ICPR'02 - IN PROCEEDINGS OF THE SIXTEENTH IAPR INTERNATIONAL CONFERENCE ON PATTERN RECOGNITION,
QUEBEC CITY, CANADA, PP. 560-563, VOL. II, AUGUST 2002.

A Unified Framework for Quasi-Linear Bundle Adjustment

Adrien Bartoli, *Adrien.Bartoli@inria.fr*
INRIA Rhône-Alpes, 655, av. de l'Europe, 38334 St. Ismier cedex, France.

Abstract

Obtaining 3D *models from long image sequences is a major issue in computer vision. One of the main tools used to obtain accurate structure and motion estimates is bundle adjustment. Bundle adjustment is usually performed using non-linear Newton-type optimizers such as Levenberg-Marquardt which might be quite slow when handling a large number of points or views.*

We investigate an algorithm for bundle adjustment based on quasi-linear optimization. The method is straightforward to implement and relies on solving weighted linear systems obtained as simple functions of the input data. Important features are that (i) *the original cost function is preserved,* (ii) *different projection models, either calibrated or not, are handled in a unified framework and* (iii) *any number of views and points as well as missing data can be handled.*

Experimental results on simulated and real data show that the algorithm is as accurate as standard techniques while requiring less computational time to converge.

1. Introduction

Recovering metric structure and motion from uncalibrated images is a central problem for computer vision. Most of the time, a sub-optimal solution is obtained in projective space for motion, then for structure [1], or jointly [6]. Self-calibration is then performed to upgrade the reconstruction to metric space.

In this paper, we investigate a very simple technique for bundle adjustment. Experimental results reveal that it is as accurate as traditional techniques while requiring less CPU time. We consider both the full perspective and the affine projection, either calibrated or uncalibrated. We may end up therefore with projective, affine or metric reconstructions.

Let us consider point features. Bundle adjustment basically consists in finding the minimum of a cost function defined as the sum of discrepancies between actual and predicted image points. These discrepancies are measured using the Euclidean distance. The optimization is performed over a set of parameters that represents structure and motion using most of the time non-linear Newton-type optimization techniques such as Levenberg-Marquardt [7]. This technique has the following drawbacks. Firstly, handling the free scale of homogeneous entities such as vectors or matrices representing points or cameras may be non-trivial. Secondly, it requires the computation of at least the first order Jacobian matrix of the residuals with respect to structure and motion parameters which might be non-trivial. Thirdly, the computational cost may be non-negligible since the Hessian matrix has to be inverted, even if specific techniques have been proposed to speed up the process [7].

In this paper, we adapt the resection-intersection technique [2, 5, 7] to perform bundle adjustment using quasi-linear optimizations. Our contribution differs from previous ones in that (i) we use the original cost function of bundle adjustment, which preserves optimality and (ii) we handle a great variety of camera models in a unified manner. In particular, we deal with calibrated configurations where non-linear constraints hold on the recovered motion. The final algorithm is simplicity itself. The method relies on rewriting the non-linear Euclidean distance used in bundle adjustment as a weighted bilinear algebraic distance [4]. In practice, it consists in iteratively solving weighted least squares systems that are simple functions of image point positions. Experimental results reveal that it performs as well as standard techniques in terms of convergence accuracy while greatly reducing computational cost.

This paper is organized as follows. In §2, we give our notation and preliminaries. §3 introduces the principle and the algorithm for quasi-linear bundle adjustment. These results are independent of the camera model. The approach is specialized in §§4.1 and 4.2 to uncalibrated and calibrated perspective cameras respectively. Finally, §5 validates the approach on simulated and real data and §6 gives our conclusions and perspectives.

2. Notation and preliminaries.

Vectors are typeset using bold letters $(\mathbf{q}, \mathbf{Q})$, matrices using sans-serif fonts (P, S) and scalars in italics (x, w). We make no formal distinction between coordinate vectors and physical entities. Everything is expressed in homogeneous coordinates, e.g. $\mathbf{q}^\mathsf{T} \sim (q_1 \; q_2 \; q_3)$ is an image point, where $\sim$ means equality up to scale and $^\mathsf{T}$ is vector/matrix transpo-

sition.

