A Highly Efficient GPU Implementation for Variational Optic Flow Based on the Euler-Lagrange Framework

Pascal Gwosdek[1], Henning Zimmer[1], Sven Grewenig[1],
Andrés Bruhn[2], and Joachim Weickert[1]

[1] Mathematical Image Analysis Group,
Faculty of Mathematics and Computer Science, Bldg. E1.1,
Saarland University, 66041 Saarbrücken, Germany
{gwosdek,zimmer,grewenig,weickert}@mia.uni-saarland.de
[2] Vision and Image Processing Group,
Cluster of Excellence Multimodal Computing and Interaction,
Saarland University, Campus E 1.1, 66123 Saarbrücken, Germany
bruhn@mmci.uni-saarland.de

Abstract. The Euler-Lagrange (EL) framework is the most widely-used strategy for solving variational optic flow methods. We present the first approach that solves the EL equations of state-of-the-art methods on sequences with 640×480 pixels in near-realtime on GPUs. This performance is achieved by combining two ideas: *(i)* We extend the recently proposed Fast Explicit Diffusion (FED) scheme to optic flow, and additionally embed it into a coarse-to-fine strategy. *(ii)* We parallelise our complete algorithm on a GPU, where a careful optimisation of global memory operations and an efficient use of on-chip memory guarantee a good performance. Applying our approach to the variational 'Complementary Optic Flow' method (Zimmer *et al.* (2009)), we obtain highly accurate flow fields in less than a second. This currently constitutes the fastest method in the top 10 of the widely used Middlebury benchmark.

1 Introduction

A fundamental task in computer vision is the estimation of the optic flow, which describes the apparent motion of brightness patterns between two frames of an image sequence. As witnessed by the Middlebury benchmark [1] [1], the accuracy of optic flow methods has increased tremendously over the last years. This trend was enabled by the recent developments in energy-based methods (e.g. [2,3,5,6,7,8,9,10,11]) that find the flow field by minimising an energy, usually consisting of a data and a smoothness term. While the data term models constancy assumptions on image features like the brightness, the smoothness term (regulariser) penalises fluctuations in the flow field.

To achieve state-of-the-art results, a careful design of the energy is mandatory. In the *data term*, robust subquadratic penaliser functions reduce the influence of outliers [5,7,11,10], higher-order constancy assumptions [7,10] help to deal with illumination changes, and a normalisation [4,10] prevents an overweighting at large image gradients. In the *smoothness term*, subquadratic penalisers yield a discontinuity-preserving

[1] Available at http://vision.middlebury.edu/flow/eval/

K.N. Kutulakos (Ed.): ECCV 2010 Workshops, Part II, LNCS 6554, pp. 372–383, 2012.

isotropic smoothing behaviour [5,7,11]. Anisotropic strategies [3,6,8,9,10] additionally allow to steer the smoothing direction, which in [10] yields an optimal complementarity between data and smoothness term.

A major problem of recent sophisticated methods is that their energies are highly nonconvex and nonlinear, rendering the minimisation a challenging task. Modern multigrid methods are well-known for their good performance on CPUs [12,13], but still do not achieve even near-realtime performance on larger image sequences. Multigrid methods on GPUs do achieve realtime performance, but due to their complicated implementation, they were only realised for basic models so far [14].

Another class of efficient algorithms that can easily be parallelised for GPUs and additionally support modern models are primal-dual approaches; see e.g. [11,9]. These methods typically introduce an auxiliary variable to decouple the minimisation w.r.t. the data and smoothness term. For the data term, one ends up with a thresholding that can be efficiently implemented on the GPU. For the smoothness term, a projected gradient descent algorithm similar to [15] is used. Problems of primal-dual approaches are *(i)* the rather limited number of data terms that can be efficiently implemented and *(ii)* the required adaptation of the gradient descent algorithm to the smoothness term. The latter is especially challenging for anisotropic regularisers, see [9].

The most popular minimisation strategy for continuous energy-based (variational) approaches is the Euler-Lagrange (EL) framework, e.g. [2,3,7,10,16]. Following the calculus of variations, one derives a system of coupled partial differential equations that constitute a necessary condition for a minimiser. The benefits of this framework are: *(i) Flexibility:* The EL equations can be derived in a straightforward manner for a large variety of different models. Even non-differentiable penaliser functions like the TV penaliser [17] can be handled by introducing a small regularisation parameter. *(ii) Generality:* The EL equations are of diffusion-reaction type. This does not only allow to use the same solution strategy for different models, but also permits to adapt solvers known from the solution of diffusion problems. However, one persistent issue of the EL framework is an efficient solution. As mentioned above, multigrid strategies are either restricted to basic models [14] or do not give realtime performance for modern test sequences [12].

Our Contribution. In the present paper, we present the first method that achieves near-realtime performance on a GPU for solving the EL equations. To this end, we adapt the recent *Fast Explicit Diffusion* (FED) scheme [18] to the EL framework. FED is an explicit solver with varying time step sizes, where some time steps can significantly exceed the stability limit of classical explicit schemes. If a series of time step sizes is carefully chosen, the approach can be shown to be unconditionally stable. The already high performance is further boosted by a coarse-to-fine strategy. Finally, our whole approach is parallelised on a GPU using the NVidia CUDA architecture [19]. By doing so, we introduce FED for massively parallel computing, where it unifies algorithmic simplicity with state-of-the-art performance. To obtain high performance despite the large amounts of data involved in the computation, we pay particular attention to an efficient use of on-chip memory to reduce transfers from and to global memory.

