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:

$$\min_w f(w),$$

where $f(w) = D(w) + \alpha R(w)$, $w = (u, v)$ is the optical flow field and $\alpha$ 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
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 $\alpha_k$ 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 $q_k$,

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

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

$$H_k s = -g_k \quad (2.2)$$
The current solution $w_k$ is then updated by scaling the step $s_k$ by a factor $\alpha_k$ 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 $B_k$ 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 $s \in B_k$. 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 truncated 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 $(|r_j^T v_j| < \epsilon$ or $|p_j^T H_k p_j| < \epsilon$) 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 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: $r_{j+1} = r_j - \alpha_j H_k p_j$, $v_{j+1} = M_k^{-1} r_{j+1}$
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: $\beta_j = r_{j+1}^T (v_{j+1} - v_j)/r_j^T v_j$, $p_{j+1} = v_{j+1} + \beta_j p_j$
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 $w_0$, $M_0 = I_d$, $\epsilon_g, \epsilon_f, \epsilon_x$
2: for $k = 0$ to $\text{max.outer}$ do
3: \hspace{1em} $g_k = \nabla f(w_k)$
4: \hspace{1em} if $\|g_k\| < \epsilon_g$ then
5: \hspace{2em} exit with solution $w_k$
6: \hspace{1em} end if
7: \hspace{1em} Compute $s_k$ by calling Algorithm 1.
8: \hspace{1em} Perform a line search to scale the step $s_k$ by $\alpha_k$.
9: \hspace{1em} $w_{k+1} = w_k + \alpha_k s_k$
10: \hspace{1em} Update $M_{k+1}$ by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula
11: \hspace{1em} if $|f_{k+1} - f_k| < \epsilon_f$ or $\|w_{k+1} - w_k\| < \epsilon_x$ then
12: \hspace{2em} exit with solution $w_k$
13: \hspace{1em} 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, $p^T H_k p_j < 0$, 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 $B_k$ defined as

$$B_k := \{w_k + s \in \mathbb{R}^n \mid \|s\|_k \leq \Delta_k\}$$

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 $\Delta_k$ 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 $\Delta_k$ is too large. In such a case, we reduce the value of $\Delta_k$ 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 $B_k$. 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

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

(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 preconditioning $M_k$ 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 preconditioning 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 $B_k$ 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 $\Delta_k$. 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 $|(r_j^Tv_j) < \epsilon$ or $|p_j^TH_kp_j| < \epsilon$ then
5: exit with $s_k = z_j$ (for $j = 0$ take $s_k = -g_k$
6: end if
7: $\alpha_j = r_j^Tv_j/p_j^TH_kp_j$ , $z_{j+1} = z_j + \alpha_jp_j$
8: // Descent direction replaces negative curvature test
9: if $(g_k^Tz_{j+1} \geq g_k^Tz_j - \epsilon)$ then
10: exit with $s_k = z_j$ if $p_j^TH_kp_j > 0$
11: otherwise $s_k = z_j + \sigma_jp_j$
12: where $\sigma_j$ is the positive root of $||z_j + \sigma_jp_j|| = \Delta_k$
13: (for $j = 0$ take $s_k = -g_k$
14: end if
15: // Trust region boundary test
16: if $||z_{j+1}|| \Delta_k$ then
17: exit with $s_k = \Delta_k/||z_{j+1}||
18: end if
19: $r_{j+1} = r_j - \alpha_jH_kp_j$ , $v_{j+1} = M_k^{-1}r_{j+1}$
20: // Truncation test
21: if $(r_{j+1}^Tv_{j+1} \leq \zeta_kr_0^Tv_0)$ then
22: exit with $s_k = z_{j+1}$
23: end if
24: $\beta_j = r_{j+1}^T(v_{j+1} - v_j)/r_j^Tv_j$ , $p_{j+1} = v_{j+1} + \beta_jp_j$
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 $\rho_k$ to decide whether to accept the trial step $s_k$ or not and accordingly how to change the trust region $\Delta_k$:

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

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 $q_k$:

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

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
be positive for each iteration \(w_k\) in order to correspond to a decrease of the objective function \(f\). However, \(\text{Area}_k\) may be negative, \(f(w_{k+1}) > f(w_k)\), if \(\Delta_k\) 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 \(\rho_k\) 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 \(\Delta_{k+1}\) as follows: given constants \(0 \leq \eta_0 \leq \eta_1 < \eta_2 < 1\) and \(0 \leq \lambda_0 \leq \lambda_1 < 1 \leq \lambda_2\), we have for the next outer iterate

\[
\begin{align*}
 w_{k+1} & = \begin{cases} 
 w_k & \text{if } \rho_k \leq \eta_0 \\
 w_k + s_k & \text{otherwise}
 \end{cases} \\
 \Delta_{k+1} & = \begin{cases} 
 \min(\lambda_0 \Delta_k, \lambda_1 ||s_k||_k) & \text{if } \rho_k < \eta_1 \\
 \Delta_k & \text{if } \eta_1 \leq \rho_k \leq \eta_2 \\
 \max(\lambda_2 ||s_k||_k, \Delta_k) & \text{if } \rho_k > \eta_2.
 \end{cases}
\end{align*}
\]

