IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2013–07–19.
Submitted on 2012–07–05, accepted on 2012–10–29.
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.2013.26

TV-L1 Optical Flow Estimation

Javier Sánchez[1], Enric Meinhardt-Llopis[2], Gabriele Facciolo[3]

[1] CTIM, Universidad de Las Palmas de Gran Canaria, Spain (jsanchez@dis.ulpgc.es)
[2] CMLA, ENS Cachan, France (enric.meinhardt@cmla.ens-cachan.fr)
[3] CMLA, ENS Cachan, France (gabriele.facciolo@cmla.ens-cachan.fr)

Abstract

This article describes an implementation of the optical flow estimation method introduced by Zach, Pock and Bischof in 2007. This method is based on the minimization of a functional containing a data term using the L^1 norm and a regularization term using the total variation of the flow. The main feature of this formulation is that it allows discontinuities in the flow field, while being more robust to noise than the classical approach by Horn and Schunck. The algorithm is an efficient numerical scheme, which solves a relaxed version of the problem by alternate minimization.

Source Code

A C implementation of this algorithm is provided. The source code and an online demo are accessible at the web page of this article[1].

Keywords: optical flow, total variation

1 Introduction

The method described in Zach, Pock and Bischof's article [6] is based on the brightness constancy assumption. Let $I(x, y, t)$ be a video sequence, and let $(x(t), y(t))$ be the trajectory of a point in the image plane, then the brightness constancy assumption states that $I(x(t), y(t), t)$ is constant:

$$\frac{d}{dt} I(x(t), y(t), t) = 0. \tag{1}$$

Applying the chain rule,

$$\nabla I \cdot (\dot{x}, \dot{y}) + \frac{\partial}{\partial t} I = 0. \tag{2}$$

This last identity must hold for the trajectories of every point in the image domain, whose velocities at one instant define a vector field $\mathbf{u}(x, y) = (u_1(x, y), u_2(x, y))$. Thus, the vector field

[1] http://dx.doi.org/10.5201/ipol.2013.26

$\mathbf{u}(x, y)$ satisfies pointwise the following linear condition, called the *optical flow constraint equation* [2]:

$$\nabla I \cdot \mathbf{u} + \frac{\partial}{\partial t} I = 0. \tag{3}$$

For every point in the image domain, the condition $\nabla I \cdot \mathbf{u} + \frac{\partial}{\partial t} I = 0$ is a linear equation in two variables (the components of $\mathbf{u}$). Thus, there are altogether twice as many variables as equations, and the resulting linear system is underdetermined. A standard way to solve underdetermined systems is to add a *smoothness condition*, that forces $\mathbf{u}$ to be regular in some sense. The proposal of Horn and Schunck [2] was to select the $\mathbf{u}$ that minimizes the following functional:

$$E_{\texttt{Horn\_Schunck}}(\mathbf{u}) = \int_\Omega \left(\nabla I \cdot \mathbf{u} + \frac{\partial}{\partial t} I \right)^2 + \alpha \left(|\nabla u_1|^2 + |\nabla u_2|^2 \right). \tag{4}$$

This minimization problem is easy to solve by standard methods, and the resulting flow estimations are good enough for many purposes. The main shortcoming of the $|\nabla u_1|^2 + |\nabla u_2|^2$ term is that it penalizes high gradients of $\mathbf{u}$ and it effectively disallows discontinuities. Equation (3) is suitable if the image data is continuous in time. Typically, this equation is replaced by the non-linear formulation, $I_1(\mathbf{x} + \mathbf{u}) - I_0(\mathbf{x}) = 0$, to account for general image sequences. The non-linear term $I_1(\mathbf{x} + \mathbf{u})$ can be linearized using Taylor expansions, yielding the following equation:

$$\rho(\mathbf{u}) = \nabla I_1(\mathbf{x} + \mathbf{u}^0) \cdot (\mathbf{u} - \mathbf{u}^0) + I_1(\mathbf{x} + \mathbf{u}^0) - I_0(\mathbf{x}) = 0, \tag{5}$$

with $\mathbf{u}^0$ a close approximation to $\mathbf{u}$. The Horn–Schunck functional can be modified to allow discontinuities in the flow field by changing the quadratic factors, and this results in the method described here. The proposed algorithm can be understood as a minimization of the following energy functional, which is the sum of the total variation of $\mathbf{u}$ and an L^1 attachment term:

$$E(\mathbf{u}) = \int_\Omega |\nabla u_1| + |\nabla u_2| + \lambda \left| \rho(\mathbf{u}) \right|. \tag{6}$$

An efficient way to minimize this energy functional is to introduce the following convex relaxation:

$$E_\theta(\mathbf{u}, \mathbf{v}) = \int_\Omega |\nabla u_1| + |\nabla u_2| + \frac{1}{2\theta} |\mathbf{u} - \mathbf{v}|^2 + \lambda \left| \rho(\mathbf{v}) \right|. \tag{7}$$

Setting θ to a very small value forces the minimum of E_θ to occur when $\mathbf{u}$ and $\mathbf{v}$ are nearly equal, reducing to the original energy E, defined in equation (6). The interest of this relaxation is that E_θ can be minimized by alternatively fixing one of $\mathbf{u}$ or $\mathbf{v}$, and solving for the other variable.

1. Fixed $\mathbf{v}$, solve
$$\min_{\mathbf{u}} \int_\Omega |\nabla u_1| + |\nabla u_2| + \frac{1}{2\theta} |\mathbf{u} - \mathbf{v}|^2. \tag{8}$$
2. Fixed $\mathbf{u}$, solve
$$\min_{\mathbf{v}} \int_\Omega \frac{1}{2\theta} |\mathbf{u} - \mathbf{v}|^2 + \lambda \left| \rho(\mathbf{v}) \right|. \tag{9}$$

The first sub-problem fits the total variation denoising model of Rudin–Osher–Fatemi [4], which can be solved by Chambolle's duality-based algorithm [1]. The second sub-problem does not depend on spatial derivatives of $\mathbf{v}$, so it can be solved point-wise by thresholding.

