

CTIM Technical Report

ISSN 2254-2353

Parallel Implementation of a Robust Optical Flow Technique

*Javier Sánchez Pérez, Nelson Monzón López
and Agustín J. Salgado de la Nuez*

No. 1

Las Palmas de Gran Canaria
16 March 2012

Parallel Implementation of a Robust Optical Flow Technique

Javier Sánchez, Nelson Monzón and Agustín Salgado

Centro de Tecnologías de la Imagen
Universidad de Las Palmas de Gran Canaria
35017 Las Palmas de Gran Canaria, Spain

Abstract

The accuracy and performance of current variational optical flow methods have considerably increased during the last years. The complexity of these techniques is high and enough care has to be taken for the implementation. The aim of this work is to present a comprehensible implementation of recent variational optical flow methods. We start with an energy model that relies on brightness and gradient constancy terms and a flow-based smoothness term. We minimize this energy model and derive an efficient implicit numerical scheme. In the experimental results, we evaluate the accuracy and performance of this implementation with the Middlebury benchmark database. We show that it is a competitive solution with respect to current methods in the literature. In order to increase the performance, we use a simple strategy to parallelize the execution on multi-core processors.

Keywords: Optical flow, Variational methods, PDE, Multi-core, OpenMP

1 Introduction

The estimation of motion fields in image sequences is an important challenge in computer vision. It has received much attention during the last two decades as it is a fundamental application. It is the base for many other high level applications, such as stereoscopic vision, medical image analysis, ambient intelligence, fluid flow analysis, meteorological prediction and many others.

We focus on variational optical flow methods, which have demonstrated to be among the most accurate methods in the literature. One of the firsts to propose such a variational approach were Horn and Schunck [6], whose method is the basis for many recent methods. They opened a research line that is still continued and improved. There has been many important contributions to this basic model, such as the use of robustification functions [3], the preserving of image discontinuities by means of anisotropic diffusion tensors [7] or the implementation of multiresolution schemes to cope with large displacements [1].

The purpose of this paper is twofold: on the one hand, our objective is to develop a comprehensible implementation of up-to-date variational techniques to compute the optical flow; on the other hand, we also explain how to parallelize the algorithms in order to take advantage of current multi-core CPUs. We have implemented a method similar to the work presented in [4] and later extended in [8]. These are based on energy

functionals that include several smoothness and attachment constraints – the brightness and gradient constancy assumptions and a flow driven smoothness regularization. These methods replace the traditional quadratic penalty function by a continuous L^1 norm, which makes them more robust to outliers and illumination changes. A study on robustification functions has been realized in [3] and is currently widespread in most of optical flow methods. Our implementation is different to the work presented in [8] in that we derive a gradient descent approach that ends up in a diffusion-reaction Partial Differential Equation (PDE). We do not include the temporal derivative of the flow in the smoothness term, because it is not justified in the case of large displacements.

The implementation of this kind of variational methods is traditionally slow, since they rely on numerical schemes that need a large number of iterations to converge. Some works have overcome this problem and are actually achieving real time performance. Examples are the work by [5] that implements an efficient multigrid method and the work by [9] that accelerates the execution by means of an implementation on the GPU. In our case, we use the OpenMP library for the parallelization of the algorithms. This library is thought to easily parallelize low level pieces of code. Our code is developed in standard C++ and it has been compiled under Windows and Linux, using the GNU gcc compiler. We have placed our code in a web page with a very general open source license.

In the experimental results, we study the accuracy and performance of our implementation. We show that our implementation is competitive with other similar methods and the parallel implementation considerably improves the velocity. Although the above mentioned multigrid and GPU methods may outperform our implementation, its simplicity makes it worth using the OpenMP library.

In Section 2 we explain the details of the optic flow method. In Section 4 we derive an implicit numerical scheme and propose the algorithm for its implementation. The multi-core extension of the method is developed in Section 5. Some experimental results are presented in Section 6. We study the behavior of the method utilizing image sequences from the Middlebury benchmark database and draw some conclusions about the performance on multi-core processors.

