Robust Optical Flow Estimation

Javier Sánchez[1], Nelson Monzón[2] and Agustín Salgado[3]

[1]CTIM, University of Las Palmas de Gran Canaria, Spain (jsanchez@dis.ulpgc.es)
[2]CTIM, University of Las Palmas de Gran Canaria, Spain (nmonzon@ctim.es)
[3]CTIM, University of Las Palmas de Gran Canaria, Spain (asalgado@dis.ulpgc.es)

Communicated by Enric Meinhardt-Llopis   *Demo edited by* Enric Meinhardt-Llopis

Abstract

In this work, we describe an implementation of the variational method proposed by Brox et al. in 2004, which yields accurate optical flows with low running times. It has several benefits with respect to the method of Horn and Schunck: it is more robust to the presence of outliers, produces piecewise-smooth flow fields and can cope with constant brightness changes. This method relies on the brightness and gradient constancy assumptions, using the information of the image intensities and the image gradients to find correspondences. It also generalizes the use of continuous L^1 functionals, which help mitigate the effect of outliers and create a Total Variation (TV) regularization. Additionally, it introduces a simple temporal regularization scheme that enforces a continuous temporal coherence of the flow fields.

Source Code

The source code, the code documentation, and the online demo are accessible at the IPOL web page of this article[1]. In this page an implementation is available for download. This file contains two directories: one for the spatial method and another for the temporal method. The spatial method is suitable for general image sequences, while the temporal method should be used when the flow fields are known to be very continuous.

Keywords: optical flow, motion estimation, variational techniques, PDE

1 Introduction

The estimation of accurate motion fields is an important challenge in image processing and computer vision. Among the most accurate methods in the literature, variational approaches have proven an outstanding performance with respect to other strategies. Starting from the first variational model of Horn and Schunck [5], we find an extensive literature on such methods.

Brox et al. [3] proposed a technique that is based on differentiable L^1 functionals. This kind of functionals allows to create piecewise-smooth flow fields. A more efficient TV-L^1 optical flow method

---

[1]https://doi.org/10.5201/ipol.2013.21

was presented in Zach et al. [11], which has also been implemented on an IPOL article [9]. The benefit of the Brox et al. method is that it keeps a better spatial coherence between both unknowns of the optical flow, at the expense of creating rounded effects at flow discontinuities. The Zach et al. method creates sharper discontinuities, but it strongly suffers from the staircase effect of pure TV schemes. Rounded effects at motion discontinuities can be mitigated with the use of decreasing functions, like in the work of Wedel et al. [10]. A different alternative is to use image-based anisotropic schemes, like in the work of Álvarez et al. [1], that allows respecting the image discontinuities.

In this work, we implement the spatial and temporal methods as in the original proposal. Both methods are very similar, with the main difference being that the temporal method includes a continuous smoothing scheme in the temporal dimension. This is suitable when the optical flow functions are smooth. Nevertheless, the flow discontinuities are typically degraded by the temporal regularizer. In the presence of large displacements, it is better to use a nonlinear temporal scheme, like in Salgado and Sánchez [8].

Given a sequence of images, $I : \Omega \subset \mathbb{R}^3 \to \mathbb{R}$, of gray level values in space and time, $\mathbf{x} = (x, y, t)^T \in \Omega$, the optical flow is defined as a dense mapping, $\mathbf{w} = (u(\mathbf{x}), v(\mathbf{x}), 1)^T$, between the pixels of every two consecutive images. The scalar fields $u(\mathbf{x})$ and $v(\mathbf{x})$ are the x and y displacements in the 3D volume, respectively. Each frame is considered to be at a distance 1 in time from the previous and following frames.

The spatial gradient of the image is given by $\nabla I = (I_x, I_y)^T$, with I_x, I_y the first order derivatives in x and y. The gradient of the optical flow is defined as

$$\nabla u = \begin{cases} (u_x, u_y)^T, \\ (u_x, u_y, u_t)^T. \end{cases} \tag{1}$$

The first spatial gradient is used in the *Spatial method*, where the optical flow is computed between two frames. The second spatio-temporal gradient is used in the *Temporal method*. Since the abstract framework is identical in both cases, we keep the same notation for both methods. This distinction appears later in the numerical scheme (section 2).

We suppose that the pixel intensities remain constant along the trajectories of the moving particles. This is normally referred to as the *brightness constancy assumption*, which in a continuous setting yields the *optical flow constraint* equation, proposed by Horn and Schunck [5] (see also the related IPOL implementation [6]). In general, we suppose that our problem is not continuous, so that the brightness constancy assumption can be written in its nonlinear form as

$$I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x}) = 0. \tag{2}$$

The following energy model is a slight variation of the one proposed by Brox et al. in 2004 [3]:

$$E(\mathbf{w}) = \int_\Omega \Psi \left((I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x}))^2 \right) \mathbf{dx} + \gamma \int_\Omega \Psi \left(|\nabla I(\mathbf{x} + \mathbf{w}) - \nabla I(\mathbf{x})|^2 \right) \mathbf{dx}$$

$$+ \alpha \int_\Omega \Psi \left(|\nabla u|^2 + |\nabla v|^2 \right) \mathbf{dx}. \tag{3}$$

with $\Psi(s^2) = \sqrt{s^2 + \epsilon^2}$ and $\epsilon := 0.001$ a small constant. In the original proposal [3], the brightness and gradient constancy terms (first two terms) were included inside the same Ψ function. As proposed by Bruhn and Weickert [4], the separation of these two assumptions, like in equation (3), is better justified. In this sense, our model is more similar to the one by Bruhn and Weickert [4].

The last term corresponds to the regularizing strategy, which is a differentiable approximation of the TV scheme. The method of Zach et al. [11], or Wedel et al. [10], uses a pure TV scheme and

separates both components of the optical flow. Although both approaches rely on L^1 functionals, they provide very different results, as can be seen in the IPOL article of Sánchez et al. [9].

The minimum of the previous energy model can be found by solving the associated Euler-Lagrange equations, given by

$$
\begin{aligned}
0 =& \Psi'_D \cdot (I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x})) \cdot I_x(\mathbf{x} + \mathbf{w}) \\
& + \gamma\, \Psi'_G \cdot ((I_x(\mathbf{x} + \mathbf{w}) - I_x(\mathbf{x})) \cdot I_{xx}(\mathbf{x} + \mathbf{w}) + (I_y(\mathbf{x} + \mathbf{w}) - I_y(\mathbf{x})) \cdot I_{xy}(\mathbf{x} + \mathbf{w})) \\
& - \alpha\, \mathrm{div}\, (\Psi'_S \cdot \nabla u)\,, \\
0 =& \Psi'_D \cdot (I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x})) \cdot I_y(\mathbf{x} + \mathbf{w}) \\
& + \gamma\, \Psi'_G \cdot ((I_x(\mathbf{x} + \mathbf{w}) - I_x(\mathbf{x})) \cdot I_{xy}(\mathbf{x} + \mathbf{w}) + (I_y(\mathbf{x} + \mathbf{w}) - I_y(\mathbf{x})) \cdot I_{yy}(\mathbf{x} + \mathbf{w})) \\
& - \alpha\, \mathrm{div}(\Psi'_S \cdot \nabla v),
\end{aligned}
\tag{4}
$$

with $\Psi'(s^2) = \frac{1}{2\sqrt{s^2 + \epsilon^2}}$. In order to simplify the equations, we use the following notation:

$$
\begin{aligned}
\Psi'_D :=& \Psi'\left((I(\mathbf{x} + \mathbf{w}) - I(\mathbf{x}))^2\right), \\
\Psi'_G :=& \Psi'\left(|\nabla I(\mathbf{x} + \mathbf{w}) - \nabla I(\mathbf{x})|^2\right), \\
\Psi'_S :=& \Psi'\left(|\nabla u|^2 + |\nabla v|^2\right).
\end{aligned}
\tag{5}
$$

Equation (4) cannot be solved easily because it is nonlinear in expressions like $I(\mathbf{x} + \mathbf{w})$. To avoid these nonlinearities, we use first order Taylor expansions. We introduce an index, k, so that our current unknown, $\mathbf{w}^{k+1}$, depends on a previous constant value, $\mathbf{w}^k$.

