x_i) \left(-\sum_{j=1}^{n} a_{ji} l'_j(\mu_j)^- + \lambda [\mathcal{J}'_i(\mathbf{x})]^+ \right), \end{aligned} \tag{10}$$

which gives

$$x_i^{(k+\frac{1}{2})} = b - (b - x_i^{(k)})\frac{\delta_{2i}^{(k)}}{\delta_{1i}^{(k)}}, \tag{11}$$

where $\delta_{1i}^{(k)}$ and $\delta_{2i}^{(k)}$ are given by (8) and (9) respectively. We comment that δ_{1i} and δ_{2i} in (7) and (11) must appear in the order explicated above. These arrangements of δ_{1i} and δ_{2i} ensure that the x_i updates are proceeded along the down-hill direction of $\Phi(\mathbf{x})$, and thus, the corresponding iteration can converge to the minimum of $\Phi(\mathbf{x})$. Iteration (11) is performed only if the corresponding (7) gives an estimate greater than b. From (11), if $x_i^{(k)} \le b$ then $x_i^{(k+\frac{1}{2})} \le b$, but we must also show that these $x_i^{(k+\frac{1}{2})}$ are non-negative. In fact we can prove that if $x_i^{(k)} \in [0,b]$ then $x_i^{(k+1)} \in [0,b]$ for all i. The statement of this property is given in Lemma 1 of Section III.

The updates given by (7) and (11) need to be further improved by a line search step to give an update $\mathbf{x}^{(k+1)}$ such that $\Phi(\mathbf{x})$ decreases when moving from the kth to the $(k+1)$th iteration. We first unify the updating formulae (7) and (11) into a single gradient formula. Combining (7) and (11) we have

$$x_i^{(k+\frac{1}{2})} = \begin{cases} x_i^{(k)}\frac{\delta_{1i}^{(k)}}{\delta_{2i}^{(k)}} & \text{if } x_i^{(k)}\delta_{1i}^{(k)}/\delta_{2i}^{(k)} \le b, \\ b - (b - x_i^{(k)})\frac{\delta_{2i}^{(k)}}{\delta_{1i}^{(k)}} & \text{otherwise,} \end{cases} \tag{12}$$

which leads to the following gradient expression

$$x_i^{(k+\frac{1}{2})} = x_i^{(k)} - s_i^{(k)}\Phi_i'(\mathbf{x}^{(k)}), \tag{13}$$

where

$$s_i^{(k)} = \begin{cases} x_i^{(k)}/\delta_{2i}^{(k)} & \text{if } x_i^{(k)}\delta_{1i}^{(k)}/\delta_{2i}^{(k)} \le b, \\ (b - x_i^{(k)})/\delta_{1i}^{(k)} & \text{otherwise.} \end{cases} \tag{14}$$

Here, attention must be paid to the situation where $x_i^{(k)} = 0$ or $x_i^{(k)} = b$, as this leads to $s_i^{(k)} = 0$ so that x_i cannot be further improved. We adopt the following strategy to treat this circumstance. We keep

$$s_i^{(k)} = 0 \qquad \begin{array}{l} \text{if } x_i^{(k)} = 0 \text{ and } \Phi_i'(\mathbf{x}^{(k)}) \ge 0, \\ \text{or if } x_i^{(k)} = b \text{ and } \Phi_i'(\mathbf{x}^{(k)}) \le 0 \end{array} \tag{15}$$

since they satisfy the KKT condition. For those i where $x_i^{(k)} = 0$ with $\Phi_i'(\mathbf{x}^{(k)}) < 0$, we set $s_i^{(k)} = k_1/\delta_{2i}^{(k)}$ with a nonzero constant k_1. We select $k_1 = 0.02b$ in this paper; but this k_1 must also ensure the corresponding $x_i^{(k+\frac{1}{2})} = -k_1\Phi_i'(\mathbf{x}^{(k)})/\delta_{2i} < b$, and so that

$$s_i^{(k)} = \min\{0.02b/\delta_{2i}^{(k)}, -b/\Phi_i'(\mathbf{x}^{(k)})\}. \tag{16}$$

Similarly, for the case of $x_i^{(k)} = b$ with $\Phi_i'(\mathbf{x}^{(k)}) > 0$ we set $s_i^{(k)} = (b - k_2)/\delta_{1i}$. We choose $k_2 = 0.98b$, and after considering $x_i^{(k+\frac{1}{2})} > 0$ we have

$$s_i^{(k)} = \min\{0.02b/\delta_{1i}^{(k)}, b/\Phi_i'(\mathbf{x}^{(k)})\}. \tag{17}$$

Note that the constant ε only affects the s_i values, so it will only change the convergence speed of the MI algorithm.

Thus, after incorporating a line search step we have

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \omega^{(k)}S^{(k)}\Phi'(\mathbf{x}^{(k)}), \tag{18}$$

where $S^{(k)}$ is a diagonal matrix with diagonal entries $s_i^{(k)} \ge 0$, and $\omega^{(k)}$ is the step size determined by the line search. Since $S^{(k)}$ is non-negative definite, iteration (18) with $\omega^{(k)} = 1$ (which is actually $\mathbf{x}^{(k+\frac{1}{2})}$) cannot move along the uphill direction. Thus if $\Phi(\mathbf{x}^{(k+\frac{1}{2})}) > \Phi(\mathbf{x}^{(k)})$, it indicates the necessity of reducing the step size. This explains that the line search range of this MI algorithm is fixed at $(0,1]$ for all $\omega^{(k)}$. Moreover, this range ensures $0 \le \mathbf{x}^{(k+1)} \le b$ once $\mathbf{x}^{(k)}$ is in $[0,b]$. Due to the fixed search interval, this line search is remarkably simple, and it demands that $A\mathbf{x}^{(k+\frac{1}{2})}$ be calculated only once in each iteration. Let $\mathbf{d}^{(k)} = \mathbf{x}^{(k+\frac{1}{2})} - \mathbf{x}^{(k)}$, which defines the line search direction. One simple and efficient search strategy is provided by Armijo's rule (e.g. [13]). Armijo line search is a finite terminating algorithm. Briefly, it starts with $\omega = 1$, and for each ω it checks if the following Armijo condition is satisfied:

$$\Phi(\mathbf{x}^{(k)} + \omega\mathbf{d}^{(k)}) \le \Phi(\mathbf{x}^{(k)}) - \xi\omega\Phi'(\mathbf{x}^{(k)})^T\mathbf{d}^{(k)}, \tag{19}$$

where $0 < \xi < 1$ is a fixed parameter such as $\xi = 10^{-2}$. If (19) is true then stop; otherwise, reset $\omega = \rho\omega$ (such as $\rho = 0.6$) and reevaluate the Armijo condition (19). We name this MI algorithm with Armijo's line search the boxed-constrained multiplicative iterative (BCMI) algorithm. A summary of BCMI is given in Fig. 1.

We complete this section by making a comment on some of the parameters required by BCMI. ε in equations (8) and (9) only affects the speed of convergence, and we usually set it to a small value, such as 10^{-3} for images with Poisson or Gaussian noises. We find that $k_1 = 0.02b$, $k_2 = 0.98b$ and $\rho = 0.6$ always give good convergence properties.

III. Convergence properties

In this section we study the convergence properties of the BCMI algorithm. We first demonstrate that the BCMI update $x_i^{(k+1)}$ will be bounded by $[0,b]$ if $x_i^{(k)}$ is in $[0,b]$. We then provide the convergence theorems of the BCMI algorithm.

Lemma 1: Consider the box-constrained optimization problem defined by (2). Assume that the first derivative of $\Phi(\mathbf{x})$ exists for $0 \le \mathbf{x} \le b$. For the BCMI algorithm given by (18), if $x_i^{(k)} \in [0,b]$ then also $x_i^{(k+1)} \in [0,b]$ for $i = 1,\ldots,n$.

Proof: First consider $0 < x_i^{(k)} < b$. According to (12), for those i where $x_i^{(k)}\delta_{1i}^{(k)}/\delta_{2i}^{(k)} \le b$, the corresponding $x_i^{(k+\frac{1}{2})} \in [0,b]$. For other i, their $x_i^{(k+\frac{1}{2})} \le b$ are obvious from (12), and their $x_i^{(k+\frac{1}{2})} \ge 0$ can be obtained from the fact that

$$x_i^{(k+\frac{1}{2})} = b\left(1 - \frac{\delta_{2i}^{(k)}}{\delta_{1i}^{(k)}}\right) + x_i^{(k)}\frac{\delta_{2i}^{(k)}}{\delta_{1i}^{(k)}}$$

and the fact that $0 < \delta_{2i}^{(k)}/\delta_{1i}^{(k)} < 1$. For $x_i^{(k)} = 0$ or b, the way we select the corresponding $s_i^{(k)}$ ascertains that $x_i^{(k+\frac{1}{2})} \in [0,b]$. Now from $x_i^{(k+\frac{1}{2})} \in [0,b]$ we have $x_i^{(k+1)} \in [0,b]$ since $x_i^{(k+1)} = (1-\omega^{(k)})x_i^{(k)} + \omega^{(k)}x_i^{(k+\frac{1}{2})}$ and $0 < \omega^{(k)} \le 1$. ■

Lemma 1 explains that if the initial $\mathbf{x}^{(0)}$ is in $(0,b)$ then all $\mathbf{x}^{(k)}$ are in $[0,b]$. In the following, we show that BCMI converges if Φ has the following property:

Box-constrained MI (BCMI) image restoration algorithm

Let k be the iteration number, $\mathbf{x}$ be the n-vector representing the true but unknown image, $\mathbf{y}$ be the n-vector representing the observed blurry and noisy image and $\Phi(\mathbf{x})$ be the objective function defined in (2).

Initialization:

Set $k = 0$. Choose an initial value $0 < \mathbf{x}^{(0)} < b$ such that $\Phi(\mathbf{x}^{(0)}) < +\infty$. Select an $\varepsilon \geq 0$.

Iteration $k+1$:

• For $i = 1, \dots, n$ do Steps 1 – 3.