The standard cost function for bundle adjustment is defined by $\mathcal{C} = \sum_i \sum_j d^2(\mathbf{q}_{ij}, \widehat{\mathbf{q}}_{ij})$, where d^2 is the squared Euclidean distance, $\widehat{\mathbf{q}}_{ij}$ the reprojection of the i-th estimated 3D point in the j-th image and $\mathbf{q}_{ij}$ the corresponding image point. We drop the indices i and j for clarity. Bundle adjustment can be formulated as $\min_{\mathsf{P},\mathbf{Q}} \mathcal{C}$, where P and $\mathbf{Q}$ respectively designate the estimates of the parameters of motion and structure.

Distances in the image. The Euclidean distance is employed for bundle adjustment and in many criteria since it is physically meaninful. Most of the time, its non-linearity induces that of the corresponding criterion which therefore can not be used to linearly estimate e.g. structure or motion. On the other hand, consider the squared *algebraic distance* d_a defined by [4]:

$$d^2(\mathbf{x}, \mathbf{y}) = w^2_{\mathbf{x},\mathbf{y}} d^2_a(\mathbf{x}, \mathbf{y}) \text{ where } w_{\mathbf{x},\mathbf{y}} = (x_3 y_3)^{-2}. \quad (1)$$

The non-linearity is hidden in the weight factors $w_{\mathbf{x},\mathbf{y}}$ which can be thought of as a bias of the algebraic distance with respect to the Euclidean one. A convenient expression may be derived as follows:

$$d^2_a = ||\mathsf{S}[\mathbf{x}]_\times \mathbf{y}||^2 \text{ and } \mathsf{S} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad (2)$$

where $||.||^2$ is the $\mathcal{L}_2$ or squared Frobenius norm for vectors or matrices respectively and $[.]_\times$ the skew-symmetric 3×3-matrix associated with the cross product, i.e. $[\mathbf{v}]_\times \mathbf{q} = \mathbf{v} \times \mathbf{q}$. The algebraic distance does not have any physical meaning since it depends upon the relative scale of the homogeneous representation of $\mathbf{x}$ and $\mathbf{y}$. Its advantage is that it is bilinear. Since in most estimation problems one of the points compared is known, d_a becomes linear for the other.

Quasi-linear optimization. The relationship (1) between the Euclidean and the algebraic distance is the basis for quasi-linear optimization. The principle is to initialize weight factors to unity and compute a biased estimate using the weighted algebraic distance. When such an estimate has been obtained, weight factors can be computed using (1). The process is then iterated until convergence. Typically, 3 or 4 iterations are enough.

3. Quasi-Linear Bundle Adjustment

We derive the bundle adjustment algorithm, which is valid for most camera models. We then detail camera-dependent elements.

Consider the initial bundle adjustment problem which involves optimizing simultaneaously over the motion and structure parameters. It can be split, as indicated in [2, 5, 8] and becomes $\min_{\mathsf{P}} (\min_{\mathbf{Q}} \mathcal{C})$. This is strictly equivalent to the initial problem in that the same minima are kept. We split accordingly the optimization process into two distinct steps, for motion on the one hand and for structure on the other hand. These optimization steps can be alternatively performed to refine structure and motion. This scheme is termed resection-intersection. Each step can then be conducted independently using non-linear optimizers such as Levenberg-Marquardt, as proposed in [8]. The advantages of doing so are numerous:

- it reduces the complexity, since the optimization consists of $(m-1)$ independent outer steps[1] over an 11-dimentional space and n independent 3-dimensional inner steps (m is the number of views of n is the number of points);
- it performs as well as directly optimizing over all the parameters in terms of convergence accuracy, as reported by [2, 5, 8] and confirmed by our results;
- each of the two alternated optimization steps can be performed quasi-linearly, see the next section.

