

Published in Image Processing On Line on YYYY-MM-DD. ISSN 2105-1232 © YYYY IPOL & the authors CC-BY-NC-SA This article is available online with supplementary materials, software, datasets and online demo at http://www.ipol.im/pub/pre/44/

PREPRINT July 4, 2014

An Implementation of Multiscale Combined Local-Global Optical Flow

Jorge Jara¹, Mauricio Cerda², José Delpiano³, Steffen Härtel²

¹ DCC, SCIAN-Lab, BNI, University of Chile, Chile (jjara@dcc.uchile.cl) ² SCIAN-Lab, ICBM, BNI, University of Chile, Chile (mauriciocerda,shartel@med.uchile.cl) ³ University of the Andes, Chile (jdelpian@uandes.cl)

Abstract

Optical Flow (OF) approaches for motion estimation calculate vector fields which determine the apparent velocities of objects in time varying image sequences. OF was introduced in 1981 by Horn and Schunck (HS) with two basic assumptions: "brightness value constancy" and "smooth variation" of the motion field across the image space. At about the same time, Lucas and Kanade (LK) presented a method for motion estimation between images, considering constant motion patterns for image patches. While the HS method is aimed to solve the problem with a smooth flow field over the entire image -global approach- the LK method can produce homogeneous piece-wise motion field "patches" -local approach-. Several variations of the original HS- and LK-OF approaches have been published. Here we present the *combined local-global* (CLG) approach of Bruhn et al. which encompasses properties of both HS- and LK-OF models. CLG-OF aims to improve the accuracy of the OF motion field for small-scale variations while retaining the HS-OF benefits of dense and smooth vector fields.

A CLG-OF implementation is provided for 2D images, using a multiscale strategy and with two numerical solvers available: Successive Over-Relaxation and the faster Pointwise-Coupled Gauss-Seidel, proposed also by Bruhn et al. It must be noted that the algorithm works on greyscale (single channel) images. Thus, color images will be converted prior to the OF computation.

Source Code

The source code (ANSI C), its documentation, and the online demo are accessible at the IPOL web part of this article¹.

Supplementary Material

Sample images and the CLG-OF demo are available here².

¹http://www.ipol.im/pub/pre/44/ ²http://www.ipol.im/pub/pre/44/

1 Introduction

In this article we describe the combined local-global optical flow approach introduced by Bruhn et al. [2]. A review of the OF approaches from Horn & Schunck and Lucas & Kanade is also provided, as their constraints have been encompassed by the CLG-OF formulation. We also present the description for the CLG-OF numerical solution, together with (i) a multiscale strategy to increase the detectable motion range of OF methods, and (ii) two numerical solvers: Successive Over-Relaxation and the faster Pointwise-Coupled Gauss-Seidel.

The article is organized as follows. In the remainder of this section, a brief overview of HS-, LK- and CLG-OF methods is presented, together with a standard multiscale approach. Section 2 introduces the CLG-OF numerical solution, considering the discrete equations and the two solvers implemented. Section 3 presents the implemented algorithm pseudocode, function descriptions, and complexity analysis. Section 4 provides example results of the presented implementation, together with error evaluations in well known OF database sequences.

It must be noted that, even when the CLG-OF parameters are set to equal the HS-OF model (by setting $\rho = 0$), the results from this implementation often yield worse error metrics compared to other IPOL HS-OF versions. Other than the number of scales and parameter values, we have found that the main reason accounting for these difference between is the stage at which the linearization (image warping) is performed. Other HS-OF IPOL implementations perform late linearization, while ours performs early, as in the original article of Bruhn et al. (see Subsection 1.4). It is known that late linearization improves OF results for large displacements; however, it was not originally proposed for the HS or CLG methods. We opted for keeping the implementation close to the original approach.

1.1 Horn-Schunck Global Approach

Let I = I(x, y, t) the input image sequence (two or more images), and $\underline{V} = [u(x, y, t), v(x, y, t), 1]^T$ the optical flow vector field. The HS model assumes that the image total brightness level is constant across time, and that the estimated flow vectors vary smoothly across the image space. The brightness constancy assumption is expressed by making the time derivative $I_t = dI/dt = 0$. The HS-OF [8] is defined from a first-order Taylor expansion (linearization):

$$I(x + u, y + v, t + \Delta t) - I(x, y, t) = 0 \implies I_x u + I_y v + I_t = 0.$$
 (1)

Integrating over the image domain Ω , an "energy functional" is defined as

$$E_{HS}(\underline{V}) = \int_{\Omega} (I_x u + I_y v + I_t)^2 + \alpha (|\nabla u|^2 + |\nabla v|^2) dx dy,$$
(2)

with I_x, I_y the spatial derivatives, and $|\nabla u|, |\nabla v|$ the norms of the gradient vectors for u and $v, \nabla u$ and ∇v , respectively. $\alpha(|\nabla u|^2 + |\nabla v|^2)$ acting as a smoothing constraint for the OF field, since large variations in <u>V</u> account for an increase the magnitude of ∇u and ∇v . The weight coefficient $\alpha > 0$ is henceforth called **global regularization coefficient**. Higher values of α yield more homogeneous fields, while lower values allow more dissimilar displacement vectors. It must be noted that this coefficient is denoted by α^2 in the HS-OF article [8], instead of α in the CLG-OF article [2].