Step 1: Compute

$$g_{1i}^{(k)} = \sum_{j=1}^{n} a_{ji} l_j'(\mu_j^{(k)})^{+} - \lambda[\mathcal{J}_i'(\mathbf{x}^{(k)})]^{-},\ g_{2i}^{(k)} = -\sum_{j=1}^{n} a_{ji} l_j'(\mu_j^{(k)})^{-} + \lambda[\mathcal{J}_i'(\mathbf{x}^{(k)})]^{+},$$

and then $\delta_{1i}^{(k)} = g_{1i}^{(k)} + \varepsilon$ and $\delta_{2i}^{(k)} = g_{2i}^{(k)} + \varepsilon$. Compute $g_i^{(k)} = \delta_{1i}^{(k)} - \delta_{2i}^{(k)}$.

Step 2: For $x_i^{(k)} \neq 0$ and $x_i^{(k)} \neq b$, compute $s_i^{(k)} = x_i^{(k)}/\delta_{2i}^{(k)}$ if $x_i^{(k)}/\delta_{2i}^{(k)} \leq b/\delta_{1i}^{(k)}$ and $s_i^{(k)} = (b - x_i^{(k)})/\delta_{1i}^{(k)}$ if $x_i^{(k)}/\delta_{2i}^{(k)} > b/\delta_{1i}^{(k)}$.

For $x_i^{(k)} = 0$, compute $s_i^{(k)} = \min\{0.02b/\delta_{2i}^{(k)}, -b/g_i^{(k)}\}$ if $g_i^{(k)} < 0$ and $s_i^{(k)} = 0$ if $g_i^{(k)} \geq 0$.

For $x_i^{(k)} = b$, compute $s_i^{(k)} = \min\{0.02b/\delta_{1i}^{(k)}, b/g_i^{(k)}\}$ if $g_i^{(k)} > 0$ and $s_i^{(k)} = 0$ if $g_i^{(k)} \leq 0$.

Step 3: Compute $x_i^{(k+\frac{1}{2})} = x_i^{(k)} - s_i^{(k)} g_i^{(k)}$.

• *Line search:* If $\Phi(\mathbf{x}^{(k+\frac{1}{2})}) < \Phi(\mathbf{x}^{(k)})$ then $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+\frac{1}{2})}$; otherwise $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega^{(k)} S^{(k)} \mathbf{g}^{(k)}$, where $\mathbf{g}^{(k)} = (g_1^{(k)}, \dots, g_n^{(k)})^{\top}$ and $\omega^{(k)} \in (0, 1)$ is the step size obtained using Armijo's line search.

Fig. 1. The BCMI algorithm.

Property A:

1) $\Phi(\mathbf{x})$ is bounded, continuous and differentiable for $0 \leq \mathbf{x} \leq b$, and
2) $\Phi_i'(\mathbf{x})$ are bounded and continuous over $0 \leq \mathbf{x} \leq b$ for $i = 1, \dots, n$.

We will see in Section IV that image restoration problems with TV penalty will have this property after smoothing the TV norm.

Theorem 1: Assume that Property A holds for $\Phi(\mathbf{x})$ defined in (2). Let Γ be the set of stationary points of $\Phi(\mathbf{x})$ satisfying the box constraints. For sequences $\{\mathbf{x}^{(k)}\}$ generated by the BCMI algorithm, all the limit points of $\{\mathbf{x}^{(k)}\}$ are in Γ. Moreover, there exist some $\mathbf{x}^* \in \Gamma$ such that $\Phi(\mathbf{x}^{(k)})$ converges monotonically to $\Phi(\mathbf{x}^*)$.

Proof: Let $M(\mathbf{x})$ denote the iteration mapping for the BCMI algorithm. Let $\Omega = \{\mathbf{x} : 0 \leq x_i \leq b \text{ for } i = 1, \dots, n\}$. According to (18), mapping M is given by

$$M(\mathbf{x}) = \mathbf{x} - \omega(\mathbf{x}) S(\mathbf{x}) \Phi'(\mathbf{x}). \tag{20}$$

Clearly, $M(\mathbf{x}) : \Omega \to \Omega$ is a one-to-many mapping as $\omega(\mathbf{x})$ is not unique. Firstly, Lemma 1 shows that if the initial guess satisfies $0 < x_i^{(0)} < b$ for all i, then $\mathbf{x}^{(k)} \in \Omega$ for any iteration number k. Set Ω is closed and bounded, and thus is compact. Secondly, due to the line search step of BCMI, its iteration $\mathbf{x}^{(k+1)} \in M(\mathbf{x}^{(k)})$ satisfies: $\Phi(\mathbf{x}^{(k+1)}) \leq \Phi(\mathbf{x}^{(k)})$, where equality holds only when $\mathbf{x}^{(k)} \in \Gamma$. Finally, let $\mathbf{z} = M(\mathbf{x})$ and $\mathbf{d} = S(\mathbf{x})\Phi'(\mathbf{x})$. Assume $\mathbf{x}^{(k)} \to \mathbf{x}_0 \notin \Gamma$ as $k \to \infty$. Since the expression $x_i \delta_{1i}(\mathbf{x})/\delta_{2i}(\mathbf{x})$ in (14) is continuous in $\mathbf{x}$, we have $\mathbf{d}^{(k)} \to \mathbf{d}_0 \neq 0$. Suppose $\{\mathbf{z}^{(k)}\}$ is a sequence obtained according to $\mathbf{z}^{(k)} = M(\mathbf{x}^{(k)})$ and that $\mathbf{z}^{(k)} \to \mathbf{z}_0$ as $k \to \infty$. We wish to show that $\mathbf{z}_0 \in M(\mathbf{x}_0)$, and so that M is closed at points outside Γ. For the sequence $\{\omega^{(k)}\}$ we have

$$\omega^{(k)} = \frac{||-\mathbf{z}^{(k)} + \mathbf{x}^{(k)}||}{||\mathbf{d}^{(k)}||} \to \frac{||-\mathbf{z}_0 + \mathbf{x}_0||}{||\mathbf{d}_0||} \equiv \omega_0,$$

so that $\mathbf{z}_0 = \mathbf{x}_0 - \omega_0 S(\mathbf{x}_0)\Phi'(\mathbf{x}_0)$. As all $\omega^{(k)} \leq 1$, and so will ω_0. On the other hand, as $\Phi(\mathbf{z}^{(k)}) < \Phi(\mathbf{x}^{(k)})$ for each k and Φ is continuous, $\Phi(\mathbf{z}_0) < \Phi(\mathbf{x}_0)$. Thus $\mathbf{z}_0 \in M(\mathbf{x}_0)$, and so that M is closed at $\mathbf{x}_0 \notin \Gamma$. The above three properties explain that $\{\mathbf{x}^{(k)}\}$ satisfies the conditions of the Zangwill's global convergence theorem [14], and hence the results of this theorem follow. ■

Theorem 1 mainly shows that the limit of any convergent subsequence of $\{\mathbf{x}^{(k)}\}$ is in the solution set Γ, and that $\Phi(\mathbf{x}^{(k)})$ converges to $\Phi(\mathbf{x}^*)$ for some $\mathbf{x}^* \in \Gamma$. It remains to prove that the sequence $\{\mathbf{x}^{(k)}\}$ is convergent itself. This result is acquired through Theorems 2 and 3. Note that the proofs of Theorems 2 and 3 closely follow [15].

Theorem 2: Assume that Property A holds for $\Phi(\mathbf{x})$. Then BCMI sequences $\{\mathbf{x}^{(k)}\}$ satisfy $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| \to 0$ as $k \to \infty$.

Proof: From Theorem 1 we can conclude that if $\Phi(\mathbf{x})$ is bounded over the set Ω (Ω is defined in the proof of Theorem 1) then $\Phi(\mathbf{x}^{(k+1)}) - \Phi(\mathbf{x}^{(k)}) \to 0$ when $k \to \infty$. Now from (18) we have

$$\Phi'(\mathbf{x}^{(k)}) = -\frac{1}{\omega^{(k)}}[S^{(k)}]^{-1}(\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}). \tag{21}$$

Hence, according to Armijo condition (19), we have

$$\begin{aligned} & -\Phi(\mathbf{x}^{(k+1)})+\Phi(\mathbf{x}^{(k)}) \\ & \geq \frac{\xi}{\omega^{(k)}}(\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)})^T[S^{(k)}]^{-1}(\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}) \\ & \geq \frac{\xi}{s_{(n)}^{(k)}\omega^{(k)}}\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\|^2, \end{aligned} \tag{22}$$

where $s_{(n)}^{(k)}$ denotes the maximum diagonal values of $S^{(k)}$. Clearly, we can safely assume $s_{(n)}^{(k)} \neq 0$ as otherwise the sequence $\{\mathbf{x}^{(k)}\}$ has already converged. ■

Theorem 3: Assume that Property A holds for $\Phi(\mathbf{x})$. According to Theorem 1, for BCMI sequences $\{\mathbf{x}^{(k)}\}$ there exist some stationary points $\mathbf{x}^* \in \Gamma$ such that $\Phi(\mathbf{x}^{(k)}) \to \Phi(\mathbf{x}^*)$ as $k \to \infty$. Let Γ_0 be the set of these stationary points. If Γ_0 is discrete then $\mathbf{x}^{(k)}$ converges to an $\mathbf{x}^* \in \Gamma_0$.

Proof: From Theorem 2 we have $\|\mathbf{x}^{(k+1)}-\mathbf{x}^{(k)}\| \to 0$ as $k \to \infty$. Since all $\mathbf{x}^{(k)}$ are bounded, $\mathbf{x}^{(k)}$ is convergent, and its limits are in a connected and compact subset of Γ_0 [16, Theorem 28.1]. Therefore there is a single stationary point. ■

Finally, we provide a theorem explaining that the limit of $\{\mathbf{x}^{(k)}\}$ meets the KKT condition.

Theorem 4: Assume that Property A holds for $\Phi(\mathbf{x})$. The limit of BCMI sequence $\{\mathbf{x}^{(k)}\}$ satisfies the KKT condition (3).

