Designing 3-D Nonlinear Diffusion Filters for High Performance Cluster Computing

Andrés Bruhn[1], Tobias Jakob[2], Markus Fischer[2], Timo Kohlberger[3], Joachim Weickert[1], Ulrich Brüning[2], and Christoph Schnörr[3]

[1] Mathematical Image Analysis Group,
Faculty of Mathematics and Computer Science,
Building 27.1, Saarland University, 66041 Saarbrücken, Germany,
{bruhn,weickert}@mia.uni-saarland.de
[2] Computer Architecture Group,
Department of Mathematics and Computer Science,
University of Mannheim, 68131 Mannheim, Germany,
{ulrich,mfischer,tjakob}@mufasa.informatik.uni-mannheim.de
[3] Computer Vision, Graphics, and Pattern Recognition Group,
Department of Mathematics and Computer Science,
University of Mannheim, 68131 Mannheim, Germany,
{schnoerr,tkohlber}@uni-mannheim.de

Abstract. This paper deals with parallelization and implementation aspects of PDE based image processing models for large cluster environments with distributed memory. As an example we focus on nonlinear isotropic diffusion filtering which we discretize by means of an additive operator splitting (AOS). We start by decomposing the algorithm into small modules that shall be parallelized separately. For this purpose image partitioning strategies are discussed and their impact on the communication pattern and volume is analyzed. 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.

Keywords: diffusion filtering, additive operator splitting, cluster computing.

1 Introduction

In the last decade PDE based models have become very popular in the fields of image processing and computer vision. The efforts in this paper focus on nonlinear isotropic diffusion models that allow to denoise images while preserving edges. This property makes them useful for restauration and segmentation purposes. Nonlinear diffusion models were first introduced by a work of Perona and Malik [5]. After some years their original model was improved by Catté et al. [1] from both a theoretical and practical viewpoint, and anisotropic extensions

L. Van Gool (Ed.): DAGM 2002, LNCS 2449, pp. 290–297, 2002.

with a diffusion tensor [7] followed. Efficient realizations include, among others, adaptive finite volume schemes [3], linearizations using auxiliary variables [2] or approximations in graphics hardware [6]. The use of fast additive operator splitting schemes [8] has triggered first parallel implementations for diffusion filtering [9]. These approaches have been generally restricted to systems based on *shared* memory. In recent years a rapid progress in this sector changed this 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. This increases 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, parallelization approaches must fit the underlying network topology. This motivated us to show that a 3-D isotropic nonlinear diffusion process can be parallelized in such way, that it owns excellent scaling properties regarding both computation and communication on a *distributed* memory system.

The paper is organized as follows. In Section 2 a review on diffusion filtering and the AOS scheme is given. Furthermore a modular decomposition before parallelization is shown. In Section 3 partitioning and communication models are discussed. Relevant parallelization 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.

2 Nonlinear Isotropic Diffusion Using AOS

In the following we give a very short review of the nonlinear diffusion model of Catté et al. [1]. A grey value image f is considered as a function from a given domain $\Omega_1 \subset \mathbb{R}^m$ into $\Omega_2 \subset \mathbb{R}$. In our case we have $m \in \{2, 3\}$, what corresponds to 2-D and 3-D images. The basic nonlinear diffusion problem then reads:

Find a function $u(x, t)$: $\Omega_1 \times \mathbb{R}_0^+ \to \Omega_2$ that solves the diffusion equation

$$\partial_t u = \operatorname{div}\Big(g(|\nabla u_\sigma|^2)\nabla u\Big) \quad \text{on} \quad \Omega_1 \times \mathbb{R}_0^+ \tag{1}$$

with f as initial value,

$$u(x, 0) = f(x) \quad \text{on} \quad \Omega_1 \tag{2}$$

and reflecting boundary conditions:

$$\partial_n u = 0 \quad \text{on} \quad \partial\Omega_1 \times \mathbb{R}_0^+. \tag{3}$$

where σ is the standard deviation of the Gaussian kernel that is applied prior to differentiation, n is a normal vector perpendicular to $\partial\Omega_1$, 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 over t, where the time t acts as a scale parameter. An example illustrating the performance of this diffusion filter is given in Figure 2.

Nonlinear diffusion filters require numerical approximations. In [8] a finite difference scheme based on an additive operator splitting (AOS) technique [4] is used for this purpose. This AOS technique is the basis for our parallelization 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 discretization of the divergence expression along the l-th coordinate axis. 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. It 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.

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. Presmoothing of the image $u_\sigma^k = K_\sigma * u^k$
2. Computation of the derivatives $|\nabla u_\sigma^k|^2$ and the diffusivities $g(|\nabla u_\sigma^k|^2)$.
3. Resolution of the tridiagonal systems $(I - m\tau A_l(u_\sigma^k))\, u_l^{k+1} = u^k$

   Averaging the results : $u^{k+1} = \frac{1}{m} \sum_{l=1}^{m} u_l^{k+1}$

3 Parallelization Models

The following parallelization 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 neighborhood operations. This raises problems at partition boundaries, since required information is missing. There are two communication models to handle this problem :

Repartitioning. The basic idea of the repartitioning strategy is to find an appropriate partitioning for each operation, such that the problem of missing neighborhood 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.

