Domain Decomposition for Variational Optical-Flow Computation

Timo Kohlberger, Christoph Schnörr, Andrés Bruhn, and Joachim Weickert

***Abstract*—We present an approach to parallel variational optical-flow computation by using an arbitrary partition of the image plane and iteratively solving related local variational problems associated with each subdomain. The approach is particularly suited for implementations on PC clusters because interprocess communication is minimized by restricting the exchange of data to a lower dimensional interface. Our mathematical formulation supports various generalizations to linear/nonlinear convex variational approaches, three-dimensional image sequences, spatiotemporal regularization, and unstructured geometries and triangulations. Results concerning the effects of interface preconditioning, as well as runtime and communication volume measurements on a PC cluster, are presented. Our approach provides a major step toward real-time two-dimensional image processing using off-the-shelf PC hardware and facilitates the efficient application of variational approaches to large-scale image processing problems.**

***Index Terms*—Domain decomposition, image processing, optical flow, parallel computation, partial differential equations, substructuring, variational techniques.**

I. Introduction

TWO decades after the work of Horn and Schunck [19], both the mathematical understanding and algorithmic implementations of variational approaches to optical-flow computation have reached a stage where they outperform alternative approaches in many respects. Beginning with the work of Nagel [26], [27], more and more advanced versions of the prototypical approach of Horn and Schunck within the rich class of convex functionals have been developed including anisotropic and nonlinear regularization preserving motion boundaries [36]. Concerning benchmark experiments [21], they compute accurate optical flow everywhere in the image plane [36]. More robust local evaluation schemes, as well as spatiotemporal coherency, can be exploited within the same mathematical framework [4], [37].

A recurring argument against this class of approaches refers to the computational costs introduced by variational regularization. In our opinion, this argument is strongly misleading since it neglects the costs of alternative approaches related to heuristic post processing of locally computed motion data (interpolation and segmentation). Moreover, besides computer vision, in many fields of application, like medical imaging or remote sensing, variational regularization is the only mathematically sound way for taking into account prior knowledge about the structure of motion fields. This motivates our work on fast algorithms for *variational* optical-flow computation.

In this context, the most common approach to accelerate computations is multigrid iteration [29], [38]. Again, beginning with early work by Terzopoulos and Enkelmann, much progress has been made during the last years [9], [13], [18], [20], [34], and current advanced implementations run in real-time for 200×200 pixel sized image sequences on standard PC hardware [4]. Nevertheless, since the number of pixels per frame steadily increase in applications—e.g., 1500×700 pixels/frame in fluid mechanics [22], and even more in three-dimensional (3-D) medical image sequences—parallelization of computations is inevitable. Due to the *nonlocal* nature of variational models, however, this is not a trivial task.

To illustrate the main difficulty of parallelizing variational optic flow approaches, Fig. 1(b) depicts the result of an *ad hoc* parallelization where the variational problem was *independently* solved in each subregion. Due to the above-mentioned global nature of variational motion estimation, strong artefacts arise at the boundaries of the subregions. In contrast, our approach below solves the *original* variational problem by iteratively exchanging data on the one-dimensional (1-D) boundaries of adjacent subregions.

A. Contribution and Organization

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

1) suitability for the implementation on PC clusters through the minimization of interprocess communication;
2) availability of a mathematical framework as basis for generalizations to the whole class of linear and nonlinear variational models characterized in [36].

Our approach draws upon the general mathematical literature on domain decomposition, and especially upon *substructuring methods* in connection with the solution of partial differential equations [5], [30], [33]. After introducing some necessary mathematical prerequisites in Section II, we specifically derive in Section III an approach for computing the *global* variational solution in terms of an arbitrary number of *local* variational solutions, each of which can be computed in parallel on the *partitioned* image domain (Fig. 2).

Manuscript received June 18, 2003; revised August 14, 2004. This work was supported by the Deutsche Forschungsgemeinschaft (DFG) under Grant Schn457/4. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Tamas Sziranyi.

T. Kohlberger and C. Schnörr are with the Computer Vision, Graphics, and Pattern Recognition Group, Department of Mathematics and Computer Science, University of Mannheim, D-68131 Mannheim, Germany (e-mail: timo.kohlberger@uni-mannheim.de; schnoerr@uni-mannheim.de).

A. Bruhn and J. Weickert are with the Mathematical Image Analysis Group, Department of Mathematics and Computer Science, Saarland University, D-66041 Saarbrücken, Germany (e-mail: bruhn@mia.uni-saarland.de; weickert@mia.uni-saarland.de).

Digital Object Identifier 10.1109/TIP.2005.849778

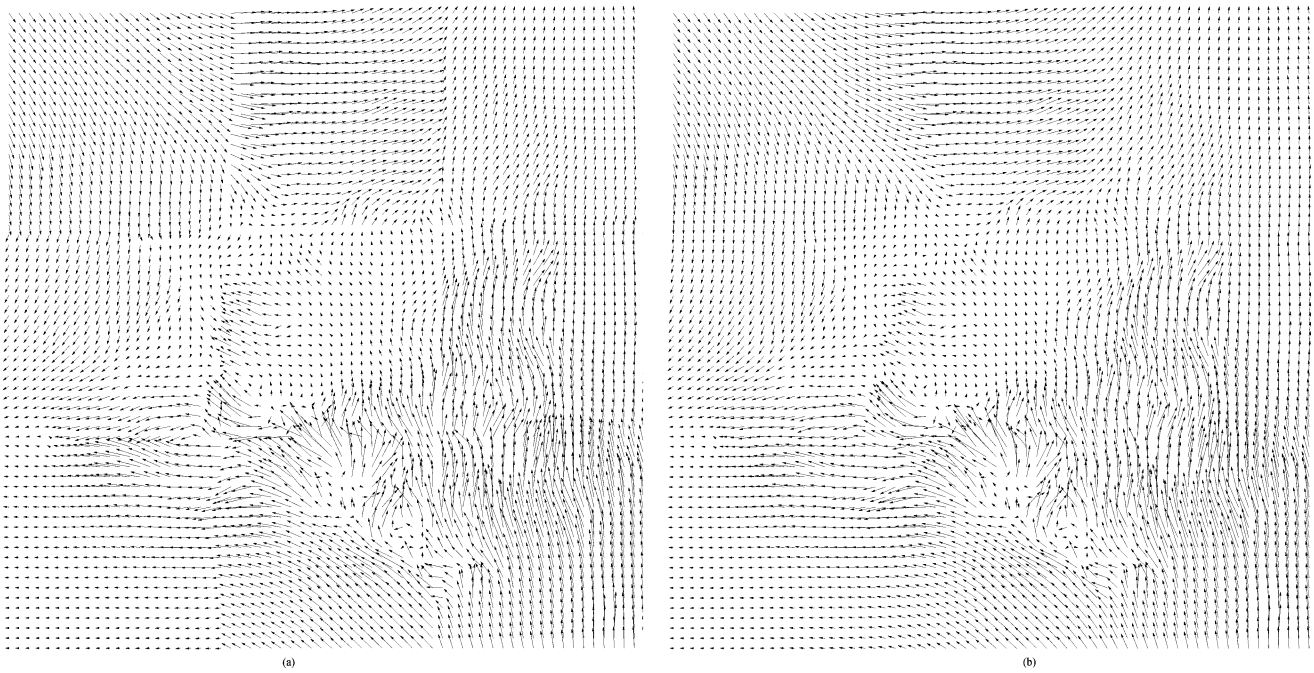


Fig. 1. (a) True solution of an optic flow problem with global smoothness assumption. (b) Result of an *ad hoc* problem decomposition.

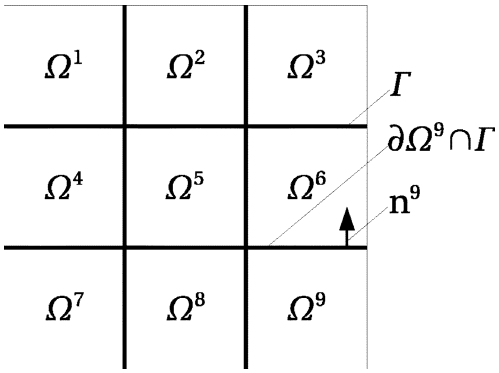


Fig. 2. Example of a squared 3×3 partitioning of the image plane Ω into subdomains Ω^i with shared boundaries Γ^i.

Concerning PC clusters as the target architecture for implementations, an important feature of our approach is that interprocess communication is minimized by restricting the exchange of data to a *lower dimensional* interface Γ. This requires a careful treatment of the variational models within each subdomain (boundary conditions and discretization).

The results of Section III are then generalized to optical-flow computation by applying the common abstract variational formulation (see, e.g., [31]) in Section IV. Subsequently, a proper discretization with *finite elements* is described.

Furthermore, this formulation ensures the applicability of our approach 1) to the whole class of quadratic variational models as characterized in [36] and 2) to nonquadratic discontinuity-preserving convex functionals [36] which can be minimized by means of suitable sequences of quadratic functional approximations [35], [16].