1.2 Lucas-Kanade Local Approach

The LK approach [10] assumes that the flow is constant within a neighborhood of size ρ , computing the optical flow for all the pixels in that neighborhood, by the least squares criterion:

$$E_{LK}(u,v) = G_{\rho} * (I_x u + I_y v + I_t)^2.$$
(3)

where G_{ρ} * denotes convolution with a Gaussian kernel of size ρ . A minimum (u, v) for E_{LK} satisfying $\partial_u E_{LK} = 0$ and $\partial_v E_{LK} = 0$ yields the following system of equations:

$$\begin{bmatrix} G_{\rho} * (I_x^2) & G_{\rho} * (I_x I_y) \\ G_{\rho} * (I_x I_y) & G_{\rho} * (I_y^2) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -G_{\rho} * (I_x I_t) \\ -G_{\rho} * (I_y I_t) \end{bmatrix}.$$
(4)

If the image gradients are not zero then the system matrix is invertible, yielding a unique solution.

1.3 Bruhn et al. Combined Local-Global Approach

Bruhn et al. [2] defined the CLG-OF by an integral functional that encompasses both the HS and the LK approaches. This combined approach aims to produce dense flow fields (characteristic of the global approaches) which are robust against noise, by employing smoothing terms based on both the global and local approaches.

Let $\nabla_3 I = [I_x, I_y, I_t]^T$, then the HS-OF energy functional can be written as:

$$E_{HS}(\underline{V}) = \int_{\Omega} (\underline{V}^T (\nabla_3 I \nabla_3 I^T) \underline{V} + \alpha (|\nabla u|^2 + |\nabla v|^2)) dx dy.$$
(5)

The CLG-OF energy functional introduces a smoothing term in $\nabla_3 I$. It is defined as

$$E_{CLG}(\underline{V}) = \int_{\Omega} (\underline{V}^T J_{\rho}(\nabla_3 I) \underline{V} + \alpha (|\nabla u|^2 + |\nabla v|^2)) dx dy,$$
(6)

with $J_{\rho}(\nabla_3 I) = G_{\rho} * (\nabla_3 I \nabla_3 I^T)$. J_{ρ} acts as a **local spatio-temporal derivative smoothing** term, defined as a convolution * between a bi-dimensional Gaussian kernel G_{ρ} , with standard deviation ρ , with the matrix $(\nabla_3 I \nabla_3 I^T)$. If $\rho = 0$ no local smoothing occurs, making the CLG-OF functional equal to the HS-OF. If $\alpha = 0$ the functional becomes equivalent to the LK-OF.

1.4 Multiscale Optical Flow

In order to fulfill the OF assumption, HS and CLG approaches impose a linearization of the brightness constancy constraint, thus the vector field \underline{V} is required to be small. This can be a problem for image sequences with large object displacements that can appear as discontinuities and lead to erroneous solutions. To address this issue, multiscale (MS) strategies have been proposed [4, 11], using a coarse-to-fine approach to successively compute more accurate OF vector fields.

The main idea of the MS strategy is to compute an image pyramid with decreasing size for each image in the set where OF is going to be estimated (typically two frames, I_1 and I_2 , at times t_1 and t_2). Starting from the smallest (coarsest) image of the pyramid, the OF is computed, and for the next (finer) scales, the coarse estimation is used to "warp" the second frame at t_2 , denoted as $I(x + \delta \underline{V})$ and compute the corresponding OF field. Thus, at each level of the pyramid the OF increment is computed ($\delta \underline{V}$), and the final OF estimation is the interpolated sum of the estimations from all the pyramid levels. The MS CLG-OF can be written as:

$$E_{CLG}(\underline{\delta V}^m) = \int_{\Omega} (\underline{\delta V}^T J_{\rho}(\nabla_3 I(x+\underline{\delta V})) \underline{\delta V} + \alpha (|\nabla \delta u|^2 + |\nabla \delta v|^2)) dxdy, \tag{7}$$

where $\delta \underline{V} = (\delta u, \delta v)$. In general, the MS strategy can be applied to any OF algorithm, by considering two additional parameters: the total number of scales and the scaling factor.

It is worth note that other IPOL OF methods available [11, 13] perform **late linearization** which yields better accuracy for larger displacements $\underline{d} = [d_x, d_y]$, by making $I_1(x, y) - I_2(x+d_x, y+d_y) = 0$ instead of the right side in Eq 1. In the MS CLG-OF, the warping of the second image is carried out only once per scale. As can be observed in those IPOL OF publications, an outer loop is used to solve the system several times, updating the value of $I(x + \delta \underline{V})$ in each iteration, improving the OF accuracy.

2 Numerical Solution Schemes

The minimum of the energy functional must satisfy the Euler-Lagrange equation, in the form of a system of partial differential equations

$$\alpha \Delta u - (J_{11}u + J_{12}v + J_{13}) = 0, \tag{8}$$

$$\alpha \Delta v - (J_{21}u + J_{22}v + J_{23}) = 0, \tag{9}$$

with J_{ik} denoting the elements of the matrix J_{ρ} . Neumann boundary conditions are assumed

$$\partial_n u = 0, \partial_n v = 0. \tag{10}$$

Although the CLG-OF can become equivalent to the HS-OF depending on the parameter values, different discretization schemes were used in the original articles, leading to slightly different results when both methods are applied to the same set of images. This also occurs for other OF approaches and numerical schemes, as shown by Delpiano et al. [5] and Hubený et al. [9] for fluorescent moving objects in microscopy images, and Sun et al [14] for a general overview.

