SECOND-ORDER CONE PROGRAMMING METHODS FOR TOTAL VARIATION-BASED IMAGE RESTORATION

DONALD GOLDFARB* AND WOTAO YIN†

May 25, 2004

Abstract. In this paper we present optimization algorithms for image restoration based on the total variation (TV) minimization framework of L. Rudin, S. Osher and E. Fatemi (ROF). Our approach formulates TV minimization as a second-order cone program which is then solved by interior-point algorithms that are efficient both in practice (using nested dissection and domain decomposition) and in theory (i.e., they obtain solutions in polynomial time). In addition to the original ROF minimization model, we show how to apply our approach to other TV models including ones that are not solvable by PDE based methods. Numerical results on a varied set of images are presented to illustrate the effectiveness of our approach.

Key words. image denoising, total variation, second-order cone programming, interior-point methods, nested dissection, domain decomposition.

AMS subject classifications. 68U10, 65K10,90C25, 90C51.

1. Introduction. Digital images, no matter what their source, usually contain noise. Consequently, a fundamental problem in image processing is the restoration or denoising of such images. Early methods for doing this were based on least squares and had the unfortunate property of either smoothing edges or creating spurious oscillations near edges, i.e., the well known ringing phenomenon. In [15], Rudin, Osher and Fatemi (ROF) proposed a method based on minimizing the *total variation* (TV) (also referred to as the bounded variation (BV) semi-norm) of the image. Specifically, they proposed solving the minimization problem:

$$\begin{array}{ll} \min & \int_\Omega |\nabla u|\, dx \\ \text{s.t.} & u + v = f \\ & \int_\Omega |v|^2\, dx \le \sigma^2. \end{array} \tag{1.1}$$

where Ω is an open (typically rectangular) domain in $\mathbb{R}^2$, $f : \Omega \to \mathbb{R}$ is a given noisy image in $L^2(\Omega)$, u is the computed estimate of the original image and σ^2 is an estimate of the variance of the noise in the image f. Minimizing the TV of u allows u to have discontinuities; hence edges and important features in the original image are preserved by the ROF approach.

Rather than solving the constrained minimization problem (1.1), ROF and subsequent researchers also considered the unconstrained minimization problem:

$$\min \quad \int_\Omega |\nabla u|\, dx + \lambda \int_\Omega |f - u|^2\, dx, \tag{1.2}$$

which yields that same solution as (1.1) for a suitable choice of the Lagrange muiltiplier λ. To solve (1.2), ROF proposed using an artificial time stepping method to compute the steady state solution u of the parabolic system $\frac{\partial u}{\partial t} = g(u)$ with initial condition $u = f$ and homogeneous Neumann boundary conditions, where

$$g(u) \equiv \frac{1}{2\lambda} \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right) + f - u. \tag{1.3}$$

*Columbia University, Dept. of IEOR, New York, NY 10027, USA. e-mail: goldfarb@columbia.edu

†Columbia University, Dept. of IEOR, New York, NY 10027, USA. e-mail: wy2002@columbia.edu

Research supported by NSF Grant DMS 01-04282, ONR Grant N00014-03-1-0514 and DOE Grant GE-FG01-92ER-25126

Note that $g(u) = 0$ is the Euler-Lagrange equation for (1.2), which is a necessary and sufficient condition for u to be a solution of the convex minimization problem (1.2). Other algorithms that have been proposed for solving problem (1.2) can be found in [4, 5, 7, 18].

We propose here a direct method for solving (1.1) based on the the observation that a discretized version of this convex optimization problem can be reformulated as a second-order program (SOCP). One can also reformulate a discretized version of the convex unconstrained minimization problem (1.2) as an SOCP. Our approach is to solve the SOCP representation of (1.1) by an interior point method taking advantage of the specific structure imparted to the SOCP by the image restoration problem and the TV-based denoising method. Of the many methods that have been proposed for solving (1.1) (or (1.2)) or related problems, it is most closely related to the one proposed by Chan et al. in [7] based on the interior point algorithm of Andersen et al [3].

In the next section we provide a formal definition of an SOCP, present the discretized version of Problem (1.1) that we will address, and introduce some notation. In section 3 we give an SOCP formulation of the ROF-based image restoration problem. We also present a dual formulation of problem (1.1) that is essentially equivalent to the dual formulation proposed by Chambolle [4], and show how it can be reformulated as an SOCP. In section 4 we discuss implementational aspects that enable interior point methods to solve these problems efficiently in practice. In particular, we show how nested dissection can be used to reduce storage and computational effort. A detailed analysis is presented in the Appendix. In theory the solution is obtained in polynomial time. Section 5 contains our numerical results. It also contains a discussion of how "overlapping domain decomposition" can be used to further reduce storage and run times. Finally, in section 6 we discuss how our approach can be applied to other image processing models. In particular, we show that it can be applied to the model introduced by Y. Meyer in [12] that is not solvable by PDE based methods.

2. Notation and Preliminaries. The images we discuss in this paper are defined as 2-dimensional $n \times n$ matrices in $\mathbb{R}^{n^2}$. We restrict our discussion to square domains only for the sake of simplicity. Let f be an observed image, which, in real applications, is a noisy version of an original image u. By introducing the noise term v, f and u have the following relation:

$$f_{i,j} = u_{i,j} + v_{i,j}, \quad \text{for } i, j = 1, \ldots, n. \tag{2.1}$$

$f_{i,j}$ and $u_{i,j}$ are the values of the observed image and the original image, respectively, at pixel (i, j). Let ∂^+ be the discrete differential operator defined by

$$\partial^+ u_{i,j} \stackrel{\text{def}}{=} \left((\partial_x^+ u)_{i,j}, (\partial_y^+ u)_{i,j} \right) \tag{2.2}$$

where

$$\begin{aligned} (\partial_x^+ u)_{i,j} &\stackrel{\text{def}}{=} u_{i+1,j} - u_{i,j}, \qquad \text{for } i = 1, \ldots, n-1, j = 1, \ldots, n, \\ (\partial_y^+ u)_{i,j} &\stackrel{\text{def}}{=} u_{i,j+1} - u_{i,j}, \qquad \text{for } i = 1, \ldots, n, j = 1, \ldots, n-1. \end{aligned} \tag{2.3}$$

In addition, the differentials on the image edges, $(\partial_x^+ u)_{n,j}$, for $j = 1, \ldots, n$, and $(\partial_y^+ u)_{i,n}$, for $i = 1, \ldots, n$, are defined to be zero. Consequently, they do not contribute to the total variation, which in the discrete case is defined by:

$$TV(u) \stackrel{\text{def}}{=} \sum_{1 \le i,j \le n} \|\partial^+ u_{i,j}\|, \tag{2.4}$$

where $\|\cdot\|$ denotes the Euclidean norm, i.e., $\|\partial^+ u_{i,j}\| = (\ ((\partial_x^+ u)_{i,j})^2 + ((\partial_y^+ u)_{i,j})^2\ )^{1/2}$.

Applying our definitions of f, u, v, and $\partial^+ u$ to the ROF restoration model (1.1), we obtain the finite dimensional minimization problem

$$\begin{array}{ll} \min & \sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| \\ \text{s.t.} & u + v = f \\ & \|v\| \le \sigma. \end{array} \tag{2.5}$$

Problem (2.5) can be reformulated as a second-order cone program (SOCP), the *standard form* of which we now define. The vector of variables $\mathbf{x} \in \mathbb{R}^n$ in a standard form SOCP is composed of subvectors $\mathbf{x_i} \in \mathbb{R}^{n_i}$ – i.e., $\mathbf{x} \equiv (\mathbf{x_1}; \mathbf{x_2}; \ldots; \mathbf{x_r})$ – where $n = n_1 + n_2 + \ldots + n_r$ and each subvector $\mathbf{x_i}$ must lie in an elementary *second-order cone* of dimension n_i

$$\mathcal{K}^{n_i} \equiv \{\mathbf{x_i} = (x_i^0; \bar{\mathbf{x}}_\mathbf{i}) \in \mathbb{R} \times \mathbb{R}^{n_i - 1} \mid \|\bar{\mathbf{x}}_\mathbf{i}\| \le x_i^0\}.$$

Definition 2.1. *The standard form SOCP (e.g., see [1]) is:*

$$\begin{array}{ll} \min & \mathbf{c_1}^\top \mathbf{x_1} + \cdots + \mathbf{c_r}^\top \mathbf{x_r} \\ \text{s.t.} & A_1 \mathbf{x_1} + \cdots + A_r \mathbf{x_r} = \mathbf{b} \\ & \mathbf{x_i} \in \mathcal{K}^{n_i}, \quad \text{for } i = 1, \ldots, r, \end{array} \tag{2.6}$$

where $\mathbf{c_i} \in \mathbb{R}^{n_i}$ *and* $A_i \in \mathbb{R}^{m \times n_i}$*, for* $i = 1, \ldots, r$ *and* $\mathbf{b} \in \mathbb{R}^m$*.*

Since a one-dimensional second-order cone corresponds to a semi-infinite ray, the above standard form SOCP can accomodate nonnegative variables. In fact if all cones $\mathcal{K}_i$ are one-dimensional, then the above SOCP is just a standard form linear program.

If we define $\mathcal{K}$ to be the Cartesian product $\mathcal{K} = \mathcal{K}^{n_1} \times \cdots \times \mathcal{K}^{n_r}$ and the vector $\mathbf{c}$ and the matrix A by $\mathbf{c} = (\mathbf{c_1}; \ldots; \mathbf{c_r})$, and $A = (A_1, \ldots, A_r)$, then the above SOCP can be written compactly as

$$\begin{array}{ll} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b} \\ & \mathbf{x} \in \mathcal{K} \end{array} \tag{2.7}$$

A crucial aspect of the above SOCP is that the cone $\mathcal{K}$ is pointed, closed, convex and *self-dual*– i.e., the *dual cone* $\mathcal{K}^* \equiv \left\{\mathbf{z} : \mathbf{x}^\top \mathbf{z} \ge 0, \forall \mathbf{x} \in \mathcal{K}\right\} = \mathcal{K}$ as is the cone corresponding to the nonnegative orthant in $\mathbb{R}^n$. This similarity between SOCPs and linear programs is more than superficial; as is the case for linear programs, SOCPs can be solved in polynomial time by interior point methods. This is the approach we propose in this paper.