Proof: Theorem 3 explains $\mathbf{x}^{(k)} \to \mathbf{x}^*$ as $k \to \infty$. For any $0 < x_i^* < b$ it must satisfy $x_i^* = x_i^* - \omega^* s_i^* \Phi_i'(\mathbf{x}^*)$ according to (18), where s_i^* denotes s_i at $\mathbf{x}^*$. From the way we select $s_i^{(k)}$ explicated before, $s_i^* \neq 0$ in this case. Thus we have $\Phi_i'(\mathbf{x}^*) = 0$ if $0 < x_i^* < b$. For $x_i^* = 0$, there are two possibilities: (i) $x_i^{(k)} \to 0$ as $k \to \infty$ but all $x_i^{(k)} \neq 0$; (ii) there exists an integer $K > 0$ such that $x_i^{(k)} = 0$ and $\Phi_i'(\mathbf{x}^{(k)}) > 0$ for all $k > K$. For case (i) it must have, according to (12), $\delta_{1i}^* < \delta_{2i}^*$, where δ_{1i}^* and δ_{2i}^* are the limits (when $k \to \infty$) of $\delta_{1i}^{(k)}$ and $\delta_{2i}^{(k)}$ respectively, and thus the corresponding $\Phi_i'(\mathbf{x}^*) = -\delta_{1i}^* + \delta_{2i}^* > 0$. For case (ii), $\Phi_i'(\mathbf{x}^*) > 0$ is obvious. Similarly, if any $x_i^* = b$, the corresponding $\Phi_i'(\mathbf{x}^*) < 0$. ■

IV. Box-constrained image restoration with TV penalty

Total variation (TV) [8] penalty has been widely used in image restoration due to its remarkable ability to reserve edges. However, most existing TV restoration algorithms, such as [8], [10], [17], [18], [19], [20], [21], [22], and [23], ignore the positivity or box constraints. We explain in this section that the BCMI algorithm can be easily implemented to box-constrained TV image restorations. The TV penalty term is defined as

$$\sum_{j=1}^{n}\sqrt{(R_j\mathbf{x})^2+(C_j\mathbf{x})^2},$$

where R_j and C_j are respectively the jth row of the $n \times n$ matrices R and C, which are defined such that entries of $R\mathbf{x}$ and $C\mathbf{x}$ represent the first-order differences of the 2D version of $\mathbf{x}$ along the row and column directions respectively. One well-known numerical difficulty in using the TV penalty is that its derivative can be singular. One way to get around this is to consider the "smoothed" TV penalty; see, for example, [8] and [21], where

$$\mathcal{J}(\mathbf{x}) = \sum_{j=1}^{n}\sqrt{(R_j\mathbf{x})^2+(C_j\mathbf{x})^2+\beta}. \tag{23}$$

The smoothing factor $\beta > 0$ is included to avoid degenerate derivative of $\mathcal{J}(\mathbf{x})$. Note that

$$\mathcal{J}_i'(\mathbf{x}) = \sum_{j=1}^{n}\frac{r_{ji}R_j\mathbf{x}+c_{ji}C_j\mathbf{x}}{\sqrt{(R_j\mathbf{x})^2+(C_j\mathbf{x})^2+\beta}}. \tag{24}$$

Thus $\mathcal{J}'(\mathbf{x})$ can be computed as the summation of the first order row difference of $R\mathbf{x}/\sqrt{(R\mathbf{x})^2+(C\mathbf{x})^2+\beta}$ and the first order column difference of $C\mathbf{x}/\sqrt{(R\mathbf{x})^2+(C\mathbf{x})^2+\beta}$, where squares represent element-wise squares of the corresponding matrices. Clearly with $\beta > 0$, $\mathcal{J}(\mathbf{x})$ satisfies Property A.

Below, we demonstrate that BCMI can be easily implemented to TV image restoration for the three most commonly used noise models, namely Gaussian, Poisson and salt-and-pepper noises.

A. Gaussian noise

For this noise model, the observed pixel image intensity y_j follows a Gaussian distribution with mean $\mu_j = A_j\mathbf{x}$ and variance σ^2. The log-likelihood of y_j (i.e. $l_j(\mu_j)$) contains $1/\sigma^2$. After multiplying σ^2 with $\Phi(\mathbf{x})$ of (2), and re-defining $\lambda\sigma^2$ as λ, we have $l_j(\mu_j) = -\frac{1}{2}(y_j-\mu_j)^2$. Thus $l_j'(\mu_j) = y_j - \mu_j$. This gives $l_j'(\mu_j)^+ = y_j$ and $l_j'(\mu_i)^- = -\mu_j$, and hence, $\delta_{1i}^{(k)} = \sum_{j=1}^{n} a_{ji}y_j - \lambda[\mathcal{J}_i'(\mathbf{x}^{(k)})]^- + \varepsilon$ and $\delta_{2i}^{(k)} = \sum_{j=1}^{n} a_{ji}\mu_j^{(k)} + \lambda[\mathcal{J}_i'(\mathbf{x}^{(k)})]^+ + \varepsilon$. Then image pixels are updated according to

$$x_i^{(k+1)} = x_i^{(k)} - \omega^{(k)}s_i^{(k)}(\delta_{2i}^{(k)} - \delta_{1i}^{(k)}) \tag{25}$$

for $i = 1, \ldots, n$, where $s_i^{(k)}$ values are computed according to (14), (15), (16) and (17), and $\omega^{(k)}$ is determined by Armijo line search. When using the TV penalty, $\mathcal{J}_i'(\mathbf{x}^{(k)})$ is given by (24). Clearly, the iteration given by (25) is extremely easy to implement.

B. Poisson noise

For the Poisson noise model, observed pixel image intensity y_j follows a Poisson distribution with mean $\mu_j = A_j\mathbf{x}$. The log-likelihood of y_j corresponding to this noise model is $l_j(\mu_j) = -\mu_j + y_j \log\mu_j$, which gives $l_j'(\mu_j) = -1 + y_j/\mu_j$. Thus $\delta_{1i}^{(k)} = \sum_{j=1}^{n} a_{ji}y_j/\mu_j^{(k)} - \lambda[\mathcal{J}_i'(\mathbf{x}^{(k)})]^- + \varepsilon$ and $\delta_{2i}^{(k)} = \sum_{j=1}^{n} a_{ji} + \lambda[\mathcal{J}_i'(\mathbf{x}^{(k)})]^+ + \varepsilon$. Image pixels are then updated by (25). We comment that when $\lambda = 0$, $\varepsilon = 0$ and $b = \infty$, this algorithm coincides with the well known Expectation-Maximization (EM) image reconstruction algorithm in emission tomography [24].

C. Salt-and-pepper noise

Salt-and-pepper noise removal has been considered by many researchers, for example [25], [26] and [27]. This type of noise can be effectively dealt with by the Laplace distribution [2]. More specifically, we can assume that image pixel intensity y_j follows a Laplace distribution with mean $\mu_j = A_j\mathbf{x}$ and variance σ^2. Again, after multiplying σ^2 with $\Phi(\mathbf{x})$, we can express the log-likelihood of y_j as $l_j(\mu_j) = -|y_j - \mu_j|$. An immediate problem here is that $l_j(\mu_j)$ is not differentiable at $\mu_j = y_j$. Adopting the same idea of approximating the TV penalty function in (23), we approximate $l_j(\mu_j)$ by

$$l_j(\mu_j) \approx -\sqrt{(y_j - \mu_j)^2 + \gamma}, \tag{26}$$

where γ is a small positive constant to smooth $l_j(\mu_j)$ at $\mu_j = y_j$. Thus

$$l'_j(\mu_j) \approx \frac{y_j - \mu_j}{\sqrt{(y_j - \mu_j)^2 + \gamma}},$$

and the corresponding

$$\delta_{1i}^{(k)} = \sum_{j=1}^{n} \frac{a_{ji}y_j}{\sqrt{(y_j - \mu_j^{(k)})^2 + \gamma}} - \lambda[\mathcal{J}'_i(\mathbf{x}^{(k)})]^- + \varepsilon$$

and

$$\delta_{2i}^{(k)} = \sum_{j=1}^{n} \frac{a_{ji}\mu_j^{(k)}}{\sqrt{(y_j - \mu_j^{(k)})^2 + \gamma}} + \lambda[\mathcal{J}'_i(\mathbf{x}^{(k)})]^+ + \varepsilon.$$

Again, all the x_i's are then updated using (25).

V. Results

In this section, we provide and discuss simulation results on TV image restorations under the three noise models discussed in the previous section, namely Gaussian, Poisson and salt-and-pepper noises. This simulation study had five goals. 1) The first goal was to show that the BCMI algorithm can effectively solve the box-constrained TV image restoration problem with these three noise distributions. 2) The second goal was to compare our BCMI algorithm with another fast box-constrained and TV-based image restoration algorithm, namely the fast iterative shrinkage/thresholding algorithm (FISTA) of [5]. We demonstrated that in general the BCMI iterations improved signal-to-noise ratios much faster than FISTA. 3) Its third goal was to demonstrate that box-constrained PL estimates given by BCMI usually attain better signal-to-noise ratios than the estimates given by simply projecting unconstrained or only positively constrained TV restorations onto the box constraints. The unconstrained algorithm used for comparison was the Fast Total Variation Deconvolution (FTVD) algorithm [10], and the positively constrained algorithm was the TV Minimization by Augmented Lagrangian (TVAL) algorithm from the the same research group. 4) The fourth goal was to explain, through Monte-Carlo simulations, that over a range of smoothing values, box constraints usually produce restorations with better mean and variance properties of the signal-to-noise ratio than projecting FTVD or TVAL onto the box constraints. 5) The last goal was to assess BCMI for image denoising. This was accomplished by comparing BCMI with the kernel regression method of [28], one of the state-of-the-art image denoising methods.

These goals are elaborated and demonstrated in details through Sections V-A – V-E below. All simulation computations were conducted using MATLAB.

FISTA is a gradient-based backward-forward splitting algorithm for box-constrained image deblurring and is supposed to converge fast. FTVD uses variable-splitting and penalty techniques, together with fast Fourier transform (FFT), to deconvolve and denoise images. TVAL adopts the augmented Lagrangian method with an alternating minimization approach (e.g. [20]) to produce estimates with positive constraint. MATLAB programs for FISTA are available at "http://iew3.technion.ac.il/˜becka/papers/tv_fista.zip", for FTVD at "http://www.caam.rice.edu/˜optimization/L1/ftvd/" and for TVAL at "http://www.caam.rice.edu/ ˜optimization/L1/TVAL3/". FISTA can only process images containing Gaussian noises. The TVAL MATLAB program has an option for the positivity constraint, and it again can only be used to remove Gaussian noises. The FTVD algorithm can remove Gaussian and salt-and-pepper noises, but it cannot impose the positivity constraint. Nevertheless, FISTA, FTVD and TVAL cannot address Poisson noise, and we, hence, only applied BCMI to Poisson noise examples in the simulation.