For computing J_{ρ} , the following discrete derivatives are used:

$$I_x \approx I * \frac{1}{2h} [-1 \ 0 \ 1]_x |_{x_i, y_j, t_k}, \tag{11}$$

$$I_y \approx I * \frac{1}{2h} [-1 \ 0 \ 1]_y |_{x_i, y_j, t_k}, \tag{12}$$

$$I_t \approx I(x_i, y_j, t_{k+1}) - I(x_i, y_j, t_k),$$
(13)

with h the step size of the discrete domain (h = 1 for one pixel, for instance), and $A_l|_{x_i,y_j,t_k}$ the mask vector A applied along l axis at image position (x_i, y_j, t_k) (i.e. the central element of the mask corresponds to the pixel (x_i, y_j, t_k)). For time derivatives, a warped version of $I(x_i, y_j, t_{k+1})$ is used. Then, J_{ρ} is computed as follows:

- 1. $J_0 = \nabla_3 I \nabla_3 I^T;$
- 2. if $\rho = 0$ then $J_{\rho} = J_0$; otherwise
- 3. a square gaussian matrix G_{ρ} with standard deviation ρ is computed; and
- 4. $J_{\rho} = J_0 * G_{\rho}$.

The laplacian is computed by a convolution between the OF components and a kernel matrix M,

$$M = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

$$\Delta u_{x_i,y_j,t_k} \approx u * M|_{x_i,y_j,t_k},\tag{14}$$

$$\Delta v_{x_i, y_j, t_k} \approx v * M|_{x_i, y_j, t_k}.$$
(15)

The coupled system in Eqs. (8-9) can be written as:

$$\alpha u_{x_i, y_i, t_k} * M - J_{11} u_{x_i, y_i, t_k} = -(J_{13} + J_{12} v_{x_i, y_i, t_k}), \tag{16}$$

$$\alpha v_{x_i, y_j, t_k} * M - J_{22} v_{x_i, y_j, t_k} = -(J_{23} + J_{21} u_{x_i, y_j, t_k}).$$

$$(17)$$

This coupled system can be solved by multiple numerical schemes. For this work we implemented two iterative schemes to solve Eqs. (16-17), also used by Bruhn et al. [3, 4]: Successive Over-Relaxation (SOR), which solves sequentially u_{x_i,y_j,t_k} and then v_{x_i,y_j,t_k} , and Pointwise-Coupled Gauss-Seidel (PCGS), which simultaneously solves $(u_{x_i,y_j,t_k}, v_{x_i,y_j,t_k})$ at each pixel.

2.1 Successive Over-Relaxation Scheme

Equations (16-17) can be solved as linear system in the form Ax = b (with x being either u or v), where the matrix A is decomposed in the form A = D - L - U, with D a diagonal matrix and L, U lower and upper triangular matrices, respectively. From a given initial value x^0 , the equation is solved iteratively until a convergence criterion is reached. The k-th SOR iteration for x is given by $x^{k+1} = (D - L)^{-1}(Ux^k + b)$. In this case, two equations must be solved for the OF field components, (u, v) as described next. For these equations the subscript notation is changed from two indexes (e.g. u_{ij}) to one index (u_i) , in order to be consistent with the original CLG-OF article [4].

Let U and V be the matrices storing the x and y components of the OF field at given position i and iteration k, the values of the OF field (u_i, v_i) for the next iteration are computed as

$$u_{i}^{k+1} = (1-w)u_{i}^{k+1} + w \frac{\sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left(\sum_{j \in N_{l}^{-}(i)} u_{j}^{k+1} + \sum_{j \in N_{l}^{+}(i)} u_{j}^{k}\right) - (J_{12i}v_{i}^{k} + J_{13i})}{\sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} |N_{l}(i)| + J_{11i}}, \quad (18)$$

$$v_{i}^{k+1} = (1-w)v_{i}^{k+1} + w \frac{\sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left(\sum_{j \in N_{l}^{-}(i)} v_{j}^{k+1} + \sum_{j \in N_{l}^{+}(i)} v_{j}^{k}\right) - (J_{21i}u_{i}^{k+1} + J_{23i})}{2}, \quad (19)$$

$$= (1 - w)v_i^{k+1} + w \frac{1}{\sum_{l=1}^{2} \frac{\alpha}{h_l^2} |N_l(i)| + J_{22i}},$$
(19)

where $w \in (0, 2)$ is the relaxation term (w = 1 corresponds the Gauss-Seidel method), and $N_l(i)$ denotes the neighbors of pixel *i* in direction of axis l (l = 1 for *x*-axis, l = 2 for *y*-axis) belonging to Ω , making

$$N_l^+(i) = \{j \in N_l(i) | j > i\},$$
(20)

$$N_l^-(i) = \{j \in N_l(i) | j < i\}.$$
(21)

Given the definition of M, $N_l^+(i)$ represents two pixels: the right neighbor, and the lower pixel. Similarly, $N_l^-(i)$ are also two pixels: the left neighbor, and the upper pixel, and |N(i)| = 4.