To prove the merits of our approach, we apply it within the recent variational optic flow method of Zimmer *et al.* [10], which gives qualitatively good results. Moreover,

due to its anisotropic regulariser, it can easily be specialised to less complicated smoothness terms. Experiments with our GPU-based algorithm show speedups by more than one order of magnitude over CPU implementations of both a multigrid solver and an FED scheme. Compared to the anisotropic primal-dual method of Werlberger *et al.* [9], we obtain better results in an equivalent runtime. In the Middlebury benchmark, we rank among the top 10 methods, and can report the smallest runtime among them.

Paper Organisation. In Sec. 2 we review the optic flow model of Zimmer *et al.* [10]. We then adapt the FED framework in Sec. 3, and present details on the GPU implementation in Sec. 4. Experiments demonstrating the efficiency and accuracy of our method are shown in Sec. 5, followed by a summary in Sec. 6.

2 Variational Optic Flow

Let $\boldsymbol{f}(\boldsymbol{x}) = (f^1(\boldsymbol{x}), f^2(\boldsymbol{x}), f^3(\boldsymbol{x}))^\top$ denote an image sequence where f^i represents the i-th RGB colour channel, $\boldsymbol{x} := (x, y, t)^\top$, with $(x, y)^\top \in \Omega$ describing the location within a rectangular image domain $\Omega \subset \mathbb{R}^2$ and $t \geq 0$ denotes time. We further assume that $\boldsymbol{f}$ has been presmoothed by a Gaussian convolution of standard deviation σ. The sought optic flow field $\boldsymbol{w} := (u, v, 1)^\top$ that describes the displacements from time t to $t+1$ is then found by minimising a global energy functional of the general form

$$E(u, v) = \int_\Omega \left[M(u, v) + \alpha\, V(\boldsymbol{\nabla} u, \boldsymbol{\nabla} v) \right] \, dx\, dy \;\; , \tag{1}$$

where $\boldsymbol{\nabla} := (\partial_x, \partial_y)^\top$ denotes the spatial gradient operator, and $\alpha > 0$ is a smoothness weight.

2.1 Complementary Optic Flow

The model we will use to exemplify our approach is the recent method of Zimmer *et al.* [10], because it gives favourable results at the Middlebury benchmark and uses a general anisotropic smoothness term.

Data Term. For simplicity, we use a standard RGB colour representation instead of the HSV model from the original paper. Our data term is given by

$$\begin{aligned} M(u, v) := \Psi_M \left(\sum_{i=1}^{3} \; \theta_0^i \left(f^i(\boldsymbol{x}+\boldsymbol{w}) - f^i(\boldsymbol{x}) \right)^2 \right) \\ + \gamma\, \Psi_M \left(\sum_{i=1}^{3} \left(\theta_x^i \left(f_x^i(\boldsymbol{x}+\boldsymbol{w}) - f_x^i(\boldsymbol{x}) \right)^2 + \theta_y^i \left(f_y^i(\boldsymbol{x}+\boldsymbol{w}) - f_y^i(\boldsymbol{x}) \right)^2 \right) \right) \;\; , \end{aligned} \tag{2}$$

where subscripts denote partial derivatives. The first line in (2) models the brightness constancy assumption [2], stating that image intensities remain constant under the displacement, i.e. $\boldsymbol{f}(\boldsymbol{x} + \boldsymbol{w}) = \boldsymbol{f}(\boldsymbol{x})$. To prevent an overweighting of the data term at large image gradients, a normalisation in the spirit of [4] is performed. To this end, one uses a normalisation factor $\theta_0^i := (|\boldsymbol{\nabla} f^i|^2 + \zeta^2)^{-1}$, where the small parameter $\zeta > 0$

avoids division by zero. Finally, to reduce the influence of outliers caused by noise or occlusions, a robust subquadratic penaliser function $\Psi_M(s^2) := \sqrt{s^2+\varepsilon^2}$ with a small parameter $\varepsilon > 0$ is used [7].

Weighted by $\gamma > 0$, the second line in (2) models the gradient constancy assumption $\boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{x}+\boldsymbol{w}) = \boldsymbol{\nabla} \boldsymbol{f}(\boldsymbol{x})$ that renders the approach robust under additive illumination changes [7]. The corresponding normalisation factors are defined as $\theta^i_{\{x,y\}} := (|\boldsymbol{\nabla} f^i_{\{x,y\}}|^2 + \zeta^2)^{-1}$. As proposed in [12] a separate penalisation of the brightness and the gradient constancy assumption is performed, which is advantageous if one assumption produces an outlier.

Smoothness Term. The data term only constraints the flow vectors in one direction, the *data constraint direction*. In the orthogonal direction, the data term gives no information (aperture problem). Thus, it makes sense to use a smoothness term that works *complementary* to the data term: In data constraint direction, a reduced smoothing should be performed to avoid interference with the data term, whereas a strong smoothing is desirable in the orthogonal direction to obtain a filling-in of missing information.

