IPOL IMAGE PROCESSING ON LINE

Published in Image Processing On Line on 2015–06–19.
Submitted on 2014–03–06, accepted on 2014–11–21.
ISSN 2105–1232 © 2015 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.2015.112

# A Line Search Multilevel Truncated Newton Algorithm for Computing the Optical Flow

Luis Garrido[1], El Mostafa Kalmoun[2]

[1] Department of Applied Mathematics and Analysis, University of Barcelona, Barcelona, Spain (lluis.garrido@ub.edu)
[2] Department of Mathematics, College of Science, King Khalid University, Abha, Saudi Arabia
On leave from Cadi Ayyad University, ENSA Marrakech, Morocco (ekalmoun@kku.edu.sa)

**Abstract**

We describe the implementation details and give the experimental results of three optimization algorithms for dense optical flow computation. In particular, using a line search strategy, we evaluate the performance of the unilevel truncated Newton method (LSTN), a multiresolution truncated Newton (MR/LSTN) and a full multigrid truncated Newton (FMG/LSTN). We use three image sequences and four models of optical flow for performance evaluation. The FMG/LSTN algorithm is shown to lead to better optical flow estimation with less computational work than both the LSTN and MR/LSTN algorithms.

**Source Code**

The ANSI C implementation of the source code, the code documentation, and the online demo are accessible at the IPOL web page of this article[1].

**Keywords:** optical flow; line search multigrid optimization; multiresolution; truncated Newton

## 1 Introduction

Let us consider a sequence of gray level images $I(k, m, n)$, where $k = 0, \ldots, K$ denotes the frame number and $(m, n)$ denotes the pixel coordinates, $m$ (respectively $n$) corresponds to the discrete column (respectively row) of the image, being the coordinate origin located at the top-left corner of the image. We denote by $I^t, I^x, I^y$ the discrete partial derivatives of $I$ in the $k, m, n$ directions, respectively, which are computed using high-pass gradient filters. Finally, we denote by $w_{m,n} = (u_{m,n}, v_{m,n})$ the optical flow between two successive frames at pixel $(m, n)$.

From the brightness constancy assumption (BCA) [3]

$$I(k+1, m+u_{m,n}, n+v_{m,n}) = I(k, m, n) \text{ for } k = 0, \ldots, K \text{ and each pixel } (m, n),$$

[1] http://dx.doi.org/10.5201/ipol.2015.112

and its first-order linear approximation, we consider four variational optical flow models that we describe here in their discrete forms

$$\min_w \sum_{m,n} \psi\left(I^x u_{m,n} + I^y v_{m,n} + I^t\right) + \alpha \sum_{m,n} ||\nabla_{m,n}(w)||^2 \tag{1}$$

$$\min_w \sum_{m,n} \psi\left(I(k+1, m+u_{m,n}, n+v_{m,n}) - I(k,m,n)\right) + \alpha \sum_{m,n} ||\nabla_{m,n}(w)||^2 \tag{2}$$

$$\min_w \sum_{m,n} \psi\left(I^x u_{m,n} + I^y v_{m,n} + I^t\right) + \alpha \sum_{m,n} \sqrt{||\nabla_{m,n}(w)||^2 + \mu^2} \tag{3}$$

$$\min_w \sum_{m,n} \psi\left(I(k+1, m+u_{m,n}, n+v_{m,n}) - I(k,m,n)\right) + \alpha \sum_{m,n} \sqrt{||\nabla_{m,n}(w)||^2 + \mu^2}, \tag{4}$$

where $\alpha$ is a weighting positive parameter between data and regularization terms, $\mu$ is a small positive parameter to avoid non differentiability of the absolute value function at zero, and $||\nabla_{m,n}(w)||^2$ is computed as

$$||\nabla_{m,n}(w)||^2 = \frac{1}{2}\left((u^{x,+}_{m,m})^2 + (u^{x,-}_{m,n})^2 + (u^{y,+}_{m,n})^2 + (u^{y,-}_{m,n})^2 + (v^{x,+}_{m,n})^2 + (v^{x,-}_{m,n})^2 + (v^{y,+}_{m,n})^2 + (v^{y,-}_{m,n})^2\right). \tag{5}$$

The discrete partial derivatives of $u, v$ are computed via forward and backward finite differences, that is

$$u^{x,+}_{m,n} = \frac{u_{m+1,n} - u_{m,n}}{h}\ ,\ u^{x,-}_{m,n} = \frac{u_{m,n} - u_{m-1,n}}{h},$$
$$u^{y,+}_{m,n} = \frac{u_{m,n+1} - u_{m,m}}{h}\ ,\ u^{y,-}_{m,n} = \frac{u_{m,n} - u_{m,n-1}}{h},$$

where $h$ is the grid distance, see Section 4 for a discussion of its value. Derivatives $v^x$ and $v^y$ are computed similarly.

The function $\psi$ is used to enhance robustness with respect to outliers. In our work we have used

$$\psi(\theta) = \begin{cases} \theta^2/2 & \text{if } |\theta| \le \gamma \\ \gamma^2/2 & \text{otherwise,} \end{cases} \tag{6}$$

where $\gamma$ is a given threshold.

Notice that all the four models in equations (1)-(4) can be written into the general minimization form:

$$\min_w f(w) = D(w) + \alpha R(w), \tag{7}$$

where $D$ is a data term modelling the optical flow and $R$ is a regularization term, which corresponds here to standard convex regularizers; namely a quadratic term in case of the first two models and a differentiable approximation of the total variation (TV) for the last two models.

