Published in Image Processing On Line on 2013–07–19.
Submitted on 2012–06–14, accepted on 2013–04–17.
ISSN 2105–1232 © 2013 IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials, software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.2013.20

Horn–Schunck Optical Flow with a Multi-Scale Strategy

Enric Meinhardt-Llopis[1], Javier Sánchez[2], Daniel Kondermann[3]

[1] CMLA, ENS Cachan, France (enric.meinhardt@cmla.ens-cachan.fr)
[2] CTIM, University of Las Palmas de Gran Canaria, Spain (jsanchez@dis.ulpgc.es)
[3] Heidelberg Collaboratory for Image Processing, Interdisciplinary Center for Scientific Computing, Heidelberg University (daniel.kondermann@iwr.uni-heidelberg.de)

Abstract

The seminal work of Horn and Schunck is the first variational method for optical flow estimation. It introduced a novel framework where the optical flow is computed as the solution of a minimization problem. From the assumption that pixel intensities do not change over time, the *optical flow constraint* equation is derived. This equation relates the optical flow with the derivatives of the image. There are infinitely many vector fields that satisfy the optical flow constraint, thus the problem is ill-posed. To overcome this problem, Horn and Schunck introduced an additional regularity condition that restricts the possible solutions. Their method minimizes both the optical flow constraint and the magnitude of the variations of the flow field, producing smooth vector fields. One of the limitations of this method is that, typically, it can only estimate small motions. In the presence of large displacements, this method fails when the gradient of the image is not smooth enough. In this work, we describe an implementation of the original Horn and Schunck method and also introduce a multi-scale strategy in order to deal with larger displacements. For this multi-scale strategy, we create a pyramidal structure of downsampled images and change the optical flow constraint equation with a nonlinear formulation. In order to tackle this nonlinear formula, we linearize it and solve the method iteratively in each scale. In this sense, there are two common approaches: one that computes the *motion increment* in the iterations; or the one we follow, that computes the full flow during the iterations. The solutions are incrementally refined over the scales. This pyramidal structure is a standard tool in many optical flow methods.

Source Code

A standalone ANSI C implementation is available[1]. This file contains two main programs: `horn_schunck_classic.c`, which implements the original Horn and Schunck method; and the implementation of the multi-scale approach, in file `horn_schunck_pyramidal.c`. This latter implementation is best suited for general image sequences.

Keywords: optical flow, multiscale, pyramidal

[1]Version 3.0, from the article web page http://dx.doi.org/10.5201/ipol.2013.20.

1 Introduction

Let $I(x, y, t)$ be a video sequence, and let $(x(t), y(t))$ be the trajectory of a material point projected to the image plane. The brightness constancy assumption states that the pixel intensities, $I(x(t), y(t), t)$ remain constant over time:

$$\frac{\mathrm{d}I}{\mathrm{d}t} = 0. \tag{1}$$

This identity must hold for the trajectories of every point in the image domain, whose velocities at one instant define a vector field

$$\mathbf{h} = (u, v) = \left(\frac{\mathrm{d}x}{\mathrm{d}t}, \frac{\mathrm{d}y}{\mathrm{d}t}\right). \tag{2}$$

Thus, the vector field $\mathbf{h}$ satisfies pointwise the following linear condition, which is derived from the brightness constancy assumption by applying the chain rule:

$$\nabla I \cdot \mathbf{h} + I_t = 0, \tag{3}$$

with $\nabla I = (I_x, I_y)$. This is the *optical flow constraint* equation. This equation cannot be solved pointwise, since the number of parameters to be estimated is larger than the number of linearly independent equations. This indeterminacy is called the *aperture problem*: there is not enough information to recover the optical flow at one point by only looking at first order derivatives of the image intensity. This is illustrated in figure 1.

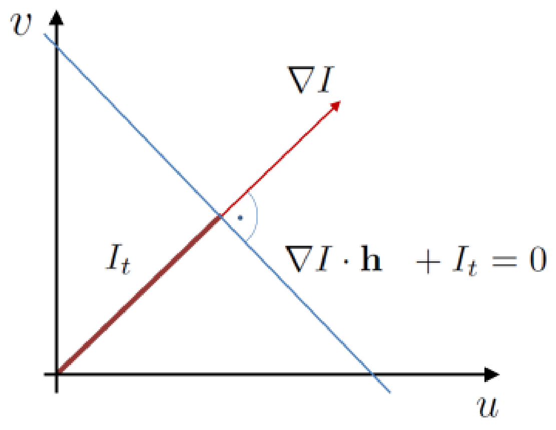


Figure 1: Optical flow constraint.

The optical flow constraint equation represents a line perpendicular to the image gradient. Thus, given this constraint alone, it is only possible to estimate the magnitude of the motion in the gradient direction as:

$$\mathbf{h} = -I_t \frac{\nabla I}{|\nabla I|^2}, \tag{4}$$

which is the smallest motion satisfying the optical flow constraint. This equation is also called *normal flow*.

2 The Classical Horn and Schunck Method

The proposal of Horn and Schunck consists in formulating the problem of optical flow estimation as a variational problem, where the desired vector field $\mathbf{h}$ is defined as the minimizer of a certain energy functional $J(\mathbf{h})$. This functional has two terms: a data attachment term, given by the optical flow constraint, and a regularity term that is based on the gradient of the flow:

$$J(\mathbf{h}) = \int_\Omega (I_x u + I_y v + I_t)^2 + \alpha^2 \left(|\nabla u|^2 + |\nabla v|^2\right), \tag{5}$$

where α is a parameter to control the weight of the smoothness term compared to the optical flow constraint. The parameter α is squared so that its units are units of grey-level, and it can be regarded as the intensity of an additive Gaussian noise present in the input images. This energy model uses quadratic functionals in both terms. This assumes that the image noise and the flow derivatives are expected to follow a Gaussian distribution. A direct consequence of this kind of functionals is that the method is very sensitive to the presence of noise and the computed flow fields are very smooth. These shortcomings have led to the appearance of many research works that try to deal with these limitations. The use of L^1 functionals [7, 20, 17], has turned out to be a better approach.

The minimization of the above functional yields the following Euler-Lagrange equations:

$$\begin{aligned} I_x^2 u + I_x I_y v &= \alpha^2 \mathrm{div}(\nabla u) - I_x I_t, \\ I_x I_y u + I_y^2 v &= \alpha^2 \mathrm{div}(\nabla v) - I_y I_t. \end{aligned} \tag{6}$$

The Laplacian can be approximated with the following expressions, which will be useful for the discretization below:

$$\begin{aligned} \mathrm{div}\,(\nabla u) &\approx (\bar{u} - u), \\ \mathrm{div}\,(\nabla v) &\approx (\bar{v} - v), \end{aligned} \tag{7}$$

where $(\bar{u}, \bar{v})$ are local averages of (u, v). This approximation is analogous to the commonly used "difference of gaussians", where the Laplacian operator is approximated by the difference of blurred versions of the image [13]; in this case the smallest blur is zero. Solving the equations above for (u, v) and re-arranging the terms, we obtain the following system of equations:

$$\begin{aligned} \left(\alpha^2 + I_x^2 + I_y^2\right)(u - \bar{u}) &= -I_x \left(I_x \bar{u} + I_y \bar{v} + I_t\right), \\ \left(\alpha^2 + I_x^2 + I_y^2\right)(v - \bar{v}) &= -I_y \left(I_x \bar{u} + I_y \bar{v} + I_t\right). \end{aligned} \tag{8}$$

Writing these equations for each pixel of the input images, we obtain a sparse system of linear equations (see figure 2). This system can be solved efficiently with an iterative scheme, as explained in the next section.

2.1 Numerical Scheme

The partial derivatives, I_x, I_y and I_t, are approximated using forward differences and averaging between two consecutive frames:

$$\begin{aligned} I_x &\approx \frac{1}{4}\left(I_{i,j+1,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i+1,j,k+1}\right), \\ I_y &\approx \frac{1}{4}\left(I_{i+1,j,k} - I_{i,j,k} + I_{i+1,j+1,k} - I_{i,j+1,k} + I_{i+1,j,k+1} - I_{i,j,k+1} + I_{i+1,j+1,k+1} - I_{i,j+1,k+1}\right), \\ I_t &\approx \frac{1}{4}\left(I_{i,j,k+1} - I_{i,j,k} + I_{i+1,j,k+1} - I_{i+1,j,k} + I_{i,j+1,k+1} - I_{i,j+1,k} + I_{i+1,j+1,k+1} - I_{i+1,j+1,k+1}\right), \end{aligned} \tag{9}$$

