FFT BASED SOLUTION FOR MULTIVARIABLE L_2 EQUATIONS USING KKT SYSTEM VIA FFT AND EFFICIENT PIXEL-WISE INVERSE CALCULATION

Keiichiro Shirai
Shinshu Univ.

Masahiro Okuda
Univ. of Kitakyushu

ABSTRACT

When solving l_2 optimization problems based on linear filtering with some regularization in signal/image processing such as Wiener filtering, the fast Fourier transform (FFT) is often available to reduce its computational complexity. Most of the problems, in which the FFT is used to obtain their solutions, are based on single variable equations. On the other hand, the Karush-Kuhn-Tucker (KKT) system, which is often used for solving constrained optimization problems, generally results in multivariable equations.

In this paper, we propose a FFT based computational method for multivariable l_2 equations. Our method applies a FFT to each block of the KKT system, and represents the equation as an image-wise simultaneous equation consisting of Fourier transformed filters and images. In our method, an inverse matrix calculation that consists of *complex* pixel values gathered from each transformed image is required for each pixel. We exploit the homogeneity of neighboring values and solve them efficiently.

Index Terms— Optimization, regularization, inverse problem, KKT system, fast Fourier transform

1. INTRODUCTION

In the calculation of filtering used in image processing, mainly three types of forms are used: (i) matrix form $\mathbf{A}\mathbf{x}=\mathbf{b}$ denoted by the matrix multiplication of a filter matrix $\mathbf{A}$ and vectorized images $\mathbf{x}$ and $\mathbf{b}$; (ii) filter form $A \otimes X = B$ denoted by the convolution of a filter kernel A and images X and B, and (iii) a form using the fast *Fourier transform* (FFT) $\mathscr{F}(A) \circ \mathscr{F}(X) = \mathscr{F}(B)$ denoted by Hadamard product (pixel-wise multiplication) of the FFTed filter $\mathscr{F}(A)$ and FFTed images $\mathscr{F}(X)$ and $\mathscr{F}(B)$. In particular, they are completely compatible if the matrix $\mathbf{A}$ has a structure called *block circulant with circulant blocks* (BCCB). In short, *convolution* with circular boundary padding is defined as

$$\mathbf{A}\mathbf{x}=\mathbf{b} \;\leftrightarrow\; A\otimes X=B \;\leftrightarrow\; \mathscr{F}(A)\circ\mathscr{F}(X)=\mathscr{F}(B). \quad (1)$$

The calculation efficiency increases from left to right especially for long filters since they utilize the feature of matrix structure and can be regarded as specialized (pre-conditioned and diagonalized) ones of the matrix form.

The well known filter formula using the correspondences is the Weiner filter [1], and its calculation is generally performed in the frequency domain. It is, however, difficult to apply the filter form directly to ill-posed problems such as deblurring because the matrix $\mathbf{A}$ often becomes *singular* and it causes zero division $1/\mathscr{F}(A)$ in the frequency domain, which results in artifacts on resulting images. On the other hand, the development of regularization using image features and the convex optimization algorithms [2, 3] changes the situation because the well regularized matrix $\mathbf{A}$ becomes *non-singular* and its FFT form does not generate the artifacts. Thus, the FFT form is used in many methods, *e.g.*, deblurring [4] and smoothing [5].

The aforementioned methods [1, 4, 5] handle l_2 equations with a single variable. When the equations with two or more variables such as Lagrangian multipliers used in the above optimization algorithm [3], the matrix form is usually employed, and the multiple variables are concatenated into the single variable using a direct product expression that is called *Karush-Kuhn-Tucker* (KKT) system [6, 7]. In this case, the matrix $\mathbf{A}$ does not hold the BCCB structure, and thus the FFT form is not directly applicable to the calculation. In the optimization algorithms, usually we have to solve this l_2 equation iteratively. Thus applying the FFT will significantly reduce the total executing time.

In this paper, we propose an efficient calculation method for multivariable l_2 equations using FFT form. We utilize an assumption that the block components of a KKT system holds the BCCB structure, and represent its block-wise matrix multiplication using FFT form. As a result, a simultaneous equation using FFTed images is obtained. Solving this equation and then performing the inverse FFT, we can solve the KKT system efficiently. In our method, the inverse matrix calculation that consists of *complex* pixel values gathered from each transformed image is required for each pixel. To solve them efficiently, we utilize the homogeneity of neighboring values. Our approach is actually the same as that used in 1D signal source separation in the frequency domain [8, 9], however our method can handle more general l_2 equations, and the computational method is further considered.

2. SINGLE VARIABLE L_2 EQUATION IN FFT FORM

In this section we describe a typical example of an inverse calculation in FFT form using Tikhonov regularization [10].

First we show *correlation* that forms a counterpart to the *convolution* shown in (1):

$$\mathbf{A}^T\mathbf{x}=\mathbf{b} \;\leftrightarrow\; A^*\otimes X=B \;\leftrightarrow\; \overline{\mathscr{F}(A)}\circ\mathscr{F}(X)=\mathscr{F}(B), \quad (2)$$

where A^* is the 180° rotated mirror kernel of A[1], and $\overline{\mathscr{F}(\cdot)}$ is the conjugate of $\mathscr{F}(\cdot)$. Using this relationship, for example, a forward differential filter in the convolution $A := [1, -1, 0]$ gives the corresponding differential operator in the correlation $A^* := [0, -1, 1]$.