3. Second-Order Cone Programming Formulation.

3.1. Primal Formulation. By introducing $t_{i,j}$ and the relations $((\partial_x^+ u)_{i,j})^2 + ((\partial_y^+ u)_{i,j})^2 \le (t_{i,j})^2$ for each pixel $i, j = 1, \ldots, n$, and including equations (2.3) that

define $(\partial_x^+ u)_{i,j}$ and $(\partial_y^+ u)_{i,j}$, problem (2.5) can be reformulated as:

$$\begin{array}{ll} \min & \sum_{1\le i,j\le n} t_{i,j} \\ \\ \text{s.t.} & u_{i,j} + v_{i,j} = f_{i,j}, \qquad\qquad\qquad \text{for } i,j = 1,\dots,n \\ & -(\partial_x^+ u)_{i,j} + (u_{i+1,j} - u_{i,j}) = 0, \quad \text{for } i = 1,\dots,n-1,\ j = 1,\dots,n \\ & -(\partial_y^+ u)_{i,j} + (u_{i,j+1} - u_{i,j}) = 0, \quad \text{for } i = 1,\dots,n,\ j = 1,\dots,n-1 \\ & (\partial_x^+ u)_{n,j} = 0, \qquad\qquad\qquad \text{for } j = 1,\dots,n \\ & (\partial_y^+ u)_{i,n} = 0,, \qquad\qquad\qquad \text{for } i = 1,\dots,n \\ & v_0 = \sigma \\ & (t_{i,j}; (\partial_x^+ u)_{i,j}; (\partial_y^+ u)_{i,j}) \in \mathcal{K}^3, \qquad \text{for } i,j = 1,\dots,n \\ & (v_0; v) \in \mathcal{K}^{n^2+1}, \end{array} \tag{3.1}$$

where, for $i,j = 1,\dots,n$, $u_{i,j}$, $(\partial_x^+ u)_{i,j}$, $(\partial_y^+ u)_{i,j}$, $v_{i,j}$ and $t_{i,j}$ are treated as variables and $f_{i,j}$, for $i,j = 1,\dots,n$, and σ as constants.

Although solving for u is our ultimate goal, in the above formulation u can be eliminated and $(\partial_x^+ u)_{n,j} = 0$, for $j = 1,\dots,n$, and $(\partial_y^+ u)_{i,n} = 0$, for $i = 1,\dots,n$, can be removed; hence, (3.1) simplifies to:

$$\begin{array}{ll} \min & \sum_{1\le i,j\le n,(i,j)\neq(n,n)} t_{i,j} \\ \\ \text{s.t.} & (\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j}, \quad \text{for } i = 1,\dots,n-1,\ j = 1,\dots,n \\ & (\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j}, \quad \text{for } i = 1,\dots,n,\ j = 1,\dots,n-1 \\ & v_0 = \sigma \\ & (t_{i,j}; (\partial_x^+ u)_{i,j}; (\partial_y^+ u)_{i,j}) \in \mathcal{K}^3, \qquad \text{for } i,j = 1,\dots,n-1 \\ & (t_{n,j}; (\partial_y^+ u)_{n,j}) \in \mathcal{K}^2, \qquad \text{for } j = 1,\dots,n-1 \\ & (t_{i,n}; (\partial_x^+ u)_{i,n}) \in \mathcal{K}^2, \qquad \text{for } i = 1,\dots,n-1 \\ & (v_0; v) \in \mathcal{K}^{n^2+1}. \end{array} \tag{3.2}$$

After solving the above problem, u can be recovered via $u_{i,j} = f_{i,j} - v_{i,j}$, for $i,j = 1,\dots,n$.

Problem (3.2) is a standard form second-order cone program. It has $4n^2 - 2n$ variables, $2n(n-1)+1$ equality constraints, $(n-1)^2$ 3-dimensional and $2(n-1)$ 2-dimensional second-order cone constraints, and one large second-order cone constraint.

3.2. Alternative (Dual) Formulations. Recalling the definition (2.2) of $\partial^+ u_{i,j}$, let us define $\partial^+ u \equiv \{\partial^+ u_{i,j}\} \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2}$. If we further define $q_{i,j} \equiv (q^1_{i,j}, q^2_{i,j})$ and $(\operatorname{div} q) \equiv \{(\operatorname{div} q)_{i,j}\}$, where $(\operatorname{div} q)_{i,j} = (\partial_x^- q^1)_{i,j} + (\partial_y^- q^2)_{i,j}$ and

$$\begin{array}{ll} (\partial_x^- q^1)_{i,j} \stackrel{\text{def}}{=} q^1_{i,j} - q^1_{i-1,j}, & \text{for } i = 2,\dots,n-1, j = 1,\dots,n, \\ (\partial_x^- q^1)_{1,j} \stackrel{\text{def}}{=} q^1_{1,j}, & \text{for } j = 1,\dots,n, \\ (\partial_x^- q^1)_{n,j} \stackrel{\text{def}}{=} -q^1_{n-1,j}, & \text{for } j = 1,\dots,n, \\ (\partial_y^- q^2)_{i,j} \stackrel{\text{def}}{=} q^2_{i,j} - q^2_{i,j-1}, & \text{for } i = 1,\dots,n, j = 2,\dots,n-1 \\ (\partial_y^- q^2)_{i,1} \stackrel{\text{def}}{=} q^2_{i,1}, & \text{for } i = 1,\dots,n, \\ (\partial_y^- q^2)_{i,n} \stackrel{\text{def}}{=} -q^2_{i,n-1}, & \text{for } i = 1,\dots,n-1. \end{array} \tag{3.3}$$

then $q \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2}$ and

$$\langle q, \partial^+ u\rangle = \sum_{i,j} (q^1_{i,j}(\partial_x^+ u)_{i,j} + q^2_{i,j}(\partial_y^+ u)_{i,j}) = \sum_{i,j} -(\operatorname{div} q)_{i,j} u_{i,j} = \langle -\operatorname{div} q, u\rangle. \tag{3.4}$$

Let z^* denote the optimal value of

$$\begin{array}{ll} \min & \sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| \\ \text{s.t.} & \|u-f\|^2 \le \sigma^2. \end{array} \tag{3.5}$$

Using the fact that $\|a\| = \max_{\|b\|\le 1} a^\top b$ (an immediate consequence of the Cauchy-Schwarz inequality), if we define the set

$$S \equiv \{q \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} \mid \|q_{i,j}\| \equiv \|(q^1_{i,j}, q^2_{i,j})\| \le 1, \text{ for all } i,j\},$$

then

$$z^* = \min_{\|u-f\|^2\le\sigma^2} \max_{q\in S} \langle q, \partial^+ u\rangle \tag{3.6}$$

$$= \min_{\|u-f\|^2\le\sigma^2} \max_{q\in S} \langle -\operatorname{div} q, u\rangle \tag{3.7}$$

$$= \max_{q\in S} \min_{\|u-f\|^2\le\sigma^2} \langle -\operatorname{div} q, u\rangle. \tag{3.8}$$

From the inner minimization, u must satisfy $u = f + \frac{\operatorname{div} q}{\|\operatorname{div} q\|}\sigma$. Hence, substituting this expression for u and letting $p = -q$ yields the following maximization problem:

$$z^* = \max_{p\in S} \left\{ \langle \operatorname{div} p, f\rangle - \sigma \|\operatorname{div} p\| \right\}. \tag{3.9}$$

In [4] Chambolle derived a dual formulation of problem (2), which is the Lagrangian version of (3.5), and proposed a method for solving this formulation using a partial differential equations approach. We now show how to derive this formulation in a direct manner. Specifically, Chambolle considered

$$\tilde{z}^* = \min_u \left\{ \frac{\|u-f\|^2}{2\lambda} + \sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| \right\}, \tag{3.10}$$

which, by arguments identical to those used to derive (3.8), is equivalent to

$$\tilde{z}^* = \frac{1}{2\lambda} \max_{q\in S} \left\{ \min_u \{\|u-f\|^2 + \langle -2\lambda \operatorname{div} q, u\rangle\} \right\}. \tag{3.11}$$

Since $u = f + \lambda \operatorname{div} q$ solves the inner minimization problem, substituting this expression for u yields

$$\tilde{z}^* = \frac{1}{2\lambda} \max_{q\in S} \left\{ \|\lambda \operatorname{div} q\|^2 + \langle -2\lambda \operatorname{div} q, f + \lambda \operatorname{div} q\rangle \right\} \tag{3.12}$$

$$= \frac{1}{2\lambda} \max_{q\in S} \left\{ -\|\lambda \operatorname{div} q + f\|^2 + \|f\|^2 \right\}, \tag{3.13}$$

which, letting $p = -q$, is equivalent to

$$\tilde{z}^* = \frac{1}{2\lambda}\|f\|^2 - \frac{1}{2\lambda} \min_{p\in S} \left\{ \|\lambda \operatorname{div} p - f\|^2 \right\}. \tag{3.14}$$

This minimization problem is identical to problem (8) in [4].

In fact, problems (3.9) and (3.14) are equivalent. Since (3.5) and its Lagrangian version (3.10) are equivalent, for a fixed q, the solutions to the inner problems of (3.8) and (3.11) are equal. Hence, it holds that

$$u = f + \frac{\operatorname{div} q}{\|\operatorname{div} q\|}\sigma = f + \lambda \operatorname{div} q,$$

which gives

$$\sigma = \lambda \|\operatorname{div} q\|. \tag{3.15}$$

Clearly, by adding the term $\|u-f\|^2/2\lambda$ to (3.9) and substituting the above expression for σ in it, we get (3.14). Therefore, solving (3.9), which gives $u = (f - \sigma \operatorname{div} p/\|\operatorname{div} p\|)$, and solving (3.14), which gives $u = (f - \lambda \operatorname{div} p)$, are essentially equivalent.

