Parallel Variational Motion Estimation by Domain Decomposition and Cluster Computing

Timo Kohlberger[1], Christoph Schnörr[1], Andrés Bruhn[2], and Joachim Weickert[2]

[1] University of Mannheim, Dept. of Mathematics and Computer Science,
D-68131 Mannheim, Germany
{kohlberger,schnoerr}@uni-mannheim.de
http://www.cvgpr.uni-mannheim.de
[2] Saarland University, Dept. of Mathematics and Computer Science,
D-66041 Saarbrücken, Germany
{bruhn,weickert}@mia.uni-saarland.de
http://www.mia.uni-saarland.de

Abstract. We present an approach to parallel variational optical flow computation on standard hardware by domain decomposition. Using an arbitrary partition of the image plane into rectangular subdomains, the *global* solution to the variational approach is obtained by iteratively combining *local* solutions which can be efficiently computed in parallel by separate multi-grid iterations for each subdomain. The approach is particularly suited for implementations on PC-clusters because inter-process communication between subdomains (i.e. processors) is minimized by restricting the exchange of data to a *lower*-dimensional interface. By applying a dedicated interface preconditioner, the necessary number of iterations between subdomains to achieve a fixed error is bounded independently of the number of subdomains. Our approach provides a major step towards real-time 2D image processing using off-the-shelf PC-hardware and facilitates the efficient application of variational approaches to large-scale image processing problems.

1 Introduction

1.1 Overview and Motivation

Motion estimation in terms of optical flow [3,25] is an important prerequisite for many applications of computer vision including surveillance, robot navigation, and dynamic event recognition. Since real-time computation is required in many cases, much work has been done on parallel implementations of *local* motion estimation schemes (differential-, correlation-, or phase-based methods) [31,14]).

In contrast to local estimation schemes, less work has been done on the parallelization of *non-local variational* schemes for motion estimation, despite considerable progress during the last years related to robustness, non-linear regularization schemes, preservation of motion boundaries, and corresponding successful applications [27,26,13,1,1,21, 5,28,20,18,4,15,23,32,11,24]. It is precisely the *non-local* nature of these approaches which on the one hand allows to impose structural constraints on motion fields during estimation but, on the other hand, hampers a straightforward parallel implementation by simply partitioning the image domain into disjoint subdomains (see Figure 1).

T. Pajdla and J. Matas (Eds.): ECCV 2004, LNCS 3024, pp. 205–216, 2004.

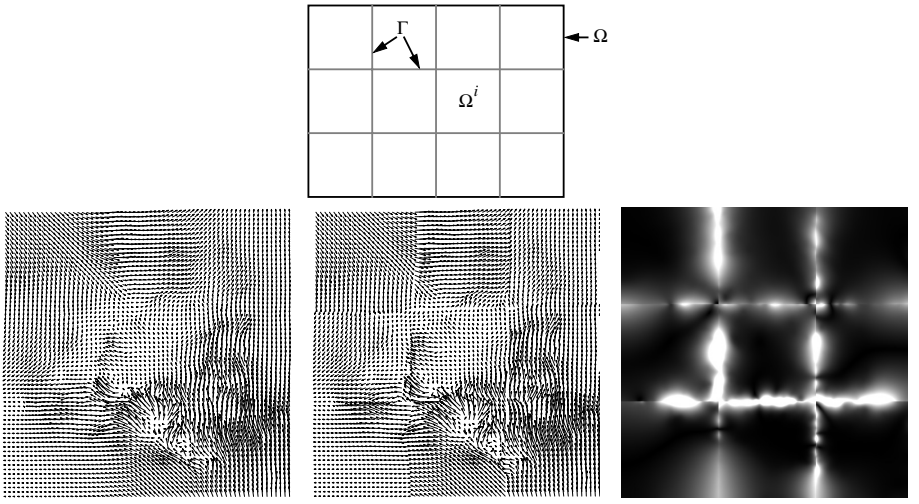


Fig. 1. Top: A partition of the image domain Ω into subdomains $\{\Omega^i\}$ and a lower-dimensional interface Γ. **Bottom, left:** Motion field estimated with a non-local variational approach. **Bottom, center:** Naive parallelization by estimating motion independently in each subdomain leads to boundary effects (a coarse 3×3 partition was used for better visibility). **Bottom, right:** The l_2-error caused by naive parallelization as grayvalue plot. The *global* relative error is 11.3%. The *local* error near boundaries of subdomains is much higher!

