Domain Decomposition for Parallel Variational Optical Flow Computation

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

[1] CVGPR-Group, University of Mannheim, 68131 Mannheim, Germany
{tiko,schnoerr}@uni-mannheim.de, www.cvgpr.uni-mannheim.de

[2] MIA-Group, Saarland University, 66041 Saarbrücken, Germany
{bruhn,weickert}@mia.uni-saarland.de, www.mia.uni-saarland.de

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 inter-process communication is minimized by restricting the exchange of data to a lower-dimensional interface. Our mathematical formulation supports various generalizations to linear/non-linear convex variational approaches, 3D image sequences, spatio-temporal regularization, and unstructured geometries and triangulations. Results concerning the effects of interface preconditioning, inexact subdomain solvers, and the number of subdomains are presented. 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

Overview and Motivation. Two decades after the work of Horn and Schunck [1] 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. Starting with the work of Nagel [2], 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 non-linear regularization preserving motion boundaries [3]. Concerning benchmark experiments [4], they compute accurate optical flow everywhere in the image plane [3]. More robust local evaluation schemes can be exploited within the same mathematical framework [5].

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, 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. Again, starting with early work by Terzopoulos and Enkelmann, much progress has been made during the last years [6,7], and current advanced implementions run in real-time for 200×200 pixel sized image sequences on standard PC-hardware [8]. Nevertheless, since the number of pixels per frame steadily increase in applications – e.g. 1500×700 pixels/frame in fluid mechanics, and even more in 3D medical image sequences – parallelization of computations is inevitable. Due to the *non-local* nature of variational models, however, this is not a trivial task.

Contribution and Organization. We present an approach to the parallelization of variational optical flow computation which fulfils the following requirements: Firstly, suitability for the implementation on PC-clusters through the minimization of inter-process communication. Secondly, availability of a mathematical framework as basis for generalizations to the whole class of linear and non-linear variational models characterized in [3].

Starting points of our work are (i) the variational formulation developed in [9] of the prototypical approach of Horn and Schunck (section 2), and (ii) the general mathematical literature on domain decomposition in connection with the solution of partial differential equations [10,11].
Based on this theory, we derive in section 3 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 plane. An important feature of this approach is that inter-process 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, discretization). In section 4, we confirm the theoretical properties of our approach by numerical experiments and study the effect of its components on the speed of convergence. The results show that our approach provides a basis for the computation of 2D optical flow in real-time as well as for large-scale applications in other fields including 3D medical imaging, remote sensing and experimental fluid mechanics.

2 Variational Approach and Discretization

Following [9], we summarize the variational formulation of the approach of Horn and Schunck [1] and its discretization. This approach serves as a prototype for a large class of approaches to optical flow computation studied in [3].

Throughout this paper, $x = (x_1, x_2)^\top \in \Omega$ denotes some point in the image plane, $g(x)$ the image function, $\nabla = (\partial_{x_1}, \partial_{x_2})^\top$ 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 some linear space V (see [9]).

The variational problem to be solved reads:

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

Vanishing of the first variation yields the variational equation:

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

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

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

Under weak conditions with respect to the image data g, the existence of a constant $c > 0$ was proven in [9] such that $a(v,v) \geq c\|v\|_V^2 \ , \ \forall v \in V$. 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) by means of Green's formula, we derive the system of Euler-Lagrange equations:

$$Lu = f \quad \text{in} \quad \Omega \ , \qquad \partial_n u = 0 \quad \text{on} \quad \partial\Omega \ , \quad \text{where} \tag{5}$$

$$Lu = -\lambda \Delta u + (\nabla g \cdot u)\nabla g, \tag{6}$$

and n denoting the exterior unit normal.