In Section V, we report experimental results of a parallel implementation on a PC cluster for image sequences of (spatial) sizes 512×512 and 2000×2000, and discuss the main characteristics of our approach: preconditioning the system of interface variables, including a coarse grid correction step, dependency of the convergence rate on the granularity of the domain partition, and the scalability behavior for four to up to 144 processors. The results show that our approach provides a basis for the computation of two-dimensional (2-D) optical flow in real-time as well as for large-scale applications in other fields including 3-D medical imaging, remote sensing and experimental fluid mechanics.

II. Preliminaries and Model Problem

A. Mathematical Preliminaries and Notation

In this section, we introduce some notation and concepts necessary for the following. All details can be found in standard textbooks like, e.g., [1] and [6].

Let $\Omega, \Omega^1, \Omega^2, \ldots \subset \mathbb{R}^2$ denote opened and bounded domains with "sufficiently smooth" (e.g., Lipschitz continuous) boundaries $\partial\Omega, \partial\Omega^1, \partial\Omega^2 \ldots$ and exterior unit normals $n, n^1, n^2, \ldots$. Furthermore, $\{\Omega^i | i = 1, \ldots, N\}$ is a nonoverlapping partition of the image plane Ω, i.e., $\Omega = \bigcup_i \Omega^i$, $\Omega^i \cap \Omega^j = \emptyset, \forall j \neq i$, and the set of shared boundaries is defined by $\Gamma := \bigcup_i \partial\Omega^i \setminus \partial\Omega$, see Fig. 2 for an example.

We need the usual Sobolev space for second-order elliptic boundary value problems

$$V = H^1(\Omega) = \{v \in L^2(\Omega) : \partial^\alpha v \in L^2(\Omega),\ 0 \leq |\alpha| \leq 1\}.$$

The corresponding scalar product is defined as

$$(u, v)_1 = \sum_{|\alpha| \leq 1} (\partial^\alpha u, \partial^\alpha v)$$

with the scalar product of $L^2(\Omega)$ denoted with

$$(u, v) = \int_\Omega u(x)v(x)\, dx.$$

We do not need the notion of "traces" and "trace spaces" in the following. Hence, we loosely speak of functions with vanishing boundary values

$$V_0 = H_0^1(\Omega) \subset H^1(\Omega).$$

Likewise, we use the symbolic notation

$$\int_\Gamma \frac{\partial u}{\partial n} v\, ds = \langle \partial_n u, v\rangle_\Gamma$$

for the duality pairing with respect to the trace space $H^{1/2}(\Gamma)$ and its dual, and call $\partial_n u$ a function, as well.

Furthermore, we need *any extension* $P : u_\Gamma \to u \in V$ of boundary values u_Γ on Γ to a function $u \in V$ such that $u|_\Gamma = u_\Gamma$.

For $u, v \in H^1(\Omega)$ and $\Delta u \in L^2(\Omega)$, the following version of Green's formula holds:

$$\int_\Omega \nabla u \cdot \nabla v\, dx = (-\Delta u, v) + \langle \partial_n u, v\rangle_{\partial\Omega}. \tag{1}$$

Throughout this paper, all functions are discretized with standard conforming piecewise linear finite elements. To simplify notation, we use the *same symbols* for some function $v = v(x)$ and the coefficient vector $v \in \mathbb{R}^N$ representing the approximation of $v(x)$ in the subspace spanned by the piecewise linear basis functions $\{\phi(x)\}_{i=1,\ldots,N}$

$$v(x) \in H^1(\Omega) \quad \leftrightarrow \quad v \in \mathbb{R}^N \quad \leftrightarrow \quad \sum_{i=1}^N v_i \phi_i(x). \tag{2}$$

Furthermore, we use the *same symbol* for the vector obtained by discretizing the action of some *linear* functional on some function $v(x)$. For example, we simply write f and w for the discretized versions of the linear functionals (f, v) and $a(w, v)$ [with $a(\cdot, \cdot)$ being a bilinear form and $w(x)$ fixed].

Again, we refer to standard textbooks like, e.g., [6], [32], for the discretization of boundary value problems with finite elements.

B. Model Problem: The Definite Helmholtz Equation

For the reader's convenience, we introduce in this section all relevant concepts first for a model problem closely related to the prototypical variational problem (34) for optical-flow computation. The generalization of the results in Section IV then will be straightforward.

Let $V = H^1(\Omega)$. We consider the following problem:

$$J(u) = \inf_{v\in V} \int_\Omega \left[(v - f)^2 + \lambda|\nabla v|^2\right] dx. \tag{3}$$

Vanishing of the first variation yields

$$\frac{d}{d\tau} J(u + \tau v)\Big|_{\tau=0} = a(u, v) - (f, v) = 0\,, \quad \forall v \in V$$

with the bilinear form

$$a(u, v) = \int_\Omega \left[uv + \lambda \nabla u \cdot \nabla v\right] dx.$$

Since

$$a(v, v) \geq \min\{1, \lambda\}\|v\|_V^2\,, \quad \forall v \in V$$

J is strictly convex, and the global minimum is the unique solution to the variational equation:

$$a(u, v) = (f, v)\,, \quad \forall v \in V. \tag{4}$$

By applying (1), partial integration yields the Euler–Lagrange equation along with the *natural* boundary condition [14]

$$Lu = -\lambda\Delta u + u = f \quad \text{in } \Omega\,, \quad \partial_n u = 0 \quad \text{on } \partial\Omega. \tag{5}$$

The solution u to (4) is the so-called weak solution to (5).

Discretization yields a linear system as the algebraic counterpart of (4)

$$Au = f \tag{6}$$

with a symmetric, sparse, and positive definite matrix A.

C. Nonhomogeneous Boundary Conditions

The decomposition of problem (5) into a set of parallel solvable problems (Section III) requires boundary conditions different from the natural boundary condition in (5). We will collect necessary details in the following section.

1) Dirichlet Conditions: Suppose we wish to have $u|_\Gamma = u_\Gamma$ on Γ, with some given function u_Γ

$$Lu = f \quad \text{in } \Omega, \quad \partial_n u = 0 \quad \text{on } \partial\Omega \setminus \Gamma, \quad u = u_\Gamma \text{ on } \Gamma. \tag{7}$$

To obtain the corresponding variational formulation as basis of a proper finite element discretization, we define the subspace

$$V_{0|\Gamma} = \{v \in V : v|_\Gamma = 0\}.$$

The variational formulation of (7) then reads: Find $u_{0|\Gamma} \in V_{0|\Gamma}$ such that

$$a(u_{0|\Gamma}, v) = (f, v) - a(Pu_\Gamma, v)\,, \quad \forall v \in V_{0|\Gamma}. \tag{8}$$

The desired solution is $u = u_{0|\Gamma} + Pu_\Gamma \in V$, with an arbitrary extension Pu_Γ.

2) Neumann Conditions: Alternatively, suppose we wish to have $\partial_n u = u_n$ on Γ, with a given function u_n

$$Lu = f \quad \text{in } \Omega\,, \quad \partial_n u = 0 \quad \text{on } \partial\Omega \setminus \Gamma\,, \quad \partial_n u = u_n \quad \text{on } \Gamma. \tag{9}$$

The corresponding variational formulation reads: Find $u \in V$ such that

$$a(u,v) = (f,v) + \langle u_n, v\rangle_\Gamma\,, \quad \forall v \in V. \tag{10}$$

III. Problem Decomposition and Parallelization

A. Two Domains

1) Approach: Let $\Omega^1 \cup \Omega^2$ be a partition of Ω with a common boundary $\Gamma = \partial\Omega^1 \cap \partial\Omega^2$, as detailed in Section II-A. We denote the corresponding function spaces with V^1, V^2. In the following, superscripts refer to subdomains.

We wish to represent u from (5) 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

$$u(x) = \begin{cases} u^1(x) & x \in \Omega^1 \\ u^2(x) & x \in \Omega \setminus \Omega^1 \end{cases} \tag{11}$$

obviously holds if the following is true:

$$Lu^1 = f^1 \quad \text{in } \Omega^1 \qquad \partial_{n^1} u^1 = 0 \quad \text{on } \partial\Omega^1 \cap \partial\Omega \tag{12}$$
$$Lu^2 = f^2 \quad \text{in } \Omega^2 \qquad \partial_{n^2} u^2 = 0 \quad \text{on } \partial\Omega^2 \cap \partial\Omega \tag{13}$$
$$u^1 = u^2 \quad \text{on } \Gamma \tag{14}$$
$$\partial_{n^1} u^1 = -\partial_{n^2} u^2 \quad \text{on } \Gamma. \tag{15}$$

We observe that (5) cannot simply be solved by separately computing u^1 and u^2 in each domain Ω_1, Ω_2 because the natural boundary conditions have to be changed on Γ, due to (14) and (15).

As a consequence, in order to solve the system of (12)–(15), we equate the restriction to the interface Γ of the two solutions u^1, u^2 to (12) and (13), due to (14), $u_\Gamma := u^1|_\Gamma = u^2|_\Gamma$, and substitute u_Γ into (15). This will be achieved by means of the *Steklov–Poincaré operator* introduced in the following. Once the resulting equation has been solved for u_Γ, the functions u^1 and u^2 then follow from the substitution of u_Γ back into (12) and (13) with boundary condition (14). This approach is known as *substructuring*, cf., e.g., [33].