The algorithm, given below, needs an initialization of either structure or motion whereas standard bundle adjusters require both:

1. *initialize weight factors* to unity: $w_{\mathbf{q},\widehat{\mathbf{q}}} = 1$. These are not necessary for the affine camera model;
2. quasi-linearly *solve for structure*;
3. quasi-linearly *solve for motion*;
4. *iterate* steps 2 and 3 until convergence (see below);

The estimation of weight factors, which corrects the bias of the algebraic distance with respect to the Euclidean one, is done inside steps 2 and 3.

Convergence. Convergence is achieved when steps 2 and 3 become idempotent for structure and motion, i.e. when the difference between two consecutive values of the cost function $\mathcal{C}$ is lower than a threshold, typically $10e^{-8}$ pixel, or when the residual error begins to increase.

4. Resection-Intersection

4.1. Uncalibrated Pin-Hole Cameras

We consider uncalibrated full perspective projection to model cameras. This yields projective reconstructions. We derive the equations that lead to structure, then motion, quasi-linear estimation. Consider one term $d^2(\mathbf{q}, \widehat{\mathbf{q}})$ of the double sum of the cost function $\mathcal{C}$. Using the link (1) between the Euclidean and the algebraic distances d and d_a, then the expression (2) of the algebraic distance, each of these terms can be written $d^2(\mathbf{q}, \widehat{\mathbf{q}}) = w^2_{\mathbf{q},\widehat{\mathbf{q}}} ||\mathsf{S}[\mathbf{q}]_\times \widehat{\mathbf{q}}||^2$. Using perspective projection $\widehat{\mathbf{q}} \sim \mathsf{P}\mathbf{Q}$ and some minor algebraic manipulations:

$$d^2(\mathbf{q}, \widehat{\mathbf{q}}) = w^2_{\mathbf{q},\widehat{\mathbf{q}}} \left|\left| \begin{pmatrix} \mathbf{p}_2^\mathsf{T}\mathbf{Q} - q_2 \mathbf{p}_3^\mathsf{T}\mathbf{Q} \\ \mathbf{p}_1^\mathsf{T}\mathbf{Q} - q_1 \mathbf{p}_3^\mathsf{T}\mathbf{Q} \end{pmatrix} \right|\right|^2, \quad (3)$$

where $\mathbf{p}_i^\mathsf{T}$ are the rows of P. Using (1), we obtain the expression for the weight factors as:

$$w_{\mathbf{q},\widehat{\mathbf{q}}} = (\mathbf{p}_3^\mathsf{T}\mathbf{Q})^{-2}. \quad (4)$$

[1] we do not optimize the first camera parameters to (partially) fix the coordinate frame.

Intersection — Solving for the structure. Using equation (3), and if weight factors $w_{\mathbf{q},\widehat{\mathbf{q}}}$ are assumed to be known, one can recover the structure by solving the linear system $\mathsf{M}^{\text{us}}\mathbf{Q} = \mathbf{0}$ for each point feature $\mathbf{Q}$ considered where:

$$(\mathsf{M}^{\text{us}})_{2m\times 4} = \begin{pmatrix} \vdots \\ w_{\mathbf{q}\widehat{\mathbf{q}}} \begin{pmatrix} \mathbf{p}_2^\mathsf{T} - q_2\mathbf{p}_3^\mathsf{T} \\ \mathbf{p}_1^\mathsf{T} - q_1\mathbf{p}_3^\mathsf{T} \end{pmatrix} \\ \vdots \end{pmatrix}. \quad (5)$$

Each of the m views where the point $\mathbf{Q}$ is visible provides 2 equations. The solution is the unit singular vector corresponding to the smallest singular value of M^{us} which can be obtain using e.g. SVD. This naturally ensures the free scale of the recovered homogeneous vector since $||\mathbf{Q}||^2 = 1$.

The process is then iterated while re-estimating weight factors $w_{\mathbf{q},\widehat{\mathbf{q}}}$ to reduce the bias of the algebraic distance until convergence, which is assessed as in §3.