As a typical example of a single variable l_2 equation, we show the FFT form of a Tikhonov regularized equation [10]:

$$\arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{\Gamma}\mathbf{x}\|_2^2, \quad (3)$$

where $\mathbf{x}\in R^N$ is a vectorized image of N pixels, $\mathbf{A}$ and $\mathbf{\Gamma}\in R^{N\times N}$ are filter matrices[2], and $\|\mathbf{x}\|_p := (\sum_i |x_i|^p)^{1/p}$ denotes l_p norm. The

[1] This notation is corresponding to correlation $A^*\otimes X = A * X$, where *convolution* is defined as $(f\otimes g)(x)=\sum_u f(u)g(x-u)$ and *correlation* is defined as $(f*g)(x)=\sum_u f(u)g(x+u)$.

[2] In a case of deblurring, Gaussian filter matrix and Laplacian filter matrix are typically used for $\mathbf{A}$ and $\mathbf{\Gamma}$ respectively.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

second term is a regularizer that indicates a constraint $\|\mathbf{\Gamma x}\|_2^2 \leq \epsilon$ and also guarantees the solution of the inverse calculation. The solution in matrix form is obtained by setting the first-order derivative w.r.t. $\mathbf{x}$ to zero:

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda\mathbf{\Gamma}^T\mathbf{\Gamma})^{-1}\mathbf{A}^T\mathbf{b}. \tag{4}$$

The corresponding FFT form is written as

$$\begin{aligned}\Big(\overline{\mathscr{F}(A)}\circ\mathscr{F}(A) + \lambda\overline{\mathscr{F}(\Gamma)}\circ\mathscr{F}(\Gamma)\Big)\circ\mathscr{F}(X) = \overline{\mathscr{F}(A)}\circ\mathscr{F}(B),\\ X = \mathscr{F}^{-1}\left(\frac{\overline{\mathscr{F}(A)}\circ\mathscr{F}(B)}{\overline{\mathscr{F}(A)}\circ\mathscr{F}(A) + \lambda\overline{\mathscr{F}(\Gamma)}\circ\mathscr{F}(\Gamma)}\right),\end{aligned} \tag{5}$$

where $\mathscr{F}^{-1}(\cdot)$ is the inverse FFT. The shape of (5) is of the same form as the Wiener filter [1].

3. MULTIVARIABLE L_2 EQUATION IN FFT FORM

We describe our method in this section. First we show an example of a KKT system [6, 7], and then describe its FFT form and our pixel-wise inverse calculation [11] that uses propagation based matrix inversion.

3.1. Representation in Karush-Kuhn-Tucker system

As an example of a multivariable l_2 equation, we consider the following problem with a constraint:

$$\arg\min_{\mathbf{x},\mathbf{y}} \tfrac{1}{2}\|\mathbf{Ax} - \mathbf{y}\|_2^2 \quad \text{s.t. } \mathbf{\Gamma x} = \mathbf{b}, \tag{6}$$

where $\mathbf{y}$ and $\mathbf{b} \in R^N$ are vectorized images. Note that this equation has two variables $\mathbf{x}$ and $\mathbf{y}$. To solve this problem, typically the method of *Lagrange multipliers* is employed:

$$\arg\min_{\mathbf{x},\mathbf{y},\mathbf{z}} \Big(\mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{z}) = \tfrac{1}{2}\|\mathbf{Ax} - \mathbf{y}\|_2^2 + \mathbf{z}^T(\mathbf{\Gamma x} - \mathbf{b})\Big), \tag{7}$$

where the dual variable $\mathbf{z} \in R^N$ called Lagrange multiplier is newly added and three variables are used in total. To solve this equation, the KKT system is available. First the derivatives of (7) w.r.t. $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are set to zero:

$$\begin{aligned}\partial\mathcal{L}/\partial\mathbf{x} &\rightarrow \mathbf{A}^T(\mathbf{Ax} - \mathbf{y}) + \mathbf{\Gamma}^T\mathbf{z} = \mathbf{0},\\ \partial\mathcal{L}/\partial\mathbf{y} &\rightarrow -(\mathbf{Ax} - \mathbf{y}) = \mathbf{0},\\ \partial\mathcal{L}/\partial\mathbf{z} &\rightarrow \mathbf{\Gamma x} - \mathbf{b} = \mathbf{0}.\end{aligned} \tag{8}$$

Then a direct product is expressible by concatenated variables $\mathbf{r} = [\mathbf{x}^T, \mathbf{y}^T, \mathbf{z}^T]^T$, and (8) is combined to yield the KKT system:

$$\begin{pmatrix}\mathbf{A}^T\mathbf{A} & -\mathbf{A}^T & \mathbf{\Gamma}^T\\ -\mathbf{A} & \mathbf{I}_N & \mathbf{0}\\ \mathbf{\Gamma} & \mathbf{0} & \mathbf{0}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\\ \mathbf{z}\end{pmatrix} = \begin{pmatrix}\mathbf{A}^T\mathbf{y}\\ \mathbf{0}\\ \mathbf{b}\end{pmatrix}, \tag{9}$$

where $\mathbf{I}_N \in R^{N\times N}$ is an identity matrix. This equation can be regarded as $\mathbf{Pr} = \mathbf{q}$ with a symmetric structure $\mathbf{P}^T = \mathbf{P}$, and solved by $\mathbf{r} = \mathbf{P}^{-1}\mathbf{q}$ using basic solvers such as the conjugate gradient [12]. Note that since $\mathbf{P}$ no longer holds the BCCB structure, FFT based methods are not directly available.

