J Supercomput (2011) 57:339–356
DOI 10.1007/s11227-010-0417-z

A parameter study of a hybrid Laplacian mean-curvature flow denoising model

Ioana Banicescu · Hyeona Lim · Ricolindo L. Cariño · Seongjai Kim

Published online: 26 March 2010

Abstract This article presents results of a parameter study for a new denoising model, using parallel computing and advanced dynamic load balancing techniques for performance improvement of implementations. A denoising model is suggested hybridizing total variation and Laplacian mean-curvature; the fourth-order model and its numerical procedure introduce a number of parameters. As a preliminary step in the model development, a parameter study has been undertaken in order to discover solitary and interactive effects of the parameters on model accuracy. Such a parameter study is necessarily time-consuming due to the huge number of combinations of the parameter values to be tested. In addition, the study has to be performed on various images, thereby increasing the overall investigation time. The performance of this first parallel implementation of a new hybrid model for image denoising is evaluated when the application is running on heterogeneous environments. The hybrid model is simulated on a general-purpose Linux cluster for which the parallel efficiency exceeds 96%.

Keywords Dynamic load balancing algorithms · Image processing · Denoising

I. Banicescu
Department of Computer Science and Engineering, and Center for Computational Sciences—HPCC, Mississippi State University, Mississippi State, MS 39762, USA
e-mail: ioana@cse.msstate.edu

H. Lim (✉) · S. Kim
Department of Mathematics and Statistics, Mississippi State University, Mississippi State, MS 39762-5921, USA
e-mail: hlim@math.msstate.edu

S. Kim
e-mail: skim@math.msstate.edu

R.L. Cariño
Center for Advanced Vehicular Systems–HPCC, Mississippi State University, Mississippi State, MS 39762, USA
e-mail: rlc@cavs.msstate.edu

1 Introduction

Denoising, or noise reduction, is an important image processing (IP) step for various image-related applications and often necessary as a preprocessing for other imaging tasks such as segmentation and compression. Thus, image denoising methods have occupied a significant position in IP, computer graphics, and their applications [18, 29–31, 37]. Recently, as the field of IP requires higher levels of reliability and efficiency, various powerful tools of partial differential equations (PDEs) and functional analysis have been successfully applied to image restoration [1, 11, 13, 28, 32, 34, 36, 41] and color processing [7, 14, 20, 24, 38]. In particular, a considerable amount of research has been carried out for the theoretical and computational understanding of the total variation (TV) model [36] and its variants [1, 11–13, 22, 23, 26, 28, 29, 39].

In general, most of those denoising models may lose fine structures of the image due to a certain nonphysical dissipation. As remedies, the employment of methods such as Besov norm [29] and iterative refinement [33] have been proposed and studied. However, these new methods are either difficult to minimize utilizing the Euler–Lagrange equation approach or have the tendency to keep an observable amount of noise. Recently, in order to overcome the drawbacks, one of the authors suggested the method of nonflat time evolution (MONTE) [21] and the equalized net diffusion (END) approach [19]. The MONTE and END techniques are applicable to various (conventional) denoising models as either a time-stepping procedure or a variant of mathematical modeling.

As another remedy to the nonphysical dissipation, fourth-order PDE models have emerged [27, 40]. In particular, the Laplacian mean-curvature (LMC) model has been given a particular attention due to its potential capability to preserve edges of linear curvatures. However, it has been numerically verified that the LMC model can easily introduce granule-shaped spots to restored images. To overcome this granularity problem, our efforts have been directed towards studying of a hybrid model which combines a TV-based model with an LMC model. Moreover, detailed optimal choices of algorithm parameters introduced in the hybrid model have been investigated and are discussed in this article.

This paper presents a parameter study for a proposed hybrid TV-LMC model for image denoising using parallel computing and advanced techniques for performance improvement. The hybrid model introduces a number of parameters, highlighting the need for a procedure for selecting optimal parameters that will result in restored images of higher quality. The parameters studied in this paper are described in Sect. 2.3. Preliminary to the development of such a selection procedure, we have undertaken a parameter study of the hybrid model in order to discover the solitary and interactive effects of the parameters on model accuracy. Such a parameter study is necessarily time-consuming due to the huge number of combinations of the parameter values to be tested. In addition, the study has to be performed on a number of different images, thereby increasing the overall investigation time. Thus, the parameter study was implemented as a parallel computing application. The selection of optimal parameters for the hybrid TV-LMC model used for the parallel image denoising application employs advanced dynamic load balancing techniques for performance improvement.

The justification for using these techniques lies in the severe performance degradation of this computationally intensive parallel application, primarily due to load imbalance. The performance evaluation of this first parallel implementation of a new image denoising model is using advanced dynamic scheduling algorithms [2], which are essential in obtaining efficient execution when the application is running on heterogeneous environments.

The rest of this paper is organized as follows. The hybrid model for image denoising is discussed in Sect. 2. The dynamic load balancing tool utilized to parallelize the code for the parameter study is described in Sect. 3. Sample results of performance tests of the parallel implementation are presented in Sect. 4. In Sect. 5, some numerical results are presented using the outputs of the parameter studies with the dynamic load balancing tool. Conclusions of the paper are given in Sect. 6. The Appendix presents a convergence analysis of the iterative algorithm for a linearized LMC model.

2 A hybrid model for image denoising

2.1 The model

We begin with the Laplacian mean-curvature (LMC) model

$$\frac{\partial u}{\partial t} + \Delta\kappa(u) = \beta(f - u), \tag{1}$$

where $\beta \geq 0$, a constraint coefficient, and $\kappa(u)$ denotes the mean-curvature defined as

$$\kappa(u) = \nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right).$$

Here t is an artificial time parameterized for energy descent direction, which is introduced to solve nonlinear partial differential equations more conveniently. In (1), f is an observed image (possibly contaminated by a certain noise), and the solution u represents a denoised (noise-removed) image. The LMC model has a major drawback: granularity. The restored image can easily incorporate granule-shaped spots. The LMC model also shows staircasing, a phenomenon that tends to make the restored image locally constant. However, it is relatively easy to cure.

