Algebraic Line Search for Bundle Adjustment

Julien Michot[1]
julien.michot@cea.fr
Adrien Bartoli[2]
adrien.bartoli@gmail.com
François Gaspard[1]
francois.gaspard@cea.fr

[1] CEA LIST,
Embedded Vision Systems Laboratory,
Point Courrier 94, Gif-sur-Yvette,
F-91191 France
[2] LASMEA,
UMR 6602 CNRS/UBP,
Clermont-Ferrand, France

Abstract

Bundle Adjustment is based on nonlinear least squares minimization techniques, such as Levenberg-Marquardt and Gauss-Newton. It iteratively computes local parameter increments. Line Search techniques aim at providing an efficient magnitude for these increments, called the step length. In this paper, a new ad hoc Line Search technique for solving bundle adjustment is proposed. The main idea is to determine an efficient step length using an approximation of the cost function based on an algebraic distance. We use the Wolfe conditions to show that our Line Search preserves the convergence properties of the original algorithm. Our method is compared to different nonlinear optimization algorithms and Line Search techniques under several conditions, on real and synthetic data. The method improves the minimization process, decreasing the reprojection error significantly faster than the other techniques.

1 Introduction

The Computer Vision community has devoted a great interest to the SfM (Structure-from-Motion) problem for decades. Recovering the 3D scene structure and the camera motion is an important step in a wide range of applications, such as visual SLAM (Simultaneous Localization And Mapping, *e.g.* [3, 14, 26]). This technique recovers the structure of the scene and simultaneously localizes the camera in the generated map. Bundle adjustment is a nonlinear optimization process generally employed in SfM problems to refine an initial model. The optimization is performed over a set of parameters that represents the 3D structure (3D primitives) and the cameras. Classically, Gauss-Newton type minimization algorithms, such as Levenberg-Marquardt [9, 12], are used to minimize the reprojection error. This criterion is defined as the geometric distance between detected features and reprojected ones [25]. In recent years, strategies have been developed to make this nonlinear minimization process fast and effective. It is now even possible to use local bundle adjustment in real-time applications [14]. We propose a novel strategy to obtain a faster convergence to the minimum (less iterations) thanks to an effective Line Search technique.

One drawback of the LM (Levenberg-Marquardt) method is that it only calculates a direction in the parameter space, but does not necessarily provide the best magnitude, called the *step length*. The aim of Line Search techniques is to find an efficient displacement length for a given direction. Many Line Search methods have been proposed, such as [1, 4, 10, 13, 18]. These techniques are hardly ever used in SfM [8, 19] since they are iterative and usually increase the computational time compared to using a unit step length.

We propose a new Line Search technique that we call ALS (Algebraic Line Search). It can easily be plugged in existing bundle adjustment implementations (like the general framework described in [8]).

Thanks to an approximation of the reprojection error using an algebraic distance, we can calculate analytically an efficient step length for each parameter increment. Algebraic distances are typically used for initialization purposes. Since better results are obtained with a geometric reprojection error, we use an algebraic distance in one Line Search technique only and minimize the "true" geometric reprojection error. We present two variants of our method: Global and Two-way ALS. The former determines a single step length for all parameters, given by solving analytically a degree 3 polynomial. The latter variant defines two step lengths: one for the cameras and one for the 3D points. It requires one to solve a degree 5 polynomial. The proposed routines are low cost, efficient and can be used for visual SLAM problems, pose estimation and any problem requiring one to minimize the reprojection error. We have experimented ALS with Levenberg-Marquardt and Dogleg Trust Region algorithms, for calibrated and uncalibrated Batch SfM refinements. Results show that this dynamic selection of the step length, optimal for the algebraic reprojection error, provides a good step length for the geometric reprojection error.

Paper organization. Section 2 introduces the SfM problem and bundle adjustment. We describe our new approach, ALS (Algebraic Line Search) in section 3. Finally, the last section reports experimental results on both real and synthetic data. A supplementary material describes the resolution of our Two-way ALS.

Notation. Scalars are in italics (*e.g.* x), vectors in bold (*e.g.* $\mathbf{p}$), and matrices in sans-serif (*e.g.* M). I_r is the $r \times r$ identity matrix. $d(\mathbf{q},\mathbf{q}')$ is the Euclidean distance, nonlinear in homogeneous coordinates since $d^2(\mathbf{q},\mathbf{q}') = \|\Psi(\mathbf{q}) - \Psi(\mathbf{q}')\|^2$ with $\Psi(\mathbf{q}) = \frac{1}{q_3}\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$. Function *vect* creates a vector by stacking row-wise the elements of a matrix.

