Power Bundle Adjustment for Large-Scale 3D Reconstruction

Simon Weber, Nikolaus Demmel, Tin Chon Chan, and Daniel Cremers

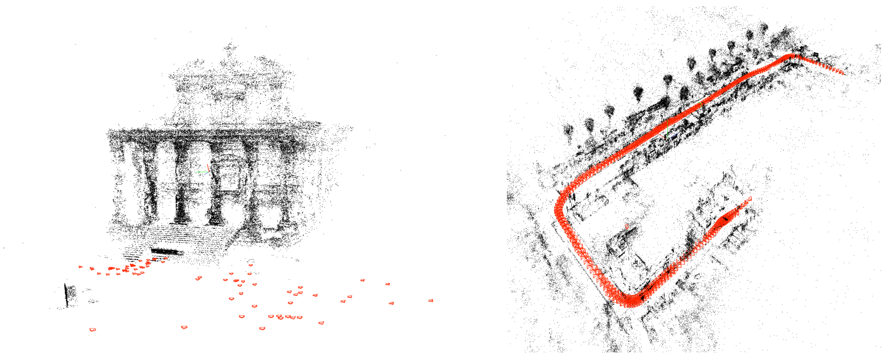
Technical University of Munich
{sim.weber,nikolaus.demmel,tinchon.chan,cremers}@tum.de

Abstract. We present the design and the implementation of a new expansion type algorithm to solve large-scale bundle adjustment problems. Our approach—called Power Bundle Adjustment—is based on the power series expansion of the inverse Schur complement. This initiates a new family of solvers that we call inverse expansion methods. We show with the real-world BAL dataset that the proposed solver challenges the traditional direct and iterative methods. The solution of the normal equation is significantly accelerated, even for reaching a very high accuracy. Last but not least, our solver can also complement a recently presented distributed bundle adjustment framework. We demonstrate that employing the proposed Power Bundle Adjustment as a sub-problem solver greatly improves speed and accuracy of the distributed optimization.

Keywords: Bundle Adjustment, 3D Reconstruction, Power Series, Schur Complement, Optimization

1 Introduction

Bundle adjustment (BA) is the core component of many 3D reconstruction methods and Structure from Motion (SfM) algorithms and represents a very popular and challenging problem in the computer vision community. BA can be very simply formulated as the joint estimation of camera parameters and 3D landmark positions via the minimization of a non-linear reprojection error. The recent emergence of large-scale internet photo collections [1] questions the scalability of BA methods. Also, building accurate city-scale maps for applications such as augmented reality or autonomous driving challenges the limits of current BA approaches. As the solution of the normal equation is the most time consuming step of BA, the Schur complement trick is usually employed to form the reduced camera system (RCS). This linear system then estimates only the pose parameters and is significantly smaller. The RCS is commonly solved by iterative methods such as the very popular preconditioned conjugate gradients algorithm for large-scale problems or by direct methods such as Cholesky factorization for small-scale problems. We challenge these two families of solvers by relying on an iterative approximation of the inverse Schur complement. In particular, our contributions are as follows:



(a) *Final-93* (b) *Ladybug-1031*

Fig. 1: (a) Optimized 3D reconstruction of a *final* BAL problem with 93 poses. PoBA is 73% faster than the best competing solver to reach a cost tolerance of 1%. (b) Optimized 3D reconstruction of a *Ladybug* BAL problem with 1031 poses. PoBA is 90% faster than the best competing solver to reach a cost tolerance of 1%.

- We introduce Power Bundle Adjustment (PoBA) for efficient large-scale BA. It inaugurates a new family of techniques that we call *inverse expansion methods* to challenge traditional direct and iterative methods.
- We link the Schur complement to the theory of power series and we provide a theoretical proof to justify this expansion in the context of BA.
- We perform extensive evaluation of the proposed approach on the BAL dataset and compare to state-of-the-art direct and iterative methods. Fig. 1 shows two out of the 97 evaluated BAL problems.
- We incorporate our solver into a recently proposed distributed BA framework and show a significant improvement in terms of speed and accuracy.
- We release our solver as open source to facilitate further research.

2 Related Work

As we propose a new way to solve large-scale bundle adjustment problems we review works on bundle adjustment and on traditional solving methods, that is, direct and iterative methods. We also briefly mention some works on power series. For a general introduction to series expansion we refer the reader to [12].

Scalable bundle adjustment.

A detailed survey of bundle adjustment can be found in [14]. The Schur complement [18] is the prevalent way to exploit the sparsity of the BA Problem. The choice of resolution method is typically governed by the size of the normal equation: With increasing size, direct methods such as sparse and dense

Cholesky factorization [13] are outperformed by iterative methods such as inexact Newton algorithms. Large-scale bundle adjustment problems with tens of thousands images are typically solved by conjugate gradients algorithm [1,8,2]. Some variants have been designed, for instance the search-space can be enlarged [15] or a visibility-based preconditioner can be used [9]. One line of work on square root bundle adjustment proposes to replace the Schur complement for eliminating landmarks with nullspace projection [4,5]. However, they still rely on traditional solvers for the reduced camera system, i.e. PCG for large-scale [4] and Cholesky decomposition for small-scale [5] problems. Nevertheless, even with PCG, solving the normal equation is still the bottleneck and finding thousands of unknown parameters needs a huge number of inner iterations. Other authors try to improve the runtime of BA with PCG by focusing on efficient parallelization [11]. Recently, Stochastic BA [20] stochastically decomposes the reduced camera system into subproblems and solves the smaller normal equation by dense factorization, which leads to a distributed optimization framework with improved speed and scalability. We propose to alleviate the shortcomings of these direct and iterative methods with a novel BA approach based on power series expansion.