Next, we present the SOCP formulation of the problem $\min_{q\in S}\left\{\|\lambda \operatorname{div} p - f\|^2\right\}$. Following the definition of S, the discretization of the above problem gives the following SOCP:

$$\begin{array}{lll}
\min & w_0 & \\
\text{s.t.} & \lambda(\operatorname{div} q)_{i,j} - w_{i,j} = f_{i,j}, & \text{for } i,j = 1,\ldots,n, \\
& (w_0; (w_{i,j})) \in \mathcal{K}^{n^2+1}, & \\
& (q^0_{i,j}; q^1_{i,j}; q^2_{i,j}) \in \mathcal{K}^3, & \text{for } i,j = 1,\ldots,n-1, \\
& (q^0_{n,j}; q^2_{n,j}) \in \mathcal{K}^2, & \text{for } j = 1,\ldots,n-1, \\
& (q^0_{i,n}; q^1_{i,n}) \in \mathcal{K}^2, & \text{for } i = 1,\ldots,n-1, \\
& q^0_{i,j} = 1, & \text{for } i,j = 1,\ldots,n,
\end{array} \tag{3.16}$$

in which $(\operatorname{div} q)_{i,j}$ is explicitly given by $(\operatorname{div} q)_{i,j} = (\partial_x^- q^1)_{i,j} + (\partial_y^- q^2)_{i,j}$ and (3.3).

Many other dual formulations are possible. For example, if rather than relaxing the constraint $\|f-u\|^2 \le \sigma^2$, we relax the equivalent one, $\|f-u\| \le \sigma$, we obtain the Lagrangian version of (2.5):

$$\min \quad \textstyle\sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| + \lambda \|f-u\|, \tag{3.17}$$

which can be reformulated as the following SOCP:

$$\begin{array}{lll}
\min & \sum_{1\le i,j\le n} t_{i,j} + \lambda\, v_0 & \\
\text{s.t.} & (\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j}, & \text{for } i = 1,\ldots,n-1,\ j = 1,\ldots,n \\
& (\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j}, & \text{for } i = 1,\ldots,n,\ j = 1,\ldots,n-1 \\
& (t_{i,j}\,;\ (\partial_x^+ u)_{i,j}\,,\ (\partial_y^+ u)_{i,j}) \in \mathcal{K}^3, & \text{for } i,j = 1,\ldots,n-1 \\
& (t_{n,j}\,;\ (\partial_y^+ u)_{n,j}) \in \mathcal{K}^2, & \text{for } j = 1,\ldots,n \\
& (t_{i,n}\,;\ (\partial_x^+ u)_{i,n}) \in \mathcal{K}^2, & \text{for } i = 1,\ldots,n \\
& (v_0; v) \in \mathcal{K}^{n^2+1}. &
\end{array} \tag{3.18}$$

By assigning dual variables $y^1_{i,j}$ and $y^2_{i,j}$ to each of the first two constraints and associating $(s^0_{i,j}; s^1_{i,j}, s^2_{i,j})$ to $(t_{i,j}; (\partial_x^+ u)_{i,j}, (\partial_y^+ u)_{i,j})$ and $(w_0; w)$ to $(v_0; v)$, we obtain

the SOCP dual of (3.18):

$$\begin{array}{lll}
\max & \sum_{1\le i,j\le n, i\ne n} y^1_{i,j}(f_{i+1,j}-f_{i,j}) & \\
& +\sum_{1\le i,j\le n, j\ne n} y^2_{i,j}(f_{i,j+1}-f_{i,j}) & \\
\\
\text{s.t.} & -(\partial_x^- y^1)_{i,j}-(\partial_y^- y^2)_{i,j}+w_{i,j}=0, & \text{for } i,j=1,\ldots,n \\
& y^1_{i,j}+s^1_{i,j}=0, & \text{for } i=1,\ldots,n-1,\ j=1,\ldots,n \\
& y^2_{i,j}+s^2_{i,j}=0, & \text{for } i=1,\ldots,n,\ j=1,\ldots,n-1 \\
& s^0_{i,j}=1, & \text{for } i,j=1,\ldots,n \\
& w_0=\lambda, & \\
& (s^0_{i,j};\ s^1_{i,j},\ s^2_{i,j})\in\mathcal{K}^3, & \text{for } i,j=1,\ldots,n-1 \\
& (s^0_{n,j};\ s^2_{n,j})\in\mathcal{K}^2, & \text{for } j=1,\ldots,n \\
& (s^0_{i,n};\ s^1_{i,n})\in\mathcal{K}^2, & \text{for } i=1,\ldots,n \\
& (w_0;\ w)\in\mathcal{K}^{n^2+1}. &
\end{array} \tag{3.19}$$

By eliminating variables $y^1_{i,j}$, $y^2_{i,j}$, $s^0_{i,j}$ and w_0, this maximization problem can be simplified to the following minimization problem:

$$\begin{array}{lll}
\min & \sum_{1\le i,j\le n, i\ne n} s^1_{i,j}(f_{i+1,j}-f_{i,j}) & \\
& +\sum_{1\le i,j\le n, j\ne n} s^2_{i,j}(f_{i,j+1}-f_{i,j}) & \\
\\
\text{s.t.} & (\partial_x^- s^1)_{i,j}+(\partial_y^- s^2)_{i,j}+w_{i,j}=0, & \text{for } i,j=1,\ldots,n \\
& (1;\ s^1_{i,j},\ s^2_{i,j})\in\mathcal{K}^3, & \text{for } i,j=1,\ldots,n-1 \\
& (1;\ s^2_{n,j})\in\mathcal{K}^2, & \text{for } j=1,\ldots,n \\
& (1;\ s^1_{i,n})\in\mathcal{K}^2, & \text{for } i=1,\ldots,n \\
& (\lambda;\ w)\in\mathcal{K}^{n^2+1}. &
\end{array} \tag{3.20}$$

4. Nested Dissection. SOCPs can be solved very efficiently in practice and in polynomial time in theory by interior-point methods. For example, path-following interior-point methods generate a sequence of interior points (i.e., points in the interior of $\mathcal{K}$) that follow the so-called central path toward an optimal solution. At each iteration, a set of primal-dual equations, which depend on current variable values and a duality gap parameter μ, are solved to yield an improving search direction. Rather than solving this system of equations directly, most implementations of interior-point methods first reduce these equations to a smaller system of linear equations involving a symmetric positive definite matrix that has the form $AFA^\top$. This system of linear equations is solved by computing the Cholesky factorization $LDL^\top$ of $AFA^\top$, where L is a unit lower triangular matrix and D is a positive diagonal matrix. Although A remains fixed from one iteration to the next, F depends on the current primal-dual solution, so $AFA^\top$ must be factorized at every iteration. Even though interior-point methods terminate in a polynomial number of iterations, each iteration can be very time consuming due to the $O(m^3)$ computational complexity of solving the Cholesky factorization of the $m\times m$ matrix $AFA^\top$, or even fail because of the lack of enough memory to store a dense $m\times m$ lower triangular matrix L. In our applications, m is quite large (e.g., in the SOCP (3.2), $m=2n(n-1)+1$, so that for $n=1024$, m is roughly 2×10^6).

This difficulty can be overcome by exploiting the sparsity of the matrix A and the structure of F. In all interior point methods, F is a block diagonal matrix $F\equiv \mathrm{Diag}\{F_1,\ldots,F_r\}$, where each of the diagonal submatrices $F_i\in\mathbb{R}^{n_i\times n_i}$ can be expressed as a sum of a scaled identity matrix and two rank-one terms (or for

some methods, three rank-one terms), one with a positive coefficient and the other with a negative coefficient. Hence,

$$AFA^\top = \sum_{i=1}^{r} A_i F_i A_i^\top, \tag{4.1}$$

where $F_i = \gamma_i I + \mathbf{u_i}\mathbf{u_i}^\top - \mathbf{v_i}\mathbf{v_i}^\top$ ($\mathbf{u_i}$ and $\mathbf{v_i}$ are two columns vectors, not the signal and noise defined before).

It is easily verified that for the SOCP (3.2) the terms $A_i F_i A_i^\top$ have the same sparsity structure as $A_i A_i^\top$ for all i corresponding to 2 and 3 dimensional cones $\mathcal{K}^2$ and $\mathcal{K}^3$. However, for the one large cone $\mathcal{K}^{n^2+1}$, $A_i F_i A_i^\top$ has a totally dense $(n^2 + 1) \times (n^2 + 1)$ block, whereas $A_i A_i^\top$ is very sparse. To take advantage of the sparsity of the latter matrix, implementations of interior point methods compute the Cholesky factorization of only the sparse part of $AFA^\top$ — i.e., for large cones, they include only the sparse term $\gamma_i A_i A_i^\top$ in the sum (4.1) when computing this factorization and then update the factorization to account for the dense part using either a *product-form* approach (see ([10]) or a version of the Sherman-Morrison-Woodbury updating identity (see ([2])). The latter approach is simpler, but unfortunately less numerically stable than the former. The interior point codes that we used in our experiments treated large cones in this fashion; hence in the following discussion we consider only the sparsity structure of $A_i A_i^\top$.