FTVD achieves best results if applied to images with pixel values in [0, 1] (see [10]), so we applied FTVD to the simulated blurry and noisy images as described below. We first divided pixel values of a simulated image by 255 and then applied FTVD to the rescaled image. The obtained FTVD restoration, however, might have pixel values outside [0, 1]. Instead of linearly scaling the range of pixels to [0,1], we projected all pixels back to [0, 1] by setting any pixel value less than 0 or greater than 1 to 0 or 1 respectively. We then multiplied the result by 255. We found this scaling-FTVD-projection procedure gave better signal-to-noise ratio for FTVD. For applying TVAL, we first turned the positivity option on so that its estimates were non-negative. After TVAL, we projected the TVAL results to [0, 255] by setting any pixel value greater than 255 to 255. For the rest of this paper, these projected FTVD and TVAL are still called FTVD and TVAL respectively when there is no confusion.

Three 512×512 (Lena, Bridge and Boat) and two 256×256 (Satellite and Church) test images were included in this simulation. Lena, Bridge and Boat images can be viewed in Fig. 2, while Satellite and Church images are displayed in Fig. 3. When comparing BCMI with FISTA we used Lena image and a blurring filter given by the 9×9 Gaussian (standard deviation 4) blur, created in MATLAB by "H=fspecial('Gaussian',[9, 9], 4)", and this is the same blur as in [5]. In all other examples, we used a motion blur created by "H=fspecial('motion',15, 30)". Poisson, Gaussian and salt-and-pepper noises were added to the blurry images by MATLAB function "imnoise".

In the simulation we assessed the quality of a restoration by peak signal-to-noise ratio (PSNR). Assuming all pixel intensities of the restored ($\hat{\mathbf{x}}$) and the original ($\mathbf{x}$) images are

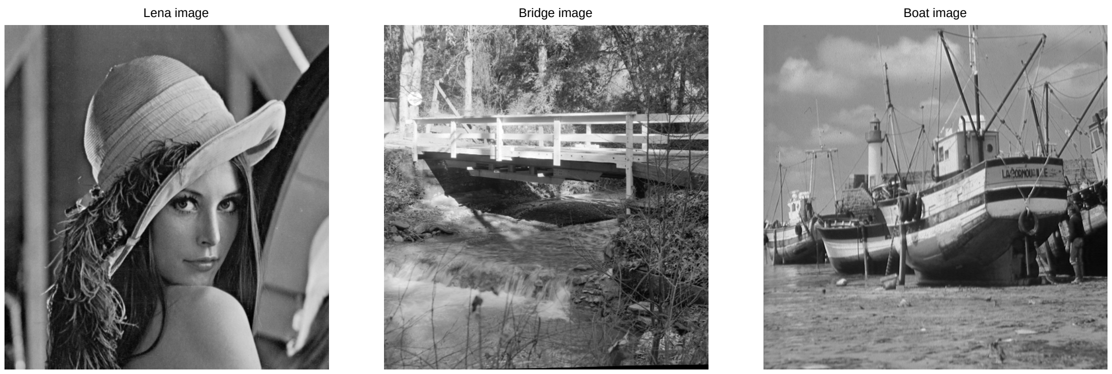


Fig. 2. Lena, Bridge and Boat images.

Fig. 3. Satellite and Church images.

in $[0, 255]$, PSNR is then defined as

$$\mathrm{PSNR} \triangleq 10\log_{10}\frac{255^2}{\frac{1}{n}\|\hat{\mathbf{x}}-\mathbf{x}\|_2^2}(\mathrm{db}), \tag{27}$$

where n is the number of pixels of $\mathbf{x}$ (or $\hat{\mathbf{x}}$). All the images in the figures are displayed over the $[0, 255]$ interval by MATLAB function "imshow(x, [0, 255])".

A. Evaluate BCMI

We first use Lena image to explore the BCMI algorithm, focusing on demonstrating that BCMI is successful in TV regularized image restorations with box constraints, where image noises are Poisson, Gaussian and salt-and-pepper. After motion blurring the Lena image we added Poisson, Gaussian (mean 0 and variances $\sigma^2 = 5$ and 40), and salt-and-pepper (60%) noises. For every simulated noisy and blurry Lena image, we experimentally found the "optimal" smoothing value for the BCMI algorithm. Here "optimal" means it gave the best restoration judged by PSNR. These smoothing values are displayed in Table I. Table I also contains the optimal smoothing values for other algorithms (i.e. FVTD, TVAL and FISTA) and other images; all were obtained by trial-and-error. Other optional parameters of BCMI were set as: $\varepsilon = 10^{-3}$ (see (8) and (9)) for Poisson and Gaussian noises and $\varepsilon = 10$ for salt-and-pepper noises, $\beta = 3$ (see (23)) for Poisson and Gaussian noises and $\beta = 10$ for salt-and-pepper noises, and $\gamma = 1$ (see (26)) for salt-and-pepper noises. Note that these ε values gave good convergence speed of BCMI but they might not be optimal. The β and γ values gave visually good restorations, but they were not fine-tuned as the final restorations were not very sensitive to β and γ. We conceived BCMI to have been converged if reconstructions in consecutive iterations differed by less than 10^{-4} for all pixels. When using the BCMI algorithm, we selected the upper limit $b = 255$ for all the test images.

Panel (a) of Fig. 4 exhibits the plots of $(\Phi(\mathbf{x}^{(k)}) - \Phi(\mathbf{x}^{(K)}))/(\Phi(\mathbf{x}^{(1)}) - \Phi(\mathbf{x}^{(K)}))$ (here K is the maximum iteration number) against iteration numbers (both in log scale) for the BCMI restorations of Lena image, with Poisson, Gaussian (two variances) and salt-and-pepper noises. This display uses $K = 500$ iterations for all the noises. Clearly, BCMI monotonically decreased all $\Phi(\mathbf{x})$ along the iterations, which agrees with Theorem 1. Panel (b) of Fig. 4 displays the plots of PSNR versus iteration number. From these plots we observe that BCMI for Gaussian ($\sigma^2 = 40$) stabilized quickly in less than 200 iterations, while BCMI for Gaussian ($\sigma^2 = 5$) and salt-and-pepper took longer iterations to converge.

Apart from testing Lena image with Poisson noise, we also tested Bridge and Boat images with Poisson noise; results are given in Fig. 5.

B. Compare BCMI with FISTA

Lena image was again used to compare BCMI with FISTA. Since FISTA demands the reflexive boundary condition in the blurring operation (see [5]), we created a blurry Lena image by "imfilter(x, H, 'symmetric')", where blur filter H was a 9×9 Gaussian blur with standard deviation 4 as explained before. We also set accordingly the boundary option for BCMI to "symmetric".

In this study, only Gaussian noises were used due to the limitation of FISTA. We used three variances: $255^2 \times 10^{-6}$, 5 and 40, where the first variance was equivalent to the variance

Algorithm	Noise	Smoothing value λ				
		Lena	Bridge	Boat	Satellite	Church
BCMI	Poisson	0.018	0.012	0.012		
	Gaussian ($\sigma^2 = 5$)	0.19	0.12	0.15	0.084	0.15
	Gaussian ($\sigma^2 = 30$)				0.36	0.53
	Gaussian ($\sigma^2 = 40$)	0.8	0.59	0.64		
	Gaussian ($\sigma^2 = 50$)				0.52	0.71
	salt-and-pepper (60%)	0.18	0.17	0.16	0.24	0.15
FTVD	Gaussian ($\sigma^2 = 5$)	1300	2000	1600	2500	1700
	Gaussian ($\sigma^2 = 30$)				630	500
	Gaussian ($\sigma^2 = 50$)				430	360
	salt-and-pepper (60%)	5	5.5	5	3.6	4.6
TVAL	Gaussian ($\sigma^2 = 40$)	38	41	46		
FISTA	Gaussian ($\sigma^2 = 255^2 \times 10^{-6}$)	0.006				
	Gaussian ($\sigma^2 = 5$)	0.11				
	Gaussian ($\sigma^2 = 40$)	0.7				

TABLE I
OPTIMAL SMOOTHING VALUES FOR DIFFERENT TEST IMAGES, NOISE MODELS AND ALGORITHMS.

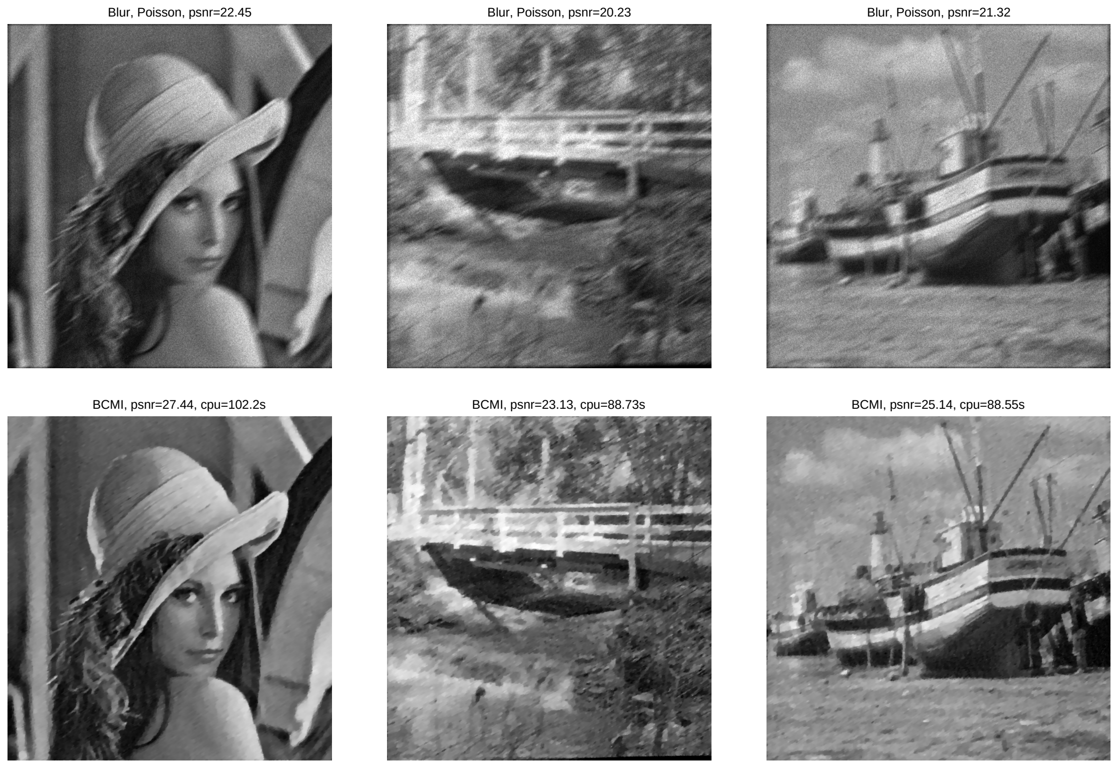

Blur, Poisson, psnr=22.45
Blur, Poisson, psnr=20.23
Blur, Poisson, psnr=21.32
BCMI, psnr=27.44, cpu=102.2s
BCMI, psnr=23.13, cpu=88.73s
BCMI, psnr=25.14, cpu=88.55s

Fig. 5. BCMI restorations of Poisson noisy images. Top row: Blurred with Poisson noises for Lena, Bridge and Boat images; bottom row: the corresponding BCMI restorations.

