Exploiting the Sparseness of Bundle Adjustment for Efficient 3D Reconstruction*

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 – http://www.ics.forth.gr/cvrl/

Abstract. Bundle adjustment amounts to a large, nonlinear least squares optimization problem that is often used as the last step of feature-based structure and motion estimation vision algorithms to obtain optimal estimates. Due to the very large number of parameters involved, a general purpose implementation of a nonlinear least squares algorithm incurs high computational and memory storage costs when applied to bundle adjustment. Fortunately, the lack of interaction among certain subgroups of parameters results in the corresponding Jacobian exhibiting a sparse block structure. In this paper we outline the mathematics of `sba`, our publicly available software package for generic bundle adjustment that exploits sparseness to achieve considerable computational savings. In addition, we provide experimental results demonstrating that `sba` can efficiently handle large bundle adjustment problems which would be intractable with general purpose nonlinear least squares implementations.

1 Introduction

Three dimensional (3D) reconstruction is a problem whose solution is called for by a wide spectrum of computer vision tasks. Generally speaking, 3D reconstruction can be defined as the problem of using 2D measurements arising from a set of images depicting the same scene from different viewpoints, aiming to derive information related to the 3D scene geometry as well as the relative motion and the optical characteristics of the camera(s) employed to acquire these images. Bundle Adjustment (BA) is almost invariably used as the last step of every feature-based 3D reconstruction algorithm; see, for example, [1, 4, 15, 9] for a few representative approaches. BA amounts to an optimization problem that is solved by simultaneously refining the 3D structure and viewing parameters (i.e., camera pose and possibly intrinsic calibration and radial distortion), to obtain a reconstruction which is optimal under certain assumptions regarding the noise pertaining to the observed image features [17]: If the image error is zero-mean Gaussian, then BA is the Maximum Likelihood Estimator. Its name refers to the "bundles" of light rays originating from each 3D feature and converging on each camera's optical center, which are adjusted optimally with respect to both the structure and viewing parameters. BA was originally conceived in the field of

* This work was partially supported by the EU FP6-507752 NoE MUSCLE and COOP-CT-2005-017405 project RECOVER.

photogrammetry during 1950's [2] and has increasingly been used by computer vision researchers during recent years. An excellent overview of its application to vision-based reconstruction is given in [17]. Apart from computer vision, BA can also find applications in areas such as robotics, image-based computer graphics, digital photogrammetry, remote sensing, etc.

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 nonlinear, real-valued functions. Thus, the minimization is achieved using nonlinear least squares algorithms, from which Levenberg-Marquardt (LM) has proven to be of the most successful due to its ease of implementation and its use of an effective damping strategy that lends it the ability to converge quickly 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*. Considering that the normal equations are solved repeatedly in the course of the LM algorithm and that each computation of the solution to a dense linear system has complexity $O(N^3)$ in the number of unknown parameters, it is clear that general purpose LM codes such as, for example, MINPACK's LMDER routine [11], are computationally very demanding when employed to minimize functions depending on a large number of parameters. The situation is further complicated by the fact that the size of the Jacobian of the objective function also increases with the number of parameters. Thus, when performing operations involving such a large Jacobian, special care has to be taken to avoid thrashing, i.e., instead of performing useful computations, wasting most CPU cycles for writing out virtual memory pages and reading them back in. Fortunately, 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, considerable computational benefits can be gained by developing a tailored, sparse variant of the LM algorithm which explicitly takes advantage of the normal equations zeroes pattern by avoiding storing and operating on zero elements.

The contribution of this paper is twofold. First, it describes a strategy for efficiently dealing with the problem of BA and materializes it through the implementation of a software package. Second, it demonstrates experimentally the impact that the exploitation of the problem's structure has on computational performance. The practical outcome of this work is `sba`, a generic sparse BA package implemented in ANSI C. C was preferred over higher level programming environments such as MATLAB owing to its far superior execution performance and its wide availability in a diverse range of computer systems. `sba` is also usable from C++ and is generic in the sense that it grants the user full control over the choice of parameters and functional relations describing cameras, 3D structure and image projections. Therefore, it can support a wide range of manifestations/parameterizations of the multiple view reconstruction problem such as metric or arbitrary projective/affine cameras, partially or fully intrinsically calibrated cameras, exterior orientation (i.e., pose) esti-

mation from fixed 3D points, 3D reconstruction from extrinsically calibrated images, refinement of intrinsic calibration parameters, etc. The `sba` package can be downloaded in source form from `http://www.ics.forth.gr/~lourakis/sba` and is distributed under the terms of the GNU General Public License (see `http://www.gnu.org/copyleft/gpl.html`).