3.2. Simultaneous equation in the Frequency domain

We consider solving (9) in FFT form and generalize (9) to M-variable equations as

$$\begin{pmatrix}\mathbf{P}_{11} & \mathbf{P}_{12} & \cdots & \mathbf{P}_{1M}\\ \mathbf{P}_{21} & \mathbf{P}_{22} & \cdots & \mathbf{P}_{2M}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{P}_{M1} & \mathbf{P}_{M2} & \cdots & \mathbf{P}_{MM}\end{pmatrix}\begin{pmatrix}\mathbf{r}_1\\ \mathbf{r}_2\\ \vdots\\ \mathbf{r}_M\end{pmatrix} = \begin{pmatrix}\mathbf{q}_1\\ \mathbf{q}_2\\ \vdots\\ \mathbf{q}_M\end{pmatrix}. \tag{10}$$

The block elements $\mathbf{P}_{ij} \in R^{N\times N}$ indicate BCCB filter matrices, and the block elements $\mathbf{q}_i$ and $\mathbf{r}_i \in R^N$ indicate the vectorized images. Thus FFTs are applicable to each block and their multiplications are expressible using Hadamard product of Fourier coefficient images with *complex* pixel values:

$$\begin{bmatrix}\mathscr{F}(P_{11}) & \mathscr{F}(P_{12}) & \cdots & \mathscr{F}(P_{1M})\\ \mathscr{F}(P_{21}) & \mathscr{F}(P_{22}) & \cdots & \mathscr{F}(P_{2M})\\ \vdots & \vdots & \ddots & \vdots\\ \mathscr{F}(P_{M1}) & \mathscr{F}(P_{M2}) & \cdots & \mathscr{F}(P_{MM})\end{bmatrix} \star \begin{bmatrix}\mathscr{F}(R_1)\\ \mathscr{F}(R_2)\\ \vdots\\ \mathscr{F}(R_M)\end{bmatrix} = \begin{bmatrix}\mathscr{F}(Q_1)\\ \mathscr{F}(Q_2)\\ \vdots\\ \mathscr{F}(Q_M)\end{bmatrix}, \tag{11}$$

where $[A, B] \star [C, D]^T = A \circ C + B \circ D$ indicates the pixel-wise calculation $a_k c_k + b_k d_k$ at each pixel k. Now we reformulate the matrix form (11) to *pixel-wise equations*. The k-th pixel values in the FFTed images of the (i, j)-th block $\widetilde{p}_{ij,k} = \mathscr{F}(P_{ij})_k$; and i-th vectors $\widetilde{q}_{i,k} = \mathscr{F}(Q_i)_k$ and $\widetilde{r}_{i,k} = \mathscr{F}(R_i)_k$ are gathered to construct a pixel-wise equation:

$$\begin{aligned}\begin{pmatrix}\widetilde{p}_{11,k} & \widetilde{p}_{12,k} & \cdots & \widetilde{p}_{1M,k}\\ \widetilde{p}_{21,k} & \widetilde{p}_{22,k} & \cdots & \widetilde{p}_{2M,k}\\ \vdots & \vdots & \ddots & \vdots\\ \widetilde{p}_{M1,k} & \widetilde{p}_{M2,k} & \cdots & \widetilde{p}_{MM,k}\end{pmatrix}\begin{pmatrix}\widetilde{r}_{1,k}\\ \widetilde{r}_{2,k}\\ \vdots\\ \widetilde{r}_{M,k}\end{pmatrix} = \begin{pmatrix}\widetilde{q}_{1,k}\\ \widetilde{q}_{2,k}\\ \vdots\\ \widetilde{q}_{M,k}\end{pmatrix},\\ \widetilde{\mathbf{P}}_k\widetilde{\mathbf{r}}_k = \widetilde{\mathbf{q}}_k.\end{aligned} \tag{12}$$

When $M = 2$, the solution is obtained using a standard adjugate method $\widetilde{\mathbf{P}}_k^{-1} = \text{adj}(\widetilde{\mathbf{P}}_k)/\det(\widetilde{\mathbf{P}}_k)$, and the corresponding notation in image-wise calculation as shown in (5) is

$$\begin{bmatrix}\mathscr{F}(R_1)\\ \mathscr{F}(R_2)\end{bmatrix} = \frac{1}{\det}\begin{bmatrix}\mathscr{F}(P_{22}) & -\mathscr{F}(P_{12})\\ -\mathscr{F}(P_{21}) & \mathscr{F}(P_{11})\end{bmatrix} \star \begin{bmatrix}\mathscr{F}(Q_1)\\ \mathscr{F}(Q_2)\end{bmatrix}, \tag{13}$$

where $\det = \mathscr{F}(P_{11}) \circ \mathscr{F}(P_{22}) - \mathscr{F}(P_{12}) \circ \mathscr{F}(P_{21})$. Then the final solution is obtained from the inverse FFT:

$$R_1 = \mathscr{F}^{-1}(\mathscr{F}(R_1)), \quad R_2 = \mathscr{F}^{-1}(\mathscr{F}(R_2)). \tag{14}$$

When $M \geq 4$, more efficient solutions such as a method using Cholesky factorization is required (particularly when handling complex numbers) because the computational complexity of the adjugate method that includes recursive cofactor expansions is $O(S(M)) = O(M^3M!)$ where $S(\cdot)$ varies depending on the solution, while that of general solvers are at most $O(M^3)$.