It is well known that sparsity in a symmetric matrix M does not guarantee that the Cholesky factor L will be sparse. For example, Cholesky factorization of the rather sparse matrix

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 7 & 0 \\ 1 & 0 & 0 & 0 & 43 \end{bmatrix}. \tag{4.2}$$

yields $D = I$ and

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & -2 & 1 & 0 \\ 1 & -1 & -2 & -6 & 1 \end{bmatrix}. \tag{4.3}$$

However, by reversing the order of the rows and columns of M, we obtain a matrix $\tilde{M}$ whose Cholesky factorization yields $\tilde{D} = \mathrm{Diag}(43, 7, 3, 2, 1)$ and

$$\tilde{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1/43 & 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}. \tag{4.4}$$

Nested dissection [8] is a scheme for symmetrically permuting the rows and columns of a symmetric matrix M so that the resulting Cholesky factorization takes optimal

advantage of certain types of sparsity in M. To describe nested dissection, let G_M be the *adjacency graph* associated with M. If $M \in \mathbb{R}^{m\times m}$, then G_M has m nodes and has an edge (i,j) if the $(i,j)^{\text{th}}$ element of M is nonzero. Nested dissection recursively applies the following *divide and conquer* strategy. Given a graph G_M, find a *separator* set of nodes N_S, whose removal from G_M divides it into two unconnected components G_U and G_V with corresponding node sets N_U and N_V. If the nodes in N_S in G_M are numbered after those in N_U and N_V, then the matrix M corresponding to this row and column ordering has the form

$$M = \begin{bmatrix} M_{UU} & \mathbf{0} & M_{US} \\ \mathbf{0} & M_{VV} & M_{VS} \\ M_{US}^\top & M_{VS}^\top & M_{SS} \end{bmatrix}, \tag{4.5}$$

and most importantly, the Cholesky factor L of M inherits a similar structure, i.e.,

$$L = \begin{bmatrix} L_{UU} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & L_{VV} & \mathbf{0} \\ L_{SU} & L_{SV} & L_{SS} \end{bmatrix}. \tag{4.6}$$

This strategy is then applied recursively to the subgraphs G_U an G_V.

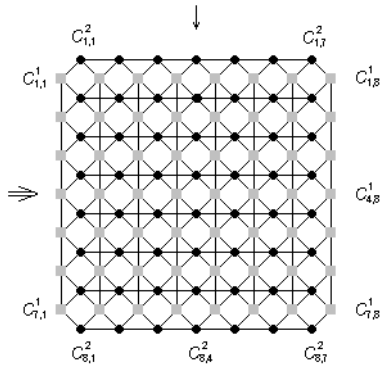

FIG. 4.1. *Constraint Dependency of (3.2)*

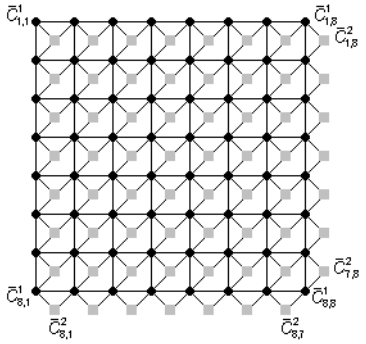

FIG. 4.2. *Constraint Dependency of (3.16)*

Let us now consider the application of nested dissection to the matrix $AA^\top$ arising from the SOCP (3.2), which as explained above corresponds to the sparse part of $AFA^\top$. $(AA^\top)_{l,k} \neq 0$ if and only if the l^{th} and k^{th} constraints share decision variables. The equality constraints in our problem are

$$(\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j}, \quad \text{for } i = 1,\ldots,n-1,\ j = 1,\ldots,n \tag{4.7}$$

$$(\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j}, \quad \text{for } i = 1,\ldots,n,\ j = 1,\ldots,n-1 \tag{4.8}$$

$$v_0 = \sigma. \tag{4.9}$$

Clearly, the last constraint shares no variable with any other constraint; thus its position in the ordering of the constraints is irrelevant . Let $C^1_{i,j}$ and $C^2_{i,j}$ denote the

$(i,j)^{\text{th}}$ constraints in (4.7) and (4.8), respectively. For a particular pair (i,j) (i.e., row of A), $C^1_{i,j}$ shares at least one decision variable with the following constraints, given their existence:

$$\begin{array}{llll} C^1_{i-1,j}, & C^2_{i,j-1}, & C^2_{i,j}, & (\text{sharing } v_{i,j}) \\ C^1_{i+1,j}, & C^2_{i+1,j-1}, & C^2_{i+1,j}. & (\text{sharing } v_{i+1,j}) \end{array}$$

$C^2_{i,j}$ shares at least one decision variable with the following constraints, given their existence:

$$\begin{array}{llll} C^1_{i-1,j}, & C^1_{i,j}, & C^2_{i,j-1}, & (\text{sharing } v_{i,j}), \\ C^1_{i-1,j+1}, & C^1_{i,j+1}, & C^2_{i,j+1}. & (\text{sharing } v_{i,j+1}), \end{array}$$

The constraint dependency relations (i.e., adjacency graph G_M of $AA^\top$ excluding the row and column corresponding to (4.9), which we shall refer to as M) are depicted in Figure 4.1 for the case of $n = 8$, in which squares represent $C^1_{i,j}$ and bullets $C^2_{i,j}$. Clearly, removal of the row of squares $C^1_{4,j}$, $j = 1,\ldots,8$, separates the dependency graph in Figure 4.1 into two independent parts. If we let $S \equiv \{C^1_{4,j} : j = 1,\ldots,8\}$ and let U and V denote the sets of constraints corresponding to the upper and the lower independent parts, and order the constraints in U before those in V and those in V before those in S, then M has the form of (4.5).

Furthermore, removal of the column of bullets $C^2_{i,4}$, $i = 1,\ldots,4$, separates U into two independent parts — U_L and U_R, and removal of the column of bullets $C^2_{i,4}$, $i = 5,\ldots,8$, separates V into two independent parts — V_L and V_R. As a result, M_{UU} and M_{VV} can be ordered so that each has the form (4.5). This process is then repeated recursively until every node has been assigned to some separator set (i.e., every row and column of M has been numbered). In the Appendix, we analyze the storage cost and the number of multiplications for factorizing $M = AA^T$ corresponding to the equality constraints (4.7) and (4.8) after ordering this matrix by nested dissection. These costs are $(31/4)n^2\log_2 n + O(n^2)$ and $943n^3/84 + O(n^2\log_2 n)$ for an $n \times n$ image, respectively.

Finally, we depict in Figure 4.2 the dependency graph of the dual SOCP formulation (3.16). In Figure 4.2 the $(i,j)^{\text{th}}$ bullet $\bar{C}^1_{i,j}$ represents the constraint $(\operatorname{div} q)_{i,j} - w_{i,j} = f_{i,j}$, and the $(i,j)^{\text{th}}$ square $\bar{C}^2_{i,j}$ represents the constraint $q^0_{i,j} = 1$. Since constraints $\bar{C}^1_{i,j}$, $\bar{C}^1_{i+1,j}$, and $\bar{C}^1_{i,j+1}$ contain either $q^1_{i,j}$ or $q^2_{i,j}$ or both of them, and the 3×3 fully dense block F_k corresponding to cone $(q^0_{i,j}; q^1_{i,j}; q^2_{i,j})$ causes the 3 corresponding columns in A to merge, constraint $\bar{C}^2_{i,j}$ in AFA^T essentially shares variables with constraints $\bar{C}^1_{i,j}$, $\bar{C}^1_{i+1,j}$, and $\bar{C}^1_{i,j+1}$; hence, in Figure 4.2, $\bar{C}^2_{i,j}$ is connected to $\bar{C}^1_{i,j}$, $\bar{C}^1_{i+1,j}$, and $\bar{C}^1_{i,j+1}$. The connections between bullets $\bar{C}^1_{i,j}$'s are due to the sharing of $q^1_{i,j}$ or $q^2_{i,j}$, for some i,j.

5. Numerical Results. In this section, we present numerical results that illustrate the effectiveness of our SOCP approach and compare it to the PDE approach. We then discuss the use of domain decomposition to reduce the storage and computational effort needed by the SOCP approach.

5.1. SOCP/PDE Comparison. We used the commercial optimization package Mosek® as our SOCP solver. Mosek is designed to solve a variety of large-scale optimization problems, including SOCPs. Before solving a large-scale problem, Mosek uses a presolver to remove redundant constraints and variables and to reorder constraints. The Mosek presolver reorders the coefficient matrix A using a combination of

a graph partitioning based method and an approximate minimum local-fill-in ordering method to increase the sparsity of the Cholesky factors of $AFA^\top$. This general-purpose reordering produced results for our problem that were very good, but not as good as nested dissection, which further reduced the time per iteration and the total time by approximately 30 percent in our tests on a 512×512 image. Therefore, we turned off Mosek's reordering heuristic and replaced it by our own nested dissection reordering code in our tests.

Existing PDE approaches require the ROF model to be represented in the relaxed form (1.2) and the existence of the Euler-Lagrange equation $g(u) = 0$, where $g(u)$ is given by (1.3). To handle the case of $|\nabla u| = 0$, most PDE approaches add a small perturbation $\epsilon > 0$ to $|\nabla u|$ in (1.3). The PDE method that we used in our numerical tests is an artificial time stepping method which computes the steady state solution u of the parabolic system

$$\frac{\partial u}{\partial t} = \frac{1}{2\lambda} \nabla \cdot \left(\frac{\nabla u}{|\nabla u| + \epsilon} \right) + f - u. \tag{5.1}$$

To compare the two approaches for each noisy image, we first solved the SOCP formulation (3.2) of (1.1) and calculated λ from the optimal dual solution using the following theorem.

THEOREM 5.1. *Problem (1.1) and (1.2) have the same optimal solution u if $2\,\lambda\,\sigma = z_0$, where z_0 is the first entry of $(z_0, z) \in \mathcal{K}^{n^2+1}$ and (z_0, z) is the dual vector corresponding to (v_0, v). (Recall: $\int_\Omega |v|^2\, dx \le \sigma^2$ is represented in the discrete form by $(v_0, v) \in \mathcal{K}^{n^2+1}$ with $v_0 = \sigma$).*

We then used this Lagrange multiplier λ in the PDE method. In practice, it is much easier to estimate the noise level of an image than it is to estimate an appropriate λ. Experiments show that a σ that is little smaller than the true one (i.e., a slight amount of under-smoothing) yields the best results.

In our tests, we called Mosek from Matlab® while we called the PDE-based code (an improved version of a C++ code provided to us by Stanley Osher and Jinjun Xu of UCLA) directly. The running times given in Table 5.2 are the total times needed to solve the problem on a Sun E450 with four 300MHz SuperSparc II processors and 4GB memory. Since both codes are single-threaded, they run only on one processor at a time. The SOCP run times do not include the time taken by our nested dissection code to reorder constraints and the time taken by Mosek for presolving, since the process of constraint reordering and presolving remains the same for all images of the same size.

