TRUST REGION VERSUS LINE SEARCH FOR COMPUTING THE OPTICAL FLOW

EL MOSTAFA KALMOUN[†] AND LUIS GARRIDO[‡]

Abstract. We consider the numerical treatment of the optical flow problem by evaluating the performance of the trust region method versus the line search method. To the best of our knowledge, the trust region method is studied here for the first time for variational optical flow computation. Four different optical flow models are used to test the performance of the proposed algorithm combining linear and nonlinear data terms with quadratic and TV regularization. We show that trust region often performs better than line search; especially in the presence of non-linearity and non-convexity in the model.

Key words. Optical flow, optimization, trust region, line search, truncated Newton, multiresolution

1. Introduction. Optical flow is a problem that consists in finding the two-dimensional field that represents the apparent motion of objects in a sequence of images. In recent years, many techniques for the computation of optical flow have been proposed in literature; see e.g. [8, 4, 5, 7]. Variational methods are a well known mathematical approach to compute dense optical flow estimations. In such context, the optical flow estimation problem is usually formulated as a minimization of an energy function, which is a weighted sum of two terms D and R. The first term D is a data term that comes from the motion modelling and is based on the conservation of some property during motion such as the gray-level and leads usually to the so-called optical flow constraint equation (OFCE). The second term R is a regularization term that allows to impose the assumption that the optic flow varies smoothly in space and ensures that the optic flow problem is well posed and can be formulated as a large-scale optimization problem:

$$\min_{w} f(w), \tag{1.1}$$

where $f(w) = D(w) + \alpha R(w)$, $w = (u, v)$ is the optical flow field and α is used to control the influence of the two terms D and R.

Traditionally, the problem (1.1) is solved by first computing the corresponding Euler-Lagrange equations, then discretizing and solving them by relaxation schemes. This approach which is called optimize-discretize has been commonly used for variational optical flow computation, e.g. [1, 2, 5, 7, 10]. A second approach, called discretize-optimize, is based on first discretizing the variational functional f and then solving the finite-dimensional problem using numerical optimization algorithms. We note that while this computational strategy has been used for many vision problems, not too much attention was paid to it in the context of optical flow. In [9], we have shown the competitiveness of the discretize-optimize strategy compared to the classical first approach. Three Newton-based optimization algorithms were superior to the Gauss-Seidel method when applied to the classical Horn-Schunck model [8]. In particular, truncated Newton (TN) method has been shown that it can be used as a suitable optimization algorithm for variational optical flow and has proved to better

[†]Department of Mathematics, Faculty of Science, King Khalid University, P.O. Box 9004, Abha, Saudi Arabia (ekalmoun@kku.edu.sa)

[‡]Departament de Matemàtica Aplicada i Anàlisi, Universitat de Barcelona, Barcelona, Spain (lluis.garrido@ub.edu)

performance than the quasi-Newton method. The method requires only the computation of the energy function and gradient values and thus has low memory requirements to solve large-scale optimization problems. We recall that this iterative method moves the current approximation w_k to the new one w_{k+1} by means of a step s_k which is obtained by solving approximately the Newton equation given by

$$\nabla^2 f(w_k)s = -\nabla f(w_k).$$

Depending on how the step s_k is computed and used, two broad classes of algorithms are distinguished: line search methods and trust region methods. Line search methods scale the step s_k by a factor α_k that approximately minimizes f along the line that passes through w_k in the direction s_k,

$$w_{k+1} = w_k + \alpha_k s_k.$$

On the other hand, trust region methods solve s_k within a certain region around w_k in which the algorithm trusts that s_k will be a good update.

As in various computer vision works, the line search strategy has been the focus of our previous paper [9] in the context of optical flow. However, recent interest is shown to the use of the trust region techniques in vision applications due to their strong convergence properties. For instance, the method was used to solve the image restoration [3], the tracking problem [12], the object pose from a single view problem [16], online inference for computer vision and robotics problem [17], logistic regression problem [11].

In the present paper we evaluate the performance of trust region methods and compare them with line search methods. To the best of our knowledge, the trust region method is studied here for the first time for variational optical flow computation. Four different optical flow models are used to test the performance of the proposed algorithm combining linear and nonlinear data terms with quadratic and TV regularization.

The paper is organized as follows: In Section 2, we recall the basics of line search and trust region methods and state the optimization algorithms used in this paper. In Section 3, we present the details used to implement the optical flow models and the optimization algorithms. In Section 4, we precise the measures we use in our experiments and we give and discuss the results. Finally, we conclude the paper in Section 5.

2. Line Search versus Trust Region Truncated Newton Methods.

Newton methods solve (1.1) with an iterative process that approximates, at each iteration w_k, the objective function f by means of a second order Taylor approximation q_k,

$$q_k(s) = f_k + g_k^T s + \frac{1}{2} s^T H_k s \tag{2.1}$$

where $f_k = f(w_k)$, $g_k = \nabla f(w_k)$ is the gradient of f, and $H_k = \nabla^2 f(w_k)$ is the Hessian.

Depending on how a step solution s_k is computed and exploited, two broad classes of algorithms are distinguished: line search methods and trust region methods. Line search methods compute s_k as the minimization of the unconstrained problem given by (2.1). The Newton step s_k is thus obtained by solving the linear system

$$H_k s = -g_k \tag{2.2}$$

The current solution w_k is then updated by scaling the step s_k by a factor α_k that approximately minimizes f along the line that passes through w_k in the direction s_k, $w_{k+1} = w_k + \alpha_k s_k$.

Trust-region methods solve (2.1) by restricting the search for s_k to some region $\mathcal{B}_k$ around the current iterate w_k in which the algorithm "trusts" that the model function q_k behaves like the objective function f. That is, the step s_k is obtained by solving (2.1) with $s \in \mathcal{B}_k$. The current iteration w_k is updated with s_k if this step produces a suitable improvement over the objective function f, $w_{k+1} = w_k + s_k$. Note that trust region methods choose the direction and length of the step simultaneously.

Let us now focus on a brief summary of the line search truncated Newton. We will afterwards detail the trust-region method.

2.1. Line search tuncated Newton. Solving exactly the linear system (2.2) will be very expensive for large-scale problems. Truncated Newton methods (TN) use rather an iterative method to find an approximate solution to (2.2). The method truncates the iterates as soon as a required accuracy is reached or whenever – in case when the Hessian matrix H_k is not positive definite – a negative curvature is detected. A non-positive H_k is in fact associated to a non-convex approximation $q_k(s)$. One of the most well known iterative method within TN methods is the Preconjugated Conjugate Gradient algorithm (PCG) due to its efficiency and small memory requirements. The PCG algorithm requires only to be able to explicitly compute the function and gradient values but not the Hessian. In the context of our paper, the process to find the step s_k is called inner iterations while the process to update w_k using the computed s_k is called outer iterations. For the line search truncated Newton method (LSTN), the inner (Algorithm 1) and outer (Algorithm 2) iterations are recalled from [9].