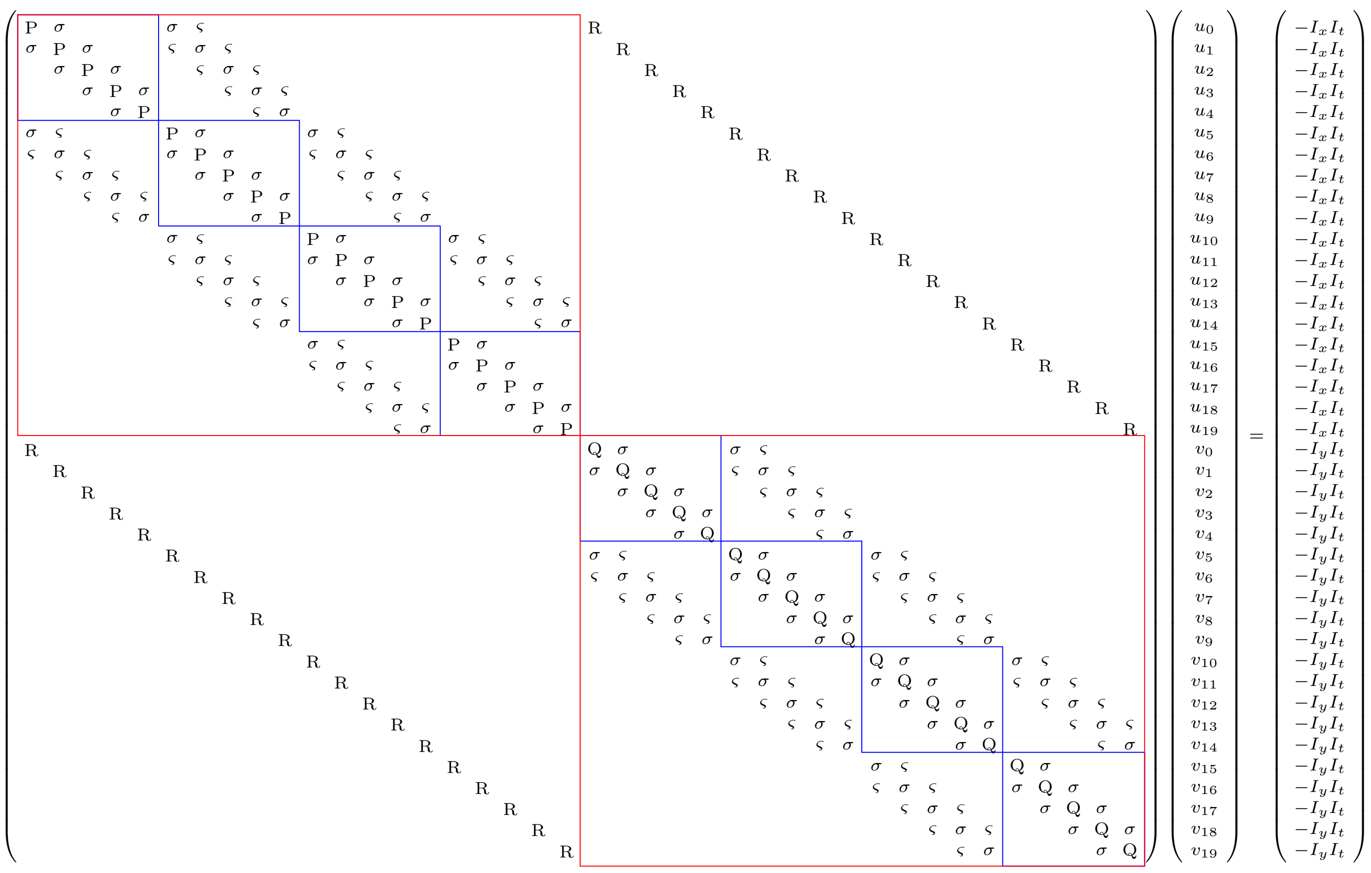


Figure 2: System of linear equations (8) for an input image of size 5×4. The following abridged notation is used to write the system in matrix form. The 40 unknowns u_i, v_i correspond to the values of the vector field (u, v), ordered lexicographically as in figure 3. All functions, such as $I_x I_t$ are evaluated at the pixel corresponding to their position. The symbol P denotes $I_x^2 + \alpha^2$. The symbol Q denotes $I_y^2 + \alpha^2$. The symbol R denotes $I_x I_y$. Finally, we use the constants $\sigma = -\alpha^2/6$ and $\varsigma = -\alpha^2/12$. This system is written assuming Dirichlet boundary conditions. For Neumann boundary conditions, the constants σ and ς corresponding to the boundaries of the image must be changed to zero.

0	1	2	3	4
5	6	7	8	9
10	11	12	13	14
15	16	17	18	19

Figure 3: Indices for the pixel positions used on figure 2.

with Neumann boundary conditions. The local averages $(\bar{u}, \bar{v})$ are estimated from the eight neighbors of (u, v) as:

$$\begin{aligned}\bar{u} \approx& \frac{1}{6}(u^n_{i-1,j} + u^n_{i+1,j} + u^n_{i,j-1} + u^n_{i,j+1}) + \frac{1}{12}(u^n_{i-1,j-1} + u^n_{i+1,j-1} + u^n_{i-1,j+1} + u^n_{i+1,j+1}),\\ \bar{v} \approx& \frac{1}{6}(v^n_{i-1,j} + v^n_{i+1,j} + v^n_{i,j-1} + v^n_{i,j+1}) + \frac{1}{12}(v^n_{i-1,j-1} + v^n_{i+1,j-1} + v^n_{i-1,j+1} + v^n_{i+1,j+1}),\end{aligned} \tag{10}$$

with Neumann boundary conditions. The coefficients of this discretization are the same as in the original paper by Horn and Schunck [12]. In order to have a correct discretization of the Laplacian, these should be chosen so that the sum of coefficients are equal to the coefficient associated with (u, v). The solution of the above sparse system of linear equations can be obtained by means of the following iterative scheme:

$$\begin{aligned}u^{n+1} :=& \bar{u}^n - I_x \frac{I_x \bar{u}^n + I_y \bar{v}^n + I_t}{\alpha^2 + I_x^2 + I_y^2},\\ v^{n+1} :=& \bar{v}^n - I_y \frac{I_x \bar{u}^n + I_y \bar{v}^n + I_t}{\alpha^2 + I_x^2 + I_y^2}.\end{aligned} \tag{11}$$

This iterative scheme is the Jacobi method for solving the linear system (8). To stop the iterative process before a fixed number of iterations, a stopping criterion based on two consecutive values of $\mathbf{h} = (u, v)$ can be used. If $\mathbf{h}^n$, $\mathbf{h}^{n+1}$ are successive approximations of the vector field, the stopping criterion is

$$\frac{1}{N}\sum_{i,j} \left(u^{n+1}_{i,j} - u^n_{i,j}\right)^2 + \left(v^{n+1}_{i,j} - v^n_{i,j}\right)^2 < \varepsilon^2. \tag{12}$$

This criterion is based on the change of the solution in each iteration. The advantage of this is that it is fast and easy to compute. Another alternative is the use of the residual [10].

In this section, we have explained the numerical scheme of the original optical flow method by Horn and Schunck [12]. The software associated to this article contains this implementation in file `horn_schunck_classic.c`.

In the following section, we modify the original method in order to handle larger displacements. We introduce a pyramidal scheme, which is the standard approach in most of nowadays variational optical flow methods. The implementation of this method is included in file `horn_schunck_pyramidal.c`.

3 The Horn and Schunck Method for Recovering Large Displacements: A Pyramidal Approach

The optical flow constraint equation (3) is only suitable when the partial derivatives can be correctly approximated. This is the case when the motion field is small enough or the gradient of the image is linear. However, in the presence of large displacements, these conditions are not typically preserved, and it is common to replace it with a nonlinear formulation:

$$I_1(\mathbf{x}) - I_2(\mathbf{x} + \mathbf{h}) = 0. \tag{13}$$

This constraint still accounts for the brightness constancy assumption with the advantage that $\mathbf{h}$ can have arbitrary large values. A linearization of this equation is equivalent to the optical flow constraint equation (3). Substituting this in the energy model yields the following functional:

$$J(\mathbf{h}) = \int_\Omega \left(I_1(\mathbf{x}) - I_2(\mathbf{x} + \mathbf{h})\right)^2 + \alpha^2 \left(|\nabla u|^2 + |\nabla v|^2\right) \mathrm{d}\mathbf{x}. \tag{14}$$

This model is essentially the same as (5) with the difference that the attachment term is nonlinear. It provides a behavior similar to the original continuous formulation of Horn–Schunck: the solutions are expected to be very smooth, provided that the regularization term is unchanged. The minimization of the energy functional yields the following Euler-Lagrange equations:

$$\begin{aligned} 0 &= -\left(I_1(\mathbf{x}) - I_2(\mathbf{x}+\mathbf{h})\right) I_{2_x}(\mathbf{x}+\mathbf{h}) - \alpha^2 \operatorname{div}(\nabla u), \\ 0 &= -\left(I_1(\mathbf{x}) - I_2(\mathbf{x}+\mathbf{h})\right) I_{2_y}(\mathbf{x}+\mathbf{h}) - \alpha^2 \operatorname{div}(\nabla v). \end{aligned} \tag{15}$$

These equations are nonlinear in $\mathbf{h}$, because of the warping $I_2(\mathbf{x}+\mathbf{h})$. We can use first order Taylor expansions to linearize it,

$$I_2(\mathbf{x}+\mathbf{h}^{n+1}) = I_2(\mathbf{x}+\mathbf{h}^n) + \nabla I_2(\mathbf{x}+\mathbf{h}^n)(\mathbf{h}^{n+1}-\mathbf{h}^n), \tag{16}$$

where $\mathbf{h}^n$ is close to $\mathbf{h}^{n+1}$. Note that, if $\mathbf{h}^n = 0$, and we combine equations (13) and (16), then the resulting formula boils down to the optical flow constraint equation (3). The main difference with respect to the original continuous model is that the temporal derivative is replaced by the difference between the two images. A problem that arises with this linearization is how to determine the value of $\mathbf{h}^n$.

This compels us to introduce two mechanisms: on the one hand, we make use of a multi-scale approach in order to reduce the distance between the objects in the images, so that $\mathbf{h}^n$ is not far from $\mathbf{h}^{n+1}$; on the other hand, in each scale, $\mathbf{h}^n$ is iteratively refined to assure the convergence to $\mathbf{h}^{n+1}$. This last mechanism can be implemented using the motion increment, $\mathbf{dh}$, from $\mathbf{h}^n$ to $\mathbf{h}^{n+1}$ [14, 7, 16], or directly updating the value of $\mathbf{h}^n$ in each iteration [1]. In this work, we follow the latter strategy.

To obtain a linear system of equations, we substitute the nonlinear terms in the Euler-Lagrange equations by their Taylor expansions:

$$\begin{aligned} 0 &= \big(I_1(\mathbf{x}) - I_2(\mathbf{x}+\mathbf{h}^n) - \nabla I_2(\mathbf{x}+\mathbf{h}^n)(\mathbf{h}^{n+1}-\mathbf{h}^n)\big) I_{2_x}(\mathbf{x}+\mathbf{h}^n) + \alpha^2 \operatorname{div}(\nabla u^n), \\ 0 &= \big(I_1(\mathbf{x}) - I_2(\mathbf{x}+\mathbf{h}^n) - \nabla I_2(\mathbf{x}+\mathbf{h}^n)(\mathbf{h}^{n+1}-\mathbf{h}^n)\big) I_{2_y}(\mathbf{x}+\mathbf{h}^n) + \alpha^2 \operatorname{div}(\nabla v^n). \end{aligned} \tag{17}$$

The proposed algorithm consists in solving for $\mathbf{h}^{n+1}$ this linear system of equations iteratively many times at each scale.