$$
\begin{aligned}
I(\mathbf{x} + \mathbf{w}^{k+1}) \approx& I(\mathbf{x} + \mathbf{w}^k) + I_x(\mathbf{x} + \mathbf{w}^k)du^k + I_y(\mathbf{x} + \mathbf{w}^k)dv^k \\
I_x(\mathbf{x} + \mathbf{w}^{k+1}) \approx& I_x(\mathbf{x} + \mathbf{w}^k) + I_{xx}(\mathbf{x} + \mathbf{w}^k)du^k + I_{xy}(\mathbf{x} + \mathbf{w}^k)dv^k \\
I_y(\mathbf{x} + \mathbf{w}^{k+1}) \approx& I_y(\mathbf{x} + \mathbf{w}^k) + I_{xy}(\mathbf{x} + \mathbf{w}^k)du^k + I_{yy}(\mathbf{x} + \mathbf{w}^k)dv^k,
\end{aligned}
\tag{6}
$$

with $\mathbf{w}^k = (u^k, v^k)^T$, $du^k = u^{k+1} - u^k$ and $dv^k = v^{k+1} - v^k$. We assume that $\mathbf{w}^k$ is a close approximation to our unknown $\mathbf{w}^{k+1}$. The original method, and our implementation, works directly with the *motion increments* (du^k, dv^k). The optical flow is incrementally updated from the motion increment as $u^{k+1} = u^k + du^k$, $v^{k+1} = v^k + dv^k$. This is the strategy followed in other works like, for instance, in Mémin and Pérez [7]. A different alternative is to compute the full optical flow directly, like in the work by Álvarez et al. [1].

There still remains another nonlinearity due to the Ψ' functions. Thus, our numerical scheme should be enclosed in two fixed point iterations: the outer iterations, k, related with the stability of the Taylor expansions; and the inner iterations, l, that account for the nonlinearities of the Ψ' functions. Therefore, the system of equations reads as

$$
\begin{aligned}
0 =& (\Psi'_D)^{k,l} \cdot \left(I(\mathbf{y}) + I_x(\mathbf{y})du^{k,l+1} + I_y(\mathbf{y})dv^{k,l+1} - I(\mathbf{x})\right) \cdot I_x(\mathbf{y}) \\
& + \gamma\, (\Psi'_G)^{k,l} \cdot \left(\left(I_x(\mathbf{y}) + I_{xx}(\mathbf{y})du^{k,l+1} + I_{xy}(\mathbf{y})dv^{k,l+1} - I_x(\mathbf{x})\right) \cdot I_{xx}(\mathbf{y})\right. \\
& \left. + \left(I_y(\mathbf{y}) + I_{xy}(\mathbf{y})du^{k,l+1} + I_{yy}(\mathbf{y})dv^{k,l+1} - I_y(\mathbf{x})\right) \cdot I_{xy}(\mathbf{y}^{k,l})\right) \\
& - \alpha\, \mathrm{div}\left((\Psi'_S)^{k,l} \cdot \nabla(u^{k,l} + du^{k,l+1})\right) \\
0 =& (\Psi'_D)^{k,l} \cdot \left(I(\mathbf{y}) + I_x(\mathbf{y})du^{k,l+1} + I_y(\mathbf{y})dv^{k,l+1} - I(\mathbf{x})\right) \cdot I_x(\mathbf{y}) \\
& + \gamma\, (\Psi'_G)^{k,l} \cdot \left(\left(I_x(\mathbf{y}) + I_{xx}(\mathbf{y})du^{k,l+1} + I_{xy}(\mathbf{y})dv^{k,l+1} - I_x(\mathbf{x})\right) \cdot I_{xx}(\mathbf{y})\right. \\
& \left. + \left(I_y(\mathbf{y}) + I_{xy}(\mathbf{y})du^{k,l+1} + I_{yy}(\mathbf{y})dv^{k,l+1} - I_y(\mathbf{x})\right) \cdot I_{xy}(\mathbf{y}^{k,l})\right) \\
& - \alpha\, \mathrm{div}\left((\Psi'_S)^{k,l} \cdot \nabla(v^{k,l} + dv^{k,l+1})\right),
\end{aligned}
\tag{7}
$$

with $\mathbf{y} = \mathbf{x} + \mathbf{w}^{k,l}$.

2 Numerical Scheme

The system of equations (7) can be efficiently solved using the SOR method. The unknowns $du^{k,l+1}$ and $dv^{k,l+1}$, in pixel (i,j,k), are expressed in terms of the remaining terms, and their values are iteratively updated until the method converges to a steady state solution. In this sense, we introduce an additional fixed point iteration scheme, s, for the SOR method.

Partial derivatives are approximated using central differences. The discretization of the divergence is separated in three variables: $\text{div}\left((\Psi'_S)^{k,l} \cdot \nabla(u^{k,l} + du^{k,l+1})\right) = \text{div}\left((\Psi'_S)^{k,l} \cdot \nabla u^{k,l}\right) + \text{div}\left((\Psi'_S)^{k,l} \cdot \nabla du^{k,l+1}\right) \approx div\_u + (div\_du - div\_d \cdot du^{k,l+1}_{i,j,k})$, where div_u discretizes the first divergence term, div_d and div_du correspond to the second term. In the second term, div_du stands for the values corresponding to the neighbors of du, and div_d stands for the coefficients accompanying du at the current pixel, $du^{k,l+1}_{i,j,k}$. These variables are given by the following expressions:

$$
\begin{aligned}
div\_u :=& \frac{(\Psi'_S)_{i+1,j,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}\left(u^{k,l}_{i+1,j,k} - u^{k,l}_{i,j,k}\right) + \frac{(\Psi'_S)_{i-1,j,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}\left(u^{k,l}_{i-1,j,k} - u^{k,l}_{i,j,k}\right) + \\
& \frac{(\Psi'_S)_{i,j+1,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}\left(u^{k,l}_{i,j+1,k} - u^{k,l}_{i,j,k}\right) + \frac{(\Psi'_S)_{i,j-1,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}\left(u^{k,l}_{i,j-1,k} - u^{k,l}_{i,j,k}\right) + \\
& \frac{(\varPsi'_S)_{i,j,k+1} + (\varPsi'_S)^{k,l}_{i,j,k}}{2}\left(u^{k,l}_{i,j,k+1} - u^{k,l}_{i,j,k}\right) + \frac{(\varPsi'_S)_{i,j,k-1} + (\varPsi'_S)^{k,l}_{i,j,k}}{2}\left(u^{k,l}_{i,j,k-1} - u^{k,l}_{i,j,k}\right).
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
div\_du :=& \frac{(\Psi'_S)_{i+1,j,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}du^{k,l+1}_{i+1,j,k} + \frac{(\Psi'_S)_{i-1,j,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}du^{k,l+1}_{i-1,j,k} + \\
& \frac{(\Psi'_S)_{i,j+1,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}du^{k,l+1}_{i,j+1,k} + \frac{(\Psi'_S)_{i,j-1,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2}du^{k,l+1}_{i,j-1,k} + \\
& \frac{(\varPsi'_S)_{i,j,k+1} + (\varPsi'_S)^{k,l}_{i,j,k}}{2}du^{k,l+1}_{i,j,k+1} + \frac{(\varPsi'_S)_{i,j,k-1} + (\varPsi'_S)^{k,l}_{i,j,k}}{2}du^{k,l+1}_{i,j,k-1}.
\end{aligned} \tag{9}
$$

$$
\begin{aligned}
div\_d :=& \frac{(\Psi'_S)_{i+1,j,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2} + \frac{(\Psi'_S)_{i-1,j,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2} + \\
& \frac{(\Psi'_S)_{i,j+1,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2} + \frac{(\Psi'_S)_{i,j-1,k} + (\Psi'_S)^{k,l}_{i,j,k}}{2} + \\
& \frac{(\varPsi'_S)_{i,j,k+1} + (\varPsi'_S)^{k,l}_{i,j,k}}{2} + \frac{(\varPsi'_S)_{i,j,k-1} + (\varPsi'_S)^{k,l}_{i,j,k}}{2}.
\end{aligned} \tag{10}
$$