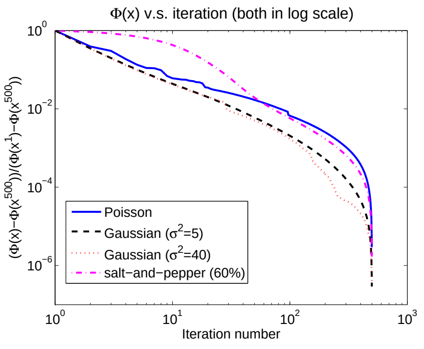
used in [5]. From the generated Lena noisy images we found the optimal smoothing values of FISTA by trial-and-error and they can be found in Table I. We also applied these smoothing values in BCMI although they might not be optimal for BCMI. Our objective function $\Phi(x)$ and the objective function used by FISTA (equation (1.1) in [5]) are basically equivalent, where only the penalties differ slightly due to the approximation (23). Hence the optimal smoothing values of these two algorithms must be close to each other. Images processed by BCMI and FISTA were all assumed in the range [0, 255].

We compare BCMI and FISTA by showing their plots of PSNR values against CPU time in Fig. 6. Clearly, BCMI converged faster and achieved higher PSNR than FISTA for both the $\sigma^2 = 5$ and $\sigma^2 = 40$ cases (panels (b) and (c)). Only in the case of extremely small variance (i.e. $\sigma^2 = 255^2 \times 10^{-6}$) did FISTA eventually surpassed BCMI in the PSNR value after nearly 100 CPU seconds.

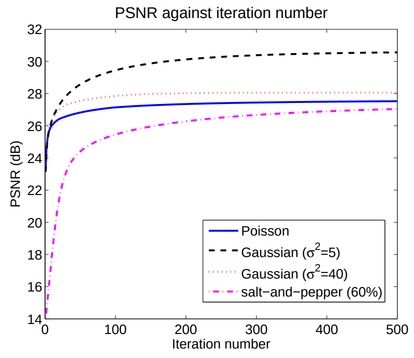
C. Compare BCMI with projected FTVD and projected TVAL

Here we used Lena, Bridge and Boat images to compare BCMI with projected FTVD and projected TVAL. We compared BCMI and FTVD using Gaussian ($\sigma^2 = 5$) and salt-and-pepper (60%) noises, and compared BCMI and TVAL using Gaussian ($\sigma^2 = 40$) noise.

FTVD is one of the state-of-the-art deblurring and denoising methods. Since FTVD adopts FFT in solving linear equations, it demands the "*circular*" boundary condition on the blurry image for accurate results. Thus, the blurry images for comparing BCMI and FTVD were generated by "imfilter(x, H, 'circular')". However, the "imfilter" function for creating



(a)



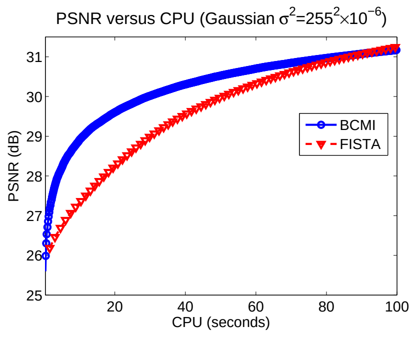
(b)

Fig. 4. The BCMI algorithm results for Lena image. Panel (a) contains the plots of scaled objective functions $(\Phi(\mathbf{x}^{(k)}) - \Phi(\mathbf{x}^{(500)}))/(\Phi(\mathbf{x}^{(0)}) - \Phi(\mathbf{x}^{(500)}))$ (log scale) against iteration number (log scale). Panel (b) contains the plot of PSNR versus iteration number.

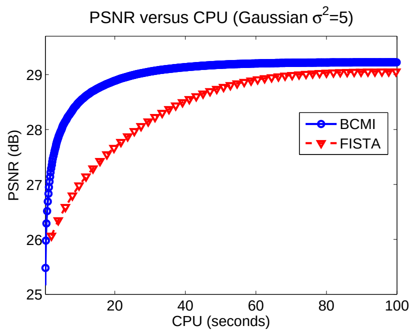
blurry images for comparing BCMI and TVAL did not have any boundary conditions (i.e. the zero boundary condition). When applying BCMI to image with a circular boundary, we also had to set BCMI boundary option to "circular". We emphasize, however, that there is no need to impose such a condition in BCMI in general as there is no FFT (for matrix inversion) involved in BCMI. We set the maximum iteration number for BCMI to 300 for Gaussian ($\sigma^2 = 40$) noise, and to 1000 for Gaussian ($\sigma^2 = 5$) and salt-and-pepper noises.

Fig. 7 presents the plots of PSNR against CPU time (log scale) for processing Lena images with Gaussian noises. We observe that BCMI produced monotonic increments in PSNR for both variances. Plots in panel (a) explain that FTVD was really fast for Gaussian ($\sigma^2 = 5$); it took only 0.5 CPU seconds to produce an image with the PSNR value which required about 60 CPU seconds for BCMI. This fast speed of FTVD, however, depends heavily on the correctness of the FFT operations which require the circular boundary condition. For example, FTVD was extremely poor if applied to an image with the zero boundary (plot not shown) and in this case, its PSNR was more than 3dB less than BCMI. Although TVAL was terminated within 40 CPU seconds, its PSNR values were not at all stabilized.

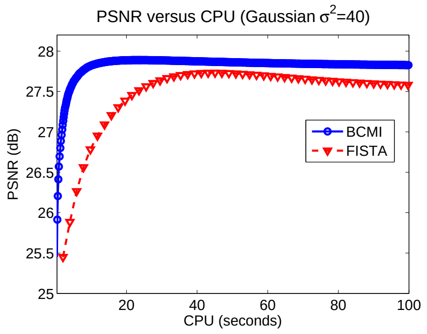
Figures 8 – 10 contain the restored Lena, Bridge and Boat images by BCMI, projected FTVD and projected TVAL; all were created using the optimal smoothing values provided in Table I. Some of their zoomed in images, in fact those corresponding to the Gaussian ($\sigma^2 = 5$) cases for BCMI and FTVD, are displayed in Fig. 11. Clearly, box constraints can,



(a)

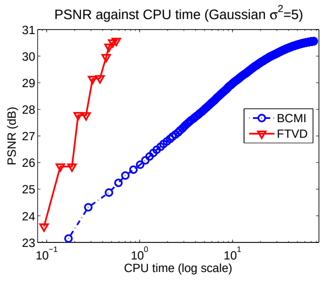


(b)

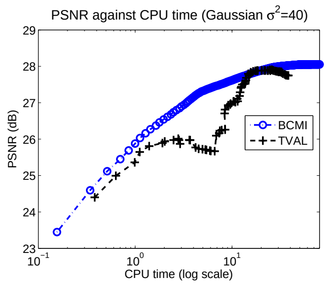


(c)

Fig. 6. PSNR versus CPU time plots for comparing BCMI and FISTA. Computations were based on restorations of Lena image with Gaussian noises . Panels (a), (b) and (c) contain the plots corresponding to Gaussian ($\sigma^2 = 255^2 \times 10^{-6}$), Gaussian ($\sigma^2 = 5$) and Gaussian ($\sigma^2 = 40$) noises respectively.



(a)



(b)

Fig. 7. Plots of PSNR versus CPU time for Lena image with Gaussian noises. Panel (a) contains the plots for BCMI and FTVD with Gaussian ($\sigma^2 = 5$) noise. Panel (b) contains the plots for BCMI and TVAL with Gaussian ($\sigma^2 = 40$) noise.

