Evaluation of Parallel Paradigms on Anisotropic Nonlinear Diffusion[*]

S. Tabik, E.M. Garzón, I. García, and J.J. Fernández

Dept. Computer Architecture and Electronics. University of Almería, Almería 04120, Spain
{siham-ester-inma-jose}@ace.ual.es

Abstract. Anisotropic Nonlinear Diffusion (AND) is a powerful noise reduction technique in the field of computer vision. 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, and the requirements may become so huge that parallel computing turns out to be essential. This work addresses the parallel implementation of AND. The parallelization is carried out by means of three paradigms: (1) Shared address space paradigm, (2) Message passing paradigm, and (3) Hybrid paradigm. The three parallel approaches have been evaluated on two parallel platforms: (1) a DSM (Distributed Shared Memory) platform based on cc-NUMA memory access and (2) a cluster of Symmetric biprocessors. An analysis of the performance of the three strategies has been accomplished to determine which is the most suitable paradigm for each platform.

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 non-linear. 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 computer vision [1]. 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 no significant quantitative distortions of the signal. Pioneered in 1990 by Perona and Malik [2], AND has grown up to become a well-established tool in the last decade [1,3,4]. AND has already been successfully applied in different disciplines, such as medicine [5,6,7] or biology [8,9,10], for denoising multidimensional images. AND has actually been crucial to achieve some recent breakthroughs [11,12,13,14].

[*] This work was supported by the Spanish Ministry of Education and Science through grant TIN2005-00447.

The mathematical basis of AND is a partial differential equation (PDE) tightly coupled with a massive set of eigensystems [10]. The computational cost of AND may be very high, depending on the size of the images. There are some disciplines where the requirements may be so huge –much more than 1 Gbyte in size [15,16]– that parallel computing proves to be essential.

The standard numerical scheme for solving PDEs is based upon an explicit finite difference discretization. More efficient schemes have been specifically designed for nonlinear diffusion [17], though. However, they are complex to implement and, despite their efficiency, they still require to be parallelized [18].

In this work we address the parallelization of AND for its application to denoising of large three-dimensional (3D) volumes in biomedicine and structural cellular biology. We make use of the standard explicit numerical scheme for the discretization. This scheme is commonly used in other fields where PDEs are involved [19] and, as a consequence, the parallel approaches that are presented and discussed here may be valuable for them too.

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 [20]. This section presents local structure determination via structure tensors, the concept of diffusion, a diffusion approach commonly used in image processing and, finally, details of the numerical implementation.

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:

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

where $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, $I_z = \frac{\partial I}{\partial z}$ are the derivatives of the image with respect to x, y and z, respectively.

The eigen-analysis of the structure tensor allows determination of the local structural features in the image [1]:

$$\mathbf{J}(\nabla I) = [\mathbf{v_1}\,\mathbf{v_2}\,\mathbf{v_3}] \cdot \begin{bmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{bmatrix} \cdot [\mathbf{v_1}\,\mathbf{v_2}\,\mathbf{v_3}]^T \tag{2}$$

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 $\mu_1 \geq \mu_2 \geq \mu_3$) 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).

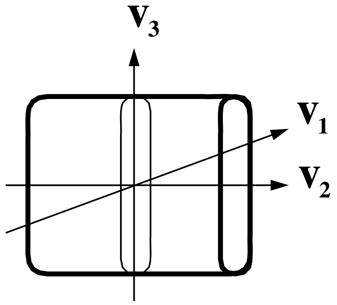

Fig. 1. Local structure found by eigen-analysis of the structure tensor. $\mathbf{v_1}$, $\mathbf{v_2}$, $\mathbf{v_3}$ are the corresponding eigenvectors. $\mathbf{v_1}$ is the direction normal to the local structure.

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* [1]:

$$I_t = \mathrm{div}(\mathbf{D} \cdot \nabla I) \tag{3}$$

where $I_t = \frac{\partial I}{\partial t}$ 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:

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

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

$$\mathbf{D} = [\mathbf{v_1}\,\mathbf{v_2}\,\mathbf{v_3}] \cdot \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \cdot [\mathbf{v_1}\,\mathbf{v_2}\,\mathbf{v_3}]^T \tag{4}$$

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 [1,8,9,10].