This motivates to investigate computational approaches for the parallelization of variational motion estimation by means of domain decompositions as shown in Figure 1, top. Ideally, any of the approaches cited above should be applicable independently in each subdomain. In addition to that, however, a mechanism is needed which fits together the subdomain solutions so as to yield the same global solution which is obtained when applying the respective approach in the usual non-parallel way to the entire image domain Ω. The investigation of such a scheme is the objective of this paper.

1.2 Contribution and Organization

We present an approach to the parallelization of *variational* optical flow computation which fulfils the following requirements:

(i) Suitability for the implementation on PC-clusters through the minimization of inter-process communication,
(ii) use of a mathematical framework which allows for applications to a large class of variational models.

In order to meet requirement (ii), our approach draws upon the general mathematical theory on domain decomposition in connection with the solution of partial differential equations [9,30,29]. Requirement (i) addresses a major source for degrading the performance of message-passing based parallel architectures. To this end, we focus on

the subclass of *substructuring methods* because inter-process communication is minimized by restricting the exchange of data to a *lower*-dimensional interface between the subdomains.

After sketching a prototypical variational approach to motion estimation in section 2, we develope a corresponding domain decomposition framework in section 3. In section 4, we describe features of our parallel implementation, the crucial design of an interface-preconditioner, and report the influence of both preconditioning and the number of subdomains on the convergence rate. Finally, we report measurements for experiments on a PC-cluster with nodes in section 5.

2 Variational Motion Estimation

In this section, we sketch the prototypical variational approach of Horn and Schunck [19] and its discretization as a basis for domain decomposition. Note that our formulation suffiently abstracts for this particular approach. As a consequence, the framework developed in section 3 can be applied to more general variational approaches to motion estimation, as discussed in section 1.1.

2.1 Variational Problem

Throughout this section, $g(x) = g(x_1, x_2)$ is the grayvalue function, $\nabla = (\partial_{x_1}, \partial_{x_2})^\top$ denotes the gradient with respect to spatial variables, ∂_t the partial derivative with respect to time, and $u = (u_1, u_2)^\top, v = (v_1, v_2)^\top$ denote vector fields in the linear space $V = H^1(\Omega) \times H^1$.

With this notational convention, the variational problem to be solved reads [19]:

$$J(u) = \inf_{v \in V} \int_\Omega \left\{ (\nabla g \cdot v + \partial_t g)^2 + \lambda \left(|\nabla v_1|^2 + |\nabla v_2|^2 \right) \right\} dx \tag{1}$$

Vanishing of the first variation of the functional J in (1) yields the variational equation:

$$a(u, v) = f(v) \ , \quad \forall v \in V \ , \tag{2}$$

where

$$a(u, v) = \int_\Omega \left\{ (\nabla g \cdot u)(\nabla g \cdot v) + \lambda (\nabla u_1 \cdot \nabla v_1 + \nabla u_2 \cdot \nabla v_2) \right\} dx \tag{3}$$

$$f(v) = -\int_\Omega \partial_t g \nabla g \cdot v dx \tag{4}$$

Under weak conditions with respect to the image data g (i.e. $\partial_{x_1} g$ and $\partial_{x_2} g$ have to be independent in the L^2-sense, there exists a constant $c > 0$ such that:

$$a(v, v) \geq c \|v\|_V^2 \ , \quad \forall v \in V \tag{5}$$

As a consequence, J in (1) is strictly convex and its global minimum u is the unique solution to the variational equation (2). Partially integrating in (2), we derive the *system* of Euler-Lagrange equations:

$$Lu = f \quad \text{in} \quad \Omega\,, \qquad \partial_n u = 0 \quad \text{on} \quad \partial\Omega\,, \tag{6}$$

where

$$Lu = -\lambda \Delta u + (\nabla g \cdot u)\nabla g$$

2.2 Discretization

To approximate the vector field u numerically, equation (2) is discretized by piecewise linear finite elements over the triangulated section Ω of the image plane [10]. We arrange the vectors of nodal variables u_1, u_2 corresponding to the finite element discretizations of $u_1(x), u_2(x)$ as follows[1]: $u = (u_1^\top, u_2^\top)^\top$. Taking into consideration the symmetry of the bilinear form (3), this induces the following block structure of the discretized version $Au = f$ of (2):