Power series solver.

While power series expansion is common to solve differential equations [3], it has, to the best of our knowledge, never been employed for solving the bundle adjustment problem. A recent work [19] links the Schur complement to Neumann polynomial expansion to build a new preconditioner. Nevertheless this technique still reverts to an iterative method for solving the linear system. In contrast, we propose to directly apply the power series expansion of the inverse Schur complement to solve the BA problem. Our solver therefore falls in the category of expansion methods that have never been proposed for the BA problem.

3 Power Series

We briefly introduce power series expansion of a matrix. Let $\rho(A)$ denote the spectral radius of a square matrix A, i.e. the largest absolute eigenvalue and denote the spectral norm by $\|A\| = \rho(A)$. The following proposition holds:

Proposition 1. *Let M be a $n \times n$ matrix. If the spectral radius of M satisfies $\rho(M) < 1$, then*

$$(I - M)^{-1} = \sum_{i=0}^{m} M^i + R \,, \tag{1}$$

where the error matrix

$$R = \sum_{i=m+1}^{\infty} M^i \,, \tag{2}$$

satisfies

$$\|R\| \leqslant \frac{\|M\|^{m+1}}{1 - \|M\|} \,. \tag{3}$$

A proof is provided in Appendix and an illustration with real problems is given in Section 5.

4 Power Bundle Adjustment

We consider a very general form of bundle adjustment with n_p poses and n_l landmarks. Let $x = (x_p, x_l)$ be the state vector containing all the optimization variables, with x_p of length $d_p n_p$ associating to extrinsic and eventually intrinsic camera parameters for all poses and x_l of length $3n_l$ associating to the 3D coordinates of all landmarks. Generally, $d_p = 6$ if only extrinsic parameters are unknown. For the evaluated BAL problems we additionally estimate intrinsic parameters and $d_p = 9$. The objective is to minimize the total bundle adjustment energy

$$F(x) = \frac{1}{2}\|r(x)\|_2^2 = \frac{1}{2}\sum_i \|r_i(x)\|_2^2 \,, \tag{4}$$

where $r(x) = [r_1(x), ..., r_k(x)]$ is the concatenation of all residuals for a 3D reconstruction.

4.1 Least Squares Problem

This minimization problem is commonly solved with the Levenberg-Marquardt (LM) algorithm, which is based on the first-order Taylor approximation of $r(x)$ around the current state estimate $x^0 = (x_p^0, x_l^0)$. By adding a regularization term to improve convergence the minimization turns into

$$\min_{\Delta x_p, \Delta x_l} \frac{1}{2}(\|r^0 + \begin{pmatrix} J_p & J_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix}\|^2 + \lambda \|\begin{pmatrix} D_p & D_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix}\|_2^2) \,, \tag{5}$$

with $r^0 = r(x^0)$, $J_p = \frac{\partial r}{\partial x_p}|_{x^0}$, $J_l = \frac{\partial r}{\partial x_l}|_{x^0}$, λ a damping coefficient, and D_p and D_c diagonal damping matrices for pose and landmark variables. This damped problem leads to the corresponding normal equation

$$H \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} = - \begin{pmatrix} b_p \\ b_l \end{pmatrix} \,, \tag{6}$$

where

$$H = \begin{pmatrix} U_\lambda & W \\ W^\top & V_\lambda \end{pmatrix} \,, \tag{7}$$

$$U_\lambda = J_p^\top J_p + \lambda D_p^\top D_p, \tag{8}$$

$$V_\lambda = J_l^\top J_l + \lambda D_l^\top D_l, \tag{9}$$

$$W = J_p^\top J_l, \tag{10}$$

$$b_p = J_p^\top r^0, \; b_l = J_l^\top r^0 \,. \tag{11}$$

U_λ, V_λ and H are symmetric positive-definite [14].

4.2 Schur Complement

As inverting the system matrix H of size $(d_p n_p + 3n_l)^2$ directly tends to be excessively costly for large-scale problems it is common to reduce it by using the Schur complement trick. The idea is to form the reduced camera system

$$S \Delta x_p = -\tilde{b}\,, \tag{12}$$

with

$$S = U_\lambda - W V_\lambda^{-1} W^\top\,, \tag{13}$$

$$\tilde{b} = b_p - W V_\lambda^{-1} b_l\,. \tag{14}$$

(12) is then solved for Δx_p. The optimal Δx_l is obtained by back-substitution:

$$\Delta x_l = -V_\lambda^{-1}(-b_l + W^\top \Delta x_p)\,. \tag{15}$$

4.3 Power Series Expansion

The Schur complement can be rewritten as

$$S = U_\lambda (I - U_\lambda^{-1} W V_\lambda^{-1} W^\top)\,, \tag{16}$$

and then

$$S^{-1} = (I - U_\lambda^{-1} W V_\lambda^{-1} W^\top)^{-1} U_\lambda^{-1}\,. \tag{17}$$