We note also that the data term in the first model corresponds to a robust version of the Horn-Schunck method [3] which combines a linear version of the BCA assumption with a quadratic regularization. In particular, the data term in model 1 is a truncated quadratic function, which for inliers behaves as Horn-Schunck data term. The second model uses the original non-linear version of the constancy assumption as a data term, which makes the energy functional highly non-convex. On the other hand, the difficulty in model 3 arises from the presence of the TV term which is known to be computationally more demanding than the quadratic regularization term. Finally, model 4 combines the non-linear data term with total variation regularization, which makes this model the most complicated among the four models used in this paper due to the high non-convexity of the energy

functional and the presence of the TV regularization. Table 1 summarizes the four models that have been implemented in our code. The four models are minimized using three optimization algorithms based on a line search truncated Newton method known to be able to handle non-convex problems. More precisely, we use a unilevel implementation of this method together with its multiresolution and multigrid versions. In particular, for a given pair of images, twelve results can be produced using the three optimization algorithms applied on the four optical flow models. The optimization algorithms will be described in the next sections but we need first to describe how the gradient of the energy functional is computed.

| | Data term $D(w)$ | Regularization term $R(w)$ |
|---|---|---|
| Model 1 | Linear BCA | Quadratic |
| Model 2 | Non linear BCA | Quadratic |
| Model 3 | Linear BCA | Total variation |
| Model 4 | Non linear BCA | Total variation |

Table 1: Description of the four optical flow models that have been implemented.

## 2 Energy Gradient

The proposed algorithms only requires the computation, for each iteration $k$, of the energy $f(w)$ and its gradient $\nabla f(w)$. The Hessian $\nabla^2 f(w)$, as indicated in Section 3, is approximated by means of the gradient.

The gradient of the objective function in Equation (7) is calculated analytically and given by

$$g = \nabla f = \begin{pmatrix} f^u \\ f^v \end{pmatrix} = \begin{pmatrix} D^u + \alpha R^u \\ D^v + \alpha R^v \end{pmatrix},$$

where $f^u$ and $f^v$ denote, respectively, the partial derivative of function $f$ with respect to variables $u$ and $v$. In our case we discretize the grid prior to computing the gradient. Thus, we denote

$$D^u = \begin{pmatrix} D^u_{m,n} \\ D^v_{m,n} \end{pmatrix} \quad \text{and} \quad R^u = \begin{pmatrix} R^u_{m,n} \\ R^v_{m,n} \end{pmatrix},$$

where $D^u_{m,n}$ and $D^v_{m,n}$ (resp. $R^u_{m,n}$ and $R^v_{m,n}$) denote the partial derivative of $D(m,n)$ (resp. $R(m,n)$) with respect to variables $u_{m,n}$ and $v_{m,n}$, respectively.

For the linear data term in equations (1) and (3), the gradient for $|\theta| \leq \gamma$, see equation (6), is

$$\begin{aligned} D^u_{m,n} &= I^x \left[ I^x u_{m,n} + I^y v_{m,n} + I^t \right] \\ D^v_{m,n} &= I^y \left[ I^x u_{m,n} + I^y v_{m,n} + I^t \right], \end{aligned}$$

where $I^x, I^y, I^t$ are the spatial and temporal image derivatives. Note that for $|\theta| > \gamma$ the gradient $D$ is $(D^u_{m,n}, D^v_{m,n})^T = (0,0)^T$.

For the non-linear data term in equations (2) and (4), the gradient for $|\theta| \leq \gamma$ is given by

$$\begin{aligned} D^u_{m,n} &= I^x_2(m + u_{m,n}, n + v_{m,n}) \left[ I_2(m + u_{m,n}, n + v_{m,n}) - I_1(m,n) \right] \\ D^v_{m,n} &= I^y_2(m + u_{m,n}, n + v_{m,n}) \left[ I_2(m + u_{m,n}, n + v_{m,n}) - I_1(m,n) \right]. \end{aligned}$$

As before, for $|\theta| > \gamma$ the gradient $D$ is $(D^u_{m,n}, D^v_{m,n})^T = (0,0)^T$.

We consider now the quadratic regularization term in equations (1) and (2). The gradient of this functional is obtained as

$$\begin{aligned} R^u_{m,n} &= \frac{2}{h^2}\left(4u_{m,n} - u_{m-1,n} - u_{m,n-1} - u_{m+1,n} - u_{m,n+1}\right) \\ R^v_{m,n} &= \frac{2}{h^2}\left(4v_{m,n} - v_{m-1,n} - v_{m,n-1} - v_{m+1,n} - v_{m,n+1}\right) \end{aligned}$$

Finally, the gradient of the total variation approximation used in equations (3) and (4) is

$$\begin{aligned} R^u_{m,n} &= \frac{1}{2\,h}\left(\frac{(u^{x,-}_{m,n} + u^{y,-}_{m,n}) - (u^{x,+}_{m,n} + u^{y,+}_{m,n})}{\phi_{m,n}} + \frac{u^{x,+}_{m-1,n}}{\phi_{m-1,n}} + \frac{u^{y,+}_{m,n-1}}{\phi_{m,n-1}} - \frac{u^{x,-}_{m+1,n}}{\phi_{m+1,n}} - \frac{u^{y,-}_{m,n+1}}{\phi_{m,n+1}}\right) \\ R^v_{m,n} &= \frac{1}{2\,h}\left(\frac{(v^{x,-}_{m,n} + v^{y,-}_{m,n}) - (v^{x,+}_{m,n} + v^{y,+}_{m,n})}{\phi_{m,n}} + \frac{v^{x,+}_{m-1,n}}{\phi_{m-1,n}} + \frac{v^{y,+}_{m,n-1}}{\phi_{m,n-1}} - \frac{v^{x,-}_{m+1,n}}{\phi_{m+1,n}} - \frac{v^{y,-}_{m,n+1}}{\phi_{m,n+1}}\right), \end{aligned}$$

where

$$\phi_{m,n} = \sqrt{||\nabla_{m,n}(w)||^2 + \mu^2}. \tag{8}$$

# 3 Line Search Truncated Newton

In general, the line search Newton method solves the minimization problem (7) by the following iterations, which we will call outer iterations

$$w_{k+1} = w_k + \lambda_k s_k, \tag{9}$$

where $\lambda_k$ is the line search step size and $s_k$ is the search direction solution to the following linear system