2 Energy model

Let $\mathbf{h}(\mathbf{x}) = (u(\mathbf{x}), v(\mathbf{x}))^T : \mathbb{R}^2 \longrightarrow \mathbb{R}^2$, be a vectorial function representing the optical flow field in the continuous domain Ω . For every position $\mathbf{x} = (x, y)^T$, the optical flow depicts a directional vector of the apparent displacement at a given position. The optical flow is an ill-posed problem in the sense that there may exist an infinity of solutions for the matching of two images. A traditional approach to overcome this problem is to impose regularity constraints that ensure the continuity of the flow field. In our variational formulation, a general energy model contains a set of constancy and smoothness assumptions that enables us to compute the optical flow by applying optimization techniques. Our energy model can be written as follows:

$$\begin{aligned}
E(\mathbf{h}) = & \int_{\Omega} \Phi \left((I_1(\mathbf{x}) - I_2(\mathbf{x} + \mathbf{h}(\mathbf{x})))^2 \right) + \gamma \Phi \left(\|\nabla I_1(\mathbf{x}) - \nabla I_2(\mathbf{x} + \mathbf{h}(\mathbf{x}))\|^2 \right) d\mathbf{x} \\
& + \alpha \int_{\Omega} \Phi \left(\|\nabla u(\mathbf{x})\|^2 + \|\nabla v(\mathbf{x})\|^2 \right) d\mathbf{x},
\end{aligned} \tag{1}$$

where $\Phi(s^2) = \sqrt{s^2 + \epsilon}$, with $\epsilon = 0.0001$. The Φ function turns the method robust against outliers.

The first term, which corresponds to the brightness constancy assumption, attracts pixels with the same intensity in both images. It fails in the presence of noise or changes in illumination. The second term, corresponding to the gradient constancy assumption, compares the structure of the objects in both images: it is invariant to constant changes of illumination, but is sensitive to the presence of noise and non-translational displacements.

Finally, the smoothness term is in charge of creating a continuous and dense solution. This energy model is similar to the models presented in [4] and [8]. The main difference is that we do not use the temporal derivative of the flow in the smoothness term. The reason to remove the temporal derivative is to avoid incongruencies with the data terms.

3 Energy minimization

The solution to the above energy model (1) can be obtained from its associated Euler-Lagrange equations:

$$\begin{aligned}
\mathcal{F}_u(\mathbf{h}) &= \Phi'_A \left(I_1 - I_2^{\mathbf{h}} \right) I_{2x}^{\mathbf{h}} + \gamma \Phi'_B \left(\left(I_{1x} - I_{2x}^{\mathbf{h}} \right) I_{2xx}^{\mathbf{h}} + \left(I_{1y} - I_{2y}^{\mathbf{h}} \right) I_{2yx}^{\mathbf{h}} \right) \\
&\quad + \alpha \operatorname{div}(\Phi'_C \nabla u) \\
\mathcal{F}_v(\mathbf{h}) &= \Phi'_A \left(I_1 - I_2^{\mathbf{h}} \right) I_{2y}^{\mathbf{h}} + \gamma \Phi'_B \left(\left(I_{1x} - I_{2x}^{\mathbf{h}} \right) I_{2yx}^{\mathbf{h}} + \left(I_{1y} - I_{2y}^{\mathbf{h}} \right) I_{2yy}^{\mathbf{h}} \right) \\
&\quad + \alpha \operatorname{div}(\Phi'_C \nabla v)
\end{aligned} \tag{2}$$

with

$$\begin{aligned}
\Phi'_A &= \Phi' \left((I_1(\mathbf{x}) - I_2(\mathbf{x} + \mathbf{h}))^2 \right) \\
\Phi'_B &= \Phi' \left(\|\nabla I_1(\mathbf{x}) - \nabla I_2(\mathbf{x} + \mathbf{h})\|^2 \right) \\
\Phi'_C &= \Phi' \left(\|\nabla u\|^2 + \|\nabla v\|^2 \right).
\end{aligned} \tag{3}$$

The derivative of the robust function is $\Phi'(s^2) = \frac{1}{2\sqrt{s^2 + \epsilon}}$. Note that we have used a different notation for the warping of the target image and its derivatives, $I_2^{\mathbf{h}} = I_2(\mathbf{x} + \mathbf{h})$. Other alternatives to minimize this kind of energy models are reviewed in [2]. The minimum is achieved in the roots of the system of equations (2).