In order to expand (17) into a power series as detailed in Proposition 1, we require the following result:

Lemma 1. *The spectral radius of* $U_\lambda^{-1} W V_\lambda^{-1} W^\top$ *satisfies*

$$\rho(U_\lambda^{-1} W V_\lambda^{-1} W^\top) < 1\,. \tag{18}$$

Proof. Let $\mu(U_\lambda^{-1} W V_\lambda^{-1} W^\top)$ be a given eigenvalue of the matrix $U_\lambda^{-1} W V_\lambda^{-1} W^\top$. In the following, we will derive the inequality

$$-1 < \mu(U_\lambda^{-1} W V_\lambda^{-1} W^\top) < 1\,, \tag{19}$$

that proves Lemma 1. On the one hand $U_\lambda^{-\frac{1}{2}} S U_\lambda^{-\frac{1}{2}}$ is symmetric positive definite as S and U_λ are both symmetric positive definite. It follows that the eigenvalues of $U_\lambda^{-1} S$ are all real and positive, due to its similarity with $U_\lambda^{-\frac{1}{2}} S U_\lambda^{-\frac{1}{2}}$. From

$$\begin{aligned} U_\lambda^{-1} W V_\lambda^{-1} W^\top &= U_\lambda^{-1}(U_\lambda - S) \\ &= I - U_\lambda^{-1} S\,, \end{aligned} \tag{20}$$

we conclude that

$$\mu(U_\lambda^{-1} W V_\lambda^{-1} W^\top) < 1\,. \tag{21}$$

On the other hand

$$\Phi = U_\lambda^{-\frac{1}{2}}(U_\lambda + WV_\lambda^{-1}W^\top)U_\lambda^{-\frac{1}{2}} \tag{22}$$

is symmetric positive definite as U_λ and V_λ^{-1} are symmetric positive definite. From

$$\begin{aligned} 2I - U_\lambda^{-1}S &= U_\lambda^{-1}(2U_\lambda - S) \\ &= U_\lambda^{-1}(U_\lambda + WV_\lambda^{-1}W^\top)\,, \end{aligned} \tag{23}$$

it follows that the eigenvalues of $2I - U_\lambda^{-1}S$ are all positive due to its similarity with Φ and then

$$\mu(U_\lambda^{-1}WV_\lambda^{-1}W^\top) > 1 - 2 = -1\,, \tag{24}$$

that concludes the proof. □

Due to Lemma 1 the assumption of Proposition 1 is fulfilled. By applying a power series expansion to

$$(I - U_\lambda^{-1}WV_\lambda^{-1}W^\top)^{-1}\,, \tag{25}$$

the approximate inverse Schur Complement at order m is of the form

$$S^{-1} \approx \sum_{i=0}^{m}(U_\lambda^{-1}WV_\lambda^{-1}W^\top)^i U_\lambda^{-1}\,. \tag{26}$$

Alternative factorization. Alternatively to the left-factorization (16) a right-factorization is possible:

$$S = (I - WV_\lambda^{-1}W^\top U_\lambda^{-1})U_\lambda\,, \tag{27}$$

that gives the following power series expansion:

$$S^{-1} \approx \sum_{i=0}^{m} U_\lambda^{-1}(WV_\lambda^{-1}W^\top U_\lambda^{-1})^i\,, \tag{28}$$

mathematically equivalent to (26). The implementation is different but gives similar results in term of runtime. The interested reader can find more details in Appendix.

Stopping criterion. As the power series expansion of the inverse Schur is derived iteratively a termination rule is necessary. By definition of the error matrix R in Proposition 1 the sequence of summands

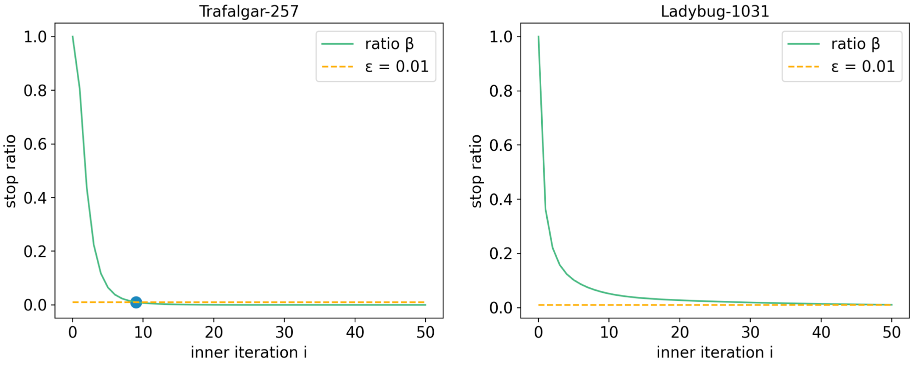
$$(\|(U_\lambda^{-1}WV_\lambda^{-1}W^\top)^i U_\lambda^{-1}\tilde{b}\|_2)_{i>=0} \tag{29}$$

converges to 0. By analogy with inexact Newton methods [16] we set a stop criterion depending on the initial iteration:

$$U_\lambda^{-1}\tilde{b}\,, \tag{30}$$

ϵ	10^{-1}	10^{-2}	10^{-3}	0
average relative runtime	1.52	1.12	1.16	1.22
average outer iterations	4.0	2.7	2.6	2.6
average mean inner iterations	12	31	47	50