2) Steklov–Poincaré Operator S: In the previous section, we have shown that in order to solve system (12)–(15), we have to make explicit the dependency between $\partial_n u|_\Gamma$ and $u|_\Gamma$ of the solution u to a boundary value problem.

Let u be the solution to problem (7). We decompose u into two functions

$$u = u_0 + u_f$$

which are the unique solutions to the following problems:

$$Lu_0 = 0 \text{ in } \Omega, \quad \partial_n u_0 = 0 \text{ on } \partial\Omega \setminus \Gamma, \quad u_0 = u_\Gamma \text{ on } \Gamma \tag{16}$$
$$Lu_f = f \text{ in } \Omega, \quad \partial_n u_f = 0 \text{ on } \partial\Omega \setminus \Gamma, \quad u_f = 0 \text{ on } \Gamma. \tag{17}$$

Clearly, we have $u_f \in V_{0|\Gamma}$ and

$$u|_\Gamma = u_0|_\Gamma \tag{18}$$
$$\partial_n u = \partial_n u_0 + \partial_n u_f. \tag{19}$$

The definition of the *Steklov–Poincaré operator* S is (cf. e.g., [30])

$$S : u_\Gamma \to \partial_n u_0|_\Gamma. \tag{20}$$

Applying this mapping to the solutions u^1, u^2 of (12) and (13) in the domains Ω^1 and Ω^2, respectively, (15) becomes

$$(S^1 + S^2)u_\Gamma + \partial_{n^1} u_f^1|_\Gamma + \partial_{n^2} u_f^2|_\Gamma = 0 \tag{21}$$

with $u_\Gamma = u^1|_\Gamma = u^2|_\Gamma$ due to (14). Equation (21) is denoted as the *interface equation*. It remains to solve this equation for u_Γ. Since S is known to be a symmetric, and positive definite operator, preconditioned conjugate gradient (PCG) methods can be applied for solving (21), which will be detailed later on. To this end, we have to make explicit how the action of S and S^{-1}, respectively, can be computed. In the following two sections we show that this amounts to solving two associated boundary value problems.

3) Action of S: By (8), the variational formulation of problem (16) reads

$$\begin{aligned} a(u_{0|\Gamma}, v) &= -a(Pu_\Gamma, v), \ \forall v \in V_{0|\Gamma} \\ u_0 &= u_{0|\Gamma} + Pu_\Gamma \in V. \end{aligned} \tag{22}$$

Discretization yields a linear system for $u_{0|\Gamma}$ (cf. the convention stated after (1) regarding notation)

$$A_{II} u_{0|\Gamma} = -Pu_\Gamma \tag{23}$$

where index I refers to all nodal variables excluding those on Γ. The extension operator P supplements the boundary values u_Γ with zeros as values of interior nodal variables, and, thus, defines, by using (2), a function $Pu_\Gamma \in V$.

Let us compare the systems (23) and (6), the latter corresponding to the boundary value problem (5). To this end, we decompose the linear system (6) according to $(\overline{\Omega} \setminus \Gamma) \cup \Gamma$

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} u_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I \\ f_\Gamma \end{pmatrix}.$$

Note that the dimension of this linear system is larger because v runs through V in (4), whereas in (22), v only varies in $V_{0|\Gamma}$.

Now, consider $u_0 = u_{0|\Gamma} + Pu_\Gamma$, with $u_{0|\Gamma}$ from (22) and (23), respectively, and let v vary in V. Note that for $v \in V_{0|\Gamma} \subset V$, $a(u_0, v) = a(u_{0|\Gamma} + Pu_\Gamma, v) = a(u_{0|\Gamma}, v) + a(Pu_\Gamma, v) = 0$, due to (22). For $v|_\Gamma \neq 0$ and taking into consideration $Lu_0 = 0$ from (16), we obtain by (1)

$$a(u_0, v) = \langle \partial_n u_0, v\rangle_\Gamma\,, \quad \forall v \in V. \tag{24}$$

Since $u_0|_\Gamma = u_\Gamma$ due to (16), discretization of this variational equation yields the linear system

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} (u_0)_I \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} 0 \\ \partial_n u_0|_\Gamma \end{pmatrix} \tag{25}$$

from which we conclude by algebraically eliminating $(u_0)_I$ [cf. definition (20)]

$$Su_\Gamma = (A_{\Gamma\Gamma} - A_{\Gamma I} A_{II}^{-1} A_{I\Gamma}) u_\Gamma = \partial_n u_0|_\Gamma \tag{26}$$

which is also known as the *Schur complement* of A. Hence, the *action of* S on some boundary data u_Γ involves the solution of problems (22) and (23).

4) Action of S^{-1}: In order to make the *action of* S^{-1} explicit, as well, we may formally invert (26). Since S is dense, this is not advisable, however. Therefore, in practice, one solves the Neumann problem (24) for u_0 with $f = 0$ and *given* boundary data $\partial_n u_0$ (compare with (10)) and obtains by restriction to the interface Γ : $u_\Gamma = u_0|_\Gamma$. Alternatively, this can also be algebraically derived from (25) by the following factorization:

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{\Gamma I}A_{II}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{II} & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & A_{II}^{-1}A_{I\Gamma} \\ 0 & I \end{pmatrix}.$$

Inverting this matrix yields

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix}^{-1} = \begin{pmatrix} I & -A_{II}^{-1}A_{I\Gamma} \\ 0 & I \end{pmatrix} \begin{pmatrix} A_{II}^{-1}0 & \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -A_{\Gamma I}A_{II}^{-1} & I \end{pmatrix}.$$

Hence

$$S^{-1}\partial_n u_0|_\Gamma = (\,0 \quad I\,) \begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ I \end{pmatrix} \partial_n u_0|_\Gamma = u_0|_\Gamma = u_\Gamma. \quad (27)$$

5) Interface Equation: Using the results of the previous sections, we return now to (21) in connection with solving the system of (12)–(15).

Suppose the boundary values $u_\Gamma = u^1|_\Gamma = u^2|_\Gamma$ on the interface Γ separating Ω^1 and Ω^2 were known. Then, u^1 and u^2 within Ω^1 and Ω^2, respectively, can be exactly computed as discussed for problem (7). Here, $u_{0|\Gamma} = u_I^i$ is given by

$$u_I^i = (A_{II}^i)^{-1}(f_I^i - A_{\Gamma I}^i u_\Gamma)\,, \quad i = 1, 2.$$

Thus, it remains to compute the unknown boundary function $u_\Gamma := u^1|_\Gamma = u^2|_\Gamma$. This will be done by solving (15) using the formulation (21).

To set up (21), we have to compute $\partial_{n^i} u_f^i$, $i = 1, 2$. This can be reached by the same procedure used to compute (24). Since $Lu_f^i = f^i$, $i = 1, 2$, we obtain

$$a(u_f^i, v) = (f^i, v) + \langle \partial_{n^i} u_f^i, v\rangle_\Gamma, \quad i = 1, 2, \quad \forall v \in V.$$

Discretization yields the linear systems

$$\begin{pmatrix} A_{II}^i & A_{I\Gamma}^i \\ A_{\Gamma I}^i & A_{\Gamma\Gamma}^i \end{pmatrix} \begin{pmatrix} u_f^i \\ 0 \end{pmatrix} = \begin{pmatrix} f_I^i \\ f_\Gamma^i + \partial_{n^i} u_f^i|_\Gamma \end{pmatrix}, \quad i = 1, 2. \quad (28)$$

Due to the system (12)–(15), we have to solve these two linear systems simultaneously, along with the system (25) applied to either domain Ω_i, $i = 1, 2$. Since $u^i = u_0^i + u_f^i$, summation of the two systems (25) and (28) 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

$$\begin{pmatrix} A_{II}^1 & 0 & A_{I\Gamma}^1 \\ 0 & A_{II}^2 & A_{I\Gamma}^2 \\ A_{\Gamma I}^1 & A_{\Gamma I}^2 & A_{\Gamma\Gamma}^1 + A_{\Gamma\Gamma}^2 \end{pmatrix} \begin{pmatrix} u_I^1 \\ u_I^2 \\ u_\Gamma \end{pmatrix} = \begin{pmatrix} f_I^1 \\ f_I^2 \\ f_\Gamma + \partial_{n^1} u^1|_\Gamma + \partial_{n^2} u^2|_\Gamma \end{pmatrix} \quad (29)$$

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 (29), we conclude that (15) holds if

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

By applying (26), we finally obtain

$$(S^1 + S^2)u_\Gamma = f_\Gamma - A_{\Gamma I}^1(A_{II}^1)^{-1}f_I^1 - A_{\Gamma I}^2(A_{II}^2)^{-1}f_I^2 \quad (30)$$

which is also known as the *Schur complement problem* of (6). Recall that (30) was derived by substituting (12)–(14) into (15). Accordingly, imposing the solution u_Γ to (30) as boundary conditions as required in (14), the functions u^1 and u^2 can be computed from (12) and (13) such that (11) holds! In addition, the computation of $(S^1 + S^2)u_\Gamma = S^1u_\Gamma + S^2u_\Gamma$, needed during CG iteration, can be carried out in parallel.

