CAAM Technical Report TR07-10

A Fast Algorithm for Image Deblurring with Total Variation Regularization

Yilun Wang, Wotao Yin and Yin Zhang

Department of Computational and Applied Mathematics
Rice University, Houston, Texas, 77005, U.S.A.

June 22, 2007 (Revised July 2, 2007)

Abstract

We propose and test a simple algorithmic framework for recovering images from blurry and noisy observations based on total variation (TV) regularization when a blurring point-spread function is given. Using a splitting technique, we construct an iterative procedure of alternately solving a pair of easy subproblems associated with an increasing sequence of penalty parameter values. The main computation at each iteration is three Fast Fourier Transforms (FFTs). We present numerical results showing that a rudimentary implementation of our algorithm already performs favorably in comparison with two of the existing start-of-the-art algorithms. In particular, it runs orders of magnitude faster than a number of existing algorithms for solving TVL2-based de-convolution problems to good accuracies.

1 Introduction

One of the important tasks in signal processing is to recover signals like images from noisy and blurry observations. There are many approaches based on statistics [8, 2], Fourier and/or wavelet transforms [11, 13], or variational analysis [19, 3, 6] for this task. Among them the total variation regularization [25] has become one of the standard techniques known for preserving sharp discontinuities. The sharp discontinuities in images are edges and object boundaries, often the most important features to keep and capture in image processing.

The standard total variation model recovers an image from a blurry and noisy observation f by solving the problem:

$$\min_u \int_\Omega \|\nabla u(x)\| \mathrm{d}x + \frac{\lambda}{2} \int_\Omega |Ku(x) - f(x)|^2 \mathrm{d}x \tag{1}$$

where Ω is an open rectangular domain in $\mathbb{R}^2$, K is a given linear blurring operator, $f : \Omega \to \mathbb{R}$ is the observed image in $L^2(\Omega)$, u is the unknown image to restore, and $\|\nabla u(x)\|$ is the Euclidean norm of the gradient vector $\nabla u(x) \in \mathbb{R}^2$. The term $\int_\Omega \|\nabla u(x)\| \mathrm{d}x$ is the total variation (TV) of the function u whenever u is differentiable. We note that in general functions of finite total variations are not necessarily differentiable (see the book [25] for details). However, since we are only interested in solving (1) in a discrete domain for digital image processing, we can tacitly assume differentiability of u and use the expression $\int_\Omega \|\nabla u(x)\| \mathrm{d}x$ without loss of generality.

In the model (1), the observed image f is formulated as the sum of a Gaussian noise n and a blurry image $K\bar{u}$ resulting from the linear blurring operator K acting on the clean image $\bar{u}$, i.e.,

$$f(x) = K\bar{u}(x) + n(x). \tag{2}$$

Among all linear blurring operators, many are shift-invariant and can be expressed in the form of convolution [15]:

$$(Ku)(x) = \int_\Omega h(x - s)u(s)\mathrm{d}s \equiv (h \star u)(x) \tag{3}$$

where h is the so-called *point spread function* (PSF) associated with K. Let $\mathcal{F}(\cdot)$ denote the discrete Fourier transform, and the symbol "$\circ$" the point-wise product. We will base our method on the property $\mathcal{F}(Ku) = \mathcal{F}(h \star u) = \mathcal{F}(h) \circ \mathcal{F}(u)$; therefore, we assume that K is shift-invariant and its corresponding PSF h is known.

We now briefly describe some existing state-of-the-art methods for image deblurring. We first mention the algorithm ForWaRD by Neelanmani, Choi and Baraniuk [14]. It is a hybrid method based on the Fourier-based deconvolution [11] and the wavelet-vaguelette deconvolution [9]. The code ForWaRD has exhibited impressive performance for image deblurring in all aspects except contrast preservation and artifact control. The restored images often have slightly lower contrast compared to the originals, as well as some ripple artifacts, but its computation is very fast.

Following the pioneering work [20, 19], various numerical methods have been developed for solving the total variation regularization model (1), which is known for preserving sharp edges while removing singles of small-scales including noise [7]. Since (1) is a convex minimization problem, a function u is optimal if and only if it satisfies the first-order optimality condition for (1), which is the Euler-Lagrange equation:

$$\nabla \cdot \frac{\nabla u}{\|\nabla u\|} - \lambda K^\star(Ku - f) = 0, \tag{4}$$

along with the Neumann condition on the boundary. To avoid division by zero, it is common to approximate (4) by:

$$\nabla \cdot \frac{\nabla u}{\|\nabla u\|_a} - \lambda K^\star (Ku - f) = 0, \tag{5}$$

where $\|\cdot\|_a$ denotes $\sqrt{\|\cdot\|^2 + a^2}$ for some small $a > 0$, and $K^\star$ is the adjoint operator of K.