2.2 Pointwise-Coupled Gauss-Seidel Scheme

A variant of the SOR scheme was proposed by Bruhn et al. [3, 7], aimed to improve its convergence properties for real-time implementation, called *Pointwise-Coupled Gauss-Seidel*. In this scheme, a coupled 2×2 system is solved at each pixel, and a synchronous update of the OF values for each pixel is performed. Starting from Eqs. (16-17), and defining the vector $q_i^{k+1} = (u_i^{k+1}, v_i^{k+1})$ at pixel *i*, the system can be written as:

$$M_i q_i^{k+1} = g_i^{k+1/2} \tag{22}$$

with

$$M_{i} = \begin{bmatrix} \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} |N_{l}(i)| + J_{11i} & J_{12i} \\ \\ J_{21i} & \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} |N_{l}(i)| + J_{22i} \end{bmatrix},$$

and

$$g_i^{k+1/2} = \begin{bmatrix} \sum_{l=1}^2 \frac{\alpha}{h_l^2} \left(\sum_{j \in N_l^-(i)} u_j^{k+1} + \sum_{j \in N_l^+(i)} u_j^k \right) - J_{13i} \\ \sum_{l=1}^2 \frac{\alpha}{h_l^2} \left(\sum_{j \in N_l^-(i)} v_j^{k+1} + \sum_{j \in N_l^+(i)} v_j^k + \right) - J_{23i} \end{bmatrix}$$

From Eq. (22) it can be shown that an iterative solution can be found by solving the following 2×2 linear system (here using Cramer's rule):

$$u_{i}^{k+1} = \frac{1}{\det(M_{i})} \det \left(\begin{bmatrix} \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left(\sum_{j \in N_{l}^{-}(i)} u_{j}^{k+1} + \sum_{j \in N_{l}^{+}(i)} u_{j}^{k} \right) - J_{13i} & J_{12i} \\ \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left(\sum_{j \in N_{l}^{-}(i)} v_{j}^{k+1} + \sum_{j \in N_{l}^{+}(i)} v_{j}^{k} + \right) - J_{23i} & \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left| N_{l}(i) \right| + J_{22i} \end{bmatrix} \right), \quad (23)$$
$$v_{i}^{k+1} = \frac{1}{\det(M_{i})} \det \left(\begin{bmatrix} \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left| N_{l}(i) \right| + J_{11i} & \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left(\sum_{j \in N_{l}^{-}(i)} u_{j}^{k+1} + \sum_{j \in N_{l}^{+}(i)} u_{j}^{k} \right) - J_{13i} \\ J_{21i} & \sum_{l=1}^{2} \frac{\alpha}{h_{l}^{2}} \left(\sum_{j \in N_{l}^{-}(i)} v_{j}^{k+1} + \sum_{j \in N_{l}^{+}(i)} v_{j}^{k} + \right) - J_{23i} \end{bmatrix} \right). \quad (24)$$

Although the computations for u_i^{k+1} and v_i^{k+1} can be performed sequentially (i.e. one iteration loop for u, then another for v), the alternating computation for both can prevent problems for small α values that make the term $q^T J_{\rho}(\nabla_3 f) q$ dominate in the solution. This scheme has been also described with other solver approaches such as multigrid CLG-OF [3].

3 Algorithm

This section presents the MS CLG-OF implementation as follows: Algorithms (1-2) depict the main algorithm pseudo-code; constants and function descriptions are given as a complement to the source code documentation; finally, a time complexity analysis of the algorithm is presented.

3.1 Constants

- JROWS = 3 Number of rows of the J_{ρ} matrix associated to a given image position.
- JCOLS = 3 Number of columns of the J_{ρ} matrix associated to a given image position.
- $MIN_GAUSSIAN_SIZE = 3$ Minimum gaussian kernel size allowed for smoothing images/frames. If the computed kernel size for ρ,σ is lower than this value, no smoothing is performed.
- $EPS = 1 \cdot 10^{-12}$ Precision threshold (epsilon) value for the numerical relaxation computations.
- $MIN_ERROR = 1 \cdot 10^{-4}$ Convergence threshold for the main iteration loop. It is compared to the mean variation of the OF vectors between iterations (the mean variation value must be lower). At the iteration k, for each pixel i (N total), the variation is computed as $\sqrt{\frac{\sum_i (u_i^k u_i^{k-1})^2 + (v_i^k v_i^{k-1})^2}{N}}$.

Algorithm 1: calcCLG_OF Input

- I_1, I_2 , input images for the OF computation.
- *uOut*, *vOut*, output pointers to the OF vector field.
- *nRows*, *nCols*, number of image/OF field rows and columns, respectively.
- $\alpha > 0$, global regularization coefficient value.
- $\rho > 0$, local derivative regularization coefficient value.
- *numIterations*, number of relaxation iterations for the OF vector field.
- coupledMode > 0, binary flag to use either PCGS (1) or SOR (0) as relaxation scheme.
- wFactor > 0, relaxation factor (applies only for SOR).
- verbose > 0, binary flag to either display debug messages (1), or not (0).

```
1 Initialize 2D optical flow arrays pointed by u, v;
 2 foreach i, j in [0, nRows - 1] \times [0, nCols - 1] do
       J(i,j) \leftarrow J_0(i,j) = \nabla_3 I \nabla_3 I^T(i,j);
 3
 4 end
 5 if \rho > 0 then
       filterSize \leftarrow | DEFAULT_GAUSSIAN_WINDOW_SIZE \times \rho | + 1;
 6
       // J(i,j) = J_0(i,j) * G_{\rho} with kernel G_{\rho} of filterSize \times filterSize elements.
       for i, j in [0, nRows - 1] \times [0, nCols - 1] do
 \mathbf{7}
           smooth(J(i, j), filterSize, \rho);
 8
       end
 9
10 end
11 i \leftarrow 1
12 while i \leq numIterations AND error decrease \geq MIN\_ERROR do
       // Pointwise-Coupled Gauss-Seidel relaxation.
       if coupledMode == 1 then
\mathbf{13}
           relax system for *u, *v using Eq. (22);
\mathbf{14}
       end
15
       // Successive Over-Relaxation.
       if coupledMode == 0 then
\mathbf{16}
           relax system for *u, *v using Eq. (19);
17
       end
18
19 end
20 uOut, vOut \leftarrow u, v;
21 return uOut, vOut
```