As a remedy, consider the following hybrid model:

$$\frac{\partial u}{\partial t} - \sigma\,\widetilde{\kappa}(u) + \Delta\widetilde{\kappa}(u) = \beta(f - u), \tag{2}$$

where $\sigma \geq 0$ is a regularization parameter, and

$$\widetilde{\kappa}(u) = |\nabla u|\,\kappa(u) = |\nabla u|\,\nabla \cdot \left(\frac{\nabla u}{|\nabla u|} \right). \tag{3}$$

Here the gradient magnitude $|\nabla u|$ has been incorporated into $\widetilde{\kappa}(u)$, as a scaling factor, in order to reduce staircasing [28]. The second-order term is introduced for (2) to

hold a certain degree of maximum principle, with which the model in turn can eliminate the granularity. In this article, we will call (2) the *generalized LMC* (GLMC) model. In the following subsection, we present an efficient numerical procedure for the GLMC model.

2.2 A numerical procedure

Let Δt be a timestep size, and $t^n = n\Delta t$. Set $u^n = u(\cdot, t^n)$, $n = 0, 1, \ldots,$ with $u^0 = f$. Let $(D_{x_1}, D_{x_2})^{\mathrm{T}}$ and $(D_{2x_1}, D_{2x_2})^{\mathrm{T}}$ be the half-step (regular) central difference and one-step (wider) central difference operators for the gradient ∇, respectively. Assume that the iterates have been computed up to the $(n-1)$th time level. For the computation of the solution in the nth time level, define matrices: for $\ell = 1, 2$ and $m = n-1, n$,

$$\begin{aligned}
\mathcal{K}_\ell u^m &= -\big|\nabla_E u^{n-1}\big| D_{x_\ell}\left(\frac{D_{x_\ell} u^m}{|\nabla_h u^{n-1}|}\right),\\
\mathcal{K}_\ell^2 u^m &= -\big|\nabla_E u^{n-1}\big| D_{2x_\ell}\left(\frac{D_{2x_\ell} u^m}{|\nabla_h u^{n-1}|}\right),\\
\mathcal{K}_\ell^\alpha u^m &= (1-\alpha)\mathcal{K}_\ell\, u^m + \alpha \mathcal{K}_\ell^2\, u^m, \quad \alpha \in [0,1],\\
\mathcal{L}_\ell u^m &= -D_{x_\ell} D_{x_\ell} u^m,
\end{aligned} \tag{4}$$

where $|\nabla_E u^{n-1}|$ and $|\nabla_h u^{n-1}|$ denote appropriate finite difference approximations of $|\nabla u^{n-1}|$. (The above matrices depend on u^{n-1}; however, we did not put the dependence on the matrices for a simpler presentation.) Let

$$\mathcal{K} = \mathcal{K}_1 + \mathcal{K}_2, \quad \mathcal{K}^\alpha = \mathcal{K}_1^\alpha + \mathcal{K}_2^\alpha, \quad \mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2,$$

and

$$\mathcal{D} = \beta + \sigma\mathcal{K}^\alpha + \mathcal{L}\mathcal{K}. \tag{5}$$

Then, a linearized Crank–Nicolson (CN) difference equation for (2) reads

$$\frac{v^n - v^{n-1}}{\Delta t} + \mathcal{D}\frac{v^n + v^{n-1}}{2} = \beta f. \tag{6}$$

Now, let

$$\mathcal{A}_\ell = \frac{\beta}{2} + \sigma\mathcal{K}_\ell^\alpha + \mathcal{L}_\ell\mathcal{K}_\ell, \quad \ell = 1, 2.$$

Since

$$\mathcal{D} = (\mathcal{A}_1 + \mathcal{A}_2) + (\mathcal{L}_1\mathcal{K}_2 + \mathcal{L}_2\mathcal{K}_1),$$

Equation (6) can be rewritten as

$$\begin{aligned}
&\frac{v^n - v^{n-1}}{\Delta t} + (\mathcal{A}_1 + \mathcal{A}_2)\frac{v^n + v^{n-1}}{2}\\
&\quad = \beta f - \frac{1}{2}(\mathcal{L}_1\mathcal{K}_2 + \mathcal{L}_2\mathcal{K}_1)\big(v^n + v^{n-1}\big).
\end{aligned} \tag{7}$$

Thus, replacing the last term in (7) by known values with the error not larger than the truncation error, an alternating direction implicit (ADI) method for (6) can be formulated as [15, 16]

$$\begin{aligned}\left(1+\frac{\Delta t}{2}\mathcal{A}_1\right)w^* &= \left(1-\frac{\Delta t}{2}\mathcal{A}_1-\Delta t\mathcal{A}_2\right)w^{n-1}+\Delta t\beta f\\ &\quad -\frac{\Delta t}{2}(\mathcal{L}_1\mathcal{K}_2+\mathcal{L}_2\mathcal{K}_1)\big(3w^{n-1}-w^{n-2}\big),\\ \left(1+\frac{\Delta t}{2}\mathcal{A}_2\right)w^n &= w^*+\frac{\Delta t}{2}\mathcal{A}_2w^{n-1}.\end{aligned} \tag{8}$$

The quantity w^* is an intermediate solution. In this article, we will call (8) the *Crank–Nicolson ADI* (CN-ADI) method for (2). Each half step of CN-ADI requires inverting a series of quint-diagonal matrices, which is computationally inexpensive.

It is not difficult to see that CN-ADI (8) is a second-order perturbation of the Crank–Nicolson difference equation (6). Indeed, eliminating the intermediate solution w^* from (8), we can obtain

$$\frac{w^n-w^{n-1}}{\Delta t}+\mathcal{D}\frac{w^n+w^{n-1}}{2}=S^{n-1/2}-\big(\mathcal{Q}^n+\mathcal{R}^n\big), \tag{9}$$

where

$$\begin{aligned}\mathcal{Q}^n &= \frac{\Delta t}{4}\mathcal{A}_1\mathcal{A}_2\big(w^n-w^{n-1}\big),\\ \mathcal{R}^n &= -\frac{1}{2}(\mathcal{L}_1\mathcal{K}_2+\mathcal{L}_2\mathcal{K}_1)\big(w^n-2w^{n-1}+w^{n-2}\big).\end{aligned} \tag{10}$$

Here both $\mathcal{Q}^n$ and $\mathcal{R}^n$ are $\mathcal{O}(\Delta t^2)$ for smooth solutions. It should be noticed that the term $\mathcal{Q}^n$ in (10) is the *standard* splitting error of the ADI method [15, 16], while $\mathcal{R}^n$ has arisen during the replacement of the last term in (7) by known values. That is,

$$-\frac{1}{2}(\mathcal{L}_1\mathcal{K}_2+\mathcal{L}_2\mathcal{K}_1)\big(w^n+w^{n-1}\big)-\mathcal{R}^n=-\frac{1}{2}(\mathcal{L}_1\mathcal{K}_2+\mathcal{L}_2\mathcal{K}_1)\big(3w^{n-1}-w^{n-2}\big). \tag{11}$$

Note that the CN-ADI algorithm (8) is a three-step procedure, defined well for $n\geq 2$. For $n=1$, one may conveniently assume $w^{-1}=w^0\,(=f)$. Assuming the model is linear, the CN-ADI algorithm is analyzed for stability and convergence in the Appendix.