3.2.1. Computational efficiency by symmetric block structures and symmetry of FFTed images

When the block structure is symmetric: $\{\mathbf{P}_{ij} = \mathbf{P}_{ji}^T \mid i < j\}$, the FFTs denoted in (11) are required only at the upper (or lower) triangular part since the opposite triangular part is the conjugate of them: $\{\mathscr{F}(P_{ij}) = \overline{\mathscr{F}(P_{ji})} \mid i < j\}$. The local matrices $\widetilde{\mathbf{P}}_k$ also hold that property and become *Hermitian* matrices: $\widetilde{\mathbf{P}}_k = \widetilde{\mathbf{P}}_k^H$ where H denotes the conjugate transpose. Thus we can use solvers specialized for Hermitian matrices instead of general solvers. In this case, the complexities of the FFTs become $O(\frac{M(M+1)}{2} \cdot N\log N)$.

As for the FFTed BCCB filter matrices, a set of two pixels k and k' located at the symmetric quadrants in the frequency domain has conjugate values, which results in $\widetilde{\mathbf{P}}_k = \widetilde{\mathbf{P}}_{k'}^H$. In addition, the inverse of Hermitian matrices are also given as Hermitian $\widetilde{\mathbf{P}}_k^{-1} = \widetilde{\mathbf{P}}_k^{-H}$. Thus only a half of quadrants requires the inverse computation, and their conjugate $\widetilde{\mathbf{P}}_k^{-1} = \widetilde{\mathbf{P}}_{k'}^{-H}$ are available to the other quadrants. The total complexity is estimated as $O(\frac{M(M+1)}{2} N\log N \cdot \frac{N}{2} S(M))$.

3.2.2. Preliminarily explicit computation of inverse matrices

When calculating the equation $\widetilde{\mathbf{P}}_k\widetilde{\mathbf{r}}_k=\widetilde{\mathbf{q}}_k$ just one time, it is better not to compute the inverse matrix $\widetilde{\mathbf{P}}_k^{-1}$ explicitly. However, when using iterative methods like the ADMM (described in the latter section), we need to solve Eq. (12) iteratively by $\widetilde{\mathbf{r}}_k^{(t)}=\widetilde{\mathbf{P}}_k^{-1}\widetilde{\mathbf{q}}_k^{(t)}$ at each iteration t. In this case, precomputing the inverse matrix $\widetilde{\mathbf{P}}_k^{-1}$ is efficient because $\widetilde{\mathbf{P}}_k$ is generally constant in the iterations.

3.3. Propagation based pixel-wise inverse calculation

To compute the pixel-wise equations (12), we employ our computational approach described in [11]. In short, to obtain $\widetilde{\mathbf{P}}_k^{-1}$ at the current pixel, we use the matrix at the preceding pixel: $\widetilde{\mathbf{P}}_{k-1}^{-1}$. Our previous approach uses singular value decomposition (SVD), while we employ the iterative algorithm [13, 14] for the matrix inversion in this framework.

The purpose is to compute $\widetilde{\mathbf{r}}_k=\widetilde{\mathbf{P}}_k^{-1}\widetilde{\mathbf{q}}_k$ at each pixel k by exploiting the feature of the Hermitian matrix $\widetilde{\mathbf{P}}_k\in R^{M\times M}$ and its homogeneity among neighboring pixels (*e.g.*, $k-1$):

$$\begin{cases}\widetilde{\mathbf{P}}_k^H=\widetilde{\mathbf{P}}_k,\\ \widetilde{\mathbf{P}}_k\approx\widetilde{\mathbf{P}}_{k-1}\rightarrow\widetilde{\mathbf{P}}_k^{-1}\approx\widetilde{\mathbf{P}}_{k-1}^{-1},\end{cases}\tag{15}$$

For utilizing these properties, we employ an iterative algorithm for Hermitian matrices [13, 14] that uses an initial inverse matrix $\mathbf{\Lambda}_k$ and improves the accuracy of it by

$$\mathbf{\Lambda}_k^{(t+1)}:=\mathbf{\Lambda}_k^{(t)}+\mathbf{\Lambda}_k^{(t)}(\mathbf{I}_M-\widetilde{\mathbf{P}}_k\mathbf{\Lambda}_k^{(t)}).\tag{16}$$

If the initial inverse matrix is a good approximation, that is $\widetilde{\mathbf{P}}_k^{-1}\approx\mathbf{\Lambda}_k^{(0)}$, this iteration will converge in a few iterations. Thus we apply the already computed matrix at the preceding pixel $\mathbf{\Lambda}_{k-1}$ to the initial matrix at the current pixel $\mathbf{\Lambda}_k$:

$$\mathbf{\Lambda}_k^{(0)}=\mathbf{\Lambda}_{k-1}^{(\text{converged})}=\widetilde{\mathbf{P}}_{k-1}^{-1}.\tag{17}$$

In our implementation, we compute the first inverse matrix $\mathbf{\Lambda}_{k=0}$ at the boundary of an image using Cholesky factorization. The algorithm needs only one or two iterations to converge in practice, thanks to the strong homogeneity of the spectrum of the FFTed filters $\{\mathscr{F}(P_{ij})\}$[3].

