High performance noise reduction for biomedical multidimensional data

S. Tabik, E.M. Garzón, I. García, J.J. Fernández [*]

Department of Computer Architecture and Electronics, University of Almería, Almería 04120, Spain

Available online 30 November 2006

Abstract

Anisotropic nonlinear diffusion (AND) is one of the most powerful noise reduction techniques in the field of image processing. This method is based on a partial differential equation (PDE) tightly coupled with a massive set of eigensystems. Denoising large 3D images in biomedicine and structural cellular biology by AND is extremely expensive from a computational point of view, with huge memory needs. In this work, high performance computing techniques have been applied to AND. An strategy for optimal memory usage has been designed, which has allowed a remarkable reduction of the memory requirements. Parallel implementations of AND have been developed with special focus on clusters of symmetric multiprocessors (SMPs), currently a dominant platform in high performance computing. Different programming models have been used for the parallelization of AND: (1) Message-passing paradigm using MPI and (2) a hybrid paradigm that uses message passing among nodes plus the shared address space paradigm between the processors within the nodes. The parallel approaches have been evaluated on a cluster of dual-Xeon nodes, a representative example of clusters of SMPs. The conclusion drawn is that the hybrid approach is the most suitable for the parallelization of AND for this kind of computing platforms.

Keywords: Anisotropic nonlinear diffusion; Image processing; Parallel computing; Symmetric multiprocessors; Message passing; Distributed shared memory; MPI; Pthreads

1. Introduction

In many disciplines, raw data acquired from instruments are substantially corrupted by noise and sophisticated filtering techniques are then indispensable for a proper interpretation or post-processing. In general terms, smoothing techniques can be classified into linear and nonlinear [1]. Standard linear filtering techniques based on local averages or Gaussian kernels succeed in reducing the noise, but at expenses of poor feature preservation. In other words, they may severely blur the features as their edges are attenuated. However, nonlinear filtering techniques achieve better feature preservation as they try to adaptively tune the strength of the smoothing to the local structures found in the image.

Anisotropic nonlinear diffusion (AND) is currently one of the most powerful noise reduction techniques in the field of image processing [2]. This technique takes into account the local structures found in the image to filter noise, preserve edges and enhance some features, thus considerably increasing the signal-to-noise ratio (SNR) with

[*] Corresponding author. Fax: +34 950 015 486.
E-mail address: jose@ace.ual.es (J.J. Fernández).

no significant quantitative distortions of the signal. Pioneered in 1990 by Perona and Malik [3], AND has grown up to become a well-established tool in the last decade [2,4–6]. AND has already been successfully applied in different disciplines, such as medicine [7–9] or biology [1,10–12], for denoising multidimensional images. AND has actually been crucial to achieve some recent breakthroughs [13–16].

The mathematical basis of AND is a partial differential equation (PDE) tightly coupled with a massive set of eigensystems [1]. The computational cost of AND may be very high, depending on the size of the images. There are some disciplines, such as electron tomography of large biological specimens [17,18], where the requirements may be so huge—much more than 1 Gbyte in size—that high performance computing proves to be essential [19,20]. Therefore, denoising volumetric reconstructions in electron tomography is a highly expensive computational procedure in terms of time and memory consumption.

In this work, we describe, analyze and compare several parallel implementations of AND for its application to denoising large three-dimensional (3D) volumes in biomedicine and structural cellular biology. We focus on parallel approaches for clusters of symmetric multiprocessors, since they are currently the dominant systems in high performance computing laboratories [21]. We have also designed a strategy for optimal memory usage that significantly reduces the huge memory requirements.

2. Review of anisotropic nonlinear diffusion

AND accomplishes a sophisticated edge-preserving denoising that takes into account the structures at local scales. AND tunes the strength of the smoothing along different directions based on the local structure estimated at every point of the multidimensional image. Conceptually speaking, AND can be considered as an adaptive Gaussian filtering technique in which, for every voxel in the volume, an anisotropic 3D Gaussian function is computed whose widths and orientations depend on the local structure [22]. This section presents local structure determination via structure tensors, the concept of diffusion, a diffusion approach commonly used in image processing, details of the numerical implementation and, finally, our diffusion approach with optimal memory usage.

2.1. Estimation of local structure

The *structure tensor* is the mathematical tool that allows us to estimate the local structure in a multidimensional image. Let $I(\mathbf{x})$ denote a 3D image, where $\mathbf{x} = (x, y, z)$ is the coordinate vector. The structure tensor of I is a symmetric positive semi-definite matrix given by

$$\mathbf{J}(\nabla I) = \nabla I \cdot \nabla I^T = \begin{bmatrix} I_x^2 & I_x I_y & I_x I_z \\ I_x I_y & I_y^2 & I_y I_z \\ I_x I_z & I_y I_z & I_z^2 \end{bmatrix}, \tag{1}$$

where $I_x = \partial I/\partial x$, $I_y = \partial I/\partial y$, $I_z = \partial I/\partial z$ are the derivatives of the image with respect to x, y and z, respectively. In practice, for regularization purposes [2], the structure tensor is computed from a smoothed version of the image I_σ obtained by convolution with a Gaussian K_σ with standard deviation σ, i.e.,

$$\mathbf{J}(\nabla I_\sigma) = \nabla I_\sigma \cdot \nabla {I_\sigma}^T \tag{2}$$

with

$$I_\sigma(\mathbf{x}) = (K_\sigma * I)(\mathbf{x}) \tag{3}$$

and

$$K_\sigma(\mathbf{x}) = \frac{1}{(\sqrt{2\pi}\sigma)^3} \cdot \exp\left(-\frac{|\mathbf{x}|^2}{2\sigma^2}\right). \tag{4}$$

The eigen-analysis of the structure tensor allows determination of the local structural features in the image [2]:

$$\mathbf{J}(\nabla I_\sigma) = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3] \cdot \begin{bmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{bmatrix} \cdot [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]^T. \tag{5}$$

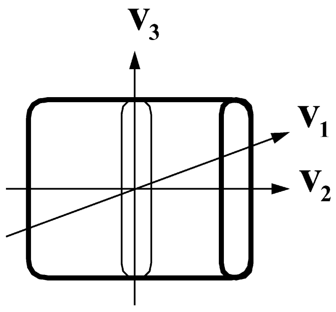

Fig. 1. Local structure found by eigen-analysis of the structure tensor. $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$ are the corresponding eigenvectors. $\mathbf{v_1}$ is the direction normal to the local structure.

The orthogonal eigenvectors $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ provide the preferred local orientations, and the corresponding eigenvalues μ_1, μ_2, μ_3 (assume $\mu_1 \geqslant \mu_2 \geqslant \mu_3$) provide the average contrast along these directions. The first eigenvector $\mathbf{v}_1$ represents the direction of the maximum variance. Therefore, $\mathbf{v}_1$ represents the direction normal to the local feature (see Fig. 1).