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. To minimize notation, we denote the column vectors of nodal variables corresponding to the finite element discretizations of $u_1(x), u_2(x)$ again with u_1, u_2 and arrange them as follows: $u = (u_1^\top, u_2^\top)^\top$. Taking into consideration the symmetry of the bilinear form (3), this induces the following block structure of the discretized version of (2):

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

where for all pixel positions $i, j = 1, \ldots, N$ and corresponding basis functions ϕ_i, ϕ_j:

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

We point out that in connection with the decomposition into subproblems (section 3) and parallel implementations, a proper discretization and treatment of boundary conditions is essential to obtain convergence and numerical stability.

3 Problem Decomposition

This section summarizes the representation and parallel solution of the variational approach (1) using a partition of the image section Ω into a number of subdomains. For a detailed exposition we refer to [12]. Our formulation supports the application to more general variational approaches and inherits the flexibility of finite element discretizations with respect to unstructured geometries and triangulations.

Two Subdomains and Interface Equation. Let $\overline{\Omega^1} \cup \overline{\Omega^2}$ be a partition of $\overline{\Omega}$ with a common boundary $\Gamma = \overline{\Omega^1} \cap \overline{\Omega^2}$. We denote the corresponding spaces of vector fields with V^1, V^2. In the following, superscripts refer to subdomains.

We wish to represent u from (5) by two vector fields $u^1 \in V^1, u^2 \in V^2$ by solving two related problems in Ω^1, Ω^2, respectively. The relation:

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

obviously holds iff the following is true (cf. (5)):

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

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

$$u^1 = u^2 \quad \text{on } \Gamma \qquad \partial_{n^1} u^1 = -\partial_{n^2} u^2 \quad \text{on } \Gamma \tag{11}$$

In order to solve this system of equations, we equate the restriction to the interface Γ of the two solutions u^1, u^2 to (9) and (10) according to the first equation in (11), $u_\Gamma := u^1|_\Gamma = u^2|_\Gamma$, and substitute u_Γ into the second equation in (11). This is accomplished by means of the decomposition $u^i = u_0^i + u_f^i,\ i = 1, 2$, where u_0^i and u_f^i are the unique solutions of the equations:

$$Lu_0^i = 0 \quad \text{in } \Omega^i\ , \quad \partial_{n^i} u_0^i = 0 \quad \text{on } \partial\Omega^i \setminus \Gamma\ , \quad u_0^i = u_\Gamma \quad \text{on } \Gamma\ ,\ i = 1, 2 \tag{12}$$

$$Lu_f^i = f^i \quad \text{in } \Omega^i\ , \ \partial_{n^i} u_f^i = 0 \quad \text{on } \partial\Omega^i \setminus \Gamma\ , \quad u_f^i = 0 \quad \text{on } \Gamma\ , \quad i = 1, 2 \tag{13}$$

Defining the *Steklov-Poincaré operators* [11]: $S^i : u_\Gamma \to \partial_{n^i} u_0^i|_\Gamma$, the second equation in (11) becomes:

$$(S^1 + S^2) u_\Gamma + \partial_{n^1} u_f^1|_\Gamma + \partial_{n^2} u_f^2|_\Gamma = 0 \tag{14}$$

It remains to solve this equation for u_Γ. This will be discussed in the remainder of this section. Once this is done, u in (8) can be computed using (9) and (10) with the boundary conditions replaced by the first equation in (11).

Steklov-Poincaré operator. In order to make explicit the action of the operators S^i on some given boundary data u_Γ, we distinguish in (7) between nodal variables being on and off the interface by indices Γ and I, respectively. Rearranging (7) accordingly for domain $\Omega_i, i = 1, 2$ reads:

$$A^i u^i = f^i \quad \to \quad \begin{pmatrix} A_{II}^i & A_{I\Gamma}^i \\ A_{\Gamma I}^i & A_{\Gamma\Gamma}^i \end{pmatrix} \begin{pmatrix} u_I^i \\ u_\Gamma^i \end{pmatrix} = \begin{pmatrix} f_I^i \\ f_\Gamma^i \end{pmatrix}$$

Using this representation, the action of S^i in (14) is given by [12]:

$$S^i u_\Gamma = (A_{\Gamma\Gamma}^i - A_{\Gamma I}^i (A_{II}^i)^{-1} A_{I\Gamma}^i) u_\Gamma = \partial_{n^i} u_0^i|_\Gamma \tag{15}$$

The implementation of this equation involves – besides matrix-vector multiplications – the computation of $(A_{II}^i)^{-1}$ which amounts to solve a Dirichlet problem for the operator L in (6) with the right hand side given by the extension of boundary values u_Γ to Ω_i.