4. APPLICATION FOR TGV SMOOTHING

As an application of our proposal, here we describe a smoothing method using the *total generalized variation* (TGV) regularization of the second order [15, 16] and the *alternating direction method of multipliers* (ADMM) [3]. The purpose is to solve the bottleneck part of the multivariable l_2 equation in the ADMM.

The formula of smoothing with the TGV regularizer (we use the matrix form of [16, 17]) is defined as

$$\arg\min_{\mathbf{x}}\ \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2+\|\mathbf{x}\|_{\text{TGV}}^{\alpha_1,\alpha_2}.\tag{18}$$

The TGV term is defined as

$$\|\mathbf{x}\|_{\text{TGV}}^{\alpha_1,\alpha_2}:=\min_{\mathbf{Dx}=\mathbf{s}+\mathbf{t}}\left(\alpha_1\|\mathbf{s}\|_{1,2}^{(2,N)}+\alpha_2\|\mathbf{Gt}\|_{1,2}^{(3,N)}\right)\tag{19}$$

[3]Obtained solutions using $\mathbf{\Lambda}^{(\text{converged})}$ are prone to generate small imaginary values after the inverse FFT due to computational error. We only use the real part in this paper.

by introducing auxiliary variables $\{\mathbf{s},\mathbf{t}\in R^2|\ \mathbf{Dx}=\mathbf{s}+\mathbf{t}\}$ that indicate differentiated values, and using differential filter matrices for the first-order and second-order derivatives:

$$\mathbf{D}:=\begin{pmatrix}\mathbf{D}_h\\ \mathbf{D}_v\end{pmatrix},\quad \mathbf{G}^T:=\begin{pmatrix}\mathbf{D}_h & \mathbf{D}_v & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_h & \mathbf{D}_v\end{pmatrix},\tag{20}$$

where $\mathbf{D}_h$ and $\mathbf{D}_v\in R^{N\times N}$ are differential filter matrices having the BCCB structure for horizontal and vertical directions, $\|\mathbf{s}\|_{1,2}^{(M,N)}=\sum_{k=1}^{N}\sqrt{\sum_{m=0}^{M-1}s_{k+mN}^2}$ is a mixed norm.

The ADMM form of (18) is defined as

$$\begin{aligned}\arg\min_{\mathbf{x},\mathbf{s}}\quad & \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2+\alpha_1\|\mathbf{z}_1\|_{1,2}^{(2,N)}+\alpha_2\|\mathbf{z}_2\|_{1,2}^{(3,N)}\\ \text{s.t.}\quad & \mathbf{z}_1=\mathbf{Dx}-\mathbf{t},\quad \mathbf{z}_2=\mathbf{Gt},\end{aligned}\tag{21}$$

with the use of auxiliary variables $\mathbf{z}_1\in R^{2N}$ and $\mathbf{z}_2\in R^{3N}$. The *augmented Lagrangian* used in ADMM is written as

$$\begin{aligned}&\mathcal{L}(\mathbf{x},\mathbf{t},\mathbf{z}_1,\mathbf{z}_2,\mathbf{u}_1,\mathbf{u}_2)=\tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2\\ &+\alpha_1\|\mathbf{z}_1\|_{1,2}^{(2,N)}+(\rho/2)\|\mathbf{Dx}-\mathbf{t}-\mathbf{z}_1+\mathbf{u}_1\|_2^2\\ &+\alpha_2\|\mathbf{z}_2\|_{1,2}^{(3,N)}+(\eta/2)\|\mathbf{Gt}-\mathbf{z}_2+\mathbf{u}_2\|_2^2,\end{aligned}\tag{22}$$

where newly introduced variables $\mathbf{u}_1\in R^{2N}$ and $\mathbf{u}_2\in R^{3N}$ are called the scaled dual variable. Then the iterative optimization steps of ADMM are defined as

$$\begin{aligned}\{\mathbf{x}^{(t+1)},\mathbf{t}^{(t+1)}\}&:=\arg\min_{\mathbf{x},\mathbf{t}}\mathcal{L}(\mathbf{x},\mathbf{t},\mathbf{z}_1^{(t)},\mathbf{z}_2^{(t)},\mathbf{u}_1^{(t)},\mathbf{u}_2^{(t)})\\ \mathbf{z}_1^{(t+1)}&:=\arg\min_{\mathbf{z}_1}\mathcal{L}(\mathbf{x}^{(t+1)},\mathbf{t}^{(t+1)},\mathbf{z}_1,\mathbf{u}_1^{(t)})\\ \mathbf{z}_2^{(t+1)}&:=\arg\min_{\mathbf{z}_2}\mathcal{L}(\mathbf{x}^{(t+1)},\mathbf{t}^{(t+1)},\mathbf{z}_2,\mathbf{u}_2^{(t)})\\ \mathbf{u}_1^{(t+1)}&:=\mathbf{u}_1^{(t)}+(\mathbf{Dx}^{(t+1)}-\mathbf{t}^{(t+1)}-\mathbf{z}_1^{(t+1)})\\ \mathbf{u}_2^{(t+1)}&:=\mathbf{u}_2^{(t)}+(\mathbf{Gt}^{(t+1)}-\mathbf{z}_2^{(t+1)}).\end{aligned}\tag{23}$$