Rudin and Osher [19] applied the artificial time marching method, equivalent to the steepest descent method,

$$u \leftarrow u + \Delta t \left(\nabla \cdot \frac{\nabla u}{\|\nabla u\|} - \lambda K^\star (Ku - f) \right) \tag{6}$$

for solving (5). This method is easy to implement, but converges slowly due to poor conditioning of the diffusion operator $\nabla \cdot (\nabla(\cdot)/\|\nabla(\cdot)\|_a)$. To address this difficulty, in [24, 5] Vogel and Oman proposed a "lagged diffusivity" procedure that solves the following equations for $u^{(n+1)}$ iteratively,

$$\nabla \cdot \left(\frac{\nabla u^{(n+1)}}{\|\nabla u^{(n)}\|_a} \right) - \lambda K^\star (Ku^{(n+1)} - f) = 0, \tag{7}$$

with the Neumann boundary condition. In [4], Chan, Golub and Mulet introduced the primal-dual formulation:

$$\begin{aligned} \nabla \cdot \vec{w} - \lambda K^\star (Ku - f) &= 0, & (8) \\ \vec{w} \|\nabla u\| - \nabla u &= 0, & (9) \end{aligned}$$

which is then solved by Newton's method. Recently, Goldfarb and Yin [10] proposed a unified approach for various total variation models including (1) based on second-order cone programming. They model problem (1) as a second-order cone program, which is then solved by modern interior-point methods. Standard interior-point methods for second-order cone programs require $K^\star K$ to be explicitly expressed, and a matrix involving K and $K^\star$ to be factorized at every iteration. Thus this approach requires excessive memory and becomes impractical for large-scale problems. However, this method yields highly accurate solutions that can be used as references for evaluating solution qualities of other methods.

In this paper, we propose a simple algorithmic framework for solving (1) that does not require any modification to the non-differentiable term $\|\nabla u\|$. We first replace the gradient ∇u by a new variable $\mathbf{w}$, and then penalize the discrepancy between $\mathbf{w}$ and ∇u by the quadratic penalty term $(\beta/2) \int_\Omega \|\mathbf{w} - \nabla u\|^2 \mathrm{d}x$. Our method then alternately updates $\mathbf{w}$ and u with an increasing penalty parameter β until "convergence". At each iteration, the dominant computation is three fast Fourier transforms (FFTs), or alternatively three discrete cosine transforms (DCTs). The proposed framework is so simple that our MATLAB implementation contains only a few dozen lines. Yet, in

numerical experiments our MATLAB code competes comfortably with C/C++ implementations of two state-of-the-art algorithms.

The rest of the paper is organized as follows. In Section 2, we describe a discrete model of (1) for digital gray-scale images and present the proposed algorithmic framework in details. In Section 3, we compare a very basic implementation of the proposed algorithm to the lagged diffusivity and ForWaRD algorithms by conducting numerical experiments on four test images with either motion or Gaussian blurring and additive white noise. Finally, we offer some concluding remarks in Section 4.

2 An FFT-based algorithm–FTVd

2.1 A New Formulation

We begin by introducing a new variable $\mathbf{w}$ to represent the gradient term ∇u in model (1), generating an equivalent, constrained convex minimization problem:

$$\begin{array}{ll} \min\limits_{u,\mathbf{w}} & \int_\Omega \|\mathbf{w}\| \mathrm{d}x + \dfrac{\lambda}{2}\int_\Omega |Ku-f|^2 \mathrm{d}x \\ \text{s.t.} & \mathbf{w}(x) = \nabla u(x), \quad x \in \Omega. \end{array} \tag{10}$$

We consider the L_2-norm-square penalty formulation

$$\min_{\mathbf{w},u} \int_\Omega \|\mathbf{w}\| \mathrm{d}x + \frac{\lambda}{2}\int_\Omega |Ku-f|^2 \mathrm{d}x + \frac{\beta}{2}\int_\Omega \|\mathbf{w}-\nabla u\|^2 \mathrm{d}x \tag{11}$$

where $\beta > 0$ is the penalty parameter. It is well-known (see [1], for example) that as β goes to infinity, the solution of the convex minimization problem (11) converges to that of (10), or of (1). For practical purposes, it suffices to seek a solution of (11) for a sufficiently large β in order to approximate the solution of (1) to a requested accuracy.

Let us consider the discrete form of the problem. We let matrix $f \in \mathbb{R}^{N\times N}$ represent a two-dimensional gray-scale digital image, where each component f_{ij} is the intensity value of pixel (i,j) for $i,j \in \{1,2,\cdots,N\}$. Similarly, we let matrix $u \in \mathbb{R}^{N\times N}$ represent the unknown image to be restored. For simplicity we have tacitly assumed that the images are square images, though rectangular images can be treated in exactly the same way.

Let us define the forward finite difference operator $\partial^+ : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N} \otimes \mathbb{R}^{N\times N}$ by

$$(\partial^+ u)_{ij} := \begin{pmatrix} (\partial_1^+ u)_{ij} \\ (\partial_2^+ u)_{ij} \end{pmatrix} := \begin{pmatrix} u_{i+1,j} - u_{ij} \\ u_{i,j+1} - u_{ij} \end{pmatrix} \in \mathbb{R}^2, \tag{12}$$

for $i,j = 1,\cdots,N$ with appropriate boundary adjustments for $i = N$ and $j = N$. We mention that other finite difference operators also work in the proposed algorithmic framework as long as they can be represented as discrete convolution operators.