2.2. *Concept of diffusion in image processing*

Diffusion is a physical process that equilibrates concentration differences as a function of time, without creating or destroying mass. In image processing, density values play the role of concentration. This observation is expressed by the *diffusion equation* [2]:

$$I_t = \operatorname{div}(\mathbf{D} \cdot \nabla I), \tag{6}$$

where $I_t = \partial I/\partial t$ denotes the derivative of the image I with respect to the time t, ∇I is the gradient vector, $\mathbf{D}$ is a square matrix called *diffusion tensor*, and div is the *divergence* operator:

$$\operatorname{div}(\mathbf{f}) = \frac{\partial f_x}{\partial x} + \frac{\partial f_y}{\partial y} + \frac{\partial f_z}{\partial z}.$$

In AND the smoothing depends on both the strength of the gradient and its direction measured at a local scale [1]. The diffusion tensor $\mathbf{D}$ is therefore defined as a function of the structure tensor $\mathbf{J}$:

$$\mathbf{D} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3] \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \cdot [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3]^T, \tag{7}$$

where $\mathbf{v}_i$ denotes the eigenvectors of the structure tensor. The values of the eigenvalues λ_i define the strength of the smoothing along the direction of the corresponding eigenvector $\mathbf{v}_i$. The values of λ_i rank from 0 (no smoothing) to 1 (strong smoothing).

Therefore, this approach allows smoothing to take place anisotropically according to the eigenvectors determined from the local structure of the image. Consequently, AND allows smoothing on the edges: Smoothing runs along the edges so that they are not only preserved but smoothed. AND has turned out, by far, the most effective denoising method by its capabilities for structure preservation and feature enhancement [1,2,10,12].

2.3. *Edge enhancing diffusion*

One of the most common ways of setting up the diffusion tensor $\mathbf{D}$ gives rise to the so-called edge enhancing diffusion (EED) approach [2]. The primary effects of EED are edge preservation and enhancement. Here strong smoothing is applied along the preferred directions of the local structure (the second and third eigenvectors, $\mathbf{v}_2$ and $\mathbf{v}_3$). The strength of the smoothing along the normal of the structure, i.e., the eigenvector $\mathbf{v}_1$, depends on the gradient: the higher the value is, the lower the smoothing strength is. Consequently, λ_i are then set up as

$$\begin{cases} \lambda_1 = g(|\nabla I_\sigma|), \\ \lambda_2 = 1, \\ \lambda_3 = 1 \end{cases} \tag{8}$$

with g being a monotonically decreasing function, such as $g(x) = 1/\sqrt{(1 + x^2/K^2)}$, where $K > 0$ acts as a contrast parameter [2]; structures with $|\nabla I_\sigma| > K$ are regarded as edges, otherwise they are considered to belong to the interior of a region. Therefore, smoothing along edges is preferred over smoothing across them, hence edges are preserved and enhanced.

2.4. Numerical discretization of the diffusion equation

The diffusion equation (6), can be numerically solved using the standard numerical scheme that is based upon an explicit finite difference discretization [23]. The term $I_t = \partial I/\partial t$ can be replaced by an Euler forward difference approximation. The resulting explicit scheme allows calculation of subsequent versions of the image iteratively:

$$I^{s+1} = I^s + \tau \cdot \left(\frac{\partial}{\partial x}(D_{11} I_x) + \frac{\partial}{\partial x}(D_{12} I_y) + \frac{\partial}{\partial x}(D_{13} I_z) + \frac{\partial}{\partial y}(D_{21} I_x) + \frac{\partial}{\partial y}(D_{22} I_y) + \frac{\partial}{\partial y}(D_{23} I_z) \right.$$
$$\left. + \frac{\partial}{\partial z}(D_{31} I_x) + \frac{\partial}{\partial z}(D_{32} I_y) + \frac{\partial}{\partial z}(D_{33} I_z) \right), \tag{9}$$

where s is the iteration index, τ denotes the time step size, I^s denotes the image at time $t_s = s\tau$, the terms I_x, I_y, I_z are the derivatives of the image I^s with respect to x, y, and z, respectively. Finally, the D_{mn} terms represent the components of the diffusion tensor $\mathbf{D}^s$. The standard scheme to approximate the spatial derivatives ($\partial/\partial x$, $\partial/\partial y$, $\partial/\partial z$) is based on central differences. For any function $g_{i,j,k}$, these derivatives are given by

$$\frac{\partial}{\partial x} g_{i,j,k} = \frac{g_{i+1,j,k} - g_{i-1,j,k}}{2h_x},$$
$$\frac{\partial}{\partial y} g_{i,j,k} = \frac{g_{i,j+1,k} - g_{i,j-1,k}}{2h_y},$$
$$\frac{\partial}{\partial z} g_{i,j,k} = \frac{g_{i,j,k+1} - g_{i,j,k-1}}{2h_z},$$

where h_x, h_y, and h_z denote the pixel distance along the x-, y-, and z-directions, and are typically 1. See Appendix A in [10] for a detailed explanation of the numerical discretization.

In this traditional explicit scheme for solving the PDE in Eq. (6), the stability is an issue [2]. The maximum time step that is allowed is $\tau \leqslant 0.5/N_\mathrm{d}$, where N_d is the number of dimensions of the problem. In our case, we are dealing with a three-dimensional problem, so $N_\mathrm{d} = 3$. In the experiments carried out in this work, we used a conservative value of $\tau = 0.1$. As far as the number of iterations is concerned, a range 60–100 iterations is typically used in 3D problems with that value of τ. A detailed analysis of the influence of the parameters on execution times and the quality in the resulting images can be found elsewhere [1,2,10,12].

On the other hand, the Gaussian kernel used to compute the smoothed version of the image I_σ has been restricted in this work to a neighbourhood of radius 1. This extension allows regularization in the computation of gradients and structure tensors, without significant extension of the global data dependence (see below).

For illustration purposes, Fig. 2 shows the results of the application of 60 iterations of AND, with $\tau = 0.1$ and $\sigma = 0.5$, to a volume of a mitochondrion, a cell organelle, that was obtained by electron micrscope tomography [19,20]. Figure 3 shows another example of application of AND, with the same parameters and iterations, to a volume of Vaccinia virus, the virus used for vaccination against small pox. This latter dataset was obtained by cryoelectron tomography [16], a technique that is characterized by a extremely low SNR. In both figures, the enhancement in visualizing the slices of the volumes is apparent.