$$H_k s = -g_k. \tag{10}$$

Here $g_k = \nabla f(w_k)$ is the gradient and $H_k = \nabla^2 f(w_k)$ is the Hessian. Notice that the search direction from the last equation is based on a Taylor series expansion near the solution of the optimization problem (7), and hence there is no guarantee that the Newton search direction will be crucial far away from the exact solution; especially during the first iterations. Moreover, for large scale problems as the optical flow, solving exactly the linear system (10) will be very time-consuming. As a remedy, line search truncated Newton methods (LSTN) use an iterative method to find an approximate solution to (10) and truncate the iterates as soon as a required accuracy is reached or whenever a negative curvature is detected in the non-convex case. In our algorithm, $s_k$ is obtained by approximately solving the linear system (10) using the preconditioned conjugate gradient algorithm (PCG), see Algorithm 1. In the proposed algorithm the Hessian does not need to be explicitly computed. Indeed, note that in Algorithm 1 only the product of the Hessian $H_k$ with a vector $p$ needs to be computed. In our case this product is computed via forward finite differences

$$H_k\, p = \frac{g(w_k + \epsilon p) - g(w_k)}{\epsilon},$$

where $\epsilon$ is chosen to be the square root of the machine precision divided by the norm of $w_k$.

We will refer to the process of finding $s_k$ as inner iterations. In the PGC algorithm (Algorithm 1), the inner iterations are truncated as soon as a negative curvature direction is detected, that is, when $p_j^T H_k p_j$ is negative. In our case, we replace the negative curvature test with the equivalent descent direction test [10], see lines 8-11 of Algorithm 1. From our practical experience, the descent direction test has a better numerical behavior than the negative curvature test.

**Algorithm 1** Preconditioned Conjugate Gradient (inner iterations of LSTN)

**Input** $g_k$, $M_k$: gradient and preconditioning matrix at Algorithm 2, line 11.
**Output** $s_k$: search direction; see return value for Algorithm 2, line 11.
**Parameters**: $\epsilon = 10^{-10}$, $\zeta_k$ (see (11))
maxinner: maximum number of iterations to approximately solve (10)
**Initialization**: $z_0 = 0$, $r_0 = -g_k$, $v_0 = M_k^{-1} r_0$, $p_0 = v_0$
*// Algorithm code*
**for** $j = 0$ to maxinner **do**
  *// Singularity test*
  **if** $\left(|r_j^T v_j| < \epsilon \text{ or } |p_j^T H_k p_j| < \epsilon\right)$ **then**
    **exit** with $s_k = z_j$ (for $j = 0$ take $s_k = -g_k$)
  **end if**
  $\lambda_j = r_j^T v_j / p_j^T H_k p_j \quad , \quad z_{j+1} = z_j + \lambda_j p_j$
  *// Descent Direction Test replaces Negative Curvature test*
  **if** $(g_k^T z_{j+1} \geq g_k^T z_j - \epsilon)$ **then**
    **exit** with $s_k = z_j$ (for $j = 0$ take $s_k = -g_k$)
  **end if**
  $r_{j+1} = r_j - \lambda_j H_k p_j \quad , \quad v_{j+1} = M_k^{-1} r_{j+1}$
  *// Truncation test (note that* $||r_{j+1}||_{M_k^{-1}} = r_{j+1}^T v_{j+1}$*)*
  **if** $(r_{j+1}^T v_{j+1} \leq \zeta_k g_0^T v_0)$ **then**
    **exit** with $s_k = z_{j+1}$
  **end if**
  $\beta_j = r_{j+1}^T (v_{j+1} - v_j) / r_j^T v_j \quad , \quad p_{j+1} = v_{j+1} + \beta_j p_j.$
**end for**
**exit** with $s_k = z_j$

The inner iterations are also truncated when

$$||r_j||_{M_k^{-1}} \leq \zeta_k ||r_0||_{M_k^{-1}},$$

where $r_j$ is the PCG residual at inner iteration $j$ and $||r_j||_{M_k^{-1}} = \sqrt{r_j^T M_k^{-1} r_j}$. $M_k$ is the preconditioning matrix and

$$\zeta_k = max\left(0.5/(k+1), ||r_0||_{M_k^{-1}}\right) \tag{11}$$

are both being provided at outer iteration $k$. Note that a maximum number of inner iterations are performed in Algorithm 1 in order to unnecessarily increase the computational cost associated to computing $s_k$.

The computation of the search direction $s_k$ is embedded within a line search algorithm as shown in Algorithm 2. That is, Algorithm 2 iteratively computes a search direction $s_k$ and then updates the current point $w_k$ in the direction of $s_k$ by a step $\lambda_k$. For computing the step size $\lambda_k$ exact line search is avoided due to expensive function evaluations. We have used instead a cubic interpolation [6] until Wolf's conditions are satisfied

$$f_{k+1} \leq f_k + c_1 \lambda_k g_k^T s_k \tag{12}$$

$$g_{k+1}^T s_k \geq c_2 g_k^T s_k, \tag{13}$$

where $0 < c_1 < c_2 < 1$.

Algorithm 2 is associated to the outer iterations of LSTN, see Equation (9), in contrast to Algorithm 1 which is associated to the inner iterations of LSTN, see Equation (10). Algorithm 2

**Algorithm 2** Line Search Truncated Newton (outer iterations of LSTN)

**Input** $w_0$: Input flow (initialization)
**Output** $w_k$: Output flow
**Parameters** $\epsilon_g$, $\epsilon_f$, $\epsilon_w$: convergence thresholds
maxouter: maximum number of iterations of (9)
**Initialization** $M_0$ = Identity matrix: preconditionning matrix
*// Algorithm code*
**for** $k = 0$ to maxouter **do**
  $g_k = \nabla f(w_k)$
  **if** $||g_k|| < \epsilon_g$ **then**
    **exit** with solution $w_k$
  **end if**
  Compute $s_k$ by calling Algorithm 1.
  Perform a line search to scale the step $s_k$ by $\lambda_k$.
  $w_{k+1} = w_k + \lambda_k s_k$
  Update from $M_k$ to $M_{k+1}$ by the BFGS formula
  **if** $|f_{k+1} - f_k| < \epsilon_f$ or $||w_{k+1} - w_k|| < \epsilon_w$ **then**
    **exit** with solution $w_k$
  **end if**
**end for**

iteratively updates $w_k$ by computing a search direction and performing a line search until a given tolerance is reached. In our case we set tolerances on the gradient, $\epsilon_g$, to detect local minima, and on the function values and iterate values, $\epsilon_f$ and $\epsilon_w$ to detect convergence. Note also that the preconditioning matrix $M_k$ is updated here, which can be easily done from previous values of $w_k$ and $g(w_k)$, see [8]. For the preconditioning strategy, we have used a scaled two-step limited memory BFGS [7] with a diagonal scaling for preconditioning the CG method.