Under the above definitions, the corresponding discrete form of (11) is

$$\min_{\mathbf{w},u} \sum_{i,j=1}^{N} \|\mathbf{w}_{ij}\| + \frac{\beta}{2} \sum_{i,j=1}^{N} \|(\partial^+ u)_{ij} - \mathbf{w}_{ij}\|^2 + \frac{\lambda}{2}\|Ku - f\|_F^2, \tag{13}$$

where $K : \mathbb{R}^{N\times N} \to \mathbb{R}^{N\times N}$ is a discrete convolution operator, $\|\cdot\|_F$ is the Frobenius norm, and

$$\mathbf{w}_{ij} = \begin{pmatrix} (w_1)_{ij} \\ (w_2)_{ij} \end{pmatrix} \in \mathbb{R}^2, \quad i, j \in \{1, 2, \cdots, N\}.$$

2.2 Alternating Minimization

We will solve problem (13) by alternately minimizing the objective function with respect to $\mathbf{w}$ while fixing u, and vice versa. As such, we need to solve two types of subproblems. One is the so-called "$\mathbf{w}$-subproblem" for a fixed u:

$$\min_{\mathbf{w}} \sum_{i,j=1}^{N} \left(\|\mathbf{w}_{ij}\| + \frac{\beta}{2}\|\mathbf{w}_{ij} - (\partial^+ u)_{ij}\|^2 \right), \tag{14}$$

where we observe that the objective function, which consists of the first two terms in (13), is separable for each $\mathbf{w}_{ij}$.

The other subproblem is the "u-subproblem" for a fixed $\mathbf{w}$:

$$\min_{u} \frac{\lambda}{2}||Ku - f||_F^2 + \frac{\beta}{2}||\partial_1^+ u - w_1||_F^2 + \frac{\beta}{2}||\partial_2^+ u - w_2||_F^2 \tag{15}$$

where $\partial_1^+ u = [(\partial_1^+ u)_{ij}] \in \mathbb{R}^{N\times N}$, $w_1 = [(w_1)_{ij}] \in \mathbb{R}^{N\times N}$, and so on. It is easy to see that the above objective function is just another expression of the last two terms in (13).

Obviously, the effectiveness of our alternating minimization approach will largely depend on how fast we can solve the two subproblems. We now show that indeed these two subproblems can be solved by very fast procedures.

It is not difficult to verify that the $\mathbf{w}$-subproblem permits a closed-form solution:

$$\mathbf{w}_{ij} = \max\left(\|(\partial^+ u)_{ij}\| - \frac{1}{\beta}, 0 \right) \frac{(\partial^+ u)_{ij}}{\|(\partial^+ u)_{ij}\|}, \quad 1 \le i, j \le N, \tag{16}$$

where we follow the convention $0 \cdot (0/0) = 0$. The calculation in (16) has a linear-time complexity of order $O(N^2)$ for an N by N image. Hence, the $\mathbf{w}$-subproblem can be solved quickly.

The u-subproblem is a linear least-squares problem with a special structure; that is, the three involved linear operators, K, ∂_1^+ and ∂_2^+, are all discrete convolutions. Hence, according to the well-known convolution theorem (see [23], for example), these operators become point-wise products under the Fourier transform (as well as under a number of other transforms).

Let $\mathcal{F}(u)$ denote the discrete Fourier transform of u, and similarly for other quantities in $\mathbb{R}^{N\times N}$. Since the Fourier transform does not change the Frobenius norm (Parseval's theorem) and converts convolutions into point-wise products, the least squares problem (15) takes the following form in the Fourier frequency domain:

$$\min_u ||\mathcal{F}(h)\circ\mathcal{F}(u)-\mathcal{F}(f)||_F^2+\gamma\left(||\mathcal{F}(\partial_1^+)\circ\mathcal{F}(u)-\mathcal{F}(w_1)||_F^2+||\mathcal{F}(\partial_2^+)\circ\mathcal{F}(u)-\mathcal{F}(w_2)||_F^2\right)$$

where "$\circ$" denotes element-wise multiplication and $\gamma=\beta/\lambda$. After solving the above problem for $\mathcal{F}(u)$, we take an inverse discrete Fourier transform, $\mathcal{F}^{-1}(\cdot)$, to obtain the solution to the u-subproblem (15) in the image space, that is,

$$u=\mathcal{F}^{-1}\left(\frac{\mathcal{F}(h)^*\circ\mathcal{F}(f)+\gamma\left(\mathcal{F}(\partial_1^+)^*\circ\mathcal{F}(w_1)+\mathcal{F}(\partial_2^+)^*\circ\mathcal{F}(w_2)\right)}{\mathcal{F}(h)^*\circ\mathcal{F}(h)+\gamma\left(\mathcal{F}(\partial_1^+)^*\circ\mathcal{F}(\partial_1^+)+\mathcal{F}(\partial_2^+)^*\circ\mathcal{F}(\partial_2^+)\right)}\right) \tag{17}$$

where "$*$" denotes complex conjugacy and the division is element-wise.

For any given $\beta>0$, we will solve problem (13) by applying formulas (16) and (17) alternately, starting from an initial u. For our convex program (13), this alternating minimization procedure will have guaranteed convergence (for example, see [22]).

We observe that in the right-hand side of (17) only w_1 and w_2 change at every iteration. Hence the solution of the u-subproblem (15) requires two fast Fourier transforms and one inverse transform and a total complexity in the order of $O(N^2\log(N^2))=O(N^2\log(N))$. Clearly, the discrete Fourier transform $\mathcal{F}$ can be replaced by the discrete cosine transform, which also avoids the use of complex conjugacy. The discrete cosine transform entails the same complexity $O(N^2\log(N))$ as the discrete Fourier transform, but implies an *even* extension of the data rather than a *periodic* extension with the discrete Fourier transform. This difference in boundary conditions may affect the accuracies of image processing. For further discussion on this topic, we refer the interested reader to the book [16].