Resection — Solving for the motion. We use equation (3) to form a linear system $\mathsf{M}^{\text{um}}\mathbf{p}$ that can be solved using SVD where $\mathbf{p} \sim (\mathbf{p}_1^\mathsf{T}\ \mathbf{p}_2^\mathsf{T}\ \mathbf{p}_3^\mathsf{T})$ encapsulates the rows of P and:

$$(\mathsf{M}^{\text{um}})_{2n\times 12} = \begin{pmatrix} \vdots \\ w_{\mathbf{q}\widehat{\mathbf{q}}} \begin{pmatrix} \mathbf{0}_4^\mathsf{T} & \mathbf{Q}^\mathsf{T} & -q_2\mathbf{Q}^\mathsf{T} \\ \mathbf{Q}^\mathsf{T} & \mathbf{0}_4^\mathsf{T} & -q_1\mathbf{Q}^\mathsf{T} \end{pmatrix} \\ \vdots \end{pmatrix}.$$

Each of the n points that project onto view P provides 2 equations. The iteration is then conducted while re-estimating the weight factors, as for the structure. The free scale of P is fixed since $||\mathsf{P}||^2 = ||\mathbf{p}||^2 = 1$.

4.2. Calibrated Pin-Hole Cameras

We consider the calibrated full perspective camera model. This leads to metric reconstructions. In this case, resection is termed pose estimation and has been widely studied, see e.g. [3]. When the motion is frozen, solving for the structure can be conducted using the quasi-linear algorithm of §4.1.

Pose estimation from minimal data, i.e. $n = 3$ points is non-linear by nature since there exist four solutions in this case. However, when an initial guess is known, it can be refined using linear algebra. Difficulties arise because of non-linear constraints on the orthonormality of the recovered rotation matrix. An algorithm for pose refinement with $n \geq 3$ points that fits into our quasi-linear framework is described in the next section.

Quasi-linear pose estimation. Let $\mathsf{P} \sim \mathsf{K}(\mathsf{R}\ \mathbf{t})$ be the projection matrix where K is a 3×3 matrix containing the known intrinsic parameters, R a 3×3 rotation matrix and $\mathbf{t}$ a 3-vector. The unknowns are therefore R and $\mathbf{t}$. Under the assumption of zero skew and unity aspect ratiowhich are valid for most modern cameras, distances measured onto the image plane are equal to distances measured onto the retina up to a scale factor as $d^2(\mathbf{q},\widehat{\mathbf{q}}) = f^2 d^2(\mathbf{x},\widehat{\mathbf{x}})$ where $\mathbf{x} = \mathsf{K}^{-1}\mathbf{q}$ and $\widehat{\mathbf{x}}^{-1} = \mathsf{K}^{-1}\widehat{\mathbf{q}}$ are retinal points and f is the focal length (in number of pixels).

Linearly update the motion. Each term $d^2(\mathbf{q},\widehat{\mathbf{q}})$ of the double sum of the cost function $\mathcal{C}$ can be expanded as in the uncalibrated case, but using retinal points. This gives $d^2(\mathbf{q},\widehat{\mathbf{q}}) = f^2 w_{\mathbf{x},\widehat{\mathbf{x}}}^2 ||\mathsf{S}[\mathbf{x}]_\times \widehat{\mathbf{x}}||^2$. Using then perspective projection $\widehat{\mathbf{x}} \sim (\mathsf{R}\ \mathbf{t})\mathbf{Q}$ of the point $\mathbf{Q}^\mathsf{T} \sim (\bar{\mathbf{Q}}^\mathsf{T}\ \alpha)$ on the retina, we obtain $d^2(\mathbf{q},\widehat{\mathbf{q}}) = f^2 w_{\mathbf{x},\widehat{\mathbf{x}}}^2 ||\mathsf{S}[\mathbf{x}]_\times (\mathsf{R}\bar{\mathbf{Q}} + \alpha\mathbf{t})||^2$. If we approximate the rotation between two iterations of the algorithm by a first order taylor expansion, $(\mathrm{I} + [\mathbf{w}]_\times)$, we obtain after some minor algebraic manipulations $d^2(\mathbf{q},\widehat{\mathbf{q}}) = f^2 w_{\mathbf{x},\widehat{\mathbf{x}}}^2 ||\mathsf{S}[\mathbf{x}]_\times \mathsf{R}\bar{\mathbf{Q}} - \mathsf{S}[\mathbf{x}]_\times [\bar{\mathbf{Q}}]_\times \mathbf{w} + \alpha \mathsf{S}[\mathbf{x}]_\times \mathbf{t}||^2 + \mathcal{O}$, where $\mathcal{O}$ contains higher order terms. By neglecting $\mathcal{O}$, we can estimate the first order pose update $\mathbf{m}^\mathsf{T} = (\mathbf{w}^\mathsf{T}\ \mathbf{t}^\mathsf{T})$ by solving the linear system $\mathsf{M}^{\text{cm}}\mathbf{m} = \mathbf{d}^{\text{cm}}$, where:

$$(\mathsf{M}^{\text{cm}})_{2n\times 6} = \begin{pmatrix} \vdots \\ f^2 w_{\mathbf{x},\widehat{\mathbf{x}}}^2 \begin{pmatrix} \mathsf{S}[\mathbf{x}]_\times \mathsf{R}[\bar{\mathbf{Q}}]_\times & -\alpha \mathsf{S}[\mathbf{x}]_\times \end{pmatrix} \\ \vdots \end{pmatrix}$$

$$(\mathbf{d}^{\text{cm}})_{2n} = \begin{pmatrix} \vdots \\ f^2 w_{\mathbf{x},\widehat{\mathbf{x}}}^2 \left(\mathsf{S}[\mathbf{x}]_\times \mathsf{R}\bar{\mathbf{Q}}\right) \\ \vdots \end{pmatrix}.$$

Algorithm. The algorithm for quasi-linear pose estimation combines first order updates of pose and re-estimation of calibrated weight factors $w_{\mathbf{x},\widehat{\mathbf{x}}} = \frac{1}{f} w_{\mathbf{q},\widehat{\mathbf{q}}}$:

1. *solve for pose update* using $\mathsf{M}^{\text{cm}}\mathbf{m} = \mathbf{d}^{\text{cm}}$;
2. *update rotation*: $\mathsf{R} \leftarrow \mathsf{R}\mathsf{R}_{\mathbf{w}}$, where $\mathsf{R}_{\mathbf{w}}$ gives the rotation corresponding to $\mathbf{w}^\mathsf{T} = (w_1\ w_2\ w_3)$ using e.g. Euler angles;
3. re-estimate *calibrated weight factors* $w_{\mathbf{x},\widehat{\mathbf{x}}}$ using (4);
4. *iterate* steps 1-3 until convergence, see §3.

5. Experimental Results

In the following two sections, we compare previously described algorithms to existing ones. We use simulated data to compare them in the most general case, i.e. projective. We then consider real images and both projective and metric spaces. More detailled results, in particular on pose estimation and for the affine camera will be made available in a longer version of the paper. We compare the following algorithms. LIN is structure and motion initialization [1], QLIN is the proposed quasi-linear method and LM relies on the Levenberg-Marquardt non-linear optimizer. Note that our implementation uses analytical differentiation and a sparse inverser for the Hessian matrix, see [7]. We also compared a full Hessian inversion-based implementation that we denote as LM-full.

The threshold for convergence (see §3) is the same for QLIN and LM and is equal to 10e^{-8}. Image point coordinates are standardized such that they lie in $[-1\ldots 1]^2$.

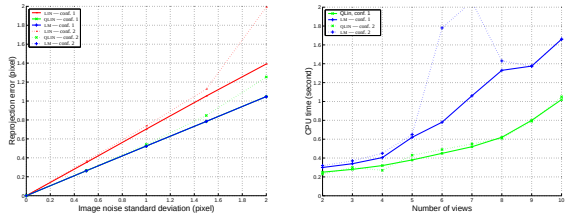
5.1. Simulated Data

The simulated scene consists of 3D point features chosen at random inside a cube of 2 meters side length. We simulate two camera configurations. The first one is stable and the second one contains nearly unreconstructable points. We measure the reprojection error to assess the convergence accuracy

and the CPU time to convergence to compare computational costs. In order to simulate realistic situations, we adopted the following parameters: the focal length of the cameras is 1000 (expressed is number of pixels) and the image size is 512×512. Values plotted have been averaged over 50 trials. Median values gave graphs similar to those for the means. The curves for 3D errors show expected results, see e.g. [6].