# 4 Multiresolution Line Search Truncated Newton

Multiresolution methods use a series of coarse to fine resolution levels to obtain an estimate of the solution to the problem. Let us denote by $\Omega_i$ the image domain at level $i$, where $i = 0, \ldots, L-1$, where $i = 0$ corresponds to the finest level of resolution and $i = L-1$ to the coarsest one. Here $L$ refers to the number of resolution levels used. Grid spacing at the grid $\Omega_{i+1}$ is, in our implementation, twice the spacing at the grid $\Omega_i$. By default, the grid spacing at the finest level is $h = 1$, whereas at the remaining levels the spacing is $h = 2^i$.

In the multiresolution algorithm, we start at the coarsest level by applying the LSTN algorithm (see Algorithm 2). Then, we prolongate this coarsest estimate to the next level of resolution where it is refined again by the LSTN algorithm. We repeat this process from one coarse level to another finer level until reaching the finest level. This multiresolution function (MR) is shown in Algorithm 3 where it is called with $\mathrm{MR}(L-1, w_{L-1,0})$, where $L$ is the number of resolution levels and $w_{L-1,0}$ is the initial estimate on the coarsest level.

Using a coarse-to-fine strategy, one can obtain a good initial estimate for the finest grid problem from a series of coarse grid estimates. Nevertheless, multiresolution algorithms, which are a one-way multilevel process, are known to be less efficient than standard two-way multilevel algorithms known as multigrid methods.

**Algorithm 3** Multiresolution Line Search Truncated Newton

**Input** $w_{L-1,0}$: initial flow estimate for coarsest level
**Output** $w_{0,1}$: output flow for finest level
**Parameter** $L$: number of resolution levels
**Initialization** $i = L - 1$: coarsest level of resolution
*// Algorithm code*
**repeat**
  Apply LSTN (Algorithm 2) with initial estimate $w_{i,0}$
  Let $w_{i,1}$ be the returned solution
  **if** $i > 0$ **then**
    *// Prolong current estimate to next finer level*
    $w_{i-1,0} = P\, w_{i,1}$
  **end if**
  $i = i - 1$
**until** $i < 0$
**exit** with output flow $w_{0,1}$

# 5 Full Multigrid Line Search Truncated Newton

Full multigrid method (FMG) combines a multiresolution approach with a multigrid cycle. In the multiresolution approach the search direction $s_k$ is computed, for a given level $r$, using the image domain $\Omega_r$. In multigrid the idea is to update $w_k$ by using $\Omega_r$ and it's coarser levels. This allows to reduce the overall computational cost associated to minimizing Equation (7). The method is described next. For more details we refer the reader to [4].

The FMG starts at the coarsest grid level and solves a very low-dimensional problem (using Algorithm 2) as is done for multiresolution. Then the solution is extended to a finer space and one (or several) multigrid cycles are performed to solve the problem on that level. The solution is then again extended to a finer space to perform one (or several) multigrid cycles on the next level. This process is repeated until the finest level is reached. In this way, a good initialization is obtained to start the multigrid cycle on the finest level, which usually reduces the total number of cycles required. To finish one (or several) multigrid cycles are performed on the finest grid level. The method is illustrated in Figure 1 for three levels and using a V-cycle.

Let us consider the optimization problem (7) over the finest grid level $i = 0$. In a multigrid cycle a sequence of optimization subproblems are considered on nested coarser grids. For the finest level $i = 0$, the optimization problem in the multigrid cycle corresponds to the minimization of $h_0 = f_0$, the finest representation of the objective function $f$. However, for a coarser level $i$, the optimization problem in the multigrid cycle corresponds to the minimization of a function $h_i$ that shall be specified next.

Let $w_{i,0}$ be an initial fine approximation to the optimization problem at level $i$. The first step in the multigrid procedure is called pre-optimization and consists in applying $N_0$ iterations of LSTN to $h_i$ to obtain $w_{i,N_0}$. Then $w_{i,N_0}$ is transferred to a coarser grid to obtain $w_{i+1,0} := R\, w_{i,N_0}$ where $R$ is a restriction (downsampling) operator. The residual at this level is given by

$$r_{i+1} := \nabla f_{i+1}(w_{i+1,0}) - R\nabla h_i(w_{i,N_0}).$$

The function

$$h_{i+1}(w_{i+1}) = f_{i+1}(w_{i+1}) - r_{i+1}^T w_{i+1} \tag{14}$$

is the function to be minimized on the coarse grid level $i + 1$, where $f_i$ with $i \geq 0$ $f_i$ denotes a representation of the objective function $f$ on this level, where $h_0 := f_0 = f$ on the finest level.