2 Numerical Scheme

The solution to the first minimization problem stated above can be obtained by computing the fixed point of the following iteration over the dual vector fields $\mathbf{p}_1$ and $\mathbf{p}_2$:

$$\mathbf{p}_d^{k+1} := \frac{\mathbf{p}_d^k + \tau/\theta \nabla \left(v_d^{k+1} + \theta \text{div}(\mathbf{p}_d^k)\right)}{1 + \tau/\theta \left|\nabla \left(v_d^{k+1} + \theta \text{div}(\mathbf{p}_d^k)\right)\right|}, d \in \{1, 2\}, \tag{10}$$

and recovering $\mathbf{u}$ as

$$u_d^{k+1} := v_d^{k+1} + \theta \text{div}(\mathbf{p}_d^k), d \in \{1, 2\}. \tag{11}$$

The second minimization problem can be solved as follows:

$$\mathbf{v}^{k+1} := \mathbf{u}^{k+1} + TH(\mathbf{u}^{k+1}, \mathbf{u}^0), \tag{12}$$

with the thresholding operation

$$TH(\mathbf{u}, \mathbf{u}^0) := \begin{cases} \lambda\theta \nabla I_1(\mathbf{x} + \mathbf{u}^0) & \text{if} \rho(\mathbf{u}, \mathbf{u}^0) < -\lambda\theta \left|\nabla I_1(\mathbf{x} + \mathbf{u}^0)\right|^2 \\ -\lambda\theta \nabla I_1(\mathbf{x} + \mathbf{u}^0) & \text{if} \rho(\mathbf{u}, \mathbf{u}^0) > \lambda\theta \left|\nabla I_1(\mathbf{x} + \mathbf{u}^0)\right|^2 \\ -\rho(\mathbf{u}, \mathbf{u}^0) \frac{\nabla I_1(\mathbf{x}+\mathbf{u}^0)}{|\nabla I_1(\mathbf{x}+\mathbf{u}^0)|^2} & \text{if } |\rho(\mathbf{u}, \mathbf{u}^0)| \leq \lambda\theta \left|\nabla I_1(\mathbf{x} + \mathbf{u}^0)\right|^2 \end{cases}. \tag{13}$$

This thresholding operation includes the information of the attachment term. When the objects move beyond the image limits, it is not possible to compute ρ, and it is convenient to disable this thresholding. Over these points, it is better to use only the regularization term.

The input of the algorithm is a pair of images $I_0(\mathbf{x})$ and $I_1(\mathbf{x})$, with $\mathbf{x} = (i, j)$ the pixel index. The output is a vector field $\mathbf{u}(\mathbf{x}) = (u_1(\mathbf{x}), u_2(\mathbf{x}))$. Note that the residual, $\rho(\mathbf{u})$, is a scalar field (i.e., a gray-valued image), and its computation involves a warping of I_1 and ∇I_1 by the deformation $\mathbf{u}^0$. The vector field $\mathbf{u}^0$ must be close to $\mathbf{u}$, so that the approximation error of the Taylor expansions above is small. The approximation field $\mathbf{u}^0$ is effectively computed by a multiscale scheme.

2.1 Numerical Details

There are some numerical issues to be taken into account when implementing the algorithm. For example, it is essential for the difference schemes used to compute the divergence and the gradient to be adjoint linear operators, so that the Stokes theorem holds exactly. Here are the precise choices for this algorithm, for an image of size (N_x, N_y):

- To compute the gradient of the image I_1, we use central differences along each direction, with Neumann boundary conditions.

$$\begin{aligned} \frac{\partial}{\partial x} I_1(i, j) &= \begin{cases} \frac{I_1(i+1,j) - I_1(i-1,j)}{2} & \text{if} 1 < i < N_x \\ 0 & \text{otherwise} \end{cases}, \\ \frac{\partial}{\partial y} I_1(i, j) &= \begin{cases} \frac{I_1(i,j+1) - I_1(i,j-1)}{2} & \text{if} 1 < j < N_y \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \tag{14}$$

- To compute the gradient of each component of the flow $\mathbf{u}$, we use forward differences with Neumann boundary conditions.

$$\begin{aligned} \frac{\partial}{\partial x} u(i, j) &= \begin{cases} u(i+1, j) - u(i, j) & \text{if} 1 \leq i < N_x \\ 0 & \text{if} i = N_x \end{cases}, \\ \frac{\partial}{\partial y} u(i, j) &= \begin{cases} u(i, j+1) - u(i, j) & \text{if} 1 \leq j < N_y \\ 0 & \text{if} j = N_y \end{cases}. \end{aligned} \tag{15}$$

- For computing the divergences of the dual variables $\mathbf{p}$, we use the adjoint of the gradient of $\mathbf{u}$, which corresponds to using backward differences:

$$\begin{aligned}\operatorname{div}(\mathbf{p})(i,j) = &\begin{cases} p_1(i,j) - p_1(i-1,j) & \text{if} 1 < i < N_x \\ p_1(i,j) & \text{if} i = 1 \\ -p_1(i-1,j) & \text{if} i = N_x \end{cases} \\ + &\begin{cases} p_2(i,j) - p_2(i,j-1) & \text{if} 1 < j < N_y \\ p_2(i,j) & \text{if} j = 1 \\ -p_2(i,j-1) & \text{if} j = N_y \end{cases} . \end{aligned} \tag{16}$$

- To warp the image I_1 by a flow field $\mathbf{u}^0$, we evaluate $I_1(\mathbf{x} + \mathbf{u}^0(\mathbf{x}))$ using bicubic interpolation.

3 Algorithm

The algorithm that implements the method can be separated in two modules: a procedure that calculates the optical flow at a given scale, using the above numerical scheme; and a main algorithm that implements the pyramidal scheme and calls the procedure to obtain approximate solutions.

The procedure updates a vector field $\mathbf{u}$ and uses three temporary vector fields $\mathbf{v}$, $\mathbf{p}_1$ and $\mathbf{p}_2$, to perform intermediate computations. The initial value $\mathbf{u}^0$ of $\mathbf{u}$ is given by the enclosing multiscale procedure, and it is zero at the coarsest level.