The rest of the paper is organized as follows. Section 2 briefly explains the conventional, dense LM algorithm for solving nonlinear least squares minimization problems. Section 3 develops a sparse BA algorithm by adapting the LM to exploit the sparse block structure of the normal equations. Experimental results from the application of the developed `sba` package on real problems are presented in section 4 and the paper is concluded with a brief discussion in section 5.

2 The Levenberg-Marquardt Algorithm

The LM algorithm is an iterative technique that finds a local minimum of a multivariate function that is expressed as the sum of squares of nonlinear real-valued functions. It has become a standard technique for nonlinear 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 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. For the sake of completeness, a short description of the LM algorithm based on the material in [10] is supplied 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 [14, 7, 10] for more extensive treatments.

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 an affine approximation to f in the neighborhood of $\mathbf{p}$. For a small $||\delta_\mathbf{p}||$, f is approximated by (see [3], p. 75)

$$f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p}, \tag{1}$$

where $\mathbf{J}$ is the Jacobian of f. Like all nonlinear 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 $||\mathbf{x} - f(\mathbf{p}+\delta_\mathbf{p})|| \approx ||\mathbf{x} - f(\mathbf{p}) - \mathbf{J}\delta_\mathbf{p}|| = ||\epsilon - \mathbf{J}\delta_\mathbf{p}||$. 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 $\delta_\mathbf{p}$ as the solution of the so-called *normal equations*:

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T\epsilon. \tag{2}$$

Matrix $\mathbf{J}^T\mathbf{J}$ in the above equation is the first order approximation to the Hessian of $\frac{1}{2}\epsilon^T\epsilon$ [14] and $\delta_\mathbf{p}$ is the *Gauss-Newton* step. Note also that $\mathbf{J}^T\epsilon$ corresponds to 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. (2), known as the *augmented normal equations*

$$\mathbf{N}\delta_\mathbf{p} = \mathbf{J}^T\epsilon, \text{ with } \mathbf{N} \equiv \mathbf{J}^T\mathbf{J} + \mu\mathbf{I}, \ \mu > 0. \tag{3}$$

The strategy of altering the diagonal elements of $\mathbf{J}^T\mathbf{J}$ is called *damping* and μ is referred to as the *damping term*. If the updated parameter vector $\mathbf{p} + \delta_\mathbf{p}$ with $\delta_\mathbf{p}$ computed from Eq. (3) 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. (3) 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. (3) is nearly diagonal and the LM update step $\delta_\mathbf{p}$ is skewed towards 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 [3]. The damping term can be chosen so that matrix $\mathbf{N}$ in Eq. (3) is safely nonsingular and, therefore, positive definite, thus ensuring that the $\delta_\mathbf{p}$ computed from it is a descent direction. If the damping is small, the LM step approximates the Newton minimizer of the local quadratic model $m(\delta_\mathbf{p}) = \frac{1}{2}\epsilon^T\epsilon - (\mathbf{J}^T\epsilon)^T\delta_\mathbf{p} + \frac{1}{2}\delta_\mathbf{p}^T\mathbf{J}^T\mathbf{J}\delta_\mathbf{p}$ of $\frac{1}{2}\epsilon^T\epsilon$ about $\mathbf{p}$ (cf. Eq. (2)), which is exact in the case of a fully linear problem [7]. 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 convergence when being at the minimum's neighborhood. An efficient updating strategy for the damping term that is also used in this work is described in [13]. The LM algorithm terminates when at least one of the following conditions is met:

- The magnitude of the gradient drops below a threshold ε_1.
- The relative change in the magnitude of $\delta_\mathbf{p}$ drops below a threshold depending on a parameter ε_2.
- The magnitude of the residual ϵ drops below a threshold ε_3.
- A maximum number of iterations k_{max} is reached.

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 $(||\delta_\mathbf{p}|| \leq \varepsilon_2(||\mathbf{p}|| + \varepsilon_2))$
            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)
    stop:=$(||\epsilon_\mathbf{p}|| \leq \varepsilon_3)$;
endwhile
$\mathbf{p}^+ := \mathbf{p}$;

Fig. 1. Pseudocode for the Levenberg-Marquardt nonlinear least-squares algorithm; see text for details. ρ 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). The sign of ρ determines whether $\delta_\mathbf{p}$ is accepted or not. Furthermore, in the case of accepted steps, the value of ρ controls the reduction in the damping term. The reason for enclosing a statement in a rectangular box will be explained in section 3.