2.3 A Continuation Algorithm: FTVd

How well the solution of (13) approximates that of (1) or its constrained equivalent (10) depends on the magnitude of β which determines the amount of penalty applied to the discrepancy between ∇u and $\mathbf{w}$ in the squared L^2-distance. In practice, β cannot be too small because it would allow $\nabla u=\mathbf{w}$ to be violated excessively. Though it may be tempting to set β to a very large value from outset to force an accurate $\mathbf{w}$-approximation to ∇u, using a single large β value is far from desirable. It should be clear from the $\mathbf{w}$-updating formula (16) that the larger the β is, the less the amount of update applied to $\mathbf{w}$ (and subsequently to u), making the algorithm take tiny steps and many more iterations. Therefore, we should choose β in a way that balances the speed and

accuracy considerations. We achieve this by a continuation strategy using an increasing sequence of β. That is, we start with a small β value and gradually increase it to a desired level, while using the approximate solution obtained for the current β value as the initial guess for the next β value. From a theoretical point of view, we try to approximately follow the path $(\mathbf{w}(\beta), u(\beta))$ from a small β value to infinity, where the pair $(\mathbf{w}(\beta), u(\beta))$ represent the solution of (13) (assuming it is unique) for any fixed $\beta > 0$.

To test the viability of the proposed continuation (or path-following) algorithmic framework, we choose to implement a most basic version of the framework, which we call "Fast Total Variation de-convolution" or FTVd, as is given below.

Algorithm FTVd:

Input: f, $tol > 0$ and $\lambda > 0$

Initialize: $u = f$, $u_p = 0$ and $\beta > 0$ and $\beta_{max} \gg 0$

While $\beta < \beta_{max}$, **Do**

While $\|u - u_p\|_F \geq tol\ \|u_p\|_F$, **Do**

1) Save the previous iterate: $u_p = u$.

2) Compute $\mathbf{w}$ according to (16) for fixed u.

3) Compute u according to (17) for fixed $\mathbf{w}$.

End Do

Increase β .

End Do

Output: u

This procedure contains two loops of iterations where each outer iteration corresponds to a fixed β value. One can easily modify the above framework to make it more flexible. For example, one could implement an adaptive tolerance that varies with β from one outer iteration to another, or use some other stopping criterion for the inner iterations.

In our numerical experiments, we used the following "default" parameter setting. We set the initial value of β to 4, its terminal value to $\beta_{\max} = 2^{20}$, and the inner-loop tolerance to $tol = 5.0 \times 10^{-4}$ constantly. At the end of each outer iteration, we *double* the value of β. This very basic version of implementation has proved to be sufficiently effective to serve the purpose of our proof-of-concept experiments.

3 Numerical Experiments

In model (1), the parameter λ controls the amount of penalty applied to the L^2-distance square between Ku and f. According to (2), the squared L^2-distance between $K\bar{u}$ and f, where $\bar{u}$ is the true clean image, is equal to the variance of the additive Gaussian noise n. Therefore, an appropriate λ should give a solution u of (1) satisfying $\|Ku - f\| \approx \|K\bar{u} - f\|$. In our experiments, we have observed, as others have, that when λ was too large the restored images were noisy; when λ was too small, the restored images were blocky and had small-scale details missing. The issue of how to select λ is important but beyond the scope of this work. In our experiments, we used the formula $\lambda = 0.05/\max(\sigma^2, 10^{-12})$ for images with intensity scaled to the range $[0, 1]$, where σ is the standard deviation of the Gaussian noise in (2). This formula is based on the observation that λ should be inversely proportional to the noise variance, while the constant 0.05 was determined empirically so that the restored images had reasonable signal-to-noise ratios (SNRs) and relative errors (their precise definitions are given in Subsection 3.4). This formula is sufficient for the purpose of our experimentation, but not meant for general use. For one thing, it cannot be applied when the value of σ is unknown, which is normally the case in practice. More practical techniques exist for choosing λ, often by testing on a small window of image. The reader interested in the topic of selecting λ is referred to [21] for a study on the relationship between λ and the geometric scales of u.

3.1 Overall Impressions with Various Algorithms

We mentioned several popular algorithms for solving the TV deblurring model (1) in the introduction, and tested all of them except the algorithm (PD) by Chan, Golub and Mulet [4] based on the primal-dual equations (8) and (9) due to technical difficulties. Here we briefly describe our experience with these algorithms. In the next subsection, we present in greater details the comparison results between FTVd and the lagged diffusivity algorithm (LD) and between FTVd and ForWaRD.

The artificial time-marching algorithm [20] can be easily implemented in MATLAB. It is a good choice if one is satisfied with below-average solution accuracies. When higher accuracies were required, it took a large number of iterations, each with a small step size governed by the CFL condition. The second-order cone programming (SOCP) approach on the contrary attained solutions with at least five digits of accuracy but took longer times and a large amount of memory for processing mid/large-sized images. The size 512×512 is close to the limit of what the SOCP solver Mosek [12] can handle on a workstation with 2Gb of memory. The paper [10] presented a comprehensive comparison between the artificial time-marching algorithm and the second-order

cone programming approach. We did not test PD, but expect its performance to be similar to SOCP since PD also solves large systems of linear equations in the framework of Newton's method. We also tested the recent method [18] based on an iterative reweighted norm algorithm, and obtained consistent speed and accuracies as reported by the authors. In particular, it was slower than the lagged diffusivity algorithm (LD) for solving the TV deblurring model (1) (though faster on a pure denoising model, as well as those with the L_1-fidelity). Therefore, the algorithm LD was most efficient among those described in the introduction for solving model (1) besides our new algorithm. The algorithm ForWaRD dose not solve model (1), but utilizes sophisticated techniques based on the Fourier and wavelet transforms. It is very fast, but generate outputs with somewhat different characteristics.