2.3 Algorithm parameters

In addition to having the three model parameters (β, σ, and α), the CN-ADI iteration involves two extra algorithm parameters, Δt and n. The denoising computation must stop after an appropriate number of iterations. However, it is hard to analytically determine the right time to stop. Thus finding an appropriate iteration count, n, is an important problem. In practice, it is often the case that one needs to run the algorithm a few times to select the best result by comparison. The timestep size Δt is also

```
Establish uncontaminated image g
Add Gaussian noise to g to produce contaminated image f
Establish parameter counts (N)_β, (N)_σ, (N)_α, (N)_Δt, (N)_n
Establish parameter values β[1], ..., β[(N)_β];   σ[1], ..., σ[(N)_σ];
        α[1], ..., α[(N)_α];   Δt[1], ..., Δt[(N)_Δt];   n[1], ..., n[(N)_n]
For each combination of β, σ, α, Δt, n values
    Apply denoising procedure on f to produce restored image u
    Calculate PSNR; output β, σ, α, Δt, n and PSNR
End for
```

Fig. 1 High-level outline of parameter study

important for effectiveness of the algorithm and the quality of resulting images as well.

Thus, for a given contaminated image, values have to be selected for the algorithm parameters $\beta, \sigma, \alpha, \Delta t$, and n which result in the best restored image. However, when the original uncontaminated image is not known, assessing the quality of the restored image is difficult, if not impossible. In order to gain insight on the influence of algorithm parameters on the quality of the restored image, we simulate the proposed denoising model on known images with synthetically added noise. This simulation is discussed in the next section.

2.4 Parameter study

Preliminary to the development of a procedure for selecting parameters for the proposed hybrid model described in the previous section, we simulated the model by applying it to restore known images with artificial Gaussian noise. As a measure of the quality of the restored images, we adopted the peak signal-to-noise ratio (PSNR) defined as

$$\mathrm{PSNR} \equiv 10 \log_{10}\left(\frac{\sum_{ij} 255^2}{\sum_{ij} (g_{ij} - u_{ij})^2} \right) \mathrm{dB},$$

where g denotes the original uncontaminated image, and u is the restored image.

In order to gain insight on the influence of the parameters $\beta, \sigma, \alpha, \Delta t$, and n on PSNR, we conducted a parameter study, following the pseudocode in Fig. 1. Various plots from the outputs of the study could be produced, including animations of PSNR as functions of β, σ, α, with either Δt or n fixed and using the other as the variable for the animation.

The number of combinations of parameter values is simply $(\mathrm{N})_\beta \times (\mathrm{N})_\sigma \times (\mathrm{N})_\alpha \times (\mathrm{N})_{\Delta t} \times (\mathrm{N})_n$, which could be huge even for small to moderate values of the parameter counts. Fortunately, the denoising procedure can be computed simultaneously for several combinations of the parameters, on a parallel machine. However, the denoising procedure performs nonuniform amounts of computations for each parameter combination; therefore, dynamic load balancing is necessary for efficient utilization of the parallel machine. For the parameter study, we used a dynamic load balancing

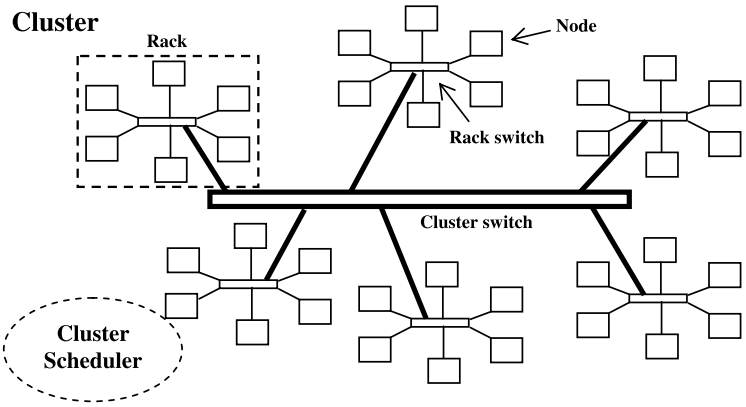

Fig. 2 A popular interconnection network for clusters

tool we developed to parallelize serial codes with a structure similar to Fig. 1. This tool is described in the next section.

3 Dynamic load balancing tool

We describe in this section a dynamic load balancing tool we developed to simplify the parallelization and load balancing of applications that contain computationally intensive parallel loops (like in Fig. 1) on message-passing clusters. These clusters are usually organized into racks that are connected by a cluster switch, each rack consisting of a number of nodes connected by a rack switch, and each node containing one or more processors. Figure 2 illustrates a popular interconnection configuration. Heterogeneity is inherent in such a cluster, more so if it was constructed incrementally over a period of time, because the processors would have different capabilities. Rates of communication between processors are also variable. Typically, the cluster scheduler attempts to assign nodes from a single rack to a job for efficient communications. Even with excellent job scheduling algorithms, the scattering of processors across a number of racks occurs with a high probability, especially for jobs that request large numbers of processors. Thus, applications running on clusters typically need to incorporate load balancing for highest possible performance.