A value of \(\rho_k\) 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. See text for a discussion.
region for the next iteration. If \( \rho_k \) 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 \( \Delta_k \) 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:

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

For the case \( \eta_1 \leq \rho_k \leq \eta_2 \), 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 \( \alpha_k \)) 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 \( w_0 = 0, M_0 = \text{Id}, \epsilon_g, \epsilon_f, \epsilon_x \)
2: for \( k = 0 \) to \( \max \text{outer} \) 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 \( \Omega_i \) the image domain at level \( i \), where \( i = 0, \ldots, r \), where \( i = 0 \) corresponds to the finest level of resolution and \( i = r \) to the coarsest one. Grid spacing
at the coarser grid $\Omega_{i+1}$ is usually twice the spacing at the grid $\Omega_i$. Multiresolution methods use the latter levels of resolution to obtain an estimate of the solution to the problem. For a given grid $\Omega_i$, the method solves the problem for that level and then extends the solution to grid $\Omega_{i-1}$, where it is refined. This process is repeated until the finest grid $\Omega_0$ is reached, where the final estimate is reached. The method is initiated by solving the problem at the coarsest grid $\Omega_r$. By using this method one expects to obtain a good initial guess for each grid $\Omega_i$.

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:

$$I(t+1, x(t) + u(t,x,y), y(t) + v(t,x,y)) = I(t, x(t), y(t)), \quad (3.1)$$

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 displacements occur between two successive 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

$$I^t + uI^x + vI^y = 0, \quad (3.2)$$

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

$$R(w) := \int_Q G(\nabla u, \nabla v) \, dx \, dy, \quad (3.3)$$

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

$$\int_Q \Psi ((I^t + uI^x + vI^y)^2) \, dx \, dy + \alpha \int_Q G(\nabla u, \nabla v) \, dx \, dy, \quad (3.4)$$

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

$$\psi(x^2) = \begin{cases} x^2 & \text{if } |x| \leq \gamma \\ \gamma^2 & \text{otherwise} \end{cases}$$
where $\gamma$ 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

$$\int_Q \psi \left( (I(t, x, y) - I(t + 1, x + u, y + v))^2 \right) \, dx \, dy + \alpha \int_Q G(\nabla u, \nabla v) \, dx \, dy. \quad (3.5)$$

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

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

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 $|x| \leq \gamma$ is therefore given by

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

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

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

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

$$\begin{pmatrix} D^u_{i,j} \\ D^v_{i,j} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} [I_x^2 (i + u_{i,j}, j + v_{i,j})] [I_x(i + u_{i,j}, j + v_{i,j}) - I_x(i,j)] \\ [I_y^2 (i + u_{i,j}, j + v_{i,j})] [I_x(i + u_{i,j}, j + v_{i,j}) - I_x(i,j)] \end{pmatrix}.$$

The image gradient $I_x^2$ and $I_y^2$ 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 commonly 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 \( \mu \) is a small positive parameter. Using again forward finite differences, the gradient of the last approximation is given by

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

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 \( \epsilon \) 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 $||z_j||^2_k = z_j^T M_k z_j$ 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 preconditioning matrix, $M_k$, remains unchanged along the inner iterations computations. Taking advantage of the iterative nature of the algorithm an iterative procedure to compute $||z_{j+1}||^2_k$ can be envisaged. Indeed, observe that $||z_{j+1}||^2_k$ can be developed as

$$z_{j+1}^T M_k z_{j+1} = (z_j + \alpha_j p_j)^T M_k (z_j + \alpha_j p_j)$$

$$= z_j^T M_k z_j + 2 \alpha_j z_j^T M_k p_j + \alpha_j^2 p_j^T M_k p_j$$

(4.1)

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

$$p_j^T M_k p_j = (v_j + \beta_{j-1} p_{j-1})^T M_k (v_j + \beta_{j-1} p_{j-1})$$

$$= v_j^T M_k v_j + 2 \beta_{j-1} v_j^T M_k p_{j-1} + \beta_{j-1}^2 p_{j-1}^T M_k p_{j-1}$$

$$= v_j^T r_j + \beta_{j-1}^2 p_{j-1}^T M_k p_{j-1}$$

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

$$z_j^T M_k p_j = z_j^T M_k (v_j + \beta_{j-1} p_{j-1})$$

$$= z_j^T M_k v_j + \beta_{j-1} z_j^T M_k p_{j-1}$$

$$= z_j^T r_j + \beta_{j-1} (z_{j-1}^T M_k p_{j-1} + \alpha_{j-1} p_{j-1}^T M_k p_{j-1})$$

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 $||z_0||^2_k = z_0^T M_k z_0 = 0$, $z_0^T M_k p_0 = 0$, and $p_0^T M_k p_0 = v_0^T v_0$. Thus, to obtain $||z_{j+1}||^2_k$ 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 $p_0^T M_k p_0 = g_0^T v_k$.