In order to stop the algorithm before the default number of iterations ($N_{maxiter}$), we use a stopping criterion based on the L^2 distance between consecutive values of $\mathbf{u}$. When this distance is smaller than a given threshold, we assume that the algorithm has already converged. If $\mathbf{u}^k$, $\mathbf{u}^{k+1}$ are successive values of $\mathbf{u}$, the stopping criterion is

$$\frac{1}{N_x N_y} \sum_{i,j} \left(u_1^{k+1}(i,j) - u_1^k(i,j)\right)^2 + \left(u_2^{k+1}(i,j) - u_2^k(i,j)\right)^2 < \varepsilon^2 \tag{17}$$

The procedure detects small displacements, but it fails when the correct magnitude of $\mathbf{u}$ is larger than about one pixel (depending on the smoothness of the image). In practice, to detect displacements larger than one pixel, it is useful to work with downscaled versions of the input images, where the sought for displacements are small enough. Then, the large (and rough) displacements obtained at the downscaled level can be refined at the original scale.

A standard way to organize this process is by means of a pyramid of scales: a set of downscaled versions of the input images. In order to create the pyramid of images, we follow the same strategy as in our article on Horn–Schunck optical flow [3]. To downscale an image, it is first convolved with a Gaussian kernel and then sampled using bicubic interpolation. We use a downsampling factor, $\eta \in (0,1)$, that allows for smoother transitions between the scales.

The algorithm is first run at the coarsest level, and the result is used as starting point in the finer levels. Algorithm 1 handles this pyramidal structure and relies on the previous procedure to estimate the optical flows at different scales.

An improved version of this algorithm is described in an article by Wedel et al. [5], which introduces two filtering steps: a preprocessing of the input images by a structure-texture decomposition; and a median filtering of the optical flow after the warping step, enhancing the regularity of the flow.

Algorithm 1: Pyramidal structure management

Input: $I_0, I_1, \tau, \lambda, \theta, \varepsilon, \eta, N_{maxiter}, N_{warps}, N_{scales}$
Output: $\mathbf{u}$

Normalize images between 0 and 255
Convolve the images with a Gaussian of $\sigma = 0.8$
Create the pyramid of images I^s using η (with $s = 0, \ldots, N_{scales} - 1$)
$\mathbf{u}^{N_{scales}-1} \leftarrow (0, 0)$
for $s \leftarrow N_{scales} - 1$ **to** 0 **do**
    `TV-L`^{1}`_optical_flow`$(I_0, I_1, u^0, \tau, \lambda, \theta, \varepsilon, N_{maxiter}, N_{warps})$
    if $s > 0$ **then**
        $\mathbf{u}^{s-1}(\mathbf{x}) := 2\mathbf{u}^s(\mathbf{x}/\eta)$
    end
end

Procedure `TV-L`^{1}`_optical_flow`$(I_0, I_1, u^0, \tau, \lambda, \theta, \varepsilon, N_{maxiter}, N_{warps})$

$\mathbf{p}_1 \leftarrow (0, 0)$
$\mathbf{p}_2 \leftarrow (0, 0)$
for $w \leftarrow 1$ **to** N_{warps} **do**
    Compute $I_1(\mathbf{x} + \mathbf{u}^0(\mathbf{x})), \nabla I_1(\mathbf{x} + \mathbf{u}^0(\mathbf{x}))$ using bicubic interpolation
    $n \leftarrow 0$
    while $n < N_{maxiter}$ **and** *stopping_criterion* $> \varepsilon$ **do**
        $\mathbf{v} \leftarrow TH(\mathbf{u}, \mathbf{u}^0)$
        $\mathbf{u} \leftarrow \mathbf{v} + \theta \mathrm{div}(\mathbf{p})$
        $\mathbf{p} \leftarrow \frac{\mathbf{p} + \tau/\theta \nabla \mathbf{u}}{1 + \tau/\theta |\nabla \mathbf{u}|}$
        $n \leftarrow n + 1$
    end
end

4 Explanation of the Parameters

In this section we explain the parameters of the method and give reasonable default values. The algorithm depends on six parameters: time step (τ), data attachment weight (λ), tightness (θ), stopping criterion threshold (ε), downsampling factor (η), number of scales (N_{scales}), number of warps (N_{warps}).

- τ is the time step of the numerical scheme. Chambolle shows [1] that the numerical scheme converges for values of $\tau < 0.125$. Empirically, its value can be set to 0.25 for a faster convergence.

- λ is the attachment parameter. This is the most relevant parameter, which determines the smoothness of the output. The smaller this parameter is, the smoother the solutions we obtain. It depends on the range of motions of the images, so its value should be adapted to each image sequence.

- θ is the tightness parameter. It serves as a link between the attachment and the regularization terms. In theory, it should have a small value in order to maintain both parts in correspondence. The method is stable for a large range of values of this parameter.

- 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.
- 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).
- 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. See our article on Horn–Schunck optical flow [3] for further details on this and the η parameters.
- N_{warps} represents the number of times that $I_1(\mathbf{x}+\mathbf{u}^0)$ and $\nabla I_1(\mathbf{x}+\mathbf{u}^0)$ are computed per scale. 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.

Table 1: Parameters of the method.

Parameter	Description	Default value
τ	time step	0.25
λ	data attachment weight	0.15
θ	tightness	0.3
ε	stopping threshold	0.01
η	zoom factor	0.5
N_{scales}	number of scales	5
N_{warps}	number of warps	5

5 Examples

Figures 2 and 3 show the optical flows for the Ettlinger–Tor and the Rheinhafen sequences, respectively. These sequences can be found at http://www.ira.uka.de/image_sequences/. The results obtained are similar to the results in Zach et al. [6]. In these examples we have used the following parameters: τ=0.25, θ=0.5, η = 0.5, ε=0.01, 5 scales and 5 warpings. The algorithm is executed for three values of λ.

