# Is Levenberg-Marquardt the Most Efficient Optimization Algorithm for Implementing Bundle Adjustment? *

Manolis I.A. Lourakis and Antonis A. Argyros
Institute of Computer Science, Foundation for Research and Technology - Hellas
Vassilika Vouton, P.O. Box 1385, GR 711 10, Heraklion, Crete, GREECE
{lourakis, argyros}@ics.forth.gr

## Abstract

*In order to obtain optimal 3D structure and viewing parameter estimates, bundle adjustment is often used as the last step of feature-based structure and motion estimation algorithms. Bundle adjustment involves the formulation of a large scale, yet sparse minimization problem, which is traditionally solved using a sparse variant of the Levenberg-Marquardt optimization algorithm that avoids storing and operating on zero entries. This paper argues that considerable computational benefits can be gained by substituting the sparse Levenberg-Marquardt algorithm in the implementation of bundle adjustment with a sparse variant of Powell's dog leg non-linear least squares technique. Detailed comparative experimental results provide strong evidence supporting this claim.*

## 1 Introduction

Bundle Adjustment (BA) is often used as the last step of many feature-based 3D reconstruction algorithms; see, for example, [6, 1, 5, 19, 12] for a few representative approaches. Excluding feature tracking, BA is typically the most time consuming computation arising in such algorithms. BA amounts to a large scale optimization problem that is solved by simultaneously refining the 3D structure and viewing parameters (i.e. camera pose and possibly intrinsic calibration), to obtain a reconstruction which is optimal under certain assumptions regarding the noise pertaining to the observed image features. If the image error is zero-mean Gaussian, then BA is the ML estimator. An excellent survey of BA methods is given in [21].

BA boils down to minimizing the reprojection error between the observed and predicted image points, which is expressed as the sum of squares of a large number of non-linear, real-valued functions. Thus, the minimization is achieved using non-linear least squares algorithms, of which the Levenberg-Marquardt (LM) [10, 14] has become very popular due to its relative ease of implementation and its use of an effective damping strategy that lends it the ability to converge promptly from a wide range of initial guesses. By iteratively linearizing the function to be minimized in the neighborhood of the current estimate, the LM algorithm involves the solution of linear systems known as the *normal equations*. When solving minimization problems arising in BA, the normal equations matrix has a sparse block structure owing to the lack of interaction among parameters for different 3D points and cameras. Therefore, a straightforward means of realizing considerable computational gains is to implement BA by developing a tailored, sparse variant of the LM algorithm which explicitly takes advantage of the zeroes pattern in the normal equations [8].

Apart from exploiting sparseness, however, very few research studies for accelerating BA have been published. In particular, the LM algorithm is the de facto standard for most BA implementations [7]. This paper suggests that Powell's dog leg (DL) algorithm [20] is preferable to LM, since it can also benefit from a sparse implementation while having considerably lower computational requirements on large scale problems. The rest of the paper is organized as follows. For the sake of completeness, sections 2 and 3 provide short tutorial introductions to the LM and DL algorithms for solving non-linear least squares problems. Section 4 discusses the performance advantages of DL over LM. Section 5 provides an experimental comparison between LM- and DL-based BA, which clearly demonstrates the superiority of the latter in terms of execution time. The paper is concluded with a brief discussion in section 6.

## 2 Levenberg-Marquardt's Algorithm

The LM algorithm is an iterative technique that locates a local minimum of a multivariate function that is expressed as the sum of squares of several non-linear, real-valued functions. It has become a standard technique for non-linear least-squares problems, widely adopted in various disciplines for dealing with data-fitting applications. LM can be thought of as a combination of steepest descent and

*This paper is dedicated to the memory of our late advisor Prof. Stelios C. Orphanoudakis. Work partially supported by the EU FP6-507752 NoE MUSCLE.

the Gauss-Newton method. When the current solution is far from a local minimum, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to a local minimum, it becomes a Gauss-Newton method and exhibits fast convergence. To help the reader follow the comparison between LM and DL that is made in section 4, a short description of the LM algorithm based on the material in [13] is provided next. Note, however, that a detailed analysis of the LM algorithm is beyond the scope of this paper and the interested reader is referred to [18, 13, 9] for more extensive treatments.