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix}\,, \tag{7}$$

where $\forall i, j = 1, \ldots, N$:

$$\begin{aligned} (A_{11})_{ij} &= a\big((\phi_i, 0)^\top, (\phi_j, 0)^\top\big) \\ (A_{12})_{ij} &= a\big((\phi_i, 0)^\top, (0, \phi_j)^\top\big) \\ (A_{21})_{ij} &= (A_{12})_{ji} \\ (A_{22})_{ij} &= a\big((0, \phi_i)^\top, (0, \phi_j)^\top\big) \\ (f_1)_i &= f\big((\phi_i, 0)^\top\big) \\ (f_2)_i &= f\big((0, \phi_i)^\top\big) \end{aligned}$$

Here, ϕ_k denotes the linear basis function corresponding to the nodal variable $(u_1)_k$ or $(u_2)_k$, respectively.

3 Domain Decomposition

3.1 Two Subdomains

Let $\Omega^1 \cup \Omega^2$ be a partition of Ω with a common boundary $\Gamma = \overline{\Omega^1} \cap \overline{\Omega^2}$. We denote the corresponding function spaces with V^1, V^2. In the following, superscripts refer to subdomains.

We wish to represent u from (6) by two functions $u^1 \in V^1, u^2 \in V^2$ which are computed by solving two related problems in Ω^1, Ω^2, respectively. The relation:

$$u(x) = \begin{cases} u^1(x) & x \in \Omega^1 \\ u^2(x) & x \in \Omega^2 \end{cases} \tag{8}$$

[1] With slight abuse of notation, we use the same symbols u_1, u_2, for simplicity.

obviously holds iff the following is true:

$$Lu^1 = f^1 \quad \text{in } \Omega^1 \qquad \partial_{n^1} u^1 = 0 \quad \text{on } \partial\Omega^1 \cap \partial\Omega \tag{9}$$

$$Lu^2 = f^2 \quad \text{in } \Omega^2 \qquad \partial_{n^2} u^2 = 0 \quad \text{on } \partial\Omega^2 \cap \partial\Omega \tag{10}$$

$$u^1 = u^2 \quad \text{on } \Gamma \tag{11}$$

$$\partial_{n^1} u^1 = -\partial_{n^2} u^2 \quad \text{on } \Gamma \tag{12}$$

We observe that (6) cannot simply be solved by separately computing u^1 and u^2 in each domain Ω_1, Ω_2 (see also Fig. 1!) because the natural boundary conditions have to be changed on Γ, due to (11) and (12).

As a consequence, in order to solve the system of equations (9)-(12), we equate the restriction to the interface Γ of the two solutions u^1, u^2 to (9) and (10), due to equation (11), $u_\Gamma := u^1|_\Gamma = u^2|_\Gamma$, and substitute u_Γ into equation (12) (see Eqn. (19) below).

To this end, we solve

$$Lu = f \quad \text{in } \Omega\ , \quad \partial_n u = 0 \quad \text{on } \partial\Omega \setminus \Gamma\ , \quad u = u_\Gamma \quad \text{on } \Gamma \tag{13}$$

and decompose u into two functions,

$$u = u_0 + u_f\ ,$$

which are the unique solutions to the following problems:

$$Lu_0 = 0 \quad \text{in } \Omega\ , \quad \partial_n u_0 = 0 \quad \text{on } \partial\Omega \setminus \Gamma\ , \quad u_0 = u_\Gamma \quad \text{on } \Gamma \tag{14}$$

$$Lu_f = f \quad \text{in } \Omega\ , \quad \partial_n u_f = 0 \quad \text{on } \partial\Omega \setminus \Gamma\ , \quad u_f = 0 \quad \text{on } \Gamma \tag{15}$$

Clearly, the restriction of u_f to the interface Γ is zero, $u_f|_\Gamma = 0$, and:

$$u|_\Gamma = u_0|_\Gamma \tag{16}$$

$$\partial_n u = \partial_n u_0 + \partial_n u_f \tag{17}$$

The definition of the *Steklov-Poincaré operator* S is:

$$S : u_\Gamma \to \partial_n u_0|_\Gamma \tag{18}$$

Applying this mapping to the solutions u^1, u^2 of equations (9) and (10) in the domains Ω^1 and Ω^2, respectively, equation (12) becomes with $u_\Gamma = u^1|_\Gamma = u^2|_\Gamma$ due to (11):

$$(S^1 + S^2)u_\Gamma + \partial_{n^1} u_f^1|_\Gamma + \partial_{n^2} u_f^2|_\Gamma = 0 \tag{19}$$

It remains to discretize this equation in order to solve for u_Γ. This will be done in the following section.