To realise this strategy, one needs to determine the data constraint direction. This can be achieved by considering the largest eigenvector of the regularisation tensor

$$R_\rho := \sum_{i=1}^{3} K_\rho * \left[\theta^i_0 \boldsymbol{\nabla} f^i \left(\boldsymbol{\nabla} f^i\right)^\top + \gamma \left(\theta^i_x \boldsymbol{\nabla} f^i_x \left(\boldsymbol{\nabla} f^i_x\right)^\top + \theta^i_y \boldsymbol{\nabla} f^i_y \left(\boldsymbol{\nabla} f^i_y\right)^\top\right)\right] \quad , \tag{3}$$

where K_ρ is a Gaussian of standard deviation ρ, and $*$ denotes the convolution operator. Apart from this convolution, the regularisation tensor is a spatial version of the motion tensor that occurs in a linearised data term. For more details, see [10].

Let $\boldsymbol{r}_1 \geq \boldsymbol{r}_2$ denote the two orthonormal eigenvectors of R_ρ, i.e. $\boldsymbol{r}_1$ is the data constraint direction. Then, the complementary regulariser is given by

$$V(\boldsymbol{\nabla} u, \boldsymbol{\nabla} v) = \Psi_V\left(\left(\boldsymbol{r}_1^\top \boldsymbol{\nabla} u\right)^2 + \left(\boldsymbol{r}_1^\top \boldsymbol{\nabla} v\right)^2\right) + \left(\boldsymbol{r}_2^\top \boldsymbol{\nabla} u\right)^2 + \left(\boldsymbol{r}_2^\top \boldsymbol{\nabla} v\right)^2 \quad . \tag{4}$$

To reduce the smoothing in data constraint direction, we use the subquadratic Perona-Malik penaliser (Lorentzian) [5,20] given by $\Psi_V(s^2) := \lambda^2 \ln(1 + (s^2/\lambda^2))$ with a contrast parameter $\lambda > 0$. In the orthogonal direction, a strong quadratic penalisation allows to fill in missing information.

2.2 Energy Minimisation via the Euler-Lagrange Framework

According to the calculus of variations, a minimiser (u, v) of the proposed energy (1) necessarily has to fulfil the associated Euler-Lagrange equations

$$\partial_u M - \alpha \operatorname{div}\left(D\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{\nabla} u, \boldsymbol{\nabla} v\right) \boldsymbol{\nabla} u\right) = 0 \quad , \tag{5}$$

$$\partial_v M - \alpha \operatorname{div}\left(D\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{\nabla} u, \boldsymbol{\nabla} v\right) \boldsymbol{\nabla} v\right) = 0 \quad , \tag{6}$$

with reflecting boundary conditions. These equations are of diffusion-reaction type, where the reaction part ($\partial_u M$ and $\partial_v M$) stems from the data term, and the diffusion part (written in divergence form) stems from the smoothness term.

To write down the reaction part of the EL equations, we use the abbreviations $f^i_{**} := \partial_{**} f^i(\boldsymbol{x}+\boldsymbol{w})$, $f^i_z := f^i(\boldsymbol{x}+\boldsymbol{w}) - f^i(\boldsymbol{x})$ and $f^i_{*z} := \partial_* f^i(\boldsymbol{x}+\boldsymbol{w}) - \partial_* f^i(\boldsymbol{x})$, where $** \in \{x, y, xx, xy, yy\}$ and $* \in \{x, y\}$. With their help, we obtain

$$\begin{aligned}\partial_u M = \quad & \Psi'_M \left(\sum_{i=1}^{3} \theta_0^i \left(f_z^i\right)^2 \right) \cdot \left(\sum_{i=1}^{3} \theta_0^i f_z^i f_x^i \right) \\ & + \gamma\, \Psi'_M \left(\sum_{i=1}^{3} \left(\theta_x^i \left(f_{xz}^i\right)^2 + \theta_y^i \left(f_{yz}^i\right)^2 \right) \right) \cdot \left(\sum_{i=1}^{3} \left(\theta_x^i f_{xz}^i f_{xx}^i + \theta_y^i f_{yz}^i f_{xy}^i \right) \right) \ , \end{aligned} \tag{7}$$

$$\begin{aligned}\partial_v M = \quad & \Psi'_M \left(\sum_{i=1}^{3} \theta_0^i \left(f_z^i\right)^2 \right) \cdot \left(\sum_{i=1}^{3} \theta_0^i f_z^i f_y^i \right) \\ & + \gamma\, \Psi'_M \left(\sum_{i=1}^{3} \left(\theta_x^i \left(f_{xz}^i\right)^2 + \theta_y^i \left(f_{yz}^i\right)^2 \right) \right) \cdot \left(\sum_{i=1}^{3} \left(\theta_x^i f_{xz}^i f_{xy}^i + \theta_y^i f_{yz}^i f_{yy}^i \right) \right) \ . \end{aligned} \tag{8}$$

The joint diffusion tensor $D\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{\nabla} u, \boldsymbol{\nabla} v\right)$ is given by

$$D\left(\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{\nabla} u, \boldsymbol{\nabla} v\right) := \Psi'_V \left(\left(\boldsymbol{r}_1^\top \boldsymbol{\nabla} u\right)^2 + \left(\boldsymbol{r}_1^\top \boldsymbol{\nabla} v\right)^2 \right) \boldsymbol{r}_1 \boldsymbol{r}_1^\top + \boldsymbol{r}_2 \boldsymbol{r}_2^\top \ . \tag{9}$$

Analysing the diffusion tensor, one realises that the resulting smoothing process is not only complementary to the data term, but can also be characterised as joint image- and flow driven: The smoothing direction is adapted to the direction of *image* structures, encoded in $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$. The smoothing strength depends on the *flow* contrast given by the expression $(\boldsymbol{r}_1^\top \boldsymbol{\nabla} u)^2 + (\boldsymbol{r}_1^\top \boldsymbol{\nabla} v)^2$. As a result, one obtains the same sharp flow edges as image-driven methods, but does not suffer from their oversegmentation problems.