If a covariance matrix $\mathbf{\Sigma_x}$ for the measured vector $\mathbf{x}$ is available, it can be incorporated into the LM algorithm by minimizing the squared $\mathbf{\Sigma_x^{-1}}$-norm $\epsilon^T\mathbf{\Sigma_x^{-1}}\epsilon$ instead of the Euclidean norm $\epsilon^T\epsilon$. Accordingly, the minimum is found by solving a weighted least squares problem defined by the augmented *weighted normal equations*

$$(\mathbf{J}^T\mathbf{\Sigma_x^{-1}}\mathbf{J} + \mu\mathbf{I})\delta_\mathbf{p} = \mathbf{J}^T\mathbf{\Sigma_x^{-1}}\epsilon. \tag{4}$$

The rest of the algorithm remains unchanged. It is noted that the initial damping factor is chosen equal to the product of a parameter τ with the maximum element of $\mathbf{J}^T\mathbf{J}$ in the main diagonal. The complete LM algorithm is shown in pseudocode in Fig. 1; more details regarding it can be found in [10]. Indicative values for the user-defined parameters are $\tau = 10^{-3}$, $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = 10^{-12}$, $k_{max} = 100$.

At this point, it should be mentioned that rather than directly controlling the damping parameter μ in Eq. (3), modern implementations of the Levenberg-Marquardt algorithm such as [11], seek a nearly exact solution for μ using Newton's root finding algorithm in a trust-region framework [12]. This approach, however, 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.

3 Sparse Bundle Adjustment

This section shows how a sparse variant of the LM algorithm presented in section 2 can be developed to deal efficiently with the problem of bundle adjustment. The developments that follow are along the lines of the presentation regarding sparse bundle adjustment in Appendix 4 of [5]. To begin, assume that n 3D points are seen in m views and let $\mathbf{x}_{ij}$ be the projection of the i-th point on image j. Bundle adjustment is equivalent to jointly refining a set of initial camera and structure parameter estimates for finding the set of parameters that most accurately predict the locations of the observed n points in the set of the m available images. More formally, assume that each camera j is parameterized by a vector $\mathbf{a}_j$ and each 3D point i by a vector $\mathbf{b}_i$. For notational simplicity it is also assumed that all points are visible in all images. This assumption, however, is not necessary and, as will soon be made clear, points may in general be visible in any subset of the m views. BA minimizes the *reprojection error* with respect to all 3D point and camera parameters, specifically

$$\min_{\mathbf{a}_j, \mathbf{b}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i),\ \mathbf{x}_{ij})^2, \tag{5}$$

where $\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)$ is the predicted projection of point i on image j and $d(\mathbf{x},\ \mathbf{y})$ denotes the Euclidean distance between the image points represented by the inhomogeneous vectors $\mathbf{x}$ and $\mathbf{y}$. It is clear from (5) that BA is by definition tolerant to missing image projections and, in contrast to algebraic approaches for multiview reconstruction, minimizes a physically meaningful criterion. Observe that through $\mathbf{Q}()$, the definition in (5) is general enough to accommodate any camera and structure parameterization. Note also that if κ and λ are respectively the dimensions of each $\mathbf{a}_j$ and $\mathbf{b}_i$, the total number of minimization parameters in (5) equals $m\kappa + n\lambda$ and is therefore large even for BA problems defined for rather short sequences. For example, in the case of projective reconstruction, $\kappa = 12$ and $\lambda = 3$ and, therefore, a moderately sized BA problem defined by, say, 1000 points projecting on each of 30 images involves 3360 variables.

BA can be cast as a nonlinear minimization problem as follows. A parameter vector $\mathbf{P} \in \mathcal{R}^M$ is defined by all parameters describing the m projection matrices and the n 3D points in Eq. (5), namely $\mathbf{P} = ({\mathbf{a}_1}^T, \ldots, {\mathbf{a}_m}^T, \ldots, {\mathbf{b}_1}^T, \ldots, {\mathbf{b}_n}^T)^T$. A measurement vector $\mathbf{X} \in \mathcal{R}^N$ is made up of the measured image point coordinates across all cameras:

$$\mathbf{X} = ({\mathbf{x}_{11}}^T, \ldots, {\mathbf{x}_{1m}}^T,\ {\mathbf{x}_{21}}^T, \ldots, {\mathbf{x}_{2m}}^T,\ \ldots,\ {\mathbf{x}_{n1}}^T, \ldots, {\mathbf{x}_{nm}}^T)^T. \tag{6}$$

Let $\mathbf{P}_0$ be an initial parameter estimate and $\mathbf{\Sigma_X}$ the covariance matrix corresponding to the measured vector $\mathbf{X}$ (in the absence of any further knowledge, it is assumed that $\mathbf{\Sigma_X}$ is the identity matrix). For each parameter vector, an estimated measurement vector $\hat{\mathbf{X}}$ is generated by a functional relation $\hat{\mathbf{X}} = f(\mathbf{P})$, defined by

$$\hat{\mathbf{X}} = ({\hat{\mathbf{x}}_{11}}^T, \ldots, {\hat{\mathbf{x}}_{1m}}^T,\ {\hat{\mathbf{x}}_{21}}^T, \ldots, {\hat{\mathbf{x}}_{2m}}^T,\ \ldots,\ {\hat{\mathbf{x}}_{n1}}^T, \ldots, {\hat{\mathbf{x}}_{nm}}^T)^T, \tag{7}$$

with $\hat{\mathbf{x}}_{ij} = \mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i)$.