**Input**: A vector function $f : \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.
**Output**: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $||\mathbf{x} - f(\mathbf{p})||^2$.
**Algorithm**:
$k := 0$; $\nu := 2$; $\mathbf{p} := \mathbf{p}_0$;
$\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
stop:=($||\mathbf{g}||_\infty \leq \varepsilon_1$); $\mu := \tau * \max_{i=1,\ldots,m}(A_{ii})$;
while (not stop) and ($k < k_{max}$)
  $k := k + 1$;
  repeat
    Solve $(\mathbf{A} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{g}$;
    if ($||\delta_\mathbf{p}|| \leq \varepsilon_2||\mathbf{p}||$)
      stop:=true;
    else
      $\mathbf{p}_{new} := \mathbf{p} + \delta_\mathbf{p}$;
      $\rho := (||\epsilon_\mathbf{p}||^2 - ||\mathbf{x} - f(\mathbf{p}_{new})||^2)/(\delta_\mathbf{p}^T(\mu\delta_\mathbf{p} + \mathbf{g}))$;
      if $\rho > 0$
        $\mathbf{p} = \mathbf{p}_{new}$;
        $\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_\mathbf{p} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_\mathbf{p}$;
        stop:=($||\mathbf{g}||_\infty \leq \varepsilon_1$);
        $\mu := \mu * \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$; $\nu := 2$;
      else
        $\mu := \mu * \nu$; $\nu := 2 * \nu$;
      endif
    endif
  until ($\rho > 0$) or (stop)
endwhile
$\mathbf{p}^+ := \mathbf{p}$;

**Figure 1. Levenberg-Marquardt non-linear least squares algorithm. $\rho$ is the *gain ratio*, defined by the ratio of the actual reduction in the error $||\epsilon_\mathbf{p}||^2$ that corresponds to a step $\delta_\mathbf{p}$ and the reduction predicted for $\delta_\mathbf{p}$ by the linear model of Eq. (1). See text and [13, 17] for details. When LM is applied to the problem of BA, the operation enclosed in the rectangular box is carried out by taking into account the sparse structure of the corresponding Hessian matrix A.**

In the following, vectors and arrays appear in boldface and $^T$ is used to denote transposition. Also, $||.||$ and $||.||_\infty$ respectively denote the 2 and infinity norms. Let $f$ be an assumed functional relation which maps a *parameter vector* $\mathbf{p} \in \mathcal{R}^m$ to an estimated *measurement vector* $\hat{\mathbf{x}} = f(\mathbf{p})$, $\hat{\mathbf{x}} \in \mathcal{R}^n$. An initial parameter estimate $\mathbf{p}_0$ and a measured vector $\mathbf{x}$ are provided and it is desired to find the vector $\mathbf{p}^+$ that best satisfies the functional relation $f$ locally, i.e. minimizes the squared distance $\epsilon^T\epsilon$ with $\epsilon = \mathbf{x} - \hat{\mathbf{x}}$ for all $\mathbf{p}$ within a sphere having a certain, small radius. The basis of the LM algorithm is a linear approximation to $f$ in the neighborhood of $\mathbf{p}$. Denoting by $\mathbf{J}$ the Jacobian matrix $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$, a Taylor series expansion for a small $||\delta_\mathbf{p}||$ leads to the following approximation:

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p}. \quad (1)$$

Like all non-linear optimization methods, LM is iterative. Initiated at the starting point $\mathbf{p}_0$, it produces a series of vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots$, that converge towards a local minimizer $\mathbf{p}^+$ for $f$. Hence, at each iteration, it is required to find the step $\delta_\mathbf{p}$ that minimizes the quantity

$$||\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p})|| \approx ||\mathbf{x} - f(\mathbf{p}) - \mathbf{J}\delta_\mathbf{p}|| = ||\epsilon - \mathbf{J}\delta_\mathbf{p}||. \quad (2)$$