B. Multiple Domains

In this section, we generalize the results to the case of multiple subdomains $\Omega^1, \Omega^2, \ldots$ (see Fig. 2). Furthermore, we discuss preconditioners for the interface equation, a critical issue for an efficient parallel implementation of the overall approach.

1) Interface Equation: Let R^i denote the restriction of the vector of nodal variables u_Γ on the interface Γ to those on $\partial\Omega^i \cap \Gamma$. Analogously to the case of two domains detailed above, the interface equation for multiple domains reads

$$\left(\sum_i (R^i)^\top S^i R^i\right) u_\Gamma = f_\Gamma - \sum_i (R^i)^\top A_{\Gamma I}^i (A_{II}^i)^{-1} f_I^i. \quad (31)$$

Once the values on in the interface u_Γ are known, the inner nodal variables can be determined by

$$u_{II}^i = (A_{II}^i)^{-1}(f_I^i - A_{\Gamma I}R^i u_\Gamma). \quad (32)$$

2) Interface Preconditioners: 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 [30]. 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. [5], [33]), we examined the *Neumann–Neumann preconditioner (NN)* [2], [7], [23] and

the *balancing-Neumann–Neumann preconditioner (BNN)* [8], [24], [25]. These preconditioners applied in connection with conjugate gradient iteration [15] preserve the symmetry of S and have natural extensions to more general problems related to 3-D image sequences or unstructured geometries and/or triangulations.

The NN preconditioner reads

$$P_{\mathrm{NN}}^{-1} := D\left(\sum_i (R^i)^\top \left(S^i\right)^{-1} R^i\right) D \tag{33}$$

where D denotes a diagonal scaling matrix whose entries D_{jj} are the reciprocals of the number of subdomains shared by nodes j on Γ. Note that the computation of $(S^i)^{-1}$ leads to a local problem in subdomain Ω^i [see Section III-A, 4)] which can be solved in parallel for all subdomains.

The BNN preconditioner reads

$$P_{\mathrm{BNN}}^{-1} := (I - (R^0)^\top (S^0)^{-1} R^0 S) P_{\mathrm{NN}}^{-1} (I - S(R^0)^\top \cdot (S^0)^{-1} R^0) + (R^0)^\top (S^0)^{-1} R^0.$$

In comparison with (33), this preconditioner additionally carries out a correction step (denoted as "balancing" in literature) before and after the application of the NN preconditioner on a coarse grid given by the partition of the domain Ω into subdomains (see Fig. 2).

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 a particular node, i.e.

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

Then, S^0 is defined by $S^0 := R^0 S (R^0)^\top$.

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.

IV. Variational Optical Flow Computation

We generalize the results obtained so far to optical-flow computation. To this end, we replace model problem (3) by the variational approach (34). As explained in Section I, this approach represents a large class of alternative approaches to which our decomposition approach can be applied.

A. Variational Problem

In order to point out the common problem structure with the model problem discussed in Section II-B, we denote in this section the image function with $g(x)$, and keep the symbol f for the linear functional induced by the image data which in connection with optical-flow computation differs from (f, v) in Section II-B [see (37)].

Throughout this section, $\nabla = (\partial_{x_1}, \partial_{x_2})^\top$ 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 $V = H^1(\Omega) \times H^1(\Omega)$ [see [31]].

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

$$J(u) = \inf_{v \in V} \int_\Omega \left[(\nabla g \cdot v + \partial_t g)^2 + \lambda \left(|\nabla v_1|^2 + |\nabla v_2|^2 \right) \right] dx. \tag{34}$$

Analogously to the derivation in Section II-B, vanishing of the first variation of the functional J in (34) yields the variational equation

$$a(u, v) = f(v), \quad \forall v \in V \tag{35}$$

where

$$a(u,v) = \int_\Omega \left[(\nabla g \cdot u)(\nabla g \cdot v) + \lambda (\nabla u_1 \cdot \nabla v_1 + \nabla u_2 \cdot \nabla v_2) \right] dx \tag{36}$$

$$f(v) = -\int_\Omega \partial_t g \nabla g \cdot v dx. \tag{37}$$

Under weak conditions with respect to the image data g, the existence of a constant $c > 0$ was proven in [31] such that

$$a(v, v) \geq c \|v\|_V^2, \quad \forall v \in V.$$

As a consequence, J in (34) is strictly convex and its global minimum u is the unique solution to the variational (35). Partially integrating in (35) by means of (1), we derive the *system* of Euler–Lagrange equations

$$Au = f \quad \text{in} \quad \Omega, \qquad \partial_n u = 0 \quad \text{on} \quad \partial\Omega \tag{38}$$

where

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

We emphasize that the prototypical problem (34) can be easily generalized to other convex functionals as characterized in [36]. In the experiments presented in the following, we have utilized the *CLG* approach [4], which replaces the first term in (34) by the *spatiotemporal structure tensor* [28], [37], with smoothness parameter ρ. Modifying (36) and (37) accordingly, their discrete representations (39) can be automatically computed by discretization with finite elements, and our decomposition approach for parallelization can be immediately applied.

B. Discretization

To approximate the vector field u numerically, (35) is discretized by piecewise linear finite elements over the triangulated section Ω of the image plane. 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: $u = (u_1^\top, u_2^\top)^\top$ (cf. Section II-A). Taking into consideration the symmetry of the bilinear form (36), this induces the following block structure of the discretized version $Au = f$ of (35)

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \tag{39}$$

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. Apart from the block structure of this linear system $Au = f$, all results of Sections III-A and III-B carry over verbatim.

V. Parallel Implementation and Experiments

A. Parallel Implementation

As already mentioned, the great advantage of domain decomposition methods, and especially of substructuring methods in our case, is their inherent coarse-grain parallelism. In particular, since the Steklov–Poincaré operator S can be written as $\left(\sum_i (R^i)^\top S^i R^i\right)$, its application in a PCG iteration for solving (30) or (31), respectively, can be seperated into simultaneous applications of the local operators S_i on $\partial\Omega^i \cap \Gamma$. This mainly amounts to solving local *Dirichlet systems* as detailed in Section III-A, 3). Analogously, the computation of preconditioner P_{NN} can be parallelized in solving the local *Neumann problems* associated with each occurence of $(S^i)^{-1}$, as described in Section III-A, 4), in parallel.

In a parallel framework, those local problems are usually solved in *slave processes* which exchange their local data with a dedicated *master process* carrying out the PCG iteration. In such a setting, the restriction operators R^i, as introduced in Section III-B 1), correspond to a *scatter operation*, and it transposes $(R^i)^\top$ to a *reduce operation* between the master process and the slave processes. In particular, a scatter operation consists of sending to each slave process, associated with subdomain Ω^i, the subset $\partial\Omega^i \cap \Gamma$ of nodal variables. Conversely, the transposed operator $(R_i)^\top$ amounts to a reduce operation, i.e., receiving and adding the nodal values of all local shared boundaries.

Additionally, the coarse operator $S^0 = R^0 S (R^0)^\top$ has to be inverted in connection with the BNN preconditioner P_{BNN}^{-1}. Since S^0 is global, as is S, parallelization is not applicable here. Therefore, S^0 is computed explicitly during initialization. Then, the associated system $S^0 x = y$ is solved serially during PCG iteration. Since S^0 is much smaller than S, the nonparallel computational effort for doing so is negligible.

It remains to find an efficient scheme for calculating the entries of S^0. We use a column-wise initialization by

$$
S^0_j \leftarrow R^0 \left(\sum_i (R^i)^\top S^i R^i\right) (R^0)^\top e_j, \qquad j : 1 \ldots N
$$

$$
e_j : j\text{th unit vector}
$$

which has the inherent disadvantage of the number of S^i applications being equal the number of subdomains. Taking a closer look at $(R^0)^\top$ shows that its jth column affects only the shared boundaries of the subdomain Ω^j, as well as its left, right, upper, and lower neighboring subdomains, if any, as well as the nodal variables at the vertices of the diagonal neighbors. Since the latter ones have turned out to be negligible in practice, the initialization of S^0 can be partially parallelized by computing every fourth column of S^0 in a group of columns corresponding to every second row of the subdomain partition; see Fig. 3 for illustration. With this optimized initialization scheme, the number of initial applications of S^i is always eight, independent of the total number of subdomains.

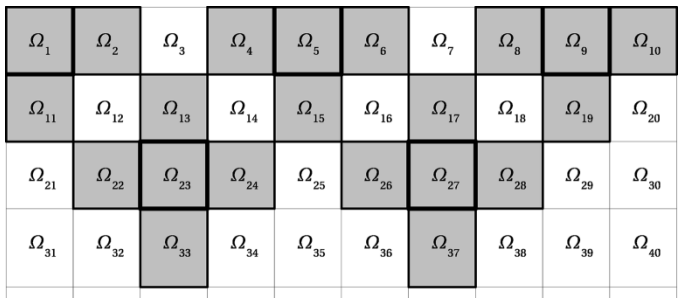


Fig. 3. Initialization scheme for the coarse grid operator S^0 on 10×10 subdomains. Each gray cross at a particular Ω^j depicts the area of nodes where $R^0 S (R^0)^\top e_j$ is nonzero.

