Universität des Saarlandes

Fachrichtung 6.1 – Mathematik

Preprint Nr. 87

High Performance Cluster Computing with 3-D Nonlinear Diffusion Filters

Andrés Bruhn, Tobias Jakob, Markus Fischer, Timo Kohlberger, Joachim Weickert, Ulrich Brüning and Christoph Schnörr

Saarbrücken 2003

Fachrichtung 6.1 – Mathematik
Universität des Saarlandes

Preprint No. 87
submitted: May 29, 2003

High Performance Cluster Computing with 3-D Nonlinear Diffusion Filters

Andrés Bruhn
Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science, Saarland University, Building 27.1, 66041 Saarbrücken, Germany
bruhn@mia.uni-saarland.de

Tobias Jakob
Computer Architecture Group, Faculty of Mathematics and Computer Science, University of Mannheim, 68131 Mannheim, Germany
tjakob@mufasa.informatik.uni-mannheim.de

Markus Fischer
Computer Architecture Group, Faculty of Mathematics and Computer Science, University of Mannheim, 68131 Mannheim, Germany
mfischer@mufasa.informatik.uni-mannheim.de

Timo Kohlberger
Computer Vision, Graphics and Pattern Recognition Group, Faculty of Mathematics and Computer Science, University of Mannheim, 68131 Mannheim, Germany
tkohlber@uni-mannheim.de

Joachim Weickert
Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science, Saarland University, Building 27, 66041 Saarbrücken, Germany
weickert@mia.uni-saarland.de

Ulrich Brüning
Computer Architecture Group, Faculty of Mathematics and Computer Science, University of Mannheim, 68131 Mannheim, Germany
ulrich@mufasa.informatik.uni-mannheim.de

Christoph Schnörr
Computer Vision, Graphics and Pattern Recognition Group, Faculty of Mathematics and Computer Science, University of Mannheim, 68131 Mannheim, Germany
schnoerr@uni-mannheim.de

Edited by
FR 6.1 – Mathematik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

Fax: + 49 681 302 4443
e-Mail: preprint@math.uni-sb.de
WWW: http://www.math.uni-sb.de/

Abstract

This paper deals with parallelisation and implementation aspects of PDE-based image processing models for large cluster environments with distributed memory. As an example we focus on nonlinear diffusion filtering which we discretise by means of an additive operator splitting (AOS). We start by decomposing the algorithm into small modules that shall be parallelised separately. For this purpose image partitioning strategies are discussed and their impact on the communication pattern and volume is analysed. Based on the results we develop an algorithmic implementation with excellent scaling properties on massively connected low latency networks. Test runs on a high–end Myrinet cluster yield almost linear speedup factors up to 209 for 256 processors. This results in typical denoising times of 0.5 seconds for five iterations on a $256 \times 256 \times 128$ data cube.

AMS 2000 Subject Classification: 68T45, 65Y05, 65N06
Key Words: diffusion filtering, additive operator splitting, cluster computing.

Contents

1 Introduction

Partial differential equations (PDEs) form the basis of a number of recent methods in the fields of image processing and computer vision. In this paper we focus on nonlinear diffusion techniques that allow to denoise images while preserving edges. This property makes them useful for various restoration and segmentation purposes. Nonlinear diffusion models were first introduced by a work of Perona and Malik [22]. After some years their original model was improved by Catté et al. [4] from both a theoretical and practical viewpoint, and anisotropic extensions with a diffusion tensor [31] followed.

Many nonlinear diffusion algorithms are based on the simplest numerical scheme, an explicit finite difference discretisation. While such schemes are easy to implement, they require small time steps for stability reasons. Hence, many iterations are needed to reach some interesting diffusion time, and the entire procedure is relatively inefficient. This has triggered a number of researchers to look for alternative algorithmic realisations of nonlinear diffusion filtering and related variational approaches.

These alternatives include three-level methods [8], semi-implicit approaches [4, 11] and their multiplicative [36] or additive operator splitting variants [37], multigrid methods [1], finite element techniques [2, 16, 23], finite and complementary volume methods [11], numerical schemes with wavelets as trial functions [7, 8], pseudospectral methods [8], lattice Boltzmann techniques [15], and stochastic simulations [24]. Approximations in graphics hardware have been considered in [26], and realisations on analog hardware are discussed in [9, 21]. Related variational approaches have been treated using linearisations by means of auxiliary variables [5]. Also here it is possible to use adaptive finite elements [27] and to consider analog hardware realisations [39]. Parallel implementations on shared memory systems are studied in [12, 13].

In the present paper we shall focus on additive operator splitting (AOS) schemes for nonlinear diffusion filters. These specific semi-implicit schemes have been first introduced to image analysis in [37]. They have been used for medical imaging problems [25], for regularisation methods [33], image registration [6] and for optic flow computations [35]. Recently, they have also been used successfully for a number of active contour approaches [10, 17, 19, 20, 32, 34]. The basic idea behind AOS schemes is to decompose a multi-dimensional problem into one-dimensional ones that can be solved very efficiently. Then the final multi-dimensional solution is approximated by averaging the one-dimensional solutions. AOS schemes inherit a number of favourable properties from the continuous diffusion processes and they reveal linear complexity [37]. The usefulness of AOS ideas has also been shown in a