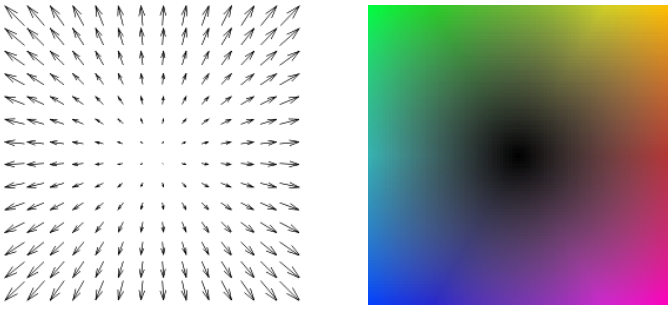


Figure 1: Color scheme used to represent the orientation and magnitude of optical flows.

The method detects the displacement of the objects in the Ettlinger–Tor traffic scene. It also detects a regular background motion maybe due to a small displacement of the camera and the effect of noise. For small values of λ (λ=0.03), the solution is smoother and the optical flow is

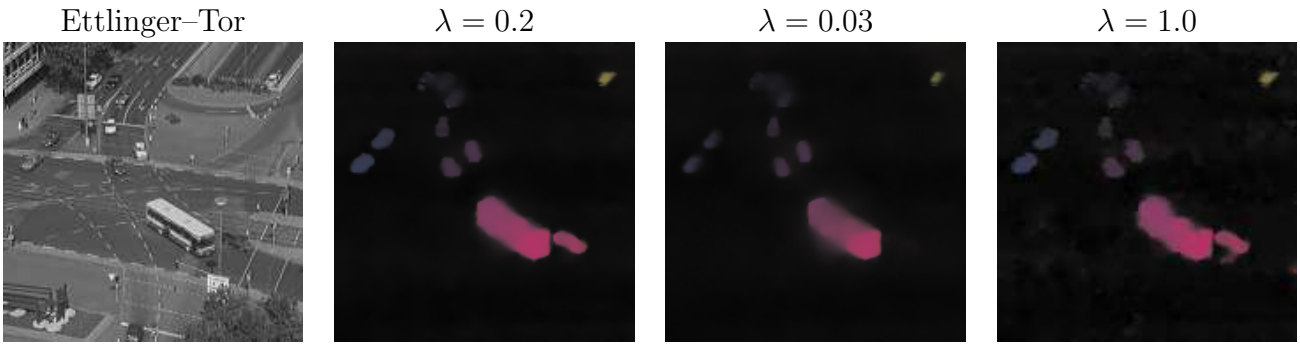


Figure 2: Ettlinger–Tor sequence.

underestimated at the cars and the bus. Note that the method does not estimate the correct flow for the car next to the bus. When λ is big, the attachment term becomes more important and the method is more sensitive to the influence of noise, resulting in unstable flow fields.

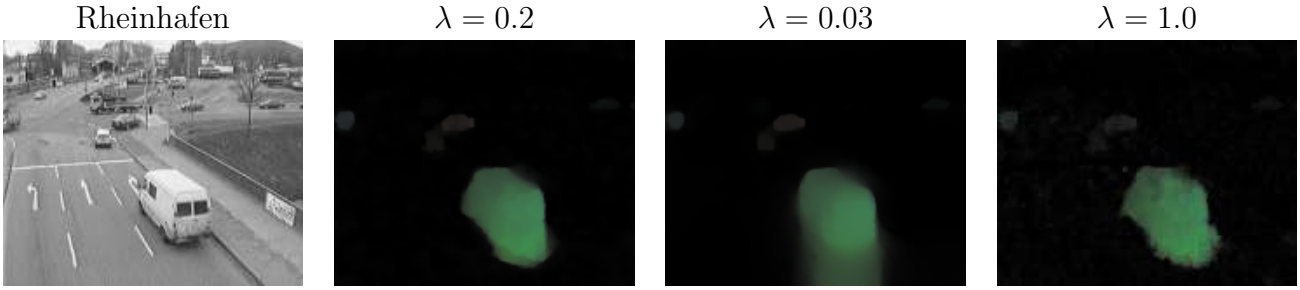


Figure 3: Rheinhafen sequence.

The results for the Rheinhafen sequence are similar: the method detects the movement of the cars and a small background shift. It also detects the motion of the shadow behind the truck. When λ is smaller, the flow fields are smoother and the effect of the shadow is spread.

5.1 Yosemite Sequence

In this example (see figure 4) we analyze the behavior for the the Yosemite sequences, both with and without clouds. The flow is estimated between frames 8 and 9, as Zach et al. [6]. The parameters used for the sequence without clouds are τ=0.25, λ=0.11, θ=0.45, $\eta = 0.5$, ε=0.01, 5 scales and 5 warpings. For the sequence with clouds the set of parameters are the same except for λ=0.025 and $\theta = 0.6$.

The experiments in this section have been carried out in an Intel Core2 CPU at 2.4 GHz with 2 GB of RAM. The source code uses OpenMP directives to parallelize several loops, but in the following examples, the running times are calculated for one processor only.

In table 2, we show the Average End-Point Error and Average Angular Error (EPE and APE, respectively). The results are slightly better than in Zach et al. [6]. Note that our implementation uses several warpings per scale and a different stopping criterion, which depends on the convergence rate between intermediate solutions. The formulas used to calculate the EPE and AAE are:

$$EPE := \frac{1}{N}\sum_{i=1}^{N}\sqrt{\left(u_{1,i}-u_{1,i}^{gt}\right)^2+\left(u_{2,i}-u_{2,i}^{gt}\right)^2}, \tag{18}$$