These expressions are the same for the other component of the optical flow, changing u by v. The last two terms of equations (8), (9) and (10), in italics, correspond to the temporal regularization of the optical flow. This is implemented in the Temporal method and removed in the Spatial one. The finite difference scheme in the temporal method is computed using information from the previous, $k-1$, and following, $k+1$, frames. If we define $\mathbf{y} = \mathbf{x} + \mathbf{w}^{k,l}$ and separate the parts of the equation

that remain constant during the SOR iterations, we may define the following variables:

$$
\begin{aligned}
Au :=&- (\Psi'_D)^{k,l} \left(I(\mathbf{y}) - I(\mathbf{x})\right) I_x(\mathbf{y}) + \alpha \, div\_u, \\
&- \gamma \, (\Psi'_G)^{k,l} \left((I_x(\mathbf{y}) - I_x(\mathbf{x})) I_{xx}(\mathbf{y}) + (I_y(\mathbf{y}) - I_y(\mathbf{x})) I_{xy}(\mathbf{y})\right), \\
Av :=&- (\Psi'_D)^{k,l} \left(I(\mathbf{y}) - I(\mathbf{x})\right) I_y(\mathbf{y}) + \alpha \, div\_v \\
&- \gamma \, (\Psi'_G)^{k,l} \left((I_x(\mathbf{y}) - I_x(\mathbf{x})) I_{xy}(\mathbf{y}) + (I_y(\mathbf{y}) - I_y(\mathbf{x})) I_{yy}(\mathbf{y})\right), \\
Du :=&(\Psi'_D)^{k,l} I_x^2(\mathbf{y}) + \gamma \, (\Psi'_G)^{k,l} \left(I_{xx}^2(\mathbf{y}) + I_{xy}^2(\mathbf{y})\right) + \alpha \, div\_d, \\
Dv :=&(\Psi'_D)^{k,l} I_y^2(\mathbf{y}) + \gamma \, (\Psi'_G)^{k,l} \left(I_{yy}^2(\mathbf{y}) + I_{xy}^2(\mathbf{y})\right) + \alpha \, div\_d, \\
D :=&(\Psi'_D)^{k,l} I_x(\mathbf{y}) I_y(\mathbf{y}) + \gamma \, (\Psi'_G)^{k,l} \left(I_{xx}(\mathbf{y}) + I_{yy}(\mathbf{y})\right) I_{xy}(\mathbf{y}).
\end{aligned}
\tag{11}
$$

In order to compute expressions like $I(\mathbf{x} + \mathbf{w}^{k,l})$, we use bicubic interpolation. Putting all together, we arrive to the SOR scheme, which is given by

$$
\begin{aligned}
du^{k,l,s+1} :=&\frac{(1-w)\, du^{k,l,s} + w \left(Au - D \cdot dv^{k,l,s+1} + \alpha \, div\_du\right)}{Du}, \\
dv^{k,l,s+1} :=&\frac{(1-w)\, dv^{k,l,s} + w \left(Av - D \cdot du^{k,l,s+1} + \alpha \, div\_dv\right)}{Dv},
\end{aligned}
\tag{12}
$$

with $w \in (0, 2)$ the SOR relaxation parameter. In our implementation, we choose $w = 1.9$ by default.

This numerical approximation is calculated until the method converges to a steady state solution or it exceeds a maximum number of iterations. The stopping criterion is

$$
\frac{1}{N} \sum_{i,j,k} \left(du_{i,j,k}^{s+1} - du_{i,j,k}^s\right)^2 + \left(dv_{i,j,k}^{s+1} - dv_{i,j,k}^s\right)^2 < \varepsilon^2,
\tag{13}
$$

with N the number of pixels in all frames and ε the stopping criterion threshold. The iterative process stops when condition (13) is true or a maximum number of iterations is reached. This is different from the original article [3], where a fixed number of SOR iterations is used. Once it has converged, we go to the next inner iteration, $l + 1$, and restart the variables in (11).

In our implementation, we make use of the OpenMP library to enable multiple threads during the computations. This allows us to take advantage of multiple cores, accelerating the run time of the algorithm. Nevertheless, due to the nature of the SOR scheme, it is probable that the results are slightly different, even when the process is launched with the same number of processors.

3 Pyramidal Structure

In order to estimate large displacements, we embed the optical flow method in a pyramidal structure. We follow the same strategy presented in a previous IPOL article [6] and reproduce here the basic ideas.

Our algorithm creates a pyramid of down-sampled 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

$$
I^s(\eta \mathbf{x}) := G_\sigma * I^{s-1}(\mathbf{x}).
\tag{14}
$$

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

$$
\sigma(\eta) := \sigma_0 \sqrt{\eta^{-2} - 1}, \; with \; \sigma_0 := 0.6.
\tag{15}
$$

Then, starting at the coarsest scale, the system of equations is solved in each scale to get successive approximations of the optical flow. Every intermediate solution is used as initialization in the following scale. To transfer the values from a coarser scale, the flow field is updated as

$$
\begin{aligned}
u^{s-1}(\mathbf{x}) &:= \frac{1}{\eta} u^s(\eta \mathbf{x}) \\
v^{s-1}(\mathbf{x}) &:= \frac{1}{\eta} v^s(\eta \mathbf{x})
\end{aligned}
\tag{16}
$$

4 Parameters of the Method

This method depends on the parameters given in table 1. These parameters are: α and γ, which define the smoothness and the preservation of gradient structures in the optical flow, respectively; the parameters for the pyramidal scheme, N_{scales} and η, that stand for the number of scales and the downsampling factor; and the parameters for the numerical scheme, composed of the inner and outer iterations, and the stopping criterion threshold (ε).

Table 1: Parameters of the method

Parameter	Explanation
α	Regularization parameter. It determines the smoothness of the output. The bigger this parameter is, the smoother the solutions we obtain.
γ	Parameter associated with the gradient constancy term in equation (3).
N_{scales}	Number of scales in the pyramidal structure. 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 (see the previous article by Meinhardt-Llopis and Sánchez [6] for more details).
η	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.
ε	Stopping criterion threshold. It is the threshold used to stop the SOR iterations, given in equation (13).
$inner\_iterations$	Number of inner iterations in the numerical scheme. It corresponds to index l in equation (12).
$outer\_iterations$	Number of outer iterations in the numerical scheme. It corresponds to index k in equation (12).

5 Algorithm

Next, we describe the algorithm that implements the numerical scheme in equation (12). The algorithm takes a set of gray level images as input data and computes the optical flows between every pair of consecutive images. The number of calculated optical flows is one less than the number of input images. We separate the algorithm in two modules: one procedure that computes the optical flows in each scale, and the main algorithm that is in charge of handling the pyramidal structure.

In the procedure, $MAXITER$ is the maximum number of iterations allowed for the convergence of the SOR method. Its value is constant and is high enough to let the method converge. In the

---

Procedure `brox_optic_flow`$(I, u, v, \alpha, \gamma, \varepsilon, inner\_iterations, outer\_iterations)$

---

1 Compute I_x, I_y

2 Compute I_{xx}, I_{yy}, I_{xy}

3 **for** $no \leftarrow 0$ **to** $outer\_iterations - 1$ **do**

4   Compute $I(\mathbf{x} + \mathbf{w}), I_x(\mathbf{x} + \mathbf{w}), I_y(\mathbf{x} + \mathbf{w})$ using bicubic interpolation

5   Compute $I_{xx}(\mathbf{x} + \mathbf{w}), I_{xy}(\mathbf{x} + \mathbf{w}), I_{yy}(\mathbf{x} + \mathbf{w})$ using bicubic interpolation

6   Compute u_x, u_y, v_x, v_y

7   Compute Ψ'_S using equation (5)

8   Compute div_u, div_v, div_d using equations (8) and (10)

9 $du \leftarrow 0$

10 $dv \leftarrow 0$

11   **for** $ni \leftarrow 0$ **to** $inner\_iterations - 1$ **do**

12   Compute Ψ'_D, Ψ'_G using equation (5)

13   Compute Au, Av, Du, Dv, D using equation (11)

14   **while** $error > \varepsilon$ **and** $nsor < MAXITER$ **do**

15 $du \leftarrow (1 - \omega) du + \omega (Au - D\, dv + \alpha\, div\_du) / Du$

16 $dv \leftarrow (1 - \omega) dv + \omega (Av - D\, du + \alpha\, div\_dv) / Dv$