The sought $\delta_\mathbf{p}$ is thus the solution to a linear least-squares problem: the minimum is attained when $\mathbf{J}\delta_\mathbf{p} - \epsilon$ is orthogonal to the column space of $\mathbf{J}$. This leads to $\mathbf{J}^T(\mathbf{J}\delta_\mathbf{p} - \epsilon) = \mathbf{0}$, which yields the *Gauss-Newton* step $\delta_\mathbf{p}$ as the solution of the so-called *normal equations*:

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T\epsilon. \quad (3)$$

Ignoring the second derivative terms, matrix $\mathbf{J}^T\mathbf{J}$ in (3) approximates the Hessian of $\frac{1}{2}\epsilon^T\epsilon$ [18]. Note also that $\mathbf{J}^T\epsilon$ is along the steepest descent direction, since the gradient of $\frac{1}{2}\epsilon^T\epsilon$ is $-\mathbf{J}^T\epsilon$. The LM method actually solves a slight variation of Eq. (3), known as the *augmented normal equations*:

$$\mathbf{N}\delta_\mathbf{p} = \mathbf{J}^T\epsilon, \text{ with } \mathbf{N} \equiv \mathbf{J}^T\mathbf{J} + \mu\mathbf{I} \text{ and } \mu > 0, \quad (4)$$

where $\mathbf{I}$ is the identity matrix. The strategy of altering the diagonal elements of $\mathbf{J}^T\mathbf{J}$ is called *damping* and $\mu$ is referred to as the *damping term*. If the updated parameter vector $\mathbf{p} + \delta_\mathbf{p}$ with $\delta_\mathbf{p}$ computed from Eq. (4) leads to a reduction in the error $\epsilon^T\epsilon$, the update is accepted and the process repeats with a decreased damping term. Otherwise, the damping term is increased, the augmented normal equations are solved again and the process iterates until a value of $\delta_\mathbf{p}$ that decreases the error is found. The process of repeatedly solving Eq. (4) for different values of the damping term until an acceptable update to the parameter vector is found corresponds to one iteration of the LM algorithm.

In LM, the damping term is adjusted at each iteration to assure a reduction in the error. If the damping is set to

a large value, matrix $\mathbf{N}$ in Eq. (4) is nearly diagonal and the LM update step $\delta_{\mathbf{p}}$ is near the steepest descent direction $\mathbf{J}^T\epsilon$. Moreover, the magnitude of $\delta_{\mathbf{p}}$ is reduced in this case, ensuring that excessively large Gauss-Newton steps are not taken. Damping also handles situations where the Jacobian is rank deficient and $\mathbf{J}^T\mathbf{J}$ is therefore singular [4]. The damping term can be chosen so that matrix $\mathbf{N}$ in Eq. (4) is nonsingular and, therefore, positive definite, thus ensuring that the $\delta_{\mathbf{p}}$ computed from it is in a descent direction. In this way, LM can defensively navigate a region of the parameter space in which the model is highly nonlinear. If the damping is small, the LM step approximates the exact Gauss-Newton step. LM is adaptive because it controls its own damping: it raises the damping if a step fails to reduce $\epsilon^T\epsilon$; otherwise it reduces the damping. By doing so, LM is capable of alternating between a slow descent approach when being far from the minimum and a fast, quadratic convergence when being at the minimum's neighborhood [4]. An efficient updating strategy for the damping term that is also used in this work is described in [17]. The LM algorithm terminates when at least one of the following conditions is met:

- The gradient's magnitude drops below a threshold $\varepsilon_1$.
- The relative change in the magnitude of $\delta_{\mathbf{p}}$ drops below a threshold $\varepsilon_2$.
- A maximum number of iterations $k_{max}$ is reached.

The complete LM algorithm is shown in pseudocode in Fig. 1; more details regarding it can be found in [13]. The initial damping factor is chosen equal to the product of a parameter $\tau$ with the maximum element of $\mathbf{J}^T\mathbf{J}$ in the main diagonal. Indicative values for the user-defined parameters are $\tau = 10^{-3}$, $\varepsilon_1 = \varepsilon_2 = 10^{-12}$, $k_{max} = 100$.