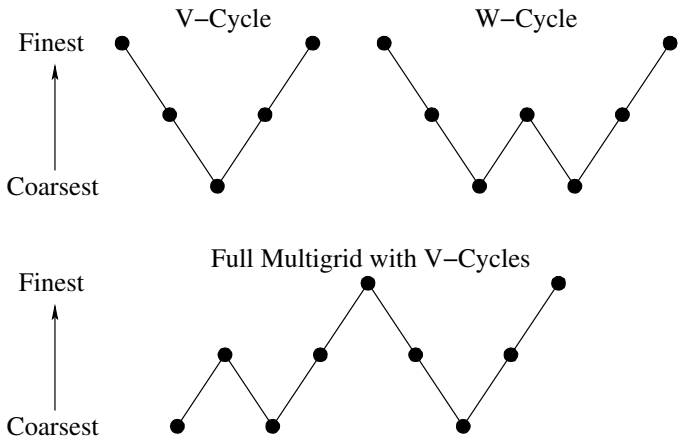


Figure 1: FMG with V-cycle. The top row shows a V and a W cycle. The bottom row shows a FMG with 3 levels and using one V-cycle for each level. Each dot represents several iterations of Equation (9).

Assume we apply Algorithm 2 at level $i+1$ to $h_{i+1}$ and let $w_{i+1,*}$ be the associated minimum. The error between $w_{i+1,*}$ and the initial approximation $w_{i+1,0}$ is called coarse grid correction. This correction is extended back to level $i$

$$s_{i,N_0} = P(w_{i+1,*} - w_{i+1,0}),$$

where $P$ is a prolongation (upsampling) operator. In an optimization context, this correction step $s_{i,N_0}$ is used to update the current solution $w_{i,N_0}$ at level $i$ to $w_{i,N_0+1}$. Note that the step $s_{i,N_0}$ has been obtained using level $i+1$ and it can be considered as a search direction. We call this search direction *recursive* to distinguish it from the *direct* search direction step that is computed by the PCG algorithm on level $i$. Finally, in order to remove the oscillatory components that may have been introduced by the correction step, one may finish with a post-optimization phase by applying $N_1$ iterations of LSTN to $h_i$ with initial guess $w_{i,N_0+1}$, obtaining $w_{i,N_0+N_1+1}$.

The previous explained algorithm forms the basis for the V-cycle in multigrid. For a given function $h_i$, its minimum is computed iteratively using Equation (9) where the search direction $s_k$ may be computed either directly (Algorithm 2) or recursively, see Figure 1. In order to improve the computational efficiency, the line search for a recursive direction step $s_i$ is performed only if $w_{i,k} + \lambda_k s_i$ with $\lambda_k = 1$ does not reduce the value of $h_i$. That is, if $h_i(w_{i,k} + s_i) < h_i(w_{i,k})$ we update with $w_{i,k+1} = w_{i,k} + s_i$. Otherwise a line search is performed.

Algorithm 4 shows the V-cycle algorithm used for FMG/LSTN. At each iteration the algorithm computes a step $s_i$ either directly using the PCG method (Algorithm 1) on the current level, or recursively by means of the multigrid strategy. However, as noted in [9, 2], the recursive call is useful only if $||Rg_{i,k}||$ is large enough compared to $||g_{i,k}||$, where $g_{i,k} = h_i(w_{i,k})$ and $k \leq N_0$. Thus we restrict the use of a coarser level $i+1$ to the case where

$$||Rg_{i,k}|| > \kappa_g ||g_{i,k}|| \text{ and } ||Rg_{i,k}|| > \epsilon_{Rg}, \tag{15}$$

for some constant $\kappa_g \in (0, \min ||R||)$ and where $\epsilon_{Rg} \in (0,1)$ is a measure of first order criticality for $h_{i+1}$. The latter condition is easy to check before trying to compute a step at level $i+1$.

Algorithm 5 shows the Full Multigrid (FMG) algorithm. The algorithm is similar to the multiresolution algorithm, see Algorithm 3. However, note that in FMG one (or more) V-cycles are performed at each resolution level, whereas in the multiresolution algorithm the LSTN algorithm is applied at each resolution level. The implementation corresponds to the representation shown in

**Algorithm 4** The V-cycle for the FMG/LSTN algorithm

**Function** MG/LSTN-cycle($i$, $h_i$, $w_{i,0}$)
**Input** $i$ and $w_{i,0}$: current resolution level and initial flow
$h_i$: coarse objective function at level $i$, recall that $h_0 = f$
**Output** $w_{i,k}$: flow after V-cycle
**Parameters** maxouter, maxpre, maxpost: maximum number of iterations
$\epsilon_f$, $\epsilon_w$: convergence thresholds
*// Algorithm code*
**if** $i$ is coarsest level **then**
  task = optimize, $N$ = maxouter *// LSTN algorithm for the coarsest scale*
**else**
  task = pre-optimize, $N$ = maxprei *// Otherwise pre-optimize*
**end if**
$k_0 = k = 0$
**loop**
  **if** task is optimize, pre-optimize or post-optimize **then**
    Compute $s_{i,k}$ by calling Algorithm 1
  **else**
    $w_{i+1,0} = R\, w_{i,k}$ *// Restrict*
    $r_{i+1} = \nabla f_{i+1}(w_{i+1,0}) - R\nabla h_i(w_{i,k})$ *// Residual*
    $h_{i+1}(w_{i+1}) = f_{i+1}(w_{i+1}) - r_{i+1}^T w_{i+1}$ *// Next coarse objective function*
    **Call function** MG/LSTN-cycle(i+1, $h_{i+1}$, $w_{i+1,0}$) *// Recursive call*
    Let $w_{i+1,*}$ be the returned solution
    $s_{i,k} = P(w_{i+1,*} - w_{i+1,0})$ *// Correction step*
  **end if**
  Perform a line search to scale the step $s_{i,k}$ by $\lambda_k$
  $w_{i,k+1} = w_{i,k} + \lambda_{i,k} s_{i,k}$
  **if** $|f_{i,k+1} - f_{i,k}| < \epsilon_f$ or $||w_{i,k+1} - w_{i,k}|| < \epsilon_w$ **then**
    **return** $w_{i,k+1}$
  **end if**
  $k = k + 1$
  *// Select next task to do*
  **if** task is pre-optimize and (15) is satisfied **then**
    task = recursive-call
  **else if** task is recursive-call **then**
    task = post-optimize, $N$ = maxpost, $k_0 = k$
  **end if**
  *// Check if maximum number of iterations has been reached*
  **if** (task is optimize, pre-optimize or post-optimize) and $(k - k_0 = N)$ **then**
    **return** $w_{i,k}$
  **end if**
**end loop**

Figure 1. The algorithm starts at the coarsest level by performing the LSTN algorithm. Then, we prolongate this coarsest estimate to the next level of resolution. As can be seen at the finer resolution levels a these V-cycle is performed. At each level, the V-cycle may be repeated several times. This process is repeated until the V-cycles at the finest level is reached. Note that, for a given level $i$, the V-cycle is performed using levels $L-1, \ldots, i$, see Algorithm 4.