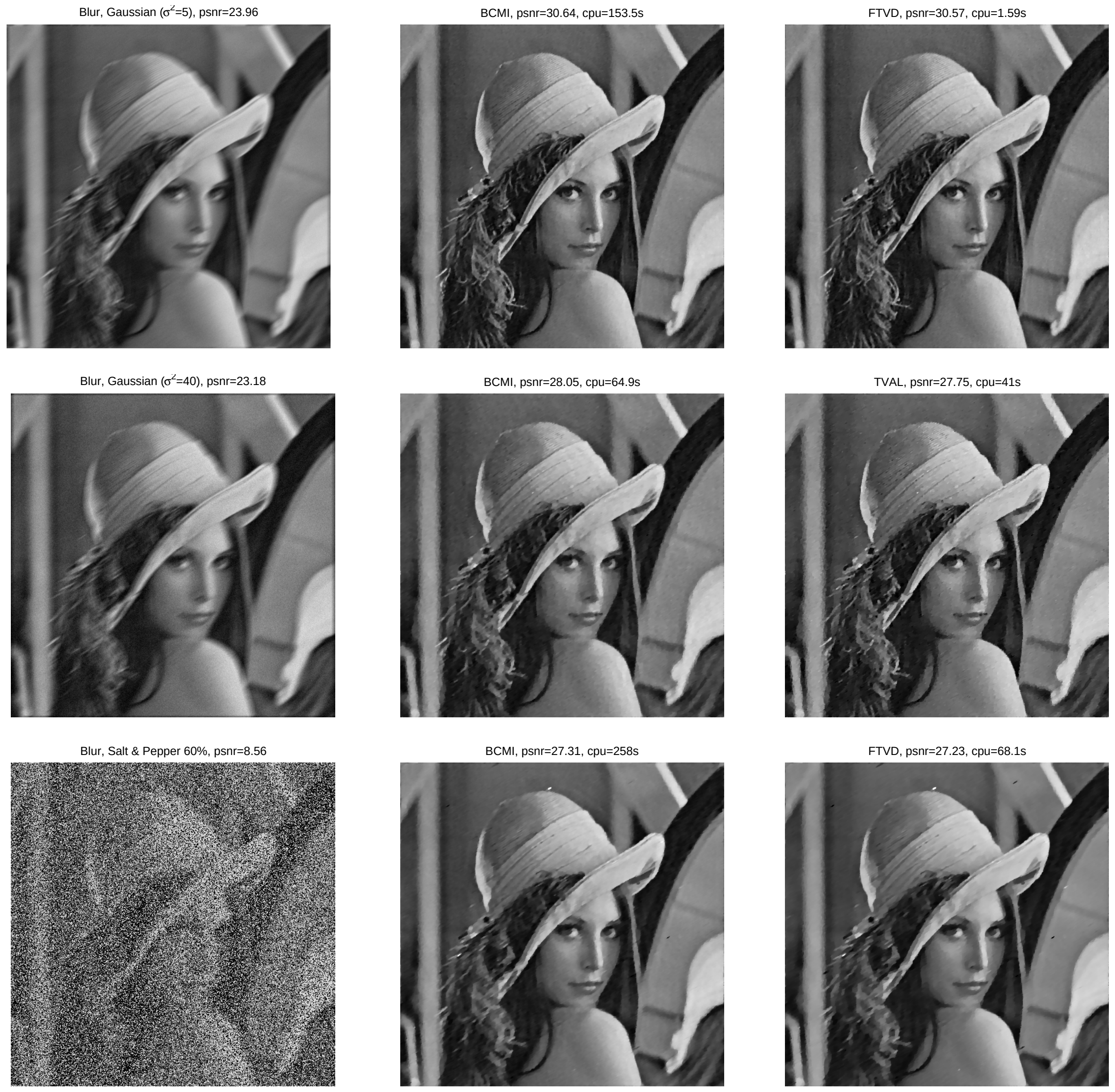

Blur, Gaussian (σ^2=5), psnr=23.96
BCMI, psnr=30.64, cpu=153.5s
FTVD, psnr=30.57, cpu=1.59s
Blur, Gaussian (σ^2=40), psnr=23.18
BCMI, psnr=28.05, cpu=64.9s
TVAL, psnr=27.75, cpu=41s
Blur, Salt & Pepper 60%, psnr=8.56
BCMI, psnr=27.31, cpu=258s
FTVD, psnr=27.23, cpu=68.1s

Fig. 8. Lena image reconstructions from blurred and noisy images with optimal smoothing values. Noises considered are Gaussian and salt-and-pepper, and estimation algorithms included are BCMI for the correct box-constrained MPL, projected FTVD and projected TVAL for approximate solutions.

in general, help to produce better (i.e. higher PSNR) restorations than methods based on simple projections. Although the improvements on PSNR were not huge, their zoomed in images demonstrate that the BCMI restorations are clearer.

We observed more significant improvements of BCMI over projected FTVD when using Satellite and Church images. A special feature of these two images is that they contain many black or white pixels, making box constraints more informative. Gaussian noises with mean 0 and variances 5, 30 and 50 were added to the motion blurred Satellite and Church images. Then BCMI and projected FTVD were applied to process these images using the optimal smoothing values given in Table I. Table II contains PSNR values and CPU seconds achieved by BCMI and projected FTVD when restoring Satellite and Church images. In all the cases BCMI outperformed projected FTVD, with PSNR improvements ranging from 0.1dB to 1.1dB.

D. *Optimal solution improves mean and variance of PSNR*

Here, we wished to establish that the optimal constrained reconstructions from BCMI are generally superior, judged by mean and variance of PSNR, than the estimates obtained from projected FTVD or projected TVAL.

We conducted a Monte-Carlo simulation with Boat image and Gaussian ($\sigma^2 = 40$) noise. In particular, we computed and compared the mean and standard deviation (std) of PSNR values for BCMI, FTVD and TVAL corresponding to different smoothing values. Since smoothing values for BCMI, FTVD and TVAL are totally different in magnitude, comparison of their restorations must be performed with care. We confronted this problem with the following strategy. Fifty noisy and blurry Boat images with Gaussian ($\sigma^2 = 40$) noise were generated using the same motion blur as before, and they were used to compare BCMI with FTVD. The circular boundary condition were included in these images. From the first five of these

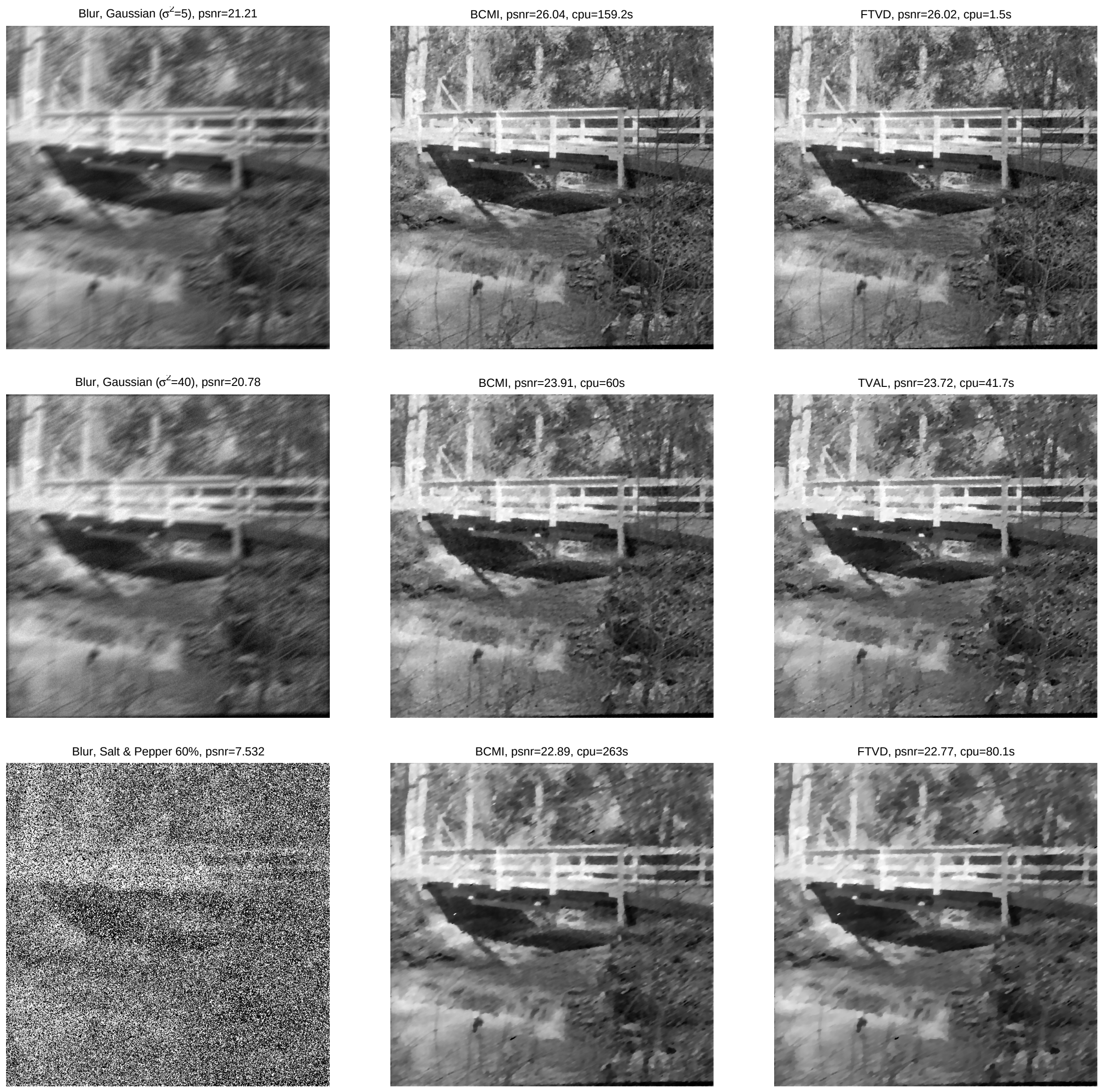

Blur, Gaussian (σ^2=5), psnr=21.21
BCMI, psnr=26.04, cpu=159.2s
FTVD, psnr=26.02, cpu=1.5s
Blur, Gaussian (σ^2=40), psnr=20.78
BCMI, psnr=23.91, cpu=60s
TVAL, psnr=23.72, cpu=41.7s
Blur, Salt & Pepper 60%, psnr=7.532
BCMI, psnr=22.89, cpu=263s
FTVD, psnr=22.77, cpu=80.1s

Fig. 9. Bridge image reconstructions from blurred and noisy images with optimal smoothing values. Noises considered are Gaussian and salt-and-pepper and estimation algorithms included are BCMI for the correct box-constrained MPL, projected FTVD and projected TVAL for approximate solutions.

images, we found by trial-and-error that the best smoothing parameter λ^* for BCMI and FTVD were 0.6 and 400 respectively. Now we chose nine equally-spaced points in the interval $[0.5\lambda^*, 1.5\lambda^*]$ (including the end-points) as possible smoothing values for the respective restoration method. For each of these nine smoothing values fifty restorations were obtained for each algorithm, thus resulted in fifty PSNR values. We calculated their mean and std; results are displayed in Fig. 12 (a) and (b) respectively. We then repeated the whole experiment to compare BCMI with TVAL, now with the zero boundary. Again, from the first five of the fifty simulated images, the best λ^* for BCMI and TVAL were 0.62 and 40 respectively. Mean and std of their PSNR values are displayed in Fig. 12 (c) and (d) respectively. We comment that the small differences in BCMI optimal smoothing values were caused by different boundary conditions.