17   Compute $error$ with equation (13)

18 $nsor \leftarrow nsor + 1$

19   **end**

20   **end**

21 $u \leftarrow u + du$

22 $v \leftarrow v + dv$

23 **end**

Algorithm 2: Pyramidal structure management

     Input: $I, u, v, \alpha, \gamma, N_{scales}, \eta, \varepsilon, inner\_iterations, outer\_iterations$

     Output: u, v

1 Normalize images between 0 and 255

2 Convolve the images with a Gaussian of $\sigma = 0.8$

3 Create the pyramid of images I^s using η (with $s = 0, \ldots, N_{scales} - 1$)

4 **for** $s \leftarrow N_{scales} - 1$ **to** 0 **do**

5   `brox_optic_flow`$(I, u^s, v^s, \alpha, \gamma, inner\_iterations, outer\_iterations)$

6   **if** $s > 0$ **then**

7 $u^{s-1}(\mathbf{x}) := \frac{1}{\eta} u^s(\eta \mathbf{x})$

8 $v^{s-1}(\mathbf{x}) := \frac{1}{\eta} v^s(\eta \mathbf{x})$

9   **end**

10 **end**

---

source code, the SOR loop is unrolled in order to avoid boundary tests when computing $div(du^{k+1})$ and $div(dv^{k+1})$. This means that the first and last columns and rows, and the four corners of the images, are computed separately.

The procedure for the spatial and temporal methods is basically the same, except for the computation of the divergence terms. They differ in the calculation of variables Ψ'_S, div_u, div_v and div_d.

In theory, as we have seen before, the method needs three levels of iterations. Nevertheless, we have found in the experiments that the inner iterations can be integrated in the outer iterations without a loss of precision. For this reason we have decided to estimate the values of Ψ'_S, div_u, div_v and div_d inside the outer iterations. We prefer this option in order to avoid the estimation of ∇du. The other computations, at the beginning of the inner loop, may also be integrated in the outer loop.

The main process is given in algorithm 2. In order to turn the method more stable to the input parameters, it first normalizes the images between 0 and 255; it convolves the finest scale images with a small Gaussian kernel; then, it creates the pyramidal structure for the whole sequence; and, finally, it goes over the set of scales computing the optical flows at different resolutions.

6 Experimental Analysis

In this section, we examine the behavior of the method for some standard image sequences. We have used the Yosemite and Yosemite with Clouds sequences, which have also been analyzed by Brox et al. in 2004 [3]. On the other hand, we use the RubberWhale and Urban2 sequences from the Middlebury benchmark database[2] [2]. The results are shown in figure 2 (the color scheme used to represent the orientation and magnitude of optical flows is displayed in figure 1). In these experiments, α and γ are adapted to get the best results for each sequence. The values for the remaining parameters are: $\eta = 0.75$, $\varepsilon = 0.0001$, $inner\_iterations = 1$, $outer\_iterations = 38$, and N_{scales} is automatically calculated so that the coarsest scale works with images around 16×16 pixels.
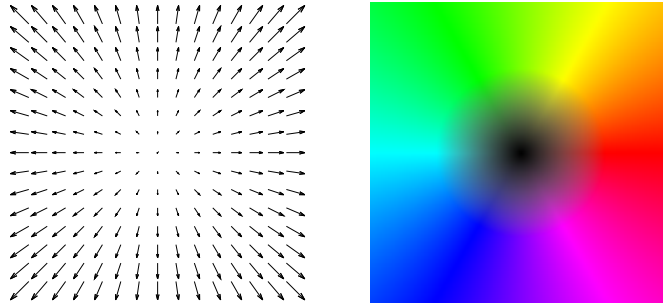
Figure 1: Color scheme used to represent the orientation and magnitude of optical flows.

In general, the method respects the motion discontinuities and creates piecewise continuous flow fields. This is due to the L^1 functionals and the TV regularization scheme. In the Yosemite with Clouds sequence, the sky motion is translational (2 pixels to the right). Although there are illumination changes in the clouds, the solution obtained by the method is very accurate. This is due to the gradient constancy term. Note that γ is larger than in Yosemite. This shows that the method can correctly handle constant brightness shifts, provided that the gradient does not vary.

Tables 2 and 3 show the Average Angular Error (AAE) and Average End-point Error (EPE) for the Spatial and Temporal methods, respectively. The AAE and EPE are calculated as in the
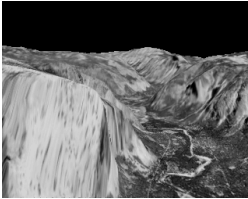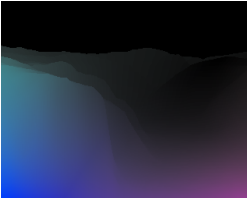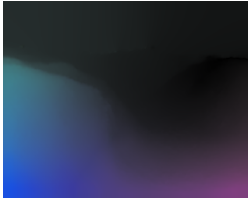
---

[2] http://vision.middlebury.edu/flow/
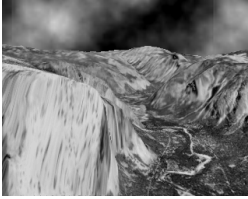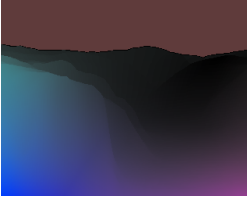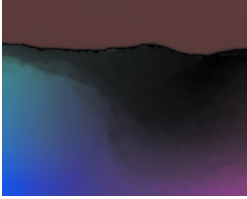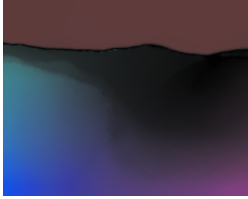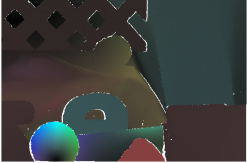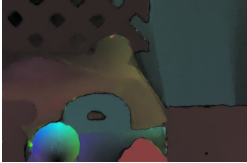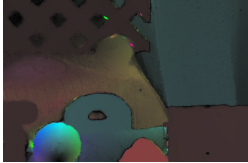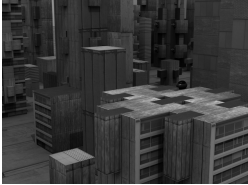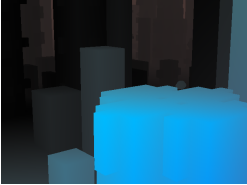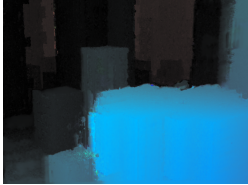
Figure 2: Results for the Yosemite, Yosemite with Clouds, RubberWhale and Urban2 sequences. First column shows frame 6 for Yosemite and Yosemite with Clouds, and frame 10 for RubberWhale and Urban2. Second column shows the corresponding ground truth optical flows. Third and fourth columns show the results for the Spatial and Temporal methods, respectively.
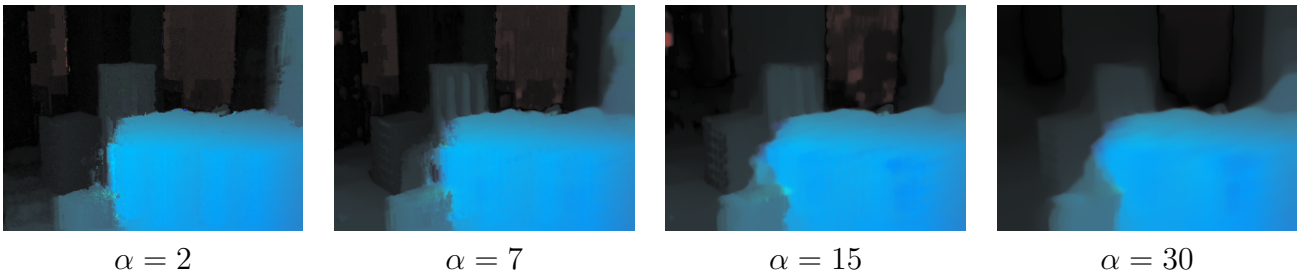

Figure 3: Results for the Urban2 sequence using the temporal method.

Middlebury benchmarks [2]. We observe that the results are similar to the results presented in the original article [3] for the Yosemite sequences. In these cases, the temporal method provides better results than the spatial method. The flow in these sequences is very continuous, so it clearly benefits from the spatio-temporal continuous regularization term.

