Simulation of a Hybrid Model for Image Denoising

Ricolindo L. Cariño[1], Ioana Banicescu[2,1], Hyeona Lim[3],
Neil Williams[3] and Seongjai Kim[3]

[1]Center for Computational Sciences
Mississippi State University
Mississippi State, MS 39762-9627
rlc@erc.msstate.edu

[2]Dept. of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762-9637
ioana@cse.msstate.edu

[3]Dept. of Mathematics and Statistics
Mississippi State University
Mississippi State, MS 39762-5921
{hlim, skim}@math.msstate.edu, tnw7@msstate.edu

Abstract

We propose a new model for image denoising which is a hybrid of the total variation model and the Laplacian mean-curvature model. An efficient numerical procedure to compute the hybrid model is also presented. The hybrid model and its computational procedure introduce a number of parameters. As a preliminary step to the synthesis of a method for selecting optimal parameters, the hybrid model was simulated on a number of known images with synthetically added noise. The parallel simulation code was easily composed from existing serial code and a dynamic load balancing tool. The estimated parallel efficiency of the simulation is in excess of 96% on 32 processors of a general-purpose Linux cluster.

1. Introduction

Denoising is an important image processing (IP) step for various image-related applications and often necessary as a pre-processing for other imaging tasks such as segmentation and compression. Thus image denoising methods have occupied a peculiar position in IP, computer graphics, and their applications [18, 28, 29, 30, 36]. 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, 10, 14, 21, 27, 31, 33, 35, 41] and color processing [6, 15, 22, 24, 37]. In particular, a considerable amount of research has been carried out for the theoretical and computational understanding of the total variation (TV) model [35] and its variants [1, 10, 11, 14, 21, 23, 27, 28, 38].

However, most of those denoising models may lose fine structures of the image due to a certain undesired dissipation. As remedies, the employment of the G-norm [28] and iterative refinement [32] have been studied. But 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) [13] and the equalized net diffusion (END) approach [12]. 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 undesired dissipation, fourth-order PDE models have been emerged [19, 26, 40]. In particular, the Laplacian mean-curvature (LMC) model has been paid a particular attention due to its potential capability to preserve edges of linear curvatures. However, it has been numerically verified [39] that the LMC model can easily introduce granule-shaped spots to restored images. To overcome the granularity, this article proposes a hybrid model which combines a TV-based model and the LMC.

In this paper, we investigate the properties of a proposed hybrid TV–LMC model for image denoising.

1-4244-0054-6/06/$20.00

The hybrid model introduces a number of parameters, highlighting the need for a procedure for selecting optimal parameters. Preliminary to the development of such a selection procedure, we have undertaken a parametric study of the hybrid model in order to discover the solitary and interactive effects of the parameters on model accuracy. Such a parametric 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 parametric study was implemented as a parallel computing application. This paper focuses on the performance of the parallel implementation on a general-purpose Linux cluster.

The rest of this paper is organized as follows. It discusses the hybrid model for image denoising in Section 2. The dynamic load balancing tool utilized to parallelize the code for the parametric study is described in Section 3. Section 4 presents sample results of performance tests of the parallel implementation. Section 5 gives a summary and describes future work.

2. A Hybrid Model for Image Denoising

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

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

In equation (1), f is a contaminated image and the solution u represents a denoised 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 [22, 27].

2.1. The model

As a remedy for the granule-shaped spots introduced by the LMC model, 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

$$\widetilde{\kappa}(u) = |\nabla u|\,\kappa(u) = |\nabla u|\,\nabla \cdot \Big(\frac{\nabla u}{|\nabla u|}\Big).$$