Solution of the Euler-Lagrange Equations. The preceding EL equations are difficult to solve because the unknown $\boldsymbol{w}$ implicitly appears in the argument of the expressions $f^i(\boldsymbol{x}+\boldsymbol{w})$. A common strategy to resolve this problem is to embed the solution into a coarse-to-fine multiscale warping approach [7]. To obtain a coarse representation of the problem, the images are downsampled by a factor of $\eta \in [0.5, 1)$. At each warping level k, the flow field is split up into $\boldsymbol{w}^k + d\boldsymbol{w}^k =: \boldsymbol{w}^{k+1}$, where $\boldsymbol{w}^k = (u^k, v^k, 1)^\top$ is the already computed solution from coarser levels and $d\boldsymbol{w}^k = (du^k, dv^k, 0)^\top$ is a small flow increment that is computed by a linearised approach.

Let us derive this linearised approach. To ease presentation, we omit the gradient constancy part, i.e. set $\gamma = 0$, and restrict ourselves to the first EL equation (5). The extension to the full model works straightforward in accordance to [7]. A first step is to perform a Taylor linearisation

$$f_z^{i,k+1} \ := \ f^i(\boldsymbol{x}+\boldsymbol{w}^{k+1}) - f^i(\boldsymbol{x}) \ \approx \ f_z^{i,k} + f_x^{i,k} du^k + f_y^{i,k} dv^k \ , \tag{10}$$

where in expressions of the form $f^{i,k}$ the flow $\boldsymbol{w}^k$ is used. Replacing all occurrences of f_z^i by this linearisation and using the information from level k for all other constituents, one obtains the linearised first EL equation (with $\gamma = 0$)

$$\begin{aligned} & \Psi'_M \left(\sum_{i=1}^{3} \theta_0^{i,k} \left(f_z^{i,k} + f_x^{i,k} du^k + f_y^{i,k} dv^k\right)^2 \right) \cdot \sum_{i=1}^{3} \theta_0^{i,k} \left(f_z^{i,k} + f_x^{i,k} du^k + f_y^{i,k} dv^k\right) f_x^{i,k} \\ & \quad - \alpha \ \mathrm{div} \left(D\left(\boldsymbol{r}_1^k, \boldsymbol{r}_2^k, \boldsymbol{\nabla}\left(u^k + du^k\right), \boldsymbol{\nabla}\left(v^k + dv^k\right)\right) \ \boldsymbol{\nabla}\left(u^k + du^k\right) \right) = 0 \ . \end{aligned} \tag{11}$$

At this point, it is feasible to use a solver for nonlinear systems of equations. However, we use a second coarse-to-fine strategy per warping level for an even faster convergence. Here the prolongated solution from a coarse level serves as initialisation for the next finer level.

3 Fast Explicit Diffusion Solver

A classical approach to solve elliptic problems such as the linearised EL equation (11) are semi-implicit schemes: They are unconstrained in their time step sizes, but require to solve large linear systems of equations in each step. In contrast, explicit schemes are much easier to implement and have a low complexity per step, but are typically restricted to very small step sizes to guarantee stability. In this paper, we use a new time discretisation that combines the advantages of both worlds [18]: *Fast Explicit Diffusion (FED)* schemes are as simple as classical explicit frameworks, but use some extremely large time steps to ensure a fast convergence. Still, the combination of large (unstable) and small (stable) time steps within one *cycle* guarantees the unconditional stability of the complete approach. Hence, FED schemes outperform semi-implicit schemes in terms of efficiency and are additionally much simpler to implement, especially on massively parallel architectures.

Let us first derive a *stabilised* explicit scheme [16] for solving the linearised EL equation (11) w.r.t. the unknown du^k. To this end, we introduce the iteration variable l:

$$\begin{aligned}\frac{du^{k,l+1}-du^{k,l}}{\tau_l} = \;& \operatorname{div}\left(D\left(\boldsymbol{r}_1^k, \boldsymbol{r}_2^k, \boldsymbol{\nabla}\left(u^k+du^{k,l}\right), \boldsymbol{\nabla}\left(v^k+dv^{k,l}\right)\right) \boldsymbol{\nabla}\left(u^k+du^{k,l}\right)\right) \\ & -\frac{1}{\alpha}\left(\psi'^{\,k,l}_M(..)\cdot\sum_{i=1}^{3}\theta_0^{i,k}\left(f_z^{i,k}+f_x^{i,k}du^{k,l+1}+f_y^{i,k}dv^{k,l}\right) f_x^{i,k}\right), \quad (12)\end{aligned}$$

where τ_l denotes the FED time step size at iteration $0 \leqslant l < n$ which is computed as [18]

$$\tau_l = \tfrac{1}{8}\cdot\left(\cos^2\left(\pi\,\tfrac{2l+1}{4n+2}\right)\right)^{-1}. \quad (13)$$

In (12), the term $\psi'^{\,k,l}_M(..)$ is an abbreviation for the expression $\Psi'_M(..)$ in the first line of (11), where we additionally replace du^k by $du^{k,l}$ and dv^k by $dv^{k,l}$. Finally, note that our scheme is stabilised by using $du^{k,l+1}$ from the next iteration in the last row.