## 3 The Dog Leg Algorithm

Similarly to the LM algorithm, the DL algorithm for unconstrained minimization tries combinations of the Gauss-Newton and steepest descent directions. In the case of DL, however, this is explicitly controlled via the use of a *trust region*. Trust region methods have been studied during the last few decades and have given rise to numerical algorithms that are reliable and robust, possessing strong convergence properties and being applicable even to ill-conditioned problems [2]. In a trust region framework, information regarding the objective function $f$ is gathered and used to construct a quadratic *model function* $L$ whose behavior in the neighborhood of the current point is similar to that of $f$. The model function is trusted to accurately represent $f$ only for points within a hypersphere of radius $\Delta$ centered on the current point, hence the name trust region. A new candidate step minimizing $f$ is then found by

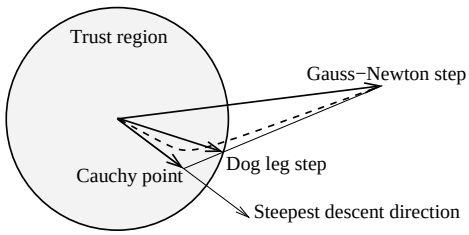


**Figure 2. Dog leg approximation of the curved optimal trajectory (shown dashed). The case with the Cauchy point and the Gauss-Newton step being respectively inside and outside the trust region is illustrated.**

(approximately) minimizing $L$ over the trust region. The model function is chosen as the quadratic corresponding to the squared right hand part of Eq. (2), namely

$$L(\delta) = 2(\frac{1}{2}\epsilon^T\epsilon - (\mathbf{J}^T\epsilon)^T\delta + \frac{1}{2}\delta^T\mathbf{J}^T\mathbf{J}\delta). \quad (5)$$

With the above definition, the candidate step is the solution of the following constrained subproblem:

$$\min_{\delta} L(\delta), \text{ subject to } ||\delta|| \leq \Delta. \quad (6)$$

Clearly, the radius of the trust region is crucial to the success of a step. If the region is too large, the model might not be a good approximation of the objective function and, therefore, its minimizer might be far from the minimizer of the objective function in the region. On the other hand, if the region is too small, the computed candidate step might not suffice to bring the current point closer to the minimizer of the objective function. In practice, the trust region radius is chosen based on the success of the model in approximating the objective function during the previous iterations. If the model is reliable, that is accurately predicting the behavior of the objective function, the radius is increased to allow longer steps to be tested. If the model fails to predict the objective function over the current trust region, the radius of the latter is reduced and (6) is solved again, this time over the smaller trust region.

The solution to the trust region subproblem (6) as a function of the trust region radius is a curve as the one shown in Fig. 2. In a seminal paper, Powell [20] proposed to approximate this curve with a piecewise linear trajectory consisting of two line segments. The first runs from the current point to the *Cauchy point*, defined by the unconstrained minimizer of the objective function along the steepest descent direction $\mathbf{g} = \mathbf{J}^T\epsilon$ and given by

$$\delta_{\mathbf{sd}} = \frac{\mathbf{g}^T\mathbf{g}}{\mathbf{g}^T\mathbf{J}^T\mathbf{J}\mathbf{g}}\mathbf{g}. \quad (7)$$

The second line segment runs from $\delta_{\mathbf{sd}}$ to the Gauss-Newton step $\delta_{\mathbf{gn}}$, defined by the solution of Eq. (3) which is repeated here for convenience:

$$\mathbf{J}^T\mathbf{J}\delta_{\mathbf{gn}} = \mathbf{g}. \quad (8)$$

Since matrix $\mathbf{J}^T\mathbf{J}$ may occasionally become positive semidefinite, Eq. (8) is solved with the aid of a perturbed Cholesky decomposition that ensures positive definiteness [4]. Formally, for $\kappa \in [0, 2]$, the dog leg[1] trajectory is defined as

$$\delta(\kappa) = \begin{cases} \kappa\delta_{\mathbf{sd}}, & 0 \leq \kappa \leq 1 \\ \delta_{\mathbf{sd}} + (\kappa - 1)(\delta_{\mathbf{gn}} - \delta_{\mathbf{sd}}), & 1 \leq \kappa \leq 2 \end{cases} \quad (9)$$