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 [1]. 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: $\lambda_1 = g(|\nabla I|)$, $\lambda_2 = 1$ and $\lambda_3 = 1$, 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 [1]; Structures with $|\nabla I| > K$ 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, Eq. (3), can be numerically solved using finite differences. The term $I_t = \frac{\partial I}{\partial t}$ can be replaced by an Euler forward difference approximation. The resulting explicit scheme allows calculation of subsequent versions of the image iteratively:

$$\begin{aligned} I^{s+1} = I^s + \tau \cdot (&\tfrac{\partial}{\partial x}(D_{11}I_x) + \tfrac{\partial}{\partial x}(D_{12}I_y) + \tfrac{\partial}{\partial x}(D_{13}I_z) \\ &+ \tfrac{\partial}{\partial y}(D_{21}I_x) + \tfrac{\partial}{\partial y}(D_{22}I_y) + \tfrac{\partial}{\partial y}(D_{23}I_z) \\ &+ \tfrac{\partial}{\partial z}(D_{31}I_x) + \tfrac{\partial}{\partial z}(D_{32}I_y) + \tfrac{\partial}{\partial z}(D_{33}I_z)) \end{aligned} \tag{5}$$

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 ($\frac{\partial}{\partial x}$, $\frac{\partial}{\partial y}$ and $\frac{\partial}{\partial z}$) is based on central differences.

In this traditional explicit scheme for solving the partial differential equation Eq. (3), the stability is an issue [1]. The maximum time step that is allowed is $\tau \leq 0.5/N_d$, where N_d is the number of dimensions of the problem. In our case, we are dealing with a three-dimensional problem, so $N_d = 3$. 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 of 60-100 iterations is typically used in 3D problems [1,8,9,10] with that value of τ.

For illustration purposes, Fig. 2 shows the result of the application of 60 iterations of AND to a volume of a mitochondrion, a cell organelle, that was obtained by electron microscope tomography [16]. The enhancement in visualizing a slice of the volume is apparent (left: slice from the original volume; right: slice from the filtered volume).

2.5 The Algorithm of the Diffusion Approach

In this work, we propose an optimized algorithm for solving the PDE in Eq. (5) that computes the volume by z-planes, where –without loss of generality– the z-axis is the

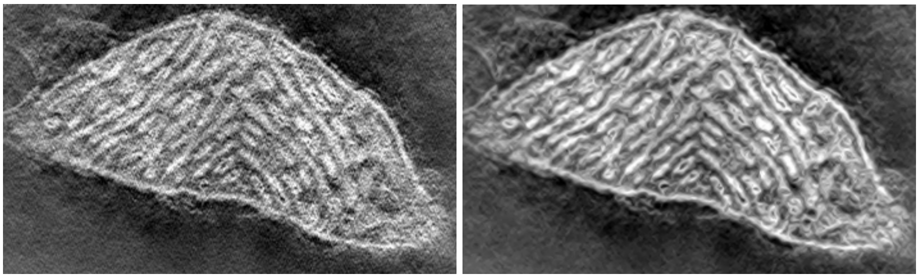

Fig. 2. Left: a slice from a volume of a mitochondrion obtained by electron microscope tomography; Right: the same slice from the volume filtered with anisotropic nonlinear diffusion

direction of the larger image dimension $N_z >= N_x, N_y$. The proposed sequential algorithm consists of the following steps:

```
Do $s = 0 \ldots n-1$
        Do $k = 1 \ldots N_z$ /* processing the volume by z-planes */
          1. Compute the structure tensor $\mathbf{J}_k^s$ (Eqs. (1) and (2)).
          2. Compute the diffusion tensor $\mathbf{D}_k^s$ from $\mathbf{J}_k^s$ (Eq. (4)).
          3. Compute the resulting z-plane of the image $\mathbf{I}_k^{s+1}$, at step $(s+1)$ from step
             $s$ by means of Eq. (5). The resulting z-plane of the image corresponds to
             the diffusion time $t_{(s+1)} = (s+1)\tau$
        End Do
End Do
```