The action of $(S^i)^{-1}$, on the other hand, is given by [12]:

$$(S^i)^{-1}\partial_{n^i}u_0^i|_\Gamma = \begin{pmatrix}0 & I\end{pmatrix}\begin{pmatrix}A^i_{II} & A^i_{I\Gamma}\\ A^i_{\Gamma I} & A^i_{\Gamma\Gamma}\end{pmatrix}^{-1}\begin{pmatrix}0\\ I\end{pmatrix}\partial_{n^i}u_0^i|_\Gamma = u_0^i|_\Gamma = u_\Gamma \tag{16}$$

The implementation of this equation involves (i) to solve a Neumann problem for the operator L in (6) with boundary data $\partial_{n^i}u_0^i|_\Gamma$, and (ii) to extract the values of the solution on the interface Γ afterwards.

Finally, it can be shown [12] that the discretized counterpart of equation (14) reads:

$$(S^1 + S^2)u_\Gamma = f_\Gamma - A^1_{\Gamma I}(A^1_{II})^{-1}f^1_I - A^2_{\Gamma I}(A^2_{II})^{-1}f^2_I \tag{17}$$

It is important to note that in order to solve equation (14) or (17), respectively, neither S^i nor $(S^i)^{-1}$ are explicitely computed (which would be expensive). Rather, the action of these operators involves the boundary value problems as explained above which can be separately solved for each domain Ω^i by fast standard methods (e.g., multigrid iteration, see [8]).

Interface preconditioner. Since the operators S^i can shown to be symmetric and coercive [11], preconditioned conjugate gradient iteration is the first choice for solving (17). Among various provably optimal ("spectrally equivalent") possibilities, we have chosen the so-called Neumann-Neumann preconditioner $1/4[(S^1)^{-1} + (S^2)^{-1}]$ [13] because it preserves symmetry and has natural extensions to multiple domains, three-dimensional problems, and problems involving unstructured geometries and/or triangulations.

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, the operator on the left side of eqn. (17) for multiple domains reads:

$$\left(\sum_i (R^i)^\top S^i R^i\right) u_\Gamma = f_\Gamma - \sum_i (R^i)^\top A^i_{\Gamma I}(A^i_{II})^{-1}f^i_I \tag{18}$$

The corresponding Neumann-Neumann preconditioner[14,15] is:

$$P^{-1}_{NN} := D\left(\sum_i (R^i)^\top (S^i)^{-1} R^i\right) D\ , \tag{19}$$

where D^{-1}_{jj} of the diagonal scaling matrix D is the number of subdomains sharing the nodal variable u_j on Γ. Since it is well known that the convergence becomes worser for an increasing number of subdomains, also the Balancing Neumann-Neumann[16,17] preconditioner is considered:

$$P^{-1}_{BNN} := (I - (R^0)^\top (S^0)^{-1} R^0 S)P^{-1}_{NN}(I - S(R^0)^\top (S^0)^{-1}R^0) + (R^0)^\top (S^0)^{-1}R^0, \tag{20}$$

introducing a so-called "balancing" step before and after the Neumann-Neumann preconditioning by solving coarse-grid discretization of the Steklov-Poincaré equation, whereas the coarse-grid is identical with the partition into subdomains. The action of the restriction operator R^0 is to calculate the weighted sum of values lying on the shared border of each subdomain. The weights are given by the reciprocal of the number of subdomains sharing each particular node.