**Algorithm 5** Full Multigrid Algorithm

**Input** $w_{L-1,0}$: initial estimate
**Output** $w_{0,*}$: output flow
**Parameter** $L$: number of resolutin levels
**Initialization** $i = L - 1$: coarsest level of resolution
*// Algorithm code*
**repeat**
  *// If at coarsest level perform LSTN, otherwise perform V-cycle*
  **if** $i$ is coarsest level **then**
    Apply LSTN (Algorithm 2) with initial estimate $w_{i,0}$
    Let $w_{i,1}$ be the returned solution
  **else**
    *// Perform Ncycles V-cycles on the current level*
    $j = 0$
    **repeat**
      Apply V-cycle (Algorithm 4) to level $i$ with initial estimate $w_{i,j}$
      Let $w_{i,j+1}$ be the returned solution
      $j = j + 1$
    **until** $j =$ Ncycles or Algorithm 4 converged
  **end if**
  **if** $i > 0$ **then**
    *// Prolong current estimate to next finer level*
    $w_{i-1,0} = P\, w_{i,j}$
  **end if**
  $i = i - 1$
**until** $i < 0$
return $w_{0,*}$

# 6 Application Execution

This section briefly discusses the way the application is executed and how the associated parameters can be defined.

The application allows to be run:

1. Without any parameters. In this case default parameters are used. The default parameters use full multigrid (Algorithm 5) with 6 resolution levels. It uses model 2 (non-linear data term and quadratic regularization) with a weight of $\alpha = 50.0$ for the regularizer. The maximum number of V-cycles at each resolution level is 5. The parameter details are included within the file `parameters_default.cfg` of the `test` folder of the source code. The latter parameters are indeed the parameters used for the IPOL's online demo.

2. Some parameters may be also specified through the command-line. In this case the default parameters are overridden by the parameters specified through command-line.

3. Parameters specified through a parameter file. This option allows to completely override the default parameters and have a fine control over the optical flow estimation. Indeed, this is the only way to test multiresolution scheme. Within the source code we include some parameter files associated to the Yosemite sequence for each of the models and both multiresolution and multigrid.

For a detailed description on the parameters we refer the reader to the `README.txt` file included within the source code. This file includes the explanation of each parameter (either specified through a parameter file or through command-line) and its recommended values. The reader should take into account that the parameter file allows to specify a rather large number of parameters. This is due to the fact that our application is able to deal with different models (1, 2, 3 or 4 as indicated in the introduction) and multiresolution and multigrid methods.

# 7 Experimental Results

We report the performance evaluation of the unilevel, the multiresolution and the multigrid optimization algorithms applied to estimate the optical flow between two successive frames of the image sequences shown in Figure 2.

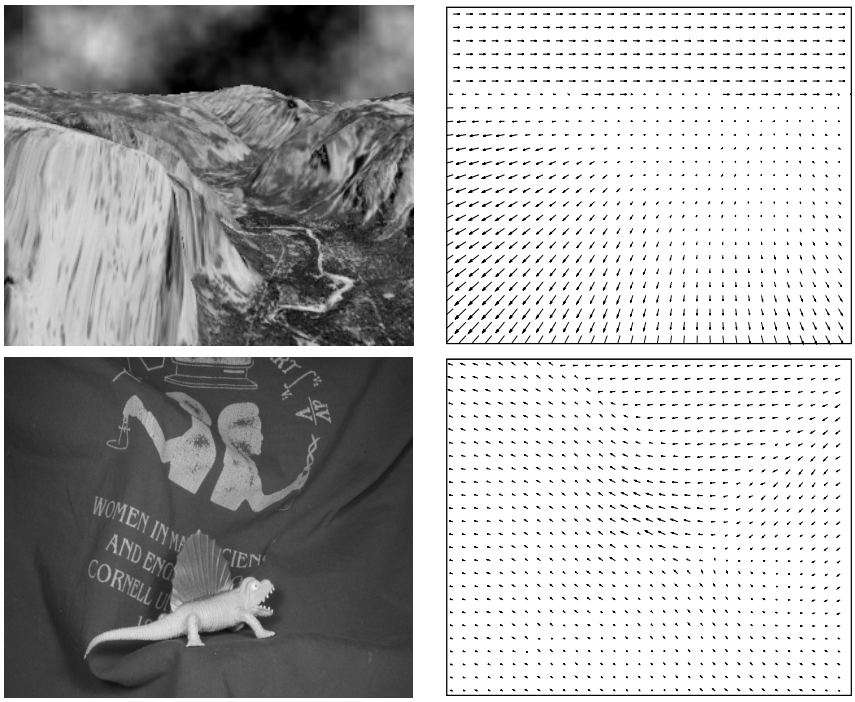

Figure 2: Top corresponds to the Yosemite sequence, bottom to the Dimetrodon sequence. On the left one frame of the sequence is shown. On the right the corresponding ground truth is depicted where motion vectors have been scaled by a factor of 2.5 for better visibility.

In tables 2-5, we summarize the quality of the solution and the computational costs of the three numerical algorithms (LSTN, MR/LSTN and FMG/LSTN) for the four optical flow models. AAE is the average angular error, EPE is the end point error, Time refers to CPU time in seconds, and Nfg is the number of function and gradient evaluations. The first colum (YosClouds) is the original Yosemite sequence whereas the second column (YosNoClouds) is the same sequence without the clouds region. In Figure 3 the colored based representation of the ground truth and estimated optical flows for models 2 and 4 (using MR/LSTN) are shown.