In our next step we discretise the expression $\operatorname{div}(D(..)\boldsymbol{\nabla}(u^k+du^{k,l}))$ in matrix-vector notation by $A(u^k+du^{k,l}, v^k+dv^{k,l})(u^k+du^{k,l}) =: A^{k+1,l}u^{k+1,l}$. This enables us to rewrite (12) as

$$\begin{aligned}du^{k,l+1} = & \left[du^{k,l}+\tau_l A^{k+1,l}u^{k+1,l}-\frac{\tau_l}{\alpha}\left(\psi'^{\,k,l}_M(..)\cdot\sum_{i=1}^{3}\theta_0^{i,k}\left(f_z^{i,k}+f_y^{i,k}dv^{k,l}\right) f_x^{i,k}\right)\right] \\ & \cdot\left(1+\frac{\tau_l}{\alpha}\cdot\psi'^{\,k,l}_M(..)\cdot\sum_{i=1}^{3}\theta_0^{i,k}\left(f_x^{i,k}\right)^2\right)^{-1}. \quad (14)\end{aligned}$$

Remarks. The number of individual time steps n in a cycle is given by $\min\{n \in \mathbb{N}^+ \mid (n^2+n)/12 \geqslant T\}$, where T denotes the desired stopping time of the cycle. For $n \geqslant 3$, one can show that an FED cycle reaches this stopping time T faster than any other explicit scheme with n stable time step sizes.

Moreover, the ordering of steps within one FED cycle is irrelevant from a theoretical point of view, but can in practice affect the influence of rounding errors to the result. However, it is possible to find permutations of the set $\{\tau_l \mid 0 \leqslant l < n\}$ that are more robust w.r.t. floating-point inaccuracies than others. Given the next larger prime number p to n and $\kappa < p$, a series $\{\tau_{\tilde{l}} \mid \tilde{l} = ((l+1) \cdot \kappa) \bmod p, \tilde{l} < n\}$ is known to give good results [18,21]. In order to find a suitable value for the parameter κ, we analysed a simple 1-D problem and choose the one κ that minimises the error between the FED output and the analytic reference solution. These values were once computed for all practical choices of n to set up a lookup table which is used throughout our implementation.

4 Implementation on the GPU

Since our algorithm is hierarchic and uses different data configurations and cache patterns for the operations it performs, we split it up into single GPU kernels of homogeneous structure. This concept allows to have a recursive program flow on the CPU, while the data is kept in GPU memory throughout the process.

FED Solver. Our stabilised fast explicit scheme forms the heart of our algorithm. It is also the most expensive GPU kernel in our framework: Due to its low arithmetic complexity, it is strictly memory bound and requires significant amounts of data. For the smoothness term, we reduce the memory complexity by exploiting the symmetry of the non-diagonal matrix A from (14), which comes down to store the four upper off-diagonals. The remaining entries can be computed in shared memory. Where offset data loads are necessary for this strategy, they can be efficiently realised by texture lookups.

Derivatives. Spatial image and flow derivatives are discretised via central finite differences with consistency order 2 and 4, respectively [12]. For the motion tensor, these derivatives are averaged from the two frames $f(x, y, t)$ and $f(x, y, t+1)$, whereas for the regularisation tensor, they are solely computed at the first frame. Where required, we compute both the first order and second order derivatives in the same GPU kernel which saves a large number of loads from global memory. Thanks to the texture cache, the slightly larger neighbourhood that is needed in this context does again not significantly affect the runtime.

Diffusion Tensor. In order to set up the diffusion tensor D for the smoothness term, we apply the diffusivity function to the eigenvalues of the structure tensor and use these new eigenvalues to assemble a new tensor. Both the derivative computation and the principal axis transforms that are used in this context are fully data parallel. Note that we do not store the tensor entries to global memory, but directly compute the weights that are later to be used in the solver. By this, we save again a significant number of global loads and stores. Due to our nonlinear model, we update the diffusion tensor after every cycle.

Gaussian Convolution. Our GPU-based Gaussian convolution algorithm is tailored to the small standard deviations σ that typically occur in the context of optic flow: We exploit the operation's separability and cut off the discretised kernel at a precision of 3σ. This allows our 'sliding window' approach to keep a full neighbourhood in shared memory, and thus to reduce global memory operations to one read and write per pixel. Along the main direction of the 2-D data in memory, we apply loop unrolling over data-independent rows and keep three consecutive sub-planes of the source image in a ring buffer. Across this direction, we cut our domain in sufficiently large chunks, and maintain a ring buffer of chunk-wide rows that cover the entire neighbourhood of the computed row.

Resampling. Key ingredients for hierarchic coarse-to-fine algorithms are prolongation and restriction operators. Several examples for such operators are known in the literature, but they are either quite expensive on GPUs due to their 'inhomogeneous' algorithmic structure, or do not possess necessary properties such as grey value preservation, aliasing artefact prevention, and flexibility with respect to the choice of the resampling factor [22,23]. As a remedy, we propose a fast but versatile technique that approximates the desired behaviour well enough to satisfy the quality requirements for optic flow. It has a uniform algorithmic structure for all target cells and uses the texturing mechanism of CUDA cards to obtain a high performance.