Table 2: AAE and EPE for the Spatial method (third column of figure 2).

Sequence	α	γ	AAE	EPE
Yosemite	50	2	1.587^o	0.075
Yosemite with Clouds	145	15	2.367^o	0.101
RubberWhale	185	60	3.467^o	0.103
Urban2	30	2	2.803^o	0.395

Table 3: AAE and EPE for the Temporal method (fourth column of figure 2).

Sequence	α	γ	AAE	EPE
Yosemite	27	2	1.297^o	0.061
Yosemite with Clouds	93	15	1.927^o	0.079
RubberWhale	91	60	4.798^o	0.152
Urban2	2	2	5.823^o	0.560

On the other hand, we observe that the results for RubberWhale and Urban2, in the temporal method, are worse than in the spatial method. Although the RubberWhale sequence presents piecewise continuous motion fields, it contains many flow discontinuities and different motion directions. In the case of Urban2, the maximum motion is about 22, so the continuous temporal regularization strongly deteriorates the results. Figure 3 shows different results for Urban2, using different α values with the temporal method. We observe that the temporal method strongly deteriorates the discontinuities of the optical flows. This shows the negative effect of the continuous temporal smoothing scheme: it is necessary to use a very small α to obtain results similar to the spatial method. In these cases, it is better to use a nonlinear temporal smoothing scheme, like in [8].

Next, we evaluate the evolution of the AAE and EPE with respect to α. We compare the spatial and temporal methods in figure 4, using the same parameters as in tables 2 and 3, and let α vary. In the Yosemite sequences (top row), the temporal method improves the results of the spatial method: the AAE and EPE attain smaller errors for smaller values of α. This comes from the fact that the regularization is more important with the temporal scheme, thus a smaller value of α is needed for the same type of smoothing. We also note that the temporal results degrade faster for increasing values of α. It crosses the spatial curve and then diverges. This may occur because the temporal regularization tends to spoil faster the flow discontinuities.

Another interesting behavior to note about the best AAE and best EPE is that they are obtained at different values of α for the same sequence. The best AAE is obtained with a larger α. This is justified because a larger α creates smoother flows, which may be better aligned with the directions of the ground truth, but with smaller magnitudes. In the RubberWhale and Urban2 sequences (bottom row of figure 4), we observe that the temporal method always provides worse results than the spatial method. In these sequences the presence of flow discontinuities and large displacements is more important, therefore the continuous temporal scheme is not suitable in these cases. Note that the temporal curve for Urban2 depicts a strong deterioration for increasing values of α.

In figure 5, we compare the spatial and temporal methods frame by frame. Both graphics depict the AAE and EPE in every frame for the Yosemite with Clouds sequence. Most of the optical flow
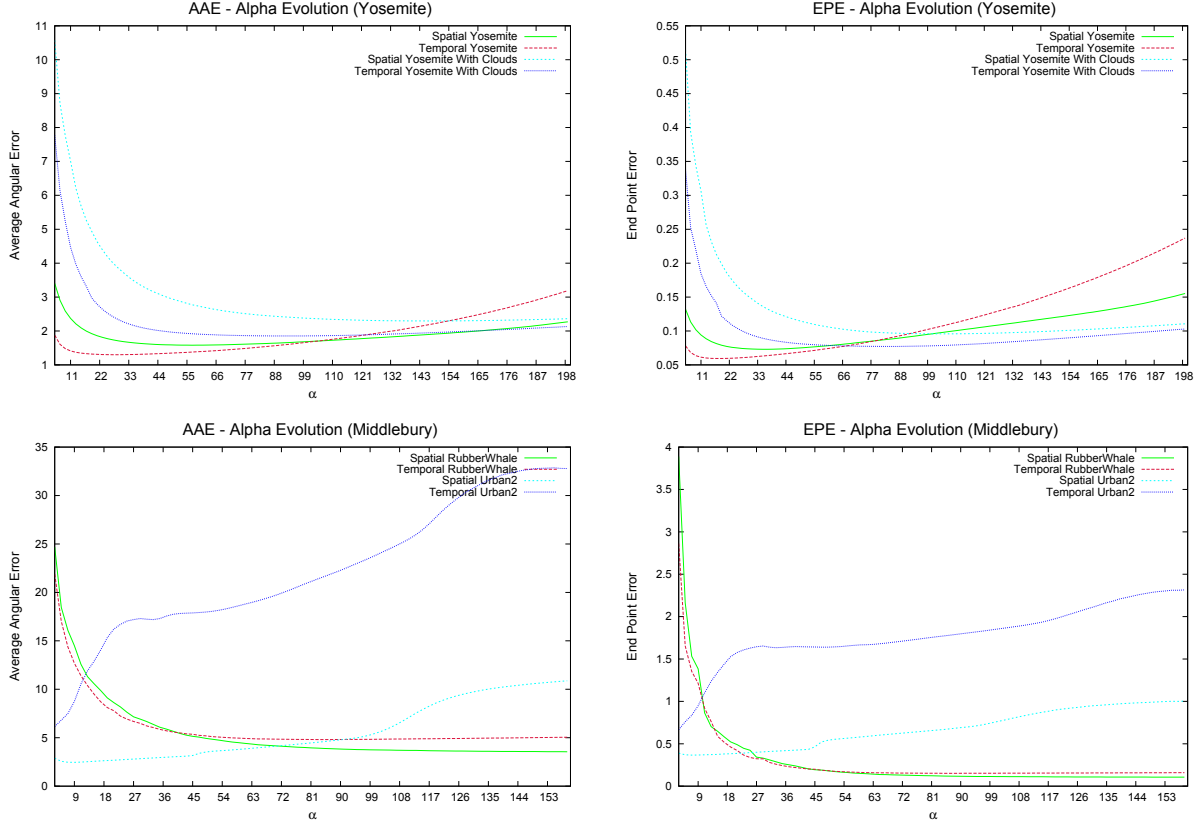
Figure 4: First row, the AAE and EPE for Yosemite and Yosemite with Clouds sequences; second row, the AAE and EPE for RubberWhale and Urban2.
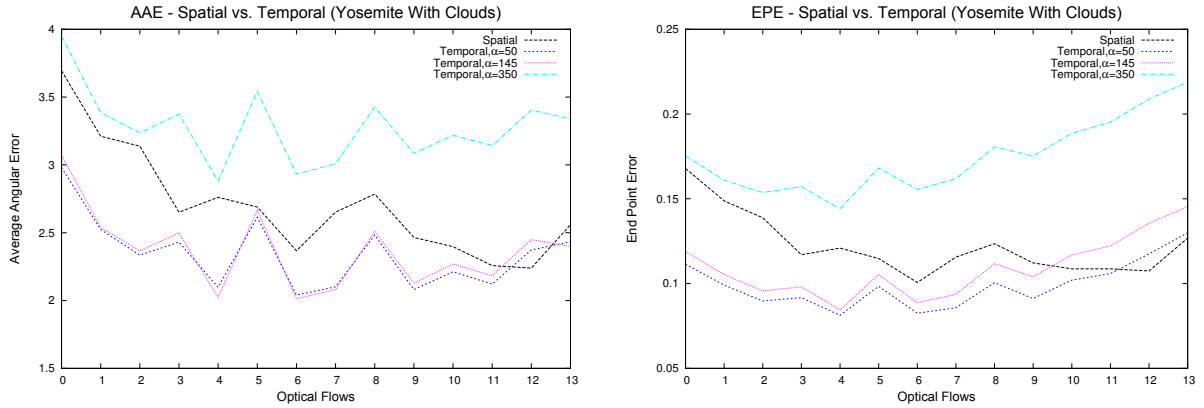

Figure 5: Comparison of the spatial and temporal methods: AAE (left) and EPE (right) for the Yosemite with Clouds sequence in every frame.

errors in the temporal method remain below the spatial errors. When α is big, $\alpha = 350$, the errors in the temporal method exceed the spatial ones.

We now study the behavior of the method with respect to the η parameter. Using the best α and γ values from tables 2 and 3, we show the error evolution with respect to different values of η. This is shown in table 4. Note that increasing the value of η and N_{scales}, the AAE and EPE do not significantly improve.