2.5. Algorithm for diffusion. Optimization of the memory management

In this work, we have implemented AND so that the volume is computed by z-planes, where—without loss of generality—the z-axis is the direction of the larger image dimension $N_z \geqslant N_x$, N_y. Note from Eq. (9) that I_k^{s+1} is only a function of I_{k-1}^s, I_k^s, I_{k+1}^s and $\mathbf{J}_{k-1}^s$, $\mathbf{J}_k^s$, $\mathbf{J}_{k+1}^s$, where I_k^s denotes the z-plane with index k of the image I^s, and $\mathbf{J}_k^s$ represents the structure tensor corresponding to the z-plane with index k computed from the image I^s. This data dependence is illustrated in Fig. 4. Our implementation can thus minimize the memory usage by allocating and

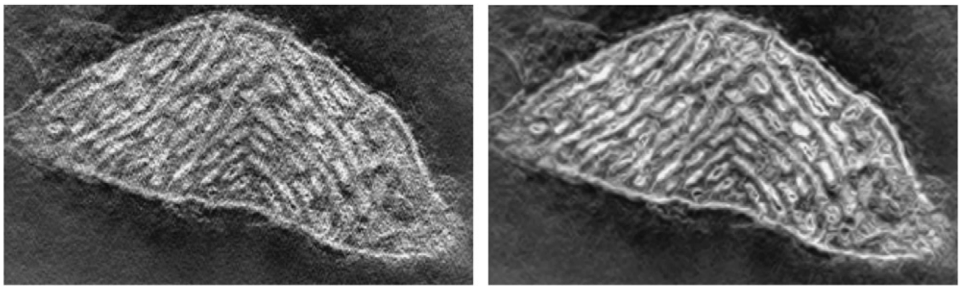

Fig. 2. Example of denoising with AND. Left: a slice from a volume of a mitochondrion obtained by electron microscope tomography; Right: the same slice from the volume filtered with anisotropic nonlinear diffusion. Sixty iterations of AND, with $\tau = 0.1$ and $\sigma = 0.5$ were used.

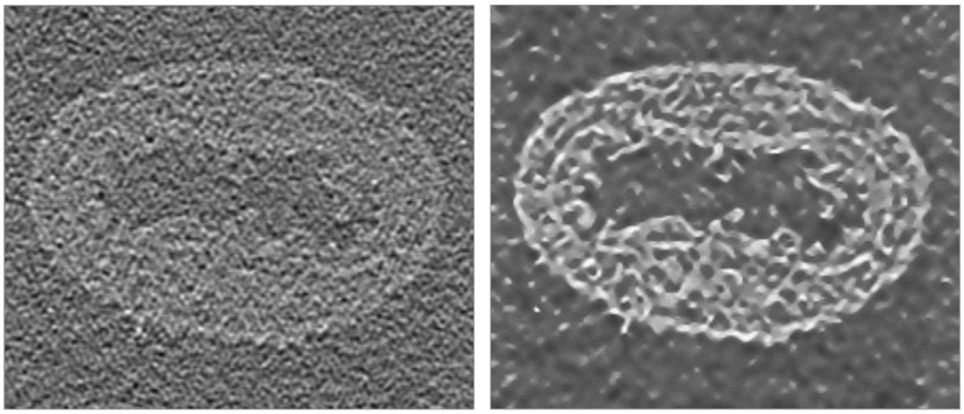

Fig. 3. Denoising Vaccinia virus with AND. Left: a slice from a volume of Vaccinia virus obtained by cryoelectron tomography; Right: the same slice from the volume filtered with anisotropic nonlinear diffusion. Sixty iterations of AND, with $\tau = 0.1$ and $\sigma = 0.5$ were used.

computing only the data strictly necessary for updating each single z-plane I_k^{s+1}. We only use a full-sized volume where I^s is initially stored and that is progressively updated with I^{s+1}. We then use three auxiliary volumes: first, a volume of size $N_x \times N_y \times 3$ to keep the three z-planes of the original image I_{k-1}^s, I_k^s, I_{k+1}^s needed to complete the diffusion process; second, a volume of size $N_x \times N_y \times 5$ to successively keep the five current Gaussian-smoothed z-planes of the image $I_{\sigma,k-2}^s$, $I_{\sigma,k-1}^s$, $I_{\sigma,k}^s$, $I_{\sigma,k+1}^s$, $I_{\sigma,k+2}^s$ needed for computation of the structure tensor; finally, a matrix of size $N_x \times N_y \times 3 \times 6$ to progressively hold the structure tensor $\mathbf{J}_{k-1}^s$, $\mathbf{J}_k^s$, $\mathbf{J}_{k+1}^s$ (note from Eq. (1) that the structure tensor is made up of 6 components). These three auxiliary volumes act as sliding windows, always keeping the data needed for computing the updated version I_k^{s+1} of the current plane with index k, as sketched in Fig. 4. This strategy for memory management substantially reduces the requirements of our sequential algorithm compared to standard strategies, where up to 8 times the size of the input volume is used [10,12].

The proposed sequential algorithm for solving the PDE in Eq. (9) that uses the discretization of the temporal and spatial derivatives described above and that works by z-planes, then consists of the following steps:

```
Do $s = 0, \dots, n-1$
  Do $k = 1, \dots, N_z$
  /* processing the volume by $z$-planes */
    (1) Compute the structure tensor $\mathbf{J}_k^s$ corresponding to the current $z$-plane, according to Eqs. (1), (2),
        and (5).
    (2) Compute the diffusion tensor $\mathbf{D}_k^s$ from the corresponding entries of $\mathbf{J}_k^s$, according to Eqs. (7) and (8).
    (3) Compute the resulting $z$-plane of the image $I_k^{s+1}$ at step $(s+1)$ from step $s$ by means of Eq. (9).
        The resulting $z$-plane of the image corresponds to the diffusion time $t_{(s+1)} = (s+1)\tau$.
  End Do
End Do
```