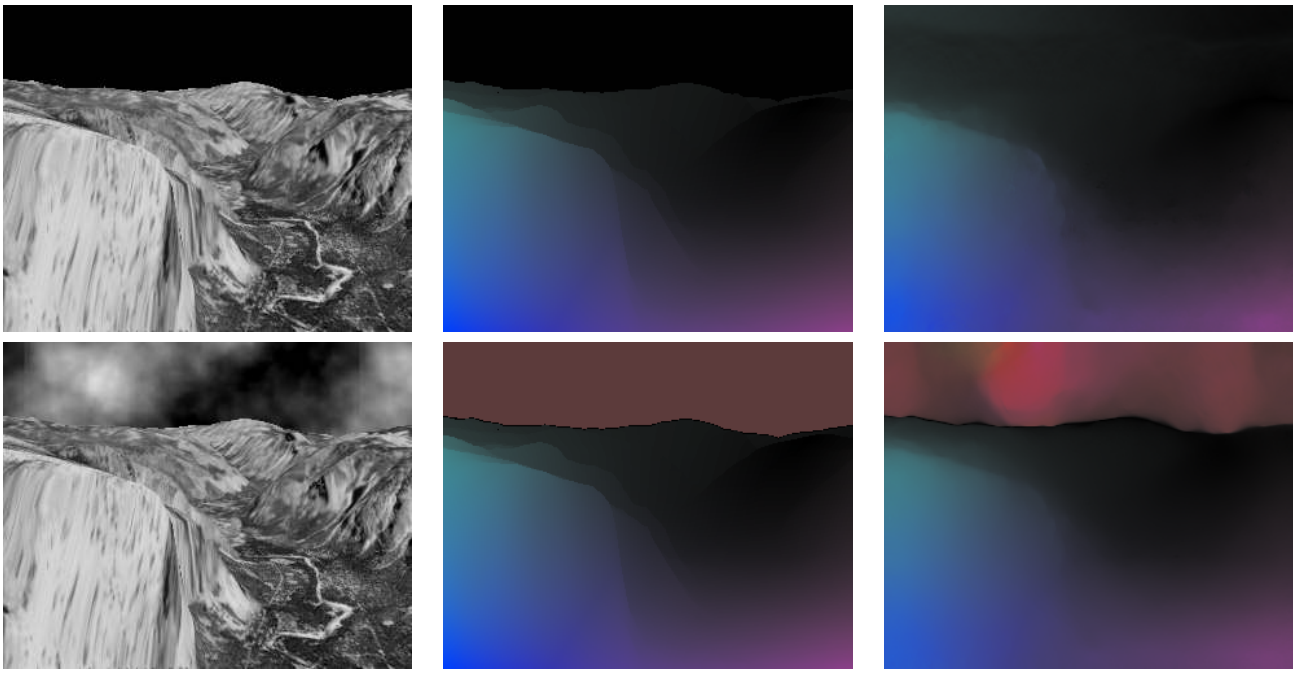


Figure 4: Yosemite and Yosemite with Clouds sequences.

Table 2: EPE and AAE for the Yosemite test sequences.

Errors	EPE	AAE	Running time
Yosemite	0.095 pixels	2.046^o	2.112s
Yosemite with Clouds	0.251 pixels	4.568^o	3.208s

$$AAE := \frac{1}{N}\sum_{i=1}^{N} \arccos\left(\frac{u_{1,i}u_{1,i}^{gt} + u_{2,i}u_{2,i}^{gt}}{\sqrt{u_{1,i}^2 + u_{2,i}^2 + 1}\sqrt{u_{1,i}^{2,gt} + u_{2,i}^{2,gt} + 1}}\right), \tag{19}$$

with $\mathbf{u}^{gt} = (u_1^{gt}, u_2^{gt})$ the ground truth solution.

The running time may be further reduced if ε is increased and/or the number of warpings is decreased. In Zach et al. [6], the implementation and experiments were carried out with only one warping per scale.

In figure 5 we observe the evolution of EPE and AAE with respect to θ, given several fixed values for λ. We observe that the errors decrease rapidly and then slowly increase. Note that, in order to appreciate the evolution of the error, the graphics are in logarithmic scale. From these graphics we observe that a value of $\theta = 0.3$ is a good choice for several values of λ.

When θ is very large, the importance of the coupling term, $\frac{1}{2\theta}|\mathbf{u} - \mathbf{v}|^2$, fades away and there is no effective transfer of information between $\mathbf{u}$ and $\mathbf{v}$. When θ is very small, the coupling term outweighs the data and regularization terms at each iteration. Thus, there is too much transfer of information in the iterations and the solution is given by $\mathbf{u} = \mathbf{v}$. In either case ($\theta \to 0$ or $\theta \to \infty$), the method provides bad results. In practice, there is a wide range of θ values for which the result is reasonable (e.g., between 0.1 and 100).

Figure 6 shows the same error graphics changing the roles of λ and θ. A small value of λ creates very smooth flow fields since it increases the weight of the regularization. A larger value increases the dependency on the attachment term and the solutions become less regular. λ has a greater influence on the results and we observe that the solutions for $\lambda \leq 0.2$ are more accurate.

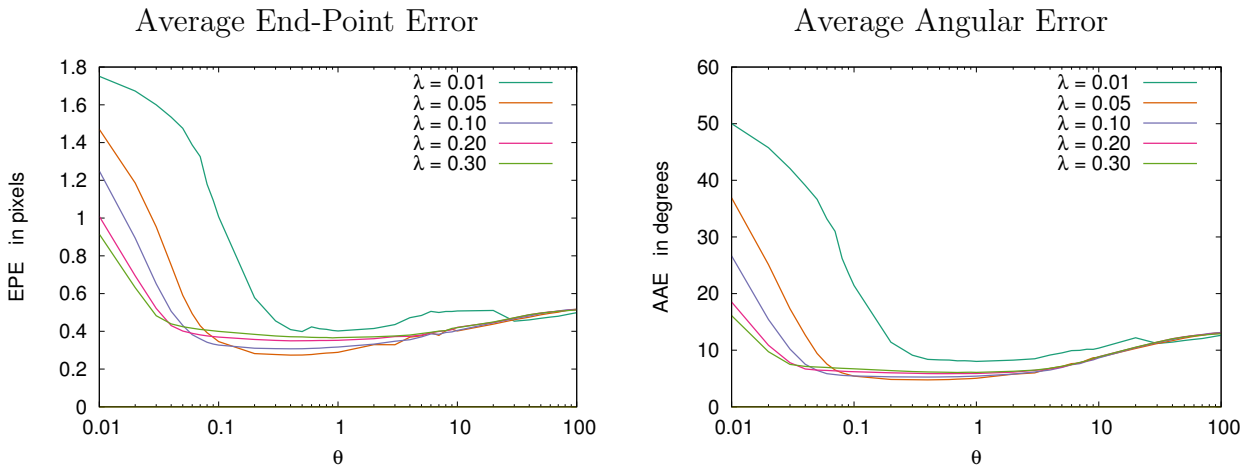


Figure 5: Errors for sequence "Yosemite with Clouds" as a function of θ.