The minimizations of $\mathbf{z}_1$ and $\mathbf{z}_2$ are performed by *shrinkage* for the mixed norm $\|\cdot\|_{1,2}^{(M,N)}$ and the complexities are quite low:

$$\begin{aligned}\mathbf{z}_1^{(t+1)}&:=\mathcal{S}_{\alpha_1/\rho}(\mathbf{Dx}^{(t+1)}-\mathbf{t}^{(t+1)}+\mathbf{u}_1^{(t)}),\\ \mathbf{z}_2^{(t+1)}&:=\mathcal{S}_{\alpha_2/\eta}(\mathbf{Gt}^{(t+1)}+\mathbf{u}_2^{(t)}),\end{aligned}\tag{24}$$

where the k'th element of the shrinkage function is given by

$$\mathcal{S}_\gamma(\mathbf{x})_k=x_k\max\Big\{1-\gamma\Big(\sum_{m=0}^{M-1}x_{k+mN}^2\Big)^{-1/2},\ 0\Big\}.\tag{25}$$

On the other hand, the calculation of the first step in (23) is given as a multivariable l_2 equation, and the calculation cost is much larger than other steps.

Differentiating (22) w.r.t. $\mathbf{x}$ and $\mathbf{t}$, and setting the derivatives to zeros, we obtain the KKT system as

$$\begin{pmatrix}\mathbf{I}_N+\rho\mathbf{D}^T\mathbf{D} & -\rho\mathbf{D}^T\\ -\rho\mathbf{D}^T & \rho\mathbf{I}_{2N}+\eta\mathbf{G}^T\mathbf{G}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{t}\end{pmatrix}=\begin{pmatrix}\mathbf{y}+\rho\mathbf{D}^T(\mathbf{z}_1-\mathbf{u}_1)\\ -\rho(\mathbf{z}_1-\mathbf{u}_1)+\eta\mathbf{G}^T(\mathbf{z}_2-\mathbf{u}_2)\end{pmatrix}.\tag{26}$$

Furthermore, we separate these blocks such that they have the BCCB structures:

$$\mathbf{D}^T\mathbf{D}=\mathbf{D}_h^T\mathbf{D}_h+\mathbf{D}_v^T\mathbf{D}_v=\Delta,\quad \mathbf{G}^T\mathbf{G}=\begin{pmatrix}\Delta & \mathbf{D}_v\mathbf{D}_h^T\\ \mathbf{D}_h\mathbf{D}_v^T & \Delta\end{pmatrix},\tag{27}$$

where Δ indicates the Laplacian filter matrix. Similarly we separate vectorized images into the unit of image:

$$\begin{aligned}\mathbf{t}:=[\mathbf{t}_h^T,\mathbf{t}_v^T]^T,\quad \mathbf{z}_1:=[\mathbf{z}_{1h}^T,\mathbf{z}_{1v}^T]^T,\quad \mathbf{u}_1:=[\mathbf{u}_{1h}^T,\mathbf{u}_{1v}^T]^T,\\ \mathbf{z}_2:=[\mathbf{z}_{2h}^T,\mathbf{z}_{2d}^T,\mathbf{z}_{2v}^T]^T,\quad \mathbf{u}_2:=[\mathbf{u}_{2h}^T,\mathbf{u}_{2d}^T,\mathbf{u}_{2v}^T]^T,\end{aligned}\tag{28}$$

Table 1. A comparison of execution times of l_2 part in TGV smoothing (at each iteration in ADMM).

Method	Form	Inverse calc.	Formula	Algorithm	time (at 512^2pix)
(i)	Matrix	Implicit	$\mathbf{Pr}=\mathbf{q}$ in (29)	by a sparse solver of MATLAB: $\mathbf{r}=\mathbf{P}\backslash\mathbf{q}$	14.4 sec
(ii)	Matrix	Implicit	$\mathbf{Pr}=\mathbf{q}$ in (29)	by a CG method [12] of MATLAB: $\mathbf{r}=\mathrm{pcg}(\mathbf{P},\mathbf{q})$	2.2 sec
(iii)	FFT	Implicit	$\forall_k, \widetilde{\mathbf{P}}_k\widetilde{\mathbf{r}}_k=\widetilde{\mathbf{q}}_k$	Cholesky decomp. by LAPACK: zpotrs	324 msec
(iv)	FFT	Explicit	$\forall_k, \widetilde{\mathbf{q}}_k=\mathrm{adj}(\widetilde{\mathbf{P}}_k)/\det(\widetilde{\mathbf{P}}_k)\widetilde{\mathbf{q}}$	Adjugate method	251 msec
(v)	FFT	Explicit	$\forall_k, \widetilde{\mathbf{q}}_k=\widetilde{\mathbf{P}}_k^{-1}\widetilde{\mathbf{q}}_k$	Cholesky decomp. by LAPACK: zpotri	378 msec
(vi)	FFT	Explicit	$\forall_k, \widetilde{\mathbf{q}}_k=\mathbf{\Lambda}_k^{(\text{converged})}\widetilde{\mathbf{q}}_k$	Our method in Sec. 3.3 (1 teration)	177 msec
(vii)	FFT	Explicit	$\forall_k, \widetilde{\mathbf{q}}=\mathbf{\Lambda}_k^{(\text{pre-computed})}\widetilde{\mathbf{q}}_k$	Pre-comput. of $\{\mathbf{P}_k^{-1}\}$ in Sec. 3.2.2	98 msec