3.1 Pyramidal Structure

In order to estimate large displacements, we embed the optical flow method in a multi-scale strategy. The idea behind a multi-scale approach is to create a coarse-to-fine structure that enables the estimation of the flow field at coarser scales and then to refine the solution at finer scales. There are two basic approaches, which theoretically amount to the same thing: the first one is to create a pyramid of downsampled images and the second one is to use a linear scale-space of images at the same resolution [1]. Both approaches provide similar results, but the former has the advantage of working with smaller images, improving the run time of the algorithm.

Our algorithm implements a pyramid of downsampled images. The pyramid is created by reducing the images by a factor $\eta \in (0,1)$. Before downsampling, the images are smoothed with a Gaussian kernel of a standard deviation that depends on η. For a set of scales $s = \{0, 1, \ldots, N_{scales}-1\}$, the pyramid of images is built as

$$I^s(\eta\mathbf{x}) := G_\sigma * I^{s-1}(\mathbf{x}). \tag{18}$$

After the convolution, the images are sampled using bicubic interpolation. The value of σ depends on η and is calculated as

$$\sigma(\eta) := \sigma_0\sqrt{\eta^{-2}-1}, \text{ with } \sigma_0 := 0.6. \tag{19}$$

Intuitively, when η is close to 1 (no downsampling), σ approaches 0, so the image does not change much; and when η is close to 0 (abrupt downsampling), σ is large, so the image becomes very smoothed. The value of σ_0 was found empirically from several values of η and different image sequences. Then, starting at the coarsest scale, the system of equations is solved in each scale to get successive approximations to the optical flow. Every intermediate solution is used as the initialization in the following scale. To transfer the values from a coarser scale, the flow field is updated as

$$\mathbf{h}^{s-1}(\mathbf{x}) := \frac{1}{\eta}\mathbf{h}^{s}(\eta\mathbf{x}). \tag{20}$$

The motion to be detected must be small at the coarsest scale. Thus, it is necessary to create as many scales so that $(1/\eta)^{N_{scales}-1}$ is larger than the magnitude of the largest expected displacement. In this respect, one drawback of the pyramidal approach is that, typically, the method cannot estimate the motion of small objects undergoing large displacements, since these may disappear in the coarsest scales.

3.2 Numerical Scheme

In each scale, the Euler-Lagrange equations are solved using a numerical scheme based on the SOR method (Successive Over-Relaxation). This means that we need to introduce another fixed point iteration level, inner iterations r, for the convergence of the SOR method. This is embedded into the outer iterations, n, of the Taylor expansions. The Laplacian is approximated with finite differences as

$$\begin{aligned}\operatorname{div}\left(\nabla u^{n,r+1}\right) \approx& \frac{1}{6}(u^{n,r+1}_{i-1,j} + u^{n,r+1}_{i+1,j} + u^{n,r+1}_{i,j-1} + u^{n,r+1}_{i,j+1})\\ &+ \frac{1}{12}(u^{n,r+1}_{i-1,j-1} + u^{n,r+1}_{i+1,j-1} + u^{n,r+1}_{i-1,j+1} + u^{n,r+1}_{i+1,j+1}) - u^{n,r+1}_{i,j},\\ \operatorname{div}\left(\nabla v^{n,r+1}\right) \approx& \frac{1}{6}(v^{n,r+1}_{i-1,j} + v^{n,r+1}_{i+1,j} + v^{n,r+1}_{i,j-1} + v^{n,r+1}_{i,j+1})\\ &+ \frac{1}{12}(v^{n,r+1}_{i-1,j-1} + v^{n,r+1}_{i+1,j-1} + v^{n,r+1}_{i-1,j+1} + v^{n,r+1}_{i+1,j+1}) - v^{n,r+1}_{i,j}.\end{aligned} \tag{21}$$

First order derivatives of the images are computed by means of central differences. In the classical method, derivatives are computed using forward differences (see equation (9)), which may lead to a de-localisation of the derivatives. Henceforth, we use the following notation:

$$\begin{aligned}I_1 &:= I_1(\mathbf{x}),\\ I_2 &:= I_2(\mathbf{x}+\mathbf{h}^n),\\ I_{2_x} &:= I_{2_x}(\mathbf{x}+\mathbf{h}^n),\\ I_{2_y} &:= I_{2_y}(\mathbf{x}+\mathbf{h}^n).\end{aligned} \tag{22}$$

Note that $\mathbf{h}^n$ remains fixed during the inner iterations. The numerical scheme is given by

$$\begin{aligned}0 &= \left(I_1 - I_2 - \nabla I_2(\mathbf{h}^{n,r+1} - \mathbf{h}^n)\right) I_{2_x} + \alpha^2 \operatorname{div}\left(\nabla u^{n,r+1}\right),\\ 0 &= \left(I_1 - I_2 - \nabla I_2(\mathbf{h}^{n,r+1} - \mathbf{h}^n)\right) I_{2_y} + \alpha^2 \operatorname{div}\left(\nabla v^{n,r+1}\right).\end{aligned} \tag{23}$$

Then, grouping terms and expressing the above equations with respect to the flow field at iteration $(n, r+1)$ we obtain the following fixed-point iterations:

$$\begin{aligned}u^{n,r+1}_{i,j} :=& (1-w)u^{n,r}_{i,j} + w\frac{\left(\left(I_1 - I_2 + I_{2_x}u^n_{i,j} - I_{2_y}(v^{n,r}_{i,j} - v^n_{i,j})\right) I_{2_x} + \alpha^2 \mathrm{A}(u^{n,r+1}_{i\pm1,j\pm1})\right)}{I^2_{2_x} + \alpha^2},\\ v^{n,r+1}_{i,j} :=& (1-w)v^{n,r}_{i,j} + w\frac{\left(\left(I_1 - I_2 - I_{2_x}(u^{n,r+1}_{i,j} - u^n_{i,j}) + I_{2_y}v^n_{i,j}\right) I_{2_y} + \alpha^2 \mathrm{A}(v^{n,r+1}_{i\pm1,j\pm1})\right)}{I^2_{2_y} + \alpha^2}.\end{aligned} \tag{24}$$

where w is the relaxation parameter of the SOR method, with $0 < w < 2$ (we choose 1.9 by default, as this value has been found to be a stable choice in our experiments). In each scale of the pyramid, the SOR scheme, corresponding to index r, is iterated until the flow converges to a steady-state solution or a maximum number of iterations is reached. Estimated values in instant $r + 1$ are used whenever they are available. This allows us to use a single array to store the optical flow. The index n is used for the outer iterations and is related with the Taylor expansions. Thus, we have two nested iterations, one for the convergence of the numerical scheme and another for the successive refinements of the Taylor linearization. Once the SOR scheme has converged, we go to the next outer iteration $n + 1$.

The value of $\mathrm{A}(u^{n,r+1}_{i\pm1,j\pm1})$ is computed from the neighbors of $u_{i,j}$ as

$$\mathrm{A}(u^{n,r+1}_{i\pm1,j\pm1}) := \frac{1}{6}(u^{n,r+1}_{i-1,j} + u^{n,r+1}_{i+1,j} + u^{n,r+1}_{i,j-1} + u^{n,r+1}_{i,j+1}) + \frac{1}{12}(u^{n,r+1}_{i-1,j-1} + u^{n,r+1}_{i+1,j-1} + u^{n,r+1}_{i-1,j+1} + u^{n,r+1}_{i+1,j+1}). \quad (25)$$

At the coarsest scale, $\mathbf{h}^{N_{scales}-1}$ can be initialized to $(0, 0)$, provided that the motion field is very small. There are some additional numerical issues to be taken into account:

- To warp the image I_2 by a flow field $\mathbf{h}$, $I_2(\mathbf{x} + \mathbf{h})$ is evaluated using bicubic interpolation.
- To upscale a vector field $\mathbf{h}$ by a factor of η, bicubic interpolation is used on each component of the motion field.
- The original images are normalized between 0 and 255 together, so that the value of α is independent of the brightness shift of the images.

4 A Generalized Formulation for Numerical Schemes

A common approach for solving variational problems consists of three steps:

1. The Euler-Lagrange equations are derived analytically.
2. If the equations are nonlinear, they are linearized around a reasonable starting point (or around zero).
3. Finally, the linear equations are discretized and solved numerically using a linear algebra library.

There are several variations of this scheme. For example, the energy functional can be discretized directly, so the analytical computation of the Euler-Lagrange equations is skipped. Sometimes the Euler-Lagrange equations are computed and directly discretized, thereby skipping the linearization step.

In the end always a large linear system of equations is constructed which can be solved by any appropriate method (e.g. Gauss-Elimination, Jacobi, SOR, Conjugate Gradients, GMRES, Multigrid, Algebraic Multigrid, etc.). This allows us to generalize the formulation of our discretized and linearized variational problems. This has the following benefits:

1. It clearly separates mathematical solution from realization.
2. Any numerical method can be employed, easing experimentation.
3. Highly optimized numerical libraries can be developed and debugged independently.

There are many libraries for solving linear systems efficiently [4, 3]. In this section we will explain a generalized formulation for numerical schemes based on the above described examples of optical flow.