Here the gradient magnitude $|\nabla u|$ has been incorporated into $\widetilde{\kappa}(u)$, as a scaling factor, in order to reduce staircasing [27]. 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 [39]. 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 $u^n = u(\cdot, t^n)$, $n = 0, 1, \cdots$, with $u^0 = f$. Let $(D_{x_1}, D_{x_2})^T$ and $(D_{2x_1}, D_{2x_2})^T$ be the half-step (regular) central difference and 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 $m = n-1, n$,

$$\begin{aligned}
\mathcal{K}_\ell u^m &= -|\nabla_E u^{n-1}| D_{x_\ell}\Big(\frac{D_{x_\ell} u^m}{|\nabla_h u^{n-1}|}\Big),\\
\mathcal{K}_\ell^2 u^m &= -|\nabla_E u^{n-1}| D_{2x_\ell}\Big(\frac{D_{2x_\ell} u^m}{|\nabla_h u^{n-1}|}\Big),\\
\mathcal{K}_\ell^\alpha u^m &= (1-\alpha)\mathcal{K}_\ell\, u^m + \alpha\mathcal{K}_\ell^2\, u^m, \quad \alpha \in [0,1],\\
\mathcal{L}_\ell u^m &= -D_{x_\ell} D_{x_\ell} u^m,
\end{aligned} \tag{3}$$

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

$$\mathcal{D} \;=\; \beta + \sigma\mathcal{K}^\alpha + \mathcal{L}\,\mathcal{K}. \tag{4}$$

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

$$\frac{u^n - u^{n-1}}{\Delta t} + \mathcal{D}\,\frac{u^n + u^{n-1}}{2} \;=\; \beta f. \tag{5}$$

Now, let

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

Since

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

(5) can be rewritten as

$$\frac{u^n - u^{n-1}}{\Delta t} + (\mathcal{A}_1 + \mathcal{A}_2)\frac{u^n + u^{n-1}}{2} = \beta f - \frac{1}{2}(\mathcal{L}_1\mathcal{K}_2 + \mathcal{L}_2\mathcal{K}_1)(u^n + u^{n-1}). \tag{6}$$

Thus, replacing the last term in (6) by known values with the error not larger than the truncation error, an alternating direction implicit (ADI) method for (5) can be formulated [16, 17] as

$$\begin{aligned}\left(1+\frac{\Delta t}{2}\mathcal{A}_1\right)u^* &= \left(1-\frac{\Delta t}{2}\mathcal{A}_1 - \Delta t\mathcal{A}_2\right)u^{n-1} \\ &\quad +\Delta t\beta f - \frac{\Delta t}{2}(\mathcal{L}_1\mathcal{K}_2+\mathcal{L}_2\mathcal{K}_1)\times \\ &\quad (3u^{n-1}-u^{n-2}), \\ \left(1+\frac{\Delta t}{2}\mathcal{A}_2\right)u^n &= u^* + \frac{\Delta t}{2}\mathcal{A}_2 u^{n-1}.\end{aligned} \tag{7}$$

The quantity u^* is an intermediate solution. In this article, we will call (7) 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.

Note that the CN-ADI algorithm (7) is a three-step procedure, defined well for $n \geq 2$. For $n = 1$, one may conveniently assume $u^{-1} = f = u^0$.

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 find an analytic stopping time. Therefore, finding an appropriate iteration count, n, is an important problem. The timestep size Δt is also important for the effectiveness of the algorithm, due to the fact that the timestep is strongly related to the frequencies of image content that will be eliminated by the algorithm [21].

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 these parameters on the quality of the restored image, we simulate the model on known images with synthetically-added noise. This simulation is discussed in the next section.

2.4. Parametric Study

Preliminary to the development of a procedure for selecting parameters for the 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

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

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 parametric study, following the pseudocode in Figure 1. Various plots from the outputs of the study could be produced, including animations of PSNR as a function 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 $\text{N}_\beta \times \text{N}_\sigma \times \text{N}_\alpha \times \text{N}_{\Delta t} \times \text{N}_n$, 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 parametric study, we used a dynamic load balancing tool we developed to parallelize serial codes with a structure similar to Figure 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 Figure 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