Textures can be queried at any point in a continuous domain, and in particular in between grid points. The resulting value is then computed in hardware by means of a bilinear interpolation. These properties alone yield an efficient prolongation algorithm: For any target cell of the result, we use the value at the corresponding point of the source texture. Note that this strategy does not guarantee grey value preservation from a theoretical point of view, but experimentally yields favourable results.

As it turns out, we must not apply the same algorithm for restriction purposes: Typical choices of restriction factors close to two cause undersampling and lead to aliasing artefacts. To overcome this problem, we use four sampling points instead of one: Let r_x, r_y be the restriction factors in x– and y–direction, respectively, and assume textures to be defined on the domain $[0, n_x - 1) \times [0, n_y - 1)$. For any target point $(x, y)^\top$, we then average over the texture values at locations

$$\left(\frac{1}{r_x} \cdot \left(x \pm \frac{1}{4} \right), \frac{1}{r_y} \cdot \left(y \pm \frac{1}{4} \right) \right)^\top \quad . \tag{15}$$

This modification allows us to choose arbitrary factors in the interval $[\frac{1}{2}, 1)$ which suffices for our purposes. Moreover, since nearby sampling points are likely to be in the 2-D texture cache at the same time, this strategy is almost as fast as prolongation.

Warping. In order to access images at warped positions, i.e. to evaluate expressions of type $f^i(\boldsymbol{x}+\boldsymbol{w}^k)$, we use the texturing mechanism of graphics cards: We store the image channel i that is to be warped in a texture, compute the target location by adding flow field and pixel coordinates, and fetch the texture at the respective point. Albeit incoherent memory access is often considered a major performance problem on massively parallel hardware, this operation turns out to be highly efficient: Optic flow is often piecewise laminar and sufficiently smooth, such that the missing data locality is largely compensated by the 2-D texture cache.

5 Experiments

Quality. We first consider a qualitative evaluation of our results. To this end, we chose 4 sequences with known ground truth from the Middlebury database, and computed the optic flow fields using our algorithm and an individual choice of parameters. A visualisation of the results is shown in Fig. 1. Like in the original CPU implementation of Zimmer *et al.* [10], the flow fields are accurate and without visual artefacts.

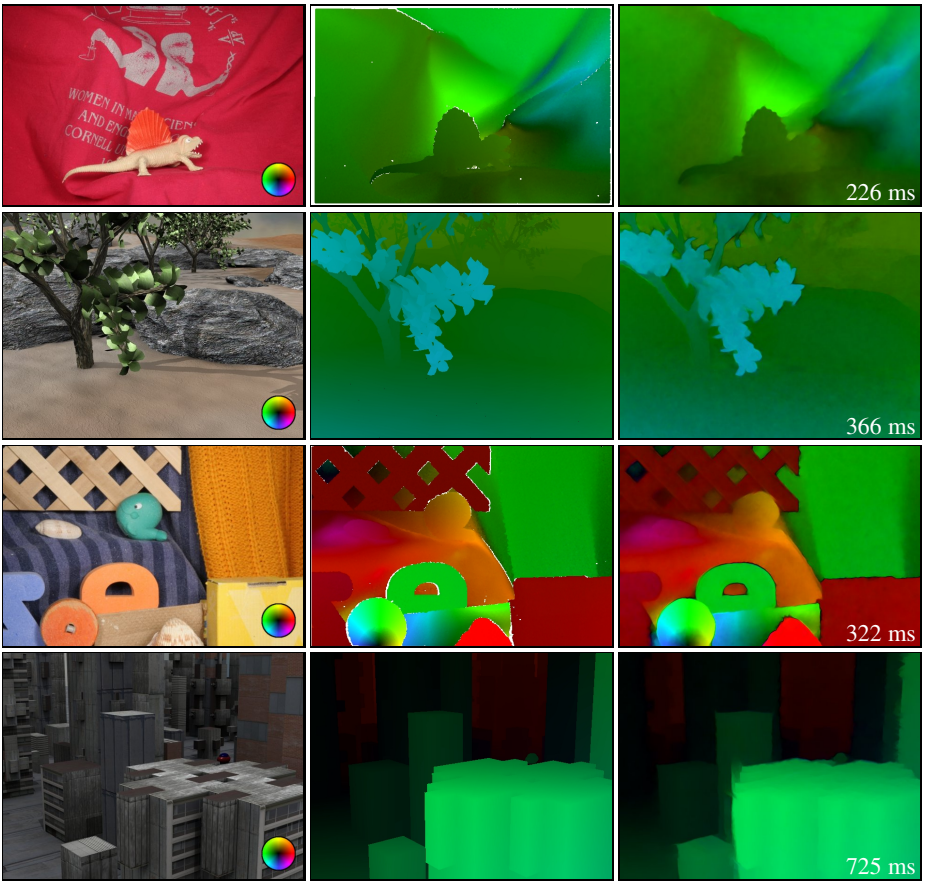

Fig. 1. Our results for 4 Middlebury sequences with ground truth. **Top to bottom:** *Dimetrodon, Grove2, RubberWhale, Urban2*. **Left to right:** First frame with flow key, ground truth, result with runtime. We use optimised parameter sets $(\alpha, \gamma, \zeta, \lambda)$ for the individual sequences (*D*: (400, 8, 1.0, 0.05), *G*: (50, 1, 1.0, 0.05), *R*: (1000, 20, 1.0, 0.05), *U*: (1500, 25, 0.01, 0.1)). Fixed parameters for all cases: $\eta = 0.91, \sigma = 0.3, \rho = 1.3$, 1 cascadic FED step with 1 nonlinear update and $T = 150$ per warp level.