4.1 Stencil Notation

Here, we assume that the Euler-Lagrange equations have been computed (equation (15)) and linearized (equation (17)) at a given pyramid level. As a first step, we group the unknowns u^{n+1}, v^{n+1} in each linear term (data term and regularizer) of the current iteration in each equation. We also move constant terms to right hand side and unknowns to the left hand side. This yields the discretized version of the previously linearized Euler Lagrange equations:

$$\begin{aligned}\mathbf{c}_x(\mathbf{x})(u^{n+1}, v^{n+1})^T - \alpha^2(u^{n+1}_{xx} + u^{n+1}_{yy}) &= b_x(\mathbf{x}),\\ \mathbf{c}_y(\mathbf{x})(u^{n+1}, v^{n+1})^T - \alpha^2(v^{n+1}_{xx} + v^{n+1}_{yy}) &= b_y(\mathbf{x}).\end{aligned} \tag{26}$$

Here, we expanded the term $\operatorname{div}(\nabla u) = \operatorname{div}((u_x, u_y))$ to $u_{xx} + u_{yy}$, denoting the second derivatives of u. For abbreviation, the terms constant with respect to the unknowns are summarized in $\mathbf{c}$ and b respectively, defined as:

$$\begin{aligned}b_{d\in\{x,y\}}(\mathbf{x}) &:= I_{2_d}(\mathbf{x}+\mathbf{h}^n)(I_1(\mathbf{x}) - I_2(\mathbf{x}+\mathbf{h}^n)) + I_{2_d}(\mathbf{x}+\mathbf{h}^n)\nabla I_2(\mathbf{x}+\mathbf{h}^n)\mathbf{h}^n,\\ \mathbf{c}_{d\in\{x,y\}}(\mathbf{x}) &:= I_{2_d}(\mathbf{x}+\mathbf{h}^n)\nabla I_2(\mathbf{x}+\mathbf{h}^n).\end{aligned}$$

The next step is to discretize the continuous derivatives in the linear equations. A straightforward approach is to use forward differences on the pixel grid. For example, equation (21) is a smoothed discrete version of $u^{n+1}_{xx} + u^{n+1}_{yy}$ and $v^{n+1}_{xx} + v^{n+1}_{yy}$ at a given pixel location (i, j).

Applying this discretization to a set of unknowns at a given pixel location can be understood as a scalar product between a lexicographically vectorized (from top left to bottom right) unknown vector $\hat{\mathbf{u}}_{i,j}$ and a weight vector $\mathbf{w}_{u_{i,j}}$. For example, we can rewrite equation (21) (skipping the iteration indices) as:

$$\begin{aligned}\operatorname{div}(\nabla u) \approx{}& \mathbf{w}^T_{u_{i,j}}\hat{\mathbf{u}}_{i,j} = \left(\frac{1}{12}, \frac{1}{6}, \frac{1}{12}, \frac{1}{6}, -1, \frac{1}{6}, \frac{1}{12}, \frac{1}{6}, \frac{1}{12}\right)\\ &(u_{i-1,j-1}, u_{i,j-1}, u_{i+1,j-1}, u_{i-1,j}, u_{i,j}, u_{i+1,j}, u_{i-1,j+1}, u_{i,j+1}, u_{i+1,j+1})^T,\\ \operatorname{div}(\nabla v) \approx{}& \mathbf{w}^T_{v_{i,j}}\hat{\mathbf{v}}_{i,j} = \left(\frac{1}{12}, \frac{1}{6}, \frac{1}{12}, \frac{1}{6}, -1, \frac{1}{6}, \frac{1}{12}, \frac{1}{6}, \frac{1}{12}\right)\\ &(v_{i-1,j-1}, v_{i,j-1}, v_{i+1,j-1}, v_{i-1,j}, v_{i,j}, v_{i+1,j}, v_{i-1,j+1}, v_{i,j+1}, v_{i+1,j+1})^T.\end{aligned} \tag{27}$$

Since the weight vectors $\mathbf{w}_{u_{i,j}}$ are always centered around pixel location (i, j), we can write them in image coordinates as:

$$\mathbf{S}_{u_{i,j}} = \begin{bmatrix}\frac{1}{12} & \frac{1}{6} & \frac{1}{12}\\ \frac{1}{6} & -1 & \frac{1}{6}\\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12}\end{bmatrix}_{u_{i,j}}, \mathbf{S}_{v_{i,j}} = \begin{bmatrix}\frac{1}{12} & \frac{1}{6} & \frac{1}{12}\\ \frac{1}{6} & -1 & \frac{1}{6}\\ \frac{1}{12} & \frac{1}{6} & \frac{1}{12}\end{bmatrix}_{v_{i,j}} \tag{28}$$

We call these discrete masks *stencils*. Each pixel location consists of one stencil per unknown (two stencils per pixel for our example). They describe the relation between the current unknown location with respect to neighboring unknowns. Subscripts of $\mathbf{S}$ define to which unknown and which location in the image a stencil corresponds to. The reference location (i, j) is encoded by the center of the stencil. In case it has an even number of rows or columns, the center has to be defined.

The discretization of the derivatives can also be achieved with more accurate methods ensuring e.g. rotational invariance of the discrete slope [19]. Another option is to make use of the closely related research on image derivative filters [8, 18]. More sophisticated approaches include the use of grid structures different from the pixel grid (which is not discussed here), going as far as using irregular grids and solving the system with finite element methods [6].

Now we can rewrite the full discretized Euler Lagrange equation (26) in stencil notation:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -I_{2_x}(\mathbf{x}+\mathbf{h}^n)^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{u_{i,j}} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & -I_{2_x}(\mathbf{x}+\mathbf{h}^n)I_{2_y}(\mathbf{x}+\mathbf{h}^n) & 0 \\ 0 & 0 & 0 \end{bmatrix}_{v_{i,j}} + \alpha^2 \mathbf{S}_{u_{i,j}} = b_x(\mathbf{x})$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & -I_{2_y}(\mathbf{x}+\mathbf{h}^n)I_{2_x}(\mathbf{x}+\mathbf{h}^n) & 0 \\ 0 & 0 & 0 \end{bmatrix}_{u_{i,j}} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & -I_{2_y}(\mathbf{x}+\mathbf{h}^n)^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{v_{i,j}} + \alpha^2 \mathbf{S}_{v_{i,j}} = b_y(\mathbf{x}) \tag{29}$$

This notation can directly be transfered to a data structure for any set of Euler-Lagrange equations to be solved.

4.2 Matrix Notation

For each unknown (in our case $u_{i,j}$ and $v_{i,j}$) at each location (i, j) we can now define a stencil instance based on equation (26), which corresponds to one equation in our large system of linear equations. The number of unknowns of this system is the number of unknowns at each pixel location times the number of pixel locations (one flow vector per pixel). For each equation one option is to group only those unknowns which have non-zero weights as seen in equation (27), yielding a scalar product between weight and unknown vector. Yet, our large linear system contains many more unknowns, each of which is touched by at least one stencil, resulting in non-zero weights. We create this full unknown vector $\hat{\mathbf{u}}$ by lexicographically rearranging the two flow component images $u(\mathbf{x})$ and $v(\mathbf{x})$ to one large vector starting with $u_{0,0}$. This rewriting is defined by a bijection, so each vector element $\hat{u}_k$ can trivially be mapped to a flow vector component $u_{i,j}$ or $v_{i,j}$.

Now, at each pixel location (i, j) we compute a row weight vector $\mathbf{w}_(i, j)$ according to equation (29) by transferring the stencil values to the corresponding unknown vector indices k in $\hat{\mathbf{u}}$. Those weights not referring to any unknowns ($b_x(\mathbf{x})$ and $b_y(\mathbf{x})$) have already been moved to the right hand side (rhs) of the linear equation. All weight vectors are stacked on top of each other, yielding the linear system matrix:

$$\mathbf{A} = \begin{pmatrix} \mathbf{w}_{0,0} \\ \mathbf{w}_{1,0} \\ \vdots \end{pmatrix}. \tag{30}$$

All rhs values are moved into a vector $\hat{\mathbf{b}}$ of the same dimension and order as $\hat{\mathbf{u}}$. This yields a sparse system of linear equations, $\mathbf{A}\hat{\mathbf{u}} = \hat{\mathbf{b}}$, which can be solved with any standard linear algebra library. To better understand the data structure we observe the following properties:

- In case no unknown derivatives (such as u_x or u_{xx}) exist in the Euler Lagrange equations, only the main diagonal would contain weights because they refer to the stencil centers (i, j). This accounts for a trivial solution by diagonal matrix inversion and is only the case in purely local methods.
- As the stencils are centered around the pixel locations and $\hat{\mathbf{u}}$ is ordered lexicographically, weights left and right of (i, j) are located next to each other in the weight vector as well.
- Hence, each line in a given stencil for a given unknown creates a small cluster of nonzero elements in the weight vector.
- Stacking all equations to create $\mathbf{A}$ will therefore yield a sparse, banded matrix, which can be efficiently stored in memory by adequate data structures.

- The creation of **A** can be understood (and rewritten) as a convolution of the stencils with the two-dimensional unknown image.

4.3 Boundary Conditions with Ghost Pixels

In the case of the commonly used Neumann boundary conditions it is assumed that the derivative with respect to the unknowns at domain boundaries are zero. For Dirichlet conditions the derivatives are manually set constants. Two ways exist to deal with such boundary conditions during system matrix construction. One option is to modify the stencil weights whenever a stencil would touch unknowns outside the image domain (e.g. applying the stencil at location $(0, 0)$ would touch for example $(0, -1)$ which does not exist). This approach is error prone as it results in code creating specific stencils at each of the four image boundaries, corners and inside regions of the image.

Alternatively, the concept of ghost pixels can be used. The idea is simple: we add one equation (as well as a pair of associated dummy unknowns) for each boundary pixel location. The stencil for these equations are all the first order derivative for the respective unknowns. This approach takes slightly more memory but can be automatically added to arbitrary variational problems.

We have not implemented this more general numerical scheme using stencil notation and ghost pixels within this paper. A complete, tested open source implementation of this framework including examples for several other optical flow methods is discussed by Gottfried et. al [11].

5 Explanation of the Parameters

In this subsection we explain the parameters of the method and suggest reasonable default values. The algorithm depends on five parameters: the regularization weight (α), the stopping criterion threshold (ε), the downsampling factor (η), the number of scales (N_{scales}) and the number of outer iterations(N_{warps}).

- α is the regularization parameter. It determines the smoothness of the output. The bigger this parameter is, the smoother the solutions we obtain. In the experiments, we observe that a default value can be set to 15. This parameter has units of gray-level. It can be interpreted as the expected standard deviation of the Gaussian noise present in the original images.