Convergence accuracy. The standard configuration is 50 points, 10 views and an added image noise with a Gaussian distribution of 0.5 pixel standard deviation.

Figures 1 (a) shows the results when varying the image noise. We observe that for camera configuration 1, there is no difference between QLIN and LM, even when the starting point given by LIN lies quite far from the final result. The error for these two methods degrades gracefully as image noise increases. For camera configuration 2, figure 1 (a) shows that there is a slight difference between QLIN and LM only when the level of added image noise is beyond 1 pixel. LM-full, not shown on these graphs, gave exactly the same results as LM.



(a) (b)

Figure 1. (a): reprojection error at convergence versus added image noise. The curves for QLIN and LM and the first configuration, and for LM and the second configuration are nearly undistinguishable. (b): CPU time to convergence versus the number of views.

Computation time. We observe on figure 1 (b) that the quasi-linear optimization requires less CPU time to convergence than the non-linear one whatever the number of views. Results from another experiment where added image noise is varied from 0 to 2 pixels, shows that CPU time to convergence linearly increases with the added image noise. The CPU time needed by LM is twice thus needed by QLIN. LM-full gave results out of range of these graphs.

5.2. Real Images

We use the 181 images of the hotel sequence[2] and 197 corner correspondences. These corners have been automatically detected and semi-automatically matched through the different views. Calibration data were approximately known for this sequence. The results given in table 1 show that QLIN converges to the same residual error as LM and LM-full. In terms of computational cost, we observe that QLIN is roughly 2.5 times faster than LM.

[2]These data have been provided by the Modeling by Videotaping group in the Robotics Institute, Carnegie Mellon University.

step	approach	first 5 images		full 181 images sequence	
		rep. error (pixel)	CPU time (second)	rep. error (pixel)	CPU time (second)
init.	-	9.4209	-	6.2524	-
ba	LM-full	0.0357	1056 ≈17 mn.	-	-
	LM	0.0357	2.55	2.2026	96
	QLIN	0.0391	0.95	2.2034	37

Table 1. Reprojection errors (pixel) and CPU time to convergence (second) for various algorithms and sequence lengths.

6. Conclusions and Perspectives

We addressed the problem of structure and motion refinement using bundle adjustment. We perform this task based on quasi-linear optimization while keeping the original cost function of bundle adjustment. We split the problem into structure refinement on the one hand and motion refinement on the other hand. The principle is then to rewrite the Euclidean distance used in bundle adjustment as a weighted algebraic distance. Quasi-linear optimization iteratively updates these weights while refining structure or motion.

The result is a disarmigly simple algorithm that comes out to be a loop over two weighted linear systems constructed as simple functions as the input data.

We conducted numerous experiments on simulated and real data. In the light of these results, we observed that this algorithm is as accurate as standard Levenberg-Marquardt-based bundle adjusters in terms of convergence accuracy while being much faster in terms of computational cost.

Acknowledgement. I wish to thank Frederik Schaffalitzky for providing his implementation of LM used in §5.

References

[1] P. Beardsley, P. Torr, and A. Zisserman. 3D model acquisition from extended image sequences. In *ECCV*, 1996.
[2] Q. Chen and G. Medioni. Efficient iterative solution to M-view projective reconstruction. In *CVPR*, 1999.
[3] R. Haralick, H. Joo, C. Lee, X. Zhuang, V. Vaidya, and M. Kim. Pose estimation from corresponding point data. IEEE *SMC*, 6(19):1426–1446, 1989.
[4] R. Hartley. Minimizing algebraic error. In *ICCV*, 1998.
[5] S. Mahamud, M. Herbert, Y. Omori, and J. Ponce. Provably-convergent iterative methods for projective structure and motion. In *CVPR*, 2001.
[6] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *ECCV*, 1996.
[7] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle ajustment — a modern synthesis. In *Vision Algorithms: Theory and Practice*, 2000.
[8] Z. Zhang. Motion and structure from two perspective views: From essential parameters to euclidean motion via fundamental matrix. *Journal of the Optical Society of America A*, 1997.