3.2 Discretizing the Interface Equation

By virtue of definitions (14) and (15) and standard results [2], we obtain in each domain respective the linear systems:

$$\begin{pmatrix} A_{II}^i & A_{I\Gamma}^i \\ A_{\Gamma I}^i & A_{\Gamma\Gamma}^i \end{pmatrix} \begin{pmatrix} (u_0^i)_I \\ u_\Gamma^i \end{pmatrix} = \begin{pmatrix} 0 \\ \partial_n u_0^i|_\Gamma \end{pmatrix}\ , \quad i = 1, 2 \tag{20}$$

and

$$\begin{pmatrix} A_{II}^i & A_{I\Gamma}^i \\ A_{\Gamma I}^i & A_{\Gamma\Gamma}^i \end{pmatrix} \begin{pmatrix} u_f^i \\ 0 \end{pmatrix} = \begin{pmatrix} f_I^i \\ f_\Gamma^i + \partial_{n^i} u_f^i|_\Gamma \end{pmatrix} , \quad i = 1, 2 \tag{21}$$

Due to the system (9)-(12), we have to solve simultaneously (20) and (21) in both domains. Since $u^i = u_0^i + u_f^i$, summation of (20) and (21) for each domain, respectively, gives:

$$\begin{pmatrix} A_{II}^i & A_{I\Gamma}^i \\ A_{\Gamma I}^i & A_{\Gamma\Gamma}^i \end{pmatrix} \begin{pmatrix} u_I^i \\ u_\Gamma^i \end{pmatrix} = \begin{pmatrix} f_I^i \\ f_\Gamma^i + \partial_{n^i} u^i|_\Gamma \end{pmatrix} , \quad i = 1, 2$$

We combine these equations into a single system:

$$\begin{pmatrix} A_{II}^1 & 0 & A_{I\Gamma}^1 \\ 0 & A_{II}^2 & A_{I\Gamma}^2 \\ A_{\Gamma I}^1 & A_{\Gamma I}^2 & A_{\Gamma\Gamma}^1 + A_{\Gamma\Gamma}^2 \end{pmatrix} \begin{pmatrix} u_I^1 \\ u_I^2 \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I^1 \\ f_I^2 \\ f_\Gamma + \partial_{n^1} u^1|_\Gamma + \partial_{n^2} u^2|_\Gamma \end{pmatrix} , \tag{22}$$

where

$$f_\Gamma = f_\Gamma^1 + f_\Gamma^2 .$$

By solving the first two equations for u_I^1, u_I^2 and substitution into the third equation of (22), we conclude that (12) holds iff:

$$A_{\Gamma I}^1 (A_{II}^1)^{-1} (f_I^1 - A_{I\Gamma}^1 u_\Gamma) + A_{\Gamma I}^2 (A_{II}^2)^{-1} (f_I^2 - A_{I\Gamma}^2 u_\Gamma) + (A_{\Gamma\Gamma}^1 + A_{\Gamma\Gamma}^2) u_\Gamma = f_\Gamma \tag{23}$$

This is just the discretized counterpart of (19):

$$(S^1 + S^2) u_\Gamma = f_\Gamma - A_{\Gamma I}^1 (A_{II}^1)^{-1} f_I^1 - A_{\Gamma I}^2 (A_{II}^2)^{-1} f_I^2 \tag{24}$$

$$S^i u_\Gamma = (A_{\Gamma\Gamma}^i - A_{\Gamma I}^i (A_{II}^i)^{-1} A_{I\Gamma}^i) u_\Gamma , \quad i = 1, 2 \tag{25}$$

Once u_Γ is computed by solving (24), u^1 and u^2 follow from (9), (10) with boundary values u_Γ on the common interface Γ.

3.3 Multiple Domains

Let R^i denote the restriction of the vector of nodal variables u_Γ on the interface Γ to those on $\overline{\Omega^i} \cap \Gamma$. Analogously to the case of two domains detailed above, the interface equation (19) in the case of multiple domains reads:

$$\left(\sum_i (R^i)^\top S^i R^i \right) u_\Gamma = f_\Gamma - \sum_i (R^i)^\top A_{\Gamma I}^i (A_{II}^i)^{-1} f_I^i , \quad \forall i \tag{26}$$

Note that all computations restricted to subdomains Ω^i can be done in parallel!