Table 4: AAE and EPE results for different values of η and N_{scales}

Sequence	η	N_{scales}	Spatial		Temporal	
			AAE	EPE	AAE	EPE
	0.1	2	1.602^o	0.076	1.314^o	0.062
	0.25	2	1.605^o	0.076	1.310^o	0.062
	0.5	4	1.594^o	0.075	1.303^o	0.061
Yosemite	0.65	7	1.589^o	0.075	1.300^o	0.061
	0.75	10	1.586^o	0.075	1.297^o	0.061
	0.85	17	1.586^o	0.075	1.297^o	0.061
	0.95	54	1.586^o	0.075	1.297^o	0.061
	0.1	2	2.461^o	0.101	2.182^o	0.091
	0.25	2	2.382^o	0.099	2.017^o	0.083
	0.5	4	2.372^o	0.101	1.944^o	0.080
Yosemite with Clouds	0.65	5	2.359^o	0.100	1.933^o	0.079
	0.75	10	2.367^o	0.100	1.927^o	0.079
	0.85	17	2.379^o	0.101	1.922^o	0.079
	0.95	54	2.393^o	0.103	1.931^o	0.079
	0.1	2	3.779^o	0.117	5.725^o	0.185
	0.25	3	3.616^o	0.109	5.091^o	0.162
	0.5	5	3.491^o	0.104	4.919^o	0.155
RubberWhale	0.65	8	3.491^o	0.104	4.847^o	0.153
	0.75	12	3.467^o	0.103	4.798^o	0.152
	0.85	20	3.458^o	0.103	4.831^o	0.154
	0.95	63	3.426^o	0.102	4.840^o	0.159
	0.1	2	3.316^o	0.570	7.592^o	0.910
	0.25	3	2.873^o	0.426	6.110^o	0.698
	0.5	5	2.836^o	0.408	5.787^o	0.614
Urban2	0.65	8	2.823^o	0.392	5.931^o	0.603
	0.75	12	2.803^o	0.395	5.823^o	0.560
	0.85	21	2.790^o	0.385	6.043^o	0.626
	0.95	67	2.784^o	0.377	6.170^o	0.623

The Yosemite sequences are hardly improved with η. Even for values of $\eta = 0.1$ the results are satisfactory. This means that only using two scales, and let the coarsest scale start with a very small image size, the method can find a very good solution. This is different to the results presented in the original article [3], where the best solutions were obtained for $\eta = 0.95$, with a large number of scales and a lot of inner and outer iterations. In that work, the differences with respect to the downsampling factor were important. The RubberWhale and Urban2 sequences present a similar behavior, but the results are more noticeable for $\eta = 0.1$ and $\eta = 0.25$.

In figures 6, 7 and 8, we show the evolution of AAE with respect to α and γ. Note that the graphic is very stable for a large range of values, especially for Yosemite with Clouds and RubberWhale. We

do not include Yosemite's graphic because it is very similar to Yosemite with Clouds. The red curve shows the best error values for every pair (α, γ). This curve approximates a straight line in the $\alpha - \gamma$ plane. For instance, in the Yosemite sequence the relation is $\alpha \simeq 20\gamma$, in Yosemite with Clouds is $\alpha \simeq 10\gamma$, in RubberWhale $\alpha \simeq 3\gamma$ and in Urban2 $\alpha \simeq 11\gamma$. If we increase the values of these parameters respecting these relations, the errors still remain low.
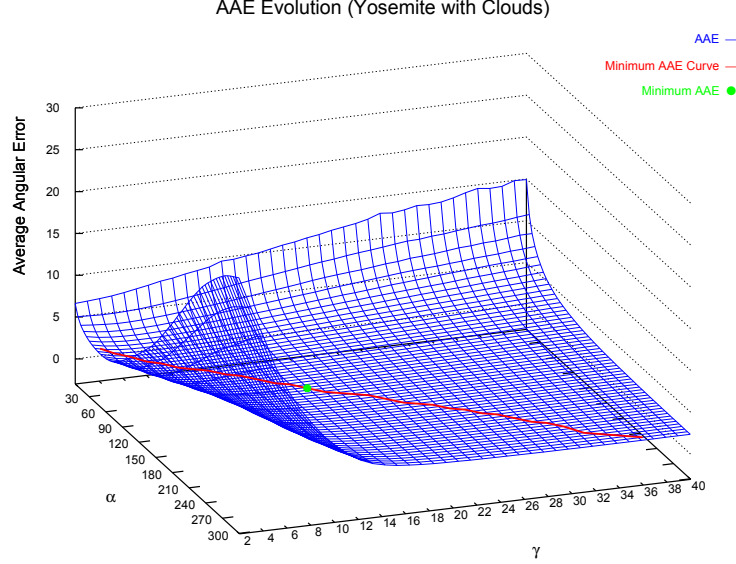
Figure 6: Yosemite with Clouds. Evolution of AAE with respect to α and γ.
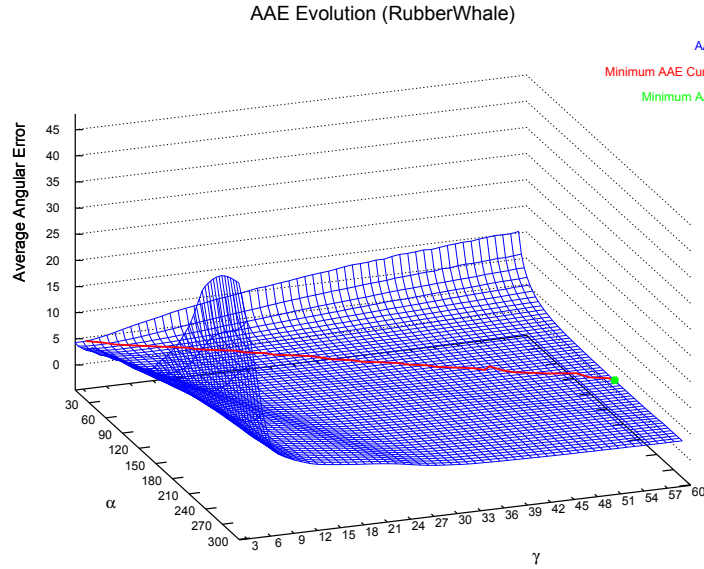
Figure 7: RubberWhale. Evolution of AAE with respect to α and γ.

Finally, a green dot represents the minimum AAE in these graphics. It is interesting to note that in the RubberWhale sequence we get slightly better results for even higher values of α and γ, although the improvements are hardly appreciable.

In figures 9, 10 and 11 we show the AAE evolution with respect to the *inner_iterations* and *outer_iterations* parameters. These graphics show that the method is very stable and converges very
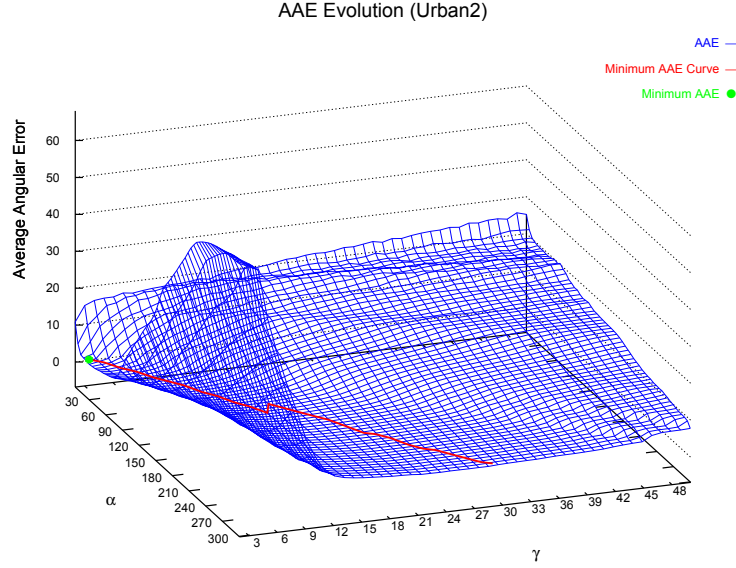
Figure 8: Urban2. Evolution of AAE with respect to α and γ.