Thus, BA corresponds to minimizing the squared $\mathbf{\Sigma}_{\mathbf{X}}^{-1}$-norm (i.e., Mahalanobis distance) $\epsilon^T \mathbf{\Sigma}_{\mathbf{X}}^{-1} \epsilon$, $\epsilon = \mathbf{X} - \hat{\mathbf{X}}$ over $\mathbf{P}$. Evidently, this minimization problem can be solved by employing the LM nonlinear least squares algorithm, which calls for repeatedly solving the augmented weighted normal equations

$$(\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{X}}^{-1} \mathbf{J} + \mu \mathbf{I}) \delta = \mathbf{J}^T \mathbf{\Sigma}_{\mathbf{X}}^{-1} \epsilon, \tag{8}$$

where $\mathbf{J}$ is the Jacobian of f and δ is the sought update to the parameter vector $\mathbf{P}$. As demonstrated below, the normal equations in Eq. (8) have a regular sparse block structure that results from the lack of interaction between parameters of different cameras and different 3D points. To keep the demonstration manageable, a case with small n and m is worked out in detail; however, as will later become apparent, the results are straightforward to generalize to arbitrary numbers of 3D points and cameras.

Assume that $n = 4$ points are visible in $m = 3$ views. The measurement vector is $\mathbf{X} = ({\mathbf{x}_{11}}^T, {\mathbf{x}_{12}}^T, {\mathbf{x}_{13}}^T, {\mathbf{x}_{21}}^T, {\mathbf{x}_{22}}^T, {\mathbf{x}_{23}}^T, {\mathbf{x}_{31}}^T, {\mathbf{x}_{32}}^T, {\mathbf{x}_{33}}^T, {\mathbf{x}_{41}}^T, {\mathbf{x}_{42}}^T, {\mathbf{x}_{43}}^T)^T$. The parameter vector is given by $\mathbf{P} = ({\mathbf{a}_1}^T, {\mathbf{a}_2}^T, {\mathbf{a}_3}^T, {\mathbf{b}_1}^T, {\mathbf{b}_2}^T, {\mathbf{b}_3}^T, {\mathbf{b}_4}^T)^T$. Notice that $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_k} = \mathbf{0}$, $\forall\ j \neq k$ and $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_k} = \mathbf{0}$, $\forall\ i \neq k$. Let $\mathbf{A}_{ij}$ and $\mathbf{B}_{ij}$ denote $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{a}_j}$ and $\frac{\partial \hat{\mathbf{x}}_{ij}}{\partial \mathbf{b}_i}$ respectively. The LM updating vector δ can be partitioned into camera and structure parameters as $({\delta_{\mathbf{a}}}^T, {\delta_{\mathbf{b}}}^T)^T$ and further as $({\delta_{\mathbf{a}_1}}^T, {\delta_{\mathbf{a}_2}}^T, {\delta_{\mathbf{a}_3}}^T, {\delta_{\mathbf{b}_1}}^T, {\delta_{\mathbf{b}_2}}^T, {\delta_{\mathbf{b}_3}}^T, {\delta_{\mathbf{b}_4}}^T)^T$. The remainder of this section is devoted to elaborating a scheme for efficiently solving the normal equations arising in LM minimization by taking advantage of their sparse structure.

Taking into account the notation for the derivatives introduced in the previous paragraph, the Jacobian $\mathbf{J}$ is given by

$$\frac{\partial \mathbf{X}}{\partial \mathbf{P}} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{12} & \mathbf{0} & \mathbf{B}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{13} & \mathbf{B}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{21} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{21} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{22} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{23} & \mathbf{0} & \mathbf{B}_{23} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{31} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_{32} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{32} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{33} & \mathbf{0} \\ \mathbf{A}_{41} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{41} \\ \mathbf{0} & \mathbf{A}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{42} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{43} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{43} \end{pmatrix}. \tag{9}$$