4 Preconditioning, Parallel Implementation, and Convergence Rates

4.1 Interface Preconditioner

While a fine partition of Ω into a large number of subdomains Ω^i leads to small-sized and "computationally cheap" local problems in each subdomain, the condition number of the Steklov-Poincaré operator S more and more deteriorates [29]. As a consequence, preconditioning of the interface equation becomes crucial for an efficient parallel implementation.

Among different but provably optimal ("spectrally equivalent") families of preconditioners (cf. [9,30]), we examined in particular the *Balancing-Neumann-Neumann-preconditioner (BNN)* [22,12]:

$$P_{BNN}^{-1} := (I - (R^0)^\top (S^0)^{-1} R^0 S) P_{NN}^{-1} (I - S(R^0)^\top (S^0)^{-1} R^0) + (R^0)^\top (S^0)^{-1} R^0 \,, \tag{27}$$

where

$$P_{NN}^{-1} := D \left(\sum_i (R^i)^\top \left(S^i\right)^{-1} R^i \right) D \tag{28}$$

This preconditioner applied in connection with conjugate gradient iteration [17] preserve the symmetry of S in (18) and naturally extends to more general problems related to three-dimensional image sequences or unstructured geometries and/or triangulations.

Preconditioner P_{BNN}^{-1} carries out a correction step (denoted as "balancing" in literature) before and after the application of the *Neumann-Neumann-preconditioner (NN)* (28) on a coarse grid given by the partition of the domain Ω into subdomains (see Figure 1).

The restriction operator R^0 sums up the weighted values on the boundary of each subdomain, where the weights are given by the inverse of the number of subdomains sharing each particular node, i.e.

$$(R^0)_{ji} := \begin{cases} \frac{1}{2} : \text{node } i \text{ is on an edge of } \Omega_j \\ \frac{1}{4} : \text{node } i \text{ is in a vertex of } \Omega_j \\ 0 \ : \text{else} \end{cases} \tag{29}$$

Then, S^0 is defined by

$$S^0 := R^0 S (R^0)^\top . \tag{30}$$

Note that S^0 is a dense matrix of small dimension (related to the number of subdomains) which can be efficiently inverted by a standard direct method.

4.2 Parallel Implementation

The preconditioned conjugate gradient iteration for solving the interface equations (19) and (26) provides several starting points for parallel computation. In the case of the NN-preconditioner (28) the calculation of $(S^i)^{-1}$, which is done indirectly by calculating

$(A^i)^{-1}$ on the whole subdomain, can be carried out in parallel. Then, the restriction matrices R^i and $(R^i)^\top$ amount to *scatter-operations* and *gather-operations* from the point of view of the central process. Furthermore, when calculating S during a conjugate gradient iteration, parallelization can be employed also. Since S is already written in decomposed form in (24) and (26) the procedure is done in an analogous manner as with P_{NN}^{-1} with the main difference (beneath leaving out the weightening by D) that here the Dirichlet system (25) has to be solved on each subdomain in order to calculate the action of S^i. Both parallelization procedures are combined in the calculation of the BNN-preconditioner (27) since both operators are involved here.

The inversion of the coarse operator S_0 does not provide any possibilities for parallelization since it has been shown to be most practical to calculate S_0 numerically in advance, by carrying out (30), and then computing S_0^{-1} in the central process by using again a conjugated gradient method. Since the coarse system is much smaller (the grid is equivalent to the subdomain partition) the computation time for this inversion has shown to be very small and can be neglected in practice. Furthermore, the initial computation of the right hand side in (19) or (26) can be parallelized in an analogous manner.

4.3 Multi-Grid Subdomain Solver

Evaluation of the right hand side in (26) as well as S^i and $(S^i)^{-1}$ (cf. (25)) needs in parallel for each domain (i.e. processor) the fast solution of the corresponding Dirichlet and Neumann boundary value problems, respectively. To this end, we implemented a multi-grid solver. Since domain decomposition methods depend strongly on the performance and accuracy of their internal solver, we considered the use of multi-grid methods [6, 16]. These methods are well-known to be among the fastest and most accurate numerical schemes for the solution of systems of linear equations.

In [7,8] such a multi-grid scheme has been proposed for the CLG approach. It allowed the computation of up to 40 flow fields for sequences of size 200×200 within a single second. Obviously, an integration of this strategy into our domain decomposition framework seems desirable. In fact, the only difference between the single and the multiple domain case is the possible cooccurrence of Neumann and Dirichlet boundary conditions in the same subdomain. Once this is taken into account, the implementation is straightforward.