It can be shown [4] that the length of $\delta(\kappa)$ is a monotonically increasing function of $\kappa$ and that the value of the model function for $\delta(\kappa)$ is a monotonically decreasing function of $\kappa$. Also, the Cauchy step is always shorter than the Gauss-Newton step. With the above facts in mind, the dog leg step is defined as follows: If the Cauchy point lies outside the trust region, the DL step is chosen as the *truncated Cauchy step*, i.e. the intersection of the Cauchy step with the trust region boundary that is given by $\frac{\Delta}{||\delta_{\mathbf{sd}}||}\delta_{\mathbf{sd}}$. Otherwise, and if the Gauss-Newton step is within the trust region, the DL step is taken to be equal to it. Finally, when the Cauchy point is in the trust region and the Gauss-Newton step is outside, the next trial point is computed as the intersection of the trust region boundary and the straight line joining the Cauchy point and the Gauss-Newton step (see Fig. 2). In all cases, the dog leg path intersects the trust region boundary at most once and the point of intersection can be determined analytically, without a search. The DL algorithm is described using pseudocode in Fig. 3 and more details can be found in [4, 13]. The strategy just described is also known as the single dog leg, to differentiate it from the double dog leg generalization proposed by Dennis and Mei [3]. The double dog leg is a slight modification of Powell's original algorithm which introduces a bias towards the Gauss-Newton direction that has been observed to result in better performance. Indicative values for the user-defined parameters are $\Delta_0 = 1.0$, $\varepsilon_1 = \varepsilon_2 = 10^{-12}$, $k_{max} = 100$.

At this point, it should be mentioned that there exists a close relation between the augmented normal equations (4) and the solution of (6): A vector $\delta^*$ minimizes (6) if and only if there is a scalar $\mu$ such that $(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})\delta^* = \mathbf{J}^T\epsilon$ and $\mu(\Delta - ||\delta^*||) = 0$ [4]. Based on this result, modern implementations of the LM algorithm such as [15], seek a nearly exact solution for $\mu$ using Newton's root finding algorithm in a trust-region framework rather than directly controlling the damping parameter $\mu$ in Eq. (4) [16]. Note, however, that since no closed form formula providing $\mu$ for

[1]The term *dog leg* comes from golf, indicating a hole with a sharp angle in the fairway.

**Input**: A vector function $f : \mathcal{R}^m \rightarrow \mathcal{R}^n$ with $n \geq m$, a measurement vector $\mathbf{x} \in \mathcal{R}^n$ and an initial parameters estimate $\mathbf{p}_0 \in \mathcal{R}^m$.
**Output**: A vector $\mathbf{p}^+ \in \mathcal{R}^m$ minimizing $||\mathbf{x} - f(\mathbf{p})||^2$.
**Algorithm**:
$k := 0$; $\Delta := \Delta_0$; $\mathbf{p} := \mathbf{p}_0$;
$\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_{\mathbf{p}}$;
stop:=($||\mathbf{g}||_\infty \leq \varepsilon_1$);
while (not stop) and ($k < k_{max}$)
  $k := k + 1$;
  $\delta_{\mathbf{sd}} := \frac{||\mathbf{g}||^2}{||\mathbf{J}\mathbf{g}||^2}\mathbf{g}$;
  GNcomputed:=false;
  repeat
    if $||\delta_{\mathbf{sd}}|| \geq \Delta$
      $\delta_{\mathbf{dl}} := \frac{\Delta}{||\delta_{\mathbf{sd}}||}\delta_{\mathbf{sd}}$;
    else
      if (not GNcomputed)
        Solve $\mathbf{A}\delta_{\mathbf{gn}} = \mathbf{g}$;
        GNcomputed:=true;
      endif
      if $||\delta_{\mathbf{gn}}|| \leq \Delta$
        $\delta_{\mathbf{dl}} := \delta_{\mathbf{gn}}$;
      else
        $\delta_{\mathbf{dl}} := \delta_{\mathbf{sd}} + \beta(\delta_{\mathbf{gn}} - \delta_{\mathbf{sd}})$; // $\beta$ *s.t.* $||\delta_{\mathbf{dl}}|| = \Delta$
      endif
    endif
    if ($||\delta_{\mathbf{dl}}|| \leq \varepsilon_2||\mathbf{p}||$)
      stop:=true;
    else
      $\mathbf{p}_{new} := \mathbf{p} + \delta_{\mathbf{dl}}$;
      $\rho := (||\epsilon_{\mathbf{p}}||^2 - ||\mathbf{x} - f(\mathbf{p}_{new})||^2)/(L(0) - L(\delta_{\mathbf{dl}}))$;
      if $\rho > 0$
        $\mathbf{p} = \mathbf{p}_{new}$;
        $\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_{\mathbf{p}}$;
        stop:=($||\mathbf{g}||_\infty \leq \varepsilon_1$);
      endif
      $\Delta$:=*updateRadius*($\rho$, $\Delta$, 0.25, 0.75); // *update* $\Delta$
      stop:=$\Delta \leq \varepsilon_2||\mathbf{p}||$;
    endif
  until ($\rho > 0$) or (stop)
endwhile
$\mathbf{p}^+ := \mathbf{p}$;