Algorithm 1 Preconditioned Conjugate Gradient (inner iterations of TN)

```
1: Initialization: $z_0 = 0$, $r_0 = -g_k$, $v_0 = M_k^{-1} r_0$, $p_0 = v_0$, $\epsilon = 10^{-10}$, $\zeta_k$(see (2.4))
2: for $j = 0$ to $max\_inner$ do
3:     // Singularity test
4:     if $\left(|r_j^T v_j| < \epsilon$ or $|p_j^T H_k p_j| < \epsilon\right)$ then
5:         exit with $s_k = z_j$ (for $j = 0$ take $s_k = -g_k$)
6:     end if
7:     $\alpha_j = r_j^T v_j / p_j^T H_k p_j \quad , \quad z_{j+1} = z_j + \alpha_j p_j$
8:     // Descent Direction Test replaces Negative Curvature test
9:     if $(g_k^T z_{j+1} \geq g_k^T z_j - \epsilon)$ then
10:        exit with $s_k = z_j$ (for $j = 0$ take $s_k = -g_k$)
11:    end if
12:    $r_{j+1} = r_j - \alpha_j H_k p_j \quad , \quad v_{j+1} = M_k^{-1} r_{j+1}$
13:    // Truncation test (note that $||r_{j+1}||_{M_k^{-1}} = r_{j+1}^T v_{j+1}$)
14:    if $(r_{j+1}^T v_{j+1} \leq \zeta_k g_0^T v_0)$ then
15:        exit with $s_k = z_{j+1}$
16:    end if
17:    $\beta_j = r_{j+1}^T (v_{j+1} - v_j) / r_j^T v_j \quad , \quad p_{j+1} = v_{j+1} + \beta_j p_j.$
18: end for
19: exit with $s_k = z_j$
```

Note that lines 14–16 in Algorithm 1 truncate the iterates as soon as the required

Algorithm 2 Line Search Truncated Newton (outer iterations of TN)

```
1: Initialization $w_0$, $M_0 = Id$, $\epsilon_g, \epsilon_f, \epsilon_x$
2: for $k = 0$ to $max\_outer$ do
3:    $g_k = \nabla f(w_k)$
4:    if $||g_k|| < \epsilon_g$ then
5:       exit with solution $w_k$
6:    end if
7:    Compute $s_k$ by calling Algorithm 1.
8:    Perform a line search to scale the step $s_k$ by $\alpha_k$.
9:    $w_{k+1} = w_k + \alpha_k s_k$
10:   Update $M_{k+1}$ by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula
11:   if $|f_{k+1} - f_k| < \epsilon_f$ or $||w_{k+1} - w_k|| < \epsilon_x$ then
12:      exit with solution $w_k$
13:   end if
14: end for
```

accuracy is reached, whereas lines 9–11 truncate the iterates as soon a negative curvature direction is detected. Here the negative curvature test, $p_j^T H_k p_j < 0$, is replaced with the equivalent descent direction test [18]. Thus, the solution z_j obtained by Algorithm 1 does not contain a negative curvature direction. One of the advantages of trust region over line search is that negative curvature directions can be properly exploited. The trust region method behaves numerically better for non-convex problems. We discuss this issue in the next section.

2.2. Trust region truncated Newton. In trust region truncated Newton (TRTN) methods the current iterate w_k is updated with a step s_k that is found by minimizing (2.1) within a given region $\mathcal{B}_k$ defined as

$$\mathcal{B}_k := \{w_k + s \in \mathbb{R}^n \mid ||s||_k \leq \Delta_k\} \tag{2.3}$$

where the norm $||s||_k$ used there is with respect the preconditionning matrix M_k: $||s_k||_k = ||s_k||_{M_k} = (s_k^T M_k s_k)^{1/2}$.

The trust region radius Δ_k is critical for the effectiveness of each iteration w_k. If the region is too small, the algorithm may miss an opportunity to take a substantial step that will move it much closer to the minimizer of the objective function f. If the region is too large, the minimizer of the model may be far from the minimizer of the objective function in the region, so one may to reduce the size of the region and try again. Thus, the trust region radius has to be automatically adapted according to previous iterations. If the quadratic model has produced, during the last iterations, reliable steps that accurately predict the behaviour of the objective function f, the size of the trust region radius can be increased to allow for longer steps. On the other hand, a step in which the quadratic model does not adequately predict the behaviour of the objective function f is an indication that the trust region radius Δ_k is too large. In such a case, we reduce the value of Δ_k and try again.

For the inner iterations (i.e. solving (2.1) constrained to (2.3)), the search direction s_k is computed by means of the Steihaug-Toint algorithm which is a truncated PCG algorithm, see Algorithm 3 (PCG-TRTN). It's basically the same algorithm as Algorithm 1 (PCG-LSTN) except the fact that the quadratic function (2.1) has to be minimized now within a trust region $\mathcal{B}_k$. In a similar way to PCG-LSTN, in PCG-TRTN we use a scaled two-step limited memory BFGS [13] for preconditioning the CG method and we truncate the inner iterations when the following criterion is satisfied:

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

where r_j is the PCG residual at inner iteration j, M_k is the preconditioning matrix and

$$\zeta_k = max\left(0.5/(k+1), ||r_0||_{M_k^{-1}}\right) \tag{2.4}$$

which are both provided at outer iteration k, and where

$$||r_j||_{M_k^{-1}} = \sqrt{r_j^T M_k^{-1} r_j}.$$

The matrix M_k^{-1} may be computed easily if the preconditionning M_k is updated using the BFGS method [14]. In this case, however, M_k^{-1} does not need to be computed since $M_k^{-1} r_j = v_j$, and thus $r_j^T M_k^{-1} r_j = r_j^T v_j$. The use of a preconditionning strategy and an adequate truncation criterion may have a large impact on the overall numerical efficiency of the method.

The fact that the PCG-TRTN is solved within a trust region $\mathcal{B}_k$ introduces two additional aspects with respect to PCG-LSTN. For the PCG-LSTN algorithm, the inner iterations are truncated as soon as a negative curvature is detected. On the other hand, in the PCG-TRTN algorithm, if a negative curvature is detected for direction p_j, the method tries to take the longest possible step within this direction; that is until the trust region boundary is crossed (see lines 9-14 of the code). Second, if the PCG iterate leaves the trust region, it is backtracked to the boundary (see lines 16-18 of the code). This additional truncation criterion is justified by the fact that no feasible iterate could be found once we are outside the trust region since all PCG iterates z_j monotonically increase in the M-norm starting from zero and as long as the Hessian H_k is symmetric positive definite.