TABLE I
CG Versus PCG Iteration of the Interface Problem

Partition	Subdomain size	Preconditioner	Iterations N
2×2	256×256	P_{NN}	6
2×2	256×256	I	42
4×4	128×128	P_{NN}	7
4×4	128×128	I	42

It remains the calculation of the right-hand sides of (30) or (31), respectively, as well as the final calculation of the inner nodal variables by (12)–(15) or (32). Both provide inherent parallelism by solving the corresponding Dirichlet system simultaneously on every subdomain.

B. Experimental Setting and Input Data

We conducted different experiments in order to investigate the following aspects of the described domain decomposition method:

1) effect of interface preconditioning on the convergence rate;
2) comparison of the convergence rates utilizing the NN or BNN preconditioner for different numbers of subdomains;
3) the total runtime and the interprocess communication volume depending on the number of subdomains for each preconditioner in comparison to fast nonparallel multigrid solving.

For ease of implementation, the image plane Ω was partitioned into equally sized, quadratic subdomains Ω^i in all experiments. The parameter values of the CLG motion estimation were $\alpha = 1000$, $\sigma = 2.6$, and $\rho = 1.8$, while the intensity

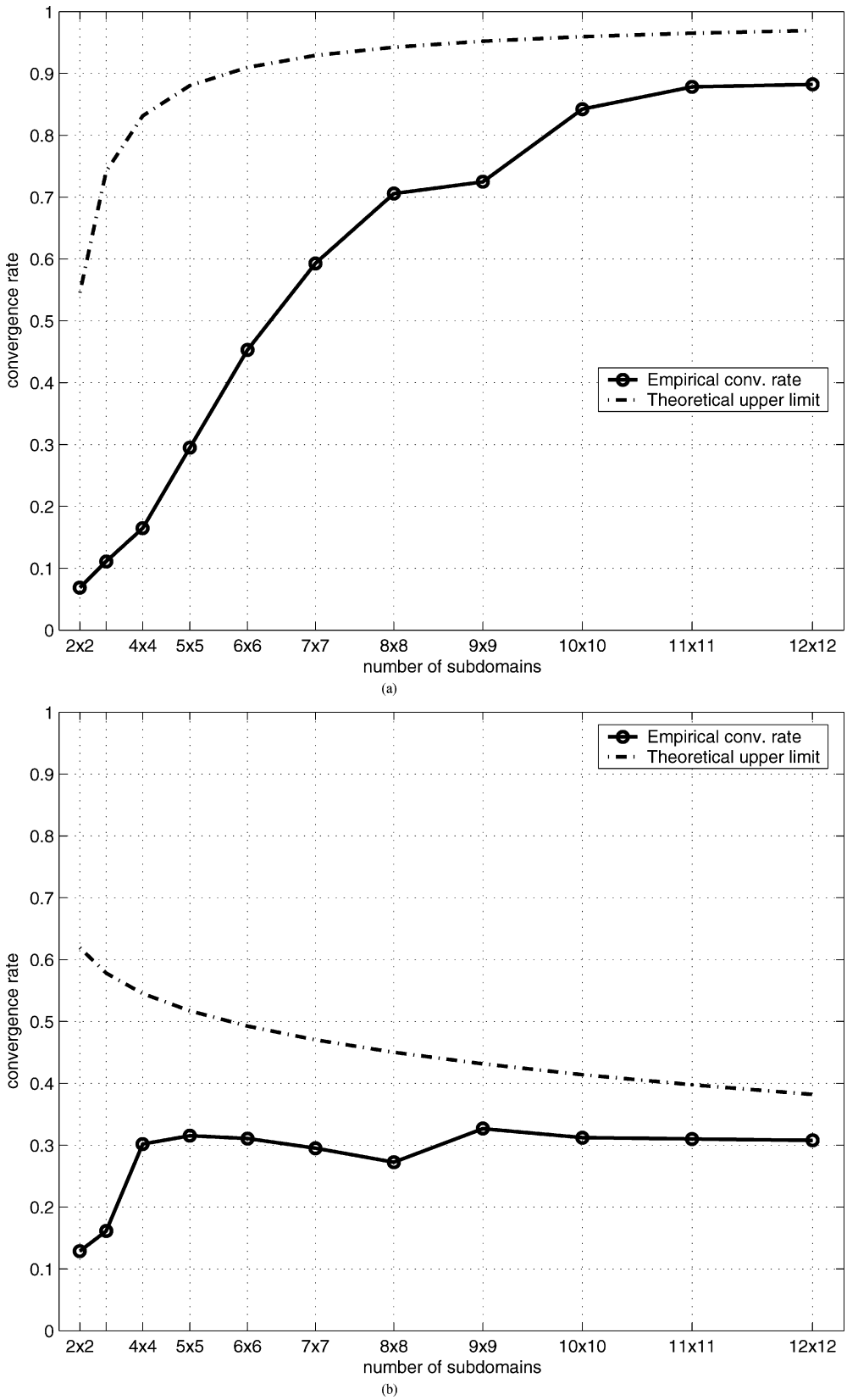


Fig. 4. Experimental convergence rates and theoretical upper limits using (a) the NN preconditioner or (b) the BNN preconditioner.

values were in the range [0,255]. The implementation was realized in C/C++ using the Intel compiler 7.0, with O3 optimization option, and MPI-conform [11], [10] interprocess communication libraries on the Linux OS on standard PC hardware. Vectorization operations [SSE(2) or 3dnow-commands] were not used.

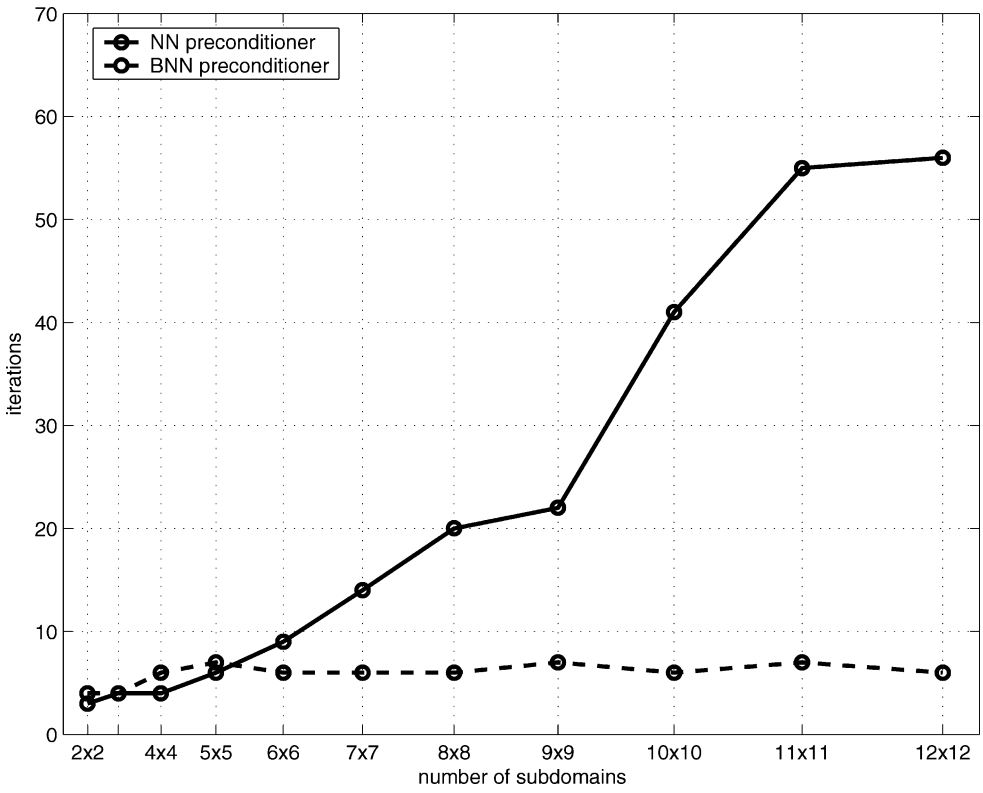


Fig. 5. Number of iterations for an error threshold of 10^{-3} using NN or BNN preconditioning on frame 16 and 17 of the marble sequence (512×512).

C. Effect of Interface Preconditioning

In order to investigate the influence of an interface preconditioner in general, (31) was solved for u_Γ without preconditioning and also by applying the NN preconditioner. As input images frame 16 and 17 of the *marble sequence*.[1] As a local solver, PCG iteration was also used, iterating until the relative residual error was below 10^{-5}. Table I depicts the number of necessary *outer* (P)CG iterations to reach an residual error of 10^{-3}, i.e., $\|\chi - Su_\Gamma^{(N)}\|_2/\|\chi\|_2 < 10^{-3}$, with χ denoting the r.h.s. of (31). It clearly shows that, in agreement with theory [30], the system becomes more and more ill conditioned if the number of subdomains increases. Using the NN preconditioner (33), however, largely compensates this effect and enables shorter computation times through parallelization.