**Figure 3. Powell's dog leg algorithm for non-linear least squares. $\Delta_0$ is the initial trust region radius. Routine *updateRadius*() controls the trust region radius based on the value of the gain ratio $\rho$ of actual over predicted reduction: $\Delta$ is increased if $\rho > 0.75$, kept constant if $0.25 \leq \rho \leq 0.75$ and decreased if $\rho < 0.25$. A detailed description of *updateRadius*() can be found in section 6.4.3 of [4]. Again, when dealing with BA, the operation enclosed in the rectangular box is carried out by taking into account the sparse structure of matrix $\mathbf{A}$.**

a given $\Delta$ exists, this approach requires expensive repetitive Cholesky factorizations of the augmented approximate Hessian and, therefore, is not well-suited to solving large scale problems such as those arising in the context of BA.

## 4 DL vs. LM: Performance Issues

We are now in the position to proceed to a qualitative comparison of the computational requirements of the LM and DL algorithms. When a LM step fails, the LM algorithm requires that the augmented equations resulting from an increased damping term are solved again. In other words, every update to the damping term calls for a new solution of the augmented equations, thus failed steps entail unproductive effort. On the contrary, once the Gauss-Newton step has been determined, the DL algorithm can solve the constrained quadratic subproblem for various values of $\Delta$, without resorting to solving again Eq. (8). Note also that when the truncated Cauchy step is taken, the DL algorithm can avoid solving Eq. (8), while the LM algorithm always needs to solve (4), even if it chooses a step smaller than the Cauchy step. Reducing the number of times that Eq. (8) needs to be solved is crucial for the overall performance of the minimization process, since the BA problem involves many parameters and, therefore, linear algebra costs dominate the computational overhead associated with every inner iteration of both algorithms in Figs. 1 and 3. For the above reasons, the DL algorithm is a more promising implementation of the non-linear minimization arising in BA in terms of the required computational effort.

## 5 Experimental Results

This section provides an experimental comparison regarding the use of the LM and DL algorithms for solving the sparse BA problem. Both algorithms were implemented in C, using LAPACK for linear algebra numerical operations. The LM BA implementation tested is that included in the `sba` package (`http://www.ics.forth.gr/~lourakis/sba`) [11] that we have recently made publicly available. Representative results from the application of the two algorithms for carrying out Euclidean BA on eight different real image sequences are given next. It is stressed that both implementations share the same data structures as well as a large percentage of the same core code and have been extensively optimized and tested. Therefore, the reported dissimilarities in execution performance are exclusively due to the algorithmic differences between the two techniques rather than due to the details of the particular implementations.

In all experiments, it is assumed that a set of 3D points are seen in a number of images acquired by an intrinsically calibrated moving camera and that the image projections of each 3D point have been identified. Estimates of the Euclidean 3D structure and camera motions are then computed using the sequential structure and motion estimation technique of [12]. Those estimates serve as starting points for bootstrapping refinements that are based on Euclidean BA using the DL and LM algorithms. Equations (4) and (8) are solved by exploiting the sparse structure of the Hessian matrix $\mathbf{J}^T\mathbf{J}$, as described in Appendix 4 of [8]. Camera motions corresponding to all but the first frame are defined relative to the initial camera location. The former is taken to coincide with the employed world coordinate frame. Camera rotations are parameterized by quaternions while translations and 3D points by 3D vectors. The set of employed sequences includes the "movi" toy house circular sequence from INRIA's MOVI group, "sagalassos" and "arenberg" from Leuven's VISICS group, "basement" and "house" from Oxford's VGG group and three sequences acquired by ourselves, namely "maquette", "desk" and "calgrid".