Clearly, there are many other deblurring algorithms that are not included in our discussion and comparison, but a more exhaustive comparison is beyond the scope of this work.

3.2 Codes and Test Platforms

From our computational experience, we consider the LD algorithm to be a good representative for those algorithms that solve the TV deblurring model (1), and ForWaRD to be one for other deblurring algorithms. Therefore, we chose to compare FTVd with LD and ForWaRD on both quality and speed. We mention that while FTVd is written entirely in MATLAB, the cores of LD and ForWaRD are written in C/C++ with MATLAB calling interfaces.

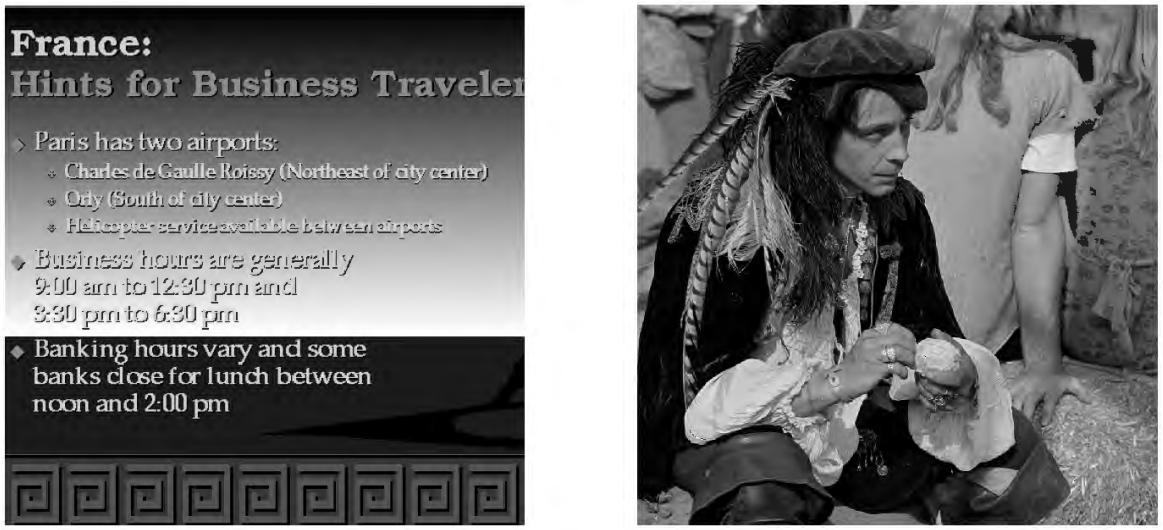
All comparisons between FTVd and LD were performed under Linux v2.6.9 and MATLAB v7.2 running on a 64-bit SUN workstation with an AMD Opteron 148 CPU and 3GB of memory. The C++ code of LD was a part of the library package NUMIPAD [17] and was compiled with gcc v3.4.6.

Since we could not compile ForWaRD under the same environment as LD or vice versa, we had to run all comparisons between FTVd and ForWaRD under Windows XP and MATLAB v7.0 running on a Dell Laptop with a Pentium M 1.70 GHz CPU and 512 MB of memory. The pre-compiled code of ForWaRD was downloaded from the webpage of the Rice DSP group at `http://www.dsp.rice.edu/software/ward.shtml`.

3.3 Original, Blurry, and Noisy Images

We obtained two gray-scale images, France (512×512) and Man (1024×1024) given in Figure 1, from the Internet at `http://decsai.ugr.es/cvg/dbimagenes/index.php`. The image France has sharp texts with a simple background that is smooth in the upper half. In contrast to France, the image Man consists of complex components in different scales, with different patterns, and under

inhomogeneous illuminations. Both image are suitable for our deblurring experiments.



(a) France: 512×512 (b) Man: 1024×1024

Figure 1: Original Test Images

We first scaled the intensities of the original images into the range between 0 and 1 before we began our experiments. We tested two typical types of blurring effects: motion and Gaussian (out-of-focus), applying motion blurring to France and Gaussian blurring to Man, each with two levels of severity. Specifically, we used the functions "fspecial" and "imfilter" from the MATLAB Image Processing Toolbox with the types "motion" and "gaussian", and then added to the resulting blurry images the Gaussian white noise with a mean 0 and a standard deviation $1.e$-3. With the "motion" blurring, we set the angle parameter "theta" to 135 and the motion distance parameter "len" to two different values 21 and 91 corresponding to the medium and severe levels of blurring, respectively. With the "medium" Gaussian blurring, we set the blurring window size "hsize" equal to 21 and the standard deviation "sigma" equal to 5, and with the "severe" Gaussian blurring, 41 and 10, respectively.

We summarize the information on the four test images in Table 1. The test images themselves are given in the first rows of Figures 4 and 5. It is worth mentioning that we use the terms "medium" and "severe" primarily for convenience of reference, knowing that the severity of these levels may be interpreted differently by others.