2 Background and Previous Work

We consider the problem of recovering the geometry of 3D scenes from multiple 2D images through SfM. The process is usually composed of an initial coarse reconstruction which is later refined with Bundle Adjustment.

2.1 Structure-from-Motion Initialization

The 3D reconstruction problem aims at recovering a model of all camera poses and all the m 3D points $\mathbf{Q}_{j=1..m}$ of a scene from multiple views. Camera projections are written as 3×4 matrices $\mathsf{P}_{i=1..n}$ and can be decomposed in some metric coordinate frame, as $\mathsf{P}_i = \mathsf{K}_i(\mathsf{R}_i|\mathbf{t}_i)$, where K_i encodes the intrinsic parameters and $(\mathsf{R}_i, \mathbf{t}_i)$ represents the orientation and position of the camera in a world coordinate frame.

There exist two main approaches to obtain an initiale estimate of the structure and motion: Incremental and Batch SfM. In this paper we use Batch SfM.

Batch Factorization SfM is based on an SVD (Singular Value Decomposition) of the measurement matrix [21, 23]. Initial factorization techniques ([21, 23]) were affected by missing data or outliers. Hierarchical [16] and Closure Constraints Factorizations [22, 24] are more robust. They consider relevant sub-blocks of the measurement matrix to recover the scene structure. These methods provide projective or affine reconstructions. To recover a metric shape, we either do self-calibration, and find intrinsic camera parameters K_i and the plane at infinity π_∞, or only find π_∞ if the K_i are already known. More details on the procedure can be found in [8].

2.2 Bundle Adjustment

Bundle Adjustment (BA) is based on nonlinear least squares minimization in order to refine the initial structure and camera motion. The parameter vector $\mathbf{x}$ to refine is composed in the uncalibrated case, of $\mathbf{Q}_j$ and P_i : $\mathbf{x}^\top = \left(\mathbf{p}_1^\top, \cdots, \mathbf{p}_n^\top, \mathbf{Q}_1^\top, \cdots, \mathbf{Q}_m^\top\right)$ with $\mathbf{p}_i = vect(\mathsf{P}_i)$. For calibrated reconstructions, we suppose that the intrinsic matrices $\mathsf{K}_i = \mathsf{K}$ are equal, constant and known. So $\mathbf{p}_i$ is replaced by $\mathbf{r}_i$ and $\mathbf{t}_i$, where $\mathsf{R}_i = R(\mathbf{r}_i)$ and $R(\theta)$ transforms the three angle vector θ to a rotation matrix. We can also consider self-calibrating bundle adjustement, that also estimate some of the intrinsics, such as the focal lengths f_i.

2.2.1 The reprojection Error

In SfM, the objective function to refine is generally is the reprojection error $\varepsilon(\mathbf{x})$. This function is the sum of the squares of the distance between 2D observations (measurements in the images) and reprojections:

$$\varepsilon(\mathbf{x}) = \sum_{i,j} v_{ij}\, d^2\left(\mathbf{q}_{ij}, \mathsf{P}_i\mathbf{Q}_j\right), \tag{1}$$

where $\mathbf{q}_{ij}$ is the observation of point $\mathbf{Q}_j$ in image (camera) P_i, and $v_{ij} = 1$ if the observation exists and 0 otherwise.The algebraic distance $\tilde{d}$ is defined by:

$$\tilde{d}(\mathbf{q}, \mathbf{q}') = \left\|\mathsf{S}[\mathbf{q}]_\times \mathbf{q}'\right\|, \tag{2}$$

with $\mathsf{S} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ and $[\mathbf{q}]_\times$ is the matrix representation of the vector cross product. It is agreed [7, 8] that under an appropriate normalization, cost functions based on this distance give satisfying results, even if this distance is not geometrically or statistically meaningful. The algebraic cost function is written as:

$$\tilde{\varepsilon}(\mathbf{x}) = \sum_{i,j} v_{ij} \left\|\mathsf{S}[\mathbf{q}_{ij}]_\times \mathsf{P}_i\mathbf{Q}_j\right\|^2. \tag{3}$$

2.2.2 Nonlinear Least Squares Optimization

We consider the least squares optimization:

$$\min_{\mathbf{x}\in\mathbb{R}^p} \nu(\mathbf{x}), \quad \text{where} \quad \nu(\mathbf{x}) = \frac{1}{2}\|\mathbf{D}(\mathbf{x})\|^2 \quad \text{and} \quad \mathbf{D}(\mathbf{x}) = \begin{pmatrix} \cdots \\ \mathbf{q}_{ij} - \Psi(\mathsf{P}_i\mathbf{Q}_j) \\ \cdots \end{pmatrix} \tag{4}$$