Table 1 illustrates several statistics gathered from the application of DL and LM-based Euclidean BA to the eight test sequences. Each row corresponds to a single sequence and columns are as follows: The first column corresponds to the total number of images that were employed in BA. The second column is the total number of motion and structure variables pertaining to the minimization. The third column corresponds to the average squared reprojection error of the initial reconstruction. The fourth column (labeled "final error") shows the average squared reprojection error after BA for both algorithms. The fifth column shows the total number of objective function/jacobian evaluations during BA. The number of iterations needed for convergence and the total number of linear systems (i.e. Eq. (4) for LM and Eq. (8) for DL) that were solved are shown in the sixth column. The last column shows the time (in seconds) elapsed during execution of BA. All experiments were conducted on an Intel P4@1.8 GHz running Linux and unoptimized BLAS. As it is evident from the final squared reprojection error, both approaches converge to almost identical solutions. However, DL-based BA is between 2.0 to 7.5 times faster, depending on the sequence used as benchmark.

These speedups can be explained by comparing the total number of iterations as well as that of evaluations of the objective function and corresponding jacobian. The DL algorithm converges in considerably less iterations, requiring fewer linear systems to be solved and fewer objective function/jacobian evaluations. Note that the difference in performance between the two methods is more pronounced for long sequences, since, in such cases, the costs of solving linear systems are increased. These experimental findings agree with the remarks made in section 4 regarding the qualitative comparison of the DL and LM algorithms. Apart

| Sequence | # imgs | # vars | initial error | final error | | func/jac evals | | iter./sys. solved | | exec. time | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | DL | LM | DL | LM | DL | LM | DL | LM |
| "movi" | 59 | 5747 | 5.03 | 0.3159 | 0.3159 | 11/3 | 18/18 | 3/3 | 18/18 | 1.06 | 5.03 |
| "sagalassos" | 26 | 5309 | 11.04 | 1.2704 | 1.2703 | 16/6 | 44/33 | 6/6 | 33/44 | 1.29 | 6.98 |
| "arenberg" | 22 | 4159 | 1.35 | 0.5400 | 0.5399 | 11/3 | 25/20 | 3/3 | 20/25 | 0.75 | 4.95 |
| "basement" | 11 | 992 | 0.37 | 0.2448 | 0.2447 | 15/9 | 29/21 | 9/9 | 21/29 | 0.18 | 0.37 |
| "house" | 10 | 1615 | 1.43 | 0.2142 | 0.2142 | 10/3 | 25/19 | 3/3 | 19/25 | 0.11 | 0.51 |
| "maquette" | 54 | 15999 | 2.15 | 0.1765 | 0.1765 | 12/3 | 31/23 | 3/3 | 23/31 | 1.77 | 12.16 |
| "desk" | 46 | 10588 | 4.16 | 1.5761 | 1.5760 | 11/3 | 32/23 | 3/3 | 23/32 | 1.27 | 9.58 |
| "calgrid" | 27 | 2355 | 3.21 | 0.2297 | 0.2297 | 10/3 | 20/20 | 3/3 | 20/20 | 2.61 | 16.96 |

**Table 1. Statistics for Euclidean BA using the DL and LM algorithms: Total number of images, total number of variables, average initial squared reprojection error in pixels, average final squared reprojection error in pixels, total number of objective function/jacobian evaluations, total number of iterations and linear systems solved, elapsed execution time in seconds. Identical values for the user-defined minimization parameters have been used throughout all experiments.**

from the above experiments, we have compared the performance of the LM and DL BA implementations on several more image sequences. In all cases, the DL variant was found to consistently outperform the LM one.