The blurring kernels were applied to the two original images by the MATLAB function "imfilter" with the "circular" option. In this setting, images are assumed to be periodic, so the pixels outside the left part of the image boundary are assumed to be identical to those on the right and the same

Table 1: Information on Test Images

Test Image	Original Image	Size	Blurring Type	Location
1	France	512 × 512	"medium" motion	Figure4
2	France	512 × 512	"severe" motion	Figure 4
3	Man	1024 × 1024	"medium" Gaussian	Figure 5
4	Man	1024 × 1024	"severe" Gaussian	Figure 5

applies to the top and bottom. The "circular" boundary with the blurring kernels is assumed by FTVd in calculating the finite differences at the far-ends of the boundaries where either $i = N$ or $j = N$. We are not entirely certain what boundary condition has been implemented in the LD code. However, we conducted tests with different boundary conditions applied to blurrings, and observed little difference in overall deblurring quality or running time by any of the codes, apparently due to the fact that the test images are much larger in size than the blurring kernels.

3.4 Comparisons in Quality and Speed

The quality of restored images are measured by their SNRs and relative errors to the original clean images. SNR stands for the signal-to-noise ratio, defined as 10 times the 10-based logarithm of the ratio of signal variance to noise variance. The relative error is defined as the norm of error over that of the clean image in Frobenius norm. The higher the SNR or the lower the relative error, the better the quality of restoration. The SNRs and relative errors of the restored images are plotted in Figures 2 and 3, respectively.

For motion deblurring, the TV-based algorithms FTVd and LD yielded images with higher SNRs. While FTVd computed the solutions of (1) with satisfactory accuracies in both tests, LD ran into numerical difficulty in the second test on the image France with severe motion blurring. As it stagnated before meeting its convergence tolerance, LD generated an output with an SNR lower than that of FTVd, and a higher relative error compared to that of the ForWaRD output. In the third and fourth tests on the image Man with two levels of Gaussian blurring, ForWaRD and FTVd performed equally well and were only slightly better than LD. However, the bar charts do not tell some visual differences between their restorations.

The input and restored images are given in Figures 4 (France) and 5 (Man), where the first rows contain the input images (with the more severely blurred on the right), the second and third rows contains corresponding restored images by ForWaRD and FTVd, respectively. The restored images by LD are not included in the figures as they are visually similar to those by FTVd.

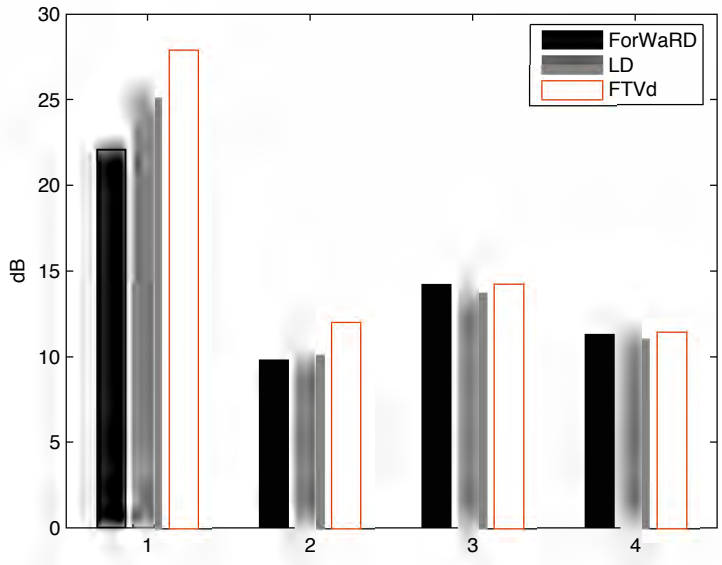


Figure 2: SNRs (higher is better) of restored images by ForWaRD, LD and FTVD: from left to right, bar groups $1, 2, 3$ and 4 correspond to the tests on the input images with medium motion, severe motion, medium Gaussian, and severe Gaussian blurrings, respectively.

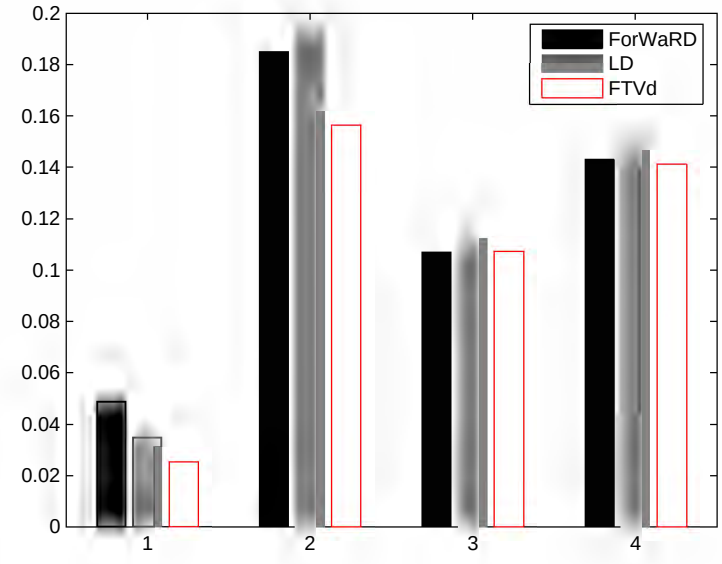


Figure 3: Relative errors (lower is better) of restored images by ForWaRD, LD and FTVD: from left to right, bar groups $1, 2, 3$ and 4 correspond to the tests on the input images with medium motion, severe motion, medium Gaussian, and severe Gaussian blurrings, respectively.