number of other applications ranging from Navier–Stokes equations [18, 29] to sandpile growth simulations [14]. It seems that Navier–Stokes equations have constituted one of their historically first application domains.
The use of AOS schemes has triggered first parallel implementations for diffusion filtering [38]. At that time, however, the development of network architectures did not allow the efficient use of distributed memory systems for such communication-intensive problems. For this reason these approaches generally stayed confined to systems based on *shared* memory. In recent years a rapid progress in this sector changed the situation completely. High performance cluster systems with massively connected low latency networks were built throughout the world. There are two reasons for this development: First cluster systems are much more attractive to customers, since they are less expensive than comparable shared memory systems. This has increased their availability for research purposes. Moreover, the number of processors is not limited by such severe hardware restrictions than in the case of shared memory systems, thus allowing larger scaling possibilities. In order to exploit this potential, parallelisation approaches must fit the underlying network topology.
The goal of the present paper is to show that a 3-D nonlinear diffusion process can be parallelised in such way, that it reveals excellent scaling properties regarding both computation and communication on a *distributed* memory system.
The paper is organised as follows. In Section 2 a review on diffusion filtering and the AOS scheme is given. Furthermore a modular decomposition before parallelisation is shown. In Section 3 partitioning and communication models are discussed. Relevant parallelisation and implementation details of our approach are explained in Section 4. In Section 5 obtained results on a high performance cluster are presented. The summary in Section 6 concludes this paper. A preliminary version of this paper has been presented at a symposium [3].

2 Nonlinear Isotropic Diffusion and AOS

In the following we give a short review of the nonlinear diffusion model of Catté et al. [4]. A grey value image f is considered as a function from a given domain $\Omega \subset \mathbb{R}^m$ into $\mathbb{R}$. In our case we have $m \in \{2, 3\}$, which corresponds to 2-D and 3-D images. The basic nonlinear diffusion problem then reads:
Find a function $u(x, t)$: $\Omega \times \mathbb{R}_0^+ \to \mathbb{R}$ that solves the diffusion equation

$$\partial_t u = \operatorname{div}\Big(g(|\nabla u_\sigma|^2)\nabla u\Big) \quad \text{on} \quad \Omega \times \mathbb{R}_0^+ \tag{1}$$

with f as initial value,

$$u(x,0) = f(x) \quad \text{on} \quad \Omega \tag{2}$$

and reflecting boundary conditions:

$$\partial_n u = 0 \quad \text{on} \quad \partial\Omega \times \mathbb{R}_0^+. \tag{3}$$

where $u_\sigma = K_\sigma * u$ denotes the convolution of u with a Gaussian of standard deviation *sigma*, n is a normal vector perpendicular to $\partial\Omega$, and the diffusivity g is a nonnegative decreasing function with $g \in C^\infty[0,\infty)$. The solution $u(x,t)$ is a family of images, where the diffusion time t acts as a scale parameter. An example illustrating the performance of this diffusion filter is given in Figure 1. For more information on theoretical aspects of nonlinear diffusion filtering the interested reader may refer to [31].

Figure 1: *From left to right:* (a) Test image with grey scale range $[0, 255]$ degraded by Gaussian noise with standard deviation $\sigma_n = 30$. (b) Image denoised by the nonlinear diffusion filter, 5 iterations with $\sigma = 2.5$, $\lambda = 0.01$ and $\tau = 20$.

Since such nonlinear diffusion equations cannot be solved analytically, numerical approximations are required. In [37] a finite difference scheme based on an additive operator splitting (AOS) technique is used for this purpose. This AOS technique is the basis for our parallelisation efforts. It is an extension on the semi-implicit scheme for nonlinear diffusion filtering and can be described as

$$u^{k+1} = \frac{1}{m} \sum_{l=1}^{m} (I - m\tau A_l(u_\sigma^k))^{-1} u^k \tag{4}$$

where u^k is a vector with the grey values at all pixels as components. The iteration index k refers to the diffusion time $t = k\tau$ where τ is the time step size. The tridiagonal matrix A_l is a discretisation of the divergence expression along the l-th coordinate axis. Let $\mathcal{N}_l(i)$ denote the set of neighbours of pixel i in direction of axis l, let h_l be the corresponding grid size and let g_i^k stand for the evaluated diffusivity at pixel i of the presmoothed image u_σ^k, then the matrix $A_l(u_\sigma^k)$ is given by

$$(A_l(u_\sigma^k))_{ij} := \begin{cases} -\sum\limits_{n \in \mathcal{N}_l(i)} \frac{g_i^k + g_n^k}{2h_l} & \text{if} \quad j = i \\ \frac{g_i^k + g_l^k}{2h_l} & \text{if} \quad j \in \mathcal{N}_l(i) \\ 0 & \text{else} \end{cases} \tag{5}$$

Therefore, in each iteration step, the AOS method requires the solution of m tridiagonal linear systems of equations. Each system describes diffusion along one coordinate direction and may even be decomposed into smaller tridiagonal systems. The final result at the next time level is obtained by averaging these 1-D diffusion results. A splitting into 1-D diffusions offers significant computational advantages: The corresponding tridiagonal systems can be solved in linear complexity by means of the so-called Thomas algorithm [30], a specific variant of the Gaussian algorithm; see [28, 37] for further details. Typical AOS schemes are one order of magnitude more efficient than simple diffusion algorithms. Although they are stable for all time step sizes τ one usually limits the step size for accuracy reasons. Hence, the scheme is applied in an iterative way in order to reach some interesting stopping time.

2.1 Algorithmic Decomposition

The following algorithmic steps can easily be derived from the iteration instruction for the AOS Scheme (4).