The tool we developed can be integrated into existing sequential applications with minimal code changes. The tool is a simplified version of the dynamic load balancing tool based on MPI described in [10]. In a parallel loop, nonuniformity of the iterate execution times or heterogeneity of the processors usually give rise to load imbalance. If the number of iterates `N` is significantly larger than the number of processors `P`, load balancing through dynamic loop scheduling will be appropriate. Figure 3 illustrates the modification of a Fortran 90 program containing a parallel loop to integrate the tool. Only a few lines are added; essentially, the original `do` loop is converted to a `while` loop where chunks of iterates can be executed concurrently on different processors. Since the `i-iterate` invokes CPU-intensive computations, which may

```
program Serial_Version
  ...
  do i=1,N
    ... i-iterate
  end do
  ...

program Parallel_Version_With_Load_Balancing
  use DLS
  include 'mpif.h'
  type (infoDLS) info
  integer method, iStart, iSize, iIters, mpierr
  double precision iTime
  call MPI_Init (mpierr)
  ...
  method = (choice of loop scheduling technique)
  call DLS_Setup (MPI_COMM_WORLD, info)
  call DLS_StartLoop (info, 1, N, method)
  do while ( .not. DLS_Terminated(info) )
    call DLS_StartChunk (info, iStart, iSize)
    do i=iStart, iStart+iSize-1 ! was i=1,N
      ... i-iterate
    end do
    call DLS_EndChunk (info)
  end do
  call DLS_EndLoop (info, iIters, iTime)
  ...
```

Fig. 3 Parallelization with dynamic load balancing of a Fortran 90 program containing a parallel loop

be expressed in hundreds or thousands of lines of code, the additional code to integrate the tool constitutes a tiny percentage of the total number of lines of code for the application.

The module `DLS` (abbreviation for Dynamic Loop Scheduling) contains the type definition of `infoDLS` and the codes for the `DLS_*` routines. Based on the Message-Passing Interface (MPI) library, the routines implement a scheduler–worker strategy of load balancing, where the scheduler participates in executing loads, in addition to being responsible for assigning loads. The DLS routines are briefly described as follows:

`DLS_Setup(MPI_COMM_WORLD,info)` initializes a dynamic loop scheduling environment on `MPI_COMM_WORLD`. Information about this environment is maintained in the data structure `info`.

`DLS_StartLoop(info,1,N,method)` is the synchronization point for the start of loop execution. `(1,N)` is the loop range, and `method` is a user-specified index for the selected loop scheduling technique. The following techniques are implemented: static scheduling, a modification of fixed size chunking [25], guided

self-scheduling [35], factoring [17], variants of adaptive weighted factoring [5, 6, 8, 9], and adaptive factoring [3, 4].
`DLS_Terminated(info)` returns true if all loop iterates have been executed.
`DLS_StartChunk(info,iStart,iSize)` returns a range for a chunk of iterates. This range starts with iterate `iStart` and contains `iSize` iterates.
`DLS_EndChunk(info)` signals the end of execution of a chunk of iterates. Internally, a worker processor requests its next chunk from the scheduler.
`DLS_EndLoop(info,iIters,iTime)` is the synchronization point at the end loop execution. `iIters` is the number of iterates done by the calling processor, and `iTime` is the cost (in seconds) measured using MPI_Wtime(). `iIters` and `iTime` are useful for assessing the performance gains achieved by dynamic load balancing. For example, the sum of the `iTimes` from all participating processors gives an estimate of the cost of executing the loop on a single processor.

After loop execution, the results of the computations (in `i-iterate`) will be distributed among the participating processors. A reduction operation like MPI_Reduce() may be necessary to collect the results in one processor, or MPI_Allreduce to make the results available to all processors in `MPI_COMM_WORLD`. This would be the responsibility of the user, since DLS only manipulates the indices of the loop. Information about the mapping of the chunks of iterates to processors is maintained in the `chunkMap` component of the `infoDLS` structure.

The performance of the code for the parameter study of the denoising procedure with the dynamic load balancing tool on a general-purpose Linux cluster is described in the next section.

4 Parallel performance

We conducted preliminary parameter studies for a number of images, considering $(\mathrm{N})_\beta = 9$, $(\mathrm{N})_\sigma = 9$, $(\mathrm{N})_\alpha = 9$, $(\mathrm{N})_{\Delta t} = 9$, and $(\mathrm{N})_n = 15$ for a total of 98,415 parameter combinations. The dynamic scheduling algorithm selected from the dynamic load balancing tool to improve the performance of our parallel application via load balancing was the adaptive factoring [3, 4].

The studies were executed on the heterogeneous general-purpose EMPIRE cluster of the Mississippi State University. The cluster can be abstracted as in Fig. 2 and has a total of 1038 processors. A rack contains 32 nodes of dual 1.0 GHz or 1.266 GHz Pentium III processors and 1.25 GB RAM. Each node is connected to a 100 Mb/s Ethernet rack switch. The rack switches are connected by a gigabyte Ethernet cluster switch. Installed software includes RedHat Linux and PBS. The general submission queue allows 64-processor, 48-hour jobs; a special queue allows 128-processor, 96-hour jobs from a restricted set of users. According to the Top 500 Supercomputer Sites list published in June 2002, EMPIRE then was the 126th fastest computer in the world and the 10th among educational institutions in the U.S.

The experimental study considered a submission of jobs to a 32-processor EMPIRE cluster, and the experimental results from running five jobs have been averaged and recorded to increase the confidence in the statistics gathered. The cluster scheduler assigned homogeneous processors to the jobs. Since other jobs were also

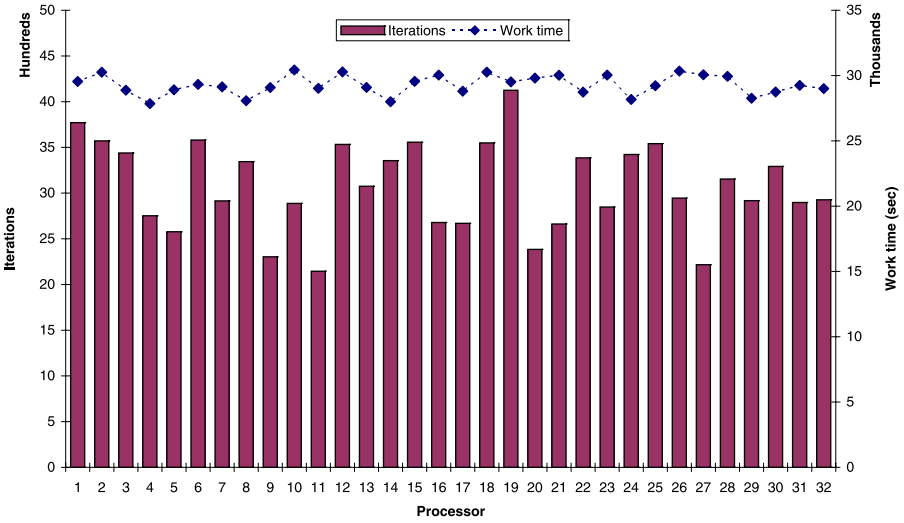

Fig. 4 Distribution of iterations and work times for the parameter study on the image *LenaGray256*

executing on the cluster along with the study, the contention for network resources was a source of system-induced load imbalance. However, the major source of load imbalance was the nonuniform amount of computations required by the denoising procedure for different sets of parameter values.