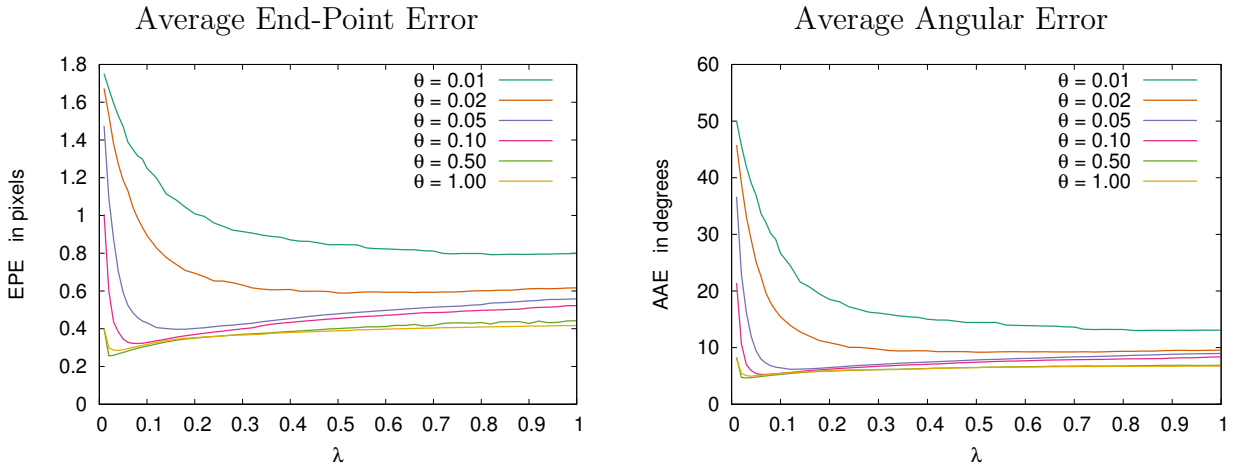


Figure 6: Errors for sequence "Yosemite with Clouds" as a function of λ.

Table 3: EPE and AAE for the Middlebury test sequences, using default parameters.

Errors	Dimetrodon	Grove2	Grove3	Hydrangea	Rubberwhale	Urban2	Urban3	Venus
EPE	0.162p	0.156p	0.721p	0.258p	0.215p	0.382p	0.711p	0.394p
AAE	2.888^o	2.311^o	6.590^o	2.814^o	6.865^o	3.016^o	6.631^o	6.831^o

Table 4: EPE and AAE for the Middlebury test sequences, using the best parameters.

Errors	Dimetrodon	Grove2	Grove3	Hydrangea	Rubberwhale	Urban2	Urban3	Venus
EPE	0.152p	0.153p	0.673p	0.244p	0.199p	0.360p	0.535p	0.296p
AAE	2.772^o	2.261^o	6.359^o	2.637^o	6.428^o	2.671^o	4.183^o	4.376^o

5.2 Middlebury Database

This section shows several tests with the sequences in the Middlebury benchmark database[2]. This database contains two types of data: those for which the ground truth are made public (called "Test sequences"), and those for which the ground truth are not public ("Evaluation sequences"). The former are used for testing purposes and finding the appropriate parameters of the method, and the latter are used to develop a ranking on the web page.

5.2.1 Test Sequences

For the test sequences, we have run the algorithm with the same parameter set: τ=0.25, λ=0.15, θ=0.3, $\eta = 0.5$, ε=0.01, 6 scales and 5 warpings. We also compute the optical flows adapting the values of λ, θ and the number of scales, in order to find a better result.

Figure 7 shows the 10th frame of the sequence, the ground truth, the solution with fixed parameters and the best solution found. These sequences are composed of more than two frames, but the ground truth is only available for the 10th frame. Thus, the estimated optical flow is between frames 10 and 11.

In table 3 we show the EPE and AAE for the Middlebury test sequences, when fixed parameters are used, which corresponds to the optical flows in the third column. In the fourth column of the figure, we show the best optical flows found and the parameters used. Table 4 shows the EPE and AAE for these experiments. In some cases, e.g. Urban3 and Venus, the improvements in accuracy are important.

As in the previous section, we show the evolution of the EPE and AAE for λ and θ. In this case, we have used the RubberWhale sequence, which is a real video sequence with ground truth. In figure 8, we observe the evolution of EPE and AAE with respect to θ given several fixed values for λ. We observe again that the error evolution in θ is very stable. As it happened for the Yosemite sequence, values of θ around 0.3 seem to provide the best results. Figure 9 shows the same error graphics for λ and several fixed values of θ.

In the case of RubberWhale, the best value for λ is around 0.3. In our experience, a value of $\lambda = 0.15$ yields good results for all the sequences.

5.2.2 Evaluation Sequences

Finally, we show several examples using the evaluation sequences in figure 10. We have used the same parameter set as for the test sequences: τ=0.25, λ=0.15, θ=0.3, $\eta = 0.5$, ε=0.01, 6 scales and 5 warpings.