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 $150 \times 150$ while the Yosemite sequence is of size $316 \times 252$.

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.
to reach a similar accuracy using four optical flow models and ii) the total number of functional and gradient evaluations $N_{fg}$ 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 $A_{red,k}$, 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 $N_{fg}$ is computed as follows,

$$N_{fg} = \frac{N_f K}{K} + N_g$$

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, $w_{i,j}^c$, and the estimated flow, $w_{i,j}^e$, is computed as:

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

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 than 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

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (s)</th>
<th>LSTN</th>
<th>TRTN</th>
<th>MR/LSTN</th>
<th>MR/TRTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>2.84</td>
<td>2.37</td>
<td>0.76</td>
<td>0.59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>686</td>
<td>583</td>
<td>201</td>
<td>165</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>1.46e5</td>
<td>1.45e5</td>
<td>7.54e4</td>
<td>7.55e4</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>1.05</td>
<td>1.00</td>
<td>0.90</td>
<td>0.83</td>
</tr>
<tr>
<td>Model 2</td>
<td>3.68</td>
<td>3.26</td>
<td>1.11</td>
<td>1.00</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>737</td>
<td>624</td>
<td>217</td>
<td>205</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>7.20e83</td>
<td>7.21e83</td>
<td>7.20e3</td>
<td>7.21e3</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>0.24</td>
<td>0.24</td>
<td>0.23</td>
<td>0.23</td>
</tr>
<tr>
<td>Model 3</td>
<td>4.65</td>
<td>3.45</td>
<td>1.52</td>
<td>1.28</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>869</td>
<td>676</td>
<td>308</td>
<td>256</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>8.70e5</td>
<td>8.61e5</td>
<td>3.74e5</td>
<td>3.74e5</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>0.97</td>
<td>0.97</td>
<td>0.82</td>
<td>0.82</td>
</tr>
<tr>
<td>Model 4</td>
<td>6.61</td>
<td>4.55</td>
<td>2.06</td>
<td>1.30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>1044</td>
<td>693</td>
<td>215</td>
<td>222</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>3.03e5</td>
<td>3.03e5</td>
<td>3.03e5</td>
<td>3.03e5</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>0.20</td>
<td>0.20</td>
<td>0.20</td>
<td>0.21</td>
</tr>
<tr>
<td>Total</td>
<td>17.78</td>
<td>13.63</td>
<td>5.45</td>
<td>4.17</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>3336</td>
<td>2576</td>
<td>1070</td>
<td>848</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Model</th>
<th>Time (s)</th>
<th>LSTN</th>
<th>TRTN</th>
<th>MR/LSTN</th>
<th>MR/TRTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model 1</td>
<td>4.46</td>
<td>4.60</td>
<td>0.50</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>1025</td>
<td>1168</td>
<td>125</td>
<td>217</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>2.00e4</td>
<td>2.00e4</td>
<td>2.00e4</td>
<td>2.01e4</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>1.93</td>
<td>1.94</td>
<td>1.89</td>
<td>1.91</td>
</tr>
<tr>
<td>Model 2</td>
<td>3.91</td>
<td>3.17</td>
<td>1.09</td>
<td>1.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>703</td>
<td>611</td>
<td>215</td>
<td>211</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>3.04e4</td>
<td>3.04e4</td>
<td>3.04e4</td>
<td>3.03e4</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>2.03</td>
<td>2.05</td>
<td>2.07</td>
<td>2.08</td>
</tr>
<tr>
<td>Model 3</td>
<td>5.28</td>
<td>4.75</td>
<td>0.61</td>
<td>1.46</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>996</td>
<td>922</td>
<td>120</td>
<td>286</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>2.78e5</td>
<td>2.78e5</td>
<td>2.78e5</td>
<td>2.78e5</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>2.39</td>
<td>2.37</td>
<td>2.36</td>
<td>2.36</td>
</tr>
<tr>
<td>Model 4</td>
<td>6.78</td>
<td>4.28</td>
<td>2.97</td>
<td>1.76</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>1064</td>
<td>669</td>
<td>508</td>
<td>285</td>
</tr>
<tr>
<td></td>
<td>Act. value</td>
<td>2.12e5</td>
<td>2.12e5</td>
<td>2.12e5</td>
<td>2.12e5</td>
</tr>
<tr>
<td></td>
<td>AAE</td>
<td>2.22</td>
<td>2.24</td>
<td>2.11</td>
<td>2.08</td>
</tr>
<tr>
<td>Total</td>
<td>20.43</td>
<td>16.80</td>
<td>5.17</td>
<td>5.03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Nfg</td>
<td>3788</td>
<td>3370</td>
<td>968</td>
<td>999</td>
</tr>
</tbody>
</table>
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

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

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

TRUST REGION METHODS FOR OPTICAL FLOW

17