Figure 4 gives a summary of the performance of the dynamic load balancing parallel code for the parameter study on the image *LenaGray256*. The left axis (for the bars) denotes the number of iterations of the loop in Fig. 1 executed by a processor, while the right axis (for the diamonds) denotes the time in seconds taken by the processor to execute the iterations. The large differences in the number of iterations executed by the processors is evidence for application-induced load imbalance. However, the difference between the maximum and minimum work times is only 2581.3 seconds, which is a relatively narrow range, given that the job time measured by the cluster scheduler was 8.453 hours. An estimate of the sequential cost of the study is 260.4547 hours (∼10.9 days), which is the sum of the work times of the 32 processors. Thus, an estimate of the efficiency is (estimated sequential cost)/(parallel cost) $= (260.4547)/(32 \times 8.453) = 0.963$. The high efficiency indicates that the dynamic load balancing tool successfully addressed the performance degradation due to load imbalance.

Figure 5 gives the summary for the parameter study on the image Black-Circle. The cluster scheduler assigned homogeneous processors to the study, and the job time was 39.546 hours. The differences in iteration counts are significant, indicating the presence of application-induced load imbalance. An estimate of the sequential cost is 1,223.279 hours (∼51 days), which is the sum of the work times of the 32 processors. Thus, an estimate of the efficiency is (estimated sequential cost)/(parallel cost) $= (1223.279)/(32 \times 39.546) = 0.967$.

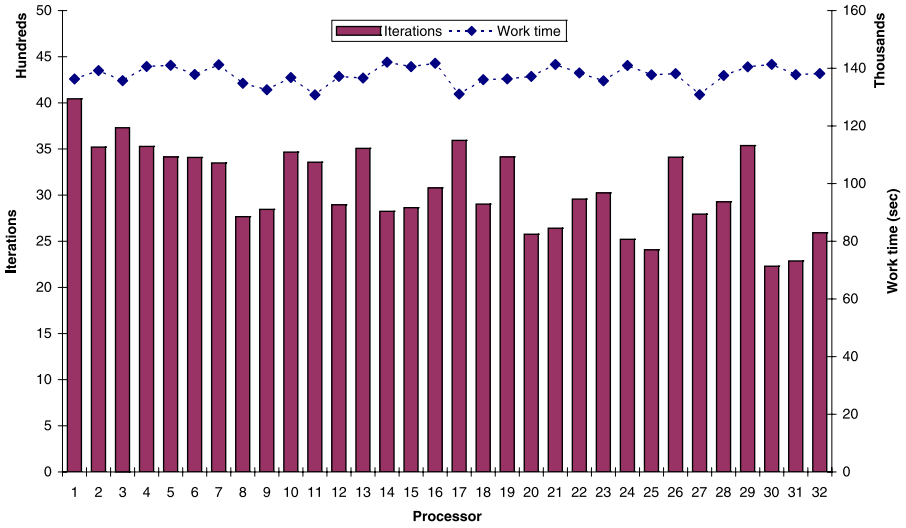

Fig. 5 Distribution of iterations and work times for the parameter study on the image *BlackCircle*

5 Numerical experiments

In this section, we present some numerical examples using the parameter studies with the dynamic load balancing tool explained in the previous sections. The image *LenaGray256*, Fig. 6(a), was contaminated by a Gaussian noise of PSNR = 24.8 as in Fig. 6(b) and restored by using the GLMC model (2). Among the 98,415 parameter combinations, the image *LenaGray256* yielded the best PSNR value (= 30.4) for the following two combinations:

$$\begin{aligned} &\beta = 3.1, \quad \sigma = 0.6, \quad \alpha = 0.1, \quad \Delta t = 0.03, \quad n = 14; \\ &\beta = 2.5, \quad \sigma = 0.3, \quad \alpha = 0.3, \quad \Delta t = 0.02, \quad n = 14. \end{aligned} \tag{12}$$

Figure 6(c) is the restored image using the first combination of parameters in (12).

For a comparison with the GLMC model, the parameter studies with the dynamic load balancing tool have been also conducted for the conventional TV model

$$\frac{\partial u}{\partial t} - \sigma\, \widetilde{\kappa}(u) = \beta(f - u), \tag{13}$$

where $\sigma = 1$ and $\alpha = 0$. In this case, it reduces to having the three algorithmic parameters β, Δt, and n, leading to a total of 1,215 parameter combinations ($(\mathrm{N})_\beta = 9$, $(\mathrm{N})_{\Delta t} = 9$, and $(\mathrm{N})_n = 15$). Figure 7 presents restored images for the image *Elaine* carried out by the conventional TV model and the new GLMC model. The noisy image in Fig. 7(b) contains a Gaussian noise of PSNR = 24.8. The best PSNR values using the GLMC model and TV model are 30.8 (200 combinations) and 30.5 (17

Fig. 6 *LenaGray256*: (**a**) The original image g, (**b**) a noisy image f with a Gaussian noise (PSNR 24.8), and (**c**) restored image u (PSNR 30.4) by using the GLMC model

combinations), respectively. Among them, the following combinations were chosen:

$$\begin{aligned} &\text{TV model: } \beta = 0.4, \quad \Delta t = 0.09, \quad n = 18; \\ &\text{GLMC model: } \beta = 1.9, \quad \sigma = 0.5, \quad \alpha = 0.6, \quad \Delta t = 0.03, \quad n = 18. \end{aligned} \tag{14}$$

TV model introduces some nonphysical dissipation to lose fine structures as in Fig. 7(c). One can see from Fig. 7(d) that the GLMC model outperforms the TV model. It preserves edges of the image more effectively.

In Table 1, we compare the best PSNR values, obtained from the parameter studies with the dynamic load balancing tool, for the two models. We select three additional images *Circle*, *Clock*, and *Pepper* in Fig. 8, in addition to the ones dealt in Figs. 6 and 7. For denoising, the images were also contaminated by the same amount of Gaussian noise of PSNR = 24.8. As shown in the table, the GLMC model has obtained the better PSNR values than the TV model for all cases. Here the differences in PSNR values are small for the selected images. However, one should notice that the GLMC model is designed to converge to locally linear images. The GLMC model must result in better restored images than the TV model, for most cases. The performance of GLMC can be improved by setting some of the parameters adaptively, particularly incorporating information on fine structures such as edges and textures.