1. Perform a Gaussian presmoothing of u using $u_\sigma^k = K_\sigma * u^k$

2. Compute the derivatives $\|\nabla u_\sigma^k|^2$ and the diffusivities $g(|\nabla u_\sigma^k|^2)$.

3. Solve the tridiagonal systems $(I - m\tau A_l(u_\sigma^k))\, u_l^{k+1} = u^k$ and average the results: $u^{k+1} = \frac{1}{m} \sum\limits_{l=1}^{m} u_l^{k+1}$

3 Parallelisation Models

The following parallelisation models are based on image partitioning. This allows parallel execution of fast sequential algorithms instead of applying slower parallel variants to the complete image domain.

3.1 Communication Models

A large part of image processing algorithms consist of neighbourhood operations. This raises problems at partition boundaries, since required information is missing. Let us now discuss two communication models to handle this problem: repartitioning and boundary communication.

Repartitioning. The basic idea of the repartitioning strategy is to find an appropriate partitioning for each operation, such that the problem of missing neighbourhood information does not occur. Therefore partitions have to be relocated and reshaped by means of communication. In many cases this communication involves data exchanges between all processes, the so called *all-to-all communication*. For large partition numbers such a connection–intensive communication pattern makes high demands to the network topology. Whether the network can satisfy these demands or not is reflected in a scaling of bandwith (pairwise disjunct communication) or a rise of communication time. For massively connected low latency networks the first case does apply. Counterexamples are Gigabit Ethernet due to its high latency as well as connection limited network topologies like e.g. token rings or computing grids.
Taking a look at the total communication volume the importance of this scaling property becomes obvious. Since non–overlapping partitions are used, each pixel is sent and received by no more than one process. Thus, the communication behavior imposes a limit to the total communication volume that is given by the image size. The number of processes and the required neighbourhood can only affect the communication volume within this scope. Hence, each scaling of bandwith is passed on to the communication time.

Boundary Communication. Keeping existing partitions the second communication model simply exchanges the missing neighbourhood information. One should note that this implies a dependency of the total communication volume on two unknowns: The number of partitions as well as the boundary size.
For moderate values of both parameters, the communication is limited to its adjacent segments. In this case the total communication volume may

drop significantly beyond that of a repartitioning strategy. Moreover such a simple communication pattern has a second advantage. Since it makes less demands to the network topology than the previously discussed all–to–all communication, also weakly connected cluster systems do benefit from a bandwidth scaling effect. Even for high latency networks this strategy is favourable due to its rather large message size that results from the limited communication pattern.
However, larger boundary sizes and partition numbers do change the situation completely. Then boundary–volume ratios deteriorate, communication patterns may require extensions to further partitions and finally an inefficient parallelisation remains. This is reflected in the worst case communication volume that is only limited by $(n-1)$ times the image size, where n is the number of partitions. Hence boundary exchange does only address operations that require information from a small neighbourhood.
In parallelised software for scientific simulations, boundary exchange is a frequently used communication model. Since the underlying theory allows a free scaling of the problem size, the loss of efficiency can be circumvented by a finer sampling of the continuous model. Thus, along with an increased accuracy, larger partition sizes and thefore better scaling properties for larger cluster systems are obtained. In the area of image processing such possibilities are not given. Accuracy and problem size are determined at the moment of image acquisition.

3.2 Partition Models

In addition to the communication models appropriate image partitioning strategies have to be chosen. In general, cuboid partitions are preferred since they can be realised with commonly used data structures and are easier to handle. There are two partitioning models that result in such cuboid partitions.

Slice Partitioning. As the name anticipates the main idea of this strategy is to partition an image along one single direction. Thus no further boundaries arise. Operations that are separable or do not require neighbourhood information from all directions can exploit this property.
However, there are two minor disadvantages of this strategy. First, the maximum number of partitions is limited by the number of pixels in the direction of partitioning, and secondly, slices have an evidently bad boundary–volume ratio. While the first drawback is only relevant for small image sizes, the second one has no relevance if repartitioning is applied.

Mesh Partitioning. This strategy focuses on partitioning an image along all directions. Thus the largest theoretical scalability is achieved, since the maximum number of partitions is only limited by the total number of pixels. Its main disadvantage is the occurrence of boundaries in all directions. In our case this drawback is quite severe, since the performance of certain operations lives on their separability property.
A special case of mesh partitioning is cube-like partitioning. Thereby an image is partitioned in such a way, that the sum of all partition boundaries is minimised. Obviously this partition strategy should be used when it comes to the exchange of boundary information.

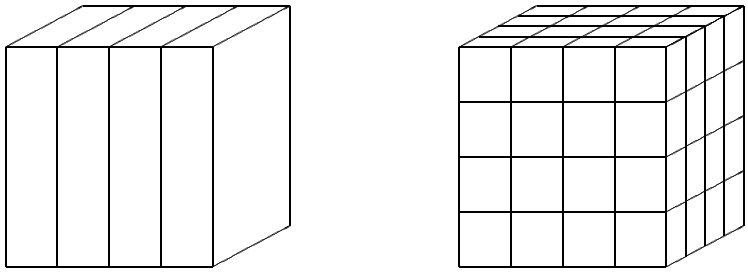

Figure 2: Partition Models. *From left to right:* (a) Slice Partitioning. (b) Mesh Partitioning.

4 Parallelisation Details

In this section all parallelised modules are discussed in detail. A global overview of all computation and communication steps is given in Figure 4. It illustrates one iteration step of the implemented algorithm.