Table 1: We investigate the sensitivity of PoBA to the stop criterion for a range of threshold $\epsilon \in \{0.1, 0.01, 0.001, 0\}$ over all BAL problems and we set the maximal number of inner iterations to 50. For each problem the relative runtime is the ratio between the runtime of the considered solver and the runtime of the most efficient solver to reach the smallest objective function up to a 1% accuracy. Similarly, we are interested in the number of LM outer iterations to reach this objective function. For each LM outer iteration we take into consideration the order m of the power expansion up to the considered threshold ϵ and take the mean over the number of LM outer iterations. We then take the average of all these values over the number of problems. It clearly appears that the stop criterion plays a role to reach the best performance but is not so decisive if we set a reasonable threshold. Compared to our baseline threshold $\epsilon = 0.01$, a larger threshold $\epsilon = 0.1$ is on average 35% slower whereas a smaller threshold $\epsilon = 0.001$ and absence of threshold are only 3% and 9% slower, respectively.



(a) *Trafalgar-257* (b) *Ladybug-1031*

Fig. 2: Illustration of the stop criterion designed in PoBA for the first LM iteration for two BAL problems: (a) *Trafalgar* with 257 poses and (b) *Ladybug* with 1031 poses. The maximal number of inner iterations is set to 50. The ratio β is plotted in green. The threshold $\epsilon = 0.01$ is plotted in yellow. The order m of the inverse Schur complement approximation is the smallest inner iteration number such that $\beta < \epsilon$. Graphically it is represented as the intersection point between yellow and green curves. In (a) $m = 9$. In (b) $m = m_{max} = 50$.

and on the current iteration i:

$$(U_\lambda^{-1} W V_\lambda^{-1} W^\top)^i U_\lambda^{-1} \tilde{b}\,. \tag{31}$$

Similar to the conjugate gradients algorithm ([8,1]) or iterative solvers like Gauss-Seidel, we simply compare the ratio between current and initial iteration to a threshold ϵ. The sum over i in (26) stops whenever the ratio is smaller than ϵ. More formally we set the stop criterion for the inverse Schur approximation as

$$\|(U_\lambda^{-1} W V_\lambda^{-1} W^\top)^i U_\lambda^{-1} \tilde{b}\|_2 < \epsilon \|U_\lambda^{-1} \tilde{b}\|_2 \tag{32}$$

for a given ϵ.

5 Implementation

Our experiments are based on the publicly available implementation[1] of Stochastic Bundle Adjustment (STBA) [20].

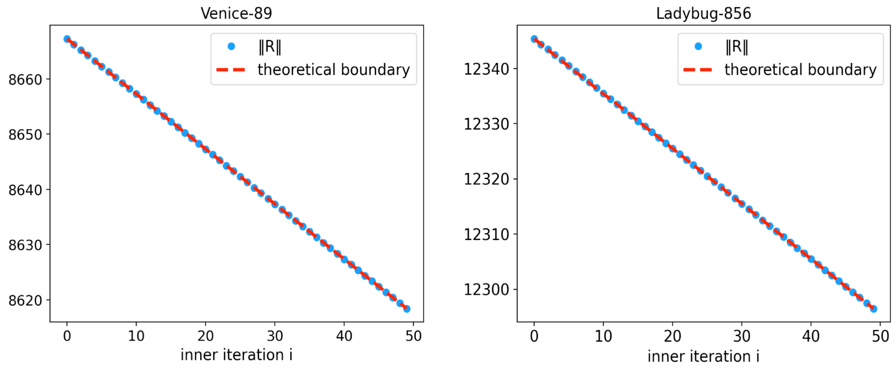
We implement our solver PoBA in C++ with Eigen [7] for linear algebra computations. For comparisons we use the same state-of-the-art baselines as [20] (see Section 5.1). All these algorithms share the same code base, so the comparisons can be made equitably. Moreover, we also incorporate our solver inside the distributed STBA framework (see Section 5.4). This new solver that we call *power stochastic bundle adjustment* (PoST) is directly compared with STBA. We run experiments on MacOS 11.2 with an Intel Core i5 and 4 cores at 2GHz.

5.1 Baseline Comparisons

As in [20] we use two standard trust region algorithms: Levenberg-Marquardt (LM) and Dogleg (DL). Inside LM algorithm we solve (12) with two variants: (a) the direct method (LM-sparse) exploits the LL^T Cholesky factorization; (b) the iterative method (LM-iterative) is the conjugate gradients preconditioned with the competitive Schur-Jacobi preconditioner [1] built with Eigen. DL is used with the same sparse solver as LM-sparse [10]. More details about these algorithms can be found in the Appendix. For the errors we use the Huber loss with a scale parameter of 1 to improve the robustness [17].