D. Convergence Rate Studies

In this experiment, we compared the linear convergence rate ρ based on the original problem (38) using NN or BNN preconditioning for varying numbers of subdomains N. PCG iteration was also used as local solver here. The convergence rates where determined by $\rho = \sqrt[k]{\epsilon^{(k)}}$, with $\epsilon^{(k)} := \|f - Au^{(k)}\|_2/\|f\|_2$ being the error after k iterations and $u^{(0)} = 0$. The examined rates are depicted in Fig. 4 for each preconditioner (solid lines). They clearly show that the convergence rate using the nonbalancing preconditioner grows with the number of subdomains whereas they remain nearly constant for the preconditioner involving a coarse grid correction step.

[1]Created in 1993 by Michael Otte, Institut für Algorithmen und Kognitive Systeme, University of Karlsruhe, Germany, available at http://i21www.ira.uka.de/image_sequences.

Furthermore, these results are consistent with theoretical upper limits which can be derived from approximations of the condition numbers κ of the preconditioned operators $P_{\mathrm{NN}}^{-1}S$ and $P_{\mathrm{BNN}}^{-1}S$, namely [7], [23] , [25]

$$\kappa(P_{\mathrm{NN}}^{-1}S) \leq C_{\mathrm{NN}}H^{-2}\left(1 + \log\frac{H}{h}\right)^2 \tag{40}$$

and

$$\kappa(P_{\mathrm{BNN}}^{-1}S) \leq C_{\mathrm{BNN}}\left(1 + \log\frac{H}{h}\right)^2 \tag{41}$$

with H denoting the mesh sizes of the coarse grid, which corresponds to the subdomain partition, and h denoting the fine discretization grid, as well as some constants C_{NN}, C_{BNN}. It is well known, cf. e.g., [15, p. 272], that the convergence rate ρ of PCG iteration depends on the condition number by

$$\rho = c(\kappa)\frac{2}{1 + c^2(\kappa)}, \quad \text{with} \quad c(\kappa) = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}. \tag{42}$$

The observations are in agreement with the theory and especially the presence of the H^{-2} term in the bound of the NN preconditioner, leading to a worse convergence for an increasing number of subdomains (which is of order $O(1/H^2)$) whereas with the two-level preconditioner it remains nearly constant due to the influence of the coarse grid couplings.

In a further experiment, we compared the number of iterations k necessary for the residual error $\epsilon^{(k)}$ to fall below 10^{-3} using the NN and BNN preconditioner. Results are shown in Fig. 5. Thus, the BNN preconditioner is much closer to an *optimal* preconditioner making the convergence rate independent w.r.t. both

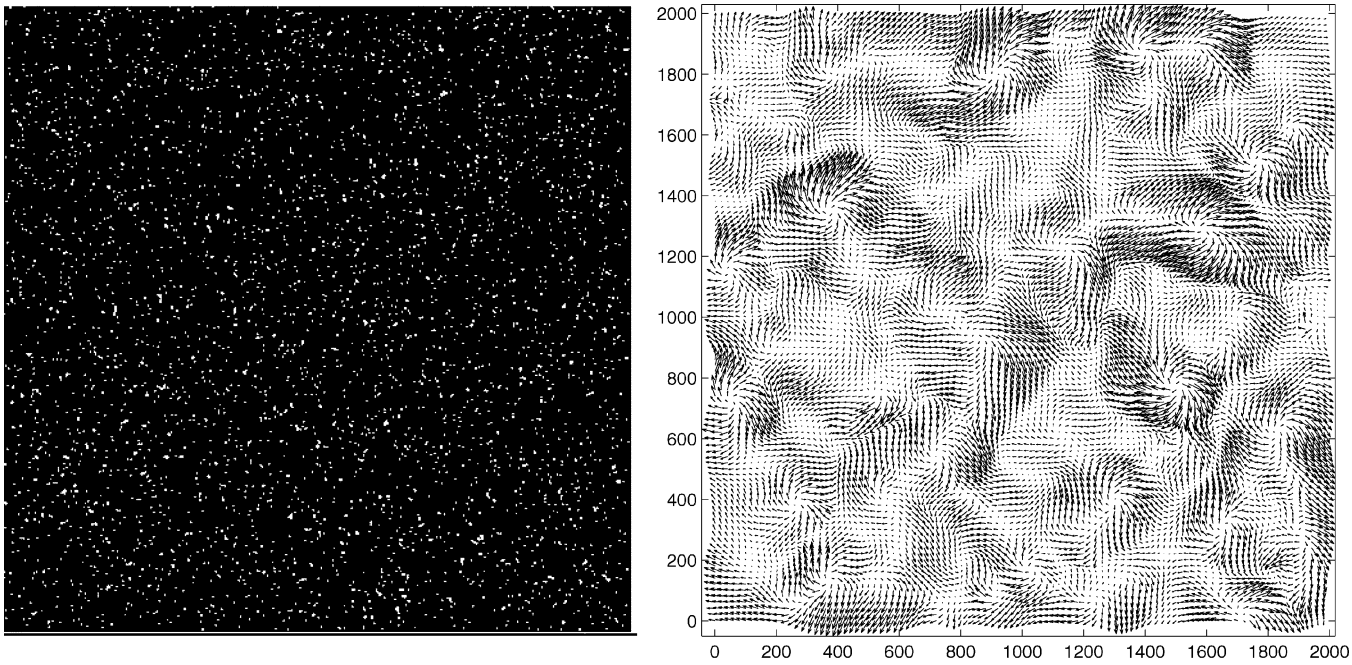


Fig. 6. (a) First frame of the synthetic particle image pair (2000×2000) used as input data in the runtime and communication measurements. (b) Applied synthetic motion field (ground truth).

the pixel meshsize h and the coarse meshsize H by the number of subdomains. However, one iteration using the BNN preconditioner involves two times the application of S and $(S^0)^{-1}$ in addition to the computational effort of the NN preconditioner.

Finally, the experiments have also shown that the proposed algorithm using either preconditioner converges to the global solution of the original problem (up to machine precision) if the local solvers are sufficiently precise.

E. Runtime and Communication Measurements

In order to examine the advantage of substructuring methods compared to nonparallel methods in practice, we conducted experiments on a dedicated PC cluster,[2] i.e., a hybrid distributed memory/shared memory parallel computer, by applying the described parallel implementation on a synthetic 2000×2000 image pair. The outer iteration was stopped when a given error threshold[3] of 10^{-3} relative to a high-precision nonparallel solution was reached . The total runtime, the communication time, as well as the total communication volume, were measured for 2×2 to 12×12 decompositions. As input data, a synthetic particle image pair and a synthetic motion field were used (Fig. 6), as they occur in current particle image velocimetry (PIV) experiments. The runtime measurements were compared to those of an cache-optimized, nonparallel implementation of a multigrid solver [3], being run on the same hardware, the same error threshold, and the same compiler options.

The same multigrid solver was also used for solving the local Dirichlet and the Neumann problems during the outer PCG iteration of the parallel algorithm, as well as for the initial calculation of the right-hand side of interface (31) and for the final calculation of the inner nodal variables (32). Due to the artificial Dirichlet boundary conditions in (22), the corresponding linear systems (23) have shown to be conditioned worse than those of the Neumann problems (27), which leads to a significantly higher number of necessary smoothing iterations and V-cycles in the multigrid solving. To be precise, one V-cycle with two smoothing iterations per resolution level were used in connection with the Neumann problems, whereas four W-cycles for each full multigrid level and two W-cycles else, together with six to eight Gauss–Seidel relaxations were carried out to solve the Dirichlet problems. During the experiments, it turned out to be sufficient to solve the Dirichlet systems for unknowns in a neighborhood of 5–7 nodes to the shared boundaries Γ^i for the second and subsequent (outer) PCG iteration, since the right-hand sides change on those boundaries only. Finally, the coarse systems $S^0 x = y$ occurring in the BNN preconditioner were solved by PCG iteration.

Fig. 7 depicts the measured run and communication times for NN and BNN preconditioning, respectively, in comparison to the nonparallel optimized multigrid solver. All values shown are averages of eight consecutive runs using the same input image pair. The diagram shows that the parallel implementation with NN preconditioning is faster than the multigrid iteration on a single machine for 5×5 subdomains and above, i.e., 26 processors and above. The speed-up factors for 5×5 to 12×12 decompositions are 1.23, 1.77, 1.91, 2.66, 3.17, 2.82, 3.75, and 3.67. Interestingly, the simpler NN preconditioner performs best, in this case, and cannot be improved by BNN preconditioning, in contrast to the experiments discussed above (Figs. 4 and 5). The reason is that for larger image sizes as in Fig. 6, and for a domain decomposition with a reasonable number of processors

[2]HELICS-Cluster, 256 Dual AMD Athlon MP 1.4 GHz processors, Myrinet2000 network, Interdisciplinary Center for Scientific Computing, University of Heidelberg, Germany.

[3]For example, $\|w^{(k)} - \hat{w}\|_2 / \|\hat{w}\|_2 < 10^{-3}$, $w^{(k)}$: solution after k iterations, $\hat{w}$ ground truth solution.