where subscripts h, d, and v indicates the horizontal, diagonal, and vertical parts respectively. As a result, we finally obtain the BCCB blocks in the form of (10) as

$$\mathbf{P}=\begin{pmatrix}\mathbf{I}_N+\rho\Delta & -\rho\mathbf{D}_h^T & -\rho\mathbf{D}_v^T\\ -\rho\mathbf{D}_h & \rho\mathbf{I}_N+\eta\Delta & \eta\mathbf{D}_v\mathbf{D}_h^T\\ -\rho\mathbf{D}_v & \eta\mathbf{D}_h\mathbf{D}_v^T & \rho\mathbf{I}_N+\eta\Delta\end{pmatrix},\quad \mathbf{r}=\begin{pmatrix}\mathbf{x}\\ \mathbf{t}_h\\ \mathbf{t}_v\end{pmatrix},$$
$$\mathbf{q}=\begin{pmatrix}\mathbf{y} & +\rho\mathbf{D}_h^T(\mathbf{z}_{1h}-\mathbf{u}_{1h})+\rho\mathbf{D}_v^T(\mathbf{z}_{1v}-\mathbf{u}_{1v})\\ -\rho(\mathbf{z}_{1h}-\mathbf{u}_{1h})+\eta\mathbf{D}_h(\mathbf{z}_{2h}-\mathbf{u}_{2h})+\eta\mathbf{D}_v(\mathbf{z}_{2d}-\mathbf{u}_{2d})\\ -\rho(\mathbf{z}_{1v}-\mathbf{u}_{1v})+\eta\mathbf{D}_h(\mathbf{z}_{2d}-\mathbf{u}_{2d})+\eta\mathbf{D}_v(\mathbf{z}_{2v}-\mathbf{u}_{2v})\end{pmatrix}. \quad (29)$$

The structure of $\mathbf{P}$ is symmetric and the diagonal has non-zero values since identity matrices are embedded. Thus $\mathbf{P}$ is *non-singular* and the inverse $\mathbf{P}^{-1}$ exists. The pixel-wise matrix $\widetilde{\mathbf{P}}_k$ shown in (12) is also given as a non-singular Hermitian matrix, and thus the pixel-wise solutions are guaranteed. Fig. 1 shows the spectrum images of FFTed BCCB filter blocks of $\mathbf{P}$ (left) and the results of inverse computation (right). One can see that those coefficient images have homogeneity, *i.e.*, the pixel values are changing gradually.

5. EXPERIMENTAL RESULTS

The results of TGV smoothing described in the previous section are shown here. As for images, the color intensities are normalized to the range $[0,1]$, and here we show the result of an image of 512×512 size. The complexity of the matrix form $O(S(N))$ drastically increases associated with the image size, while that of the FFT form is $O(\frac{M(M+1)N^2}{4}\log N\cdot S(M)), M\ll N$ (see Sec. 3.2 and 3.2.1). The main difference comes from the complexity of $S(\cdot)$.

Table 1 shows a comparison of execution times with principal methods provided as functions of MATLAB and LAPACK [19][4]. The execution times of the FFT form methods contain pixel-wise 3×3 inverse computation for all the pixels and execution time of FFTs. The "Implicit / Explicit" means whether the inverse matrix can be computed explicitly or not. The method (i) is the typical solution for optimization based methods by using linear algebra and sparse matrix expression. Then (ii) is the result of the conjugate gradient (CG) method [12], and the number of its inner iterations is set so as to give a similar PSNR with (i). On the other hand, (iii)-(vii) are the proposed solution using FFT form. Clearly the complexities of the FFT form are less than that of the matrix form. Although the propagation based method (vi) is slower than the adjugate method (iv), it will reverse when $4\le M$ because of their complexities: $O_{(\text{iv})}(M^3M!)>O_{(\text{vi})}(M^3)$.

Fig. 2 shows subjective and numerical evaluations of the matrix and FFT forms. The results were obtained after 20 iterations of ADMM. One can see the differences are vanishingly low since the difference of PSNR is only 0.01dB. Note that this difference of PSNR is independent of images since the difference mainly comes from the accuracy of the propagation based inverse calculation in (16) for the set of kernels but for the images in Fig. 1.