Eq. (9) clearly reveals the sparse nature of the matrix $\mathbf{J}$. It is due to $\mathbf{J}$'s sparseness that the normal equations are themselves sparse. Let the covariance matrix for the complete measurement vector be the block diagonal matrix

$$\mathbf{\Sigma}_{\mathbf{X}} = diag(\mathbf{\Sigma}_{\mathbf{x}_{11}}, \mathbf{\Sigma}_{\mathbf{x}_{12}}, \mathbf{\Sigma}_{\mathbf{x}_{13}}, \mathbf{\Sigma}_{\mathbf{x}_{21}}, \mathbf{\Sigma}_{\mathbf{x}_{22}}, \mathbf{\Sigma}_{\mathbf{x}_{23}}, \mathbf{\Sigma}_{\mathbf{x}_{31}}, \mathbf{\Sigma}_{\mathbf{x}_{32}}, \mathbf{\Sigma}_{\mathbf{x}_{33}}, \mathbf{\Sigma}_{\mathbf{x}_{41}}, \mathbf{\Sigma}_{\mathbf{x}_{42}}, \mathbf{\Sigma}_{\mathbf{x}_{43}}). \tag{10}$$

Substituting $\mathbf{J}$ and $\mathbf{\Sigma}_{\mathbf{X}}^{-1}$ from Eqs. (9), (10) and denoting

$$\mathbf{U}_j \equiv \sum_{i=1}^{4} \mathbf{A}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \mathbf{A}_{ij},\ \mathbf{V}_i \equiv \sum_{j=1}^{3} \mathbf{B}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \mathbf{B}_{ij},\ \mathbf{W}_{i\equiv} \mathbf{A}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \mathbf{B}_{ij}, \qquad (11)$$

the matrix product in the left hand side of Eq. (8) becomes

$$\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{X}}^{-1} \mathbf{J} = \begin{pmatrix} \mathbf{U}_1 & \mathbf{0} & \mathbf{0} & \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\ \mathbf{0} & \mathbf{U}_2 & \mathbf{0} & \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_3 & \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \\ \mathbf{W}_{11}{}^T & \mathbf{W}_{12}{}^T & \mathbf{W}_{13}{}^T & \mathbf{V}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{21}{}^T & \mathbf{W}_{22}{}^T & \mathbf{W}_{23}{}^T & \mathbf{0} & \mathbf{V}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{31}{}^T & \mathbf{W}_{32}{}^T & \mathbf{W}_{33}{}^T & \mathbf{0} & \mathbf{0} & \mathbf{V}_3 & \mathbf{0} \\ \mathbf{W}_{41}{}^T & \mathbf{W}_{42}{}^T & \mathbf{W}_{43}{}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{V}_4 \end{pmatrix}. \qquad (12)$$

Also, using Eqs. (9) and (10), the right hand side of Eq. (8) can be expanded as

$$\left(\sum_{i=1}^{4} \mathbf{A}_{i1}{}^T \mathbf{\Sigma}_{\mathbf{x}_{i1}}^{-1} \epsilon_{i1},\ \ldots,\ \sum_{i=1}^{4} \mathbf{A}_{i3}{}^T \mathbf{\Sigma}_{\mathbf{x}_{i3}}^{-1} \epsilon_{i3},\ \sum_{j=1}^{3} \mathbf{B}_{1j}{}^T \mathbf{\Sigma}_{\mathbf{x}_{1j}}^{-1} \epsilon_{1j},\ \ldots,\ \sum_{j=1}^{3} \mathbf{B}_{4j}{}^T \mathbf{\Sigma}_{\mathbf{x}_{4j}}^{-1} \epsilon_{4j} \right)^T. \qquad (13)$$

Letting

$$\epsilon_{\mathbf{a}_j} \equiv \sum_{i=1}^{4} \mathbf{A}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij},\ \epsilon_{\mathbf{b}_i} \equiv \sum_{j=1}^{3} \mathbf{B}_{ij}{}^T \mathbf{\Sigma}_{\mathbf{x}_{ij}}^{-1} \epsilon_{ij}, \qquad (14)$$

vector (13) can be abbreviated to

$$(\epsilon_{\mathbf{a}_1}{}^T, \epsilon_{\mathbf{a}_2}{}^T, \epsilon_{\mathbf{a}_3}{}^T, \epsilon_{\mathbf{b}_1}{}^T, \epsilon_{\mathbf{b}_2}{}^T, \epsilon_{\mathbf{b}_3}{}^T, \epsilon_{\mathbf{b}_4}{}^T)^T. \qquad (15)$$