Let us now sketch some technical details of our multi-grid solver. Our strategy is based on the pointwise coupled Gauß-Seidel method, which is hierarchically applied in form of a so called *full multi-grid* strategy. An example of such a full multi-grid scheme is given in Fig.2. It shows how a coarse version of the original problem is refined step by step and how correcting multi-grid methods, e.g. W-cycles, are used at each refinement level for solving. In this context, we chose W-cycles that perform two pointwise coupled Gauß-Seidel iterations in both its pre- and postsmoothing relaxation step. Besides the traversing strategy, operators have to be defined that handle the information transfer between the grids. For *restriction* (fine-to-coarse) simple averaging is used, while *prolongation* (coarse-to-fine) is performed by means of bilinear interpolation. Finally, coarser versions of the matrix operator have to be created. To this end we rediscretised the Euler-Lagrange equations. Such a proceeding is called *discretisation coarse grid approximation (DCA)*.

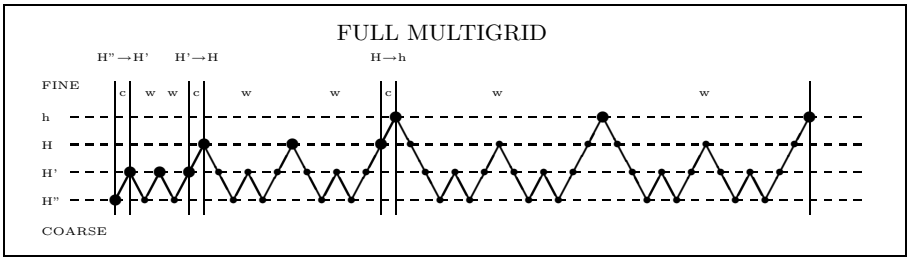


Fig. 2. Example of a full multi-grid implementation for four levels taken from [8]. Vertical solid lines separate alternating blocks of the two basic strategies : *Cascading* and *correcting* multi-grid. Blocks belonging to the cascading multi-grid strategy are marked with *c*. Starting from a coarse scale the original problem is refined step by step. This is visualised by the $\rightarrow$ symbol. Thereby the coarser solution serves as an initial approximation for the refined problem. At each refinement level, a correcting multi-grid solver is used in form of two W-cycles (marked with *w*). Performing iterations on the original equation is marked with large black dots, while iterations on residual equations are marked with smaller ones.

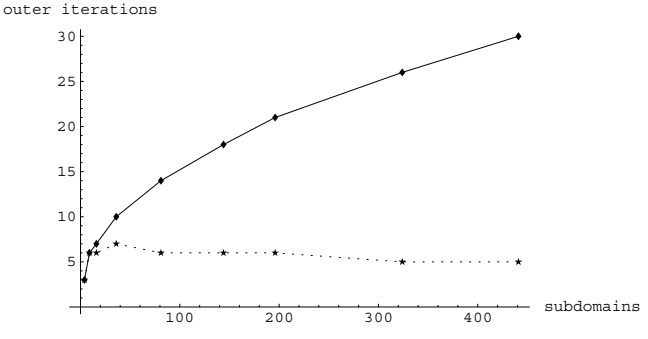
4.4 Convergence Rates

In this section, we examine the influence of both the number of subdomains and the coarse-grid correction step on the convergence rate of the preconditioned conjugate gradient iteration.

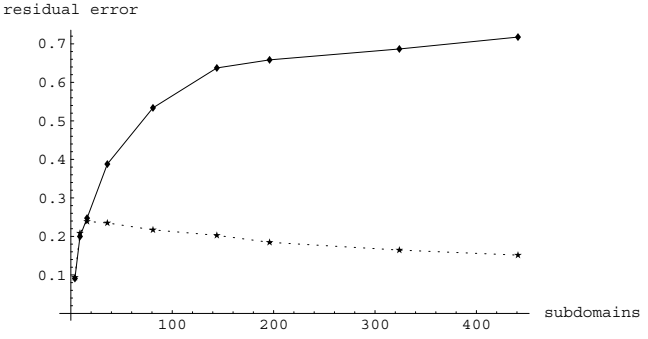
We first measured the number of outer iterations to reach a relative residual error $||u_\Gamma^k - \hat{u}_\Gamma||_S / ||\hat{u}_\Gamma||_S < 10^{-3}$ ($\hat{u}_\Gamma$: exact solution; k: iteration index) of equation (26) for different number of subdomains. The results are depicted in Figure 3 (1). They clearly show that the computational costs using the non-balancing preconditioner grow with the number of subdomains whereas they remain nearly constant for the preconditioner involving a coarse grid correction step (we neglected the time needed for solving the coarse small-sized system related to S^0). These results are confirmed by Figure 3 (2) where the residual error for a *fixed* number of outer PCG-iterations is shown. Thus, the BNN-preconditioner is much closer to an *optimal* preconditioner making the convergence rate independent w.r.t. both the pixel meshsize h and the coarse meshsize H by the number of subdomains. It also follows that the solver using this preconditioner scales much better with the number of subdomains since the reduced costs for the local problems associated with S^i by far compensate the additional costs for solving the coarse-grid system and process communication.