The above system of equations (2) is solved by means of a gradient descent technique, which results in the following scheme: $\frac{\partial u}{\partial t} = \mathcal{F}_u(\mathbf{h})$, $\frac{\partial v}{\partial t} = \mathcal{F}_v(\mathbf{h})$. These equations are not linear due to the term $I_2(\mathbf{x} + \mathbf{h}(\mathbf{x}))$ and its derivatives. To overcome these nonlinearities, we introduce first order Taylor expansions: $I_2^{\mathbf{h}^{k+1}} = I_2^{\mathbf{h}^k} + I_{2x}^{\mathbf{h}^k} (u^{k+1} - u^k) + I_{2y}^{\mathbf{h}^k} (v^{k+1} - v^k) + \mathcal{O}(h^2)$.

In order to find an initial estimate for this linearization, we embed the equations in a focusing strategy, where the system is solved at coarser levels and later updated in finer scales. This kind of coarse-to-fine approaches [1] allows for computing large displacements. We create a pyramid of images with a reduction factor of $\eta \in (0, 1)$. For the coarsest scale we compute the solution (u_n, v_n) . Then the solution is used as an initial approximation for the following scale, $u_i(\eta\mathbf{x}) = \frac{1}{\eta}u_{i-1}(\mathbf{x})$.

4 Numerical scheme

In our implementation, we choose an implicit approach for the numerical scheme. This has the advantage of being unconditionally stable for any time step. The divergence operator is approximated with finite differences as:

$$\begin{aligned} \text{div}(\Phi'_C \nabla u) \simeq & \frac{\Phi'_{C,i+1,j} + \Phi'_{C,i,j}}{2} (u_{i+1,j}^{k+1} - u_{i,j}^{k+1}) + \frac{\Phi'_{C,i-1,j} + \Phi'_{C,i,j}}{2} (u_{i-1,j}^{k+1} - u_{i,j}^{k+1}) \\ & + \frac{\Phi'_{C,i,j+1} + \Phi'_{C,i,j}}{2} (u_{i,j+1}^{k+1} - u_{i,j}^{k+1}) + \frac{\Phi'_{C,i,j-1} + \Phi'_{C,i,j}}{2} (u_{i,j-1}^{k+1} - u_{i,j}^{k+1}). \end{aligned} \quad (4)$$

Then, our numerical scheme results from substituting the divergence in our gradient descent approach, yielding $\frac{u_{i,j}^{k+1} - u_{i,j}^k}{dt} = \mathcal{F}_u(\mathbf{h}^k, \mathbf{h}^{k+1})$ and $\frac{v_{i,j}^{k+1} - v_{i,j}^k}{dt} = \mathcal{F}_v(\mathbf{h}^k, \mathbf{h}^{k+1})$. This creates a sparse system, where only the diagonal and the surrounding neighbors in the matrix are different from zero. One of the most efficient ways to solve this system is with iterative schemes such as Gauss-Seidel or SOR numerical approximations. Grouping the terms in instants k and $k+1$, we solve for the unknowns, $u_{i,j}^{k+1}$ and $v_{i,j}^{k+1}$, obtaining the following scheme:

$$\begin{aligned} u_{i,j}^{k+1} &:= \frac{u_{i,j}^k + dt \cdot Nu}{1 + dt \cdot Du} \\ v_{i,j}^{k+1} &:= \frac{v_{i,j}^k + dt \cdot Nv}{1 + dt \cdot Dv}. \end{aligned} \quad (5)$$

The numerators (Nu, Nv) correspond to the terms accompanying $(u_{i\pm 1, j\pm 1}^{k+1}, v_{i\pm 1, j\pm 1}^{k+1})$ and (u^k, v^k) in the neighborhood of pixel (i, j) . The denominators (Du, Dv) correspond to the terms accompanying $(u_{i,j}^{k+1}, v_{i,j}^{k+1})$.