We remark that the FMG/LSTN algorithm outperforms significantly both optimization algorithms in terms of the CPU running time and the number of function and gradient evaluations for the three image sequences and the four optical flow models. For model 1, while using the optimize-discretize linear multigrid approach with standard components does not work [5], our FMG/LSTN algorithm is around eighteen times faster to run than LSTN and at least twice faster than MR/LSTN.

| | | YosClouds | YosNoClouds | Dimetrodon |
|---|---|---|---|---|
| LSTN | AAE | 6.81 | 3.47 | 5.25 |
| | EPE | 0.42 | 0.24 | 0.37 |
| | Time | 5.72 | 6.95 | 24.09 |
| | Nfg | 500 | 568 | 619 |
| MR/LSTN | AAE | 6.34 | 2.68 | 5.25 |
| | EPE | 0.37 | 0.80 | 0.35 |
| | Time | 0.83 | 0.87 | 4.33 |
| | Nfg | 69 | 71 | 120 |
| FMG/LSTN | AAE | 6.29 | 2.65 | 5.10 |
| | EPE | 0.38 | 0.14 | 0.35 |
| | Time | 0.43 | 0.45 | 1.19 |
| | Nfg | 33 | 33 | 28 |

Table 2: Experimental results for Model 1.

| | | YosClouds | YosNoClouds | Dimetrodon |
|---|---|---|---|---|
| LSTN | AAE | 5.94 | 2.34 | 3.12 |
| | EPE | 0.32 | 0.10 | 0.17 |
| | Time | 12.93 | 11.15 | 36.98 |
| | Nfg | 748 | 694 | 771 |
| MR/LSTN | AAE | 5.82 | 2.38 | 3.11 |
| | EPE | 0.31 | 0.10 | 0.17 |
| | Time | 2.49 | 1.79 | 14.30 |
| | Nfg | 149 | 114 | 304 |
| FMG/LSTN | AAE | 5.81 | 2.32 | 3.02 |
| | EPE | 0.32 | 0.10 | 0.17 |
| | Time | 1.19 | 1.04 | 5.17 |
| | Nfg | 72 | 63 | 117 |

Table 3: Experimental results for Model 2.

For model 2 and model 3 which have similar computational complexity, again the FMG/LSTN performs at least eight times better than LSTN and twice better than MR/LSTN. Finally, when used for model 4, the FMG/LSTN algorithm shows a similar significant improvement of a factor of twelve over LSTN and more than one point five over MR/LSTN.

In overall, the FMG/LSTN algorithm performs at least twice better than the MR/LSTN algorithm and ten times better than the unilevel truncated Newton LSTN algorithm; see Table 6. We notice also that the FMG/LSTN algorithm is less independent of the image size because it often takes similar number of function and gradient evaluations while comparing across the same optical flow model.

# Acknowledgments

L. Garrido would like to acknowledge partial support from research project MICINN project, reference MTM2012-30772.

| | | YosClouds | YosNoClouds | Dimetrodon |
|---|---|---|---|---|
| LSTN | AAE | 6.25 | 3.40 | 5.90 |
| | EPE | 0.41 | 0.24 | 0.42 |
| | Time | 16.66 | 19.03 | 63.68 |
| | Nfg | 883 | 1002 | 1019 |
| MR/LSTN | AAE | 5.75 | 2.74 | 5.38 |
| | EPE | 0.37 | 0.18 | 0.36 |
| | Time | 3.85 | 3.96 | 12.22 |
| | Nfg | 182 | 205 | 200 |
| FMG/LSTN | AAE | 5.75 | 2.63 | 5.52 |
| | EPE | 0.36 | 0.15 | 0.32 |
| | Time | 1.46 | 1.37 | 4.09 |
| | Nfg | 75 | 73 | 71 |

Table 4: Experimental results for Model 3.

| | | YosClouds | YosNoClouds | Dimetrodon |
|---|---|---|---|---|
| LSTN | AAE | 5.31 | 2.26 | 3.66 |
| | EPE | 0.30 | 0.10 | 0.20 |
| | Time | 43.83 | 28.99 | 128.46 |
| | Nfg | 1766 | 1295 | 1734 |
| MR/LSTN | AAE | 5.09 | 2.24 | 3.68 |
| | EPE | 0.30 | 0.10 | 0.20 |
| | Time | 4.91 | 6.09 | 20.35 |
| | Nfg | 206 | 277 | 271 |
| FMG/LSTN | AAE | 5.09 | 2.24 | 3.39 |
| | EPE | 0.30 | 0.10 | 0.17 |
| | Time | 3.74 | 3.57 | 9.79 |
| | Nfg | 164 | 153 | 141 |

Table 5: Experimental results for Model 4.

| | | LSTN | MR/LSTN | FMG/LSTN |
|---|---|---|---|---|
| Model 1 | Total time (s) | 36.76 | 6.03 | 2.07 |
| | Total Nfg | 1687 | 260 | 94 |
| Model 2 | Total time (s) | 61.06 | 18.58 | 7.40 |
| | Total Nfg | 2213 | 567 | 252 |
| Model 3 | Total time (s) | 97.77 | 20.03 | 6.92 |
| | Total Nfg | 2904 | 587 | 219 |
| Model 4 | Total time (s) | 201.28 | 31.35 | 17.00 |
| | Total Nfg | 4795 | 754 | 458 |
| **All models** | **Total time (s)** | **396.87** | **75.99** | **33.39** |
| | **Total Nfg** | **11599** | **2168** | **1023** |

Table 6: Global characteristics of LSTN, MR/LSTN and FMG/LSTN for optical flow models on all the three images.