Substituting the expressions for $\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{X}}^{-1} \mathbf{J}$ and $\mathbf{J}^T \mathbf{\Sigma}_{\mathbf{X}}^{-1} \epsilon$ from (12) and (15), the normal equations (8) become

$$\begin{pmatrix} \mathbf{U}_1 & \mathbf{0} & \mathbf{0} & \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\ \mathbf{0} & \mathbf{U}_2 & \mathbf{0} & \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_3 & \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \\ \mathbf{W}_{11}{}^T & \mathbf{W}_{12}{}^T & \mathbf{W}_{13}{}^T & \mathbf{V}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{21}{}^T & \mathbf{W}_{22}{}^T & \mathbf{W}_{23}{}^T & \mathbf{0} & \mathbf{V}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{31}{}^T & \mathbf{W}_{32}{}^T & \mathbf{W}_{33}{}^T & \mathbf{0} & \mathbf{0} & \mathbf{V}_3 & \mathbf{0} \\ \mathbf{W}_{41}{}^T & \mathbf{W}_{42}{}^T & \mathbf{W}_{43}{}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{V}_4 \end{pmatrix} \begin{pmatrix} \delta_{\mathbf{a}_1} \\ \delta_{\mathbf{a}_2} \\ \delta_{\mathbf{a}_3} \\ \delta_{\mathbf{b}_1} \\ \delta_{\mathbf{b}_2} \\ \delta_{\mathbf{b}_3} \\ \delta_{\mathbf{b}_4} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{a}_1} \\ \epsilon_{\mathbf{a}_2} \\ \epsilon_{\mathbf{a}_3} \\ \epsilon_{\mathbf{b}_1} \\ \epsilon_{\mathbf{b}_2} \\ \epsilon_{\mathbf{b}_3} \\ \epsilon_{\mathbf{b}_4} \end{pmatrix}. \qquad (16)$$

Denoting

$$\mathbf{U}^* = \begin{pmatrix} \mathbf{U}_1^* & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_3^* \end{pmatrix}, \mathbf{V}^* = \begin{pmatrix} \mathbf{V}_1^* & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{V}_3^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{V}_4^* \end{pmatrix}, \mathbf{W} = \begin{pmatrix} \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\ \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\ \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \end{pmatrix} \qquad (17)$$

where * designates the augmentation of diagonal elements, allows the augmented normal equations to be further compacted to

$$\begin{pmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_{\mathbf{a}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{a}} \\ \epsilon_{\mathbf{b}} \end{pmatrix}. \qquad (18)$$

Left multiplication of Eq. (18) by the block matrix

$$\begin{pmatrix} \mathbf{I} & -\mathbf{W}\,\mathbf{V}^{*-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \tag{19}$$

results in

$$\begin{pmatrix} \mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T & \mathbf{0} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_{\mathbf{a}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{a}} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_{\mathbf{b}} \\ \epsilon_{\mathbf{b}} \end{pmatrix}. \tag{20}$$

Noting that the top right block of the left hand matrix is zero, the dependence of the camera parameters on the structure parameters has been eliminated in Eq. (20), therefore $\delta_{\mathbf{a}}$ can be determined from its top half, which is

$$(\mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T)\,\delta_{\mathbf{a}} = \epsilon_{\mathbf{a}} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_{\mathbf{b}}. \tag{21}$$

Matrix $\mathbf{S} \equiv \mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T$ is the Schur complement of $\mathbf{V}^*$ in the left hand side matrix of Eq. (18). Since the Schur complement of a symmetric positive definite matrix is itself symmetric and positive definite [16], the system of Eq. (21) can be efficiently solved using the Cholesky factorization of $\mathbf{S}$. Having solved for $\delta_{\mathbf{a}}$, $\delta_{\mathbf{b}}$ can then be computed by back substitution into the bottom half of Eq. (20), which yields

$$\mathbf{V}^*\,\delta_{\mathbf{b}} = \epsilon_{\mathbf{b}} - \mathbf{W}^T\,\delta_{\mathbf{a}}. \tag{22}$$

The choice of solving first for $\delta_{\mathbf{a}}$ and then for $\delta_{\mathbf{b}}$ is justified by the fact that the total number of camera parameters is in general much less compared to the total number of structure parameters. Therefore, Eq. (21) involves the solution of smaller systems that can be carried out with considerably fewer computations. Following the computation of $\delta_{\mathbf{a}}$ from Eq.(21), left multiplication of Eq. (22) by $\mathbf{V}^{*-1}$ yields $\delta_{\mathbf{b}}$ as

$$\delta_{\mathbf{b}}^T = \left(\mathbf{V}_1^{*-1}\,(\epsilon_{\mathbf{b}_1} - \sum_{j=1}^{3} \mathbf{W}_{1j}{}^T\,\delta_{\mathbf{a}_j}),\ \ldots,\ \mathbf{V}_4^{*-1}\,(\epsilon_{\mathbf{b}_4} - \sum_{j=1}^{3} \mathbf{W}_{4j}{}^T\,\delta_{\mathbf{a}_j}) \right)^T. \tag{23}$$

At this point, it should be evident that the approach for solving the normal equations that was illustrated above can be directly generalized to arbitrary n and m. Note that if a point k does not appear in an image l then $\mathbf{A}_{kl} = \mathbf{0}$ and $\mathbf{B}_{kl} = \mathbf{0}$. Hence, index i in the summations appearing in the definitions of $\mathbf{U}_j$ and $\epsilon_{\mathbf{a}_j}$ (see Eqs. (11) and (14)) runs through all points appearing in the specified camera j. Similarly, index j in the definitions of $\mathbf{V}_i$ and $\epsilon_{\mathbf{b}_i}$ runs through all cameras to which the given point i is projected. The procedure for solving the sparse normal equations that was outlined above can be embedded in the LM algorithm of section 2 at the point indicated by the rectangular box in Fig. 1, leading to a sparse bundle adjustment algorithm. This algorithm has been implemented by the `sba` package, which relies on LAPACK for linear algebra calculations. A user's guide for `sba` can be found in [8].