During their execution, both approaches record the objective values of (1.2):

$$z(t) = \int_\Omega |\nabla u_t|\, dx + \lambda \int_\Omega |f - u_t|^2\, dx. \tag{5.2}$$

The objective values obtained by the PDE approach decrease with time t over the long run but fluctuate and never reach as low a value as that obtained by the SOCP approach. Given a nearly optimal objective value, the PDE approach may have to iterate further for a large number of steps to obtain a lower objective value. For these reasons, we used the following stopping rules for the PDE approach:

- initially, let $z_{\min} = \infty$ and $t_{\min} = 0$, where $z_{\min}$ and $t_{\min}$ denote the minimal objective value and its time;
- at time t, update $z_{\min} := z(t)$ and $t_{\min} := t$ if $z_{\min} < z(t) - \lambda$;
- for denoising an image of size $n \times n$, stop at time t once $t - t_{\min} \ge \max\{0.15 t_{\min}, n\}$.

TABLE 5.1
Numerical comparison between SOCP(Matlab+Mosek) and PDE(C++) approaches (cont.)

Name	Size $n \times n$	Fig.	SOCP result u_1 v.s. PDE result u_2	
			$\max_{i,j} \lvert u_1 - u_2 \rvert$	$\lVert u_1 - u_2 \rVert_{L1}/n^2$
books	64×64	6.5 - 6.7	0.58	0.04
dotplus	100×100	6.8 - 6.10	0.97	0.10
scale	252×252	6.11 - 6.13	0.74	0.02
Barbara	256×256	6.14 - 6.16	0.90	0.05
Barbara	350×350	6.17 - 6.20	0.95	0.06
Barbara	512×512	6.27 - 6.30	0.96	0.06

TABLE 5.2
Numerical comparison between SOCP(Matlab+Mosek) and PDE(C++) approaches

Name	λ	SOCP			PDE ($\Delta t = 0.1$)		
		total time	itr.	obj. value	total time	itr.	obj. value
books	0.0374718930	4.42	15	128927	3.22	950	128976
dotplus	0.0180513895	13.68	16	592183	14.90	1875	592438
scale	0.0337088230	157.86	17	3706529	81.21	1479	3706980
Barbara	0.0374657886	130.13	16	1691196	61.73	1166	1692180
Barbara	0.0374969349	283.56	16	2905191	151.71	1464	2907500
Barbara	0.0375681804	806.70	16	5895359	456.25	1900	5900690

These stopping criteria ensure that the PDE method spends a reasonable amount of time searching for a possibly lower objective value, but does not waste too much time trying to obtain an insignificant improvement.

In all of our tests starting from the same noisy image, the two approaches produced images that were visually indistinguishable from one another. Therefore, we present in Figures 6.5-6.19, in addition to the original and noisy images, only the SOCP images. In the case of the 350×350 image of Barbara, we also present in Figure 6.20 the pixel-by-pixel differences between the two images u_1 (SOCP) and u_2 (PDE) amplified 10 times with the gray-scale inverted so that the darker the pixel, the greater the difference. Specifically, we used the formula $255 - 10 \times |u_1 - u_2|$, where u_1 and u_2 had double precision values and were not rounded. From Figure 6.20, we can see that the two approaches give almost identical denoising results. Similar results were obtained on the other images.

From Table 5.2, one can see that the SOCP approach requires a smaller number of iterations than the PDE approach while every iteration takes more time. Note that, as the size of the image grows, the number of iterations required by the SOCP approach remains relatively constant, while that number grows for the PDE approach. In spite of this, the SOCP approach took roughly between one and two times the total time that the PDE approach took. This is because, in theory, the PDE method takes $O(n^2)$ CPU time per iteration for an $n \times n$ image while the SOCP method (implemented as described in Section 4) takes $O(n^3)$ CPU time per iteration. In our numerical tests the SOCP approach took between $O(n^2 \log n)$ and $O(n^3)$ time per iteration due to the effect of memory references which are slower than floating-point operations.

We also measured the running times on a PC with a Pentium-IV 1.7GHz CPU

and 512MB memory. We used the MS Windows versions of Matlab, Mosek and GUN g++ compiler. On this newer computer, the SOCP code ran 2-2.5 times faster and the PDE code ran about 4 times faster solving the same set of problems. Due to the PC's smaller amount of memory and cache, the SOCP approach, which performs more memory fetches, got a smaller speedup than the PDE approach, exhibiting a running time somewhere between the $n \times n$-image storage cost of $O(n^2)$ and the nested dissection Cholesky factorization storage cost of $O(n^2 \log n)$.

5.2. Domain Decomposition. Even with nested dissection the SOCP approach requires $O(n^2 \log n)$ memory. Hence, to denoise large (or 3D) images, we consider here a *domain decomposition* approach. Let $\tilde{n} \approx n/k$, where k is a positive integer and $\tilde{n} > n/k$. By sub-dividing an n-by-n image into k^2 $\tilde{n}$-by-$\tilde{n}$ subimages with some overlap and applying the SOCP method to the subimages sequentially, we can reduce the storage needed by a factor of approximately $1/k^2$. Since, in applying the SOCP method to each subimage, the number of multiplications per iteration is reduced by a factor of approximately $1/k^3$ compared with applying the SOCP method to the entire image, and there are k^2 subimages, the total cpu time should also be reduced by a factor of approximately $1/k$. After obtaining the denoised subimages, the rows (columns) of pixels in that half of any region of overlap closest to the boundary of the subimage were discarded.

Due to the different graphical characteristics of the subimages, including SNR, contrast, and homogeneity, applying the SOCP method to all subimages using the same σ does not usually give the same output as applying the SOCP method to the entire image. In some cases the results obtained by using the former are better while in other cases they are worse. The following numerical example illustrates this. Figure 6.21 is the noisy image obtained by selecting a rectangular area of the noisy image in Figure 6.15 and removing the outside portion of the image. Figure 6.23 is the result of the ROF method (3.2) separately applied to four equal subimages of size 65×65 with the same $\sigma = 18 \times 65^2$. In the numerical tests, we chose the size of all subimages so that all adjacent image pairs had overlapping regions that were 4 pixels wide. Figure 6.22 illustrates the 2×2 domain decomposition applied to the image in Figure 6.21. Figure 6.24 is the result of ROF applied to the entire image with $\sigma = 18 \times 65^2$. In Figure 6.24, Barbara's left eye is clearer than in Figure 6.23. This shows that the subimage at the bottom right is over-smoothed. After reducing $\sigma_{2,2}$ from 18×65^2 to 16×65^2 for this over-smoothed subimage, the ROF method yields Figure 6.25, in which Barbara's left eye is much clearer. Figure 6.26 gives the 10 times amplified differences between Figure 6.24 and 6.25. The differences in the regions of the four subimages vary, but we cannot see any big differences by comparing Figure 6.24 and Figure 6.25. However, as we now shall demonstrate, we can obtain even closer results.

In [16], Strong and Chan give exact analytical solutions to the relaxed TV denoising problem (1.2) applied to 1D, 2D and 3D piecewise constant images and smooth radially symmetric ones. Their results establish an exact relationship between the Lagrange multiplier λ and the effect of TV regularization (1.2) on an image feature and show its localness property (i.e., in a piecewise constant or smooth radially symmetric region the solution of (1.2) depends only on λ and whether or not the region is on the image boundary, and is independent to the features of other regions). This can be extended approximately to complicated noisy images since they are composed of many tiny piecewise constant or smooth radially symmetric regions. Therefore, we can expect solving (1.2) with the same λ (or σ's for all subimages so that $z_0/(2\sigma) = \lambda$) as used for the entire image to give close results. Since TV regulation treats regions on

TABLE 5.3
4×4 *domain decomposition applied to Figure 6.27*

blk.	itr.	time	blk.	itr.	time	blk.	itr.	time	blk.	itr.	time
(1,1)	19	33.03	(1,2)	20	34.84	(1,3)	18	32.01	(1,4)	17	29.95
(2,1)	18	31.97	(2,2)	18	31.74	(2,3)	17	29.73	(2,4)	17	30.25
(3,1)	18	31.34	(3,2)	17	30.61	(3,3)	20	35.70	(3,4)	18	31.18
(4,1)	17	30.10	(4,2)	17	31.06	(4,3)	18	32.03	(4,4)	18	32.72
Sum of the total time for denoising all subimages (Figure 6.29):											508.26
Total time for denoising the entire image:											806.70

TABLE 5.4
8×8 *domain decomposition applied to Figure 6.27*

time		column							
(itr.)		1	2	3	4	5	6	7	8
	1	6.86(16)	6.56(16)	8.37(19)	6.24(15)	5.80(14)	5.32(13)	5.67(14)	6.12(14)
	2	7.07(17)	6.90(16)	6.51(16)	5.70(14)	6.04(14)	6.02(14)	6.38(15)	5.83(14)
r	3	5.65(14)	6.64(16)	6.90(16)	7.53(17)	7.02(16)	5.86(14)	5.77(14)	6.02(15)
o	4	6.76(16)	7.17(17)	7.21(17)	5.79(14)	5.88(13)	5.84(14)	5.38(13)	6.29(15)
w	5	6.19(15)	7.19(17)	6.63(16)	6.17(14)	5.93(14)	5.75(14)	6.74(16)	6.97(17)
	6	5.99(14)	6.75(16)	6.88(17)	7.06(16)	6.80(15)	5.73(14)	6.50(15)	6.21(15)
	7	7.06(17)	7.22(17)	7.16(16)	6.59(15)	6.35(15)	6.35(15)	6.71(16)	6.14(15)
	8	7.04(17)	7.58(17)	7.10(17)	7.83(19)	7.25(17)	6.39(15)	7.13(17)	7.57(18)
Sum of the total time for denoising all subimages:									418.06

boundaries differently, domain decomposition may give different results near subimage boundaries. This effect, however, is limited to strips a few pixels wide when noise is present. Hence, in our tests, we discarded redundant information in these boundary strips of the subimages.