Algorithm 2: calcMSCLG_OF Input

- All of the input parameters from algorithm calcCLG_OF.
- nScales > 1, total number of scales in the pyramid.
- scaleFactor < 1, scaling factor between scales.

```
1 filterSize \leftarrow 2 \times | DEFAULT_GAUSSIAN_WINDOW_SIZE \times \sigma | + 1;
   // I_i(i,j) = I_i(i,j) * G_{\sigma} with kernel G_{\sigma} of filterSize \times filterSize elements.
 2 if filterSize \geq MIN_FILTER_SIZE_SIGMA then
       smooth(I_1, filterSize, \sigma);
 3
       smooth(I_2, filterSize, \sigma);
 \mathbf{4}
 5 end
 6 Initialize 2D arrays nxx and nyy with nScales elements;
 7 nxx[0] \leftarrow nRows;
 s nxx[i] \leftarrow (int)(nxx[i-1] * scaleFactor + 0.5);
 9 nyy[0] \leftarrow nCols;
10 nyy[i] \leftarrow (int)(nyy[i-1] * scaleFactor + 0.5);
11 Initialize nScales 2D image arrays I_{1s} and I_{2s} with Gaussian pyramid;
12 Initialize empty nScales 2D OF arrays us and vs at each scale;
13 for s \leftarrow nScales - 1, s > 0, s \leftarrow s - 1 do
       // Warp I_2 with the current OF estimation of (us[s], vs[s]).
       image2warped \leftarrow bicubic_interpolation_warp(I_{2s}[s], us[s], vs[s], nxx[s], nyy[s]);
\mathbf{14}
       // Compute OF at this scale.
       us[s], vs[s] \leftarrow calcCLG_OF(I_{1s}, image2warped, nRows, nCols, \alpha, \rho, numIterations,
15
       coupledMode, wFactor, verbose, nScales, nScales, scaleFactor);
16
       if s == 0 then
\mathbf{17}
           break;
18
       end
19
       // Project current OF estimation into the next scale.
       zoom_in(us[s], us[s-1], nxx[s], nyy[s], nxx[s-1], nyy[s-1]);
\mathbf{20}
       zoom_in(vs[s], vs[s-1], nxx[s], nyv[s], nxx[s-1], nyv[s-1]);
\mathbf{21}
       // Update the current OF solution.
       us[s-1][i] \leftarrow us[s-1][i]/scaleFactor
\mathbf{22}
       vs[s-1][i] \leftarrow vs[s-1][i]/scaleFactor
23
24 end
25 return us[0], vs[0]
```

3.2 Function Description

This section summarizes the purpose and parameters of the implemented functions. For the input/output and MS functions the implementation contributed in Meinhardt-Llopis et al. [11] is used. Pointer variables to 1- and 2-dimensional arrays are denoted with the prefixes * and **, respectively.

• calcCLG_OF (*image1, *image2, *uOut, *vOut, nCols, nRows, iterations, alpha, rho, wFactor, verbose, coupledMode)

CLG-OF computation function implements Algorithm 1. *image*1 and *image*2 point to the input images for the OF (in the form of 1D arrays of $nRows \times nCols$ elements each). uOut, vOutare output pointer variables for the vertical and horizontal OF components, of $nCols \times nRows$ elements each. *iterations*, *alpha* and *rho* give the input values for maximum number of iterations, global smoothing coefficient α , and local spatio-temporal smoothing coefficient ρ , respectively. The flag *coupledMode* sets the iteration mode to SOR (0) or PCGS (1). When using SOR, $wFactor \in (0, 2)$ is the relaxation factor. The *verbose* flag enables verbose output.

- calcMSCLG_OF(*image1, *image2, *uOut, *vOut, nCols, nRows, iterations, alpha, rho, sigma, wFactor, nScales, scaleFactor, coupledMode, verbose)
 MS CLG-OF computation function implements Algorithm 2, by calling calcCLG_OF with the corresponding input parameters (which have the same names). sigma is the gaussian image smoothing parameter. scaleFactor ∈ (0, 1) is the pyramid resizing factor, e.g. scaleFactor = 0.5 accounts for half size reductions. nScales gives the number of scales, provided that the final size of the coarsest pyramid level is large enough to compute the derivatives (if not, the maximum scale number is computed and used). The verbose flag enables verbose output.
- $SOR_at(**u, **v, **J, i, j, alpha, wFactor)$ Solves the CLG-OF equations for the u, v components at pixel location [i, j], according to Eq. (19). alpha gives the value of the global smoothing coefficient. J points to the arrays with the computed (and possibly smoothed) derivatives. $wFactor \in (0, 2)$ is the relaxation factor.
- relaxSOR(**u, **v, **J, nRows, nCols, alpha, wFactor)SOR iteration for solving the CLG-OF equations by calling the function SOR_{-at} for each pixel (with equally named input parameters), and Eq. (10) for boundary conditions. The horizontal and vertical OF components are u, v of $nRows \times nCols$ elements each.
- relaxPointwiseCoupledGaussSeidel(* * u, * * v, * * J, nRows, nCols, alpha) PCGS relaxation iteration for solving the CLG-OF equations. Updates the value of the OF vector field components u, v according to Eqs. (23-24), using Cramer's rule. If the discriminant value is lower than EPS a SOR iteration is performed instead. Eq. (10) is used for boundary conditions. Function parameters are the same described for calcCLG_OF and SOR_at.
- boundaryCondition(**u, **v, nRows, nCols)Performs SOR and PCGS iterations at the boundary pixels of u, v (rows 0 and nRows - 1, columns 0 and nCols - 1), by copying values from the non-boundary neighbor pixels.