4 Parallel Processing, Computational Results, and Discussion

We conducted a number of experiments on regular $n \times n$ partitions in order to investigate (i) the effect of interface preconditioning vs. non-preconditioning, and (ii) the influence of the number of subdomains on the convergence rate both for the non-balancing and the balancing preconditioner.

All results refer to the solution of the interface equation (18) using Conjugate-Gradient Iteration with the preconditioners (19) or (20). (19) involves in each iteration step the solving of local boundary value problems *in parallel* for each subdomain, as explained in connection with equations (15) and (16). (20) additionally introduces the non-parallel calculation of $(S^0)^{-1}$, which is implemented by solving the (small) system $S^0x = b$ explicitly, using a standard method (see [12] for details). The associated systems to S^i and $(S^i)^{-1}$ were derived from the Finite Element discretization explained in connection with equation (7). The implementation was realized with C/C++ using a MPI-conform inter-process communication library for the parallel parts of the overall algorithm. The regularization parameter λ in (1) was set to 10 in all experiments leading to an amount of smoothing which is adequate for most real applications. As input data an image pair of size 252×252 pixel was used inducing an artifical optical flow field. The overall process started with the zero vector field $(0,0)^\top$.

Effect of interface preconditioning.

The image plane was partitioned horizontally into two subdomains of size 126×252 and (17) was solved to a relative residual error of 10^{-3} both without and with preconditioning using (19). In the first case 27 PCG-iterations were necessary to reach the given error threshold, whereas using the Neumann-Neumann preconditioner it was only one iteration. A similar experiment was conducted on 4×4 subdomains this time using (20) as preconditioner. Here we obtained similar results: 44 PCG-iterations without and 6 iterations with preconditioning. These and further experiments clearly show that, in agreement with theory [11], the system to be solved for the interface variables u_Γ becomes more and more ill-conditioned if the number of subdomains increases. Using the preconditioners (19) and (20), however, largely compensates this effect and enables shorter computation times through parallelization.

Number of subdomains, effect of parallelization.

In a second experiment the dependence of the convergence rate on the number of subdomains both for non-balancing and the balancing preconditioner were investigated. In addition, considerations about the overall computation time were made.

In table 1 the necessary number of PCG-iterations to reach a residual error of 10^{-3} are depicted for both preconditioner types. It shows that with the

Table 1. Number of subdomains, effect of parallelization. Columns 3 and 6 depict the number of outer iterations to reach a relative residual error of 10^{-3}, both for both preconditioner types. Columns 4 and 7 depict the average number of local iterations for an given error threshold of 10^{-5}. Columns 5 and 8 show the total number of sequentially processed pixels as a measure for the total computation time (see text).

Partition (h./v.)	Sub-domain size	Neumann-Neumann			Balancing Neumann-Neumann		
		Outer iter.	Av. inner iter.	$N_s * 10^6$	Outer iter.	Av. inner iter.	$N_s * 10^6$
2×2	126	3	370	35	3	332	63
4×4	63	7	238	13	6	197	19
6×6	42	10	179	6.3	7	141	6.9
9×9	28	14	127	2.8	6	92	1.7
14×14	18	21	86	1.2	6	70	0.5
21×21	12	30	57	0.5	5	42	0.1

non-balancing case the convergence rate depends on the number of subdomains whereas with the balancing case it is nearly independent, due to the coarse-grid correction steps. Furthmore, the average number of inner PCG-iterations for solving the local systems to a residual error of 10^{-5} are shown. Since the computation time for one inner PCG-iteration depends linearly on the number of pixels in one subdomain, the product N_s of the number of pixels in one subdomain, the aver. number of inner iterations, the number of local-system-solvings per one outer iteration (2 with (19) and 4 with (20)) and the total number of outer iteration gives a measure for the overall computation time, if communication time and the time for solving the coarse system in (20) is neglected.