$\mathbf{D}$ stacks the reprojection residuals of every observation $\mathbf{q}_{ij}$ (where $v_{ij} = 1$) in a $1 \times l$ vector ($l < nm$), $\nu : \mathbb{R}^p \rightarrow \mathbb{R}$ is a nonconvex and twice continuously differentiable function. We try to find an optimum $\mathbf{x}^*$ of ν for which:

$$\nabla \nu(\mathbf{x}^*) = 0. \tag{5}$$

The problem is usually solved by an iterative minimization algorithm which generates a series of displacements $\mathbf{x}_0, \mathbf{x}_1, \ldots \mathbf{x}_k$. Under certain conditions, which depend on the algorithm, the series converges to a stationary point (local or global minimizer) $\mathbf{x}^*$: $\mathbf{x}_{k \rightarrow \infty} \rightarrow \mathbf{x}^*$. The sequence of $\mathbf{x}_k$ is commonly determined by:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \delta_k, \tag{6}$$

where $\delta_k \in \mathbb{R}^p$ is the parameter displacement given by the constrained or unconstrained optimization, and $\alpha_k \in \mathbb{R}^{+*}$ the step length. Many strategies have been proposed to compute the δ_k, including Gradient Descent, (quasi-)Newton methods [10], Gauss-Newton, Levenberg-Marquardt [8, 9, 15], Trust-Region such as Powell's dogleg [11, 18, 20].

2.3 Line Search

Most of the above mentioned algorithms do not guarantee to provide the best step length in the chosen direction δ_k (in the parameter space). Line Search techniques try to determine an efficient step length along this direction. *Exact Line Search* optimizes this criterion:

$$\alpha_k = \arg \min_{\alpha_k > 0} \varepsilon(\mathbf{x}_k + \alpha_k \delta_k). \tag{7}$$

Unfortunately, common problems have a nonlinear cost function. Equation (7) is thus not easy to solve. *Inexact Line Search* is more interesting in this case. Most of the *Inexact LS* techniques start with an initial step length and refine it iteratively such that it verifies the Wolfe or Goldstein conditions [17].

2.3.1 The Wolfe conditions

The Wolfe conditions are usually employed together with iterative displacements to ensure the convergence to a minimum. Since along a descent displacement δ_k, multiple valid step lengths exist, the Wolfe conditions permit to select those that decrease sufficiently the cost function. The first Wolfe condition, also known as Armijo criterion, guaranties that the current step provides a sufficient decrease of the cost function:

$$\nu(\mathbf{x}_{k+1}) \leq \nu(\mathbf{x}_k) + \omega_1 \alpha_k \mathbf{g}_k^\top \delta_k \tag{8}$$

where $\mathbf{g}_k = \mathsf{J}_k^\top \mathbf{D}(\mathbf{x}_k)$ with jacobian $\mathsf{J}_k = \frac{\partial \mathbf{D}}{\partial \mathbf{x}_k}(\mathbf{x}_k)$ and $0 < \omega_1 < 1$. This condition constrains huge descent displacements to decrease even more the cost function, and thus avoid to choose long steps which provide a limited decrease (adjustable with ω_1).

The second Wolfe condition (known as the curvature condition) coerces the LS algorithm to choose at each iteration, a step which minimizes the curvature of the function:

$$\left| \mathbf{g}_{k+1}^\top \delta_k \right| \leq -\omega_2 \mathbf{g}_k^\top \delta_k \tag{9}$$

with $0 < \omega_1 < \omega_2 < 1$. These two user-specified parameters characterize the enhancement that each step length guess has to realize to be selected.

Zoutendijk theorem [17] defines that under the Wolfe conditions, Gauss-Newton techniques are guaranteed to converge (to a stationary point), since $\lim_{k\to\infty} \|\nabla \nu(\mathbf{x}_k)\| = 0$. One can also prove the convergence of Levenberg-Marquardt under special restrictions on the initial *damping parameter*. However, these restrictions are not generally used since better performances (speed) appear with user defined *damping* values.

2.3.2 Backtracking Based Line Search Techniques

The Backtracking algorithm is one of the most employed Line Search strategies. Initialized on a first step length, it generates and evaluates iteratively a series of step length guesses and returns the one which has the lowest error.