quickly to the final solution. We also show several color points for the minimum AAE and the values with an error below 0.5%, 1% and 3% with respect to the minimum. The last three values were calculated for the smallest $inner\_iterations \times outer\_iterations$ relation, i.e., the minimum number of total iterations.
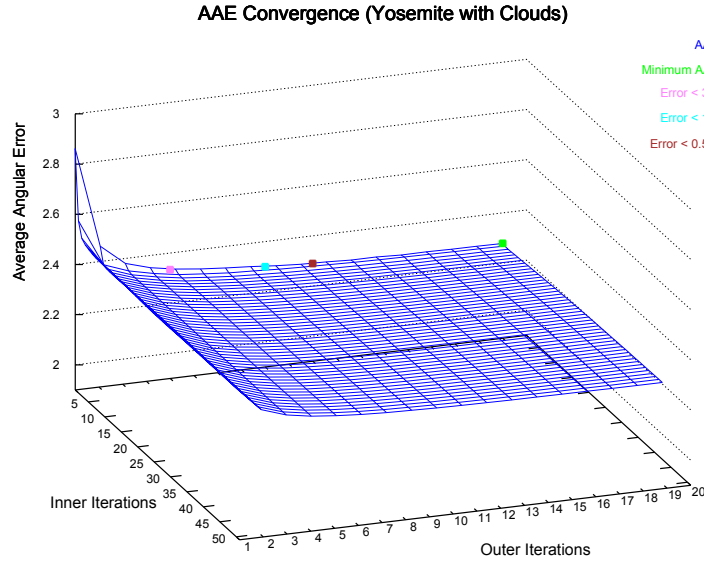

Figure 9: Yosemite with Clouds. Evolution of AAE with respect to $inner\_iterations$ and $outer\_iterations$.

After these graphics, we may conclude that the $outer\_iterations$ parameter is more relevant in the convergence of the method. Normally, for $inner\_iterations = 1$ we obtain very good accuracies; therefore, we may suppose that the inner iterations in these experiments can be integrated in the outer iterations without a loss of precision.
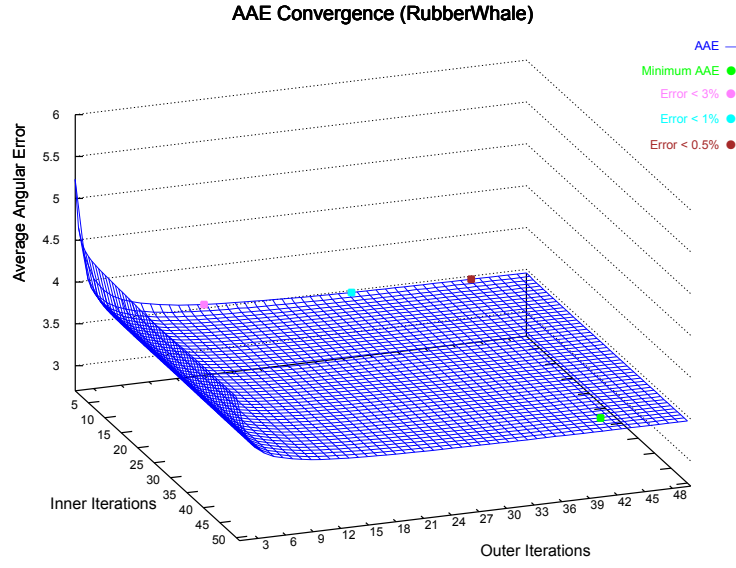
265

Figure 10: RubberWhale. Evolution of AAE with respect to *inner_iterations* and *outer_iterations*.
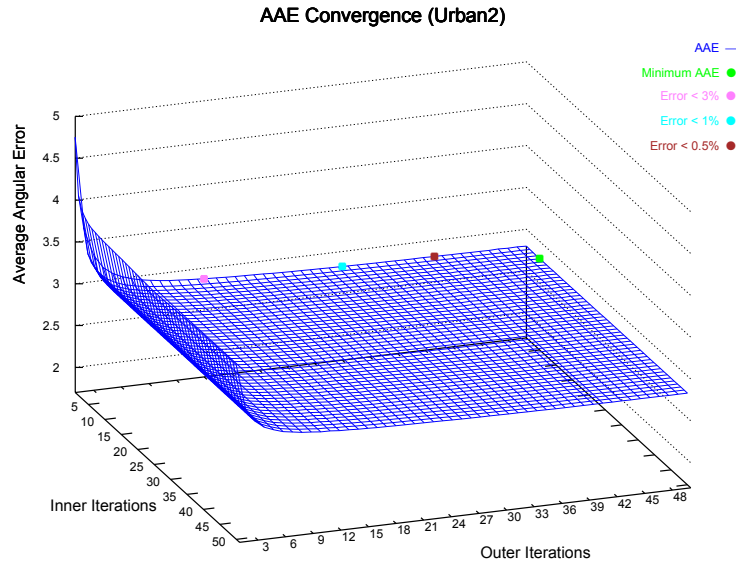

Figure 11: Urban2. Evolution of AAE with respect to *inner_iterations* and *outer_iterations*.

7 Examples

In this section, we show the results for the sequences in the Middlebury benchmark database [2]. Figure 12 depicts the results for all the tests sequences in the database, except Dimetrodon and Venus for the temporal method. This is because there are not enough images to use the method properly. Table 5 show the AAE and EPE obtained for $\alpha = 18$ and $\gamma = 7$ in the spatial method, and $\alpha = 2.5$, $\gamma = 2$ in the temporal method. In both cases, we have set the following parameters: $\eta = 0.75$, $\varepsilon = 0.0001$, *inner_iterations* = 1 and *outer_iterations* = 15. N_{scales} is automatically calculated so that the coarsest scale works with images around 16 x 16 pixels.

Table 5: AAE and EPE for the Middlebury test sequences.

Error	Grove2	Grove3	Hydrangea	RubberWhale	Urban2	Urban3	Dimetrodon	Venus
Spat. AAE	2.455^o	6.481^o	2.442^o	3.696^o	2.561^o	4.804^o	1.663^o	4.599^o
Spat. EPE	0.174	0.693	0.200	0.111	0.368	0.544	0.086	0.292
Temp. AAE	2.569^o	7.031^o	4.468^o	5.435^o	5.90^o3	6.681^o	-	-
Temp. EPE	0.184	0.796	0.346	0.168	0.628	0.784	-	-

Finally, in figure 13 we show the results for the evaluation sequences using the same parameter configuration (spatial method).

8 Video

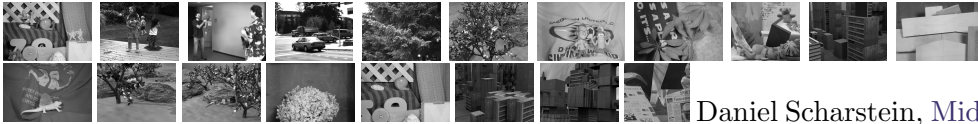This is an example of applying the spatial optical flow method, frame by frame, to a video: Karl-Wilhelm-Straβe[3] (21,1 MB). The original video of this traffic sequence can be found at the Institut für Algorithmen und Kognitive Systeme[4] web pages.

Acknowledgements

Image Credits

All images by the authors except:

 Daniel Scharstein, Middlebury benchmark database[6].

 Lynn Quam.