In FIGURE 2.1 a geometrical interpretation of the computation of the step s_k for PCG-LSTN (top) and PCG-TRTN (bottom) is shown. The plots show the contours associated to the level lines of q_k, the quadratic approximation at w_k. The plots at the left (resp. right) are associated to a convex (resp. non-convex) approximation q_k. The PCG-LSTN algorithm generates a series of iterations z_j until the (approximate) minimizer of q_k is found. For the convex approximation q_k shown at the top-left corner, the algorithm starts with $z_0 = 0$ and follows direction v_0 until it reaches z_1, then uses direction v_1 to reach z_2, the approximate minimizer of q_k. This minimizer is associated to s_k, the search direction along which the objective function f has to be minimized. For the non-convex approximation at the top-right, note that the PCG-LSTN algorithm will truncate as soon as direction v_1 is generated, since it corresponds to a negative curvature direction. The PCG-LSTN algorithm returns in this case a direction s_k which corresponds to the direction given by v_0. Then line-search will be performed along $s_k = z_j$. Thus, the PCG-LSTN algorithm generally does not proceed along negative curvature directions. The only case in which s_k proceeds along a negative curvature direction is the case in which v_0, the initial search direction for PCG-LSTN, is associated to a negative curvature. In such a case $s_k = v_0$ and thus line search will be performed along a negative curvature direction, see Algorithm 1.

On the other hand the PCG-TRTN method, see FIGURE 2.1 (bottom), optimizes q_k within a trust region radius Δ_k. For the convex approximation q_k, the PCG-TRTN method finds a direction s_k which may differ from the one obtained by the

Algorithm 3 Steihaug-Toint Preconditioned Conjugate Gradient (Inner iterations of TRTN)

```
1: Initialize $z_0 = 0$, $r_0 = -g_k$, $v_0 = M_k^{-1} r_0$, $p_0 = v_0$, $\epsilon = 10^{-10}$, $\zeta_k$ (see 2.4)
2: for $j = 0$ to $max\_inner$ do
3:    // Singularity test
4:    if $\left(|r_j^T v_j| < \epsilon \text{ or } |p_j^T H_k p_j| < \epsilon\right)$ then
5:       exit with $s_k = z_j$ (for $j = 0$ take $s_k = -g_k$)
6:    end if
7:    $\alpha_j = r_j^T v_j / p_j^T H_k p_j$ , $z_{j+1} = z_j + \alpha_j p_j$
8:    // Descent direction replaces negative curvature test
9:    if $(g_k^T z_{j+1} \geq g_k^T z_j - \epsilon)$ then
10:      exit with $s_k = z_j$ if $p_j^T H_k p_j > 0$
11:            otherwise $s_k = z_j + \sigma_j p_j$
12:               where $\sigma_j$ is the positive root of $||z_j + \sigma p_j||_k = \Delta_k$.
13:            (for $j = 0$ take $s_k = -g_k$)
14:   end if
15:   // Trust region boundary test
16:   if $(||z_{j+1}||_k > \Delta_k)$ then
17:      exit with $s_k = \frac{\Delta_k}{||z_{j+1}||_k} z_{j+1}$
18:   end if
19:   $r_{j+1} = r_j - \alpha_j H_k p_j$ , $v_{j+1} = M_k^{-1} r_{j+1}$
20:   // Truncation test
21:   if $(r_{j+1}^T v_{j+1} \leq \zeta_k r_0^T v_0)$ then
22:      exit with $s_k = z_{j+1}$
23:   end if
24:   $\beta_j = r_{j+1}^T (v_{j+1} - v_j) / r_j^T v_j$ , $p_{j+1} = v_{j+1} + \beta_j p_j$.
25: end for
26: exit with $s_k = z_j$
```

PCG-LSTN. For the non-convex approximation, the method follows direction v_1 (a negative curvature direction) until the boundary of the trust region is reached. As a consequence, trust region methods will be able to deal more effectively with non-convex problems.

The outer iterations of TRTN are shown in Algorithm 4. For each iterate w_k, a fidelity measure is introduced by the ratio ρ_k to decide whether to accept the trial step s_k or not and accordingly how to change the trust region Δ_k:

$$\rho_k = \frac{\text{Ared}_k}{\text{Pred}_k}$$

where Ared_k is the actual reduction in the objective function:

$$\text{Ared}_k = f(w_k) - f(w_k + s_k)$$

and Pred_k is the predicted reduction by the quadratic model q_k:

$$\text{Pred}_k = q_k(0) - q_k(s_k) = -(g_k^T s_k + \frac{1}{2} s_k^T H_k s_k).$$

Since the step s_k is obtained by minimizing the model $q_k(s)$ over a region that includes $s = 0$, the predicted reduction will be always nonnegative. The value of Ared_k should

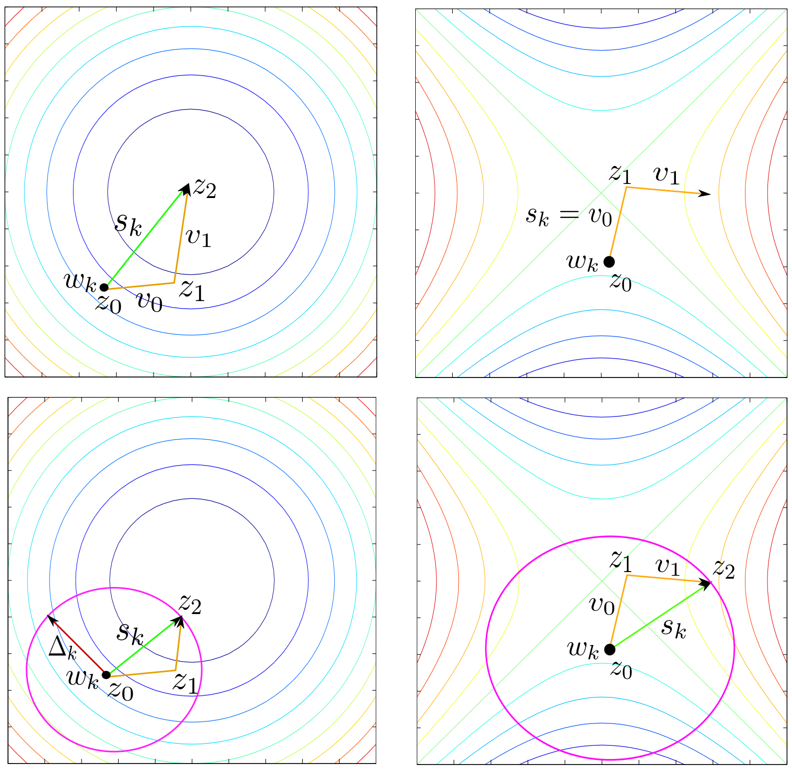

FIG. 2.1. *Geometrical interpretation of the computation of the step s_k for PCG-LSTN (top) and PCG-TRTN (bottom) for convex (left) and non-convex (right) quadratic approximations. The contours are associated to the level lines of the quadratic form q_k. The fact that trust region methods optimize the quadratic function within a trust region $\mathcal{B}_k$ makes the method behave better for such types of problems. See text for a discussion.*

be positive for each iteration w_k in order to correspond to a decrease of the objective function f. However, Ared_k may be negative, $f(w_{k+1}) > f(w_k)$, if Δ_k is too large.