Clearly, BCMI had better mean PSNR than both FTVD and TVAL over nearly all the selected smoothing values. FTVD however performed reasonably well in both mean and standard deviation: most of its mean PSNR were slightly less than BCMI and all of its standard deviation PSNR were nearly identical to BCMI. If we compare the best mean PSNR of the respective method, BCMI could give a 0.1dB improvement over both FTVD and TVAL. BCMI consistently produced smaller standard deviations in PSNR than TVAL. These findings suggest that the optimal box-constrained solutions can provide more accurate and stable restorations.

E. Compare BCMI with iterative steering kernel regression

The above studies establish that BCMI is fast and effective for image restoration from blurry and noisy images, and it can manipulate different noise models. Finally, we wished to compare this method with one of the state-of-the-art image denoising algorithms called iterative steering kernel

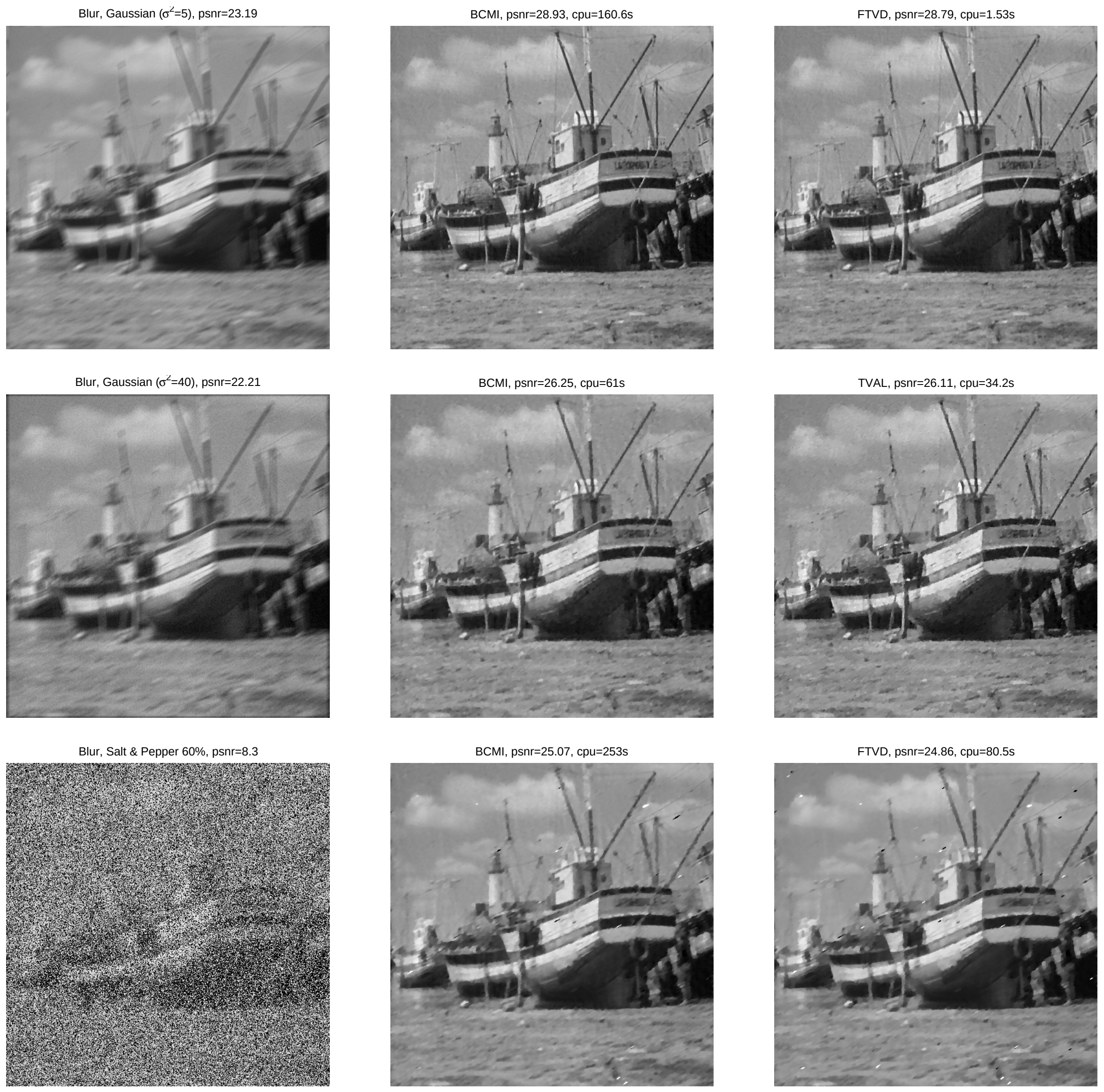

Blur, Gaussian (σ^2=5), psnr=23.19
BCMI, psnr=28.93, cpu=160.6s
FTVD, psnr=28.79, cpu=1.53s
Blur, Gaussian (σ^2=40), psnr=22.21
BCMI, psnr=26.25, cpu=61s
TVAL, psnr=26.11, cpu=34.2s
Blur, Salt & Pepper 60%, psnr=8.3
BCMI, psnr=25.07, cpu=253s
FTVD, psnr=24.86, cpu=80.5s

Fig. 10. Boat image reconstructions from blurred and noisy images with optimal smoothing values. Noises considered are Gaussian and salt-and-pepper and estimation algorithms included are BCMI for the correct box-constrained MPL, projected FTVD and projected TVAL for approximate solutions.

regression (ISKR). The details of this method can be found in [28] and its MATLAB package is available at ”http://users.soe.ucsc.edu/˜htakeda/KernelToolBox.htm”.
ISKR is data adapted kernel regression method for image processing and it is effective to clear noise from a contaminated image; see the experiment results reported in [28]. ISKR, however, cannot deblur images so we only compared BCMI and ISKR on their abilities to denoise images.

The test images used were Lena and Church. We added Gaussian noises to these images with variances 5, 20, 225 and 625. Then BCMI (with a TV penalty) and ISKR (with order $N = 2$) were applied to remove the noises. Table III compares their PSNR. It also includes information on smoothing values, number of iterations and CPU times. Note that the smoothing value for ISKR is called the “global smoothing parameter” in [28]. The BCMI smoothing values were optimal as they gave the best PSNR. The smoothing value for ISKR was less sensitive and it had to be combined with the iteration number to achieve the best PSNR.

We ran BCMI to the maximum of 1000 iterations unless it converged earlier (see Section V-A for the definition of convergence of BCMI). For BCMI, we used $\beta = 1$ and $\varepsilon = 3$ for both images. We ran ISKR to a number of iterations so that it gave the best PSNR at that iteration.

From Table III we observe that BCMI achieved higher PSNR than ISKR only when $\sigma^2 = 5$ or 20 (low noise level) for Lena image. In other cases, ISKR reached slightly higher PSNR than BCMI, particularly for heavy noises (i.e. $\sigma^2 = 225$ or 625). The CPU time demanded by BCMI is much smaller than ISKR for Lena image, but BCMI took longer CPU time than ISKR for Church image, mainly because BCMI did not converge in 1000 iterations in most cases.

The comparison in this example shows that, judged by

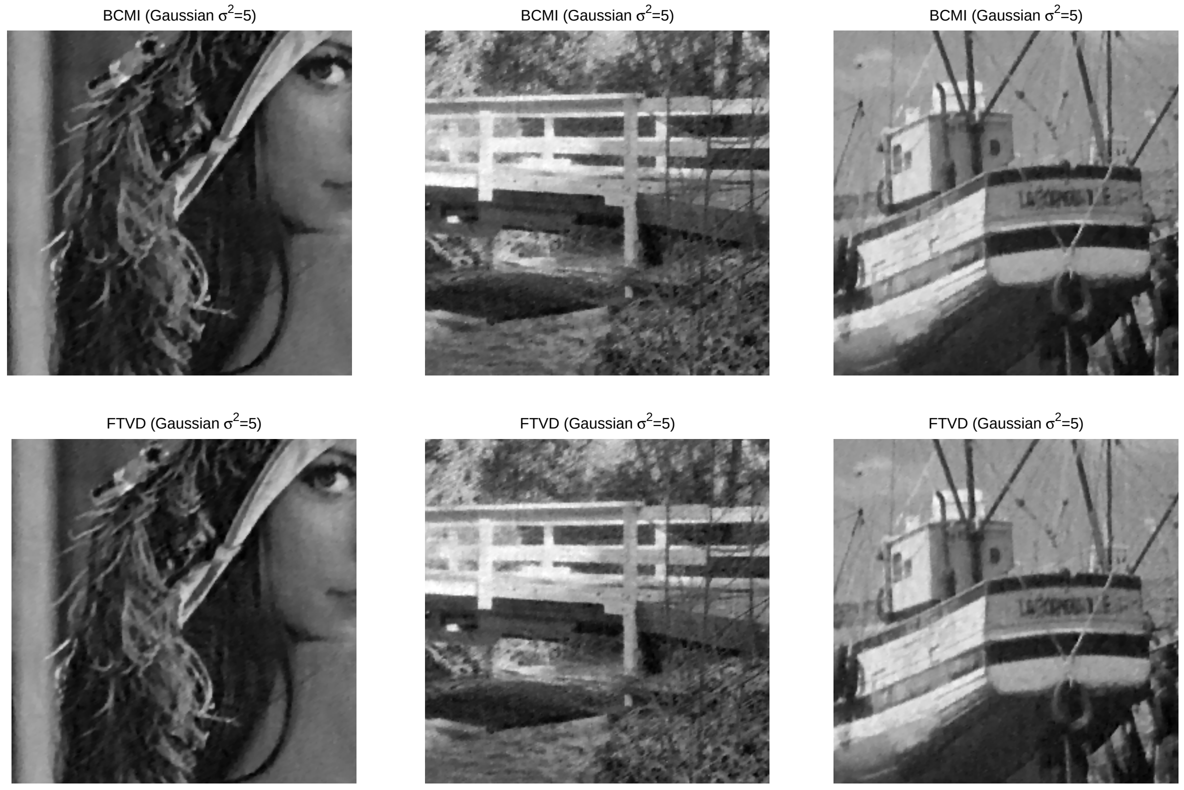


Fig. 11. Zoomed in Lena, Bridge and Boat images corresponding to the Gaussian ($\sigma^2 = 5$) cases. Algorithms included are BCMI and projected FTVD.