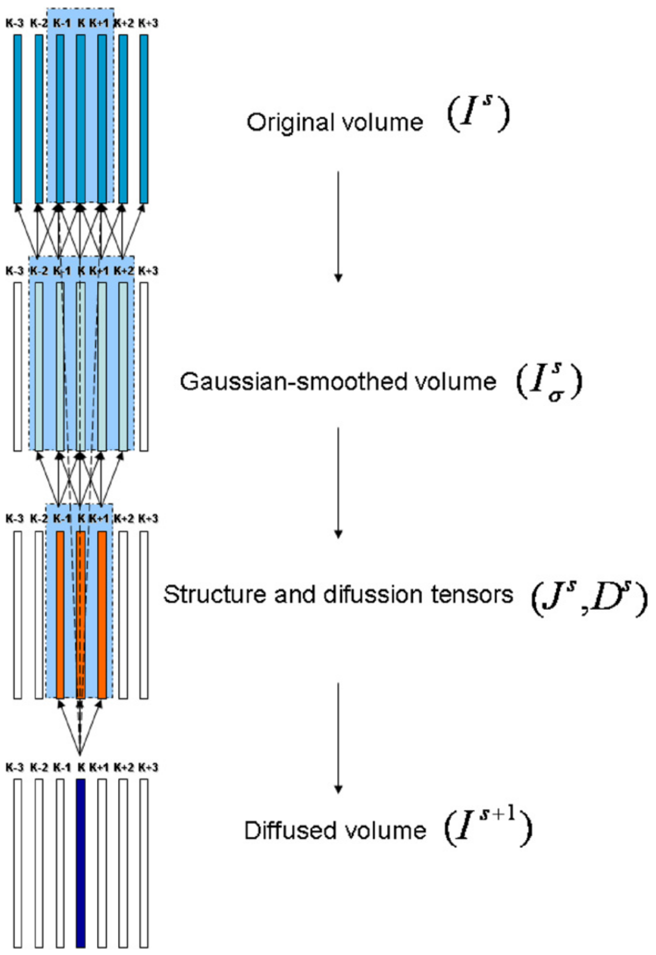
K-3 K-2 K-1 K K+1 K+2 K+3
Original volume (I^s)
Gaussian-smoothed volume (I^s_σ)
Structure and difussion tensors (J^s, D^s)
Diffused volume (I^{s+1})

Fig. 4. Data dependence to denoise the plane with index k in an iteration of AND. The diffused version of the plane I_k^{s+1} is a function of the structure tensor and the current version of the image, with a neighbourhood of unit radius using the explicit finite difference discretization and derivatives approximated by central differences, i.e., I_k^{s+1} is a function of $\mathbf{J}_{k-1}^s$, $\mathbf{J}_k^s$, $\mathbf{J}_{k+1}^s$ and I_{k-1}^s, I_k^s, I_{k+1}^s. The structure tensor of every plane $\mathbf{J}$ is computed from the Gaussian-smoothed image I_σ, which involves a neighbourhood of unit radius when central differences are used to compute the derivatives. Finally, the Gaussian-smoothed planes I_σ are computed from the current version of the image I^s, with a Gaussian kernel K_σ where, in this work, the neighbourhood has been restricted to unit radius. In summary, the neighbourhood required to compute the diffused version of the plane I_k^{s+1} has radius equal to three. In this figure, the shadowed boxes sketch the sliding windows that have been devised to optimize the memory usage by keeping the data strictly needed for computing I_k^{s+1}.

where s denotes the index of the iteration and k is the index of the z-plane. The algorithm is executed iteratively for a number of iterations n. The final image is obtained after a total diffusion time $T = n\tau$. Hereinafter, the body of this nested loop is denoted as $I_k^{s+1} = \text{AND}(I_k^s)$. Note that the computation of the structure tensor $\mathbf{J}_k^s$ already includes the Gaussian-smoothing of the corresponding z-planes of the image.

3. Parallel implementation of AND

Large scale high performance systems based on clusters of shared memory multiprocessors (also known as symmetric multiprocessors, SMPs) are today's dominant computing platforms [21,24]. They are made up of nodes connected by fast interconnection networks (typically 1 Gbit), and the nodes are composed of a number of processors (typically 2) that share the node memory. In these systems, communications and data exchange can be established through the network or/and the shared memory. These architectures support three parallel programming models:

(1) Shared address space model inside the nodes, using standards such as Posix threads [25] or OpenMP [26] to take advantage of the shared-memory feature of the nodes. This model is specially suitable for nodes with a high number of processors;
(2) Message-passing model, which uses the standard message-passing interface (MPI) [27] for message passing between different processors inside and between nodes; and finally,
(3) Hybrid model, which combines the two previous models, shared address space model inside nodes and message-passing model between nodes [28,29].

Several previous works have shown that there is no specific model that can be regarded as the ideal model for all applications [28,29].

In this work, we present and evaluate parallel strategies for anisotropic nonlinear diffusion for its execution on clusters of SMPs. One of the aims is to find out the parallel strategy that yields better performance for the AND on these computing platforms. In particular, this work describes, analyzes and compares two implementations of AND: (1) MPI based approach, where all communications and data exchange are carried out by message passing and (2) hybrid strategy, where Pthreads (Posix threads) [25] is used inside the nodes to exploit the shared memory, and MPI is used between the nodes.

An analysis of the data dependences of AND is essential to develop optimal parallel implementations. As described before, I_k^{s+1} is only a function of $\mathbf{J}_{k-1}^s$, $\mathbf{J}_k^s$, $\mathbf{J}_{k+1}^s$ and I_{k-1}^s, I_k^s, I_{k+1}^s; and the structure tensor $\mathbf{J}^s$ is computed from the Gaussian-smoothed image I_σ, according to Eq. (2). As a consequence, seven z-planes of the current version of the image I^s must be read (I_{k-3}^s, I_{k-2}^s, I_{k-1}^s,I_k^s,I_{k+1}^s, I_{k+2}^s, I_{k+3}^s) to compute five z-planes of smoothed image I_σ^s, three z-planes of structure tensor $\mathbf{J}$ and, finally, the diffused plane I_k^{s+1}. Therefore, the update of a plane with index k requires a neighbourhood of three planes at each side ($k \pm 3$). Figure 4 shows a sketch of this data dependence.

The parallel strategies that have been developed here are based on regular 1D domain decomposition: they consist of distributing the input 3D volume among the processors by blocks of consecutive z-planes, and every processor then applies the AND algorithm to its own block. This approach based on domain decomposition ensures a statically balanced computational work among the processors. In order to update the boundary z-planes of its block, every processor must have access to a neighbourhood of three planes at each boundary. These six additional planes are just required for the calculations but are not modified, and they are thus referred to as the *halo* planes of the block.

3.1. MPI implementation

In the parallel implementation based on the message-passing model, one process is spawned on each processor. Each processor then updates its own block of z-planes. According to the data dependence of AND, the halo z-planes of the current version of the image I^s must be read to update one block. To avoid excessive communications, each processor then allocates six additional z-planes to hold the halo planes. Therefore, every processor can update its block by one iteration of AND without communications. Only one communication point must be included at the end of each iteration of AND in order to update the halo planes between neighbour processors. This strategy thus involves a total amount of communicated data of 6 times the size of a z-plane. However, this approach requires extra computation be carried out by every processor, since two neighbour processors redundantly compute: (a) four (two at each boundary of the block) Gaussian-smoothed halo z-planes of the block and (b) the structure and diffusion tensors of two (one at each boundary) halo z-planes of the block.