4 Application of sba to Euclidean BA

As has already been mentioned, sba can facilitate the solution of a wide range of reconstruction-related vision problems. This section concerns the use of sba for Euclidean BA, a task which is a key ingredient for dealing with the problem of camera tracking. Camera tracking refers to using solely visual input for estimating the 3D position and orientation of a freely moving camera. The authors have routinely employed sba in this context for dealing with camera tracking problems involving a few thousands 3D points whose image projections depended on a few hundreds camera parameters. More specifically, it is assumed that a set of Euclidean 3D points are seen in a number of images acquired by an intrinsically calibrated moving camera. It is also assumed that the image projections of each Euclidean 3D point have been identified and that initial estimates of the 3D point structure and the Euclidean camera matrices have been obtained as described in [9]. The remainder of this section describes the application of sba for refining those camera matrix and structure estimates.

4.1 Parameterizing Euclidean BA

The employed world coordinate frame is taken to be aligned with the initial camera location. All subsequent camera motions are defined relative to the initial location, through the combination of a 3D rotation and a 3D translation. A 3D rotation by an angle θ about a unit vector $\mathbf{u} = (u_1, u_2, u_3)^T$ is represented by the quaternion $\mathbf{E} = (\cos(\frac{\theta}{2}),\ u_1 \sin(\frac{\theta}{2}),\ u_2 \sin(\frac{\theta}{2}),\ u_3 \sin(\frac{\theta}{2}))^T$ [6, 18]. A 3D translation is defined by a vector $\mathbf{t}$. A 3D point is represented by its Euclidean coordinate vector $\mathbf{X}$. Thus, the parameters of each camera j and point i are $\mathbf{a}_j = ({\mathbf{E}_j}^T, {\mathbf{t}_j}^T)^T$ and $\mathbf{b}_i = \mathbf{X}_i$, respectively. With the previous definitions, the predicted projection of point i on image j is

$$\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i) = \mathbf{K}\ (\mathbf{E}_j\ \mathbf{N}_i\ \mathbf{E}_j^{-1}\ +\ \mathbf{t}_j), \tag{24}$$

where $\mathbf{K}$ is the 3×3 intrinsic camera calibration matrix, $\mathbf{E}_j^{-1}$ is the inverse quaternion of $\mathbf{E}_j$ and $\mathbf{N}_i = (0,\ {\mathbf{X}_i}^T)^T$ is the vector quaternion corresponding to the 3D point $\mathbf{X}_i$. The expression $\mathbf{E}_j\ \mathbf{N}_i\ \mathbf{E}_j^{-1}$ corresponds to point $\mathbf{X}_i$ rotated by an angle θ_j about unit vector $\mathbf{u}_j$, as specified by the quaternion $\mathbf{E}_j$. Thus, each camera motion is parameterized by 7 unknowns while each 3D point by 3. The computation of the measurement vector's Jacobian relies on a routine for computing the Jacobian of function $\mathbf{Q}()$ in Eq. (24), whose code was generated automatically using MAPLE's symbolic differentiation facilities. We also note that the projection matrix of the first camera is kept constant during bundle adjustment, equal to $\mathbf{K}\ [\mathbf{I}_{3\times 3} \mid \mathbf{0}]$.

4.2 Experimental Results

Sample experimental results from a series of camera tracking experiments are presented next. 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. Initial estimates of the Euclidean 3D structure and camera motions are then computed using the sequential structure and motion estimation technique described in [9]. With the aid of `sba`, those estimates were then refined through Euclidean BA. 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". The first five are standard sequences, widely used as benchmarks in the reconstruction literature.

Table 1 illustrates several statistics gathered from the application of Euclidean sparse 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 reprojection error of the initial reconstruction. The fourth column shows the average reprojection error after BA. The fifth column shows the total number of objective function/Jacobian evaluations during BA. The number of iterations needed for convergence is shown in column six and the last column shows the time (in seconds) elapsed during execution of BA. In all cases, the sparse LM algorithm terminated due to the magnitude of the computed step δ being very small. All experiments were conducted on a 1.8 GHz Intel P4 running Linux, GCC 2.96 and unoptimized BLAS.