Establish uncontaminated image g
Add Gaussian noise to g to produce contaminated image f
Establish parameter counts $\mathrm{N}_\beta, \mathrm{N}_\sigma, \mathrm{N}_\alpha, \mathrm{N}_{\Delta t}, \mathrm{N}_n$
Establish parameter values $\beta[1], \ldots, \beta[\mathrm{N}_\beta]$; $\sigma[1], \ldots, \sigma[\mathrm{N}_\sigma]$;
$\alpha[1], \ldots, \alpha[\mathrm{N}_\alpha]$; $\Delta t[1], \ldots, \Delta t[\mathrm{N}_{\Delta t}]$; $n[1], \ldots, n[\mathrm{N}_n]$
For each combination of $\beta, \sigma, \alpha, \Delta t, n$ values
  Apply denoising procedure on f to produce restored image u
  Calculate PSNR; output $\beta, \sigma, \alpha, \Delta t, n$ and PSNR
End for

Figure 1. High-level outline of parametric study

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.

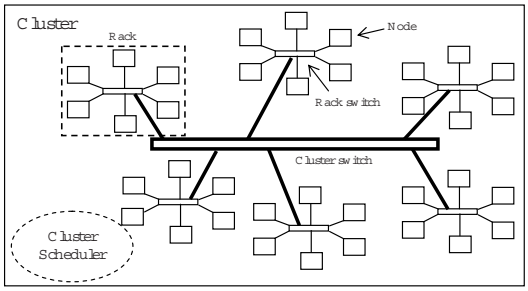

Figure 2. A popular interconnection network for clusters

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 [9]. 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 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.

```
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)
  ...
```

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

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 [34], factoring [20], variants of adaptive weighted factoring [3, 5, 7, 8], and adaptive factoring [2, 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 `iTime`s 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 parametric 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 parametric studies with $N_\beta = 9$, $N_\sigma = 9$, $N_\alpha = 9$, $N_{\Delta t} = 9$ and $N_n = 15$, for a total of 98,415 parameter combinations, for a number of images. The studies were executed on the heterogeneous general-purpose EMPIRE cluster of the Mississippi State University. The cluster can be abstracted as in Figure 2 and has a total of 1038 processors. A rack contains 32 nodes of dual 1.0GHz or 1.266GHz Pentium III processors and 1.25GB RAM. Each node is connected to a 100Mb/s Ethernet rack switch. The rack switches are connected by a gigabit 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.

Figure 4 gives a summary of the performance of the parallel code for the parametric study on the image *LenaGray256*. This study was submitted as a 32-processor job on the EMPIRE cluster; the cluster scheduler assigned homogeneous processors to the job. Since jobs were also 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. The left axis (for the bars) denotes the number of iterations of the loop in Figure 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 done 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. 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 effi-

ciency is: (estimated sequential cost)/(parallel cost) = (260.4547)/(32×8.453) = 0.963. The high efficiency indicates that the dynamic load balancing tool successfully addressed the load imbalance.

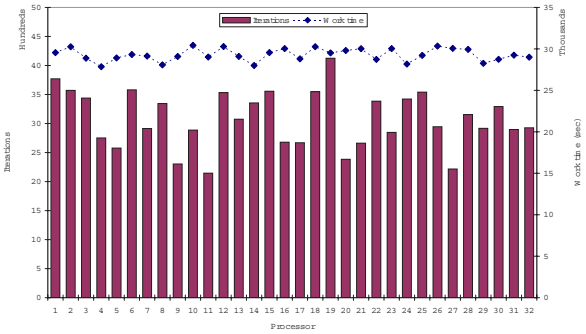

Figure 4. Distribution of iterations and work times for the parametric study on the image *LenaGray256*

Figure 5 gives the summary for the parametric study on the image *BlackCircle*. The cluster scheduler again 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)/(32×39.546) = 0.967.