It shows that the non-balancing type needs less time for small number of subdomains (until 6×6), whereas the balancing preconditioner is faster for 9×9 subdomains and above. In comparision to solving the original system (5) non-parallelly in 404 iterations on the whole image plane (252×252), i.e. $N_s \approx 26 \cdot 10^6$, parallelization significantly improves the total computation time for 4×4 subdomains and more (if communication time is neglected). Especially for the case of 21×21 subdomains it can be improved by two orders of magnitude. This factor, of course, becomes even larger for more subdomains, larger image sizes, and 3d image sequences.

5 Conclusion and Further Work

We presented an approach to parallel optical flow computation by decomposing the underlying domain Ω and iteratively solving related local variational problems in each subdomain. Inter-process communication is minimized by restricting the exchange of data to a lower-dimensional interface.
By combining our approach with advanced multigrid iteration as subdomain solvers, real-time 2d image processing will come into reach. This will be confirmed by implementations on dedicated high-speed PC-clusters. Furthermore, we will investigate how coarse-grid corrections can be incorporated into our architecture without compromising efficiency of inter-process communication.

Acknowledgement. TK and AB gratefully acknowledge support by the Deutsche Forschungsgemeinschaft DFG (grant SCHN457/4)

References

1. B.K.P. Horn and B.G. Schunck. Determining optical flow. *Art. Int.*, 17:185–203, 1981.
2. H.H. Nagel. On the estimation of optical flow: Relations between different approaches and some new results. *Artif. Intell.*, 33:299–324, 1987.
3. 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.
4. D. J. Fleet J. L. Barron and S. S. Beauchemin. Perfomance of optical flow techniques. *Int. J. Computer Vision*, 1994.
5. A. Bruhn, J. Weickert, and C. Schnörr. Combining the advantages of local and global optic flow methods. In L. van Gool, editor, *Pattern Recognition, Proc. 24th DAGM Symposium*, volume 2449 of *LNCS*, pages 454–462, Zürich, Switzerland, 2002. Springer.
6. D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Comp. Vis., Graph., and Imag. Proc.*, 24:52–96, 1983.
7. W. Enkelmann. Investigation of multigrid algorithms for the estimation of optical flow fields in image sequences. *Comp. Vis. Graphics and Imag. Proc.*, 43:150–177, 1987.
8. A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. In *Proc. Computer Analysis of Images and Patterns (CAIP'03)*, LNCS. Springer, 2003. in press.
9. C. Schnörr. Determining optical flow for irregular domains by minimizing quadratic functionals of a certain class. *Int. J. Computer Vision*, 6(1):25–38, 1991.
10. T.F. Chan and T.P. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.
11. A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford Univ. Press, 1999.
12. T. Kohlberger, C. Schnörr, A. Bruhn, and J. Weickert. Domain decomposition for variational optical flow computation. Technical Report 07/2003, Dept. Math. and Comp. Science, University of Mannheim, Germany, May 2003. Submitted to *IEEE Trans. Image Processing*.
13. J.-F. Bourgat, Glowinski R., P. Le Tallec, and Vidrascu M. Variational formulation and algorithm for trace operator in domain decomposition calculations. In T. Chan, Glowinski R., J. Périaux, and O. Widlund, editors, *Domain Decomposition Methods*, pages 3–16, Philadelphia, 1989. SIAM.
14. P. Le Tallec, De Roeck Y.-H., and Vidrascu M. Domain decomposition methods for large linearly elliptic three dimensional problems. *J. Comp. Appl. Math.*, page 34, 1991.
15. Y.-H. De Roeck and P. Le Tallec. Analysis and test of a local domain decomposition preconditioner. In R. Glowinsiki, Y. Kuznetsov, G. Meurant, J Périaux, and Widlund O., editors, *4th Int. Symp. on Domain Decomposition Methods for Part. Diff. Equations*, pages 112–128, Philadelphia, 1991. SIAM.
16. J. Mandel and M. Brezina. Balancing domain decomposition: Theory and performance in two and three dimensions. Technical report, Comp. Math. Group, University of Colorado at Denver, Denver, CO, 1, 1993.
17. J. Mandel. Balancing domain decomposition. *Comm. Numer. Math. Eng.*, 9:233–241, 1993.