Figure 5 shows a scheme of this MPI-based approach. Each node is assigned a set of N_z^{local} planes that are distributed among their processors. The processors also keep data for the halo z-planes. The processors apply the AND algorithm to their own block. At the end of each iteration, update of the halo planes takes place by means of point-to-point MPI communications with the neighbour processors.

An alternative implementation where redundant computation is avoided is possible. Here, every processor only allocates four (two at each boundary) additional halo z-planes. One communication point must then be included at the end of each iteration in order to update those z-planes. In addition, the processors must also communicate two z-planes of the diffusion tensor to their immediate neighbour at the beginning of each iteration. So, this approach only includes communications, with no redundant computation at all. However, here two communication points between

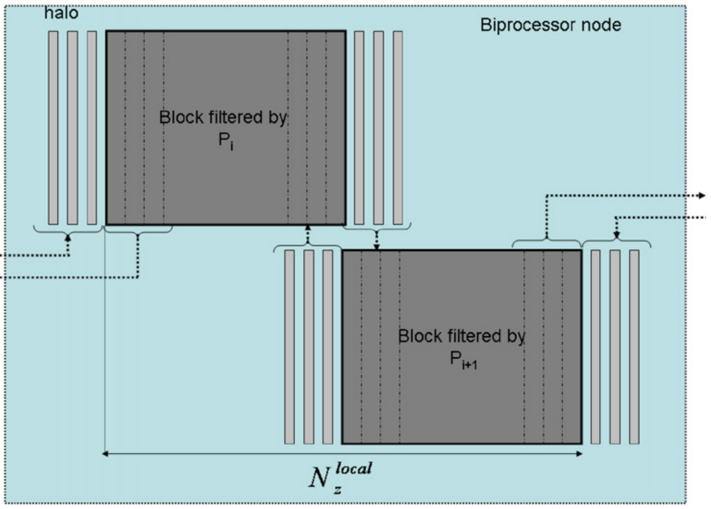

Fig. 5. Scheme of the MPI-based approach. All the processors apply the AND algorithm to its own block of z-planes (dark grey), and keep six halo planes (light grey) for the proper diffusion of the boundary planes of the block. At the end of each iteration, processors then communicate (dotted arrows) with their neighbours to update the halo planes. In this figure, a node has been considered to be made up of two processors, P_i and P_{i+1}.

neighbours must be included in every iteration, with a total amount of communicated data of 16 times the size of a z-plane. Therefore, the communication penalty is much higher in this MPI approach. This strategy was evaluated experimentally under the same conditions and computing platform as described below. The efficiency and scalability obtained were lower than for the MPI approach described above. Therefore, in this work, we show and analyze results from the parallel MPI-based approach that includes both redundant computation and communications.

3.2. Hybrid implementation

The hybrid strategy has been designed in such a way that one MPI process is spawned on each node, and the MPI process then creates as many Pthreads processes as the number of processors in the node. The block of z-planes assigned to the node will be shared by all the threads running in the node. Every thread updates its own subset of the block of z-planes. Transparently, the neighbour threads then have their halo z-planes updated thanks to the shared memory. To ensure consistency of the data throughout the algorithm, an additional structure has been defined to hold the halo z-planes before the neighbour threads modify them. At the end of the iteration, all the threads running in a node are joined. The six (three at each boundary) halo z-planes are then exchanged among the immediate neighbour nodes by MPI point-to-point communications between MPI processes.

Figure 6 shows a scheme of the hybrid approach. Each node is assigned a block of N_z^{local} planes that are distributed among their processors. The node also keeps data for the halo z-planes. The threads running in the processors apply the AND algorithm to their own subset of z-planes. At the end of each iteration, the update of the six halo planes of the block takes place by means of point-to-point MPI communications with the neighbour nodes.

The outline of the hybrid implementation would be as follows:

```
(1) Distribute $I^0$ among nodes, $N_z^{\text{local}}$ planes belong to the block assigned to each node.
(2) Do $s = 0, \dots, n-1$
    (a) Each thread initializes its auxiliary data structures.
    (b) /* each of the $T$ threads */
        Do $k = 1, \dots, \lceil (N_z^{\text{local}})/T \rceil$
            $I_k^{s+1} = \text{AND}(I_k^s)$
        End Do
    (c) Interchange three halo $z$-planes between neighbour nodes.
(3) End Do
(4) Collect the image
```

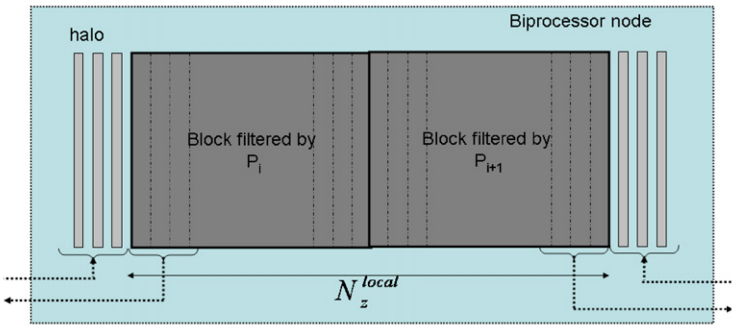

Fig. 6. Scheme of the hybrid approach. A node is assigned a block of N_z^{local} planes. The threads running in the processors of the node then apply the AND algorithm to their own subset of z-planes (dark grey). The node keeps six halo planes (light grey) for the proper diffusion of the boundary planes in the block. At the end of each iteration, nodes then communicate (dotted arrows) with their neighbours to update the halo planes. In this figure, a node has been considered to be made up of two processors, P_i and P_{i+1}.

where N_z^{local} denotes the local number of z-planes of the volume in every node, T is the number of threads running inside one node, and I^0 is the original 3D image, n denotes the number of iterations, and AND( ) represents the diffusion algorithm.

In this strategy, it is necessary to control the data distribution at two levels: at the node level and at the processor level inside the node.

(1) The total number of z-planes is distributed among the nodes: $N_z^{\text{local}} = \lceil N_z/M \rceil$ z-planes will be assigned to each node, where M denotes the number of nodes. The memory of every node also holds the six halo z-planes of the block. At the end of each iteration, nodes communicate the updated halo planes via message passing.
(2) At processor level, each thread will update its subset of $\lceil N_z^{\text{local}}/T \rceil$ z-planes by applying the AND process.

The computational complexity of these two parallel strategies is $O(N_x \times N_y \times N_z/P)$, where P is the total number of processors in the system. In message passing, only point-to-point communications among immediate neighbour nodes/processors are established and the total amount of communicated data is in the order of $O(N_x \times N_y \times P/T)$. Similarly, the amount of redundant computations is also in the order of $O(N_x \times N_y \times P/T)$. Note that T takes unit value for the MPI-based implementation.