In algorithm 1 we show the process of estimating the optical flow for a given scale. We observe the two nested loops for the inner and outer iterations and a final loop for the Gauss-Seidel implementation. Note that we have used two previous time instants (u^{k-1}, v^{k-1}) in these two nested fixed point iterations: the first one is used to warp the images, $I_2(\mathbf{x} + \mathbf{h}^{k-1})$, and the second for computing the robust function, i.e.

$\Phi' \left(\|\nabla u^k\|^2 + \|\nabla v^k\|^2 \right)$. We have implemented this algorithm and some supplementary functions in C/C++. This code is free with a general open source license. It is available at http://www.ctim.es/research_works/parallel_robust_optic_flow.

Algorithm 1: Robust optical flow

```

Input:
     $I_1, I_2, dt, \alpha, \gamma, TOL, inner\_iter, outer\_iter$ 
Process:
    compute  $I_{1x}, I_{1y}, I_{2x}, I_{2y}, I_{2xx}, I_{2xy}, I_{2yy}$ 
    for  $i \leftarrow 1$  to  $outer\_iter$  do
         $u^{k-1} \leftarrow u^{k+1}$ 
         $v^{k-1} \leftarrow v^{k+1}$ 
        compute  $I_2^h, I_{2,x}^h, I_{2,y}^h, I_{2,xx}^h, I_{2,yx}^h, I_{2,yy}^h, \Phi'_A, \Phi'_B$ 
        for  $j \leftarrow 1$  to  $inner\_iter$  do
             $u^k \leftarrow u^{k+1}$ 
             $v^k \leftarrow v^{k+1}$ 
            compute  $u_x, u_y, v_x, v_y, \Phi'_C$ 
            while  $error > TOL$  do
                 $u^p \leftarrow u^{k+1}$ 
                 $v^p \leftarrow v^{k+1}$ 
                 $u^{k+1} \leftarrow \frac{u^k + dt \cdot Nu}{1 + dt \cdot Du}$ 
                 $v^{k+1} \leftarrow \frac{v^k + dt \cdot Nv}{1 + dt \cdot Dv}$ 
                 $error \leftarrow \sum (u^{k+1} - u^p)^2 + (v^{k+1} - v^p)^2$ 
            end while
        end for
    end for

```

5 Use of multi-core infrastructure

One easy way to take advantage of the multi-core infrastructure of current CPUs is by means of the OpenMP library. It parallelizes the code with very simple preprocessor directives. It has the advantage that it is not necessary to change the original code. It automatically handles the number of threads that are thrown in every parallel region and synchronizes the data between them. In order to create a parallel region, the programmer has to insert the following preprocessor directive in C/C++:

```

#pragma omp parallel
{...
}

```

After this definition, a set of threads are automatically created. If we want to parallelize a loop, then we have to introduce the following sentences inside this region:

```
#pragma omp for schedule(dynamic) nowait
for(int i = 0; i < ny; i++)
```

We have included these kind of sentences in our loops, both in the main algorithm and in the supplementary functions. Versions of the GCC compiler above 4.2 include the OpenMP library by default.

6 Experimental results

In the experimental results, we examine the behavior of our implementation for some standard synthetic sequences. The first one is the Yosemite sequence, whose results can be seen in Fig. 2. The solution is very accurate as can be observed in table 1. A good compromise between speed and accuracy can be reached, as shown in Fig. 6. When we use 8 parallel cores and reduce the number of iterations of the numerical scheme, the euclidean error is 0,133 and the angular error $2,99^\circ$ with a running time of 10 seconds.