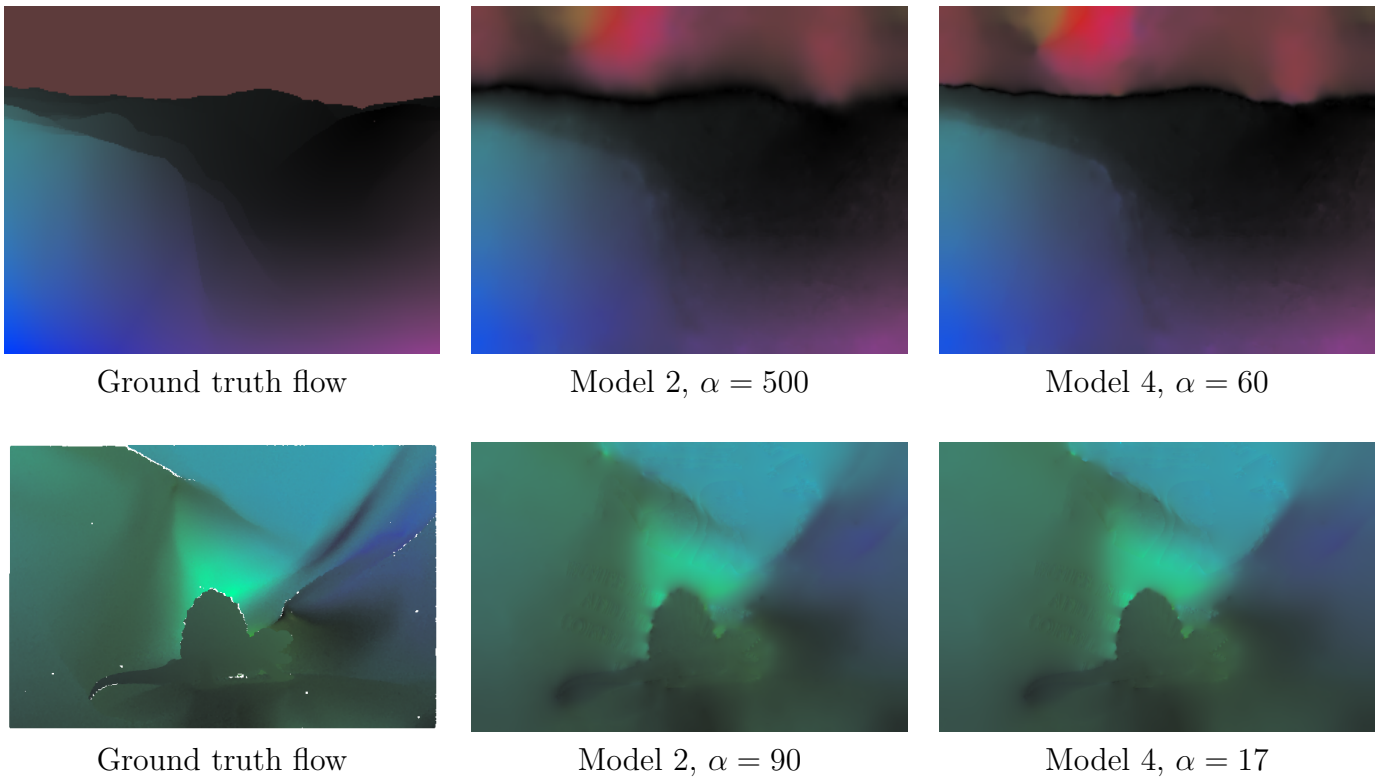


Figure 3: Colored-based representation of the optical flows for the Yosemite and Dimetrodon flows. On the left column the ground truth flow is shown. Middle and right columns show, respectively, the estimated flow for models 2 and 4.

## Image Credits

The Yosemite sequence was provided by D. L. Lynn Quam and SRI. The Dimetrodon images were provided by the Middlebury benchmark database [1].

## References

[1] S. Baker, D. Scharstein, J.P. Lewis, S. Roth, M.J. Black, and R. Szeliski, *A database and evaluation methodology for optical flow*, International Journal of Computer Vision, 92 (2011), pp. 1–31. http://dx.doi.org/10.1007/s11263-010-0390-2.

[2] S. Gratton, A. Sartenaer, and P. L. Toint, *Recursive trust region method for multiscale nonlinear optimization*, SIAM Journal on Optimization, 19 (2008), pp. 414–444. http://dx.doi.org/10.1137/050623012.

[3] B. Horn and B. Schunk, *Determining optical flow*, Artificial Intelligence, 20 (1981). http://dx.doi.org/10.1016/0004-3702(81)90024-2.

[4] E. Kalmoun, L. Garrido, and V. Caselles, *Line search multilevel optimization as computational methods for dense optical flow*, SIAM Journal of Imaging Sciences, 4 (2011), pp. 695–722. http://dx.doi.org/10.1137/100807405.

[5] E. M. Kalmoun, H. Köstler, and U. Rüde, *3D optical flow computation using a parallel variational multigrid scheme with application to cardiac C-arm CT motion*, Image and Vision Computing, 25 (2007), pp. 1482–1494. http://dx.doi.org/10.1016/j.imavis.2006.12.017.

[6] J. J. Moré and D. J. Thuente, *Line search algorithms with guaranteed sufficient decrease*, ACM Transactions on Mathematical Software, 20 (1994), pp. 286–307. http://dx.doi.org/10.1145/192115.192132.

[7] S. G. Nash, *Preconditioning of truncated-newton methods*, SIAM Journal on Scientific and Statistical Computing, 6 (1985), pp. 599–616. http://dx.doi.org/10.1137/0906042.

[8] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999. http://dx.doi.org/10.1007/b98874.

[9] Z. Wen and D. Goldfarb D., *A Line Search Multigrid Method for Large-Scale Convex Optimization*, tech. report, Department of IEOR, Columbia University, 2007. http://dx.doi.org/10.1137/08071524X.

[10] D. Xie and T. Schlick, *Efficient implementation of the truncated-Newton algorithm for large-scale chemistry applications*, SIAM Journal on Optimization, 10 (1999), pp. 132–154. http://dx.doi.org/10.1137/S1052623497313642.