	Algorithm	Satellite		Church	
		PSNR (dB)	CPU (sec)	PSNR (dB)	CPU (sec)
Gaussian					
$\sigma^2 = 5$	BCMI	31.7	53.7	30.9	53.2
	FTVD	31.2	0.59	29.9	0.59
$\sigma^2 = 30$	BCMI	28.7	16.0	27.4	15.8
	FTVD	28.3	0.42	26.3	0.53
$\sigma^2 = 50$	BCMI	27.8	17.8	26.5	17.0
	FTVD	27.5	0.58	25.4	0.69
salt-and-pepper(60%)	BCMI	25.8	107.0	25.0	99.9
	FTVD	25.7	10.0	24.5	13

TABLE II
COMPARING BCMI AND PROJECTED FTVD USING SATELLITE AND CHURCH IMAGES, WITH GAUSSIAN AND SALT-AND-PEPPER NOISES.

PSNR of the restored images, BCMI with TV penalty is reliable for image denoising, and it is competitive with ISKR which uses a kernel regression to smooth.

VI. Conclusion

We developed a multiplicative iterative algorithm for the box-constrained image restoration problem. We showed that this algorithm converges to a stationary point under very general regularity conditions. Our numerical experiments demonstrated that this algorithm is effective for box-constrained TV deblurring denoising problems under three very general noise models, namely Gaussian, Poisson and salt-and-pepper. This experiment also demonstrated that our algorithm in general converges faster and can reach higher PSNR values than another fast box-constrained TV restoration algorithm called FISTA. We observed that a correct constrained solution can have as high as 1.1dB improvement over approximate solutions given by the projected FTVD or TVAL methods. However, we also noticed that for some images a correct box-constrained method can be trivial when compared with restorations obtained with projecting an unconstrained result to box constraints.

Finally, we comment that research on other box-constrained optimization methods for large scale problems, such as the conjugate gradient algorithm (for example see [29, Chapter 10]) and the spectral projected gradient algorithm [30], may also be useful for image deblurring and denoising with box constraints. But it requires further assessments to confirm the effectiveness of these algorithms.

Acknowledgement

The authors are grateful to the comments provided by the referees and the associate editor. Their help has greatly enhanced the quality of this paper.

	Algorithm	Lena				Church			
		smoothing value	iter #	PSNR (dB)	CPU (sec)	smoothing value	iter #	PSNR (dB)	CPU (sec)
Gaussian									
$\sigma^2 = 5$	BCMI	0.6	45	42.08	9.42	0.9	1000	44.12	76.86
	ISKR	1.7	1	39.96	84.05	1.7	1	44.22	22.03
$\sigma^2 = 20$	BCMI	2	82	37.56	22.35	2.3	1000	39.23	81.54
	ISKR	1.7	1	37.42	88.80	1.7	1	39.44	22.46
$\sigma^2 = 225$	BCMI	11	528	31.49	164.19	10	1000	31.25	87.39
	ISKR	2.4	5	32.61	410.44	2.4	4	32.06	84.05
$\sigma^2 = 625$	BCMI	20	693	29.24	214.69	18	791	27.71	72.73
	ISKR	2.4	11	30.62	882.09	2.4	8	28.40	167.75

TABLE III
COMPARING BCMI AND ISKR USING THE LENA AND CHURCH IMAGES WITH GAUSSIAN NOISES.

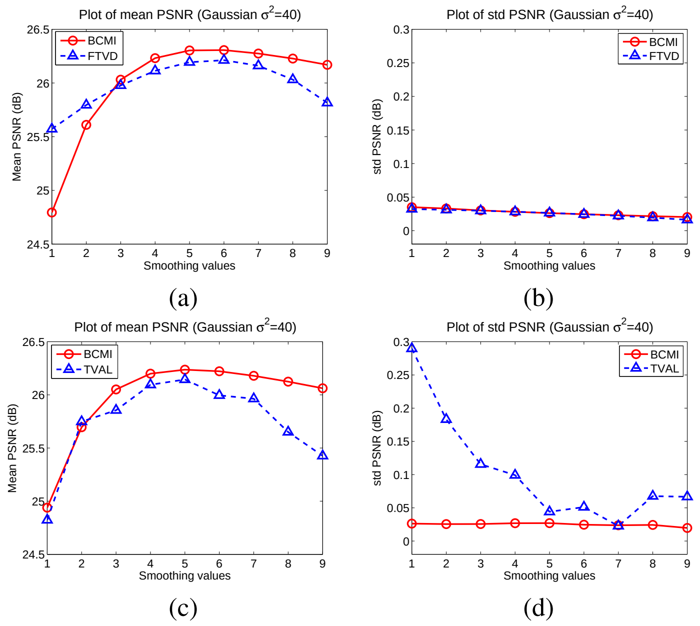


Fig. 12. Plots of the mean (panels (a) and (c)) and standard deviation (panels (b) and (d)) of PSNR corresponding to nine smoothing values. Panels (a) and (b) compare BCMI with projected FTVD, and panels (c) and (d) compare BCMI with projected TVAL.

REFERENCES

[1] A.K. Jains, *Fundamentals of Digital Image Processing*, Prentice Hall.

[2] R. J. Marks, G. L. Wise, D. G. Haldeman, and J. L. Whited, "Detection in Laplace noise," *IEEE Trans. on Aeros. and Elect. Syst.*, vol. 14, pp. 866–872, 1978.

[3] D. Kim, S. Sra, and I. Dhillon, "Tackling box-constrained optimization via a new projected quasi-newton approach," *SIAM J. Sci. Comput.*, vol. 32, pp. 3548–3563, 2010.

[4] B. Morini, M. Pocelli, and R. H. Chan, "A reduced Newton method for constrained linear least-squares problems," *J. of Comp. and Appl. Math*, vol. 233, pp. 2200–2212, 2010.

[5] A. Becky and M. Teboulle, "Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems," *IEEE Trans. Image Processing*, vol. 18, pp. 2419–2434, 2009.

[6] J. Ma, "Multiplicative algorithms for maximum penalized likelihood inversion with nonnegative constraints and generalized error distributions," *Communications in Statistics - Theory and Methods*, vol. 35, Issue 5, pp. 831–848, 2006.

[7] J. Ma, "Positively constrained multiplicative iterative algorithm for maximum penalized likelihood tomographic reconstruction," *IEEE Trans. Nuc. Scie.*, vol. 57, pp. 181–192, 2010.

[8] L. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, pp. 259–268, 1992.

[9] C. Bouman and K. Sauer, "A generalized Gaussian image model for edge-preserving MAP estimation," *IEEE Trans. Image Proc.*, vol. 2, pp. 296–310, 1993.

[10] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. IMAGING SCIENCES*, vol. 1, pp. 248–272, 2008.

[11] J. Yang, Y. Zhang, and W. Yin, "An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise," *SIAM J. Sci. Comput.*, vol. 31, pp. 2842–2865, 2009.

[12] R. H. Chan, H. Liang, and J. Ma, "Positively constrained total variation penalized image restoration," *Advances in Adaptive Data Analysis*, vol. 3, pp. 187–201, 2011.

[13] D. Luenberger, *Linear and Nonlinear Programming (2nd edition)*, J. Wiley, 1984.

[14] W. Zangwill, *Nonlinear Programming: A Unified Approach*, Prentice-Hall, New Jersey, 1969.

[15] C. F. Wu, "On the convergence properties of the EM algorithm," *The Annals of Statistics*, vol. 11, pp. 95–103, 1983.

[16] J. M. Ostrowski, *Solution of equations and system of equations (2nd edition)*, Academic, New York, 1966.

[17] J. Yang, W. Yin, and Y. Wang, "Fast algorithm for edge-preserving variational multichannel image restoration," *SIAM J. IMAGING SCIENCES*, vol. 2, pp. 569–592, 2009.

[18] A. Chambolle, "An algorithm for total variation minimization and applications," *J. Mathematical Imaging and Version*, vol. 20, pp. 89–97, 2004.

[19] K. Bredies, "A forwardbackward splitting algorithm for the minimization of non-smooth convex functionals in banach space," *Inverse Problems*, vol. 25, pp. 1–20, 2009.

[20] C. Wu, J. Zhang, and X. C. Tai, "Augmented lagrangian method for total variation restoration with non-quadratic fidelity," *Inverse Problems and Imaging*, vol. 5, pp. 237–261, 2011.

[21] C. R. Vogel and M. E. Oman, "Fast, robust total variation-based reconstruction of noisy, blurred images," *IEEE Trans. Image Processing*, vol. 7, pp. 813–824, 1998.

[22] M. Lysaker and X. C. Tai, "Iterative image restoration combining total variation minimization and a second-order functional," *Internation J. of Comp. Vision*, vol. 66, pp. 5 – 18, 2006.

[23] P. L. Combettes and V. R. Wajs, "Signal recovery by proximal forward-backward splitting," *Multiscale Model. Simul.*, vol. 4, pp. 1168–1200, 2005.

[24] Y. Vardi, L.A. Shepp, and A. Kaufman, "A statistical model for positron emission tomography (with discussion)," *J. Amer. Stat. Assoc.*, vol. 80, pp. 8–37, 1985.

[25] R. Chan, C. H. Ho, and M. Nikolova, "Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization," *IEEE Trans. Image Processing*, vol. 14, pp. 1479–1485, 2005.

[26] V. R. Vijaykumar, P. T. Vanathi, P. Kanagasabapathy, and D. Ebenezer, "Robust statistics based algorithm to remove salt and pepper noise in images," *Int. J. of Info. and Commu. Eng.*, vol. 5, pp. 164–173, 2009.

[27] E. Abreu, M. Lightstone, and S.K. Mitra, "A new efficient approach for the removal of impulse noise from highly corrupted images," *IEEE Trans. Image Processing*, vol. 5, pp. 1012–1025, 1996.

[28] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Trans. Image Processing*, vol. 16, pp. 349 – 366, 2007.

[29] R. Pytlak, *Conjugate gradient algorithms in nonconvex optimization*, Springer Verlag, Berlin.

[30] E. G. Birgin and J. M. Martinez, "Large-scale active-set box-constrained optimization method with spectral projection gradients," *Computational Optimization and Applications*, vol. 22, pp. 101 – 125, 2002.