Roughly speaking, the trial step s_k is accepted when the quadratic model q_k adequately predicts the behavior of the objective function (1.1) at w_k, that is when ρ_k is not too small or negative; otherwise s_k is rejected and recomputed in a smaller region. More precisely, in this work we will update the solution w_{k+1} and adjust the trust region radius Δ_{k+1} as follows: given constants $0 \le \eta_0 \le \eta_1 < \eta_2 < 1$ and $0 < \lambda_0 \le \lambda_1 < 1 < \lambda_2$, we have for the next outer iterate

$$w_{k+1} = \begin{cases} w_k & \text{if } \rho_k \le \eta_0 \\ w_k + s_k & \text{otherwise} \end{cases} \tag{2.5}$$

and for the next trust region radius

$$\Delta_{k+1} = \begin{cases} \min(\lambda_0 \Delta_k, \lambda_1 ||s_k||_k) & \text{if } \rho_k < \eta_1 \\ \Delta_k & \text{if } \eta_1 \le \rho_k \le \eta_2 \\ \max(\lambda_2 ||s_k||_k, \Delta_k) & \text{if } \rho_k > \eta_2. \end{cases} \tag{2.6}$$

A value of ρ_k close to 1 means that there is a good agreement between the model q_k and the objective function f over this step, so it is safe to try to expand the trust region for the next iteration. If ρ_k is positive but smaller than 1, we do not alter the trust region, but if it is close to zero or negative, we reduce the trust region by reducing the trust region radius Δ_k at the next iteration.

One might take $\eta_0 = 0$ so that all steps that provide a descent in the objective function are considered regardless of how small the reduction is with respect to the model reduction. However, stronger theoretical convergence results are proven for the case $\eta_0 > 0$ [6]. In our experiments we have used the following values:

$$\eta_0 = 10^{-4},\ \eta_1 = 0.05,\ \eta_2 = 0.9,\ \lambda_0 = 0.0625,\ \lambda_1 = 0.25 \text{ and } \lambda_2 = 2.5.$$

For the case $\eta_1 \le \rho_k \le \eta_2$, we have noted also that better performance is reached if we enlarge the radius when, for the inner iterations, the boundary is reached in very few inner iterations (one or two) and reduced in case there is no truncation (that is, when the loop of the inner iterations is exited without satisfying any of the truncation tests inside the loop).

As commented before, we use the same preconditioner as in LSTN, that is a scaled two-step limited memory BFGS. The BFGS method requires that $(g_{k+1}-g_k)^T(s_{k+1}-s_k)$ is positive between successive iterations w_k. For LSTN this restriction is not a problem since the line-search algorithm (i.e. computation of α_k) generally imposes the Wolf conditions, which ensure that the previous condition is satisfied for each iteration [14]. However, for the TRTN method we can not guarantee that the update $w_{k+1} = w_k + s_k$ satisfies the previous condition. As suggested in [15], we will not use the BFGS update when $(g_{k+1} - g_k)^T(s_{k+1} - s_k)$ is negative; which means for our algorithm that the preconditioner is not updated in this case. The BFGS algorithm is updated for the iteration $m > k$ such that $(g_m - g_k)^T(s_m - s_k)$ is positive.

Algorithm 4 Trust Region Truncated Newton (outer iterations of TRTN)

```
1: Initialize $w_0 = 0$, $M_0 = Id$, $\epsilon_g, \epsilon_f, \epsilon_x$
2: for $k = 0$ to max_outer do
3:    $g_k = \nabla f(x_k)$
4:    if $||g_k|| < \epsilon_g$ then
5:       exit with solution $w_k$
6:    end if
7:    Compute $s_k$ by calling Algorithm 3.
8:    $\rho_k = (f(w_k) - f(w_k + s_k))/(g_k^T s_k + \frac{1}{2} s_k^T H_k s_k)$
9:    Update $w_{k+1}$ by (2.5) and $\Delta_{k+1}$ by (2.6).
10:   Update $M_{k+1}$ by the BFGS formula.
11:   if $f_k - f_{k+1} < \epsilon_f$ or $||w_{k+1} - w_k|| < \epsilon_x$ then
12:      exit with solution $w_{k+1}$
13:   end if
14: end for
```

2.3. Multiresolution methods. The TRTN algorithm may be embedded in a multiresolution method. Using multiple levels of resolution allows us to deal, in the case of optical flow computation, with large displacements and enhances the chance of the convergence to the global minimum or a good local minimum since worse local minima may disappear at sufficient coarse resolutions.

Let us denote by Ω_i the image domain at level i, where $i = 0, \dots, r$, where $i = 0$ corresponds to the finest level of resolution and $i = r$ to the coarsest one. Grid spacing at the coarser grid Ω_{i+1} is usually twice the spacing at the grid Ω_i. Multiresolution methods use the latter levels of resolution to obtain an estimate of the solution to the problem. For a given grid Ω_i, the method solves the problem for that level and then extends the solution to grid Ω_{i-1}, where it is refined. This process is repeated until the finest grid Ω_0 is reached, where the final estimate is reached. The method is initiated by solving the problem at the coarsest grid Ω_r. By using this method one expects to obtain a good initial guess for each grid Ω_i.

3. Optical flow models. In the energy functional in (1.1), two data terms D and two regularization terms R have been tested. To describe them here in short, let us consider a sequence of gray level images $I(t,x,y)$, $t \in [0,T]$, $(x,y) \in Q$, where Q denotes the image domain, which we assume to be a rectangle in $\mathbb{R}^2$. The image sequence is actually sampled at the times $t_j = j\Delta t$, $j = 0,\ldots,K$.