By carefully examining the images in Figure 4, we can visualize some important differences between the restored images by ForWaRD (in the middle row) and those by FTVd (in the bottom row). Generally, the artifacts of ghost texts or rings can be found in the outputs of ForWaRD. This is especially evident in the image on the right of the second row in which a ghost copy of the word "France" is clearly visible on the background. In that image, the upper half of the background also significantly loses smoothness. Contrary to ForWaRD's loss of recovery quality due to increased motion, little difference can be found in the two output images by FTVd in the third row. These test indicate that FTVd is likely more robust in correcting the motion blurring than ForWaRD.

In the two Gaussian deblurring tests presented in Figure 5, both methods performed well but differences can still be found. The results of ForWaRD have reduced contrast compared to the blurry inputs while those of FTVd do not lose contrast at all. Comparing the two recoveries of Man from severe gaussian blurring (the second and third images on the right column), the one from ForWaRD contains multiple rings (or ripples) parallel to the object edges in the image, while that from FTVd looks shaper but more blocky. By purely eyeballing the restored images, our preferences are given to the results of FTVd.

Tables 2 and 3 give computer running times on the four tests. As is already mentioned, FTVd was tested on both the Windows and Linux platforms since we were unable to make either one of the third-party codes LD and ForWaRD to work cross-platform. From the two tables, we see that ForWaRD ran about twice as fast as FTVd under Linux. On the other hand, FTVd was orders of magnitude faster than LD under Windows, with the performance gap widening as the image size increased.

Table 2: Running Time in Seconds of FTVd and ForWaRD on Windows Platform

Code	Test 1	Test 2	Test 3	Test 4	Language
FTVd	27.5	43.3	165.1	184.2	MATLAB
ForWaRD	17.5	17.6	89.2	91.1	C
Ratio	1.57	2.46	1.85	2.02	Mean 1.97

Table 3: Running Time in Seconds of LD and FTVd on Linux Platform

Code	Test 1	Test 2	Test 3	Test 4	Language
LD	167.6	1305.2	9304.9	43972.4	C++
FTVd	11.7	19.4	85.9	93.7	MATLAB
Ratio	14.3	67.3	108.3	469.3	Mean 164.8

We can safely draw a few definitive conclusions from our numerical comparison on both quality and speed. FTVd clearly wins over LD for better quality and faster speed. Since LD compares favorably to many other existing algorithms for solving model (1), the lead of FTVd obviously extends over to those algorithms. Compared to the Fourier/wavelet-based algorithm ForWaRD, FTVd appears superior in solution quality for the tested cases and, in the meantime, is not far behind in speed, especially after taking into consideration the efficiency differences between MATLAB and C/C++ languages.

3.5 More on FTVd

As is mentioned, the per-iteration computation of FTVd algorithm is dominated by three fast Fourier transforms (or Fourier-related transforms), each at the cost of $O(N^2 \log(N))$. The question is how many iterations are needed in general for FTVd to attain a required accuracy and how they vary with the size of input image.

For tests 1 through 4, the total inner iteration numbers taken by our simple version of FTVd (with the given "default" parameter values) are, respectively, 41, 67, 58 and 65. These total inner iteration numbers are very reasonable, given that there were 19 outer-iterations corresponding to the given β-sequence $\{2^2, 2^3, \cdots, 2^{20}\}$. On average, the inner iteration number needed for a single β value falls approximately in the range of 2 to 3.5. Given that three FFTs are required by each inner iteration, the total numbers of FFTs taken range from 120 to 200 for the four test cases.

We note that the two cases of "severe blurring", tests 2 and 4, required more iterations than those required by the corresponding two cases of "medium blurring", and the gap is more pronounced from test 1 to test 2 where motion blurrings were used. This observation indicates that more severe blurrings, corresponding to larger blurring kernels in convolution, may require more FTVd iterations, which should perhaps be expected. To further investigate how the number of iterations is affected by the size of input, we also tested FTVd on images of different sizes while fixing blurring kernels. In these tests, the number of iterations stayed at a constant level as the image size increased from 128×128 to 1024×1024.

Naturally, the number of iterations varies with parameter values of the algorithm. We have experimented with different values and observed that, within the given basic framework, there is generally a tradeoff between number of iterations and solution accuracy. Our "default" parameter values seem to achieve a reasonable balance.

Upon examining detailed timing information for FTVd, we see that over two thirds of computing time was spent on solving the u-subproblem, i.e., on FFTs and related calculations in (17), while the rest one third was shared by solving the $\mathbf{w}$-subproblem, finite-defferencing on u, checking stopping

criterion for inner iterations, and so on. In particular, FFTs alone consumed about 63% of total computational effort in FTVd.

Our experiments indicate that FTVd is capable of producing highly accurate solutions, relative to many other iterative algorithms, in a moderate number of iterations. FTVd appears to be numerically stable and insensitive to badly scaled data because, for one thing, it does not require, explicitly or implicitly, any (N^2 by N^2) matrix operations like some algorithms do such as lagged diffusivity. Our experiments have shown that even as $\beta_{\max}$ continues to increases beyond 2^{20}, the relative error continues to decrease with no sign of numerical instability, until machine precision is almost reached.