A number of Line Search algorithms, such as [1, 4, 13, 18], derive from the backtracking Line Search technique. By reducing an interval, they try to find the best step length thanks to different coefficient update rules. Frandsen *et al.* [4] propose two iterative Line Search techniques (Exact and Soft LS) with quadratic interpolation of the cost function. These methods are effective but very slow, even the soft ones. Liu and Nocedal [10] describe a Line Search based on the Wolfe conditions in a Limited quasi-Newton algorithm (L-BFGS). Again, Hager and Zhang in [5] study the convergence of a conjugate gradient method with a Line Search based on Wolfe conditions. An overview on classical Line Search techniques employed in nonlinear optimization is given in [17, chap 3]. To summarize, all these Line Search methods are recursive or iterative. They are time consuming since they evaluate the error function at each iteration.

We describe our contribution in the following section: a low cost Line Search technique which can be used to decrease computation time when minimizing a reprojection based cost function.

3 Algebraic Line Search

Our idea is **not** to use the algebraic distance as the cost function of a bundle adjustment: we keep the basic geometric distance for this. Our proposition is however to use the algebraic distance to find an efficient step length, in a Line Search manner. We call this technique ALS (Algebraic Line Search). We distinguish calibrated and uncalibrated bundle adjustment. The main difference is the camera projection parametrization (either the position and orientation of the cameras or elements of matrices P_i). A general SfM algorithm for the calibrated and uncalibrated cases with ALS is given in Table 1.

Algebraic Line Search , expressed within the framework of Hartley and Zisserman's textbook [8, p.574] *Add the following substeps to step vii:*
$vii^{(1)}$. Compute the coefficients of the Global ALS (equation (10)) or Two-way ALS (defined in the Supplementary material) equation
$vii^{(2)}$. Solve the equation, check the Wolfe conditions on each solution and keep the best one (including the basic unit step length)
$vii^{(3)}$. Multiply each incremental displacement subset $\delta = (\delta_a^\top, \delta_b^\top)$ by the selected step length

Table 1: Implementing our Algebraic Line Search within the Bundle Adjustment Levenberg-Marquardt-based framework given in [8, p.574] (Algorithm A4.1).

3.1 Uncalibrated Algebraic Line Search

We investigate two different approaches for ALS. The Global ALS is a Line Search technique that aims at finding a global efficient step length for the whole parameter set (camera and scene structure). In a different way, the Two-way ALS determines two distinct step lengths, one for the cameras and the other for the scene structure.

3.1.1 Global Algebraic Line Search (G-ALS)

Once the optimization provides us with a step direction $\delta^\top = (\delta_{P_1}^\top, \cdots, \delta_{P_n}^\top, \delta_{Q_1}^\top, \cdots, \delta_{Q_m}^\top)$, with $\delta_{P_i} = vect(\Delta_{P_i})$, we want to determine the step length α. Considering the update (6), the algebraic reprojection error (3) becomes a function of α:

$$\tilde{\varepsilon}(\mathbf{x} + \alpha\delta) = \sum_{i,j} v_{ij} \left\| \mathsf{S}[\mathbf{q}_{ij}]_\times (\mathsf{P}_i + \alpha\Delta_{P_i})(\mathbf{Q}_j + \alpha\delta_{Q_j}) \right\|^2. \quad (10)$$

Since we search for the best step length α^* that minimizes $\tilde{\varepsilon}(\mathbf{x})$ in the previously computed direction δ, the optimums are the real positive solutions of $\frac{\partial\tilde{\varepsilon}}{\partial\alpha} = 0$, given by inspecting the roots of the polynomial:

$$\frac{\partial\tilde{\varepsilon}(\mathbf{x} + \alpha\delta)}{\partial\alpha} = a\alpha^3 + b\alpha^2 + c\alpha + d \quad (11)$$

with

$$\begin{aligned}
a &= \textstyle\sum_{i,j} v_{ij}\, 2(\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \Delta_{P_i}\delta_{Q_j})^\top (\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \Delta_{P_i}\delta_{Q_j}) \\
b &= \textstyle\sum_{i,j} v_{ij}\, 3\left(\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times (\Delta_{P_i}\mathbf{Q}_j + \mathsf{P}_i\delta_{Q_j})\right)^\top \mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \Delta_{P_i}\delta_{Q_j} \\
c &= \textstyle\sum_{i,j} v_{ij} \left(\left(\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \mathsf{P}_i\mathbf{Q}_j\right)^\top \mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \Delta_{P_i}\delta_{Q_j} + (\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times (\Delta_{P_i}\mathbf{Q}_j + \mathsf{P}_i\delta_{Q_j}))^2\right) \\
d &= \textstyle\sum_{i,j} v_{ij} \left(\mathsf{S}\left[\mathbf{q}_{ij}\right]_\times \mathsf{P}_i\mathbf{Q}_j\right)^\top \mathsf{S}\left[\mathbf{q}_{ij}\right]_\times (\Delta_{P_i}\mathbf{Q}_j + \mathsf{P}_i\delta_{Q_j}).
\end{aligned}$$