In our first test of the above approach we applied a 4×4 domain decomposition to a 512×512 image. The results are depicted in Figures 6.27-6.30. All subimages were of size 137×137, giving overlapping regions between adjacent subimages that were 12 pixels wide. The image values for the 6 rows (columns) of pixels in the overlapping regions closest to the boundaries of each subimage were discarded. The σ's for the full image and all of the subimages were chosen so that $z_0/(2\sigma) = 0.0375$. Table 5.3 gives the numerical results. The 10-time amplified difference between the results obtained by the two approaches is given by Figure 6.30. The difference was more than 1 on the 256 grey level scale for only 0.0008% of the pixels. The average difference per pixel was 0.04 and the maximum difference was +0.75 and -1.31. Clearly, SOCP with domain decomposition gave nearly identical results to SOCP applied to the full image.

In this example the total computation time for the domain decomposition approach was about 37 percent less than the time required by the SOCP method applied to the full image. Since $16 \cdot 137^3/512^3 \approx 0.33$, one would expect something close to a 67 percent reduction in running time. The reason this was not the case is explained by our remark in Section 5.1 about execution time and memory references when cache is limited and slow.

We also applied an 8×8 domain decomposition to the noisy 512×512 image in Figure 6.28 after extending it to one with a size of 516×516 so that all subimages were

of the same size (75 x 75) and the overlapping regions remained 12 pixels wide. As in the previous test, the σ's were chosen so that $z_0/(2\sigma) = 0.0375$ for all subimages. The numerical results given in Table 5.4 show that the reduction in the total computation time was further improved from 37 percent to 52 percent. Moreover, the total run time of 418.06 seconds was less than the 456.25-second run time of the PDE approach. The difference between the results obtained by applying the SOCP method with domain decomposition and applying it to the entire image was more than 1 on the 256 grey level scale for 0.0015% of the pixels. The average difference per pixel increased to 0.08 and the maximum difference was +1.47 and -1.06. This difference was still not perceivable to the naked eye.

6. SOCP Formulations of Other Restoration Models. Y.Meyer [12] introduced the following image restoration model based on TV and the space of oscillating functions:

$$\min \int |\nabla u| + \lambda \|v\|_*, \text{ subject to } f = u + v, \tag{6.1}$$

where $\|v\|_*$ is defined as the infimum of $\|\sqrt{g_1^2(x,y) + g_2^2(x,y)}\|_{L^\infty}$ over all $g_1, g_2 \in L^\infty(\mathbb{R}^2)$ satisfying $v = \partial_x g_1 + \partial_y g_2$. To handle $\|\sqrt{g_1^2(i,j) + g_2^2(i,j)}\|_{L^\infty}$ in the discretized version of (6.1), we introduce a 3-dimensional quadratic cone $(g_0(i,j); g_1(i,j), g_2(i,j)) \in \mathcal{K}^3$, for each i, j, and let $w \geq g_0(i,j)$, for all i, j. The infimum of $\|\sqrt{g_1^2(i,j) + g_2^2(i,j)}\|_{L^\infty}$ is then equal to the minimum of w. Thus we can solve a discretized problem of (6.1) as an SOCP [9]. It is important to note that because Meyer's model (6.1) is not amenable to the PDE approach, it has primarily been of theoretical interest. Thus, the SOCP approach can be applied to image restoration models that cannot be solved by a PDE approach.

In [17], Vese and Osher proposed an approximate version of Meyer's model and showed that their PDE-based method can decompose a given image f into u and v, where u and v represent cartoon and texture respectively. Their method solves the following minimization problem:

$$\min \int |\nabla u| + \lambda \int |f - u - \partial_x g_1 - \partial_y g_2| + \mu \left[\int (\sqrt{g_1^2 + g_2^2})^p \right]^{1/p}, \tag{6.2}$$

where λ and μ are tuning parameters and $p \geq 1$. Inequalities with p-norm for p being rational can be represented as a system of second-order conic inequalities (refer to [1]). Therefore, (6.2), after discretization, can be formulated as an SOCP and solved by interior-point methods. In a forthcoming paper [9], we solve Meyer's model as an SOCP and compare the solutions with those obtained by the Vese-Osher approximation.

In [11], Lysaker, Lundervold and Tai introduced a method for image restoration based on solving the following 2-D problem

$$\begin{array}{ll} \min & \int_\Omega |u_{xx}| + |u_{yy}| \, dx \, dy \\ \text{s.t.} & u + v = f \\ & \int_\Omega |v|^2 \, dx \, dy \leq \sigma^2. \end{array} \tag{6.3}$$

The discretization of the second derivative of $u = \{u_{i,j}\}$ is linear in the $u_{i,j}$'s. To handle absolute values in minimization problems, say $\min |x|$, one can introduce an

extra variable and transform $\min |x|$ into equivalent problems:

$$\min \quad |x| \tag{6.4}$$

$$\Longleftrightarrow \min \quad t \quad \text{s.t.} \quad x \le t,\ -x \le t \tag{6.5}$$

$$(s \overset{\text{def}}{=} t + x) \Longleftrightarrow \min(s - x) \text{ s.t.} \quad 2x \le s,\ s \ge 0. \tag{6.6}$$

Both (6.5) and (6.6) have a linear objective subject to linear constraints. In some cases, (6.6) is preferred since $s \ge 0$ is easier to handle. Therefore, expressing $\int_\Omega |v|^2\, dx\, dy \le \sigma^2$ as a second-order conic constraint as before, it is clear that problem (6.3) can be formulated as an SOCP. This also works if the objective function is replaced by $\int_\Omega |u_{xx}| + 2|u_{xy}| + |u_{yy}|\, dx\, dy$.

A variant of (1.1) was proposed by Osher, Sole and Vese in [13]. The model assumes that $f - u = \operatorname{div} g$, where $g = (g_1, g_2)$, and there exists a unique decomposition $g = \nabla P + Q$, where Q satisfies $\operatorname{div} Q = 0$. Therefore, $f - u = \operatorname{div} g = \Delta P$ or $\Delta^{-1}(f - u) = P$. The OSV model is

$$\begin{array}{ll} \min & \int_\Omega |\nabla u|\, dx \\ \text{s.t.} & \int_\Omega |\nabla(\Delta^{-1}(f-u))|\, dx \le \sigma^2, \end{array} \tag{6.7}$$

or equivalently

$$\begin{array}{ll} \min & \int_\Omega |\nabla u|\, dx \\ \text{s.t.} & \int_\Omega |\nabla P|\, dx \le \sigma^2 \\ & f - u = \Delta P. \end{array} \tag{6.8}$$

Clearly, by using a finite difference form of the Laplace operator and expressing the objective function and the constraint $\int_\Omega |\nabla P|\, dx \le \sigma^2$ as we have previously described, we can reformulate (6.8) as an SOCP.

In [6], the authors analyzed the ROF model with an L^1 fidelity term $\int_\Omega |v| \le \sigma$. Clearly, by discretizing $\int_\Omega |v|$ as the sum $\sum_{i,j} |v_{i,j}|$ and handling the absolute values $|v_{i,j}|$ using (6.5) or (6.6), this problem can be formulated as an SOCP.

It is not difficult to see that various TV based deblurring models can also be formulated as SOCPs.

Appendix. In this appendix, we present a detailed analysis of Cholesky factorization of a matrix M whose rows and columns have been symmetrically reordered by applying nested dissection to the adjacency graph of M, assuming that the latter is an $n \times n$ grid graph of the type shown in Figure 4.1. We follow an approach similar to the one given in [14]. First, let us introduce some graph notation.

A graph G is defined as the pair (V_G, E_G), where V_G is the set of nodes and E_G is the set of edges of G. In Figure 6.1, each square or bullet is a node, and each line segment connecting a pair of nodes is an edge. Figure 6.1 has 8 rows and 7 columns corresponding to the C^2 constraints, and 7 rows and 8 columns corresponding to the C^1 constraints. For convenience, we number node $C^k_{i,j}$ by $(i,j)^k$. In this way, $C^1_{3,1}$ corresponds to node $(3,1)^1$ and $C^2_{8,1}$ to node $(8,1)^2$. Due to the boundary conditions, the row of nodes $(8,\cdot)^1$ and the column of nodes $(\cdot,8)^2$, corresponding to $(\partial^+_x u)_{8,j} = 0$ and $(\partial^+_y u)_{i,8} = 0$, for $1 \le i, j \le 8$, do not appear in Figure 6.1. Let us denote by $G_{2n,2n}$ a graph of the form of Figure 6.1 with nodes $(i,j)^k$, for $k = 1, 2$ and $i, j = 1, \ldots, n-1$, $(i,n)^1$, for $i = 1, \ldots, n$, and $(n,j)^2$, for $j = 1, \ldots, n$. For example, Figure 6.1 depicts a $G_{16,16}$ mesh.