[4]The experiments run on a PC with Intel Core i7 2.7GHz and are implemented using mex (MSVC11 C++ without parallel computation) on MATLAB. We directly call the appropriate LAPACK functions of MATLAB to reduce the costs of function callbacks and checks for matrix information.

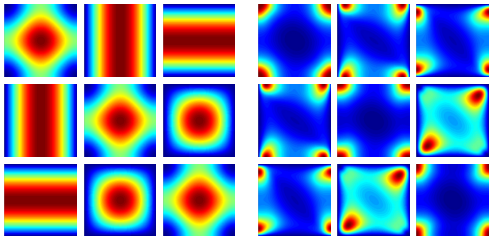
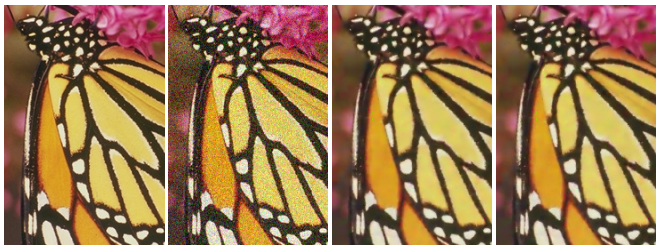


Fig. 1. Spectrums of FFTed BCCB filter blocks of TGV smoothing. (left) is blocks of $\mathbf{P}$ in (29) and (right) is their inverse.



(a) Original PSNR (dB) (b) Noisy 20.22 (c) Matrix form 28.67 (d) FFT form 28.66

Fig. 2. Subjective and numerical evaluations of matrix and FFT forms. (i) and (vii) are corresponding to methods shown in Table 1. The dB values denote PSNR between the original image (a). After YCoCg color conversion [18], each color layer is smoothed by TGV using the same parameters $\alpha_1=0.06$ and $\alpha_2=0.05$.

6. CONCLUSION

In this paper, we described an efficient computation method for multivariable l_2 equations. It utilizes the BCCB structure in the matrix blocks of the KKT system, and performs their calculation in the frequency domain using efficient pixel-wise inverse calculation. We also described the TGV smoothing method as an application, and improve the bottleneck of its l_2 part. We hope our method will help other applications.

7. ACKNOWLEDGEMENTS

The authors are grateful to S. Ono, the author of [17], for fruitful discussions about the TGV in Sec. 4.

8. REFERENCES

[1] N. Wiener, *Extrapolation, interpolation, and smoothing of stationary time series*, Wiley / MIT Press, 1942.

[2] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge Univ. Press, 2004.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[4] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM Trans. Graphics (SIGGRAPH)*, vol. 27, no. 3, pp. 73:1–73:10, 2008.

[5] L. Xu, C. Lu, Y. Xu, and J. Jia, "Image smoothing via l_0 gradient minimization," *ACM Trans. Graphics (SIGGRAPH Asia)*, vol. 30, no. 5, pp. 174:1–174:11, 2011.

[6] W. Karush, "Minima of functions of several variables with inequalities as side constraints," M.S. thesis, Dept. Math., Univ. Cicago, 1939.

[7] H.W. Kuhn and A.W. Tucker, "Nonlinear programming," in *Proc. Berkeley Symp. Math. Statist. and Prob.*, 1951, pp. 481–492.

[8] P. Smaragdis, "Blind separation of convolved mixtures in the frequency domain," *Elsevier J. Neurocomputing*, vol. 22, pp. 21–34, 1998.

[9] F. Asano, S. Ikeda, M. Ogawa, H. Asoh, and N. Kitawaki, "Combined approach of array processing and independent component analysis for blind separation of acoustic signals," *IEEE Trans. Speech and Audio Process.*, vol. 11, no. 3, pp. 204–215, 2003.

[10] A.N. Tikhonov, A.S. Leonov, and A.G. Yagola, *Nonlinear ill-posed problems*, vol. 1, Chapman and Hall, 1998.

[11] K. Shirai, M. Okuda, T. Jinno, M. Okamoto, and M. Ikehara, "Local covariance filtering for color images," in *Springer LNCS (ACCV 2012)*, 2013, pp. 1–12.

[12] R. M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[13] J. Rajagopalan, "An iterative algorithm for inversion of matrices," M.S. thesis, Dept. ECE, Concordia Univ., 1996.

[14] V. Pan and J. Reif, "Efficient parallel solution of linear systems," in *Proc. ACM Symp. Theory of Computing*, 1985, pp. 143–152.

[15] K. Bredies, K. Kunisch, and T. Pock, "Total generalized variation," *SIAM J. Imaging Sci.*, vol. 3, no. 3, pp. 92–526, 2010.

[16] S. Setzer, G. Steidl, and T. Teuber, "Infimal convolution regularizations with discrete l_1-type functionals," *Commun. in Math. Sci.*, vol. 9, no. 3, pp. 797–827, 2011.

[17] S. Ono and I. Yamada, "Optimized JPEG image decompression with super-resolution interpolation using multi-order total variation," in *Proc. IEEE ICIP*, 2013, pp. 474–478.

[18] H.S. Malvar, G.J. Sullivan, and S. Srinivasan, "Lifting-based reversible color transformations for image compression," in *SPIE Apps. Digital Image Process.*, 2008.

[19] Netlib Repository, "Linear algebra package (LAPACK)," www.netlib.org.