The algebraic reprojection error simplifies the problem. We can now calculate the optimal step length in closed-form. Equation (11) has three solutions and in most of our experiments, only one solution was real. This Line-Search technique - optimal for the algebraic cost function - is simple, fast and well approximates the geometric cost function. Most time calculation of this method lies in the computation of the coefficients of equation (11). However, these coefficients are only composed of sums and multiplications and unlike classical Line Search methods, no iterations are necessary.

3.1.2 Two-way Algebraic Line Search (T-ALS)

Since there are two different types of parameters to refine in common bundle adjustment (the scene structure and the cameras), it sounds attractive to dissociate the step length for each kind of parameters since they do not share the same units. We propose the Two-way ALS that finds two step lengths α_P^* and α_Q^*, respectively for camera and scene structure displacements:

$$\min_{\alpha_P, \alpha_Q} \sum_{i,j} v_{ij} \left\| \mathsf{S}\left[\mathbf{q}_{ij}\right]_\times (\mathsf{P}_i + \alpha_P\Delta_{P_i}) \left(\mathbf{Q}_j + \alpha_Q\delta_{Q_j}\right) \right\|^2. \quad (12)$$

The search for the global minimum (α_P^*, α_Q^*) can be performed in a fast way. Nullifying the partial derivatives of equation (12) with respect to α_P and α_Q gives a system of two polynomials in α_P and α_Q. This system can be solved using Gröbner basis with the Buchberger or

Faugère algorithm, so that α_P^* is the root of a degree 5 polynomial. α_Q^* is then deduced from α_P^*. Details on the calculation are given as supplementary material.

It is important to note that `T-ALS` alters the direction of the displacement. This is not a pure Line Search technique since it splits the displacement vector, for the cameras and the structure, and searches the best algebraic step length for each one. Nevertheless, experiments show that this modification often tends to improve the optimization speed, even sometimes allowing the minimization engine to escape from local minima.

3.2 Calibrated Algebraic Line Search

In a metric context, the camera parameters are not defined by the whole matrix P_i but only by orientation and translation parameters: $\delta^\top = \left(\delta_{r_1}^\top, \delta_{t_1}^\top, \cdots, \delta_{r_n}^\top, \delta_{t_n}^\top, \delta_{Q_1}^\top, \cdots, \delta_{Q_m}^\top\right)$. We use a local Euler angle parametrization for orientation displacements δ_{r_i}. Global camera orientations are represented by rotation matrices $\mathsf{R}_i \in SO(3)$. Our update rule for the iteration $k+1$ is thus $\mathsf{R}_i^{k+1} = \mathsf{R}_i^k R(\delta_{r_i}^k)$. In this context, and to preserve the form of equation (10), we approximate $\Delta_{P_i}^k$ by:

$$\Delta_{P_i}^k \approx \mathsf{K}\mathsf{R}_i^k([\delta_{r_i}^k]_\times | \delta_{t_i}^k). \tag{13}$$

Indeed, since

$$\mathsf{P}_i^{k+1} \quad = \quad \mathsf{P}_i^k \begin{pmatrix} R(\alpha\delta_{r_i}^k) & \alpha\delta_{t_i}^k \\ \mathbf{0} & 1 \end{pmatrix} \quad = \quad \mathsf{K}(\mathsf{R}_i^k R(\alpha\delta_{r_i}^k) | \mathbf{t}_i^k + \alpha\mathsf{R}_i^k\delta_{t_i}^k).$$

Using $R(\theta) \approx \mathrm{I}_3 + [\theta]_\times$ to approximate a local rotation, we get:

$$\mathsf{P}_i^{k+1} \approx \mathsf{K}(\mathsf{R}_i^k + \alpha\mathsf{R}_i^k[\delta_{r_i}^k]_\times | \mathbf{t}_i^k + \alpha\mathsf{R}_i^k\delta_{t_i}^k) = \mathsf{P}_i^k + \alpha\mathsf{K}\mathsf{R}_i^k([\delta_{r_i}^k]_\times | \delta_{t_i}^k).$$

This approximation of $\Delta_{P_i}^k$ allows us to preserve the form of equations (10) and (12) in the calibrated case. We apply the same procedures to find the step length(s) as in the uncalibrated adjustment, for Global and Two-way ALS.