Module 1 : Gaussian Convolution. The Gaussian convolution is implemented exploiting separability and symmetry as well as optimising the computational sequence for optimal cache use. The convolution masks are obtained by sampling the continuous Gaussian

$$G(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \tag{6}$$

and truncating it at 3 times the standard deviation. Then the mask is renormalised such that its weight sum up to 1. In our approach the repartition-

ing strategy is used in combination with slice partitioning. Thus, Gaussian convolution in two out of three directions can be performed without communication effort (Fig. 3a). Only smoothing in the third direction requires a previous repartitioning step (Fig. 3b). Moreover, this implementation allows large values for the standard deviation σ, since no boundary exchange takes place.

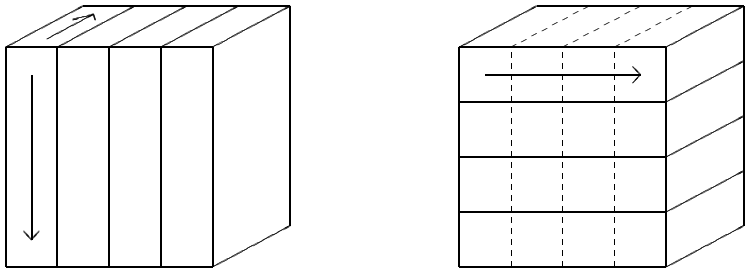

Figure 3: Partitioning scheme for module 1. *From left to right:* (a) Data cube before repartitioning step 1. (b) Data cube after repartitioning step 1. Arrows show the directions in which Gaussian convolutions are performed. Solid lines are boundaries of current partitions while dashed lines refer to boundaries of previously used partitions.

Module 2 : Derivatives and Diffusivity. Derivatives within the diffusivity are computed using central differences. Since this uses stencils of type $\frac{1}{2h}(-1, 0, 1)$, where h denotes the grid size, the boundary size is limited to 1. Besides, the computation of the diffusivity values demands matching partitions for all derivatives. Both aspects favour a boundary exchange strategy. Although cube-like partitioning would be desirable, a change of the partition model at the cost of two repartitioning steps is obviously not profitable. Hence, slice partitioning combined with boundary communication is implemented. Since parallelism is achieved via image partitioning, derivatives are computed sequentially for all directions. Again, the exchange of neighbourhood information takes place after the computation for two out of three directions is completed (Fig. 3a-b). Finally the diffusivity values are computed in place based on

$$g(|\nabla u_\sigma|^2) := \frac{1}{1 + |\nabla u_\sigma|^2/\lambda^2} \tag{7}$$

where λ is a contrast parameter.