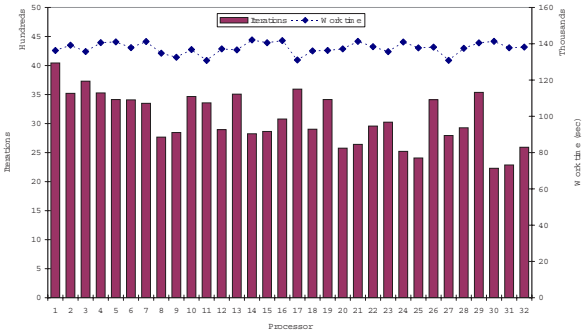

Figure 5. Distribution of iterations and work times for the parametric study on the image *BlackCircle*

5. Summary

We proposed a hybrid model for restoring noisy images. 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 the best restored images. To guide the development of a procedure for selecting the parameters, we simulated the hybrid model by applying it to restore known images that were contaminated with Gaussian noise. The simulations were conducted as parametric studies on a general-purpose heterogeneous Linux cluster. To address the load imbalance that potentially arises from algorithmic and systemic sources, we used a dynamic load balancing tool in the simulation code. The simulations achieved very high estimated efficiencies.

We are currently analyzing the outputs of the parametric studies to develop methods for selecting optimal parameters of the hybrid model. The results of this analysis will be reported in the future.

Acknowledgments

This work was partially supported by the following National Science Foundation grants: #0132618 (Cariño, Lim, Williams); #9984465 (Banicescu); and #0312223 (Kim).

References

[1] L. Alvarez, P. Lions, and M. Morel. Image selective smoothing and edge detection by nonlinear diffusion. II. *SIAM J. Numer. Anal.*, 29:845–866, 1992.

[2] I. Banicescu and Z. Liu. Adaptive factoring: A dynamic scheduling method tuned to the rate of weight changes. In *Proceedings of the High Performance Computing Symposium (HPC) 2000*, pages 122–129, 2000.

[3] I. Banicescu and V. Velusamy. Performance of scheduling scientific applications with adaptive weighted factoring. In *Proceedings of the 15th IEEE International Parallel and Distributed Processing Symposium - 10th Heterogeneous Computing Workshop (IPDPS-HCW 2001) CDROM*. IEEE Computer Society, Apr. 2001.

[4] I. Banicescu and V. Velusamy. 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) CDROM*. IEEE Computer Society, 2002.

[5] I. Banicescu, V. Velusamy, and J. Devaprasad. On the scalability of dynamic scheduling scientific applications with adaptive weighted factoring. *Cluster Com-*

puting: The Journal of Networks, Software Tools and Applications, 6:215–226, 2003.

[6] P. Blomgren and T. Chan. Color TV: Total variation methods for restoration of vector valued images. *IEEE Trans. Image Process.*, 7(3):304–309, 1998.

[7] R. L. Cariño and I. Banicescu. 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, 2002.

[8] R. L. Cariño and I. Banicescu. Load balancing parallel loops on message-passing systems. In S. Akl and T. Gonzales, editors, *Proceedings of the 14th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, pages 362–367. ACTA Press, 2002.

[9] R. L. Cariño and I. Banicescu. A load balancing tool for distributed parallel loops. *Cluster Computing: The Journal of Networks, Software Tools and Applications*, 8(4):To appear, 2005.

[10] F. Catte, P. Lions, M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM J. Numer. Anal.*, 29:182–193, 1992.

[11] Y. Cha and S. Kim. Edge-forming methods for color image zooming. (accepted to IEEE Trans. Image Process.).

[12] Y. Cha and S. Kim. Equalized net diffusion (END) for the preservation of fine structures in PDE-based image restoration. (submitted to IEEE Trans. Image Process.).

[13] Y. Cha and S. Kim. MONTE: The method of nonflat time evolution in PDE-based image restoration. (submitted to IEEE Trans. Image Process.).

[14] T. Chan, S. Osher, and J. Shen. The digital TV filter and nonlinear denoising. Technical Report #99-34, Department of Mathematics, University of California, Los Angeles, CA 90095-1555, October 1999.

[15] T. Chan and J. Shen. Variational restoration of nonflat image features: Models and algorithms. *SIAM J. Appl. Math.*, 61(4):1338–1361, 2000.