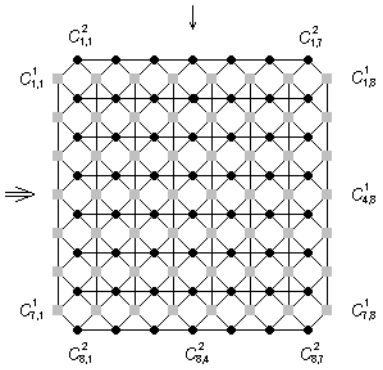

FIG. 6.1. $G_{16,16}$

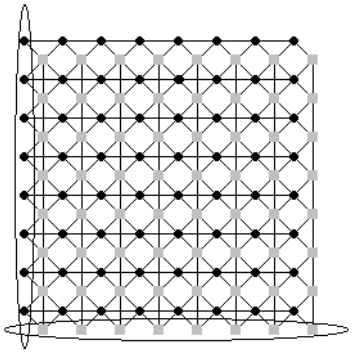

FIG. 6.2. $A_{16,16}$

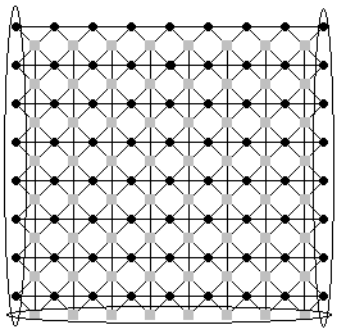

FIG. 6.3. $B_{16,16}$

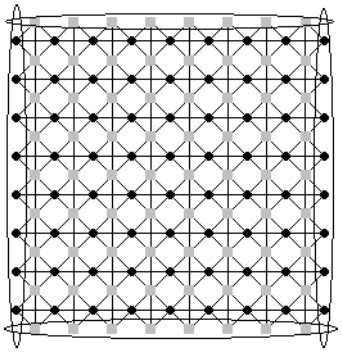

FIG. 6.4. $C_{16,16}$

Given the graph $G_{2n,2n}$, we define three extended graphs $A_{2n,2n}$, $B_{2n,2n}$, and $C_{2n,2n}$ as follows:

$$A_{2n,2n} \stackrel{\text{def}}{=} (V_A, E_A), \text{where} V_A = V_G \bigcup L_{left} \bigcup L_{bottom} \tag{6.9}$$

$$B_{2n,2n} \stackrel{\text{def}}{=} (V_B, E_B), \text{where} V_B = V_G \bigcup L_{left} \bigcup L_{bottom} \bigcup L_{right} \tag{6.10}$$

$$C_{2n,2n} \stackrel{\text{def}}{=} (V_C, E_C), \text{where} V_C = V_G \bigcup L_{top} \bigcup L_{left} \bigcup L_{bottom} \bigcup L_{right}, \tag{6.11}$$

where

$$L_{left} = \{(i,0)^2 \mid i = 1, 2, \ldots, n\} \tag{6.12}$$

$$L_{right} = \{(i,n)^2 \mid i = 1, 2, \ldots, n\} \tag{6.13}$$

$$L_{top} = \{(0,j)^1 \mid j = 1, 2, \ldots, n\} \tag{6.14}$$

$$L_{bottom} = \{(n,j)^1 \mid j = 1, 2, \ldots, n\}, \tag{6.15}$$

and E_A, E_B, and E_C are the sets of edges that connect V_A, V_B, and V_C by a grid of the type shown in Figure 6.1. For example, for $n = 8$, $A_{16,16}$, $B_{16,16}$, and $C_{16,16}$ are depicted in Figures 6.2, 6.3, and 6.4, in which extended boundaries are distinguished by being enclosed in ellipses.

The reason that we need these extended graphs in our analysis can be illustrated as follows. As we described in section 4, the graph $G_{16,16}$ can be separated into four subgraphs, each of which is a $G_{8,8}$, by separators $(4,j)^1$, $j = 1,\ldots,8$, followed by $(i,4)^2$, $i = 5,\ldots,8$, and $(i,4)^2$, $i = 1,\ldots,4$. The order of the graph separators is just reverse of the order assigned to the rows and columns in the matrix. It is well known that, in Cholesky factorization (CF), eliminating a node of degree d (i.e., the number of nodes directly connected to it) costs $K_m = d(d+3)/2$ multiplies and requires $K_s = d+1$ storage. In the next stage the case is different from that of $G_{8,8}$ since, for example, the subgraph $G_{8,8}$ in the upper right corner of $G_{16,16}$ is connected to the nodes $(4,j)^1$, $j = 5,\ldots,8$, and $(i,4)^2$, $i = 1,\ldots,4$, which are the bottom and left extended boundaries of $G_{8,8}$. Therefore, we need to consider separating $A_{8,8}$, instead of $G_{8,8}$. Since applying any rotation or reflection to $A_{8,8}$ does not change the multiplication and storage costs of CF, separating any of the other three subgraphs of $G_{16,16}$ is essentially the same as separating $A_{8,8}$. Similar remarks apply to any rotation or reflection of $A_{2n,2n}$, $B_{2n,2n}$, and $C_{2n,2n}$.

Let $\mu(\cdot)$ denote a function of a graph whose value gives the complexity of applying CF to a matrix corresponding to that graph. Specifically, we use $\mu_m(\cdot)$ and $\mu_s(\cdot)$ to present the multiplication cost and the storage cost, respectively. In the divide-and-conquer approach, the complexity of applying CF to $G_{2n,2n}$ can be calculated by the following recursion

$$\mu(G_{2n,2n}) = 4\,\mu(A_{n,n}) + \mu(S^G_{2n,2n}), \tag{6.16}$$

where $S^G_{2n,2n}$ is defined as the separator of $G_{2n,2n}$.

Following similar arguments, one can divide each of $A_{2n,2n}$, $B_{2n,2n}$, and $C_{2n,2n}$ into 4 subgraphs and derive the following recursions:

$$\mu(A_{2n,2n}) = \mu(A_{n,n}) + 2\,\mu(B_{n,n}) + \mu(C_{n,n}) + \mu(S^A_{2n,2n}) \tag{6.17}$$

$$\mu(B_{2n,2n}) = 2\,\mu(B_{n,n}) + 2\,\mu(C_{n,n}) + \mu(S^B_{2n,2n}) \tag{6.18}$$

$$\mu(C_{2n,2n}) = 4\,\mu(C_{n,n}) + \mu(S^C_{2n,2n}). \tag{6.19}$$

The next step is to calculate $\mu(S^G_{2n,2n})$, $\mu(S^A_{2n,2n})$, $\mu(S^B_{2n,2n})$, and $\mu(S^C_{2n,2n})$. Take $\mu(S^G_{16,16})$, for example. Recall that we eliminate $(i,4)^2$, $i = 1,\ldots,4$, followed by $(i,4)^2$, $i = 5,\ldots,8$, and $(4,j)^1$, $j = 1,\ldots,8$. At the time we eliminate $(1,4)^2$, all nodes except for those in $S^G_{2n,2n}$ are eliminated. Therefore, $(1,4)^2$ is connected to $(i,4)^2$, $i = 2,\ldots,4$, and $(4,j)^1$, $j = 1,\ldots,8$. Therefore, $d_{(1,4)^2} = 3\times(n/2) - 1$, where $n = 8$. Next, $(2,4)^2$ is eliminated with $d_{(2,4)^2} = 3\times(8/2) - 2$. When we eliminate $(4,4)^2$, it has degree of $3\times(8/2) - 4 = 8$ since it is connected to eight nodes $(4,j)^1$, $j = 1,\ldots,8$. Hence, the sums of d and d^2 for $G_{2n,2n}$'s separator $(i,n/2)^2$, $i = 1,\ldots,n/2$, are given by $\sum_{j=n}^{3n/2-1} j$ and $\sum_{j=n}^{3n/2-1} j^2$, respectively. By symmetry, this is also true for the separator $(i,n/2)^2$, for $i = n/2+1, n/2+2,\ldots,n$. Last, we eliminate the n nodes $(n/2,j)$, for $j = 1,2,\ldots,n$. Whenever we eliminate one of them, it is connected to all the nodes to be eliminated. Therefore, for these last n nodes,

the sums of d and d^2 are given by $\sum_{j=0}^{n-1} j$ and $\sum_{j=0}^{n-1} j^2$, respectively. Altogether,

$$\sum_{v \in S^G_{2n,2n}} d_v^2 = 2 \sum_{j=n}^{3n/2-1} j^2 + \sum_{j=0}^{n-1} j^2 = \frac{23}{12} n^3 + O(n^2) \tag{6.20}$$

$$\sum_{v \in S^G_{2n,2n}} d_v = 2 \sum_{j=n}^{3n/2-1} j + \sum_{j=0}^{n-1} j = \frac{7}{4} n^2 + O(n), \tag{6.21}$$

where the first summand is for nodes $(i, n/2)^2$, for $i = 1, \ldots, n$, and the second one for nodes $(n/2, j)^1$, for $j = 1, \ldots, 8$. Consequently,

$$\mu_m(S^G_{2n,2n}) = \sum_{v \in S^A_{2n,2n}} d_v(d_v + 3)/2 = \frac{23}{24} n^3 + O(n^2) \tag{6.22}$$

$$\mu_s(S^G_{2n,2n}) = \sum_{v \in S^A_{2n,2n}} (d_v + 1) = \frac{7}{4} n^2 + O(n). \tag{6.23}$$

We skip the detailed analysis but just list the results for A, B, and C below:

$$\sum_{v \in S_A} d_v = \sum_{j=3n/2}^{2n-1} j + \sum_{j=2n}^{5n/2-1} j + \sum_{j=2n}^{3n-1} j \tag{6.24}$$

$$\mu_m(S^A_{2n,2n}) = \frac{125}{24} n^3 + O(n^2) \tag{6.25}$$

$$\mu_s(S^A_{2n,2n}) = \frac{9}{2} n^2 + O(n); \tag{6.26}$$

$$\sum_{v \in S_B} d_v = 2 \sum_{j=5n/2}^{3n-1} j + \sum_{j=3n}^{4n-1} j \tag{6.27}$$

$$\mu_m(S^B_{2n,2n}) = \frac{239}{24} n^3 + O(n^2) \tag{6.28}$$

$$\mu_s(S^B_{2n,2n}) = \frac{25}{4} n^2 + O(n); \tag{6.29}$$

$$\sum_{v \in S_C} d_v = 2 \sum_{j=3n}^{7n/2-1} j + \sum_{j=4n}^{5n-1} j \tag{6.30}$$

$$\mu_m(S^C_{2n,2n}) = \frac{371}{24} n^3 + o(n^3) \tag{6.31}$$

$$\mu_s(S^C_{2n,2n}) = \frac{31}{4} n^2 + o(n^2). \tag{6.32}$$

While $S^G_{2n,2n}$ and $S^C_{2n,2n}$ are eliminated in the order of the column nodes $(\cdot, n/2)^2$ followed by the row nodes $(n/2, \cdot)^1$, we eliminate $S^B_{2n,2n}$ in the order of the row nodes $(n/2, \cdot)^1$ followed by the column nodes $(\cdot, n/2)^2$. Moreover, we eliminate $S^A_{2n,2n}$ by

$$\begin{aligned}
&(i, n/2)^2, \text{ for } i = 1, \ldots, n/2,\\
&(n/2, j)^1, \text{ for } i = n/2 + 1, \ldots, n,\\
&(n/2, j)^1, \text{ for } i = 1, \ldots, n/2,\\
&(i, n/2)^2, \text{ for } i = n/2 + 1, \ldots, n,
\end{aligned}$$

in this order. This approach gives a better result.