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 neighborhood 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 neighborhood 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 lesser demands to the network topology than the previously discussed all–to–all communication, also weakly connected cluster system do benefit from a bandwith scaling effect. Even for high latency networks this strategy is favorable 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 parallelization 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 neighborhood.

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 realized 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 neighborhood 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 minimized. Obviously this partition strategy should be used when it comes to the exchange of boundary information.

4 Parallelization Details

Module 1: Gaussian Convolution. The Gaussian convolution is implemented exploiting separability and symmetry as well as optimizing the computational sequence for optimal cache use. The convolution masks are obtained by sampling the continuous function and truncate it at 3 times the standard deviation. Then the masks are renormalized such that its weight sum up to 1. In our approach the repartitioning strategy is used in combination with slice partitioning. Thus, Gaussian convolution in two out of three directions can be performed without communication effort. Only smoothing in the third direction requires a previous repartitioning step. Moreover, this implementation allows large values for the standard deviation σ, since no boundary exchange takes place.

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 diffusivity values demands matching partitions for all derivatives. Both aspects put a boundary exchange strategy in an advantageous position here. 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. After the exchange of neighborhood information, the derivatives are computed sequentially for each direction. This can be done since parallelism is achieved via image

partitioning. Finally the diffusivity values are computed based on

$$g(|\nabla u_\sigma^2|) := \frac{1}{1 + |\nabla u_\sigma^2|/\lambda^2} \tag{5}$$

where λ is a contrast parameter.

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 for parallelization 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. 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 parallelization 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. Therefore, not only the original image has to be repartitioned, but also the corresponding diffusivity data prior to computing the third diffusion direction. Finally combining the results of all three diffusion processes – the averaging step in the AOS scheme – requires a third repartitioning.

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 40th in the last TOP 500 list of supercomputers.

As one can see from Figure 1 considerations regarding the parallelization 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

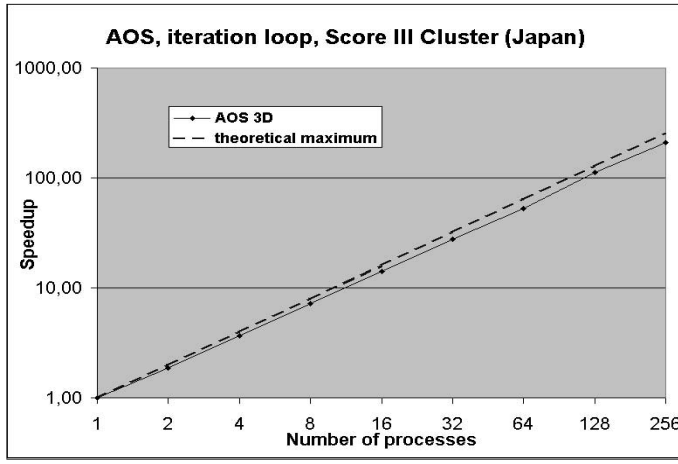

Fig. 1. Speedup Chart

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

Processors	1	2	4	8	16	32	64	128	256
Runtime [s]	212.741	114.625	57.534	29.401	15.065	7.731	4.029	1.894	1.017
Computation [s]	212.741	106.205	52.221	26.123	13.471	6.753	3.333	1.550	0.745
Communication [s]	0.000	8.420	5.313	3.278	1.594	0.978	0.696	0.344	0.272
Computation [%]	100.000	92.654	90.766	88.851	89.420	87.350	82.725	81.837	73.255
Communication [%]	0.000	7.346	9.234	11.149	10.580	12.650	17.275	18.163	26.745

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 below, 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.

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 isotropic diffusion. Based on an AOS scheme we first performed a decomposition into modules. Then parallelization 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.

Fig. 2. *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 nonlinear isotropic diffusion filter, 5 iterations with $\sigma = 2.5$, $\lambda = 0.01$ and $\tau = 20$.

Acknowledgement

Our research has been partly funded by the *Deutsche Forschungsgemeinschaft (DFG)* under the project SCHN 457/4-1.

References

1. 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.
2. 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.
3. Z. Krivá and K. Mikula. An adaptive finite volume scheme for solving nonlinear diffusion equations in image processing. *Journal of Visual Communication and Image Representation*, 13(1/2):22–35, 2002.
4. 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.
5. 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.
6. M. Rumpf and R. Strzodka. Nonlinear diffusion in graphics hardware. In *Proc. Joint Eurographics – IEEE TCVG Symposium on Visualization*, Ascona, Switzerland, May 2001.
7. J. Weickert. *Anisotropic Diffusion in Image Processing.* Teubner, Stuttgart, 1998.
8. 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.
9. 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.