where s and k denote the index of the iteration and the index of the z-plane respectively. The algorithm is executed iteratively for a number of iterations n. The final image is obtained after a total diffusion time $T = n\tau$. Note from Eq. (5) that $\mathbf{I}_k^{s+1}$ is only a function of $\mathbf{I}_{k-2}^s$, $\mathbf{I}_{k-1}^s$,$\mathbf{I}_k^s$, $\mathbf{I}_{k+1}^s$, $\mathbf{I}_{k+2}^s$. Our implementation minimizes the memory usage by allocating and computing only the necessary data for updating each single z-plane. Hereinafter, the body of this nested loop is denoted as $\mathbf{I}_k^{s+1} = AND(\mathbf{I}_k^s)$.

3 Parallel Implementation of AND

In this work, AND have been implemented using three parallel programming models: (1) shared address space model based on Pthreads,(2) message passing model, where MPI is applied for message passing between different processors; and (3) hybrid model that uses Pthreads at the node level while MPI is only applied for message passing between processors from different and/or the same nodes. Essentially, the parallel strategies are based on domain decomposition. They consist in 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. At the end of every iteration, depending on the specific implementation, boundaries planes must be updated from neighbor processors for their processing in the subsequent iteration. Next, the main characteristics of every parallel code are described:

-Pure Pthreads code. A single thread is mapped onto each processor of the system. The shared address space model allows all the processors to access the shared whole 3D volume. The thread running in each processor updates its corresponding z-planes of the shared volume. Transparently, the neighbor threads then have their boundary 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 boundary z-planes before the neighbors modify them.

-Pure MPI code. Here, one process is spawned on each processor. Each processor then updates its own block of z-planes. The update of a given local z-plane I_k^{s+1} is only a function of I_k^s and its four neighbor z-planes, I_{k-1}^s, I_{k-2}^s, I_{k+1}^s and I_{k+2}^s. Updating the boundary z-planes of the block would imply many communications during one update step. To avoid excessive communications, each processor allocates four additional z-planes to hold the two neighbor z-planes of the two boundaries. At the end of each iteration, the processor then exchanges the updated four boundary z-planes with the immediate neighbor processors by MPI point-to-point communications.

-Hybrid code. 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 is shared by all the threads running in the node. Every thread updates its own subset of the block of z-planes, similarly to the shared address space strategy above described. At the end of the iteration, all the threads running in a node are joined. The boundary z-planes are then exchanged among the immediate neighbor nodes by MPI point-to-point communications. The outline of the hybrid code would be as follows:

```
1. Distribute $I^0$ among nodes, $N_z^{nd} = \lceil N_z/P \rceil + 4$ planes are assigned to each node.
2. Do $s = 0 \ldots n-1$
   (a) Each thread initializes its auxiliary data structures.
   (b) Do $k = 1 \ldots N_z^{thr} = \lceil (N_z^{nd} - 4)/T \rceil$ /* each thread */
         $I_k^{s+1} = AND(I_k^s)$
       End Do
   (c) Interchange boundary z-planes between neighbor nodes.
3. End Do
4. Collect the image.
```

where N_z^{nd} and N_z^{thr} denote the local number of z-planes of the volume in every node and in every thread respectively, P and T denotes the number of nodes and processors inside one node respectively, and I^0 denotes the original 3D volume, 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: (1) at the node level, since the total number of z-planes is distributed among nodes, and (2) at the processor level inside the node, as each thread updates its subset of z-planes by applying the AND process.