3.3 Step Length Restriction

One drawback of the algebraic distance is that this error is only an approximation of the Euclidean distance since:

$$d^2(\mathbf{X}, \mathbf{X}') = \frac{\tilde{d}^2(\mathbf{X}, \mathbf{X}')}{ww'}, \tag{14}$$

with $\mathbf{X}^\top = (x, y, w)$ and $\mathbf{X}'^\top = (x', y', w')$, two homogeneous image points. Consequently, the minima of ε and $\tilde{\varepsilon}$ may not match. They are usually close under an appropriate normalization [2]. This implies some restrictions on the step length given by the ALS: Since ALS is a Line Search technique (or LS-like for `G-ALS`), we employ the Wolfe conditions [17, chap 3] to ensure the decrease of the norm of the gradient: $\lim_{k\to\infty} \|\nabla v(\mathbf{x}_k)\| = 0$, with $\omega_1 = 10^{-4}$ and $\omega_2 = 0.99$. In the case that ALS provides several solutions (this is rare in practice) we do an ascending sort of the step lengths. We then select the best step length which verifies the Wolfe criteria. Besides, to avoid mistaken step lengths (since the Algebraic distance is an approximation), we check whether a basic unit step length is not better than the one provided by ALS. In this special case, we obtain the same displacement (and length) that the one proposed by the minimization algorithm, *i.e.* without Line Search.

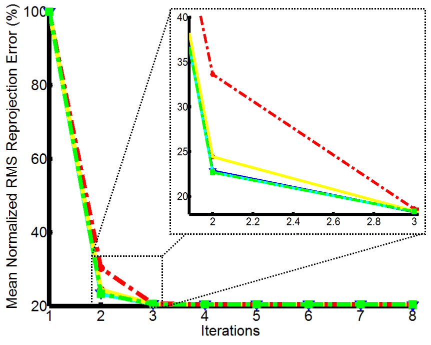
We employ ALS with two different minimization techniques: Levenberg-Marquardt [8] and Dogleg Trust Region [11]. It is important to note that when the Gauss-Newton part is preponderant, which means that we are near a minimum, it is useless to perform a Line Search since Gauss-Newton yet defines a displacement with a relatively appropriate step length (depending on the approximation of the Hessian). This means that for LM minimization, we only perform ALS in the first few iterations (5 in our experiments). We also present an ALS employed in the dogleg step of a Dogleg Trust Region minimization.

4 Experimental Results

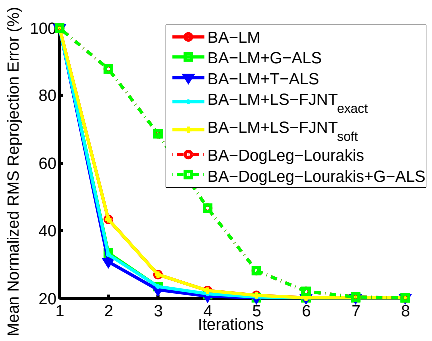
Experimental calculations were achieved in MATLAB. An isotropic normalization [6] was performed on all images points, and later rectified to get an output in pixels. We define RMS to be the Root Mean Square of the reprojection error. For each dataset, we compare our Line Search methods (`G-ALS` and `T-ALS` from section 3) with two nonlinear optimization algorithms (`BA-LM` [8], `BA-DogLeg-Lourakis` [11]) and state of the art Line Search techniques (`LS-FJNT-exact` and `LS-FJNT-soft`[1]).

4.1 Synthetic Data

We randomly generate 1000 3D points in a cube with side length 6m, and 30 camera poses positioned around the cube at a distance of 20m from the center. We assume that the camera has constant intrinsic parameters (focal length: 1000 pixels, square pixels, and principal point in the center of the $(640 \times 480\ pixels)$ images). Image points are corrupted by a gaussian noise ($\sigma = 1$ pixel). A batch reconstruction is next performed (see section 2.1) to obtain an initial vector of parameters $\mathbf{x}_0$, used by bundle adjustment algorithms.



(a)



(b)

Figure 1: Evolution of the mean normalized (by the initial RMS) RMS of 20 simulations, for calibrated 1(a) and uncalibrated 1(b) Bundle Adjustment.