## 6 Conclusions

A computationally efficient implementation of BA is beneficial to a wide range of vision tasks that are related to 3D reconstruction. This paper has questioned the current state of practice by suggesting that Powell's dog leg minimization algorithm is a more attractive alternative to Levenberg-Marquardt. As it has been demonstrated experimentally, the DL algorithm computes solutions that are of the same quality compared to those of LM, albeit at a fraction of time. Given the popularity of BA, speeding it up using the DL algorithm constitutes an important contribution towards the development of practical vision systems.

The introduction of trust region optimization methods to the solution of the BA problem has an additional important consequence. Specifically, trust region methods are easier to extend for dealing with constrained optimization problems (e.g. refer to parts III - IV of [2]). Therefore, they can support the development of constrained versions of BA, a task that is difficult to deal with using the standard LM-based approach.

## References


[1] P. Beardsley, P. Torr, and A. Zisserman. 3D Model Acquisition From Extended Image Sequences. In *Proc. of ECCV'96*, pages 683–695, 1996.

[2] A. Conn, N. Gould, and P. Toint. *Trust Region Methods*. MPS-SIAM Series On Optimization. SIAM, Philadelphia, PA, 2000.

[3] J. Dennis and H. Mei. Two New Unconstrained Optimization Algorithms Which Use Function and Gradient Values. *Journal of Optimization Theory and Applications*, 28:453–462, 1979.

[4] J. Dennis and R. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. SIAM Publications, Philadelphia, 1996.

[5] A. Fitzgibbon and A. Zisserman. Automatic Camera Recovery for Closed or Open Image Sequences. In *Proceedings of ECCV'98*, pages 311–326, 1998.

[6] R. Hartley. Euclidean Reconstruction from Uncalibrated Views. In J. Mundy and A. Zisserman, editors, *Applications of Invariance in Computer Vision*, volume 825 of Lecture Notes in Computer Science, pages 237–256. Springer-Verlag, 1993.

[7] R. Hartley. An Object-Oriented Approach to Scene Reconstruction. In *Proc. of IEEE Conf. SM&C'96*, volume 4, pages 2475–2480, Oct. 1996.

[8] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 1st edition, 2000.

[9] C. Kelley. *Iterative Methods for Optimization*. SIAM Publications, Philadelphia, 1999.

[10] K. Levenberg. A Method for the Solution of Certain Non-linear Problems in Least Squares. *Quarterly of Applied Mathematics*, 2(2):164–168, Jul. 1944.

[11] M. Lourakis and A. Argyros. The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Greece, Aug. 2004. Available at `ftp://ftp.ics.forth.gr/tech-reports/2004`.

[12] M. Lourakis and A. Argyros. Efficient, Causal Camera Tracking in Unprepared Environments. *Computer Vision and Image Understanding Journal*, 99(2):259–290, Aug. 2005.

[13] K. Madsen, H. Nielsen, and O. Tingleff. Methods for Non-Linear Least Squares Problems. Technical University of Denmark, 2004. Lecture notes, available at `http://www.imm.dtu.dk/courses/02611/nllsq.pdf`.

[14] D. Marquardt. An Algorithm for the Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal of Applied Mathematics*, 11(2):431–441, Jun. 1963.

[15] J. Moré, B. Garbow, and K. Hillstrom. User guide for MINPACK-1. Technical Report ANL-80-74, Argonne National Laboratory, Aug. 1980.

[16] J. Moré and D. Sorensen. Computing a Trust Region Step. *SIAM J. Sci. Statist. Comput.*, 4:553–572, 1983.

[17] H. Nielsen. Damping Parameter in Marquardt's Method. Technical Report IMM-REP-1999-05, Technical University of Denmark, 1999. Available at `http://www.imm.dtu.dk/~hbn`.

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

[19] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual Modeling With a Hand-Held Camera. *IJCV*, 59(3):207–232, Sep./Oct. 2004.

[20] M. Powell. A Hybrid Method for Nonlinear Equations. In P. Rabinowitz, editor, *Numerical Methods for Nonlinear Algebraic Equations*, pages 87–144. Gordon and Breach Science, London, 1970.

[21] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, 1999.