- ε is the stopping criterion threshold used in the numerical scheme, which is a trade-off between precision and running time. A small value will yield more accurate solutions at the expense of a slower convergence. Its default value can be set to 0.0001, which, in general, provides good convergence rates with a low running time.

- η is the downsampling factor. It is used to downscale the original images in order to create the pyramidal structure. Its value must be in the interval $(0, 1)$. With $\eta = 0.5$, the images are reduced to half their size in each dimension from one scale to the following. Higher values may slightly improve the results but more scales are necessary to attain the same size of displacements. In our experiments, a default value of 0.65 seems to provide a good performance.

- N_{scales} is used to create the pyramid of images. If the flow field is very small (about one pixel), it can be set to 1. Otherwise, it should be set so that $(1/\eta)^{N-1}$ is larger than the expected size of the largest displacement. A value of $N_{scales} = 5$ with $\eta = 0.5$ allows the method to estimate motions up to 16 pixels (if $\eta = 0.95$, then N_{scales} must be set to 56 to get the same displacements). N_{scales} can be calculated with the following formula: $N_{scales} := -log(max\_motion)/log(\eta) + 1$. In the experiments, its value is computed automatically so that the initial scale corresponds to an image of size about 16.

- N_{warps} represents the number of outer iterations, or warps. This is a parameter that assures the stability of the method. It also affects the running time, so it is a compromise between speed and accuracy. In the experiments, we found that $N_{warps} = 5$ provides a good performance.

Parameter	**Description**	**Default value**
α	regularization parameter	15
ε	stopping threshold	0.0001
η	downsampling factor	0.65
N_{scales}	number of scales	Automatic
N_{warps}	number of outer iterations	5

6 Algorithm

6.1 Classical Horn–Schunck Method

The original Horn and Schunck method can be implemented by Algorithm 1.

Algorithm 1: Horn–Schunck optical flow

```
   Input: $I, \alpha, \varepsilon$
   Output: $\mathbf{h} = (u, v)$
1  Compute $I_x, I_y, I_t$
2  $u \leftarrow 0$
3  $v \leftarrow 0$
4  $n \leftarrow 0$
5  while $n < N_{maxiter}$ and $stopping\_criterion > \varepsilon$ do
6      Compute $\bar{u}, \bar{v}$ using equation (10)
7      $u \leftarrow \bar{u} - I_x \frac{I_x\bar{u} + I_y\bar{v} + I_t}{\alpha^2 + I_x^2 + I_y^2}$
8      $v \leftarrow \bar{v} - I_y \frac{I_x\bar{u} + I_y\bar{v} + I_t}{\alpha^2 + I_x^2 + I_y^2}$
9      Compute $stopping\_criterion$
10     $n \leftarrow n + 1$
11 end
```

The *stopping_criterion* is calculated with the formula given in equation (12), and depends on consecutive approximations, $\mathbf{h}^n$ and $\mathbf{h}^{n+1}$.

6.2 Pyramidal Horn–Schunck Method

Typically, Algorithm 1 is suitable for the detection of small displacements. To detect large displacements, we follow the multi-scale approach explained above. For this, we have implemented a procedure that calculates the flow field in each scale and an algorithm that creates and handles the pyramidal structure. The procedure updates iteratively the vector field $\mathbf{h}$ from the previous scale approximation. The initial value for $\mathbf{h}$ is given by the enclosing multiscale procedure and it is zero at the coarsest level. The procedure that calculates the optical flow in each scale is given in Algorithm 2. The pyramidal structure is implemented in Algorithm 3. These algorithms are implemented using `float` precision numbers.

Procedure horn_schunck_optic_flow($I_1, I_2, \mathbf{h}, \alpha, \varepsilon, N_{maxiter}, N_{warps}$)

```
1  Compute $I_{2_x}, I_{2_y}$
2  for $n \leftarrow 1$ to $N_{warps}$ do
3      Compute $I_2(\mathbf{x}+\mathbf{h}), I_{2_x}(\mathbf{x}+\mathbf{h}), I_{2_y}(\mathbf{x}+\mathbf{h})$ using bicubic interpolation
4      $u^n \leftarrow u$
5      $v^n \leftarrow v$
6      $r \leftarrow 0$
7      while $r < N_{maxiter}$ and $stopping\_criterion > \varepsilon$ do
8          Compute $\mathrm{A}(u), \mathrm{A}(v)$ using equation (25)
9          $u \leftarrow (1-w)u + w\frac{\left(\left(I_1-I_2+I_{2_x}u^n-I_{2_y}(v-v^n)\right)I_{2_x}+\alpha^2\mathrm{A}(u)\right)}{I_{2_x}^2+\alpha^2}$
10         $v \leftarrow (1-w)v + w\frac{\left(\left(I_1-I_2-I_{2_x}(u-u^n)+I_{2_y}v^n\right)I_{2_y}+\alpha^2\mathrm{A}(v)\right)}{I_{2_y}^2+\alpha^2}$
11         Compute the $stopping\_criterion$
12         $r \leftarrow r+1$
13     end
14 end
```

7 Examples

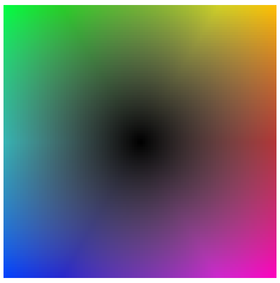


Figure 4: Color wheel used for representing the flow fields.

Let us start by showing the effect of the parameters with some simple examples. The simplest possible example is a video sequence of the form

$$I(x,y,t) = ax + by + ct, \tag{31}$$

that is, the video is described globally by an affine function. In that case, the basic method gives an accurate result in one iteration for α=0. This is due to the fact that an affine function is differentiated accurately by any finite difference scheme, and the first iteration in that case amounts to

$$\mathbf{h} = \frac{-c}{\alpha^2 + a^2 + b^2}(a,b), \tag{32}$$

which, for $\alpha = 0$ gives a motion in the direction of the ramp with the correct magnitude. When α is smaller than the gradient of I, the first iteration provides already a good approximation of the correct motion. In the experiment shown on figure 5, a correct displacement of 70 pixels is recovered accurately on the first iteration.

A slightly more realistic setting is a smooth blob that moves 3 pixels to the right (figure 6). In that case, the first iteration of the method gives a vector field that satisfies the optical flow constraint at each point, but is not globally coherent due to the aperture problem. Successive iterations propagate

Algorithm 3: Pyramidal Horn–Schunck optical flow

```
    Input: $I_1, I_2, \alpha, \varepsilon, \eta, N_{maxiter}, N_{warps}, N_{scales}$
    Output: $\mathbf{h}$
 1  Normalize $I_1, I_2$ between 0 and 255
 2  Convolve $I_1, I_2$ with a Gaussian of $\sigma := 0.8$
 3  for $s \leftarrow 0$ to $N_{scales} - 1$ do
 4      Downscale images $I_1, I_2$ by $\eta$
 5  end
 6  $\mathbf{h}^{N_{scales-1}} \leftarrow (0,0)$
 7  for $s \leftarrow N_{scales} - 1$ to $0$ step $-1$ do
 8      horn_schunck_optic_flow($I_1, I_2, \mathbf{h}^s, \alpha, \varepsilon, N_{maxiter}, N_{warps}$)
 9      if $s > 0$ then
10          $\mathbf{h}^{s-1}(\mathbf{x}) \leftarrow \frac{1}{\eta}\mathbf{h}^s(\eta\mathbf{x})$
11      end
12  end
```

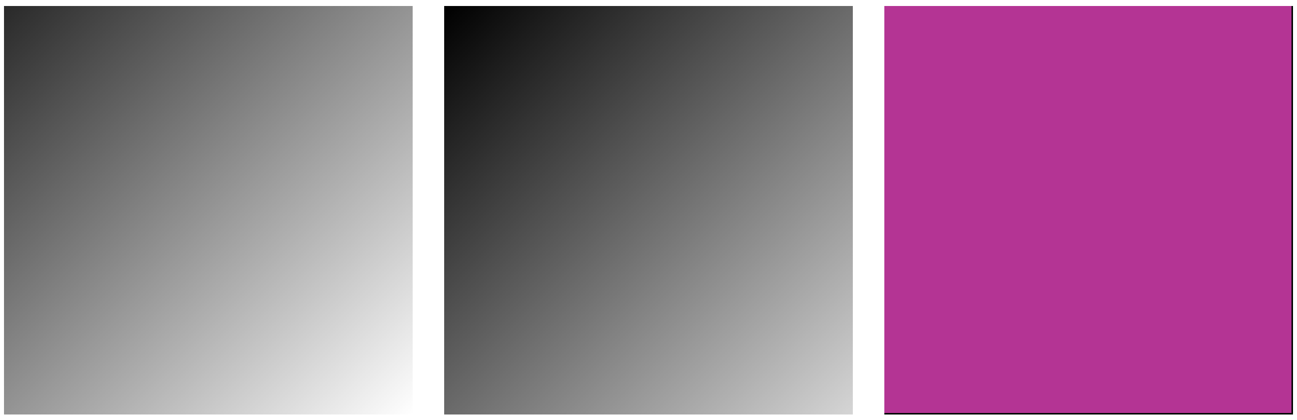


Figure 5: Ramp sequence. Left, the first image; Middle, the second image; Right, the optical flow.

the information towards the rest of the image until a constant vector field is achieved. Notice that this constant vector field is effectively the minimum of the energy functional in that case, because its energy vanishes.

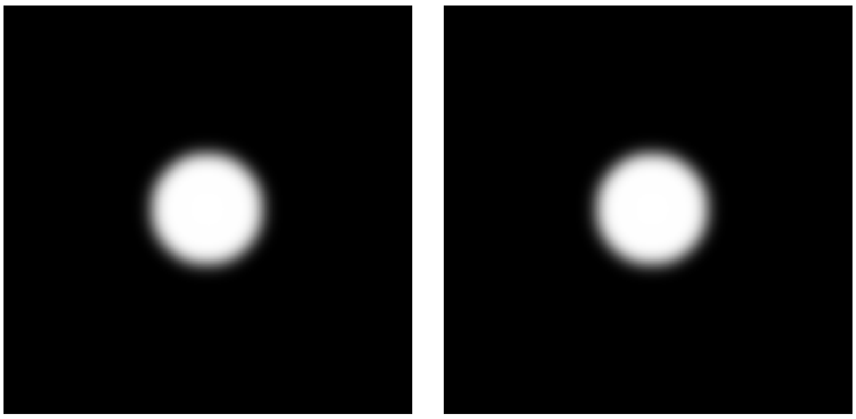


Figure 6: Smooth blob moving 3 pixels to the right.