Table 1. Error measures for 4 Middlebury sequences with known ground truth using the optimal parameter sets from Fig. 1, and a fixed parameter set (300, 20, 0.01, 0.1).

Sequence		Dimetrodon	Grove2	RubberWhale	Urban2
Optimised	**AEE**	0.08	0.16	0.09	0.29
	AAE	1.49	2.32	2.93	2.75
Fixed	**AEE**	0.11	0.19	0.11	0.36
	AAE	2.20	2.69	3.76	3.56

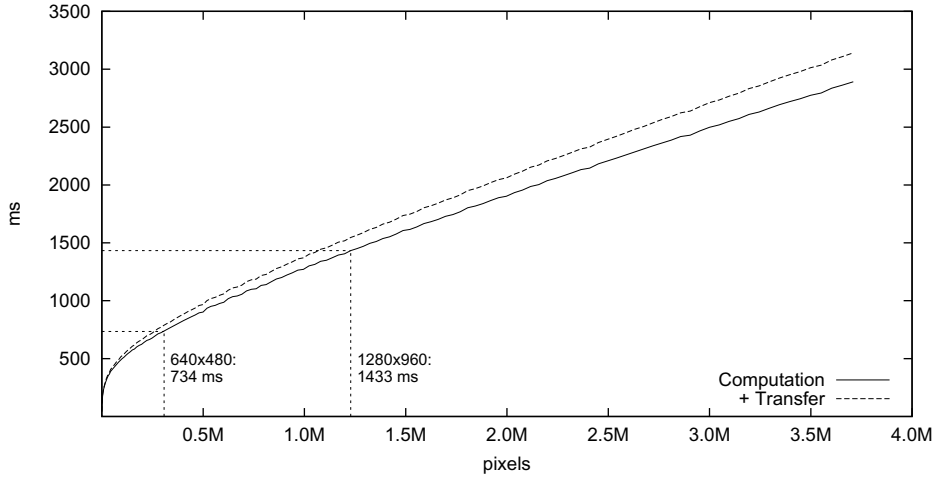

Fig. 2. Runtimes (with and without device transfer) on images with size ratio 4:3

We also evaluated our results to the ground truth by computing the Average Endpoint Error (AEE), as well as the Average Angular Error (AAE). In order to be better comparable to the results of other state-of-the-art methods, we additionally performed the same experiment on a fixed parameter set for all sequences, as it is required for the Middlebury benchmark. From Tab. 1, we see that if we use fixed parameters, we obtain results comparable to those of Werlberger *et al.* [9], which has been the top-ranked anisotropic GPU-based method in the Middlebury benchmark so far. Using individually tuned parameters as in Fig. 1, the obtained quality can be further enhanced.

The high quality of our algorithm is also reflected in the position in the Middlebury benchmark. In August 2010, it ranks seventh out of 35 both w.r.t. AAE and AEE.

Runtime. Finally, we evaluate the efficiency of our approach on image sequences of varying sizes. To this end, we benchmark the runtimes on an NVidia GeForce GTX 480 graphics card. Since runtimes are affected by the size ratio of the image sequence and the parameter set, we used a ratio of 4:3 and the fixed parameter set from Tab. 1. This is depicted in Fig. 2. On *Urban2* (640×480), our algorithm takes 734 ms. Compared to hand-optimised Multigrid (FAS) [12] and FED schemes with equivalent results on one core of an 2.33 GHz Intel Core2 Quad CPU, this performance results in speedups of 15

and 17, respectively. Thanks to a better GPU occupancy, these factors are even higher the larger the frame size, e.g. 23 and 28 for frames of 1024×768 pixels. Moreover, our algorithm has comparable runtimes to the approach of Werlberger *et al.* [9], despite yielding more accurate results, as seen in the Middlebury benchmark. Concerning the latter, our method currently is the fastest among the top 10 approaches, outperforming the competitors by one to three orders of magnitude.

6 Conclusions and Outlook

We have presented a highly efficient method for minimising variational optic flow approaches by solving the corresponding Euler-Lagrange (EL) equations. The core of our approach is the recently proposed Fast Explicit Diffusion (FED) scheme [18], which can be adapted to optic flow due to the diffusion-reaction character of the EL equations. Additionally, we apply a coarse-to-fine strategy, and parallelise our complete algorithm on a GPU, thereby introducing the first parallel FED implementation.

In our experiments, we used the proposed approach to minimise the optic flow model of Zimmer *et al.* [10], resulting in highly accurate flow fields that are computed in less than one second for sequences of size 640×480. This gives a speedup by more than one order of magnitude compared to a CPU implementation of *(i)* a multigrid solver and *(ii)* an FED solver. In the Middlebury benchmark, we rank among the top 10 and achieve the smallest runtime there.

Since most variational optic flow algorithms are based on solving the EL equations, we hope that our approach can also help to tangibly speedup other optic flow methods based on the EL framework. Note that we used an anisotropic regulariser, which results in the most general form of the diffusion part. Applying our approach with other popular smoothness terms, like TV regularisation, thus works straightforward by simply replacing the diffusion tensor by a scalar-valued diffusivity.

Our future research will be concerned with further reducing the runtimes to meet an ultimate goal: Realtime performance for state-of-the-art optic flow approaches on high resolution (maybe high-definition) image sequences.