The first data term corresponds to the classical Horn-Schunk linear version of the brightness conservation assumption:

$$I(t+1, x(t)+u(t,x,y), y(t)+v(t,x,y)) = I(t,x(t),y(t)), \tag{3.1}$$

where $(x(t),y(t))$ is the apparent trajectory of a given point at time t and the vector field $w(t,x,y) := (u(t,x,y), v(t,x,y))$ is called the optic flow. When only small displacments occur between two succesive frames, we can take a first order Taylor development of the first term and differentiate with respect to t to obtain the optical flow constraint

$$I^t + uI^x + vI^y = 0, \tag{3.2}$$

where I^t, I^x, I^y denote the partial derivatives of I with respect to t,x,y, respectively. Clearly, each of the single constraint: the nonlinear equation (3.1) and the linear equation (3.2) is not sufficient to uniquely compute the two components (u,v) of the optic flow. In the case of the second equation, this is called the aperture problem; only the component of the flow normal to the image gradient, i.e., to the level lines of the image could be computed. As it is usual, in order to recover a unique flow field a regularization constraint is added. For that, we assume that the optic flow varies smoothly in space, or better, that is piecewise smooth in Q. This can be achieved by including a smoothness term of the form

$$R(w) := \int_Q G(\nabla u, \nabla v)\, dxdy, \tag{3.3}$$

where $G : \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is a suitable function. The case $G = \|\nabla u\|^2 + \|\nabla v\|^2$ corresponds to the Horn-Schunk model [8], and the case $G = \sqrt{\|\nabla u\|^2 + \|\nabla v\|^2}$ or $G = \|\nabla u\| + \|\nabla v\|$ correspond to total variation regularization models.

Both linear data attachment and regularization terms can be combined into a single energy functional

$$\int_Q \Psi\left((I^t + uI^x + vI^y)^2\right) dxdy + \alpha \int_Q G(\nabla u, \nabla v)\, dxdy, \tag{3.4}$$

where $\alpha > 0$ is the regularization parameter weighting the relative importance of both terms and $\Psi : \mathbb{R} \to \mathbb{R}$ is an increasing smooth function used to enhance robustness with respect to outliers. Similarly to our previous work [9], we use here

$$\psi\left(x^2\right) = \begin{cases} x^2 & \text{if } |x| \le \gamma \\ \gamma^2 & \text{otherwise,} \end{cases}$$

where γ is a given threshold.

In case of large displacements the form (3.1) may be more appropriate for the data term. A corresponding energy functional can be obtained by combining the non linearized form of the brightness constancy assumption and a regularization term

$$\int_Q \Psi\left((I(t,x,y)-I(t+1,x+u,y+v))^2\right)dxdy+\alpha\int_Q G(\nabla u,\nabla v)\,dxdy. \tag{3.5}$$

The same two examples of function G as above are used for this case as well. Observe that the energy (3.5) is nonlinear and non convex.

4. Implementation Issues. The numerical algorithms LSTN, TRTN and their multiresolution versions have been implemented in C. In this section we provide some details about the implementation of the optical flow models.

4.1. Functional and gradient calculation. The gradient of the objective function in (1.1) 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 D^u, D^v, R^u and R^v) correspond to the partial derivatives of D and R with respect to u and v).

4.1.1. Linear data term. For the linear data term we use

$$D(u,v)=\frac{1}{2}\sum_{i,j}\left(\Psi[I^x u_{i,j}+I^y v_{i,j}+I^t]\right)^2.$$

where i (respectively j) corresponds to the discrete column (respectively row) of the image. The coordinate origin is located in the top-left corner of the image. The gradient D for $|x|\le\gamma$ is therefore given by

$$\begin{pmatrix} D^u_{i,j} \\ D^v_{i,j}\end{pmatrix}=\begin{pmatrix} I^x\,[I^x u_{i,j}+I^y v_{i,j}+I^t] \\ I^y\,[I^x u_{i,j}+I^y v_{i,j}+I^t]\end{pmatrix},$$

where $D^u_{i,j}$ and $D^v_{i,j}$ refer to the partial derivative of $D(u,v)$ with respect to variables $u_{i,j}$ and $v_{i,j}$, respectively. Note that for $|x|>\gamma$ the gradient D is $(D^u_{i,j}, D^v_{i,j})^T=(0,0)^T$.

4.1.2. Non-linear data term. The nonlinear data term based on the intensity constancy assumption is as follows

$$D(u,v)=\frac{1}{2}\sum_{i,j}\psi\left([I_2(i+u_{i,j},j+v_{i,j})-I_1(i,j)]^2\right).$$

Note that the previous term is not convex. The gradient for $|x|\le\gamma$ is given by

$$\begin{pmatrix} D^u_{i,j} \\ D^v_{i,j}\end{pmatrix}=\begin{pmatrix} I_2^x(i+u_{i,j},j+v_{i,j})\,[I_2(i+u_{i,j},j+v_{i,j})-I_1(i,j)] \\ I_2^y(i+u_{i,j},j+v_{i,j})\,[I_2(i+u_{i,j},j+v_{i,j})-I_1(i,j)]\end{pmatrix}.$$

The image gradient I_2^x and I_2^y is computed as explained in [9]. Note that the function and gradient evaluation requires the evaluation of the image and gradient at non-integer points. For that issue bilinear interpolation is used.

4.1.3. Quadratic regularization term. We have

$$R(u,v) = \frac{1}{2}\sum_{i,j} ||\nabla_{i,j}(u,v)||^2,$$

where $||\nabla_{i,j}(u,v)|| = \sqrt{(u^x_{i,j})^2 + (u^y_{i,j})^2 + (v^x_{i,j})^2 + (v^y_{i,j})^2}$. The partial derivatives of u, v are computed by forward finite differences with a discretization step h, that is, $u^x_{i,j} = h^{-1}(u_{i+1,j} - u_{i,j})$ and $u^y_{i,j} = h^{-1}(u_{i,j+1} - u_{i,j})$ (derivatives v^x and v^y are computed similarly). The gradient of this functional is obtained as

$$\begin{pmatrix} R^u_{i,j} \\ R^v_{i,j} \end{pmatrix} = \frac{1}{h^2}\begin{pmatrix} 4u_{i,j} - u_{i-1,j} - u_{i,j-1} - u_{i+1,j} - u_{i,j+1} \\ 4v_{i,j} - v_{i-1,j} - v_{i,j-1} - v_{i+1,j} - v_{i,j+1} \end{pmatrix}.$$

The previous gradient expression corresponds to the discretization of the Laplacian operator, the operator that is obtained from the Euler-Lagrange equations of the quadratic term.