4 Evaluation of the Parallel Implementation of AND

In this section, we evaluate the performance of three parallel implementations of the AND method: (1) Pure MPI AND-code, (2) Pure Pthreads AND-code and (3) Hybrid AND-code. The evaluation has been carried out on two parallel platforms:

-*Distributed Shared Memory (DSM) platform* SGI Altix 3700 Bx2 of 8 processors 1600 MHz Intel Itanium 2 Rev with 128 GB RAM. The Altix 3700 computer system is based on a Distributed Shared Memory architecture and uses a cache-coherent Non-Uniform Memory Access (NUMA) where the latency of processors to access to local memory is lower than the latency to access to global memory (or remote memory) [21].

-*Cluster of symmetric biprocessors* of Intel(R) Xeon(TM) 3.06 GHz with 2 GB RAM, 512 KB cache. Nodes are interconnected via two Gigabit Ethernet networks, one for data (NFS) and the other for computation. The architecture of this cluster is based on a UMA access, where all processors have equally fast (symmetric) access to the memory in the node.

Dimensions of volumes in biomedicine and structural cellular biology usually range between 256x256x256 and 640x640x640. 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 [9,10]. In this work, two test volumes with cubic symmetry of sizes 256x256x256 and 640x640x640 have been selected to carry out the evaluation process. Hereinafter, these volumes will be referenced by the size of their edges.

4.1 Distributed Shared Memory Platform: Altix 3700

Let *mpi* be the number of MPI processes and pt the number of Pthreads processes. To evaluate the hybrid implementation for a fixed number of processors p, several combinations of values of mpi and pt are possible. Experimental performance results were measured for several combinations of mpi and pt for a fixed p, obtaining similar behavior. In the results shown here, we focus on the case with $pt = 2$ Pthreads processes, and we only increment the number of MPI processes. Fig. 3 shows the speedup achieved by the pure MPI, pure Pthreads and hybrid implementations, for the two volumes, on a 8-processor SGI Altix 3700 Bx2. As it can be seen, in general the three parallel implementations have very good performance. They all approach the ideal linear speedup, with slightly better behavior for the message passing implementation. For the volume 640, some curves exhibit slight levels of superspeedup. Finally, the volume size has turned out to have a very low influence on the speedup. The excellent behavior shown by the message passing version may be thanks to the high speed interconnection technology used in this computer [21]. In summary, the three parallel strategies present very good levels of scalability on this computer platform.

4.2 Cluster of Symmetric Biprocessors

On the cluster of symmetric biprocessors, the performance has been evaluated only for two models: message passing model and hybrid model with $pt = 2$, since on this platform the evaluation of the shared address space model is limited to two processors into one node. The scalability of both models has been analyzed by means of the speedup measurements.

Traditionally, the speedup is only referred to the sequential runtime. Recently, a general concept of speedup has been introduced [22], where the parallel runtime is used as a reference instead. In this evaluation, this concept is taken into account to evaluate the performance of AND on the cluster of SMPs. Specifically, for the volume 640, the parallel runtime with four processors is considered as a reference, since it was not possible

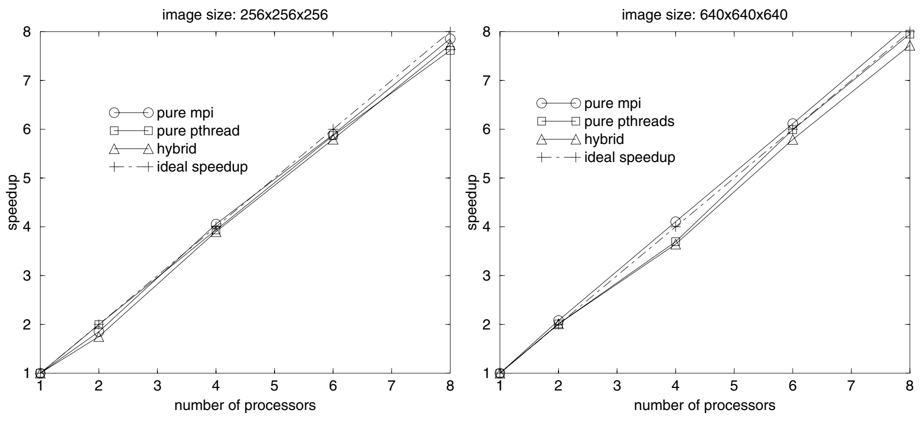

Fig. 3. Speedup on a SGI Altix 3700 Bx2 of the pure MPI, Pure Pthreads and hybrid codes for the volumes 256 and 640

to run the codes on fewer than four processors with these volume sizes. Meanwhile, the sequential runtime is used as reference for the smaller volume 256.