5 Cluster Computing

The algorithm, as described in sections 4.2 and 4.3, was implemented in C/C++ on a Linux operating system. An implementation of the *Message Passing Interface (MPI)* was used for parallelization. Benchmarking was conducted by the *MPE library* included in the MPI package. All experiments have been carried out on a dedicated PC-cluster "HELICS" at the University of Heidelberg, which comprises 256 Dual-Athlon MP



(1)



(2)

Fig. 3. Effect of coarse-grid correction. (1) Number of outer PCG-iterations until the residual error of system (26) is reduced to $||u_k - \hat{u}||_S / ||u_0 - \hat{u}||_S < 10^{-3}$ for coarse mesh sizes $H \in \{126, 84, 63, 42, 28, 21, 18, 14, 12\}$ on a 252×252 image using the NN-preconditioner (solid line) and the BNN-preconditioner (stippled line). The corresponding numbers of subdomains are $\{2^2, 3^2, 4^2, 6^2, 9^2, 12^2, 14^2, 18^2, 21^2\}$. The *local systems* were solved in each subdomain to a residual error of 10^{-5}. (2) Relative residual error after 10 outer PCG-iterations using the NN-preconditioner (solid line) and the BNN-preconditioner (stippled line) for the same set of coarse mesh sizes. The results clearly show the favourable influence of the coarse grid coupling leading to a nearly constant error and therefore to a nearly constant number of outer PCG-iterations when using the balancing preconditioner on 4×4 subdomains and above. Hence, the balancing preconditioner is much closer to an *optimal* preconditioner making the convergence rate nearly independent of h and H.

1.4 GHz nodes (i.e. 512 processors in total) connected by the interconnect network Myrinet2000.

As input data an image pair from a real air-flow sequence provided by the Onera Labs, Rennes, France, has been taken. The reference solution x_{ref} was calculated without parallelization by the use of the multi-grid solver (full multi-grid, 5 W-cycles, 3 pre- and post-relaxation-steps per level) and regularization parameter $\lambda = 0.05$ (the intensity values of the input images where normalized to $[0, 1]$). The objective was to compare the total computation time of the non-parallel solving (by multi-grid) on the whole image

Table 1. Computation times for different partitions. Compared to a dedicated one-processor multi-grid implementation, domain-decomposition accelerates the computation for 5×5 processors and above. The speed-up for 7×7 compared to the computation time on one processor is nearly 40 %.

Partition (h./v.) 512^2 pixels	Image size	Outer iter.	Run time	Comm. time
2×2	511^2	1	1550 ms	4 %
3×3	511^2	1	960 ms	5.6%
5×5	513^2	3	664 ms	10 %
6×6	511^2	4	593 ms	10 %
7×7	512^2	4	516 ms	11 %

plane on one machine to the total computation time of the parallel solving on $N \times N$ processors by using Neumann-Neumann preconditioning. Computation was stopped if an relative error of 1% had been reached, i.e. $||x^i - x_{ref}||_2/||x_{ref}||_2 < 0.01$, x^i : solution after i conjugate gradient-iterations.

Comparison to Single-Processor Multi-grid. The time for calculating the vector field without parallelization on one processor to the given accuracy was 721 ms (full-multi-grid, 2 W-cycles and 1 pre- and post-relaxations per level). In Table 1 the computation times of the parallel substructuring method for different partitions are depicted. The parameters of the local multi-grid solver where optimized by hand to minimize the total computation time. The results show that parallelization by the use of Neumann-Neumann preconditioning starts to improve the computation time from 5×5 processors and above. The speed-up for 7×7 compared to the computation time on one processor is nearly 40%.

Similar experiments for Balancing Neumann-Neumann preconditioning will be conducted in future.