4.1.4. Total variation term. Here we suppose that

$$R(u,v) = \sum_{i,j} ||\nabla_{i,j}(u,v)||.$$

To overcome the problem of non-differentiability of the total variation, a commony used technique consists in approximating R by a differentiable function:

$$R(u,v) \approx \sum_{i,j} \psi_{i,j},$$

where $\psi_{i,j} = \sqrt{(u^x_{i,j})^2 + (u^y_{i,j})^2 + (v^x_{i,j})^2 + (v^y_{i,j})^2 + \mu}$ and μ is a small positive parameter. Using again forward finite differences, the gradient of the last approximation is given by

$$\begin{pmatrix} R^u_{i,j} \\ R^v_{i,j} \end{pmatrix} = \frac{1}{h}\begin{pmatrix} \frac{u^x_{i-1,j}}{\psi_{i-1,j}} + \frac{u^y_{i,j-1}}{\psi_{i,j-1}} - \frac{u^x_{i,j}+u^y_{i,j}}{\psi_{i,j}} \\ \frac{v^x_{i-1,j}}{\psi_{i-1,j}} + \frac{v^y_{i,j-1}}{\psi_{i,j-1}} - \frac{v^x_{i,j}+v^y_{i,j}}{\psi_{i,j}} \end{pmatrix}.$$

4.2. Hessian calculation. For computing the Newton direction in TRTN methods, the linear conjugate gradient is a Hessian-free procedure (see Algorithms 3). The user only needs to supply a routine for computing the product of the Hessian with Newton direction p (see e.g. lines 7 and 19 of Algorithm 3). This matrix-vector product is computed via forward finite differences:

$$H(w)\,p = \frac{g(w+\epsilon p) - g(w)}{\epsilon}$$

where ϵ is chosen to be the square root of the machine precision divided by the norm of w. Thus, Newton methods only require the user to specify a procedure to compute the function and gradient values.

4.3. Iterative procedure for norm computation in the PCG method. Algorithm 3 requires computation of $||z_j||_k^2 = z_j^T M_k z_j$ during the inner iterations, which may lead to a high computational cost if the product $M_k z_j$ has to be explicitly computed. However the preconditionning matrix, M_k, remains unchanged along the inner iterations computations. Taking advantage of the iterative nature of the algorithm an iterative procedure to compute $||z_{j+1}||_k^2$ can be envisaged. Indeed, observe that $||z_{j+1}||_k^2$ can be developed as

$$\begin{aligned} z_{j+1}^T M_k z_{j+1} &= (z_j + \alpha_j p_j)^T M_k (z_j + \alpha_j p_j) \\ &= z_j^T M_k z_j + 2\alpha_j z_j^T M_k p_j + \alpha_j^2 p_j^T M_k p_j \end{aligned} \tag{4.1}$$

where $z_j^T M_k z_j = ||z_j||_k^2$. The last term of the previous equation, $p_j^T M_k p_j$, can be further developed as

$$\begin{aligned} p_j^T M_k p_j &= (v_j + \beta_{j-1} p_{j-1})^T M_k (v_j + \beta_{j-1} p_{j-1}) \\ &= v_j^T M_k v_j + 2\beta_{j-1} v_j^T M_k p_{j-1} + \beta_{j-1}^2 p_{j-1}^T M_k p_{j-1} \\ &= v_j^T r_j + \beta_{j-1}^2 p_{j-1}^T M_k p_{j-1} \end{aligned}$$

where $v_j^T M_k p_{j-1} = 0$ due to the fact that Algorithm 3 is a conjugate gradient. The second term of (4.1), $z_j^T M_k p_j$, is developed as

$$\begin{aligned} z_j^T M_k p_j &= z_j^T M_k (v_j + \beta_{j-1} p_{j-1}) \\ &= z_j^T M_k v_j + \beta_{j-1} z_j^T M_k p_{j-1} \\ &= z_j^T r_j + \beta_{j-1} (z_{j-1}^T M_k p_{j-1} + \alpha_{j-1} p_{j-1}^T M_k p_{j-1}) \end{aligned}$$

where we have used the recurrence $z_j = z_{j-1} + \alpha_{j-1} p_{j-1}$. The iterative process is initiated with $j = 0$ and thus $||z_0||_k^2 = z_0^T M_k z_0 = 0$, $z_0^T M_k p_0 = 0$, and $p_0^T M_k p_0 = v_0^T r_0$. Thus, to obtain $||z_{j+1}||_k^2$ only information of the previous two iterations is needed, namely the terms $z_j^T M_k z_j$, $v_j^T r_j$, $z_j^T r_j$ and $z_{j-1}^T M_k p_{j-1}$, $p_{j-1}^T M_k p_{j-1}$. These terms can be easily computed with the Algorithm 3. The algorithm is initiated at $j = 0$ with $z_0^T M_k z_0 = 0$, $z_0^T M_k p_0 = 0$ and $p_0^T M_k p_0 = g_k^T v_k$.

5. Experimental Results. This section is devoted to the assessment of the proposed trust region algorithms, namely TRTN (unilevel) and MR/TRTN (multiresolution). Our purpose is to compare their numerical performance with the corresponding line search versions previously studied in [9]: LSTN (unilevel) and MR/LSTN (multiresolution), respectively. For that issue we use three sequences of synthetic images that consist of scenes of various complexity; namely, the translating tree, the diverging tree and the Yosemite sequences. The reference frame and the corresponding ground truth of the synthetic sequences are show in Figures 5.1 and 5.2. The image size of the tree sequences is 150×150 while the Yosemite sequence is of size 316×252.

Moreover, we use four optical flow models combining linear and nonlinear data terms with quadratic and TV regularization as described in Section 3. In Tables 5.1-5.3, Model 1 refers to the linear data term plus the quadratic regularization; Model 2 refers to the nonlinear data term plus the quadratic regularization; Model 3 refers to the linear data term plus the TV regularization; and finally Model 4 refers to the nonlinear data term plus the TV regularization.