4. Evaluation of the parallel implementations of AND

In this section, we evaluate the performance and the scalability of both parallel implementations of the AND method: MPI-based and the Hybrid, MPI&Pthreads-based approaches. The evaluation has been carried out on a cluster of SMPs with the following features:

> Cluster of 16 symmetric biprocessor nodes, which consists of 32 Intel(R) Xeon(TM) processors running at 3.06 GHz each processor, with 2 GB RAM, 512 KB cache. Nodes are interconnected via two 1 Gbit Ethernet networks, one for data (NFS) and the other for computation. The architecture of this cluster is based on uniform memory access (UMA), where every pair of processors in a node have equally fast (symmetric) access to the memory in the node.

For the hybrid implementation, the number of threads per node T has been fixed to 2, which is the number of processors per node that share the address space.

Dimensions of volumes in biomedicine and structural cellular biology typically range between 256^3 and 512^3, although sizes of 1024^3 or even 2048^3 are expected in the future. Typical values for n are around 60–100 iterations with $\tau = 0.1$, where n is the number of iterations needed to denoise the volume for an acceptable result [1,12]. In this work, volumes with sizes $256 \times 256 \times 256$ and $512 \times 512 \times 512$ have been tested. For the evaluation of the performance, five iterations of AND were considered sufficient. The times dedicated to read and write the volume have not been included in the evaluation process.

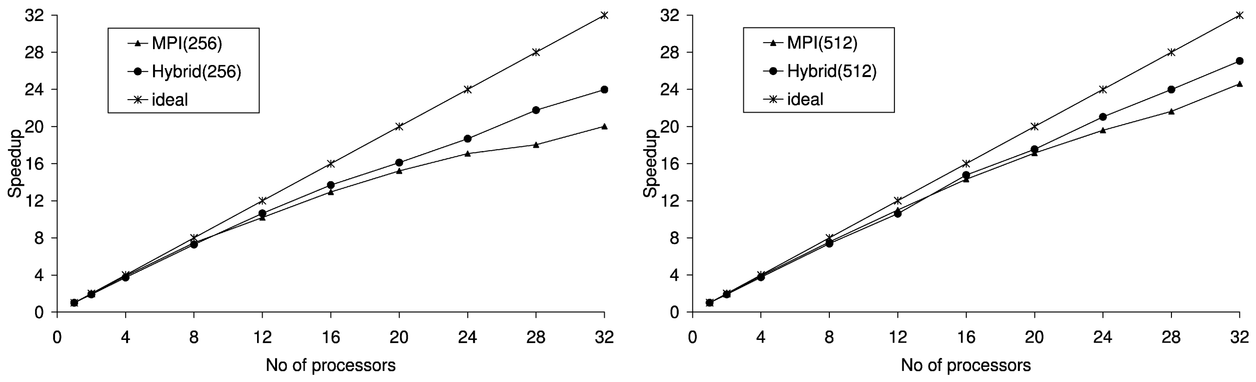

Fig. 7. Speedup of parallel AND with strategies based on (●) MPI and (▲) a hybrid one using MPI&Pthreads with 2 threads per node. Speedup obtained on a 16 dual-Xeon cluster for volume sizes of 256 × 256 × 256 (left) and 512 × 512 × 512 (right) are shown.

The performance of the parallel approaches has been evaluated in terms of speedup and the percentage of communication and redundant computation times. Speedup is the metric that is most often used to assess parallel performance and scalability [30]. It is defined as the ratio between the runtimes in the sequential and in the parallel version of the program. In this work, speedup has been computed to assess the effectiveness of the parallel approaches. The percentage of communication/redundant computation also provides information about the scalability of the parallel approaches.

4.1. Speedup

Figure 7 shows the speedup achieved by the MPI based and the hybrid implementations as a function of the number of processors and for both volume sizes. Clearly, speedup curves show better scalability for increasing volume sizes, due to the improvement in the ratio computation vs communication/redundancy. This behaviour is justified by the fact that the computational complexity depends linearly on the volume size whereas the amount of communications and redundancy is proportional to the size of a single z-slice. Therefore, any increase in the problem size is expected to imply a direct improvement in the speedup.

From the values of speedup shown, it can be concluded that for the small volume size, both implementations scale well up to 12–16 processors, approaching ideal linear speedup. However, from 16 processors on, there is a clear progressive decay of the speedup rate. For the large volume, both implementations exhibit a good scalability up to higher numbers of processors, i.e., 20–24 processors, with better behaviour for the hybrid one. As seen in Fig. 7, the hybrid approach clearly outperforms the MPI-based approach throughout the range of processors, and it is specially evident for increasing number of processors.

4.2. Percentage of communication/redundant computation

As said before, the MPI implementation presented here has two kind of penalties, the communications and the redundant computation. In order to further analyze the scalability, these penalty runtimes were measured and compared to the total runtime. These penalties were expected to be more relevant as the number of processors increases in view of the speedup curves shown above. This hypothesis is confirmed by the results shown in Figs. 8 and 9, which show the percentage of runtime dedicated to communication and the redundant computation in grey and white, respectively.

The results clearly show that the penalty due to communication is much lower for the hybrid approach than for the MPI-based one. This situation was already expected since the amount of communications in the hybrid approach is reduced with respect to that in the message passing one by a factor equal to the number of processors in the nodes. In the particular cluster where the results have been obtained, the reduction factor is about 2 as can be seen by comparing the grey columns in Figs. 8 and 9.

As far as the percentage of redundant computation is concerned, the results exhibit a much lower penalty for the hybrid approach than for the MPI-based one. This was also expected because the amount of redundant computations

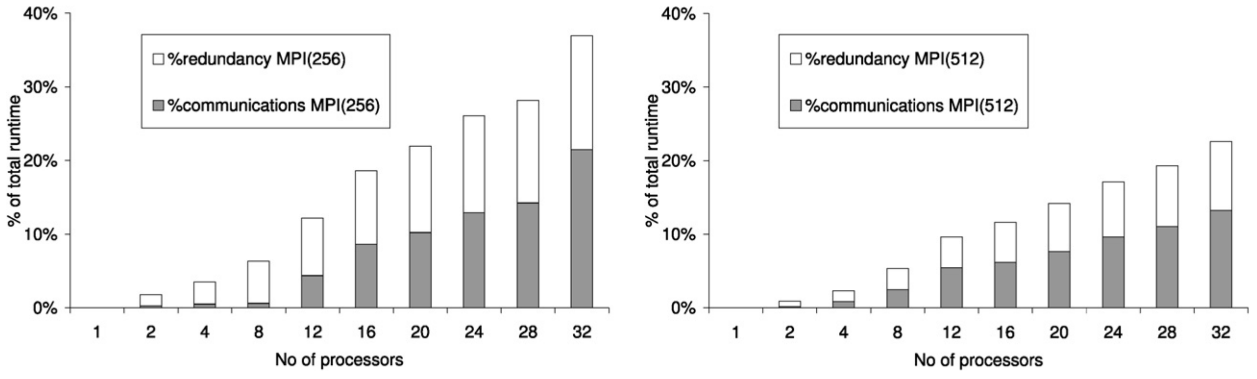

Fig. 8. Percentage of penalty runtimes due to communication (grey) and redundant computation (white) of the MPI-based implementation for volume sizes of $256 \times 256 \times 256$ (left) and $512 \times 512 \times 512$ (right).