Performance Profiles. To compare a set of solvers the user may be interested in two factors, a lower runtime and a better accuracy. Performance profiles [6] evaluate both jointly. Let S and P be respectively a set of solvers and a set of problems. Let $f_0(p)$ be the initial objective and $f(p, s)$ the final objective that is reached by solver $s \in S$ when solving problem $p \in P$. The minimum objective the

[1] `https://github.com/zlthinker/STBA`



(a) *Venice-89* (b) *Ladybug-856*

Fig. 3: Illustration of the inequality (3) in Proposition 1 for the first LM iteration of two BAL problems: (a) *Venice* with 89 poses and (b)*Ladybug* with 856 poses. The left-side of the inequality is plotted in blue colour and represents the spectral norm of the error matrix R when the order m of the approximate inverse Schur complement varies from 0 to 50. The right-side of the inequality plotted in red represents the theoretical upper bound of the spectral norm of the error matrix and depends on the considered m and on the spectral norm of $M = U_\lambda^{-1} W V_\lambda^{-1} W^\top$. With Spectra library [21] $\rho(M)$ takes the values (a) 0.999885 for *V-89* and (b) 0.999919 for *L-856*. Both values are smaller than 1, as stated in Lemma 1. Blue and red curves overlap: The inequality (3) holds and is almost an equality in these examples.

solvers in S attain for a problem p is $f^*(p) = \min_{s \in S} f(p, s)$. Given a tolerance $\tau \in (0, 1)$ the objective threshold for a problem p is given by

$$f_\tau(p) = f^*(p) + \tau(f^0(p) - f^*(p)) \tag{33}$$

and the runtime a solver s needs to reach this threshold is noted $T_\tau(p, s)$. It is clear that the most efficient solver s^* for a given problem p reaches the threshold with a runtime $T_\tau(p, s^*) = \min_{s \in S} T_\tau(p, s)$. Then, the performance profile of a solver for a relative runtime α is defined as

$$\rho(s, \alpha) = \frac{100}{|P|} |\{p \in P | T_\tau(p, s) \leqslant \alpha \min_{s \in S} T_\tau(p, s)\}| \tag{34}$$

Graphically the performance profile of a given solver is the percentage of problems solved faster than the relative runtime α on the x-axis.

5.2 Experimental Settings

Dataset. For our extensive evaluation we use all 97 bundle adjustment problems from the BAL project page. They are divided within five problems families.

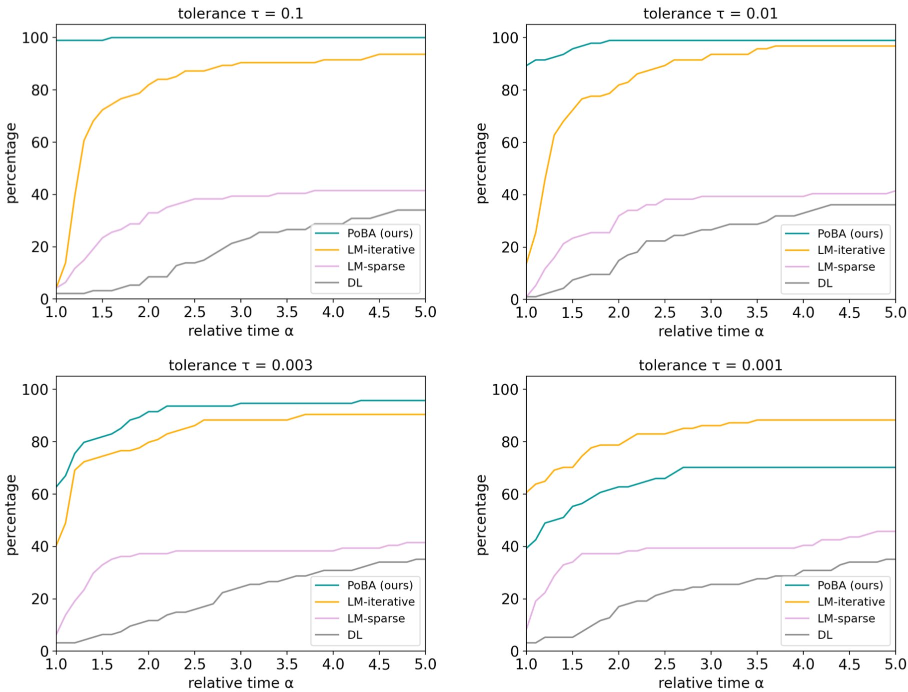
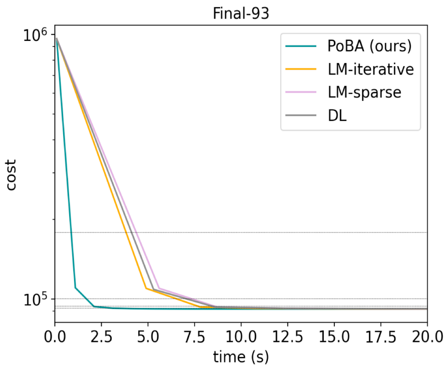


Fig. 4: Performance profiles for all BAL problems show the percentage of problems solved to a given accuracy tolerance $\tau \in \{0.1, 0.01, 0.003, 0.001\}$ with relative runtime α. Our proposed solver PoBA using series expansion of the Schur complement significantly outperforms all the competing solvers up to the high accuracy $\tau = 0.003$.