First, let us study the effect of the parameter α on the first iteration, depicted on figure 7. It only has an effect on regions of the input image where the gradient is smaller than α. In those regions, the data term is not used (so, in the first iteration, the produced field is forced to vanish). As explained in the original article by Horn and Schunck [12], the value of this parameter can be interpreted as the expected error in the image gradient. For this synthetic experiment, the only error is the numeric noise (of size about 10^{-15}) due to the Fourier transforms which were used to compute the smoothing in the frequency domain. This noise becomes clearly visible at $\alpha = 0$.

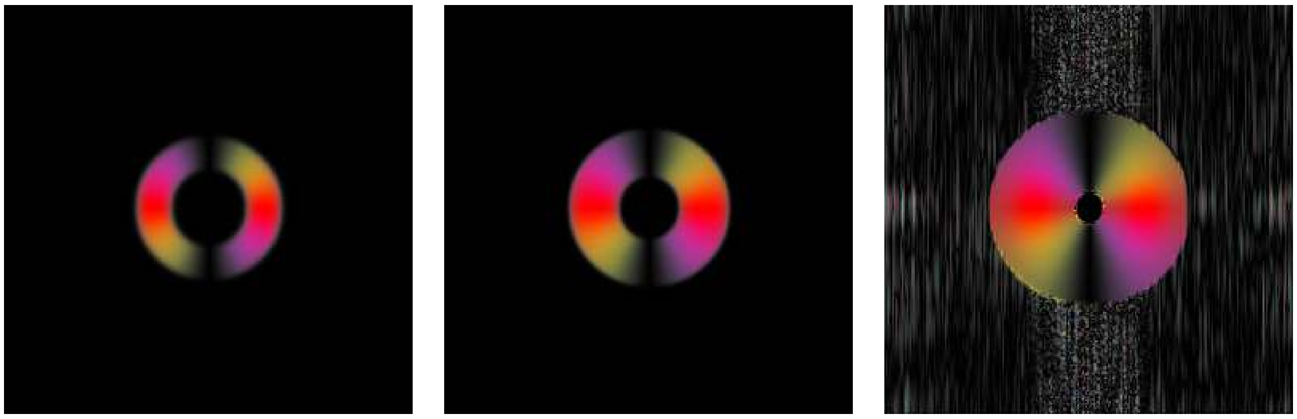


Figure 7: Optical flows obtained for the blob sequence for $N_{iter} = 1$ and different values of α: left flow with $\alpha = 2.56$; middle flow with $\alpha = 0.256$; and right flow with $\alpha = 0$.

The images in figure 8 show how the iterations propagate the correct flow information isotropically all over the image. The black lines on these images are the level lines $\|\nabla \mathbf{h}\| = c$ for $c = 1, 2, 3$ and the iterations run until 300.000.

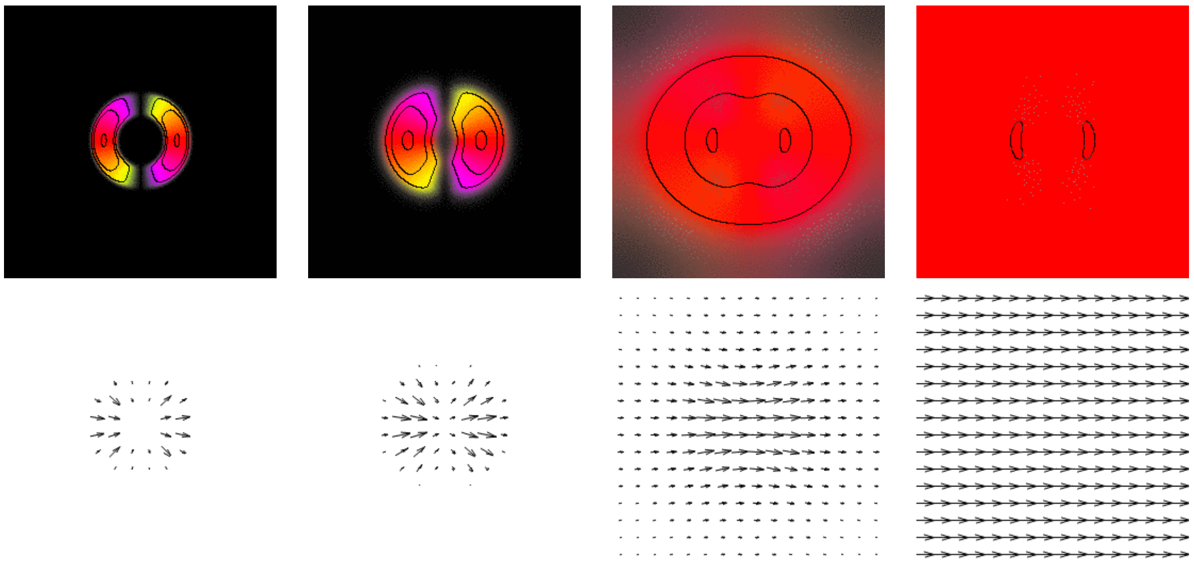


Figure 8: Evolution of the flow field for increasing number of iterations (N_{iter}).

These experiments show an interesting feature of the method: if the image and the required flow are smooth enough, the correct solution can be recovered, regardless of the magnitude of the flow. The smoothness of the image can be enforced by convolving it with a Gaussian.

To study the effect of the parameters, we run the algorithm for a range of values of each parameter, keeping the values of the others at their default values. We use three illustrative sequences (figure 9): Baboon, where the flow is smooth and small, (a rotation); Book, where the flow has large displacements and occlusions; and Urban2, from the Middlebury benchmark database.

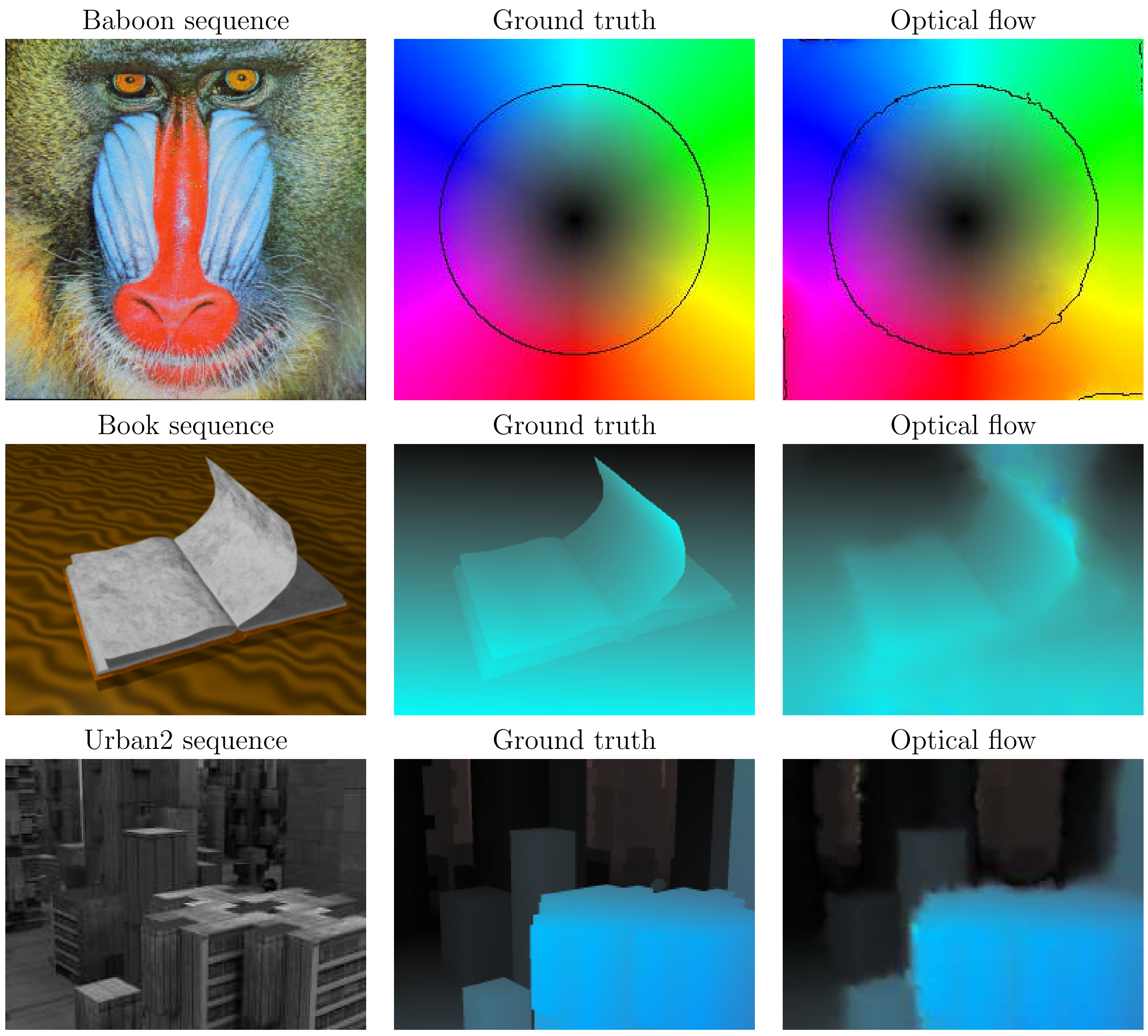


Figure 9: Results for the Baboon, Book and Urban2 sequences. The level lines in the Baboon flows, represent the flows whose magnitude is equal to one.