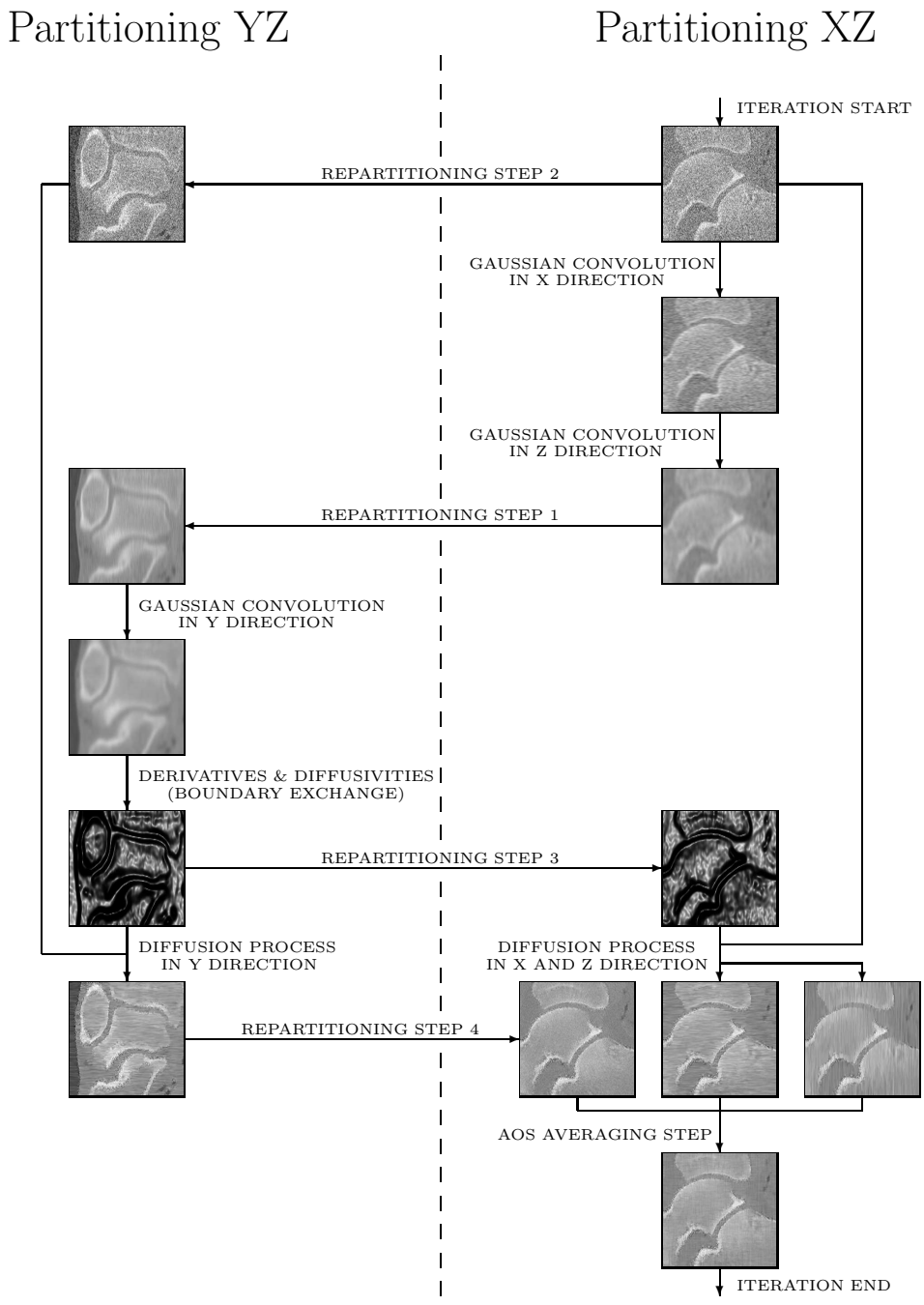

Figure 4: Flow diagram illustrating one iteration step of the implemented algorithm. Column headings give information on the direction of data partitioning and the cut direction of presented slices for intermediate results. Computation and communication steps are symbolised by arrows.

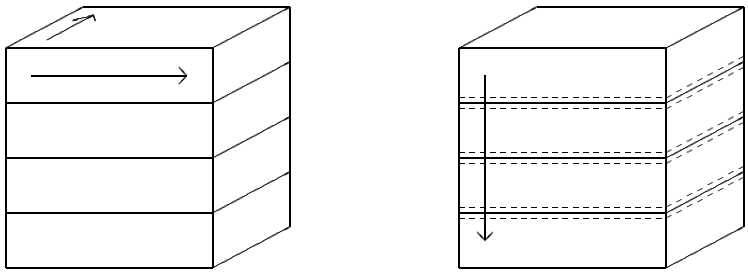

Figure 5: Partitioning scheme for Module 2. *From left to right:* (a) Data cube before boundary exchange. (b) Data cube after boundary exchange. Arrows show the directions in which derivatives are computed. Solid lines are boundaries of current partitions while dashed lines refer to boundaries that have been exchanged.

Module 3 : Diffusion and AOS. As discussed before, AOS offers parallelism on two different levels. First, it allows to decouple the diffusion processes for each direction (coarse grain parallelism). For the same reason as in the case of the derivative computation, this property will not be exploited directly for parallelisation purposes. Of major importance is the fact, that the huge linear tridiagonal equation systems for each diffusion direction can be decomposed into many small independent equation systems of same style (mid grain parallelism). Since each of these systems corresponds to the diffusion process along a complete image line in the diffusion direction, the use of a common boundary exchange approach makes no sense. Instead slice partitioning in combination with the repartitioning strategy seems desirable. Moreover, this implementation allows the application of fast sequential solvers such as the Thomas algorithm [30, 37]. It uses an LR decomposition, a forward substitution as well as a backward substitution step. Thus, special variants for a boundary exchange strategy could not have been developed without loss of parallelism and performance. However, even in the case of repartitioning the parallelisation effort is large: In order to compute the diffusion process for one direction, matching partitions for the original image *and* the diffusivity values are required. Moreover, at least two of these partitions pairs are needed to cover all three diffusion directions. Therefore not only the original image has to be repartitioned (Fig. 6a), but also the corresponding diffusivity data (Fig. 6b). Finally combining the results of all three diffusion processes – the averaging step in the AOS scheme – requires a third repartitioning.

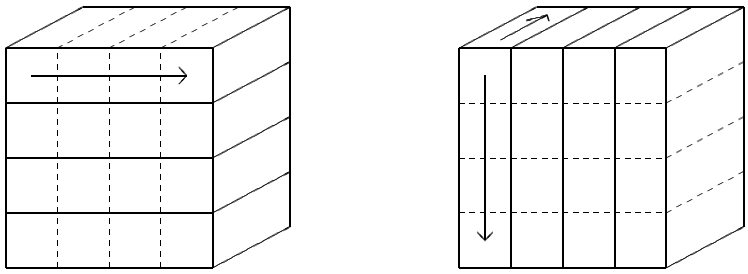

Figure 6: Partitioning scheme in Module 3. *From left to right:* (a) Data cube with repartitioned original image data (repartitioning step 2) now matching diffusivity partition shown in Fig. 5b. (b) Data cube with repartitioned diffusivities (repartitioning step 3) now matching original image partition shown in Fig. 3a. Arrows show the directions in which diffusion is computed. Solid lines are boundaries of current partitions. Dashed lines refer to previously used partitions.

5 Results

Our test runs have been performed on the Score III cluster of the *RWCP* (Real World Computing Partnership) at the Tsukuba Research Center, Japan. Running a modified Linux 2.4 SMP Kernel it consists of 524 nodes with two PIII 933 MHz processors each. Focusing on distributed memory systems only one processor per node has been used at a time. The cluster is fully connected to a CLOS network using a Myrinet2000 network interface. Due to its performance it is ranked 90th in the November 2002 TOP 500 list of supercomputers.

As one can see from Figure 7 considerations regarding the parallelisation for a specific network architecture do pay off. The obtained results demonstrate an excellent, almost linear scaling behavior up to 256 nodes with a top speedup of 209. This equals 82% of the theoretical maximum. The corresponding runtimes divided in computation and communication effort can be found in Table 1. For all test runs a 32-bit float data cube of size $256 \times 256 \times 128$ has been used resulting in communication volumes up to 1.83 Gbyte per second. These numbers show the importance of a sophisticated algorithm design that allows bandwith scaling up to a large number of processors.