Optimal parameters may differ for different images. However, an optimal set of parameters can be used for large number of images in the same class. For example, one set of parameters used for one MRI image can be used for other MRI images.

6 Conclusions

We have performed a parameter study for a new hybrid model for image restoration. The hybrid model, which is based on the total variation model and the Laplacian mean-curvature model, contains a number of parameters that need to be fine-tuned in order to produce images of high quality. To guide the development of a procedure for selecting the parameters, we have applied the hybrid model for the restoration of known images that were contaminated with Gaussian noise. The simulations have

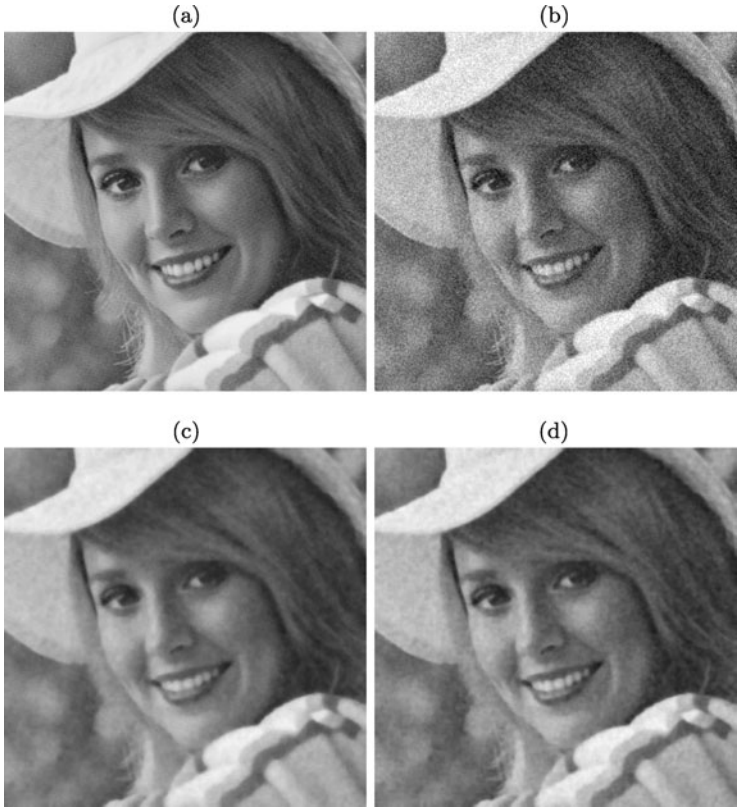

Fig. 7 *Elaine*: (**a**) The original image g, (**b**) a noisy image f with a Gaussian noise (PSNR 24.8), and restored images u by using (**c**) TV (PSNR 30.5) and (**d**) GLMC model (PSNR 30.8)

Table 1 PSNR analysis

	LenaGray256	*Elaine*	*Circle*	*Clock*	*Pepper*
TV	30.1	30.5	36.1	29.9	32.1
GLMC	30.4	30.8	36.4	30.2	32.3

Fig. 8 Additional images utilized for parameter studies: *Circle*, *Clock*, and *Pepper*

been conducted on a general-purpose heterogeneous Linux cluster. To address the load imbalance that potentially arises from algorithmic and systemic sources, we have utilized a dynamic load balancing tool in the simulation code. The simulations have achieved very high estimated efficiencies. From the experiments, the hybrid model has shown a possibility of outperforming the conventional TV model.

Acknowledgements This work was partially supported by the following National Science Foundation grants: NSF ACI-9984465, NSF EPS-0556308, NSF EPS-0132618, NSF DMS-0609815.

Appendix: Convergence analysis in linear case

In this section, we would like to understand the convergence behavior of CN-ADI (8). For an easier analysis, we consider the linear case in (2); that is, the model is assumed to incorporate

$$\widetilde{\kappa}(u) = |\nabla \widetilde{u_0}| \, \nabla \cdot \left(\frac{\nabla u}{|\nabla \widetilde{u_0}|} \right), \tag{15}$$

where $\widetilde{u_0}$ is a smoothed image of f. In practice, the model (2) has performed very similarly when (3) is replaced by (15).

Let

$$e^n = u^n - w^n,$$

where $u^n = u(\cdot, t^n)$, the true solution, and w^n denotes the solution of CN-ADI at $t = t^n$. Then, it follows from (2) and (9) that

$$\begin{aligned} &\frac{e^n - e^{n-1}}{\Delta t} + \mathcal{D}\frac{e^n + e^{n-1}}{2} + \Delta t \, \widehat{\mathcal{Q}}\left(e^n - e^{n-1}\right) \\ &\quad + \widehat{\mathcal{R}}\left(e^n - 2e^{n-1} + e^{n-2}\right) = \delta^{n-1/2}, \end{aligned} \tag{16}$$

where

$$\widehat{\mathcal{Q}} = \frac{1}{4}\mathcal{A}_1\mathcal{A}_2, \qquad \widehat{\mathcal{R}} = -\frac{1}{2}(\mathcal{L}_1\mathcal{K}_2 + \mathcal{L}_2\mathcal{K}_1), \tag{17}$$

and $\delta^{n-1/2}$ is the truncation error, which is $\mathcal{O}(|\Delta \mathbf{x}|^2 + \Delta t^2)$ for linear problems.

Define

$$\bar{\partial}_t e^n = \frac{e^n - e^{n-1}}{\Delta t}$$

and rewrite (16) as

$$\left(1 + \Delta t^2 \widehat{\mathcal{Q}}\right)\bar{\partial}_t e^n + \frac{\mathcal{D}}{2}\left(e^n + e^{n-1}\right) + \Delta t^2 \, \widehat{\mathcal{R}} \, \bar{\partial}_t^2 e^n = \delta^{n-1/2}. \tag{18}$$

Multiply this equation by the test function $\bar{\partial}_t e^n$, multiply the result by Δt, and sum from the j_0th time level to the nth time level, to have

$$\sum_{j=j_0}^{n}\big(\big[1+\Delta t^2\widehat{\mathcal{Q}}\big]\bar{\partial}_t e^j,\ \bar{\partial}_t e^j\big)\Delta t+\frac{1}{2}\big(\mathcal{D}e^n,\ e^n\big)+\Delta t^2\sum_{j=j_0}^{n}\big(\widehat{\mathcal{R}}\,\bar{\partial}_t^2 e^j,\ \bar{\partial}_t e^j\big)\Delta t$$
$$=\frac{1}{2}\big(\mathcal{D}e^{j_0-1},\ e^{j_0-1}\big)+\sum_{j=j_0}^{n}\big(\delta^{j-1/2},\ \bar{\partial}_t e^j\big)\Delta t,\quad j_0\ge 1. \tag{19}$$