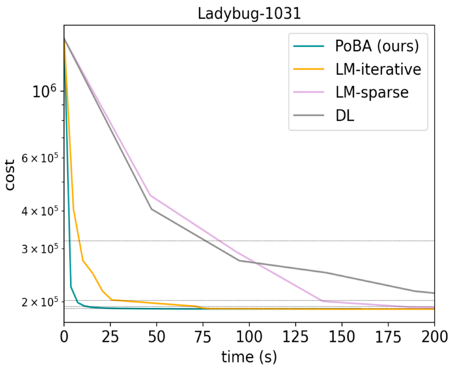
Ladybug is composed with images captured by a vehicle with regular rate. Images of *Venice*, *Trafalgar* and *Dubrovnik* come from Flickr.com and have been saved as skeletal sets [1]. Recombination of these problems with additional leaf images leads to the *Final* family. Details about these problems can be found in Appendix.

LM loop. PoBA is in line with the implementation [20]. Starting with damping parameter 10^{-4} we update λ depending on the success or failure of the LM loop. We set the maximal number of LM iterations to 50, terminating earlier if a relative function tolerance of 10^{-6} is reached. For LM-iterative we set the maximal number of inner iterations in the preconditioned conjugate gradients step to 500, terminating earlier if a threshold 10^{-6} is reached.

Power Schur complement. We directly apply the power series expansion on the right-hand-side of (12) and we get Δx_p. This step is very efficient as the



(a) *Final-93*



(b) *Ladybug-1031*

Fig. 5: Convergence plots of (a) *Final-93* from BAL dataset with 93 poses and (b) *Ladybug-1031* from BAL dataset with 1031 poses. Fig. 1 shows a visualization of 3D landmarks and camera poses for these problems. The dotted lines correspond to cost thresholds for the tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$.

application of (26) on a vector only involves matrix-vector products associated with U_λ^{-1} and $WV_\lambda^{-1}W^T$. The step-by-step algorithm is given in Appendix. We set the maximal number of inner iterations to 50 and a threshold $\epsilon = 0.01$. Table 1 presents an evaluation of the sensitivity to the stop criterion ϵ for all BAL problems and Fig. 2 illustrates the behaviour of the stop criterion ratio

$$\beta = \frac{\|(U_\lambda^{-1}WV_\lambda^{-1}W^\top)^m U_\lambda^{-1}\tilde{b}\|_2}{\|U_\lambda^{-1}\tilde{b}\|_2} \tag{35}$$

until it reaches the threshold $\epsilon = 0.01$ in the first outer iteration of two different BAL problems. Fig. 3 illustrates the bounded error (3) of Proposition 1 with two examples from BAL.

5.3 Analysis

Fig. 4 presents the performance profiles with all BAL problems for four different tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$. We can see that PoBA greatly outperforms LM-iterative, LM-sparse and DL for all relative time α when $\tau = 0.1$ and $\tau = 0.01$. PoBA is still the best one for excellent accuracy $\tau = 0.003$. Fig. 5 shows the convergence for two differently sized BAL problems. The y-axis represents the total BA cost of the energy function.

5.4 Power Stochastic Bundle Adjustment (PoST)

Stochastic Bundle Adjustment. STBA decomposes the reduced camera system into clusters inside the Levenberg-Marquardt iterations. The per-cluster

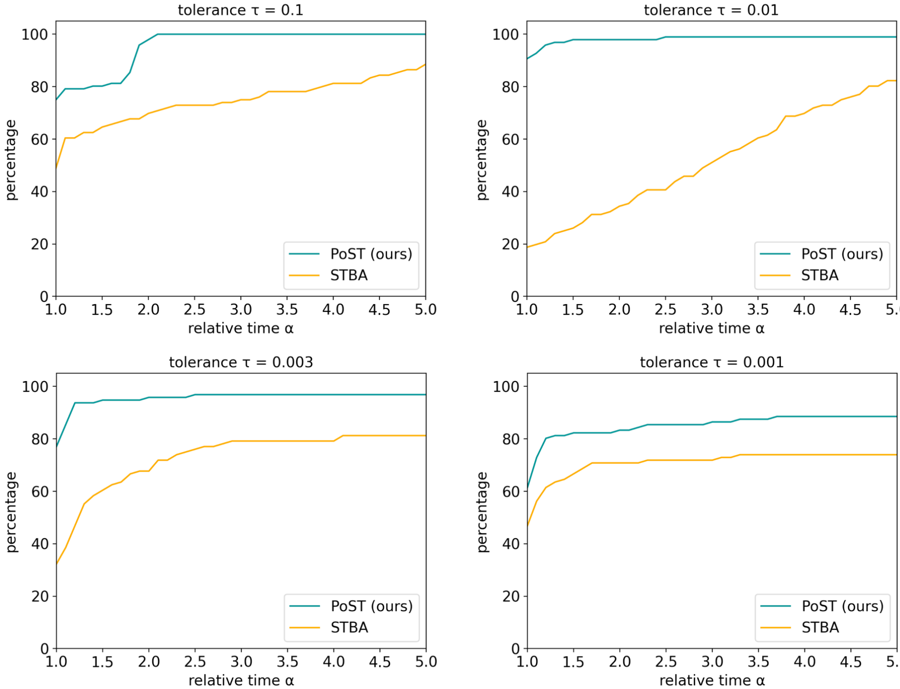
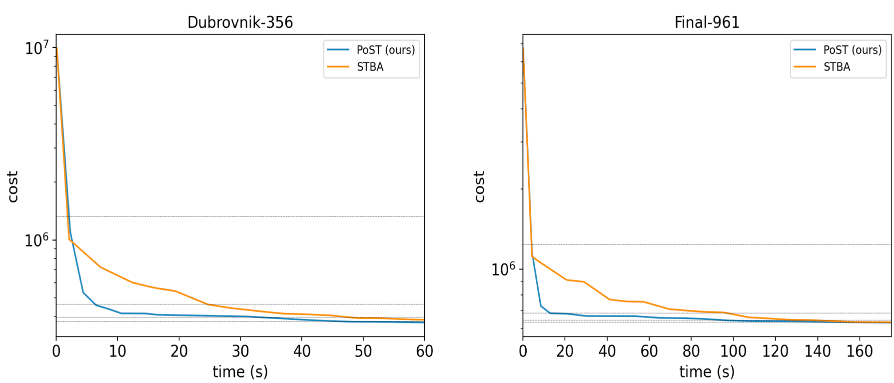