This scaling property is reflected in the percental distribution, that shows only a moderate increase of the communication part. Even in the case of 256 processors this ratio does hardly exceed one quarter of the runtime.

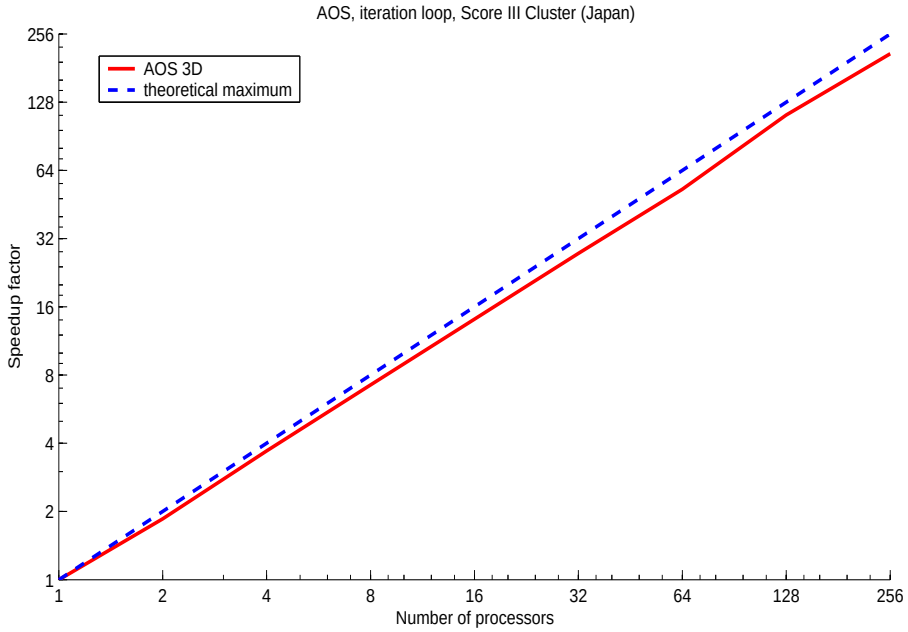

Figure 7: Speedup Chart

Figure 8 gives a qualitative example for nonlinear diffusion filtering on medical data sets. Although the image was severely degraded by Gaussian noise of standard deviation $\sigma_n = 30$, the algorithm is still able to recover basic structures. With a runtime of 0.5 seconds, this results in typical processing rates of two data cubes per second.

Table 1: Runtimes for AOS 3-D , 10 iterations

CPUs	Runtime [s]	Comp. [s]	Comm. [s]	Comp. [%]	Comm. [%]
1	212.741	212.741	0.000	100.000	0.000
2	114.625	106.205	8.420	92.654	7.346
4	57.534	52.221	5.513	90.766	9.234
8	29.401	26.123	3.278	88.851	11.149
16	15.065	13.471	1.594	89.420	10.580
32	7.731	6.753	0.978	87.350	12.650
64	4.029	3.333	0.696	82.725	17.275
128	1.894	1.550	0.344	81.837	18.163
256	1.017	0.745	0.272	73.255	26.745