The first graph in figure 10 shows that the iterations generally converge for ε around 0.001. For smaller values, the running time increases without any improvement on the result. In these graphics, we use the average end-point (EPE) and average angular (AAE) errors [2]. The second graphic in figure 10 shows that the best value of the regularization parameter α depends on the input data. For example, to recover a smooth motion it is good to set $\alpha = 20$, and to find a motion with many occlusions it is best to set α around 5.

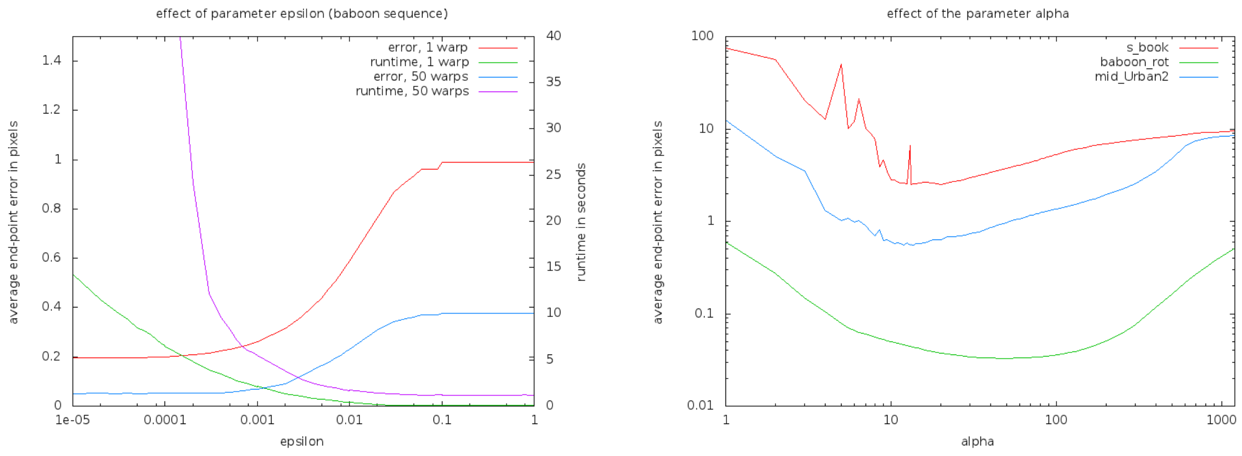


Figure 10: Effect of the ε and α parameters. Notice that the optimal value for the parameter α differs from image to image.

In the first column of figure 11, we see the effect of the scale step parameter. A conclusion that can be drawn is that when the scale step is small enough (near 0.8), the optimizing effect of the warps is less apparent. In the second column of the same figure, we observe that increasing the number of warps per scale always improves the result, but there are diminishing returns due to the dramatic rise of running times.

7.1 Middlebury Database

This subsection shows several tests with the sequences in the Middlebury benchmark database [2]. This database contains two types of data: those for which the ground truths are made public (called "Test sequences"), and those for which the ground truths are not public ("Evaluation sequences"). Other benchmarks [5, 15, 9] are not described in this article but can be tested through the demo system.

7.1.1 Test Sequences

For the test sequences, we have run the algorithm with the same parameter set: $\alpha = 15$, $\varepsilon = 0.0001$, $\eta = 0.65$ and 5 warpings. We have also computed the optical flows adapting the value of α in order to find a better result.

Figure 12 shows the 10th frame of the sequence, the ground truth, the solution with fixed parameters and the best solution we have found. These sequences are composed of more than two frames, but the ground truth is only available for the 10th frame.

In table 1 we show the EPE and AAE for the Middlebury test sequences when fixed parameters are used, which correspond to the optical flows in the third column of figure 12. In the fourth column of the figure, we show the best optical flows found and its corresponding α. The numerical results are shown in table 2.

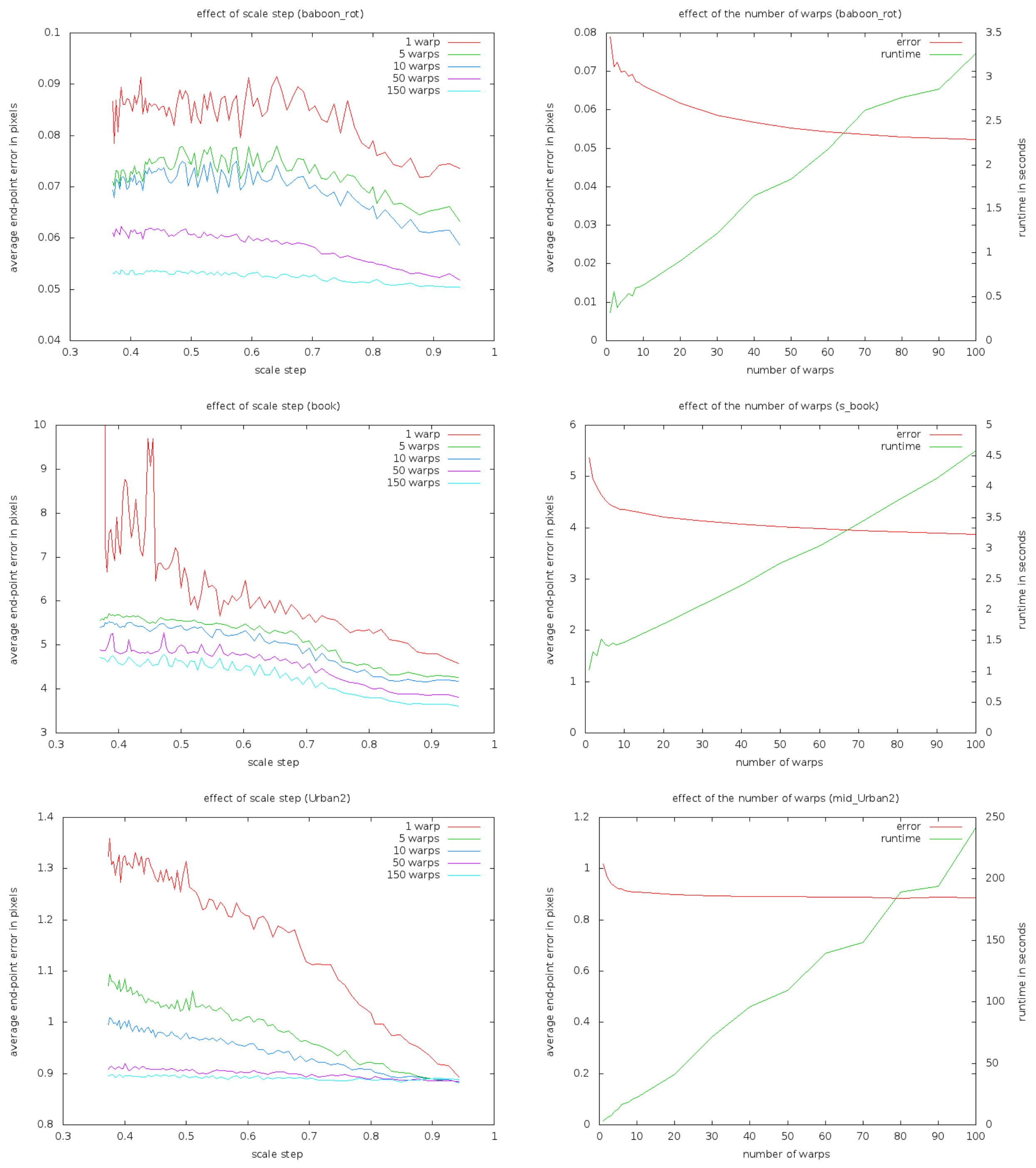


Figure 11: Effect of the scale step parameter (left column) and the number of warps (right column) for the Baboon, Book and Urban2 sequences.

Table 1: EPE and AAE for default parameters using the Middlebury test sequences.

	Dimetrodon	Grove2	Grove3	Hydrangea	Rubberwhale	Urban2	Urban3	Venus
EPE	0.151	0.219	0.847	0.327	0.241	0.561	1.071	0.451
AAE	2.768^o	3.105^o	7.586^o	3.427^o	7.913^o	5.086^o	10.614^o	7.594^o