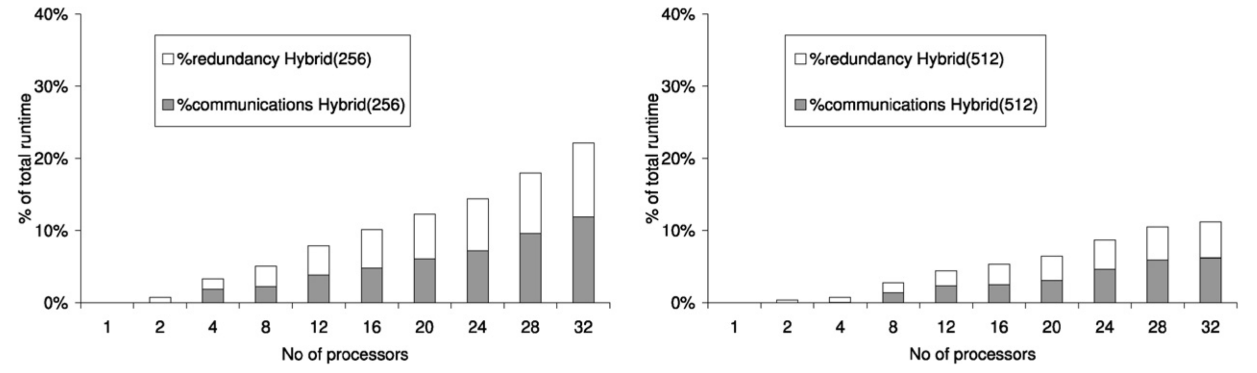

Fig. 9. Percentage of penalty runtimes due to communication (grey) and redundant computation (white) of the hybrid approach for volume sizes of $256 \times 256 \times 256$ (left) and $512 \times 512 \times 512$ (right).

is reduced by the same factor as in the communication issue. The white columns in Figs. 8 and 9 clearly show an approximate reduction factor of 2 when comparing the hybrid approach to the MPI-based one.

The fact that the hybrid approach exhibits a lower percentage of both communications and redundant computation than the MPI-based approach is consistent with the better speedups that have been shown and discussed in the previous section.

A comparison of the results as a function of the volume size clearly shows that the percentage of redundant computation is significantly reduced for larger volumes. Moreover, the reduction factor is approximately the same for the hybrid and for the MPI-based approaches. The underlying reason is the significantly higher computational load in the nodes for larger volumes, compared to the redundant computation. A volume of $512 \times 512 \times 512$ involves an increasing of the computational load of $8\times$ in all the nodes with regard to that required for a volume of $256 \times 256 \times 256$. However, the redundant computations only increases in a factor of $4\times$, as already described in Section 3. Therefore, any increase in the number of z-planes of the volume will provide a reduction of the percentage of the redundant computation, hence a improvement of the performance.

However, the behaviour of the communications as a function of the volume size is slightly different. The amount of communications is in the same order as the redundant computations, as shown in Section 3. However, Figs. 8 and 9 clearly show that the percentage of runtime dedicated to communications does not reduce in the same factor with the volume size as happened with the redundant computation. This has to do with the computational power of the processors versus the performance of the network. This behaviour points out that an increase in the redundant computation at expenses of communication may involve an increase in the performance. This conclusion is expected to be valid for current and possibly future clusters of SMPs, with fast and powerful processors (2.6–3.0 GHz) and 1 Gbit networks. Nevertheless, it may not apply for clusters of nodes with slower processors or with relatively faster networks.

5. Conclusions

In this work, we have applied high performance computing techniques for denoising large 3D volumes with AND. First we have designed an optimal usage of memory that significantly reduces the huge memory requirements of the algorithm. Then, we have presented different parallel implementations of AND specially focused on clusters of SMPs, one of the dominant high performance computing platforms. We have evaluated a strategy based on the message-passing paradigm and a hybrid approach that combines the message-passing and shared-memory paradigms. The parallel strategies were based on domain decomposition, with a static balance of the computational load. The strategies include communications among nodes and an amount of redundant computation. Both parallel approaches present good levels of scalability, in general. However, it is clear that the hybrid approach that combines message passing with MPI among the nodes and Pthreads to exploit the shared memory within the nodes is the optimal strategy for clusters of SMPs. These conclusions are applicable to other domain decomposition parallel applications where communications among immediate neighbours are involved, such as those related to image processing [20] or based on the solution of a PDE [23,31].

Acknowledgments

The authors wish to thank Dr. Guy Perkins (National Center for Microscopy and Imaging Research, San Diego, USA) and Professor J.L. Carrascosa (National Center for Biotechnology, Madrid, Spain) for kindly providing, respectively, the mitochondrion and Vaccinia virus datasets. This work was supported by the Spanish Ministry of Education and Science through grants TIC2002-00228 and TIN2005-00447.