Fig. 6: Performance profiles for all BAL problems with stochastic framework. Our proposed solver PoST outperforms the challenging STBA across all accuracy tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$, both in terms of speed and precision.



(a) *Dubrovnik-356* (b) *Final-961*

Fig. 7: Convergence plots of (a) *Dubrovnik-356* from BAL dataset with 356 poses and (b) *Final-961* from BAL dataset with 961 poses. The dotted lines correspond to cost thresholds for the tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$.

linear sub-problems are then solved in parallel with dense $LL^\top$ factorization due to the dense connectivity inside camera clusters. As shown in [20] this approach outperforms the baselines in terms of runtime and scales to very large BA problems, where it can even be used for distributed optimization. For further explanations we refer the reader to Appendix and to [20]. In the following we show that replacing the sub-problem solver with our Power Bundle Adjustment can significantly boost runtime even further.

Power Stochastic Bundle Adjustment (PoST). We extend STBA by incorporating our solver instead of the dense $LL^\top$ factorization. Each subproblem is then solved with a power series expansion of the inverse Schur complement. We keep the same parameters as in Section 5.1 and we set the maximal cluster size to 100, in accordance to [20].

Analysis. Fig. 6 presents the performance profiles with all BAL problems for different tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$. PoST clearly outperforms STBA for each tolerance, most notably for $\tau = 0.01$. Fig. 7 shows the convergence plot for two differently sized BAL problems.

6 Conclusion

We introduce a new class of large-scale bundle adjustment solvers that makes use of a power expansion of the inverse Schur complement. We prove the validity of the proposed approximation. Moreover, we experimentally confirm that the proposed power series representation of the inverse Schur complement challenges traditional direct and iterative solvers in terms of speed and accuracy. Last but not least, we show that the power series representation can complement distributed bundle adjustment methods to significantly boost performance for large-scale 3D reconstruction.

References

1. S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *European Conference on Computer Vision (ECCV)*, pages 29-42. Springer, 2010.
2. M. Byröd, K. Åström. Conjugate gradient bundle adjustment. In *European Conference on Computer Vision (ECCV)*, 2010.
3. E. A. Coddington, N. Levinson. Theory of Ordinary Differential Equations. McGraw–Hill, 1955.
4. N. Demmel et al. Square Root Bundle Adjustment for Large-Scale Reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.
5. N. Demmel et al. Square Root Marginalization for Sliding-Window Bundle Adjustment. In *International Conference on Computer Vision (ICCV)*, 2021.
6. E. D. Dolan, and J. J. More. Benchmarking optimization software with performance profiles. In *Mathematical programming* 91(2), pages 201–213, 2002.