As it is evident from these results, `sba` has successfully solved all underlying minimization problems in little time. For comparison, we have also implemented BA using a dense, general purpose version of the LM algorithm[1]. Note that the dense Jacobians for the test problems are extremely large: For example, the "movi" sequence involves 5747 variables and 12415 image projections. Taking into account that each image projection is a 2D vector and that each double precision real number requires 8 bytes to be stored, the amount of RAM necessary to store the Jacobian in this case exceeds 1Gb and is thus beyond the capacity of most current desktop computers. Therefore, using dense BA, we have been able to solve only the problems associated with the shortest of the test sequences, namely those for "basement" and "house", for which the corresponding execution times were 481.54 and 1960.92 seconds, respectively. Compared to the fractions of a second that were spent by `sba` for solving those problems, these figures clearly demonstrate the enormous computational gains achieved by the sparse BA implementation.

5 Conclusions

This paper has presented the mathematical theory behind an LM-based sparse bundle adjustment algorithm. The practical outcome of this work is a generic

[1] Our free C/C++ implementation of a dense LM algorithm can be found at `http://www.ics.forth.gr/~lourakis/levmar`.

Sequence	imgs	vars	init. err.	fin. err.	func/jac	iter.	exec. time
"movi"	59	5747	5.03	0.3159	18/18	18	3.80
"sagalassos"	26	5309	11.04	1.2703	44/33	33	4.98
"arenberg"	22	4159	1.35	0.5399	25/20	20	3.22
"basement"	11	992	0.37	0.2447	29/21	21	0.29
"house"	10	1615	1.43	0.2142	25/19	19	0.41
"maquette"	54	15999	2.15	0.1765	31/23	23	9.01
"desk"	46	10588	4.16	1.5760	32/23	23	6.92
"calgrid"	27	2355	3.21	0.2297	20/20	20	9.70

Table 1. Execution statistics for the application of `sba` to Euclidean BA: Total number of images, total number of variables, average initial reprojection error in pixels, average final reprojection error in pixels, total number of objective function/Jacobian evaluations, total number of iterations, elapsed execution time in seconds. Identical values for the user-defined parameters have been used throughout all experiments.

sparse BA package called `sba` that has experimentally been demonstrated to be capable of dealing efficiently with very large BA problems. The package can be very useful to researchers working in vision and related fields, therefore it has been made freely available.

References

1. P. Beardsley, P.H.S. Torr, and A. Zisserman. 3D Model Acquisition From Extended Image Sequences. In *Proc. of ECCV'96*, pages 683–695, 1996.
2. D.C. Brown. A Solution to the General Problem of Multiple Station Analytical Stereo Triangulation. Technical Report 43, RCA-MTP, Feb. 1958.
3. J.E. Dennis and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. SIAM Publications, Philadelphia, 1996.
4. A.W. Fitzgibbon and A. Zisserman. Automatic Camera Recovery for Closed or Open Image Sequences. In *Proceedings of ECCV'98*, pages 311–326, 1998.
5. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 1st edition, 2000.
6. B.K.P. Horn. Closed-form Solution of Absolute Orientation Using Unit Quaternions. *Journal of the Optical Society of America, A*, 4(4):629–642, 1987.
7. C.T. Kelley. *Iterative Methods for Optimization*. SIAM Publications, Philadelphia, 1999.
8. M.I.A. Lourakis and A.A. Argyros. The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm. TR-340, ICS-FORTH, Heraklion, Greece, Aug. 2004. Available at `ftp://ftp.ics.forth.gr/tech-reports/2004`.
9. M.I.A. Lourakis and A.A. Argyros. Efficient, Causal Camera Tracking in Unprepared Environments. *Computer Vision and Image Understanding Journal*, 99(2):259–290, Aug. 2005.
10. K. Madsen, H.B. 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`.
11. J.J. Moré, B.S. Garbow, and K.E. Hillstrom. User guide for MINPACK-1. Technical Report ANL-80-74, Argonne National Laboratory, Aug. 1980.
12. J.J. Moré and D.C. Sorensen. Computing a Trust Region Step. *SIAM J. Sci. Statist. Comput.*, 4:553–572, 1983.
13. H.B. 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`.
14. J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, New York, 1999.
15. M. Pollefeys, L. Van 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.
16. V.V. Prasolov. *Problems and Theorems in Linear Algebra*. American Mathematical Society, Providence, RI, 1994.
17. B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In *Proc. of the Int'l Workshop on Vision Algorithms: Theory and Practice*, pages 298–372, 1999.
18. L. Vicci. Quaternions and Rotations in 3-Space: The Algebra and its Geometric Interpretation. Technical Report 01-014, Dept. of Computer Science, UNC, 2001.