Since we are interested in comparing the computational work of the TRTN, MR/TRTN and LSTN, MR/LSTN algorithms, we use the following measures to assess the numerical performance: i) the CPU time needed by each numerical algorithm

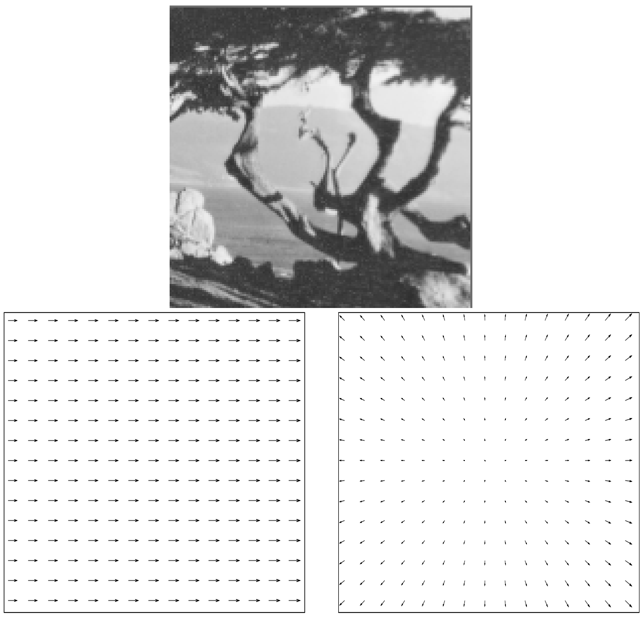

FIG. 5.1. *On the top one frame of the original sequence is shown. On the bottom the ground truth for the corresponding translating (left) and diverging (right) is shown. Motion vectors have been scaled by a factor of* 2.5 *for better visibility.*

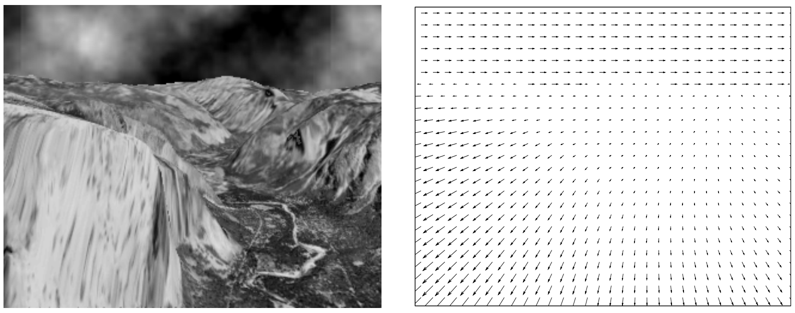

FIG. 5.2. *On the left, one frame of the Yosemite sequence is shown. On the right, the corresponding ground truth is depicted where motion vectors have been scaled by a factor of* 2.5 *for better visibility.*

to reach a similar accuracy using four optical flow models and ii) the total number of functional and gradient evaluations Nfg that were performed by each algorithm to reach the estimated optical flow. We measure the overall number of function N_f evaluations as

$$N_f = \sum_{i=0}^{r} \frac{N_{f,i}}{F_i}$$

where $N_{f,i}$ is the number of function evaluations performed by the optimization algorithm at resolution level i. In the TRTN algorithm the function is only evaluated to compute Ared_k, whereas in the LSTN the function may be evaluated several times in the line search procedure. F_i is the mesh resolution ratio of a given level i with respect to the finest level 0. In this work $F_i = 2^{2i}$.

The number of gradient evaluations N_g is defined in a similar manner. The gradient is evaluated for both TRTN and LSTN during the inner iterations of the TN algorithm (i.e. the computation of the matrix-vector product of the Hessian with the Newton direction). Additionally, the LSTN evaluates the gradient in the line search procedure (as a way to increase efficiency to get to an acceptable minimum along $w_k + \alpha_k p_k$). Thus, the number of function evaluations is expected to be always lower than the gradient evaluations. Note also that the number of gradient evaluations is more crucial to the overall computational work than that of the objective function. Here one gradient evaluation is approximately equivalent to two function evaluations when using quadratic regularization, while it takes almost three times in the case of TV regularization. Thus, the total number of function and gradient evaluations Nfg is computed as follows,

$$\text{Nfg} = \frac{N_f}{K} + N_g$$

where $K = 2$ (resp. $K = 3$) if a quadratic (resp. total variation) regularization is used.

To compute the accuracy of the optical flow estimations of the numerical algorithms, we measure the average angular error (AAE) of the estimated flow w^e with respect to the ground truth w^c. For a given pixel (i,j), the angular error (AE) between the ground truth motion vector, $w^c_{i,j}$, and the estimated flow, $w^e_{i,j}$, is computed as:

$$\text{AE}(w^c_{i,j}, w^e_{i,j}) = cos^{-1}\left(\frac{u^c_{i,j}u^e_{i,j} + v^c_{i,j}v^e_{i,j} + 1}{\sqrt{(u^c_{i,j})^2 + (v^c_{i,j})^2 + 1}\sqrt{(u^e_{i,j})^2 + (v^e_{i,j})^2 + 1}}\right).$$

The average angular error is the mean of the angular error over all pixels N_{np} of the image

$$\text{AAE}(w_c, w_e) = \frac{1}{N_{np}}\sum_{i,j}\text{AE}(w^c_{i,j}, w^e_{i,j}).$$

In Tables 5.1-5.3, we show the results obtained for the three sequences and the four optical flow models using line search algorithms (LSTN and MR/LSTN) and trust region algorithms (TRTN and MR/TRTN). All algorithms were stopped when similar accuracy was reached, which is confirmed by the similar values of AAE and the final values of the objective function. For the translating tree sequence (Table 5.1), the trust region method is better than the line search method for both unilevel and multiresolution and for all four models. The performance of trust region appears to be higher for Model 4 which has a nonlinear data term and a TV regularization. For the diverging tree sequence (Table 5.2), trust region is also better except for Model 1 and Model 3. Note here that these two models contain a quadratic regularization term. Finally, for the Yosemite sequence (Table 5.3), the trust region is still better except for the case of multiresolution with the linear data term; that is Model 1 and Model 2. Overall, in 24 cases, the trust region method performed better 19 times than the line search method. In particular, for the most complicated model, which is Model 4, the performance of trust region is always much higher for the three sequences and for both unilevel and multiresolution.