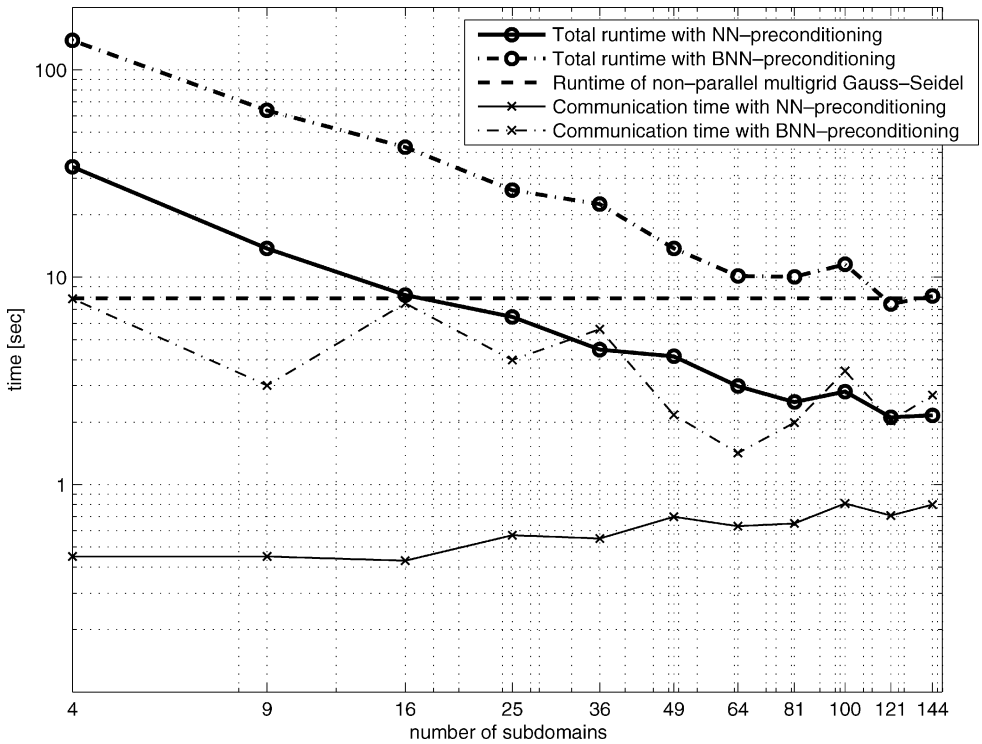


Fig. 7. Runtime measurements using NN preconditioning or BNN preconditioning in comparison to nonparallel multigrid solving on a 2000×2000 image sequence.

(cf. Fig. 7), the corresponding meshsize H is large as well (relative to the discretization parameter h). Consequently, the overall problem is still well-conditioned, and it does not pay to choose the computationally expensive BNN preconditioner.

In addition, considering the communication time measurements in Fig. 7 shows that the communication effort with NN preconditioning increases only moderately with the number of processors. With BNN preconditioning, it even decreases for less then 64 processors. As reason for this behavior we found that, although the total amount of communicated data increases with the number of subdomains, it decreases for each subdomain since the number of shared variables diminishes. In other words, the amount of data exchanged with each processor *sequentially* decreases with a growing number of subdomains. Of course, this is only an advantage if the scatter and reduce operations are sufficiently parallelized. This effect seems to be stronger for BNN preconditioning since those operations occur a multiple times per outer iteration there.

Besides measuring the communication time we also studied the communication volume, i.e., the total number of bytes sent from the master to the slave processes, and vice versa. The communication volume mainly consists of three parts: a) initial distribution of input data, i.e., the image pair, b) exchange of values on shared boundaries nodes, and c) collection of the final vector field from the slave processes. Whereas the communication volumes for a) and c) are nearly fixed, i.e., are nearly independent of the number of subdomains,[4] they grow for b) linearly with the square root of the number of subdomains, which can be nicely observed in Fig. 8. The higher volume using BNN preconditioning results from multiple occurrences of the restriction operators R^i and R^0, and their transposes. The linear dependence on the square root of the number of subdomains is coherent with the fact that the overlap between subdomains is only 1-D whereas the subdomains itself are 2-D.

This observation, and also real communication time measurements, clearly show the advantage of applying the substructuring approach with respect to high scalability, in contrast to other parallelization strategies like, e.g., Schwarz methods with 2-D overlapping subregions.

VI. Conclusion and Further Work

We presented an approach to parallel optical-flow computation by a decomposition of the original problem into independent local subproblems, according to a partition of the underlying domain Ω. We have shown, both in continuous and discretized formulation, the dependency between the Steklov–Poincaré operator S and the associated Dirichlet problem, as well as the dependency between the inverse operator S^{-1} and the associated Neumann problem.

We have demonstrated the general advantage of preconditioning the interface problem, and studied the convergence behavior in utilizing NN and BNN preconditioners for different numbers of subdomains. Finally, we have presented runtime and communication volume measurements of a real-world experiment on a PC cluster, which show that substructuring leads to a significant initial computational effort, in comparison to nonparallel multigrid solving, but provides very good scaling properties due to restricting the interprocess communication to a lower dimensional interface between the subdomains.

[4]In fact, in our implementation, it increases slightly with the number of subdomains for a), since there the distributed regions must have a small overlap in order to obtain the same convolution results on the shared boundary nodes of neighboring local systems.

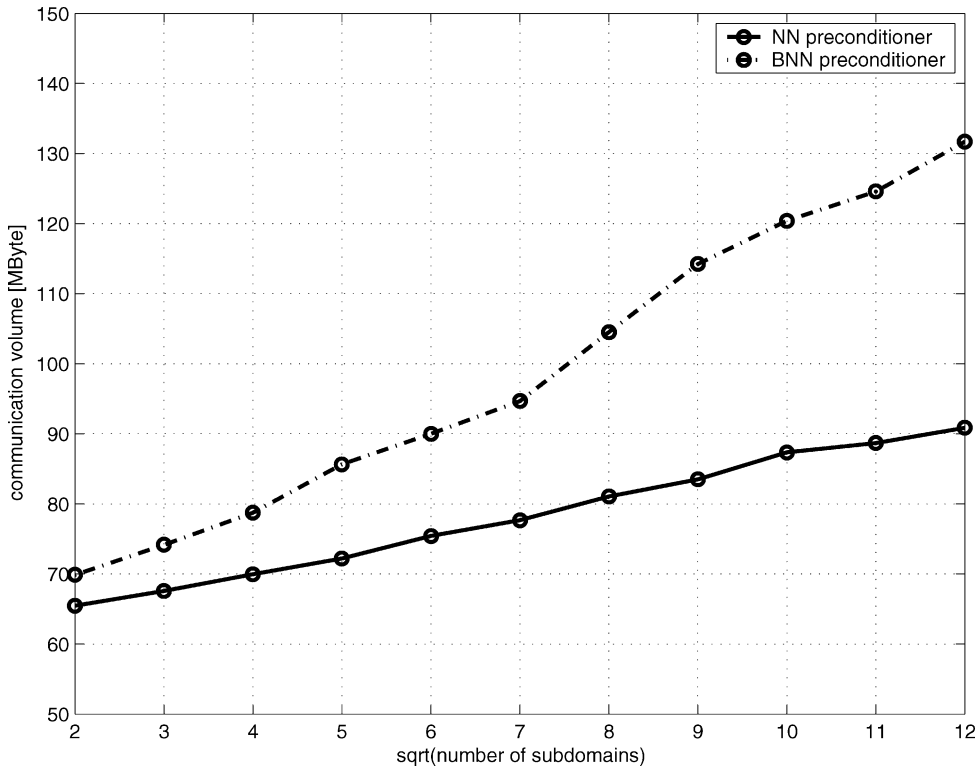


Fig. 8. Measured communication volume (between master and slave processes) using NN or BNN preconditioning.

Due to the components chosen for our approach and the use of conforming finite element discretization, our approach is applicable, in principle, to the whole class of quadratic variational approaches for optical-flow computation and less-structured problems involving irregularly shaped domains and nonuniform triangulations (i.e., experimental fluid dynamics).

By applying our approach to a higher number of processors and by exploiting vectorization and local parallelization in each processing node, real-time 2-D optic-flow estimation on large image sequences will come into reach on standard PC hardware. Furthermore, we will extend our work to nonquadratic variational approaches, like total variation-based regularization, for example, where sequences of quadratic variational approximations are used to compute minimizers.

Finally, it might be worthwile to use multigrid iteration not only as subdomain solver, but to consider parallelization at coarse grid levels [12], [17].

REFERENCES

[1] J. P. Aubin, *Approximation of Elliptic Boundary-Value Problem*. New York: Wiley, 1972.

[2] J.-F. Bourgat, R. Glowinski, P. Le Tallec, and M. Vidrascu, "Variational formulation and algorithm for trace operator in domain decomposition calculations," in *Domain Decomposition Methods*, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, Eds. Philadelphia, PA: SIAM, 1989, pp. 3–16.

[3] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr, "Variational optical flow computation in real-time," *IEEE Trans. Image Process.*, to be published.

[4] A. Bruhn, J. Weickert, and C. Schnörr, "Lucas/Kanade meets Horn/Schunck: Combining local and global optic flow methods," *Int. J. Comput. Vis.*, to be published.

[5] T. F. Chan and T. P. Mathew, "Domain decomposition algorithms," in *Acta Numerica 1994*. Cambridge, U.K.: Cambridge Univ. Press, 1994, pp. 61–143.