[16] J. Douglas, Jr. and J. Gunn. A general formulation of alternating direction methods Part I. Parabolic and hyperbolic problems. *Numer. Math.*, 6:428–453, 1964.

[17] J. Douglas, Jr. and S. Kim. Improved accuracy for locally one-dimensional methods for parabolic equations. *Mathematical Models and Methods in Applied Sciences*, 11:1563–1579, 2001.

[18] R. Gonzalez and R. Woods. *Digital Image Processing, 2nd Ed.* Prentice-Hall, Inc., Upper Saddle River, New Jersey, 2002.

[19] J. Greer and A. Bertozzi. Traveling wave solutions of fourth order PDEs for image processing. *SIAM J. Numer. Anal.*, 36(1):38–68, 2004.

[20] S. F. Hummel, E. Schonberg, and L. E. Flynn. Factoring: A method for scheduling parallel loops. *Communications of the ACM*, 35(8):90–101, Aug. 1992.

[21] S. Kim. Edge-preserving noise removal: Motion by mean curvature. (submitted to SIAM J. Sci. Comput.).

[22] S. Kim. PDE-based image restoration: A hybrid model and color image denoising. (accepted to IEEE Trans. Image Process.).

[23] S. Kim and H. Lim. A non-convex diffusion model for simultaneous image denoising and edge enhancement. (submitted to Electron. J. Differ. Equ.).

[24] R. Kimmel and N. Sochen. Orientation diffusion or how to comb a porcupine? *Special issue on PDEs in Image Processing, Computer Vision, and Computer Graphics, J. Visual Comm. Image Representation*, 13:238–248, 2002.

[25] C. Kruskal and A. Weiss. Allocating independent subtasks on parallel processors. *IEEE Transactions on Software Engineering*, SE-11(10):1001–1016, Oct. 1985.

[26] M. Lysaker, A. Lundervold, and X. Tai. 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, 2003.

[27] A. Marquina and S. Osher. 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, 2000.

[28] Y. Meyer. *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, volume 22 of *University Lecture Series*. American Mathematical Society, Providence, Rhode Island, 2001.

[29] S. Mitra and G. Sicuranza. *Nonlinear Image Processing*. Academic Press, San Diego, San Francisco, New York, Boston, London, Sydney, ToKyo, 2001.

[30] J.-M. Morel and S. Solimini. *Variational Methods in Image Segmentation*, volume 14 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser, Boston, 1995.

[31] M. Nitzberg and T. Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 14:826–833, 1992.

[32] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. 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, 2004.

[33] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Trans. on Pattern Anal. Mach. Intell.*, 12:629–639, 1990.

[34] C. Polychronopoulos and D. Kuck. Guided self-scheduling: A practical scheduling scheme for parallel supercomputers. *IEEE Transactions on Computers*, C-36(12):1425–1439, Dec. 1987.

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

[36] G. Sapiro. *Geometric partial differential equations and image analysis*. Cambridge University Press, Cambridge, 2001.

[37] N. Sochen, R. Kimmel, and R. Malladi. A general framework for low level vision. *IEEE Trans. Image Process.*, 7(3):310–318, 1998.

[38] L. Vese and S. Osher. Numerical methods for p-harmonic flows and applications to image processing. Technical Report #01-22, Department of Mathematics, University of California, Los Angeles, CA 90095, USA, August 2001. (To appear in SIAM Numer. Anal.).

[39] N. Williams, H. Lim, and S. Kim. Numerical schemes for the Laplacian mean-curvature flow and their applications to image denoising. (In preparation).

[40] Y. You and M. Kaveh. Fourth-order partial differential equations for noise removal. *IEEE Trans. Image Process.*, 9(10):1723–1730, 2000.

[41] Y. You, W. Xu, A. Tannenbaum, and M. Kaveh. Behavioral analysis of anisotropic diffusion in image processing. *IEEE Trans. Image Process.*, 5:1539–1553, 1996.