Given $\mu(S^G_{2n,2n})$, $\mu(S^A_{2n,2n})$, $\mu(S^B_{2n,2n})$, and $\mu(S^C_{2n,2n})$, we can solve from the above recursions (cf. [14]) that the multiplication cost of applying CF to $G_{2n,2n}$ is $943n^3/84+ O(n^2 \log_2 n)$ and the storage cost is $(31/4)n^2 \log_2 n + O(n^2)$. Similar results can be derived for matrices whose adjacency graphs have the form shown in Figure 4.2.

Acknowledgements. We would like to thank Stanley Osher for introducing us to total variation image restoration and for many helpful and stimulating discussions.

REFERENCES

[1] F. Alizadeh and D. Goldfarb, *Second-order cone programming*, Mathematical Programming, 95(1) (2003), pp. 3–51.

[2] F. Alizadeh and S. Schmieta, *Optimization with semidefinite, quadratic and linear constraints*, technical report, RUTCOR, Rutgers University, 1997.

[3] K. Andersen, E. Christiansen, A. Conn, and M. Overton, *An efficient primal-dual interior-point method for minimizing a sum of euclidean norms*, Siam J. Sci. Comput., 22 (2000), pp. 243–262.

[4] A. Chambolle, *An algorithm for total variation minimization and applications*, J. Math. Imaging Vis., 20 (2004), pp. 89–97.

[5] R. Chan, T. Chan, and H.-M. Zhou, *Continuation method for total variation denoising problems*, CAM Report 95-18, UCLA, 1995.

[6] T. Chan and S. Esedoglu, *Aspects of Total Variation Regularized L^1 Function Approximation*, CAM Report 04-07, UCLA, 2004.

[7] T. Chan, G. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, Siam J. Sci. Comput., 20(6) (1999), pp. 1964–1977.

[8] A. George, *Nested dissection of a regular finite-element mesh*, Siam J. Numer. Anal., 10 (1973), pp. 345–363.

[9] D. Goldfarb, S. Osher, and W. Yin, *Cartoon-texture decomposition of images via second-order cone programming*. In preparation.

[10] D. Goldfarb and K. Scheinberg, *Numerically Stable Product-Form Cholesky (LDL^T) Factorization Based Implementations of Interior Point Methods for Second-Order Cone Programming*, CORC Report TR-2003-05, IEOR Dept. Columbia University, 2003.

[11] M. Lysaker, A. Lundervold, and X.-C. Tai, *Noise removal using fourth-order partial differential equations with applications to medical magnetic resonance images in space and time*, IEEE Trans. on Image Processing, 12 (2003), pp. 1579 – 1590.

[12] Y. Meyer, *Oscillating patterns in image processing and nonlinear evolution equations*, vol. 22 of University Lecture Series, AMS, 2002.

[13] S. Osher, A. Sole, and L. Vese, *Image decomposition and restoration using total variation minimization and the H^{-1} norm*, SIAM J. Multiscale Model. Simul., 1(3) (2003), pp. 349–370.

[14] D. Rose and G. Whitten, *A recursive analysis of dissection strategies, Saprse Matrix Computations*, Academic Press, New York, 1976, pp. 59–84.

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

[16] D. Strong and T. Chan, *Exact Solutions to Total Variation Regularization Problems*, CAM Report 96-41, UCLA, 1996.

[17] L. Vese and S. Osher, *Modeling Textures with Total Variation Minimization and Oscillating Patterns in Image Processing*, J. of Scientific Computing, 19(1-3) (2003), pp. 553–572.

[18] C. Vogel and M. Oman, *Iterative methods for total variation denoising*, Siam J. Sci. Comput., 17(1) (1996), pp. 227–238.

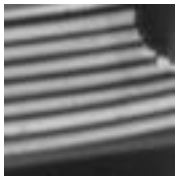

FIG. 6.5.
Original books 64×64

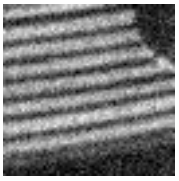

FIG. 6.6.
Noisy books ($\sigma = 20$)

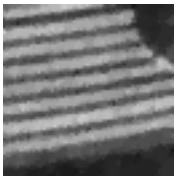

FIG. 6.7.
Donoised image by the SOCP method

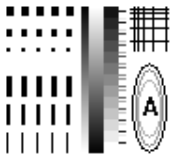

FIG. 6.8.
Original dotplus 100×100

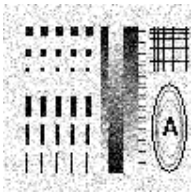

FIG. 6.9.
Noisy dotplus ($\sigma = 40$)

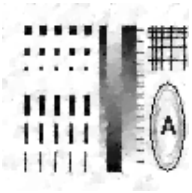

FIG. 6.10.
Donoised image by the SOCP method

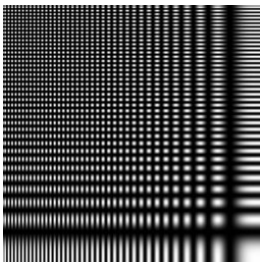

FIG. 6.11.
Original scale 252×252

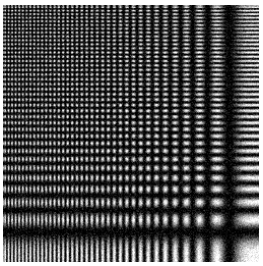

FIG. 6.12.
Noisy scale ($\sigma = 20$)

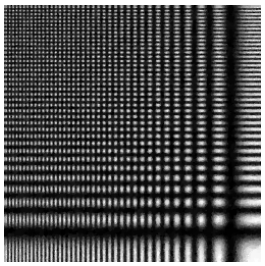

FIG. 6.13.
Donoised image by the SOCP method

FIG. 6.14.
Original Barbara 256×256

FIG. 6.15.
Noisy Barbara ($\sigma = 20$)

FIG. 6.16.
Donoised image by the SOCP method

FIG. 6.17. *Original Barbara* 350×350

FIG. 6.18. *Noisy Barbara* ($\sigma = 20$)

FIG. 6.19. *Donoised image by the SOCP method*

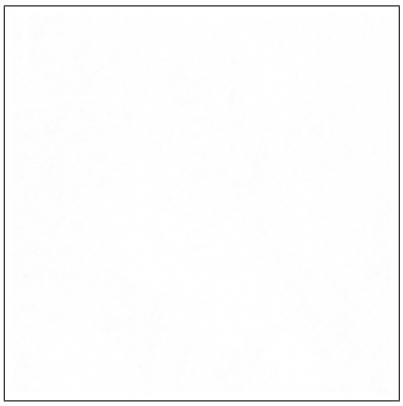

FIG. 6.20. *10 times amplified difference between the results obtained by the SOCP and the PDE methods*

FIG. 6.21.
Noisy image ($\sigma = 20 \times 65^2$)

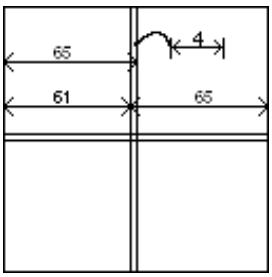

FIG. 6.22.
2×2 domain decomposition applied to image of size 126×126

FIG. 6.23.
2×2 *partitioned result* ($\sigma_{1,1} = \sigma_{1,2} = \sigma_{2,1} = \sigma_{2,2} = 18 \times 65^2$)

FIG. 6.24. *Non-partition result* ($\sigma = 18 \times 65^2$)

FIG. 6.25. 2×2 *partitioned result* ($\sigma_{1,1} = \sigma_{1,2} = \sigma_{2,1} = 18 \times 65^2, \sigma_{2,2} = 16 \times 65^2$)

FIG. 6.26. *10 times amplified difference between Fig. 6.24 and Fig. 6.25*

FIG. 6.27. *Original* 512×512 *image*

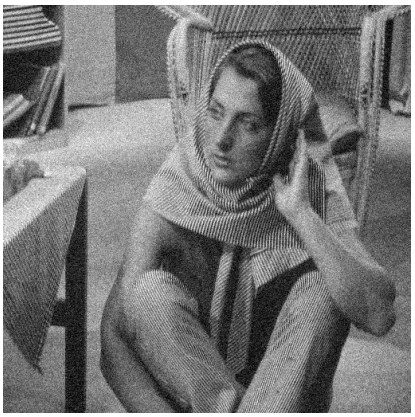

FIG. 6.28. *Noisy* 512×512 *image* ($\sigma = 20$)

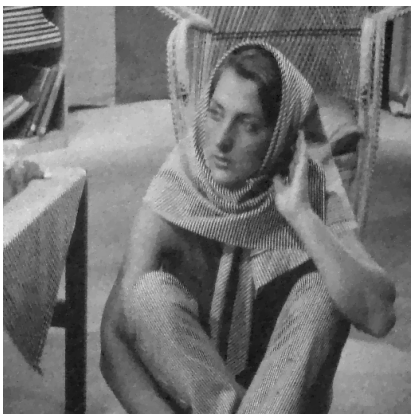

FIG. 6.29. 4×4 *domain decomposition applied to Figure 6.28*

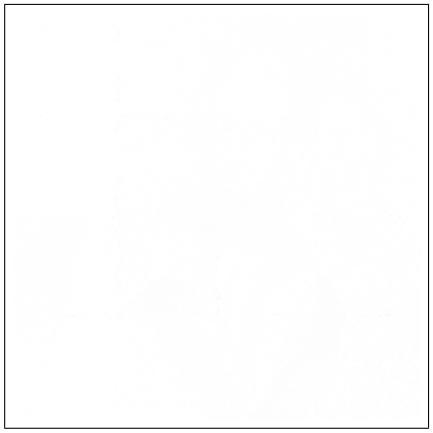

FIG. 6.30. *10 times amplified difference between the results obtained by domain decomposition Figure 6.29 and the non-partitioned one*