---

[3]https://doi.org/10.5201/ipol.2013.21
[4]http://i21www.ira.uka.de/image_sequences/
[5]http://www.ctim.es/projects/opticalflow2012/
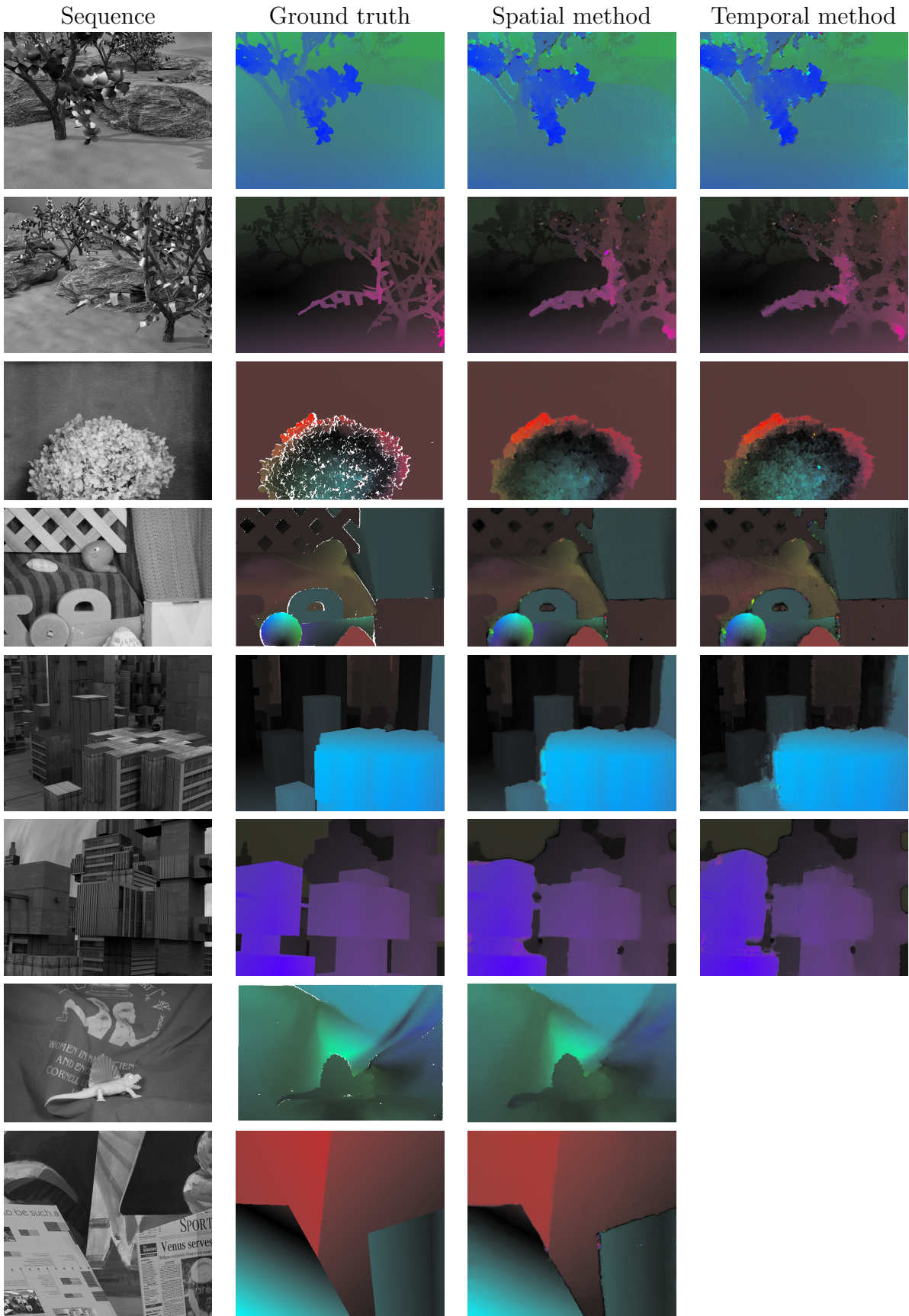[6]http://vision.middlebury.edu/flow/data/

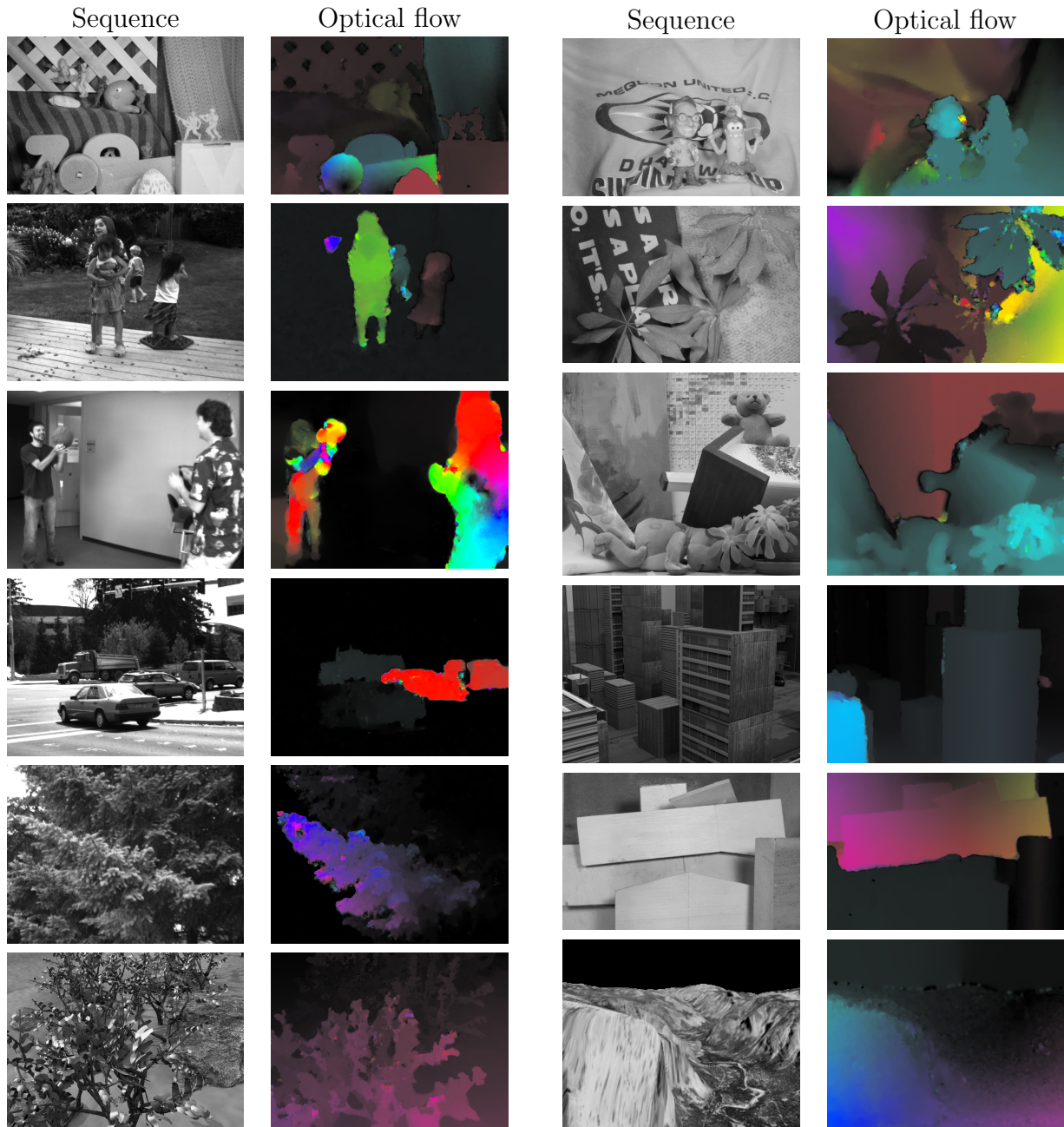Figure 12: Results for the Middlebury test sequences.

Figure 13: Results for the Middlebury evaluation sequences.

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] 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.

[4] Andrés Bruhn and Joachim Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 749–755, Washington, DC, USA, October 2005. IEEE Computer Society. http://dx.doi.org/10.1109/ICCV.2005.240.

[5] 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(81)90024-2.

[6] Enric Meinhardt-Llopis, Javier Sánchez Pérez, and Daniel Kondermann. Horn-Schunck Optical Flow with a Multi-Scale Strategy. *Image Processing On Line*, 2013:151–172, 2013. http://dx.doi.org/10.5201/ipol.2013.20.

[7] E. Mémin and P. Pérez. 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.

[8] Agustín Salgado and Javier Sánchez. A temporal regularizer for large optical flow estimation. In *IEEE International Conference on Image Processing ICIP*, pages 1233–1236, 2006. http://dx.doi.org/10.1109/ICIP.2006.312548.

[9] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. TV-L1 Optical Flow Estimation. *Image Processing On Line*, 2013:137–150, 2013. http://dx.doi.org/10.5201/ipol.2013.26.

[10] Andreas Wedel, Daniel Cremers, Thomas Pock, and Horst Bischof. Structure- and motion-adaptive regularization for high accuracy optic flow. In *IEEE International Conference on Computer Vision*, pages 1663–1668, September 2009.

[11] 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.