3.2.1 Utility functions

- computeDerivatives(* * image1, * * image2, * * df dx, * * df xy, nRows, nCols)
 - Computes the discrete spatial derivatives for the input time frames image1, image2 of $nRows \times nCols$ elements. The derivatives are stored in the arrays pointed by df dx, df dy (x, y coordinates) of $nRows \times nCols$ elements each. The derivatives are computed by using a 3×1 stencil with values [-0.5, 0.0, +0.5] (forward differences).

• computeJTensor(**dfdx,**dfdy,**dfdt,**J,nRows,nCols)

Computes and stores the values of the tensor matrix J at each image pixel. df dx, df dy, df dt point to the derivative matrices of the input images, containing $nRows \times nCols$ elements each. The result is pointed by J, with $nRows \times nCols \times JROWS \times JCOLS$ elements. For a given pixel [i, j] the associated tensor is a 3×3 matrix, with elements

$$\begin{split} J[1][1][i][j] &= I_x[i][j] * I_x[i][j] \\ J[1][2][i][j] &= I_x[i][j] * I_y[i][j] \\ J[1][3][i][j] &= I_x[i][j] * I_t[i][j] \\ J[2][2][i][j] &= I_y[i][j] * I_y[i][j] \\ J[2][3][i][j] &= I_y[i][j] * I_t[i][j] \\ J[3][3][i][j] &= I_t[i][j] * I_t[i][j], \end{split}$$

being symmetric with respect to its diagonal, so only the upper triangular part is stored.

• matrixSmooth(**matrix, nRows, nCols, kernelSigma)

Performs a gaussian smoothing over a given input matrix of $nRows \times nCols$ elements, by applying a square kernel parametrized by its standard deviation kernelSigma. It is called from calcCLG_OF for smoothing the input images and the derivatives matrix, according to the values of sigma and rho, respectively. The kernel size is computed in a way that ensures odd values (i.e. symmetric gaussian kernels with a single central value as maximum), greater than $MIN_GAUSSIAN_SIZE$.

- correctIndex(p, size)Given an index p, returns p if $p \in [0, size-1]$. If not, returns its closest number from [0, size-1].
- lin2by2det(a, b, c, d)

Computes and return the determinant of a given 2×2 matrix, $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$. The determinant D is computed as D = ad - bc.

• pMatrix(nRows, nCols)

Initializes and returns a 2D pointer array of a given number of rows and columns, nRows and nCols respectively.

- freePmatrix(* * mat, nRows) Deallocates a given 2D pointer matrix mat, given as a pointer to an array of nRows pointers.
- xmalloc(size)

Array memory allocation function, for a given memory size.

3.3 Time Complexity

Assume a square image of N_s pixels at each pyramid level s (N_0 is the original image level), and filter kernel sizes up to $k \ll N_s$ (for smoothing and derivatives). The overall cost is the sum of the cost of building the pyramid by filtering operations $O(N_s k)$ each, plus the OF iterations at each pyramid level. OF iterations by matrix inversion can be computed in $O(N_s^2)$ time, but iterative methods such as SOR and PCGS perform closer to $O(CN_s)$, where C depends on the input. However, $C \ll N_0$, for instance $C \ll 3000$ for the Middlebury dataset (Table 2). Thus, as the cost is dominated by the finer scale (of size N_0), the running time of the MS CLG-OF algorithm is $O(CN_0)$.



Figure 1: CLG-OF for a moving point source, simulating a fluorescent signal in a confocal microscopy image (adapted from Delpiano et al. [5]). **a,b**: time frames of a synthetic point signal (top), and their computed confocal PSF-convolved images (down), of 100×100 pixels each, for a confocal microscope with 60x water objective and wavelengths of 543/560 nm for excitation/emission. Pixel size is 107nm in x and y. The point displacement between the two frames is 3 pixels (321nm). **c**: CLG-OF field of the PSF-convolved images, computed with parameter values: $\alpha = 200$, $\rho = 5$, $\sigma = 0$, nScales = 1, *iterations* = 200, using PCGS solver. The OF vector color code is shown at the bottom left image.

4 Evaluation

Here we present examples of the implemented CLG-OF applied to different image sequences. First, in Subsection 4.1 we illustrate the effect of different values for the regularization coefficients α and ρ in the OF fields. Next, in Subsection 4.2 we present and briefly discuss results from synthetic sequences with known ground-truth OF, comparing the MS CLG-OF estimation error against examples from the original CLF-OF article and the Middlebury database.

4.1 Example Sequences

The first example, shown in Figure 1, is the CLG-OF field for a model structure in fluorescence microscopy, corresponding to a moving point signal (fluorescent protein, a-b). The signal was convolved with a theoretical point spread function (PSF, b-c) of a confocal microscope (a PSF acts as a smoothing function that blurs signals). See details of simulation and tests for optimal parameters in the work of Delpiano et al. [5]. Next, Figures 2-5 illustrate the effects of different values for α and ρ in the resulting OF fields.

4.2 Error Metrics

We tested the MS CLG-OF implementation with example image sequences from Bruhn et al. [4], shown in Figure 6, and from the Middlebury database [1], shown in Figure 7. In both cases, the Average Angular Error (AAE) and Average Endpoint Error (AEE) against ground-truth OF fields, as defined by Baker et al. [1], were measured. We also measured computation times, using a PC workstation with a 3.4 GHz Intel Core i7 4930K CPU and 64 GB of RAM, running the 64-bit Ubuntu Linux 12.04 operating system. The results are summarized in Tables 1&2.