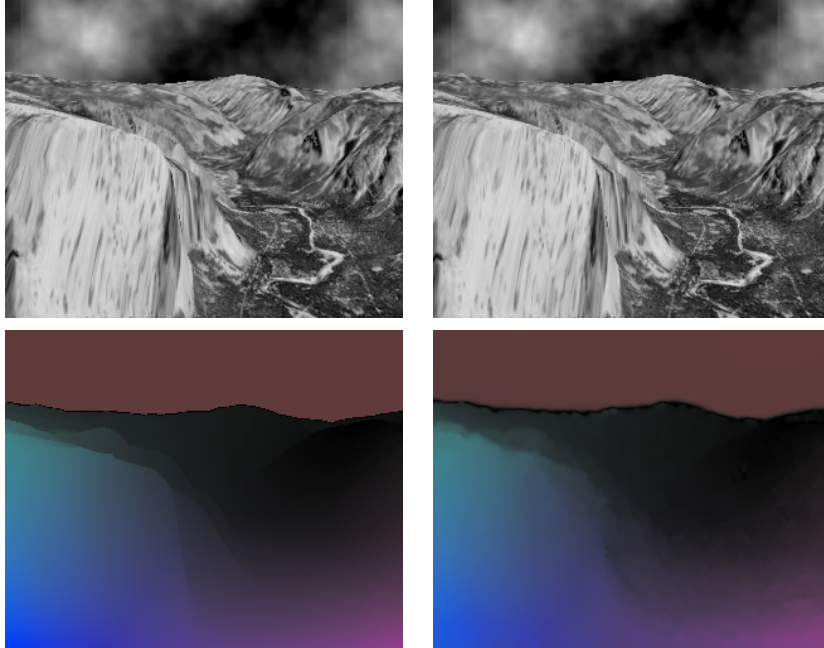


Fig. 2: Yosemite with clouds: first row depicts frames 6 and 7 of the sequence; second row shows the ground truth and the solution provided by our implementation, respectively.

In Fig. 2, frames 6 and 7 of the Yosemite sequence are shown on the top row. At the bottom-left, we show the ground truth optical flow and at bottom-right the solution for our method, using the color scheme of Fig. 3. In table 1 we show the best numerical results for the average end-point error (EPE) and average angular error (AAE).

Next, we show the results of our method for the sequences in the Middlebury benchmark database. We have used the method for two different purposes: on the one hand, we looked for the best possible results for each sequence; on the other hand, we have also sought a default configuration for the parameters that work appropriately for all sequences. You can see these results in Figs. 4 and 5.

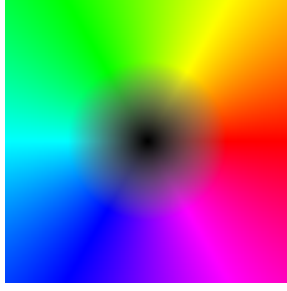


Fig. 3: Color scheme used to represent the orientation and magnitude of the flow field in every point.

Table 1: Average End-point Error (EPE) and Average Angular Error (AAE) for the Yosemite sequence.

α	γ	η	Inner Iter.	Outer Iter.	EPE	AAE
250	37	0.5	25	100	0.098	2.190°

Table 2 shows the numerical results with the best parameters used in Figs. 4 and 5 (fourth and fifth columns). The other parameters are set as follows: $\epsilon := 0.0001$, $dt := 5$ and the number of scales were chosen so that the size of the coarsest scale is small enough.

Table 2: EPE and AAE for the Middlebury datasets: configuration of the parameters that provide the best average errors.

Sequence	α	γ	η	Inner Iter.	Outer Iter.	EPE	AAE
Dimetrodon	83	48	0.8	30	120	0.087	1.686°
Grove2	83	16	0.8	25	80	0.163	2.374°
Grove3	46	16	0.5	29	70	0.700	6.541°
Hydrangea	104	16	0.8	25	120	0.168	2.102°
RubberWhale	104	38	0.8	30	120	0.111	3.751°
Urban2	51	16	0.8	30	120	0.340	2.667°
Urban3	21	16	0.8	25	100	0.492	4.234°
Venus	21	12	0.8	27	200	0.291	4.440°

In table 3, we show the numerical results with default parameters. These results correspond to the third column of Figs. 4 and 5.

Finally, we have also compared the results for three different sequences – Yosemite, RubberWhale and the Ettlingen Tor. A performance study on multi-core architecture is outlined in Fig. 6. In the first image we present the time gain when the method is executed on several cores. In the second image we show the effective time loss that occurs when there are several threads in parallel with respect to the single thread execution. This gives us an idea of the time loss due to the handling of the threads.