Fig. 4 shows the speedup achieved by the pure MPI code and the hybrid code, for the test volumes, on the cluster of SMPs described above. In general terms, both strategies yield good results, with better performance for the hybrid strategy. It is evident from these figures that the hybrid strategy yields better scalability than the strategy based on message passing, specially for increasing number of processors.

In order to explain the better behavior of the hybrid strategy compared to the message passing one, additional measures of communication times have been obtained as well (results not shown here). Clearly the penalty due to the communications is stronger on the message passing implementation than on the hybrid one, specially as the number of

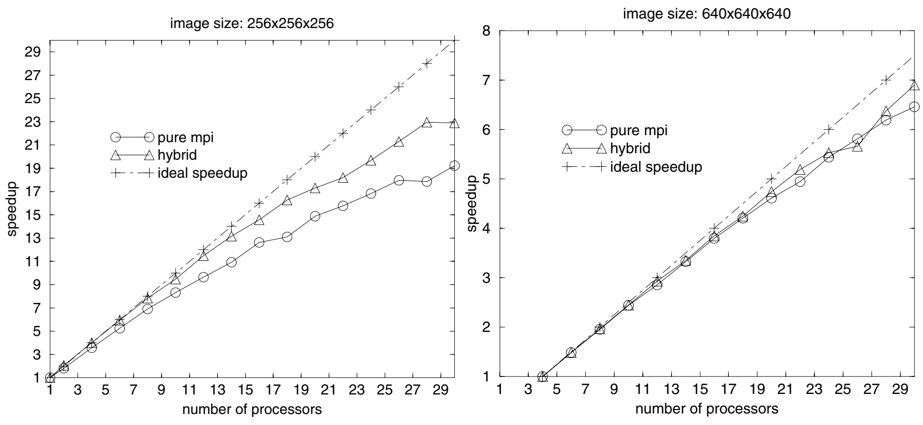

Fig. 4. Speedup on a cluster of SMPs of the pure MPI and hybrid codes for the volumes 256 and 640

processors increases since only communications between pairs of symmetric processors are involved for the hybrid code.

The influence of the problem size on the performance was also analyzed, and the conclusion is that the volume size proves to be relevant on this platform. This behavior is justified by two factors. First, the computational complexity depends linearly on the volume size whereas the amount of communications is proportional to the size of a single z-slice. Therefore, for small volumes the penalties from communications are relevant, specially for the pure MPI code. Second, the local memory hierarchy management improves for larger volume sizes and has a stronger impact in the scalability of both strategies. Therefore, any increase in the problem size is expected to imply a direct improvement in the speedup of both strategies.

5 Conclusions

In this work, we have presented parallel implementations of AND, using three strategies based on: (1) shared address space, (2) the message passing paradigm and (3) a hybrid approach. The evaluation has been carried out on two different architectures: (1) a Distributed Shared Memory platform based on cc-NUMA access and (2) a cluster of Symmetric Biprocessors based on UMA access. In view of the results, we can conclude that the parallel algorithms present good levels of scalability. Furthermore, the evaluation allows us to draw the conclusion that for DSM platforms like the Altix 3700 Bx2, all paradigms yield better and similar speedup. Consequently there is no favorable paradigm for this platform. However, for clusters of SMPs the hybrid paradigm (Pthreads+MPI) is more suitable than a strategy based solely on the message passing paradigm.

Acknowledgments

The authors wish to thank Dr. Guy Perkins (National Center for Microscopy and Imaging Research, San Diego, USA) for kindly providing the mitochondrion dataset.