4.3 Discussion

AAE and AEE metrics show that our implementation yields results similar to those of the original work from Bruhn et al. [4]. For a similar error margin in the examples, the PCGS iterative scheme performs 2 to 10 times faster than SOR, but with slightly worse error metrics.



Figure 2: Different parameter values for CLG-OF and their corresponding OF fields for the synthetic point source fluorescence image from Figure 1, with an horizontal displacement of three pixels. I_1, I_2 are the input images. Computed with parameter values: $\sigma = 0$, nScales = 1, iterations = 200, using PCGS solver. Vectors are color coded as in Figure 1.c.



Figure 3: Different parameter values for CLG-OF and their corresponding OF fields for the Lena image (256×256 pixels, grey-scale version), with an horizontal displacement of one pixel. I_1, I_2 are the input images. Computed with parameter values: $\sigma = 0$, nScales = 1, iterations = 200, using PCGS solver. Vectors are color coded as in Figure 1.c.



Figure 4: Different parameter values for MS CLG-OF and their corresponding OF fields for the baboon rotation (256×256 pixels, grey-scale version) images. I_1, I_2 are the input images. Computed with parameter values: $\sigma = 0$, nScales = 3, scaleFactor = 0.65, iterations = 200, using PCGS solver. Vectors are color coded as in Figure 1.c.



Figure 5: Different parameter values for MS CLG-OF and their corresponding OF fields for the spiral image with an homography transformation. I_1, I_2 are the input images. Computed with parameter values: $\sigma = 0$, nScales = 3, scaleFactor = 0.65, iterations = 200, using PCGS solver. Vectors are color coded as in Figure 1.c.



Figure 6: Results of the MS CLG-OF implementation with the SOR and PCGS numerical schemes, in selected images from the original GLG-OF article [4]. Computed with parameter values: $\alpha = 200$, $\rho = 5.0$, $\sigma = 0.85$, scaleFactor = 0.65, MIN_ERROR = 0.0001, iterations = 10000, and wFactor = 1.8 (SOR only). The number of scales nScales was dependent in the image size, thus chosen as 7, 5, and 6 respectively. Convergence was achieved before the indicated number of iterations, counted at the original image level, as shown in Table 1. The vector color code is shown at the top right image.



Figure 7: Results of the MS CLG-OF implementation with the SOR and PCGS numerical schemes, in selected images from the Middlebury dataset [1]. Computed with parameter values: $\alpha = 200$, $\rho = 5.0$, $\sigma = 0.85$, nScales = 7, scaleFactor = 0.65, $MIN_ERROR = 0.0001$, iterations = 10000, and wFactor = 1.8 (SOR only). Convergence was achieved before the indicated number of iterations, counted at the original image level, as shown in Table 2. Vectors are color coded as in Figure 6.

	Marble blocks	Office	Yosemite					
SOR								
AAE (°)	9.99	4.71	3.11					
AEE	0.30	0.10	0.17					
Iterations	875	2017	2118					
Time $[s]$	17.01	5.42	11.51					
PCGS								
AAE (°)	9.99	4.82	3.09					
AEE	0.30	0.10	0.17					
Iterations	190	276	1316					
Time $[s]$	9.19	1.7	6.51					
SOR by Bruhn et al. [4]								
AAE ($^{\circ}$)	5.30	4.33	2.64					

Table 1: Comparison of AAE and AEE measurements of the implemented MS CLG-OF with the SOR and PCGS numerical schemes, against the original reported results by Bruhn et al. [4] for the images shown in Figure 6. Available ground-truth was taken into account at pixels with OF vector magnitude less than 1000 in Marble blocks and Office sequences, and other than 0.0406 (clouds displacement) for Yosemite sequence. From Marble blocks and Office the first two frames were used.

	Dimetro.	Grove2	Grove3	Hydrangea	RubberWhale	Urban2	Urban3	Venus	
MS CLG SOR									
AAE (°)	4.3	4.56	9.79	4.09	11.94	7.66	15.51	10.73	
AEE	0.22	0.31	1.31	0.6	0.37	1.0	1.65	0.65	
Iterations	589	1713	1118	2772	814	3443	547	1398	
Time $[s]$	11.26	37.75	24.9	42.57	14.25	71.42	13.52	15.46	
MS CLG PCGS									
AAE (°)	7.7	4.96	10.4	6.63	12.69	8.35	18.76	11.13	
AEE	0.37	0.34	1.44	1.16	0.39	1.13	1.92	0.68	
Iterations	90	113	116	56	207	184	135	216	
Time [s]	4.52	7.17	9.15	3.38	7.01	9.5	9.05	5.58	

Table 2: AAE and AEE measurements for the implemented MS CLG-OF with the SOR and PCGS numerical schemes, for the Middlebury database images shown in Figure 7. Available ground-truth was taken into account at pixels with OF vector magnitude less than 1000 for all the sequences.

Although some parameter values of the MS CLG-OF implementation were kept fixed for the examples, it becomes clear that a fine tuning of α , ρ , *nScales*, *wFactor*, and *scaleFactor* is required for optimum results. For instance, by varying the value of wFactor between 1.8 and 1.9, the AAE can be further improved in $1^{\circ}-2^{\circ}$ (not shown). Figures 2-5 and the variability of the error metrics illustrate this issue, as well as reported results from Delpiano et al. [5], Hubený et al. [9], Meinhardt-Llopis et al. [11] and Sun et al. [14].