In the calibrated tests (Fig.1(a)), we see that yet in the first iteration, our Algebraic Line Search decreases the reprojection error better than without Line Search, for an LM-based BA or with a Dog-Leg BA. Both methods are even better than soft `LS-FJNT` and are way faster (see experiments on real data). For uncalibrated BA (Fig.1(b)), `T-ALS` appears to be more

[1] www2.imm.dtu.dk/~hbn/Software/linesearch.m

efficient (in the first few iterations) than the classical exact `LS-FJNT`. `G-ALS` with LM-based BA has good results: it even matches exact `LS-FJNT`, but does not improve Dog-Leg BA in this configuration.

4.2 Real Data

We have experimented our ALS technique on several standard datasets[2] ("dinosaur", "castle") and video sequences ("city", see Figure 2(a)). An initial computation of camera poses and 3D scene structure was performed by the batch projective factorization of [22] and rectified to a metric scene (section 2.1).

Sequence	#Imgs (n)	#Vars ($6n+3m$)	Init. RMS (pixels)	Final RMS (pixels) no LS	Final RMS G-ALS	Final RMS T-ALS	Exec. Time (s) / Iterations no LS	Exec. Time G-ALS	Exec. Time T-ALS
"dinosaur"	12	4035	6.345	1.534	1.534	1.534	19.058/8	15.872/6	14.750/6
"castle"	9	3960	5.364	3.121	3.121	3.121	138.02/25	130.15/24	129.83/24
"city-long"	30	17136	2.642	0.668	0.668	0.668	39.93/8	34.98/7	35.17/7
"city-small"	6	2466	0.456	0.361	0.361	0.361	8.83/7	10.13/7	9.84/7

Table 2: Statistics for Euclidean BA using LM [8].

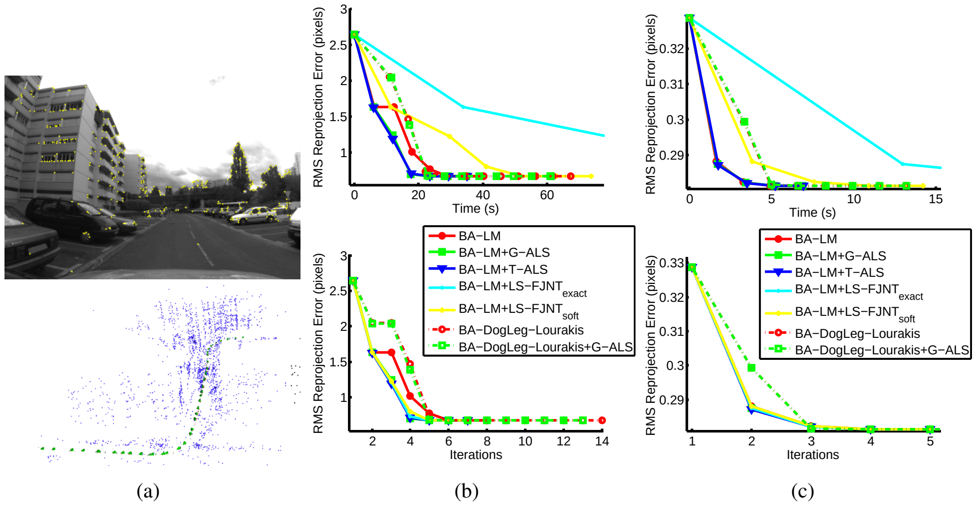


Figure 2: Results of the "city" sequence (a). Figures (b) and (c) show the evolution of the RMS (pixels) over time (top) and iterations (bottom), for 30 cameras (b) "city-long" and 6 cameras (c), "city-small".

When the initial solution is imprecise (a few pixels: sequences "dinosaur", "castle" and "city-long") BA with ALS is between 6 to 20% faster than classical BA using LM, depending on the sequence used as benchmark. This computation time decrease can be explained by comparing the number of iterations. BA with ALS often does less iterations to obtain the same solution, except for the last line (Figure 2(c)), which is a small sequence, and where the initial solution is already a good guess since the initial RMS is less than 0.5 pixels. This result agrees with the remark made in section 3.3, regarding the efficiency of a LS technique (like ALS) employed when the Gauss-Newton step well approximates the reprojection function.

[2] We thank VGG for their datasets www.robots.ox.ac.uk/∼vgg/data/data-mview.html

Figure 2 illustrates the results for BA over the "city-long" and "city-small" sequences. In figure 2(b), at the third iteration, we see that BAs using LM+LS escape from a local displacement fail. However our methods (`G-ALS` and `T-ALS`) are considerably faster than exact or even soft `LS-FJNT`.