4 Conclusions

We have demonstrated that FTVd is a simple yet computationally efficient algorithmic framework for solving the TV deblurring model (1). The algorithm is based on replacing ∇u in the total variation term by a new variable $\mathbf{w}$ and then penalizing the discrepancy between the two, as measured by L_2-norm square. The resulting functional can be easily minimized with respect to either $\mathbf{w}$ or u while fixing the other, leading to our alternating minimization procedure. The convergence of this procedure readily follows from existing results.

In our tests, FTVd produced restored images with either similar or better quality in comparison to that of two state-of-the-art algorithms. To the best of our knowledge, the simple version of FTVd implemented in this paper already appears to be the fastest algorithm developed so far for solving the deblurring model (1) with total variation regularization, as it requires a relatively small number of iterations at the cost of roughly three FFTs per iteration. In addition, the algorithm can be easily extended to cases of total variation of higher-order differential operators, as well as to other transforms such as discrete cosine transform. Given its simplicity, speed, numerical stability and extendibility, we believe that FTVd is a promising algorithmic framework for numerical de-convolution with various total variation regularizations. Of course, further theoretical and computational studies are certainly needed for us to better understand the convergence properties of the FTVd framework and to develop more efficient path-following strategies for its implementation.

Acknowledgements

We are grateful to Paul Rodríguez and Brendt Wohlberg at the Los Alamos National Lab for valuable discussions and making their algorithm library NUMIPAD [17] available to us. The work of Y. Wang and W. Yin has been supported by the latter author's internal faculty research grant

from the Dean of Engineering at Rice University. The work of Y. Zhang has been supported in part by NSF Grant DMS-0405831.

References

[1] M. Avriel. *Nonlinear Programming: Analysis and Methods.* Printice-Hall, Inc, Englewood Cliffs, New Jersey, 1976.

[2] A. S. Carasso. Linear and nonlinear image deblurring: A documented study. *SIAM J. Numer. Anal.*, 36(6):1659–1689, 1999.

[3] A. Chambolle and P. L. Lions. Image recovery via total variation minimization and related problems. *Numer. Math.*, 76(2):167–188, 1997.

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

[5] T. F. Chan and P. Mulet. On the convergence of the lagged diffusivity fixed point method in total variation image restoration. *SIAM J. Numer. Anal.*, 36(2):354–367, 1999.

[6] T. F. Chan and J. Shen. Theory and computation of variational image deblurring. *IMS Lecture Notes*, 2006.

[7] T. F. Chan, J. Shen, and L. Vese. Variational PDE models in image processing. *Notice of Amer. Math. Soc.*, 50:14-26, Jan. 2003.

[8] G. Demoment. Image reconstruction and restoration: Overview of common estimation structures and problems. *IEEE Trans. Acoust., Speech, Signal Process.*, 37(12):2024–2036, 1989.

[9] D. L. Donoho. Nonlinear solution of linear inverse problems by wavelet-vaguelette decompostion. *Appl. Comput. Harmon. Anal.*, 2:101–126, 1995.

[10] D. Goldforb and W. Yin. Second-order cone programming methods for total variation-based image restoration. *SIAM J. Sci. Comput.*, 27(2):622–645, 2005.

[11] A. K. Katsaggelos. *Digital Image Restoration.* Springer-Verlag, 1991.

[12] Mosek ApS Inc. The Mosek optimization tools, version 4, 2006.

[13] R. Neelamani, H. Choi, and R. G. Baraniuk. Wavelet-based deconvolution for ill-conditioned systems. In *Proc. IEEE ICASSP*, volume 6, pages 3241–3244, Mar. 1999.

[14] R. Neelamani, H. Choi, and R. G. Baraniuk. ForWaRD: Fourier-wavelet regularized deconvolution for ill-conditioned systems. *IEEE Trans. signal proc.*, 52:418–433, Feb. 2004.

[15] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Inc, New Jersey, 1989.

[16] K. R. Rao and P. Yip. *Discrete Cosine Transform : Algorithms, Advantages, and Applications*. Academic Press, Boston, 1990.

[17] P. Rodríguez and B. Wohlberg. Numerical methods for inverse problems and adaptive decomposition (NUMIPAD). `http://numipad.sourseforge.net/`.

[18] P. Rodríguez and B. Wohlberg. An iteratively weighted norm algorithm for total variation regularization. *To appear in IEEE Signal Processing Letters*, 2007.

[19] L. Rudin and S. Osher. Total variation based image restoration with free local constraints. *Proc. 1st IEEE ICIP*, 1:31–35, 1994.

[20] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. D*, 60:259–268, 1992.

[21] D. Strong and T. F. Chan. Edge-preserving and scale-dependent properties of total variation regularization. *Inverse Problems*, 19:165–187, 2003.

[22] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, June 2001.

[23] C. Van Loan. *Computational Frameworks for the Fast Fourier Transform*, volume 10 of *Frontiers in Applied Mathematics*. SIAM, 1992.

[24] C. R. Vogel and M. E. Oman. Iterative methods for total variation denoising. *SIAM J. Sci. Comput.*, 17(1):227–238, 1996.

[25] W. P. Ziemer. *Weakly Differentiable Functions: Sobolev Spaces and Functions of Bounded Variation*. Graduate Texts in Mathematics. Springer, 1989.

Figure 4: The first row contains images with medium (left) and severe (right) motion blurrings. The second row contains corresponding restored images by ForWaRD, and the third row by FTVd.

Figure 5: The first row contains images with medium (left) and severe (right) Gaussian blurrings. The second row contains corresponding restored images by ForWaRD, and the third row by FTVd.