[2] http://http://vision.middlebury.edu/flow

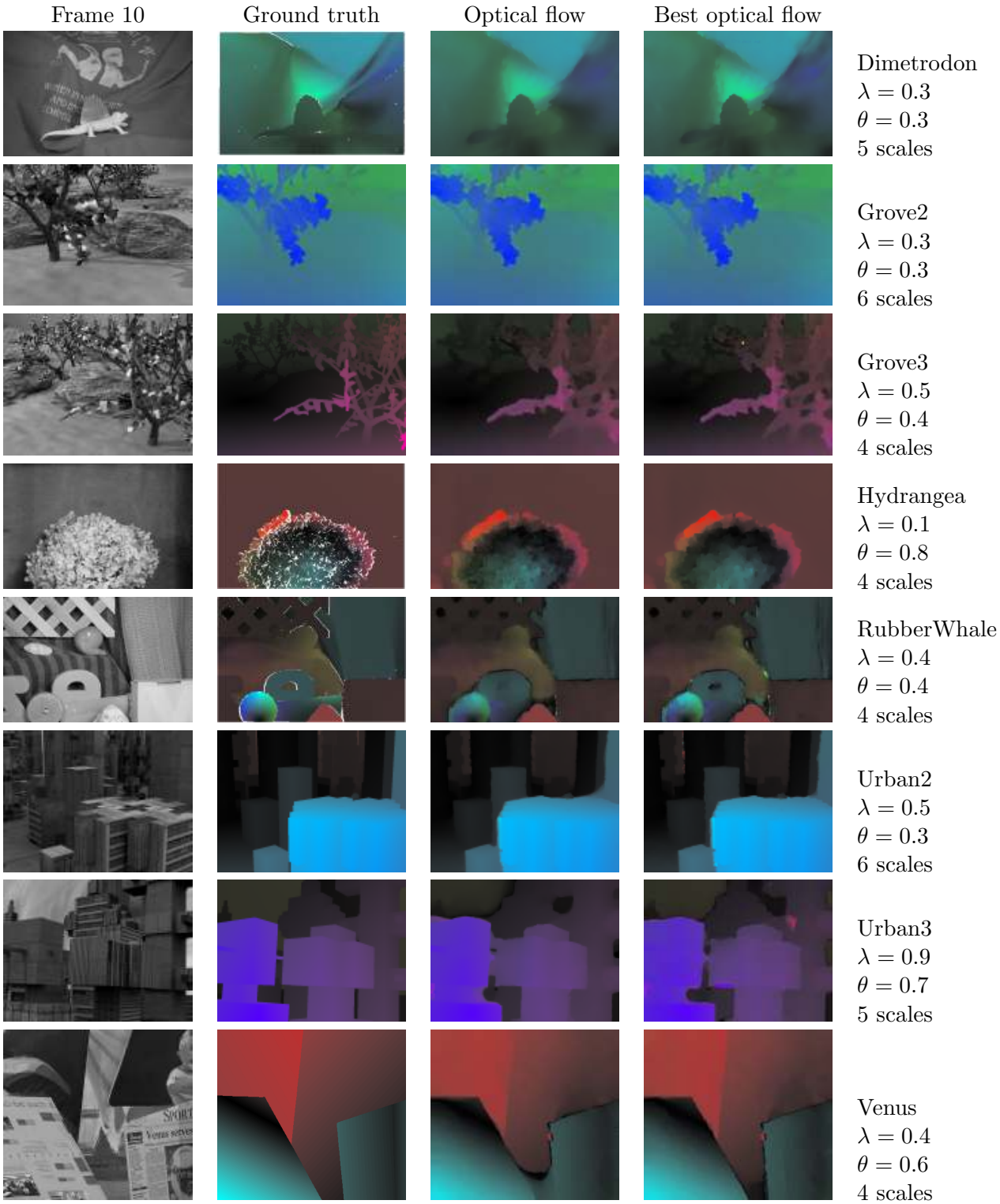


Figure 7: Results for the Middlebury test sequences.

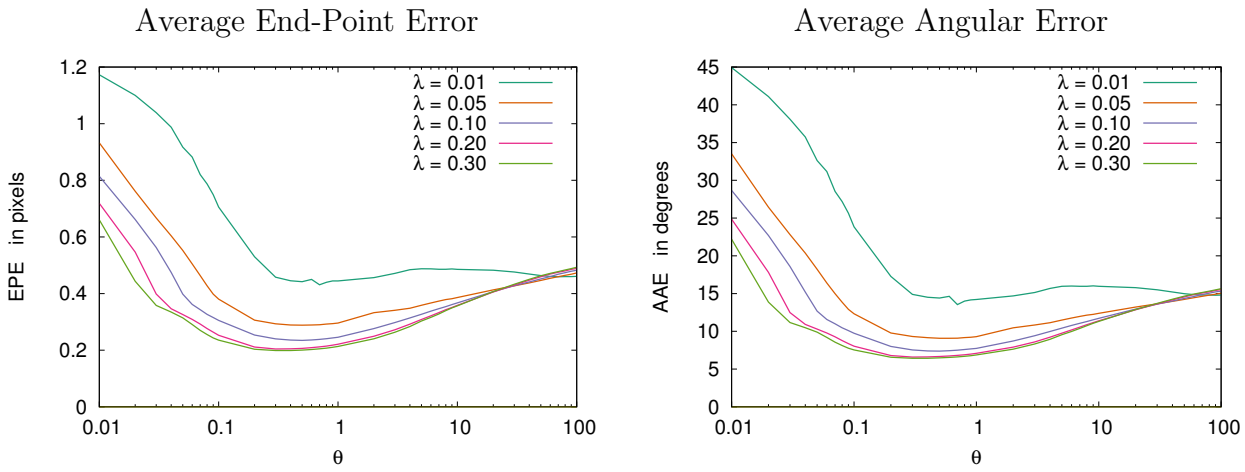


Figure 8: Errors for sequence "RubberWhale" as a function of θ.

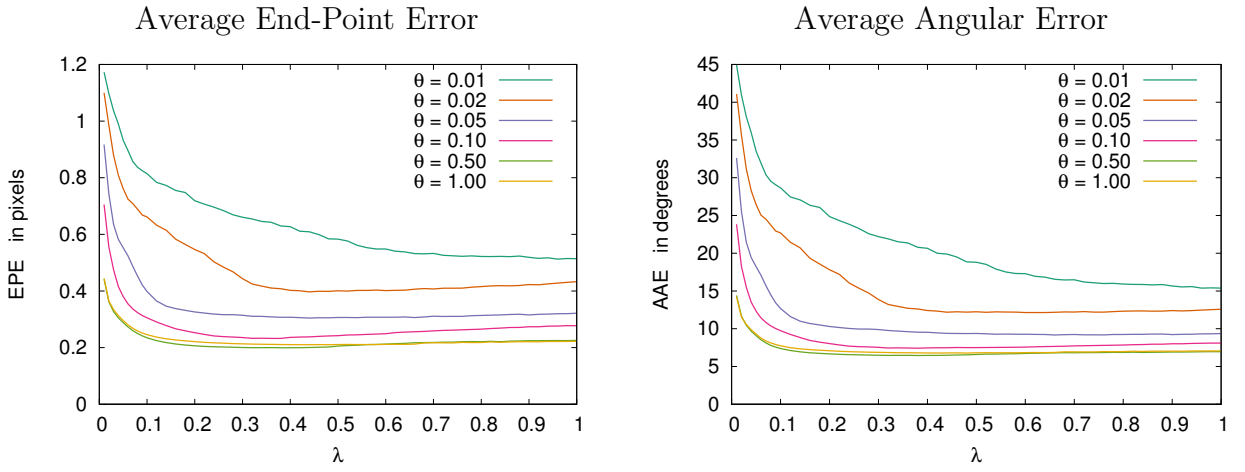


Figure 9: Errors for sequence "RubberWhale" as a function of λ.