References

1. Weickert, J.: Anisotropic Diffusion in Image Processing. Teubner (1998)
2. Perona, P., Malik, J.: Scale space and edge detection using anisotropic diffusion. IEEE Trans. Patt. Anal. Mach. Intel. **12** (1990) 629–639
3. Weickert, J.: Coherence-enhancing diffusion filtering. Int. J. Computer Vision **31** (1999) 111–127
4. J. Weickert: Coherence-enhancing diffusion of colour images. Image and Vision Computing **17** (1999) 201–212
5. Gerig, G., Kikinis, R., Kubler, O., Jolesz, F.A.: Nonlinear anisotropic filtering of MRI data. IEEE Trans. Med. Imaging **11** (1992) 221–232
6. Bajla, I., Hollander, I.: Nonlinear filtering of magnetic resonance tomograms by geometry-driven diffusion. Machine Vision and Applications **10** (1998) 243–255

7. Ghita, O., Robinson, K., Lynch, M., Whelan, P.F.: MRI diffusion-based filtering: A note on performance characterisation. Comput. Med. Imaging Graph. **29** (2005) 267–277
8. Frangakis, A.S., Hegerl, R.: Noise reduction in electron tomographic reconstructions using nonlinear anisotropic diffusion. J. Struct. Biol. **135** (2001) 239–250
9. Fernandez, J.J., Li, S.: An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms. J. Struct. Biol. **144** (2003) 152–161
10. Fernandez, J.J., Li, S.: Anisotropic nonlinear filtering of cellular structures in cryo-electron tomography. Computing in Science and Engineering **7**(5) (2005) 54–61
11. Medalia, O., Weber, I., Frangakis, A.S., Nicastro, D., Gerisch, G., Baumeister, W.: Macromolecular architecture in eukaryotic cells visualized by cryoelectron tomography. Science **298** (2002) 1209–1213
12. Grunewald, K., Desai, P., Winkler, D.C., Heymann, J.B., Belnap, D.M., Baumeister, W., Steven, A.C.: Three-dimensional structure of herpes simplex virus from cryo-electron tomography. Science **302** (2003) 1396–1398
13. Beck, M., Forster, F., Ecke, M., Plitzko, J.M., Melchior, F., Gerisch, G., Baumeister, W., Medalia, O.: Nuclear pore complex structure and dynamics revealed by cryoelectron tomography. Science **306** (2004) 1387–1390
14. Cyrklaff, M., Risco, C., Fernandez, J.J., Jimenez, M.V., Esteban, M., Baumeister, W., Carrascosa, J.L.: Cryo-electron tomography of vaccinia virus. Proc. Natl. Acad. Sci. USA **102** (2005) 2772–2777
15. Fernandez, J.J., Lawrence, A., Roca, J., Garcia, I., Ellisman, M., Carazo, J.: High performance electron tomography of biological specimens. J. Struct. Biol. **138** (2002) 6–20
16. Fernandez, J.J., Carazo, J., Garcia, I.: Three-dimensional reconstruction of cellular structures by electron microscope tomography and parallel computing. J. Paral. Distr. Computing **64** (2004) 285–300
17. Weickert, J., ter Haar Romeny, B.M., Viergever, M.A.: Efficient and reliable schemes for nonlinear diffusion filtering. IEEE Trans. Image Processing **7** (1998) 398–410
18. Bruhn, A., Jakob, T., Fischer, M., Kohlberger, T., Weickert, J., Bruning, U., Schnorr, C.: High performance cluster computing with 3D nonlinear diffusion filters. Real-Time Imaging **10** (2004) 41–51
19. Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T.: Numerical Recipes: The Art of Scientific Computing. Cambridge University Press (1992)
20. Barash, D.: A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation. IEEE Trans. Patt. Anal. Mach. Intel. **24** (2002) 844–847
21. Dunigan, T., Vetter, J., Worley, P.: Performance evaluation of the SGI Altix 3700. In: Proceedings of the IEEE Intl. Conf. Parallel Processing, ICPP. (2005) 231–240
22. Akl, S.G.: Superlinear performance in real-time parallel computation. The Journal of Supercomputing **29**(1) (2004) 89–111