[6] P. G. Ciarlet, *The Finite Element Method for Elliptic Problem*. Amsterdam, The Netherlands: North-Holland, 1978.

[7] Y.-H. De Roeck and P. Le Tallec, "Analysis and test of a local domain decomposition preconditioner," in *Proc. 4th Int. Symp. Domain Decomposition Methods for Part. Diff. Equations*, R. Glowinsiki, Y. Kuznetsov, G. Meurant, J Périaux, and O. Widlund, Eds., Philadelphia, PA, 1991, pp. 112–128.

[8] M. Dryja and O. B. Widlund, "Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems," *Commun. Pure Appl. Math.*, vol. 48, pp. 121–155, 1995.

[9] W. Enkelmann, "Investigation of multigrid algorithms for the estimation of optical flow fields in image sequences," *Comput. Vis. Graph. Imag. Process.*, vol. 43, pp. 150–177, 1987.

[10] "Message passing interface forum," in *MPI-2: Extensions to the Message-Passing Interface* Univ. Tennessee, Knoxville, 1995.

[11] "Message passing interface forum," in *MPI: A Message-Passing Interface Standard* Univ. Tennessee, Knoxville, 1995.

[12] U. Gärtel and K. Ressel, "Parallel multigrid: Grid partitioning versus domain decomposition," in *Proc. 10th Int. Conf. Computing Methods in Applied Sciences and Engineering*, R. Glowinski, Ed., New York, 1992, pp. 559–568.

[13] S. Ghosal and P. Vaněk, "A fast scalable algorithm for discontinuous optical flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 2, pp. 181–194, Feb. 1996.

[14] W. Hackbusch, *Theorie und Numerik Elliptischer Differentialgleichungen*. Stuttgart, Germany: B. G. Teubner, 1986.

[15] ——, *Iterative Solution of Large Sparse Systems of Equations*. Berlin, Germany: Springer, 1993.

[16] 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 Trans. Image Process.*, vol. 10, no. 6, pp. 852–864, Jun. 2001.

[17] B. Heise and M. Jung, "Comparison of parallel solvers for nonlinear elliptic problems based on domain decomposition ideas," Johannes Kepler Univ. Linz, Inst. Math., Linz, Austria, Tech. Rep. 494, 1995.

[18] F. Heitz, P. Perez, and P. Bouthemy, "Multiscale minimization of global energy functions in some visual recovery problems," *CVGIP: Image Understanding*, vol. 59, no. 1, pp. 125–134, 1994.

[19] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artif. Intell.*, vol. 17, pp. 185–203, 1981.

[20] S. H. Hwang and S. U. Lee, "A hierarchical optical flow estimation algorithm based on the interlevel motion smoothness constraint," *Pattern Recognit.*, vol. 26, no. 6, pp. 939–952, 1993.

[21] D. J. Fleet, J. L. Barron, and S. S. Beauchemin, "Perfomance of optical flow techniques," *Int. J. Comput. Vis.*, vol. 1, no. 12, pp. 43–77, Feb. 1994.
[22] T. Kohlberger, E. Mémin, and C. Schnörr, "Variational dense motion estimation using the helmholtz decomposition," in *Proc. 4th Int. Conf. Scale-Space Theories in Computer Vision, ScaleSpace*, vol. 2695, LNCS, L. D. Griffin and M. Lillholm, Eds., 2003, pp. 432–448.
[23] P. Le Tallec, Y.-H. De Roeck, and M. Vidrascu, "Domain decomposition methods for large linearly elliptic three dimensional problems," *J. Comput. Appl. Math.*, vol. 34, pp. 93–117, 1991.
[24] J. Mandel, "Balancing domain decomposition," *Commun. Numer. Math. Eng.*, vol. 9, pp. 233–241, 1993.
[25] J. Mandel and M. Brezina, "Balancing domain decomposition: Theory and performance in two and three dimensions," Comput. Math. Group, Univ. Colorado, Denver, Tech. Rep., vol. 1, 1993.
[26] H. H. Nagel, "Constraints for the estimation of displacement vector fields from image sequences," in *Proc. 8th Int. J. Conf. Artif. Intell.*, Karlsruhe, Germany, 1983, pp. 945–951.
[27] ——, "On the estimation of optical flow: Relations between different approaches and some new results," *Artif. Intell.*, vol. 33, pp. 299–324, 1987.
[28] ——, "Extending the 'oriented smoothness constraint' into the temporal domain an the estimation of derivatives of optical flow," in *Proc. Computer Vision ECCV*, vol. 427, O. Faugeras, Ed., Berlin, Germany, 1990, pp. 139–148.
[29] P. Oswald, *Multilevel Finite Element Approximation*. Stuttgart, Germany: B. G. Teubner, 1994.
[30] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*. Oxford, U.K.: Oxford Univ. Press, 1999.
[31] C. Schnörr, "Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class," *Int. J. Comput. Vis.*, vol. 6, no. 1, pp. 25–38, 1991.
[32] H. R. Schwarz, *Methode der Finiten Elemente*. Stuttgart, Germany: B. G. Teubner, 1980.
[33] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition: Parallel Multilevel Methods for the Solution of Elliptic Partial Differential Equations*. Cambridge, U.K.: Cambridge Univ. Press, 1996.
[34] D. Terzopoulos, "Multilevel computational processes for visual surface reconstruction," *Comput. Vis., Graph., Image Process.*, vol. 24, pp. 52–96, 1983.
[35] R. V. Vogel and M. E. Oman, "Iterative methods for total variation denoising," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 227–238, 1996.
[36] J. Weickert and C. Schnörr, "A theoretical framework for convex regularizers in PDE–based computation of image motion," *Int. J. Comput. Vis.*, vol. 45, no. 3, pp. 245–264, 2001.
[37] ——, "Variational optic flow computation with a spatio-temporal smoothness constraint," *J. Math. Imag. Vis.*, vol. 14, no. 3, pp. 245–255, 2001.
[38] P. Wesseling, *An Introduction to Multigrid Methods*. New York: Wiley, 1992.

Timo Kohlberger studied computer engineering at the University of Mannheim, Mannheim, Germany, where, during his studies, he was supported by the German National Merit Foundation, and from which he received the Diplom (Master) degree in computer science in 2001, and where he is currently pursuing the Ph.D. degree.

Since October 2001, he has been with the CVGPR Group of the University of Mannheim. His present research interests include parallelization techniques applied to image motion computation and computer vision, the design of novel approaches in motion estimation, as well as tracking and suveillance systems.

Christoph Schnörr received the Dipl.–Ing. degree in electrical engineering and the Dr.rer.nat. degree in computer science from the Technical University of Karlsruhe, Karlsruhe, Germany, and the Habilitation degree in computer science from the University of Hamburg, Hamburg, Germany, in 1987, 1991, and 1998, respectively.

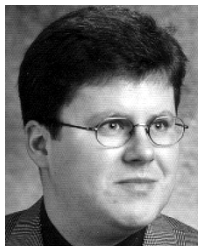
He held the position of Researcher at the Fraunhofer Institute for Information and Data Processing (IITB), Karlsruhe, from 1987 to 1992 and Researcher and Assistant Professor within the Cognitive Systems Group, University of Hamburg. During 1996, he was a Visiting Researcher at the Computer Vision and Active Perception Laboratory, KTH, Stockholm, Sweden. Since 1999, he has been with the Department of Mathematics and Computer Science, University of Mannheim, Mannheim, Germany, where he set up and is the Head of the Computer Vision, Graphics, and Pattern Recognition Group. He serves on the editorial board of the *Journal of Mathematical Imaging and Vision*. His research interests include computer vision, pattern recognition, and related aspects of mathematical modeling and optimization.

Dr. Schnörr is a member of DAGM, SIAM, and SIAGIS. He is Co-Editor-in-Chief of the *International Journal of Computer Vision* and he serves on the editorial board of the *Journal of Mathematical Imaging and Vision*.

Andrés Bruhn received the M.Sc. degree in computer engineering from the University of Mannheim, Mannheim, Germany, in 2001. He is currently pursuing the Ph.D. degree in computer science at Saarland University, Saarbrücken, Germany.

He was a Research Assistant at the University of Mannheim from October to November 2001. Since December 2001, he has been with the Mathematical Image Analysis Group, Department of Mathematics and Computer Science, Saarland University. His current research interests include optical-flow estimation, diffusion filtering, and fast numerical schemes for image processing and computer vision.

Mr. Bruhn won the prestigious Longuet–Higgins Prize at the European Conference on Computer Vision in 2004.



Joachim Weickert received the M.Sc. and Ph.D. degrees in mathematics from the University of Kaiserslautern, Kaiserslautern, Germany, and the habilitation degree in computer science from the University of Mannheim, Mannheim, Germany, in 1991, 1996, and 2001, respectively.

He was a Research Assistant at the University of Kaiserslautern; a Postdoctoral Researcher at the University of Utrecht, Utrecht, The Netherlands, and the University of Copenhagen, Copenhagen, Denmark; and an Assistant Professor at the University of Mannheim. Currently, he is a Full Professor of mathematics and computer science at Saarland University, Saarbrücken, Germany. His research interests include image processing, computer vision, partial differential equations, and scientific computing.