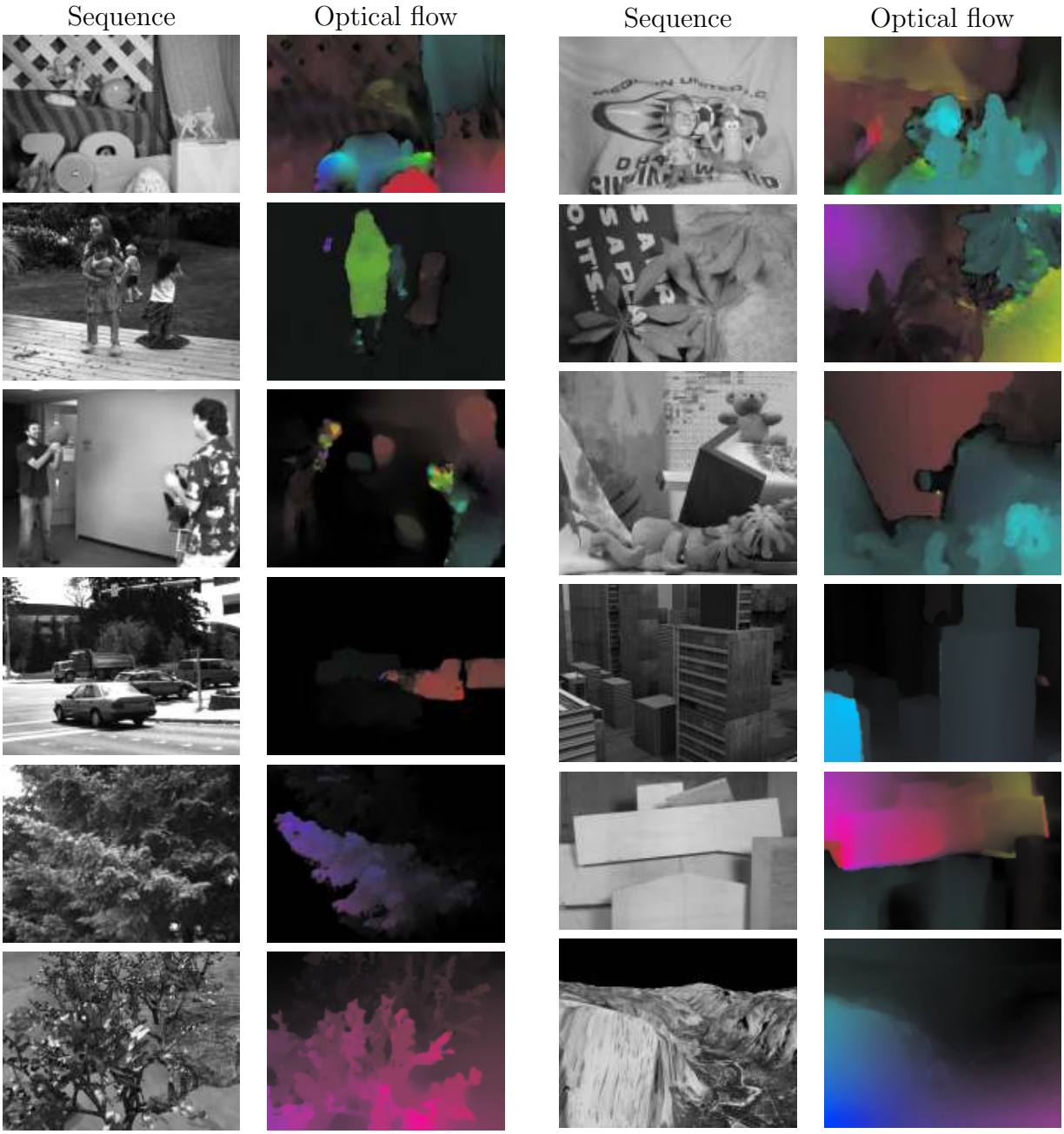


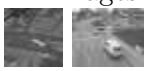
Figure 10: Middlebury evaluation sequences.

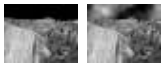
Acknowledgements

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:

 Standard test sequences, Henner Kollnig[3].

 Standard test sequences, Lynn Quam[4].

 Middlebury benchmark database[5].

References

[1] Antonin Chambolle. An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical Imaging and Vision*, 20(1-2):89–97, January 2004. http://dx.doi.org/10.1023/B:JMIV.0000011325.36760.1e

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

[3] Enric Meinhardt-Llopis and Javier Sánchez. Horn–Schunck Optical Flow with a Multi-scale Strategy. *Image Processing On Line*, preprint, 2012.

[4] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, November 1992. http://dx.doi.org/10.1016/0167-2789(92)90242-F

[5] Andreas Wedel, Thomas Pock, Christopher Zach, Horst Bischof, and Daniel Cremers. *Statistical and geometrical approaches to visual motion analysis.* An Improved Algorithm for TV-L^1 Optical Flow, pages 23–45. Springer-Verlag, Berlin, Heidelberg, 2009. http://dx.doi.org/10.1007/978-3-642-03061-1_2

[6] C. Zach, T. Pock, and H. Bischof. A Duality Based Approach for Realtime TV-L^1 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. http://dx.doi.org/10.1007%2F978-3-540-74936-3_22

[3] http://i21www.ira.uka.de/image_sequences/
[4] http://www.cs.brown.edu/~black/images.html
[5] http://vision.middlebury.edu/flow/data/