Utilizing the inequality $a^2-ab\ge(a^2-b^2)/2$, one can obtain

$$\begin{aligned}\sum_{j=j_0}^{n}\big(\widehat{\mathcal{R}}\,\bar{\partial}_t^2 e^j,\ \bar{\partial}_t e^j\big)\Delta t&=\sum_{j=j_0}^{n}\big(\widehat{\mathcal{R}}\big[\bar{\partial}_t e^j-\bar{\partial}_t e^{j-1}\big],\ \bar{\partial}_t e^j\big)\\&\ge\frac{1}{2}\sum_{j=j_0}^{n}\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^j,\ \bar{\partial}_t e^j\big)-\frac{1}{2}\sum_{j=j_0}^{n}\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^{j-1},\ \bar{\partial}_t e^{j-1}\big)\\&=\frac{1}{2}\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^n,\ \bar{\partial}_t e^n\big)-\frac{1}{2}\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^{j_0-1},\ \bar{\partial}_t e^{j_0-1}\big).\end{aligned} \tag{20}$$

We apply the Cauchy–Schwartz inequality once more to the term including $\delta^{j-1/2}$ in (19), as

$$\sum_{j=j_0}^{n}\big(\delta^{j-1/2},\ \bar{\partial}_t e^j\big)\Delta t\le\frac{1}{2}\sum_{j=j_0}^{n}\big(\delta^{j-1/2},\ \delta^{j-1/2}\big)\Delta t+\frac{1}{2}\sum_{j=j_0}^{n}\big(\bar{\partial}_t e^j,\ \bar{\partial}_t e^j\big)\Delta t. \tag{21}$$

Thus, from (19), (20), and (21) we can have

$$\begin{aligned}&\sum_{j=j_0}^{n}\big(\big[1+2\Delta t^2\widehat{\mathcal{Q}}\big]\bar{\partial}_t e^j,\ \bar{\partial}_t e^j\big)\Delta t+\big(\mathcal{D}e^n,\ e^n\big)+\Delta t^2\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^n,\ \bar{\partial}_t e^n\big)\\&\quad\le\big(\mathcal{D}e^{j_0-1},\ e^{j_0-1}\big)+\Delta t^2\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^{j_0-1},\ \bar{\partial}_t e^{j_0-1}\big)\\&\qquad+\sum_{j=j_0}^{n}\big\|\delta^{j-1/2}\big\|^2\Delta t,\quad j_0\ge 1.\end{aligned} \tag{22}$$

When $u^{-1}=u^0\,(=f)$, it is natural to set $w^{-1}=w^0=f$. In this case, choosing $j_0=1$ in (22) gives

$$\begin{aligned}&\sum_{j=1}^{n}\big(\big[1+2\Delta t^2\widehat{\mathcal{Q}}\big]\bar{\partial}_t e^j,\ \bar{\partial}_t e^j\big)\Delta t+\big(\mathcal{D}e^n,\ e^n\big)+\Delta t^2\big(\widehat{\mathcal{R}}\,\bar{\partial}_t e^n,\ \bar{\partial}_t e^n\big)\\&\quad\le\sum_{j=1}^{n}\big\|\delta^{j-1/2}\big\|^2\Delta t,\quad n\ge 1.\end{aligned} \tag{23}$$

Otherwise, we set $j_0 = 2$. Then, (22) reads

$$\sum_{j=2}^{n}\left(\left[1+2\Delta t^2\widehat{\mathcal{Q}}\right]\bar{\partial}_t e^j,\ \bar{\partial}_t e^j\right)\Delta t+\left(\mathcal{D}e^n,\ e^n\right)+\Delta t^2\left(\widehat{\mathcal{R}}\,\bar{\partial}_t e^n,\ \bar{\partial}_t e^n\right)$$

$$\leq\left(\mathcal{D}e^1,\ e^1\right)+\Delta t^2\left(\widehat{\mathcal{R}}\,\bar{\partial}_t e^1,\ \bar{\partial}_t e^1\right)+\sum_{j=2}^{n}\left\|\delta^{j-1/2}\right\|^2\Delta t,\quad n\geq 2. \tag{24}$$

Thus it is important to compute w^1 accurately when w^{-1} is not available.

Here it should be noticed that $\widehat{\mathcal{R}}$ may not be a nonnegative operator. Thus, a sufficient condition for the stability of CN-ADI reads

$$1+2\Delta t^2\widehat{\mathcal{Q}}+\Delta t\,\widehat{\mathcal{R}}=1-\frac{\Delta t}{2}(\mathcal{L}_1\mathcal{K}_2+\mathcal{L}_2\mathcal{K}_1)+\frac{\Delta t^2}{2}\mathcal{A}_1\mathcal{A}_2\geq 0. \tag{25}$$

However, $\mathcal{A}_1\mathcal{A}_2$ is nonnegative, and each of $\mathcal{L}_1$, $\mathcal{L}_2$, $\mathcal{K}_1$, and $\mathcal{K}_2$ is bounded by 4 in its matrix norm.

We summary the above analysis as follows:

Theorem *Consider the linear model* (2) *incorporating* (15). *Its CN-ADI algorithm* (8) *is stable*, *independently of* $\sigma\geq 0$ *and* $\alpha\in[0,1]$, *when*

$$\Delta t\leq\frac{1}{16}. \tag{26}$$

In this case, *the error converges in* $\mathcal{O}(\Delta t^2+h^2)$.