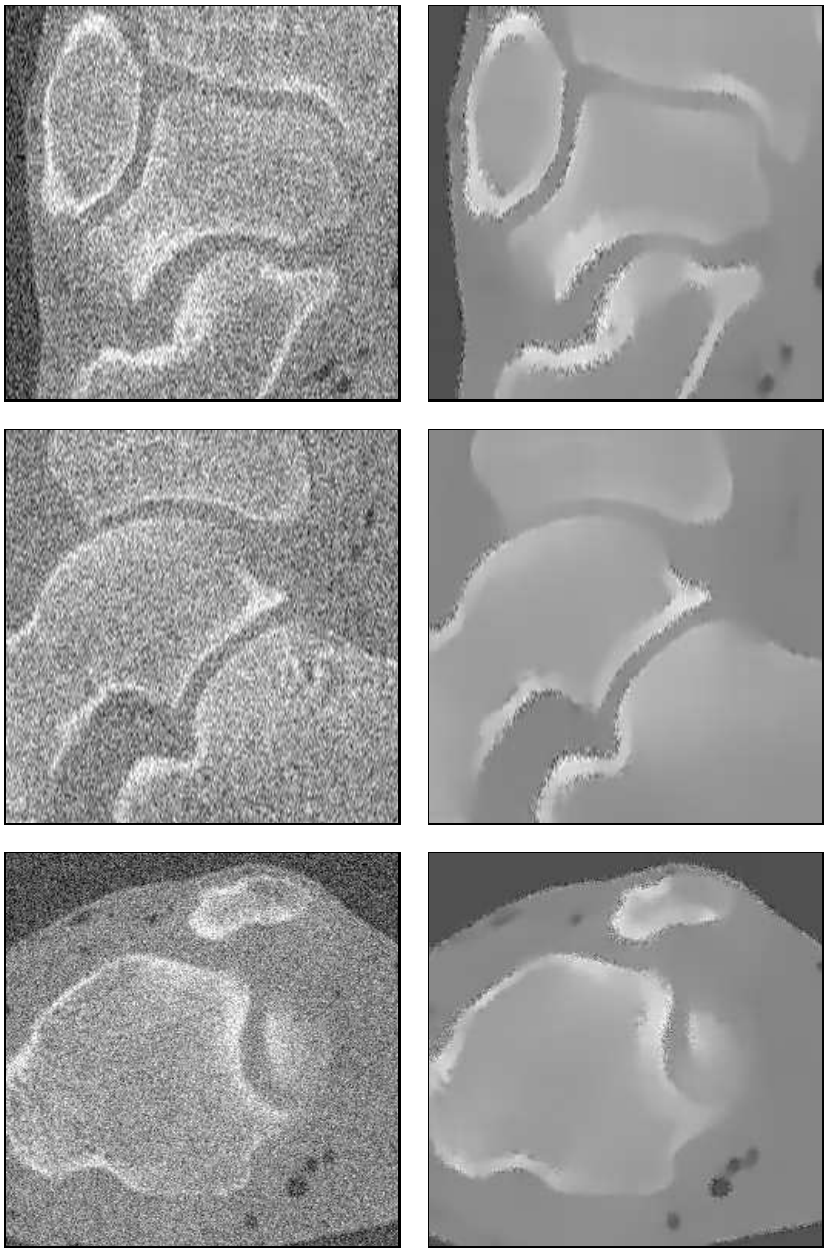

Figure 8: CT scan of a human foot area with grey scale range $[0, 255]$. The data size is $256 \times 256 \times 128$. *From top to bottom:* Slices in XY, XZ and YZ direction. *Left:* Data set degraded by Gaussian noise with standard deviation $\sigma_n = 30$. *Right:* Data set denoised by the implemented algorithm, 5 iterations with $\sigma = 2.5$, $\lambda = 0.01$ and $\tau = 20$.

6 Summary and Conclusions

The goal of this paper was to show how to design algorithms for high performance cluster systems. This was done by the example of nonlinear diffusion. Based on an AOS scheme we first performed a decomposition into modules. Then parallelisation strategies suitable for a high performance low latency network were discussed. We saw that in this case a repartitioning approach is favorable for the majority of operations. Moreover, we noticed that this strategy should be combined with slice partitioning for optimal performance. Test runs with our implementation on a high end cluster system yielded speedup factors of up to 209 for 256 nodes, proving its excellent scalability.

Acknowledgement. Our research has been partly funded by the *Deutsche Forschungsgemeinschaft (DFG)* under the project SCHN 457/4-1. This is gratefully acknowledged. We also thank Wiro Niessen at Utrecht University Hospital for providing the 3-D trabecular bone data set.

References

[1] S. T. Acton. Multigrid anisotropic diffusion. *IEEE Transactions on Image Processing*, 7(3):280–291, Mar. 1998.

[2] E. Bänsch and K. Mikula. A coarsening finite element strategy in image selective smoothing. *Computation and Visualization in Science*, 1:53–61, 1997.

[3] A. Bruhn, T. Jacob, M. Fischer, T. Kohlberger, J. Weickert, U. Brüning, and C. Schnörr. Designing 3-D nonlinear diffusion filters for high performance cluster computing. In L. Van Gool, editor, *Pattern Recognition*, volume 2449 of *Lecture Notes in Computer Science*, pages 290–297. Springer, Berlin, 2002.

[4] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll. Image selective smoothing and edge detection by nonlinear diffusion. *SIAM Journal on Numerical Analysis*, 32:1895–1909, 1992.

[5] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on Image Processing*, 6(2):298–311, 1997.

[6] B. Fischer and J. Modersitzki. Fast diffusion registration. In M. Z. Nashed and O. Scherzer, editors, *Inverse Problems, Image Analysis, and*

Medical Imaging, volume 313 of *Contemporary Mathematics*, pages 117–127. AMS, Providence, 2002.

[7] F. L. Fontaine and S. Basu. Wavelet-based solution to anisotropic diffusion equation for edge detection. *International Journal of Imaging Systems and Technology*, 9:356–368, 1998.

[8] J. Fröhlich and J. Weickert. Image processing using a wavelet algorithm for nonlinear diffusion. Technical Report 104, Laboratory of Technomathematics, University of Kaiserslautern, Germany, Mar. 1994.

[9] T. Gijbels, P. Six, L. Van Gool, F. Catthoor, H. De Man, and A. Oosterlinck. A VLSI-architecture for parallel non-linear diffusion with applications in vision. In *Proc. IEEE Workshop on VLSI Signal Processing*, volume 7, pages 398–407, 1994.

[10] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky. Fast geodesic active contours. *IEEE Transactions on Image Processing*, 10(10):1467–1475, Oct. 2001.

[11] A. Handlovičová, K. Mikula, and F. Sgallari. Variational numerical methods for solving diffusion equation arising in image processing. *Journal of Visual Communication and Image Representation*, 2001. To appear.

[12] J. Heers, C. Schnörr, and H.-S. Stiehl. Investigation of parallel and globally convergent iterative schemes for nonlinear variational image smoothing and segmentation. In *Proc. 1998 IEEE International Conference on Image Processing*, volume 3, pages 279–283, Chicago, IL, Oct. 1998.

[13] J. Heers, C. Schnörr, and H. S. Stiehl. Globally-convergent iterative numerical schemes for non-linear variational image smoothing and segmentation on a multi-processor machine. *IEEE Transactions on Image Processing*, 10(6):852–864, 2001.

[14] C. Herbe. Numerical methods for nonlinear diffusion models of sandpile growth. Master's thesis, Department of Mathematics, University of Kaiserslautern, Germany, Jan. 1999.