We remind that, other than the chosen numerical scheme and parameter values, the use of late linearization (not implemented here) allows a better handling of larger displacements $\underline{d} = [d_x, d_y]$, by making $I_1(x,y) - I_2(x + d_x, y + d_y) = 0$ instead of the right side in Eq 1. From the implementation point of view, this implies two nested iteration loops: one for fixed values of d iterating over V, and one for d performing multiple warpings at each scale. Significant improvements can be achieved this way, but still a numerical solver such as SOR or PCGS must be used for the V loop.

Acknowledgments 5

Funding: CONICYT PhD scholarship (JJ), FONDECYT 3140447 (MC) & 1120579 (SH,JJ), US-LACRN (MC), and ICM P09-015-F (BNI). The authors would like to thank Haldo Spontón and Juan Cardelino for supporting the preparation of this work, and the IPOL editor and reviewers for their valuable insights and suggestions.

Image Credits

All of the OF field and error measurement images (Figures 1,7) by the authors, except the ground truth ("True flow") images by IPOL.

(Figures 1,2) by the authors.

(Figure 3) from the USC-SIPI Image Database³.

(Figure 4) from the USC-SIPI Image Database⁴.

(Figure 5) by Mark Dow, "Logarithmic spirals, waves and tilings"⁵.

(Figure 6) from Image Sequence Server (Group Prof. Dr. H.-H. Nagel) at Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe⁶.



(Figure 6) by Lynn Quam.

database [1].

```
(Figure 7) from the Middlebury benchmark
```

³http://sipi.usc.edu/database/database.php?volume=misc

⁴http://sipi.usc.edu/database/database.php?volume=misc

 $^{^{5}}$ http://lcni.uoregon.edu/~dow/Geek_art/Logarithmic_spirals/Logarithmic_spirals_waves_tilings. html

⁶http://i21www.ira.uka.de/image_sequences/

References

- Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski, *A Database and Evaluation Methodology for Optical Flow*, International Journal of Computer Vision, vol. 92, no. 1, pp. 1-31, 2011. http://dx.doi.org/10.1007/s11263-010-0390-2.
- [2] Andrés Bruhn, Joachim Weickert, and Christoph Schnörr, Combining the Advantages of Local and Global Optic Flow Methods, in Proceedings of the 24th DAGM Symposium, 2002, Zurich, Switzerland, pp. 454-462. http://dx.doi.org/10.1007/3-540-45783-6_55.
- [3] Andrés Bruhn, Joachim Weickert, Christian Feddern, Timo Kohlberger, and Christoph Schnörr, Variational Optical Flow Computation in Real Time, IEEE Transactions on Image Processing, vol. 14, no. 5, pp. 608-615, 2005. http://dx.doi.org/10.1109/TIP.2005.846018.
- [4] Andrés Bruhn, Joachim Weickert, Lucas/Kanade Meets Horn/Schunck: Combining Local Global Optic Flow Methods, International Journal of Computer Vision, vol. 61, no. 3, pp. 211-231, 2005. http://dx.doi.org/10.1023/B:VISI.0000045324.43199.43.
- [5] José Delpiano, Jorge Jara, Jan Scheer, Omar A. Ramírez, Javier Ruiz-del-Solar, and Steffen Härtel, Performance of Optical Flow Techniques for Motion Analysis of Fluorescent Point Signals in Confocal Microscopy, Machine Vision and Applications, vol. 23, no. 4, pp. 675-689, 2012. http://dx.doi.org/10.1007/s00138-011-0362-8.
- [6] Ben Galvin, Brendan McCane, Kevin Novins, David Mason, and Steven Mills, *Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms*, in Proceedings of the British Machine Vision Conference (BMC), 1988, Southampton, England, pp. 195-204. http://dx.doi.org/10.5244/C.12.20.
- [7] Wolfgang Hackbusch, Iterative Solution of Large Sparse Systems of Equations, Springer, New York, 1993. ISBN 978-0-387-94064-9.
- [8] Berthold K. P. Horn, Brian G. Schunck, *Determining Optical Flow*, Artificial Intelligence, vol. 17, pp. 185-203, 1981. http://dx.doi.org/10.1016/0004-3702(81)90024-2.
- [9] Jan Hubený, Vladimír Ulman, and Pavel Matula, Estimating Large Local Motion in Live-Cell Imaging Using Variational Optical Flow, in Proceedings of the 2nd International Conference On Computer Vision Theory And Applications (VISAPP), 2007, INSTICC, Barcelona, Spain, pp. 542-548. ISBN 978-972-8865-74-0.
- [10] Bruce D. Lucas and Takeo Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision., in Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI), 1981, Vancouver, BC, Canada, pp. 674-679.
- [11] Enric Meinhardt-Llopis, Javier Sánchez Pérez, and Daniel Kondermann, Horn-Schunck Optical Flow with a Multi-Scale Strategy, Image Processing On Line, vol. 2013, pp. 151-172, 2013. http://dx.doi.org/10.5201/ipol.2013.20.
- [12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Numerical Recipes in C, 2nd edition, Cambridge University Press, 1992. ISBN 0-521-43108-5.

- [13] Javier Sánchez Pérez, Nelson Monzón López, and Agustín Salgado de la Nuez, Robust Optical Flow Estimation, Image Processing On Line, vol. 2013, pp. 252-270, 2013. http://dx.doi.org/10.5201/ipol.2013.21.
- [14] Deqing Sun, Stefan Roth, and Michael J. Black., A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them, International Journal of Computer Vision, vol. 106, no. 2, pp. 115-137, 2014. http://dx.doi.org/10.1007/s11263-013-0644-x.