References

[1] J.J. Fernandez, S. Li, Anisotropic nonlinear filtering of cellular structures in cryo-electron tomography, Comput. Sci. Eng. 7 (5) (2005) 54–61.
[2] J. Weickert, Anisotropic Diffusion in Image Processing, Teubner, Leipzig, 1998.
[3] P. Perona, J. Malik, Scale space and edge detection using anisotropic diffusion, IEEE Trans. Pattern Anal. Mach. Intel. 12 (1990) 629–639.
[4] J. Weickert, Coherence-enhancing diffusion filtering, Int. J. Comput. Vision 31 (1999) 111–127.
[5] J. Weickert, Coherence-enhancing diffusion of colour images, Image Vision Comput. 17 (1999) 201–212.
[6] J. Weickert, H. Scharr, A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance, J. Visual Comm. Imag. Repres. 13 (2002) 103–118.
[7] G. Gerig, R. Kikinis, O. Kubler, F.A. Jolesz, Nonlinear anisotropic filtering of MRI data, IEEE Trans. Med. Imag. 11 (1992) 221–232.
[8] I. Bajla, I. Hollander, Nonlinear filtering of magnetic resonance tomograms by geometry-driven diffusion, Machine Vision Appl. 10 (1998) 243–255.
[9] O. Ghita, K. Robinson, M. Lynch, P.F. Whelan, MRI diffusion-based filtering: A note on performance characterisation, Comput. Med. Imag. Graph. 29 (2005) 267–277.
[10] A.S. Frangakis, R. Hegerl, Noise reduction in electron tomographic reconstructions using nonlinear anisotropic diffusion, J. Struct. Biol. 135 (2001) 239–250.
[11] A.S. Frangakis, A. Stoschek, R. Hegerl, Wavelet transform filtering and nonlinear anisotropic diffusion assessed for signal reconstruction performance on multidimensional biomedical data, IEEE Trans. Biomed. Eng. 48 (2001) 213–222.
[12] J.J. Fernandez, S. Li, An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms, J. Struct. Biol. 144 (2003) 152–161.
[13] O. Medalia, I. Weber, A.S. Frangakis, D. Nicastro, G. Gerisch, W. Baumeister, Macromolecular architecture in eukaryotic cells visualized by cryoelectron tomography, Science 298 (2002) 1209–1213.
[14] K. Grunewald, P. Desai, D.C. Winkler, J.B. Heymann, D.M. Belnap, W. Baumeister, A.C. Steven, Three-dimensional structure of herpes simplex virus from cryo-electron tomography, Science 302 (2003) 1396–1398.
[15] M. Beck, F. Forster, M. Ecke, J.M. Plitzko, F. Melchior, G. Gerisch, W. Baumeister, O. Medalia, Nuclear pore complex structure and dynamics revealed by cryoelectron tomography, Science 306 (2004) 1387–1390.
[16] M. Cyrklaff, C. Risco, J.J. Fernandez, M.V. Jimenez, M. Esteban, W. Baumeister, J.L. Carrascosa, Cryo-electron tomography of vaccinia virus, Proc. Natl. Acad. Sci. USA 102 (2005) 2772–2777.
[17] J.J. Fernandez, C.O. Sorzano, R. Marabini, J.M. Carazo, Image processing and 3D reconstruction in electron microscopy, IEEE Signal Process. Mag. 23 (3) (2006) 84–94.
[18] A. Leis, M. Beck, M. Gruska, C. Best, R. Hegerl, W. Baumeister, J. Leis, Cryo-electron tomography of biological specimens, IEEE Signal Process. Mag. 23 (3) (2006) 95–103.
[19] J.J. Fernandez, A.F. Lawrence, J. Roca, I. Garcia, M.H. Ellisman, J. Carazo, High performance electron tomography of complex biological specimens, J. Struct. Biol. 138 (2002) 6–20.
[20] J.J. Fernandez, J. Carazo, I. Garcia, Three-dimensional reconstruction of cellular structures by electron microscope tomography and parallel computing, J. Parallel Distr. Comput. 64 (2004) 285–300.

[21] J. Dongarra, Trends in high performance computing, Comput. J. 47 (2004) 399–403.
[22] D. Barash, A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation, IEEE Trans. Pattern Anal. Mach. Intell. 24 (2002) 844–847.
[23] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes: The Art of Scientific Computing, Cambridge Univ. Press, Cambridge, 1992.
[24] H. Shan, J.P. Singh, L. Oliker, R. Biswas, Message passing and shared address space parallelism on an SMP cluster, Parallel Comput. 29 (2003) 167–186.
[25] D.R. Butenhof, Programming with POSIX(R) Threads, Addison–Wesley, Reading, MA, 1997.
[26] M.J. Quinn, Parallel Programming in C with MPI and OpenMP, McGraw–Hill, New York, 2004.
[27] W. Gropp, E. Lusk, A. Skjellum, Using MPI. Portable Parallel Programming with the Message Passing Interface, MIT Press, Cambridge, MA, 1999.
[28] L. Smith, M. Bull, Development of mixed mode MPI/OpenMP applications, Sci. Program. 9 (2001) 83–98.
[29] G. Krawezik, F. Cappello, Performance comparison of MPI and OpenMP on shared memory multiprocessors, Concurr. Comput. Pract. Exp. 18 (2006) 29–61.
[30] B. Wilkinson, M. Allen, Parallel Programming, second ed., Prentice Hall, New York, 2005.
[31] P. Bjorstad, M.M. Luskin (Eds.), Parallel Solution of Partial Differential Equations, IMA Volumes in Mathematics and Its Applications, vol. 120, Springer-Verlag, New York, 2000.

Siham Tabik received the B.Sc. degree in physics from University Med. V, Rabat, Morocco, in 1998. She is currently a Ph.D. student in the Department of Computer Architecture and Electronics, University of Almería, Spain, and a member of the Supercomputing-Algorithms Research Group. Her research interests include design and analysis of parallel algorithms for the numerical solution of nonlinearly coupled partial differential equations.

Ester M. Garzón received the B.Sc. degree in physics from the University of Granada, Granada, Spain, in 1985, and the Ph.D. degree in Computer Engineering from the University of Almería, Almería, Spain, in 2000. From 1989 to 1993, she was an Assistant Professor with the University of Granada. She is currently an Associate Professor with the Department of Computer Architecture and Electronics, University of Almería, Spain. She is a member of the Supercomputing-Algorithms Research Group. Her research interest focuses on the field of parallel algorithms for irregular problems in matricial algebra, data partition of sparse matrices, and parallel compilers.

Inmaculada García received the B.Sc. degree in physics in 1977 from the Complutense University of Madrid, Spain, and the Ph.D. degree in 1986 from the University of Santiago de Compostela, Spain. From 1977 to 1987 she was an Assistant Professor at the University of Granada, from 1987 to 1996 an Associated Professor at the University of Almería, and since 1997 she is a Full Professor at the University of Almería, Spain. She is head of the Department of Computer Architecture and Electronics and the Supercomputing-Algorithms Research Group. Her research interest lies in the field of parallel algorithms for irregular problems related to image processing, global optimization, and matrix computation.

José-Jesús Fernández received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, Spain, in 1992 and 1997, respectively. He was a Ph.D. student at the BioComputing Unit of the National Center for Biotechnology (CNB), Spanish Research Council (CSIC), Madrid, Spain. He became an Assistant Professor in October 1997 and since 2000 he is an Associate Professor of Computer Architecture at the University of Almería, Almería, Spain. He is a member of the Supercomputing-Algorithms Research Group at the University of Almería. His current research interests include high performance computing, image processing, and tomographic reconstruction.