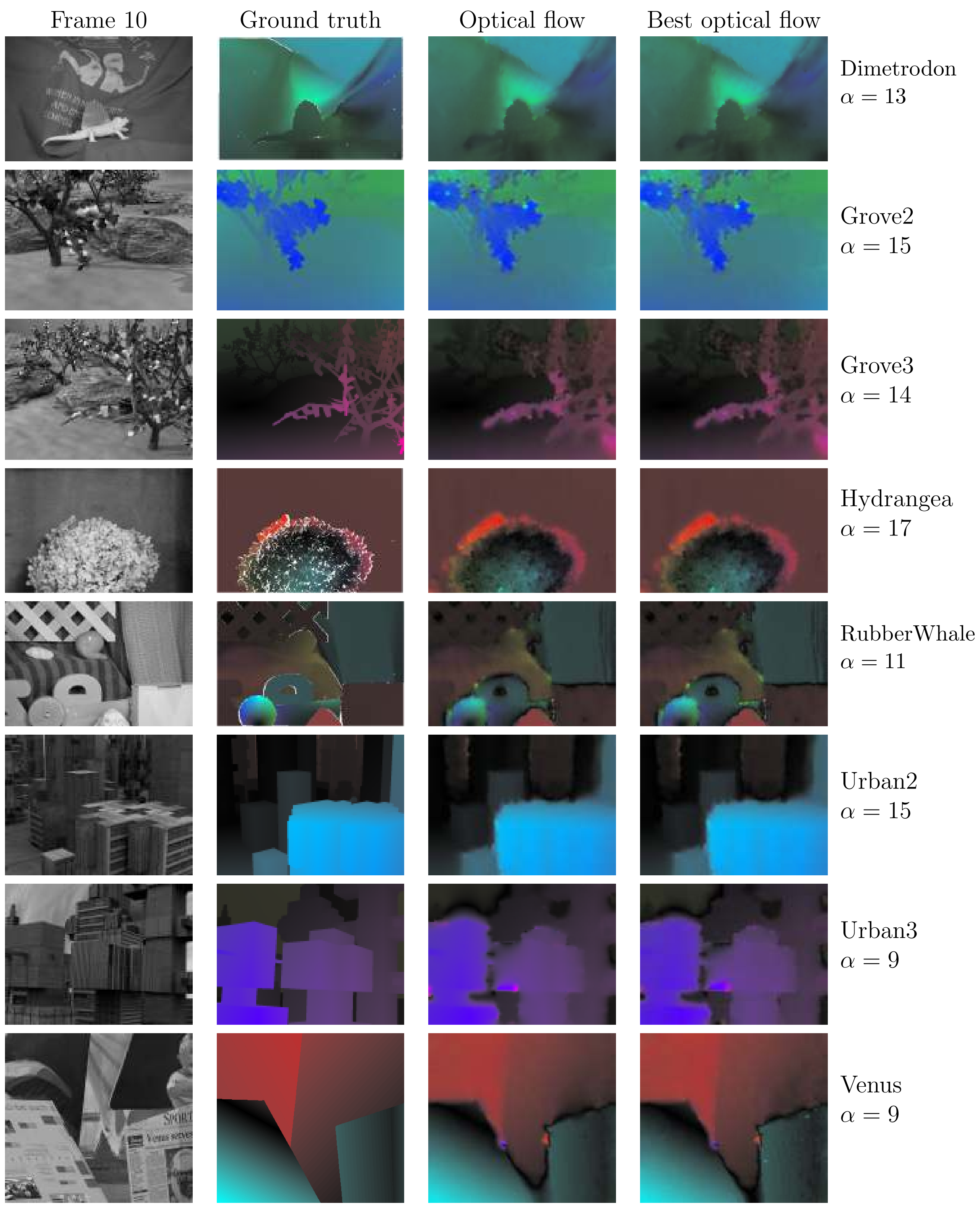


Figure 12: Middlebury test sequences.

Table 2: EPE and AAE for the best optical flows using the Middlebury test sequences.

	Dimetrodon	Grove2	Grove3	Hydrangea	Rubberwhale	Urban2	Urban3	Venus
EPE	0.150	0.219	0.842	0.327	0.235	0.561	0.941	0.407
AAE	2.767^o	3.105^o	7.608^o	3.364^o	7.684^o	5.086^o	8.830^o	6.723^o

7.1.2 Evaluation Sequences

Finally, in figure 13 we show several examples using the Middlebury evaluation sequences. We have used the following parameter set: $\alpha = 15$, $\varepsilon = 0.0001$, $\eta = 0.65$ and 5 warpings.

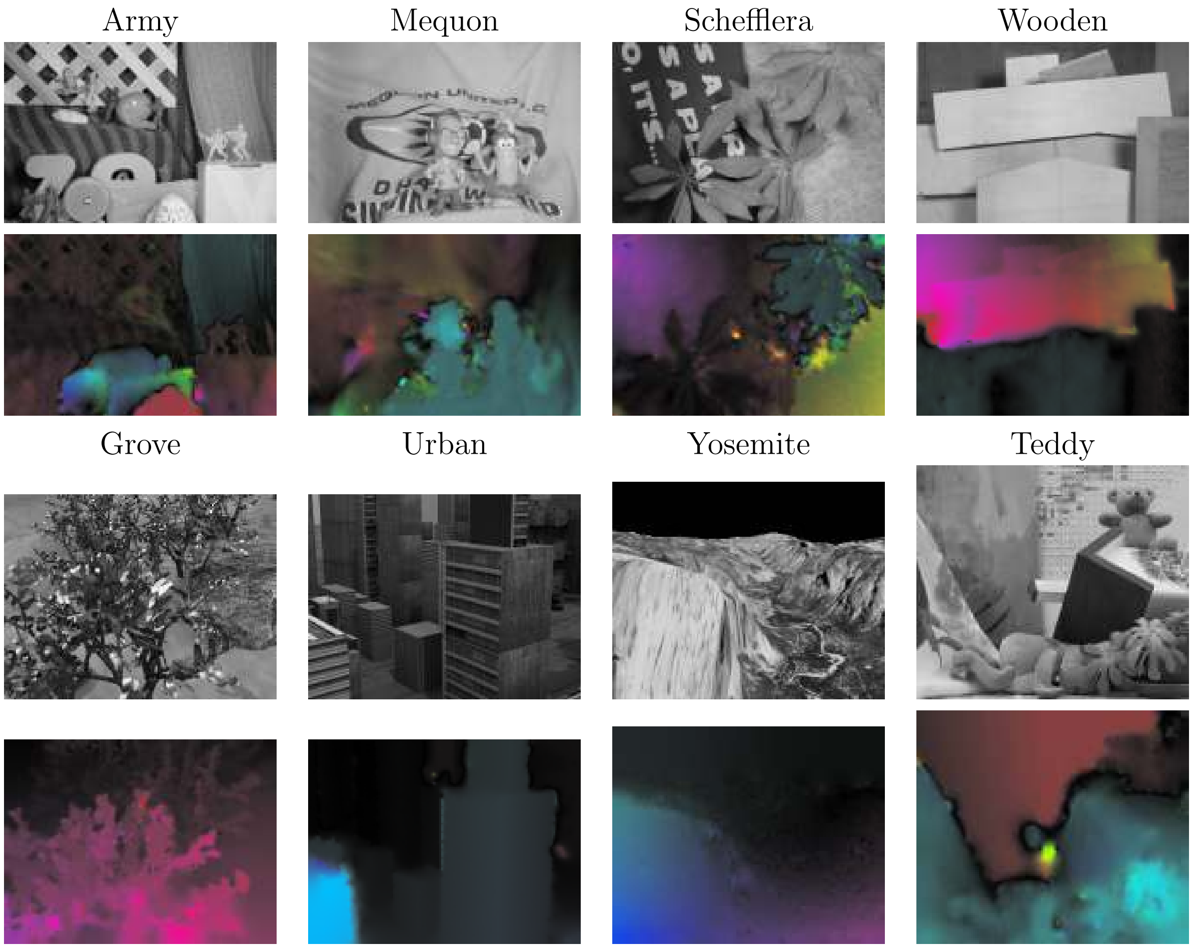


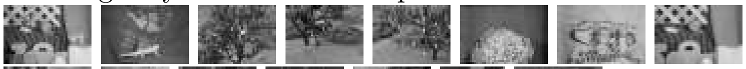
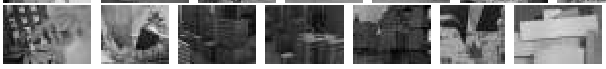
Figure 13: Middlebury evaluation sequences.

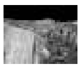
Acknowledgements

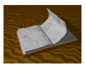
This work has been partly founded by the Spanish Ministry of Science and Innovation through the research project TIN2011-25488, by the Centre National d'Etudes Spatiales (CNES, MISS Project), by the European Research Council (advanced grant Twelve Labours) and by the Office of Naval research (ONR grant N00014-97-1-0839).

Image Credits

All images by the authors except:


 Middlebury benchmark database [2].

 Lynn Quam.

 Synthetic image. Book sequence frame 25. Computer Laboratory, Cambridge.

Standard test image.

References

[1] Luis Álvarez, Joachim Weickert, and Javier Sánchez. Reliable estimation of dense optical flow fields with large displacements. *International Journal of Computer Vision*, 39(1):41–56, 2000. http://dx.doi.org/10.1023/A:1008170101536.

[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. http://dx.doi.org/10.1109/ICCV.2007.4408903.

[3] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2001. http://www.mcs.anl.gov/petsc.

[4] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997. http://dx.doi.org/10.1007/978-1-4612-1986-6_8.

[5] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12:43–77, 1994. http://dx.doi.org/10.1007/BF01420984.

[6] S.C. Brenner and L.R. Scott. *The mathematical theory of finite element methods*, volume 15. Springer Verlag, 2008. ISBN:3540941932.

[7] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim 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 *Lecture Notes in Computer Science*, pages 25–36, Prague, Czech Republic, May 2004. Springer. http://dx.doi.org/10.1007/978-3-540-24673-2_3.

[8] H. Farid and E. Simoncelli. Optimally rotation-equivariant directional derivative kernels. In *Computer Analysis of Images and Patterns*, pages 207–214. Springer, 1997. http://dx.doi.org/10.1007/3-540-63460-6_119.

[9] B. Galvin, B. McCane, K. Novins, D. Mason, and S. Mills. Recovering motion fields: An evaluation of eight optical flow algorithms. In *British Machine Vision Conference*, pages 195–204, 1998. http://dx.doi.org/10.5244/C.12.20.

[10] C.F. Gerald and C.J. Green. *Applied numerical analysis.* Pearson Education, 2003. ISBN:0201565536.

[11] J.M. Gottfried and D. Kondermann. Charon suite software framework. In *IPOL 2012 Meeting on Image Processing Libraries*, June 2012.

[12] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981. http://dx.doi.org/10.1016/0004-3702(93)90173-9.

[13] David Marr and Ellen Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167):187–217, 1980. `http://dx.doi.org/10.1098/rspb.1980.0020`.

[14] E. Memin and P. Perez. Dense estimation and object-based segmentation of the optical-flow with robust techniques. *IEEE Transactions on Image Processing*, 7(5):703–719, May 1998. `http://dx.doi.org/10.1109/83.668027`.

[15] Amar Mitiche and Patrick Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *International Journal of Computer Vision*, 19:29–55, 1996. `http://dx.doi.org/10.1007/BF00131147`.

[16] 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. `http://dx.doi.org/10.1007/s11263-005-3960-y`.

[17] Javier Sánchez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L^1 optical flow estimation. *Image Processing On Line*, Preprint, 2013.

[18] H. Scharr. Optimal filters for extended optical flow. In *Complex Motion*, volume 3417 of *LNCS*. Springer, 2004. `http://dx.doi.org/10.1007/978-3-540-69866-1_2`.

[19] G.D. Smith. *Numerical solution of partial differential equations: finite difference methods*. Oxford University Press, USA, 1985. ISBN:0198596111.

[20] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In *Proceedings of the 29th DAGM conference on Pattern recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag. `http://dx.doi.org/10.1007/978-3-540-74936-3_22`.