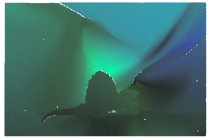
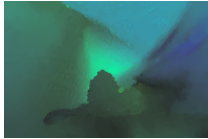
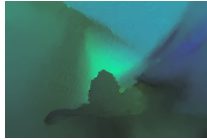
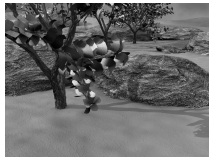
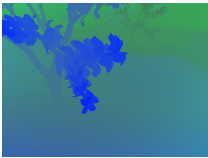
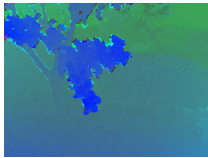
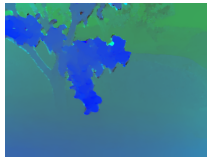
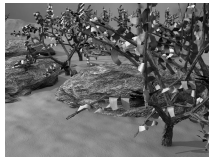

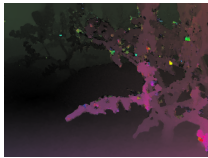
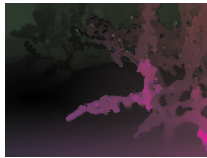

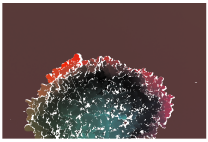
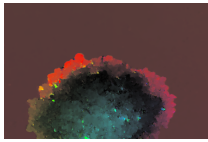
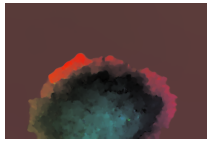
Frame 10	Ground truth	Default	Best	Parameters
				Dimetrodon $\alpha = 83$ $\gamma = 48$
				Grove2 $\alpha = 83$ $\gamma = 16$
				Grove3 $\alpha = 46$ $\gamma = 16$
				Hydrangea $\alpha = 104$ $\gamma = 16$

Fig. 4: Some results for the Middlebury database: the left column shows frame 10 of the sequence; the second column shows the ground truth; the third, the result with default parameters; the fourth, the best solution found; and the fifth, the name of the sequence and the parameters used for the best solution.

7 Conclusions

The estimation of accurate optical flow fields is a challenging task. Among the most renowned methods, current variational methods are attaining the highest degrees of accuracy with very good time performances. In this paper we have presented an implementation of an efficient optical flow method using up-to-date techniques. We have explained the necessary steps to derive the algorithm from the global energy model, going into the details of the minimization process and the numerical solution.

In particular, the variational model proposed is similar to the model presented in [8]. As a difference, we derive a gradient descent approach that ends up in a diffusion-reaction PDE. We have also introduced a simple and efficient multi-core parallel computing by means of the OpenMP library. The main advantage of this approach is that it makes it easier to manage threads and synchronization of data, without modifying the original code. As we have shown in the experimental results, our implementation yields a good accuracy and we have also observed a significant increase of performance. In future works we will continue to study different ways of improving the accuracy of the method as well



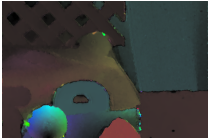
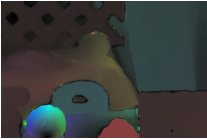
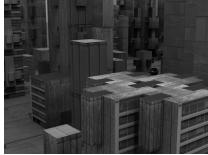
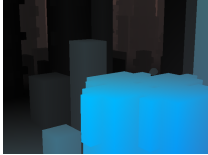
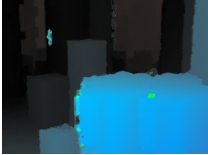

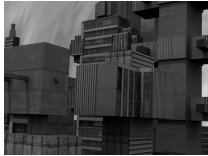




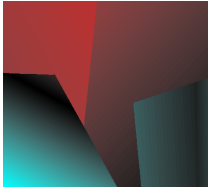
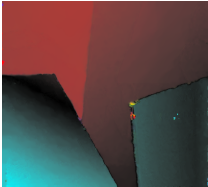
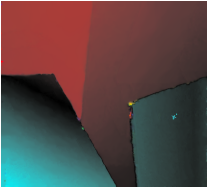
Frame 10	Ground truth	Default	Best	Parameters
				RubberWhale $\alpha = 104$ $\gamma = 38$
				Urban2 $\alpha = 51$ $\gamma = 16$
				Urban3 $\alpha = 21$ $\gamma = 16$
				Venus $\alpha = 21$ $\gamma = 12$