Acknowledgements. We gratefully acknowledge partial funding by the cluster of excellence 'Multimodal Computing and Interaction', by the International Max Planck Research School, and by the Deutsche Forschungsgemeinschaft (project We2602/7-1).

References

1. Baker, S., Roth, S., Scharstein, D., Black, M.J., Lewis, J.P., Szeliski, R.: A database and evaluation methodology for optical flow. In: Proc. 2007 IEEE International Conference on Computer Vision. IEEE Computer Society Press, Rio de Janeiro (2007)
2. Horn, B., Schunck, B.: Determining optical flow. Artificial Intelligence 17, 185–203 (1981)
3. Nagel, H.H., Enkelmann, W.: An investigation of smoothness constraints for the estimation of displacement vector fields from image sequences. IEEE Transactions on Pattern Analysis and Machine Intelligence 8, 565–593 (1986)
4. Simoncelli, E.P., Adelson, E.H., Heeger, D.J.: Probability distributions of optical flow. In: Proc. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 310–315. IEEE Computer Society Press, Maui (1991)

5. Black, M.J., Anandan, P.: The robust estimation of multiple motions: parametric and piecewise smooth flow fields. Computer Vision and Image Understanding 63, 75–104 (1996)
6. Weickert, J., Schnörr, C.: A theoretical framework for convex regularizers in PDE-based computation of image motion. International Journal of Computer Vision 45, 245–264 (2001)
7. Brox, T., Bruhn, A., Papenberg, N., Weickert, J.: High Accuracy Optical Flow Estimation Based on a Theory for Warping. In: Pajdla, T., Matas, J. (eds.) ECCV 2004. LNCS, vol. 3024, pp. 25–36. Springer, Heidelberg (2004)
8. Sun, D., Roth, S., Lewis, J.P., Black, M.J.: Learning Optical Flow. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 83–97. Springer, Heidelberg (2008)
9. Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., Bischof, H.: Anisotropic Huber-L^1 optical flow. In: Proc. 20th British Machine Vision Conference. British Machine Vision Association, London (2009)
10. Zimmer, H., Bruhn, A., Weickert, J., Valgaerts, L., Salgado, A., Rosenhahn, B., Seidel, H.-P.: Complementary Optic Flow. In: Cremers, D., Boykov, Y., Blake, A., Schmidt, F.R. (eds.) EMMCVPR 2009. LNCS, vol. 5681, pp. 207–220. Springer, Heidelberg (2009)
11. Zach, C., Pock, T., Bischof, H.: A Duality Based Approach for Realtime TV- L^1 Optical Flow. In: Hamprecht, F.A., Schnörr, C., Jähne, B. (eds.) DAGM 2007. LNCS, vol. 4713, pp. 214–223. Springer, Heidelberg (2007)
12. Bruhn, A., Weickert, J.: Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In: Proc. of the Tenth International Conference on Computer Vision, vol. 1, pp. 749–755. IEEE Computer Society Press, Beijing (2005)
13. El Kalmoun, M., Köstler, H., Rüde, U.: 3D optical flow computation using a parallel variational multigrid scheme with application to cardiac C-arm CT motion. Image and Vision Computing 25, 1482–1494 (2007)
14. Grossauer, H., Thoman, P.: GPU-Based Multigrid: Real-Time Performance in High Resolution Nonlinear Image Processing. In: Gasteratos, A., Vincze, M., Tsotsos, J.K. (eds.) ICVS 2008. LNCS, vol. 5008, pp. 141–150. Springer, Heidelberg (2008)
15. Chambolle, A.: An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision 20, 89–97 (2004)
16. Weickert, J., Schnörr, C.: Variational optic flow computation with a spatio-temporal smoothness constraint. Journal of Mathematical Imaging and Vision 14, 245–255 (2001)
17. Rudin, L.I., Osher, S., Fatemi, E.: Nonlinear total variation based noise removal algorithms. Physica D 60, 259–268 (1992)
18. Grewenig, S., Weickert, J., Bruhn, A.: From Box Filtering to Fast Explicit Diffusion. In: Goesele, M., Roth, S., Kuijper, A., Schiele, B., Schindler, K. (eds.) DAGM 2010. LNCS, vol. 6376, pp. 533–542. Springer, Heidelberg (2010)
19. NVIDIA Corporation: NVIDIA CUDA Programming Guide. 3rd edn.(2010), `http://developer.download.nvidia.com/compute/cuda/3_0/toolkit/docs/NVIDIA_CUDA_ProgrammingGuide.pdf` (retrieved June 10, 2009)
20. Perona, P., Malik, J.: Scale space and edge detection using anisotropic diffusion. IEEE Transactions on Pattern Analysis and Machine Intelligence 12, 629–639 (1990)
21. Gentzsch, W., Schlüter, A.: Über ein Einschrittverfahren mit zyklischer Schrittweitenänderung zur Lösung parabolischer Differentialgleichungen. Zeitschrift für angewandte Mathematik und Mechanik 58, T415–T416 (1978)
22. Trottenberg, U., Oosterlee, C., Schüller, A.: Multigrid. Academic Press, San Diego (2001)
23. Bruhn, A., Weickert, J., Feddern, C., Kohlberger, T., Schnörr, C.: Variational optical flow computation in real-time. IEEE Transactions on Image Processing 14, 608–615 (2005)