5 Conclusion

This paper introduces ALS (Algebraic Line Search), a new line search technique for Bundle Adjustment. The idea is to substitute the geometric distance by the algebraic distance only in the Line Search cost function. Experiments conducted on simulated and real data showed that our ALS provides an efficient step length that minimizes the reprojection error in a shorter time than with other minimization techniques. The method can be applied to any problem minimizing a geometric error, such as camera pose estimation and point triangulation. Results showed that the further away the initial solution is from the optimal one, the greater the improvement provided by ALS (typically for $RMS > 1$ pixel). Future work will consist in experimenting this approach in a real-time setup, within a hierarchical adjustment for SLAM applications.

References

[1] M. Al-Baali and R. Fletcher. An efficient line search for nonlinear least squares. *J. Optim. Theory Appl.*, 48(3):359–377, 1986.

[2] W. Chojnacki and M.J. Brooks. Revisiting hartley's normalized eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1172–1177, Sept. 2003. ISSN 0162-8828. doi: 10.1109/TPAMI.2003.1227992.

[3] A. Davison. Real-time simultaneous localisation and mapping with a single camera. In *ICCV*, pages 1403–1410 vol.2, Washington, DC, USA, October 2003. IEEE Computer Society. ISBN 0-7695-1950-4.

[4] P.E. Frandsen, K. Jonasson, H.B. Nielsen, and O. Tingleff. *Unconstrained Optimization*. 1999.

[5] W. Hager and H. Zhang. Algorithm 851: CG DESCENT, a conjugate gradient method with guaranteed descent. *ACM Trans. Math. Softw.*, 32(1):113–137, 2006. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/1132973.1132979.

[6] R. Hartley. In defence of the 8-point algorithm. In *ICCV*, page 1064, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7042-8.

[7] R. Hartley. Minimizing algebraic error. In *ICCV*, page 469, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 81-7319-221-9.

[8] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge University Press, Cambridge, UK, first edition, 2003.

[9] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, Jul. 1944.

[10] D.C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Math. Program.*, 45(3):503–528, 1989. ISSN 0025-5610. doi: http://dx.doi.org/10.1007/BF01589116.

[11] M. Lourakis and A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? *ICCV*, 2005.

[12] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431, 441, 1963.

[13] J. Moré and D. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Trans. Math. Softw.*, 20(3):286–307, 1994. ISSN 0098-3500. doi: http://doi.acm.org/10.1145/192115.192132.

[14] E. Mouragnon, M. Lhuiller, M. Dhome, F. Dekeyser, and P. Sayd. Generic and real-time structure from motion. In *BMVC*, 2007.

[15] H.B. Nielsen. Damping parameter in marquardt's method. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, apr 1999.

[16] D. Nistér. Reconstruction from uncalibrated sequences with a hierarchy of trifocal tensors. In *ECCV*, volume 1, pages 649–663, 2000.

[17] J. Nocedal and S.J. Wright. *Numerical Optimization.* Springer Series in Operations Research, New york, 1999.

[18] J. Nocedal and Y. Yuan. Combining trust region and line search techniques. Technical report, Advances in Nonlinear Programming, (Kluwer), 1992.

[19] 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, 2004. ISSN 0920-5691. doi: http://dx.doi.org/10.1023/B:VISI.0000025798.50602.3a.

[20] M.J.D. Powell. A hybrid method for nonlinear equations. *Numerical Methods for Nonlinear Algebraic Equations, P. Rabinowitz (Ed.)*, pages 87–114, 1970.

[21] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In B. Buxton and Roberto Cipolla, editors, *ECCV, Cambridge, England*, volume 1065 of *Lecture Notes in Computer Science*, pages 709–720. Sprin-ger-Ver-lag, April 1996.

[22] J-P. Tardif, A. Bartoli, M. Trudeau, N. Guilbert, and S. Roy. Algorithms for batch matrix factorization with application to structure-from-motion. In *CVPR. IEEE Conference on*, pages 1–8, 2007.

[23] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *IJCV*, 9(2):137–154, 1992. ISSN 0920-5691. doi: http://dx.doi.org/10.1007/BF00129684.

[24] B. Triggs. Linear projective reconstruction from matching tensors. *ICV*, 15:617–625, 1997.

[25] B. Triggs, P.F. Mclauchlan, R. Hartley, and A.W. Fitzgibbon. Bundle adjustment – a modern synthesis. *Lecture Notes in Computer Science*, 1883:298+, January 2000.

[26] B. Williams, G. Klein, and I. Reid. Real-time SLAM relocalisation. *ICCV*, 2007.