References

1. Alvarez L, Lions PL, Morel M (1992) Image selective smoothing and edge detection by nonlinear diffusion. II. SIAM J Numer Anal 29:845–866
2. Banicescu I, Cariño RL (2005) Addressing the stochastic nature of scientific computations via dynamic loop scheduling. Electron J Trans Numer Analysis 21:66–80
3. Banicescu I, Liu Z (2000) Adaptive factoring: A dynamic scheduling method tuned to the rate of weight changes. In: Proceedings of the high performance computing symposium (HPC 2000), pp 122–129
4. Banicescu I, Velusamy V (2002) Load balancing highly irregular computations with the adaptive factoring. In: Proceedings of the 16th IEEE international parallel and distributed processing symposium—11th heterogeneous computing workshop (IPDPS-HCW 2002), IEEE Computer Society Press
5. Banicescu I, Velusamy V (2001) Performance of scheduling scientific applications with adaptive weighted factoring. In: Proceedings of the 15th IEEE international parallel and distributed processing symposium—10th heterogenous computing workshop (IPDPS-HCW 2001) CDROM, IEEE Computer Society Press, April 2001
6. Banicescu I, Velusamy V, Devaprasad J (2003) On the scalability of dynamic scheduling scientific applications with adaptive weighted factoring. Clust Comput, J Networks Softw Tools Appl 6(3):215–226
7. Blomgren PV, Chan TF (1998) Color TV: Total variation methods for restoration of vector valued images. IEEE Trans Image Process 7(3):304–309

8. Cariño RL, Banicescu I (2002) Load balancing parallel loops on message-passing systems. In: Akl SG, Gonzales T (eds) Proceedings of the 14th IASTED international conference on parallel and distributed computing and systems (PDCS 2004). ACTA Press, Calgary, pp 362–367
9. Cariño RL, Banicescu I (2002) Dynamic scheduling parallel loops with variable iterate execution times. In: Proceedings of the 16th IEEE international parallel and distributed processing symposium—3rd workshop on parallel and distributed scientific and engineering computing with applications (IPDPS-PDSECA 2002) CDROM, IEEE Computer Society Press
10. Cariño RL, Banicescu I (2005) A load balancing tool for distributed parallel loops. Clust Comput 8(4):313–321
11. Catte F, Lions PL, Morel M, Coll T (1992) Image selective smoothing and edge detection by nonlinear diffusion. SIAM J Numer Anal 29:182–193
12. Cha Y, Kim S (2006) Edge-forming methods for color image zooming. IEEE Trans Image Process 15(8):2315–2323
13. Chan T, Osher S, Shen J (1999) The digital TV filter and nonlinear denoising. Technical report #99-34, Department of Mathematics, University of California, Los Angeles, CA 90095-1555, October 1999
14. Chan TF, Shen J (2000) Variational restoration of non-flat image features: Models and algorithms. SIAM J Appl Math 61(4):1338–1361
15. Douglas J Jr, Gunn JE (1964) A general formulation of alternating direction methods Part I. Parabolic and hyperbolic problems. Numer Math 6:428–453
16. Douglas J Jr, Kim S (2001) Improved accuracy for locally one-dimensional methods for parabolic equations. Math Models Methods Appl Sci 11:1563–1579
17. Flynn-Hummel S, Schonberg E, Flynn LE (1992) Factoring: A method for scheduling parallel loops. Commun ACM 35(8):90–101
18. Gonzalez RC, Woods RE (2002) Digital image processing, 2nd edn. Prentice-Hall, Upper Saddle River
19. Kim S (2006) Image denoising via diffusion modulation. Int J Pure Appl Math 30(1):71–92
20. Kim S (2006) PDE-based image restoration: A hybrid model and color image denoising. IEEE Trans Image Process 15(5):1163–1170
21. Kim S, Kwon S-H (2006) Explicit nonflat time evolution for PDE-based image restoration. Lect Notes Comput Sci 4338:35–44
22. Kim S, Lim H (2007) A non-convex diffusion model for simultaneous image denoising and edge enhancement. Electron J Differ Equ 15:175–192
23. Kim S, Lim H (2009) Fourth-order partial differential equations for effective image denoising. Electron J Differ Equ 17:107–121
24. Kimmel R, Sochen N (2002) Orientation diffusion or how to comb a porcupine? J Visual Comm Image Represent 13:238–248. Special issue on PDEs in Image Processing, Computer Vision, and Computer Graphics
25. Kruskal CP, Weiss A (1985) Allocating independent subtasks on parallel processors. IEEE Trans Softw Eng 11(10):1001–1016
26. Lim H, Williams TN (2007) A non-standard anisotropic diffusion for speckle noise removal. J System Cyber Inf 5(2):12–17
27. Lysaker M, Lundervold A, Tai X-C (2003) Noise removal using fourth-order partial differential equations with applications to medical magnetic resonance images in space and time. IEEE Trans Image Process 12(12):1579–1590
28. Marquina A, Osher S (2000) Explicit algorithms for a new time dependent model based on level set motion for nonlinear deblurring and noise removal. SIAM J Sci Comput 22:387–405
29. Meyer Y (2001) Oscillating patterns in image processing and nonlinear evolution equations. University lecture series, vol 22. American Mathematical Society, Providence
30. Mitra SK, Sicuranza GL (2001) Nonlinear image processing. Academic Press, San Diego
31. Morel J-M, Solimini S (1995) Variational methods in image segmentation. Progress in nonlinear differential equations and their applications, vol 14. Birkhäuser, Boston
32. Nitzberg M, Shiota T (1992) Nonlinear image filtering with edge and corner enhancement. IEEE Trans Pattern Anal Mach Intell 14:826–833
33. Osher S, Burger M, Goldfarb D, Xu J, Yin W (2004) Using geometry and iterated refinement for inverse problems (1): Total variation based image restoration. CAM Report #04-13, Department of Mathematics, UCLA, LA, CA 90095
34. Perona P, Malik J (1990) Scale-space and edge detection using anisotropic diffusion. IEEE Trans Pattern Anal Mach Intell 12:629–639

35. Polychronopoulos CD, Kuck DJ (1987) Guided self-scheduling: A practical scheduling scheme for parallel supercomputers. IEEE Trans Comput 36(12):1425–1439
36. Rudin L, Osher S, Fatemi E (1992) Nonlinear total variation based noise removal algorithms. Physica D 60:259–268
37. Sapiro G (2001) Geometric partial differential equations and image analysis. Cambridge University Press, Cambridge
38. Sochen N, Kimmel R, Malladi R (1998) A general framework for low level vision. IEEE Trans Image Process 7(3):310–318
39. Vese LA, Osher SJ (2002) Numerical methods for p-harmonic flows and applications to image processing. SIAM J Numer Anal 40(6)
40. You Y, Kaveh M (2000) Fourth-order partial differential equations for noise removal. IEEE Trans Image Process 9(10):1723–1730
41. You Y, Xu W, Tannenbaum A, Kaveh M (1996) Behavioral analysis of anisotropic diffusion in image processing. IEEE Trans Image Process 5:1539–1553