[15] B. Jawerth, P. Lin, and E. Sinzinger. Lattice Boltzmann models for anisotropic diffusion of images. *Journal of Mathematical Imaging and Vision*, 11:231–237, 1999.

[16] J. Kačur and K. Mikula. Solution of nonlinear diffusion appearing in image smoothing and edge detection. *Applied Numerical Mathematics*, 17:47–59, 1995.

[17] G. Kühne, J. Weickert, M. Beier, and W. Effelsberg. Fast implicit active contour models. In L. Van Gool, editor, *Pattern Recognition*, volume 2449 of *Lecture Notes in Computer Science*, pages 133–140. Springer, Berlin, 2002.

[18] T. Lu, P. Neittaanmäki, and X.-C. Tai. A parallel splitting up method and its application to Navier–Stokes equations. *Applied Mathematics Letters*, 4(2):25–29, 1991.

[19] R. Malladi and I. Ravve. Fast difference schemes for edge enhancing Beltrami flow. In A. Heyden, G. Sparr, M. Nielsen, and P. Johansen, editors, *Computer Vision – ECCV 2002*, volume 2350 of *Lecture Notes in Computer Science*, pages 343–357. Springer, Berlin, 2002.

[20] N. Paragios, O. Mellina-Gottardo, and V. Ramesh. Gradient vector flow fast geodesic active contours. In *Proc. Eighth International Conference on Computer Vision*, volume 1, pages 67–73, Vancouver, Canada, July 2001. IEEE Computer Society Press.

[21] P. Perona and J. Malik. A network for multiscale image segmentation. In *Proc. IEEE International Symposium on Circuits and Systems*, pages 2565–2568, Espoo, Finland, June 1988.

[22] P. Perona and J. Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:629–639, 1990.

[23] T. Preußer and M. Rumpf. An adaptive finite element method for large scale image processing. In M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, editors, *Scale-Space Theories in Computer Vision*, volume 1682 of *Lecture Notes in Computer Science*, pages 223–234. Springer, Berlin, 1999.

[24] U. S. Ranjan and K. R. Ramakrishnan. A stochastic scale space for multiscale image representation. In M. Nielsen, P. Johansen, O. F. Olsen, and J. Weickert, editors, *Scale-Space Theories in Computer Vision*, volume 1682 of *Lecture Notes in Computer Science*, pages 441–446. Springer, Berlin, 1999.

[25] J. Rexilius. Anisotrope nichtlineare Diffusion für die Bildverarbeitung. Technical Report B-99-07, Institute of Mathematics and Computer Science, Medical University of Lübeck, Germany, Dec. 1999.

[26] M. Rumpf and R. Strzodka. Nonlinear diffusion in graphics hardware. In *Proc. Joint Eurographics – IEEE TCVG Symposium on Visualization*, Ascona, Switzerland, May 2001. Springer.

[27] C. Schnörr. A study of a convex variational diffusion approach for image segmentation and feature extraction. *Journal of Mathematical Imaging and Vision*, 8(3):271–292, 1998.

[28] H. R. Schwarz. *Numerical Analysis: A Comprehensive Introduction.* Wiley, New York, 1989.

[29] R. Temam. Sur l'approximation de la solution des équations de Navier–Stokes par la méthode de pas fractionnaires (I). *Archive for Rational Mechanics and Analysis*, 32:135–153, 1969.

[30] L. H. Thomas. Elliptic problems in linear difference equations over a network. Technical report, Watson Scientific Computing Laboratory, Columbia University, New York, NJ, 1949.

[31] J. Weickert. *Anisotropic Diffusion in Image Processing.* Teubner, Stuttgart, 1998.

[32] J. Weickert. Applications of nonlinear diffusion in image processing and computer vision. *Acta Mathematica Universitatis Comenianae*, 70(1):33–50, 2001.

[33] J. Weickert. Efficient image segmentation using partial differential equations and morphology. *Pattern Recognition*, 34(9):1813–1824, Sept. 2001.

[34] J. Weickert and G. Kühne. Fast methods for implicit active contour models. In S. Osher and N. Paragios, editors, *Geometric Level Set Methods in Imaging, Vision and Graphics*, pages 43–58. Springer, New York, 2003.

[35] J. Weickert and C. Schnörr. Variational optic flow computation with a spatio-temporal smoothness constraint. *Journal of Mathematical Imaging and Vision*, 14(3):245–255, May 2001.

[36] J. Weickert, B. M. ter Haar Romeny, A. Lopez, and W. J. van Enk. Orientation analysis by coherence-enhancing diffusion. In *Proc. 1997*

Real World Computing Symposium, pages 96–103, Tokyo, Japan, Jan. 1997.

[37] J. Weickert, B. M. ter Haar Romeny, and M. A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, 7(3):398–410, Mar. 1998.

[38] J. Weickert, K. J. Zuiderveld, B. M. ter Haar Romeny, and W. J. Niessen. Parallel implementations of AOS schemes: A fast way of nonlinear diffusion filtering. In *Proc. 1997 IEEE International Conference on Image Processing*, volume 3, pages 396–399, Santa Barbara, CA, Oct. 1997.

[39] K. Wiehler, J. Heers, C. Schnörr, H.-S. Stiehl, and R.-R. Grigat. A 1D analog VLSI implementation for non-linear real-time signal preprocessing. *Real-Time Imaging*, 7(1):127–142, Feb. 2001.