Table 5.1

Unilevel and multiresolution line search truncated Newton (LSTN and MR/LSTN) versus trust region (TRTN and MR/TRTN) for two frames of the translating tree sequence using four optical flow models

		LSTN	TRTN	MR/LSTN	MR/TRTN
	Time (s)	2.84	2.37	0.76	0.59
Model 1	Nfg	686	583	201	165
	Funct. value	$1.46e^5$	$1.45e^5$	$7.54e^4$	$7.55e^4$
	AAE	1.05	1.00	0.90	0.83
	Time (s)	3.68	3.26	1.11	1.00
Model 2	Nfg	737	624	217	205
	Funct. value	$7.20e^{03}$	$7.21e^{03}$	$7.20e^3$	$7.21e^3$
	AAE	0.24	0.24	0.23	0.23
	Time (s)	4.65	3.45	1.52	1.28
Model 3	Nfg	869	676	308	256
	Funct. value	$8.70e^5$	$8.61e^5$	$3.74e^5$	$3.74e^5$
	AAE	0.97	0.97	0.82	0.82
	Time (s)	6.61	4.55	2.06	1.30
Model 4	Nfg	1044	693	344	222
	Funct. value	$3.03e^5$	$3.03e^5$	$3.03e^5$	$3.03e^5$
	AAE	0.20	0.20	0.20	0.21
Total	Time (s)	17.78	13.63	5.45	4.17
	Nfg	3336	2576	1070	848

Table 5.2

Unilevel and multiresolution line search truncated Newton (LSTN and MR/LSTN) versus trust region (TRTN and MR/TRTN) for two frames of the diverging tree sequence using four optical flow models

		LSTN	TRTN	MR/LSTN	MR/TRTN
	Time (s)	4.46	4.60	0.50	0.78
Model 1	Nfg	1025	1168	125	217
	Funct. value	$2.00e^4$	$2.00e^4$	$2.00e^4$	$2.01e^4$
	AAE	1.93	1.94	1.89	1.91
	Time (s)	3.91	3.17	1.09	1.03
Model 2	Nfg	703	611	215	211
	Funct. value	$3.04e^4$	$3.04e^4$	$3.04e^4$	$3.03e^4$
	AAE	2.03	2.05	2.07	2.08
	Time (s)	5.28	4.75	0.61	1.46
Model 3	Nfg	996	922	120	286
	Funct. value	$2.78e^5$	$2.78e^5$	$2.78e^5$	$2.78e^5$
	AAE	2.39	2.37	2.36	2.36
	Time (s)	6.78	4.28	2.97	1.76
Model 4	Nfg	1064	669	508	285
	Funct. value	$2.12e^5$	$2.12e^5$	$2.12e^5$	$2.12e^5$
	AAE	2.22	2.24	2.11	2.08
Total	Time (s)	20.43	16.80	5.17	5.03
	Nfg	3788	3370	968	999

TABLE 5.3
Unilevel and multiresolution line search truncated Newton (LSTN and MR/LSTN) versus trust region (TRTN and MR/TRTN) for two frames of the Yosemite sequence using four optical flow models

		LSTN	TRTN	MR/LSTN	MR/TRTN
	Time (s)	4.72	3.06	1.09	1.30
Model 1	Nfg	350	194	65	92
	Funct. value	$1.82e^5$	$1.82e^5$	$1.74e^4$	$1.74e^4$
	AAE	6.86	6.86	6.76	6.78
	Time (s)	11.39	9.58	2.31	2.38
Model 2	Nfg	510	423	102	114
	Funct. value	$1.70e^5$	$1.70e^5$	$1.50e^5$	$1.50e^5$
	AAE	6.26	6.25	6.23	6.24
	Time (s)	19.59	10.43	4.05	2.32
Model 3	Nfg	963	531	202	123
	Funct. value	$3.15e^5$	$3.15e^5$	$2.96e^5$	$2.99e^5$
	AAE	6.32	6.31	6.12	6.11
	Time (s)	41.72	19.55	6.37	4.51
Model 4	Nfg	1636	767	261	188
	Funct. value	$3.42e^5$	$3.20e^5$	$2.90e^5$	$2.91e^5$
	AAE	5.76	5.79	5.43	5.44
Total	Time (s)	77.42	42.62	13.82	10.51
	Nfg	3459	1915	630	517

6. Conclusion.

In this paper, we presented unilevel and multiresolution trust region algorithms for optical flow computation and investigated their performance against corresponding line search versions. We considered four optical flow models that combine linearity and non-linearity in the data term with quadratic and TV regularization. Three image sequences with different types of motion were used in the test. The trust region method performed better in 19 cases over 24 than the line search method. In particular, for Model 4 that contains non-linear data term and TV regularization, trust region had always a higher performance. This suggests that the method would be a suitable optimization algorithm for non-linear and non-convex optical flow models. Our future research will investigate the use of trust region as an optimizer for multigrid optical flow computation.

Acknowledgments. The first author is grateful to financial support from the program of research support in King Khalid University under Contract Number "KKU_S064_33". The second author acknowledges partial support by MICINN project, reference MTM2009-08171, and by GRC reference 2009 SGR 773.

REFERENCES

[1] L. Álvarez, J. Weickert, and J. Sánchez, *Reliable estimation of dense optical flow fields with large displacements*, International Journal of Computer Vision, 39 (2000), pp. 41–56.
[2] P. Anandan, *A computational framework and an algorithm for the measurement of visual motion*, International Journal of Computer Vision, 2 (1989), pp. 283–310.
[3] J. M. Bardsley, *A nonnegatively constrained trust region algorithm for the restoration of images with an unknown blur*, Electronic Transactions in Numerical Analysis, 20 (2005), pp. 139–153.

[4] J. Barron, D. Fleet, and S. Beauchemin, *Performance of optical flow techniques*, International Journal of Computer Vision, 12 (1994), pp. 43–77.

[5] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnorr, *Variational optical flow computation in real time*, IEEE Transactions on Image Processing, 14 (2005), pp. 608–615.

[6] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, SIAM, Philadelphia, 2000.

[7] W. Enkelmann, *Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences*, Computer Vision, Graphics and Image Processing, 43 (1988), pp. 150–177.

[8] B. Horn and B. Schunk, *Determining optical flow*, Artificial Intelligence, 20 (1981).

[9] E. Kalmoun, L. Garrido, and V. Caselles, *Multilevel optimization as computational methods for dense optical flow*, SIAM Journal of Imaging Sciences, 4 (2011), pp. 695–722.

[10] 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.

[11] C.-J. Lin, R. Weng, and S. Keerthi, *Trust region Newton method for logistic regression*, Jounal of Machine Learning Research, 9 (2008), pp. 627–650.

[12] T. Liu and H. Chen, *Real-time tracking using trust-region methods*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 397–402.

[13] S. G. Nash, *Preconditioning of truncated-newton methods*, SIAM Journal on Scientific and Statistical Computing, 6 (1985), pp. 599–616.

[14] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

[15] J. Nocedal and Y. Yuan, *Combining trust region and line search techniques*, in Advances in Nonlinear Programming, Y. Yuan, ed., Kluwer, 1998, pp. 153–175.

[16] T. Phong, R. Horaud, A. Yassine, and P. Tao, *Object pose from 2-d to 3-d point and line correspondences*, International Journal of Computer Vision, 15 (1995), pp. 496–508.

[17] D. Rosen, M. Kaess, and J. Leonard, *An incremental trust-region method for robust online sparse least-squares estimation*, in IEEE International Conference on Robotics and Automation, May 2012, pp. 1262–1269.

[18] 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.