References

1. P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Int. J. of Comp. Vision*, 2:283–310, 1989.
2. J.P. Aubin. *Approximation of Elliptic Boundary-Value Problems*. Wiley&Sons, New York, 1972.
3. J.L. Beauchemin, S.S. and Barron. The computation of optical flow. *ACM Computing Surveys*, 27:433–467, Sept. 1995.
4. M.J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise–smooth flow fields. *Comp. Vis. Image Underst.: CVIU*, 63(1):75–104, 1996.
5. P. Bouthemy and E. Francois. Motion segmentation and qualitative dynamic scene analysis from an image sequence. *Int. J. of Comp. Vision*, 10(2):157–182, 1993.
6. A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, April 1977.
7. A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. In N. Petkov and M. A. Westberg, editors, *Computer Analysis of Images and Patterns*, volume 2756 of *Lecture Notes in Computer Science*, pages 222–229. Springer, Berlin, 2003.

8. A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optic flow computation in real-time. Technical Report 89/2003, Dpt. of Mathematics, Saarland University, Germany, June 2003.
9. T.F. Chan and T.P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.
10. P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland Publ. Comp., Amsterdam, 1978.
11. T. Corpetti, E. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Trans. Patt. Anal. Mach. Intell.*, 24(3):365–380, 2002.
12. M. Dryja and O.B. Widlund. Schwarz methods of neumann-neumann type for three-dimensional elliptic finite element problems. *Comm. Pure Appl. Math.*, 48:121–155, 1995.
13. W. Enkelmann. Investigation of multigrid algorithms for the estimation of optical flow fields in image sequences. *Comp. Vis. Graph. Imag. Proc.*, 43:150–177, 1987.
14. M. Fleury, A.F. Clark, and A.C. Downton. Evaluating optical-flow algorithms on a parallel machine. *Image and Vision Comp.*, 19(3):131–143, 2001.
15. S. Ghosal and P. Vaněk. A fast scalable algorithm for discontinuous optical flow estimation. *IEEE Trans. Patt. Anal. Mach. Intell.*, 18(2):181–194, 1996.
16. W. Hackbusch. *Multigrid Methods and Applications*. Springer, New York, 1985.
17. W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Springer-Verlag, 1993.
18. F. Heitz, P. Perez, and P. Bouthemy. Multiscale minimization of global energy functions in some visual recovery problems. *Comp. Vis. Image Underst.: CVIU*, 59(1):125–134, 1994.
19. B.K.P. Horn and B.G. Schunck. Determining optical flow. *Artif. Intell.*, 17:185–203, 1981.
20. S.H. Hwang and S.U. Lee. A hierarchical optical flow estimation algorithm based on the interlevel motion smoothness constraint. *Patt. Recog.*, 26(6):939–952, 1993.
21. J. Konrad and E. Dubois. Bayesian estimation of motion vector fields. *IEEE Trans. Patt. Anal. Mach. Intell.*, 14(9):910–927, 1992.
22. J. Mandel. Balancing domain decomposition. *Comm. Numer. Meth. Eng.*, 9:233–241, 1993.
23. E. Mémin and P. Pérez. Optical flow estimation and object–based segmentation with robust techniques. *IEEE Trans. on Image Proc.*, 7(5):703–719, 1998.
24. E. Mémin and P. Pérez. Hierarchical estimation and segmentation of dense motion fields. *Int. J. of Comp. Vision*, 46(2):129–155, 2002.
25. A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *Int. J. of Comp. Vision*, 19(1):29–55, 1996.
26. H.H. Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artif. Intell.*, 33:299–324, 1987.
27. H.H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. *IEEE Trans. Patt. Anal. Mach. Intell.*, 8(5):565–593, 1986.
28. P. Nesi. Variational approach to optical flow estimation managing discontinuities. *Image and Vis. Comp.*, 11(7):419–439, 1993.
29. A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Univ. Press, 1999.
30. B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition: Parallel Multilevel Methods for the Solution of Elliptic Partial Differential Equations*. Cambridge Univ. Press, 1996.
31. F. Valentinotti, G. Dicaro, and B. Crespi. Real-time parallel computation of disparity and optical flow using phase difference. *Machine Vision and Appl.*, 9(3):87–96, 1996.
32. J. Weickert and C. Schnörr. A theoretical framework for convex regularizers in pde–based computation of image motion. *Int. J. Computer Vision*, 45(3):245–264, 2001.