Fig. 5: More results for the Middlebury database: the left column shows frame 10 of the sequence; the second column shows the ground truth; the third, the result with default parameters; the fourth, the best solution found; and the fifth, the name of the sequence and the parameters used for the best solution.

as increasing its velocity.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Innovation through the research project TIN2011-25488.

References

- [1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310–310, January 1989.
- [2] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. In *International Conference on Computer Vision*, pages 1–8, 2007.

Table 3: EPE and AAE for the Middlebury datasets: results obtained with default parameters.

Sequence	α	γ	η	Inner Iter.	Outer Iter.	EPE	AAE
Dimetrodon	113	83	0.8	25	120	0.088	1.704 ^o
Grove2	113	83	0.8	25	120	0.227	3.027 ^o
Grove3	113	83	0.8	25	120	0.809	7.844 ^o
Hydrangea	113	83	0.8	25	120	0.239	2.915 ^o
RubberWhale	113	83	0.8	25	120	0.127	4.127 ^o
Urban2	113	83	0.8	25	120	0.408	3.239 ^o
Urban3	113	83	0.8	25	120	0.512	4.392 ^o
Venus	113	83	0.8	25	120	0.300	4.580 ^o

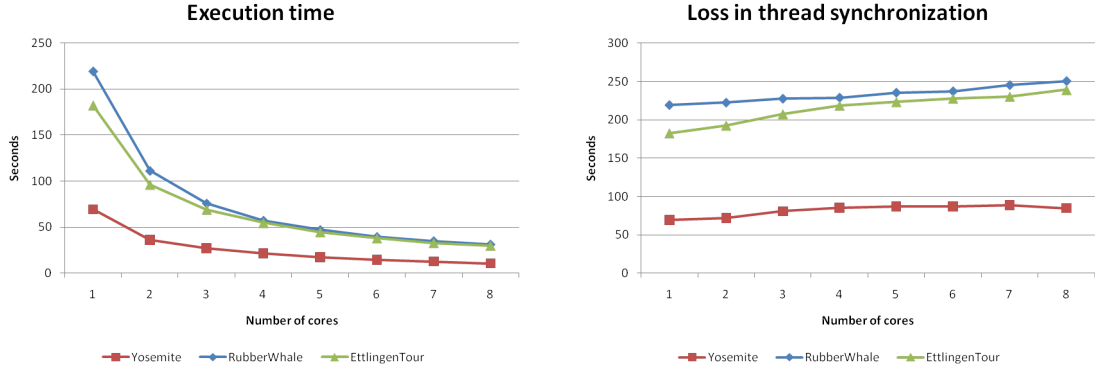


Fig. 6: Running times: The left graphic shows the time spent for computing the optical flow using increasing number of cores; the right image represents the time loss when using multiple cores.

- [3] Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75 – 104, 1996.
- [4] T. Brox, A. Bruhn, N. Papenberger, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision (ECCV)*, volume 3024 of *LNCS*, pages 25–36, Prague, Czech Republic, May 2004. Springer.
- [5] A. Bruhn, J. Weickert, T. Kohlberger, and C. Schnörr. A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. *International Journal of Computer Vision*, 70(3):257–277, 2006.
- [6] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.
- [7] H H Nagel and W Enkelmann. 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, September 1986.

- [8] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly Accurate Optic Flow Computation with Theoretically Justified Warping. *International Journal of Computer Vision*, 67(2):141–158, April 2006.
- [9] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L1 Optical Flow. In Fred A. Hamprecht, Christoph Schnörr, and Bernd Jähne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, chapter 22, pages 214–223. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.



Centro de Tecnologías de la Imagen
Universidad de Las Palmas de Gran Canaria
<http://www.ctim.es>