7. G. Guennebaud, and B. Jacob, et al. *Eigen v3*, `http://eigen.tuxfamily.org`, 2010.
8. M. R. Hestenes, and E. Stiefel. Methods of conjugate gradients for solving linear systems. In *Journal of research of the National Bureau of Standards* 49(6), pages 409-436, 1952.
9. A. Kushal, and S. Agarwal. Visibility based preconditioning for bundle adjustment. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
10. M. Lourakis, A. A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In *International Conference on Computer Vision (ICCV)*, 2005.
11. J. Ren et al. MegBA: A High-Performance and Distributed Library for Large-Scale Bundle Adjustment. arXiv. 2021.
12. Y. Saad. Itervative methods for sparse linear systems, 2nd ed. In *SIAM*, Philadelpha, PA, 2003.
13. L. Trefethen, D. Bau. Numerical linear algebra. SIAM, 1997.
14. B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment-a modern synthesis. In *International workshop on vision algorithms*, pages 298-372. Springer, 1999.
15. S. Weber, N. Demmel, and D. Cremers. Multidirectional conjugate gradients for scalable bundle adjustment. In *German Conference on Pattern Recognition (GCPR)*, pages 712-724. Springer, 2021.
16. S. J. Wright, and J. N. Holt. An inexact Levenberg-Marquardt method for large sparse nonlinear least squares. In *J. Austral. Math. Soc. Ser. B 26*, pages 387-403, 1985.
17. C. Zach. Robust bundle adjustment revisited. In *European Conference on Computer Vision (ECCV)*, 2014.
18. F. Zhang. The Schur complement and its applications. In Numerical Methods and Algorithms. Vol. 4, Springer, 2005.
19. Q. Zheng, Y. Xi, and Y. Saad. A power Schur complement low-rank correction preconditioner for general sparse linear systems. In SIAM Journal on Matrix Analysis and Applications, 2021.
20. L. Zhou, Z. Luo, M. Zhen, T. Shen, S. Li, Z. Huang, T. Fang, and L. Quan. Stochastic bundle adjustment for efficient and scalable 3d reconstruction. In European Conference on Computer Vision (ECCV), 2020.
21. https://spectralib.org

Power Bundle Adjustment for Large-Scale 3D Reconstruction Appendix

In this supplementary material we provide additional details to augment the content of the main paper. Section A contains a proof of Proposition 1 in the main paper. Section B provides further details on the baseline methods (Section B.1), on the proposed PoBA algorithm (Section B.2), and on the stochastic framework STBA as well as our extension PoST (Section B.3). This includes step-by-step algorithms and for PoBA a discussion of the alternative right-factorization. Finally, in Section C we list the evaluated problems from the BAL dataset.

A Proof of Proposition 1

Firstly, simple product expansion gives

$$(I - M)(I + ... + M^i) = I - M^{i+1} . \tag{1}$$

Since the spectral norm is sub-multiplicative and

$$\|M\| < 1 , \tag{2}$$

it is straightforward to see that

$$\|M^i\| \leqslant \|M\|^i \longrightarrow 0 . \tag{3}$$

Thus,

$$M^i \longrightarrow \mathbf{0} . \tag{4}$$

Taking the limit of both sides in (1) gives (1).

Secondly,

$$R = \sum_{i=m+1}^{\infty} M^i = M^{m+1} \sum_{i=0}^{\infty} M^i = M^{m+1}(I - M)^{-1} . \tag{5}$$

It follows that

$$\|R\| \leqslant \frac{\|M\|^{m+1}}{\|I - M\|} , \tag{6}$$

and then

$$\|I - M\| \geqslant \|I\| - \|M\| = 1 - \|M\| \tag{7}$$

gives inequality (3). □

B Algorithms

B.1 LM and DL Algorithms

In the main paper we introduced the Levenberg-Marquardt (LM) algorithm for solving BA problems, but in the evaluation we also compare to the alternative DogLeg (DL) method. Since it is lesser known, we give a short description of DL in the following. Both the LM and DL algorithms combine Gauss-Newton and steepest gradient descent directions. Nevertheless, DL explicitly uses a trust-region that adds a constraint to the minimization least squares problem. More precisely, the minimization problem turns into

$$\min_{\Delta x_p, \Delta x_l} \frac{1}{2}(\|r^0 + \begin{pmatrix} J_p \; J_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} \|_2^2) \,, \tag{8}$$

such that

$$\| \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} \|_2 \leqslant \alpha \,. \tag{9}$$

The radius of the trust-region is then crucial to the success of a step. The solution to the previous equation is decomposed into two parts. The first one depends on the Cauchy point

$$\Delta x_{CP} = -\frac{b^\top b}{(Jb)^\top (Jb)} b \,, \tag{10}$$

where

$$b = \begin{pmatrix} b_p \\ b_l \end{pmatrix} \,, \tag{11}$$

$$J = \begin{pmatrix} J_p \; J_l \end{pmatrix} \,. \tag{12}$$

The second one depends on the solution of the unconstrained minimization problem, that is the solution

$$\Delta x_{GN} = (\Delta x_p^{GN}, \Delta x_l^{GN}) \tag{13}$$

of the normal equation:

$$H \Delta x = -b \,. \tag{14}$$

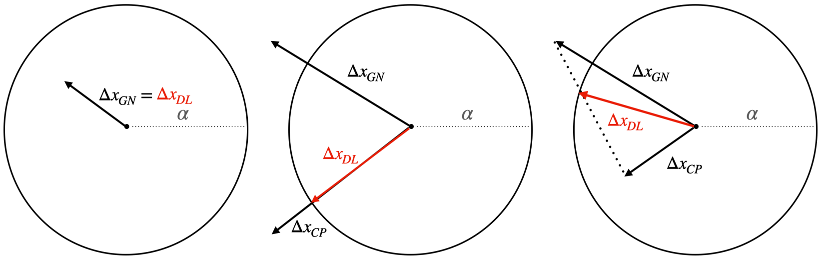
In our implementation and in accordance with [20] we first derive Δx_p^{GN} with the Schur complement trick and a sparse Cholesky factorization. Then we back-substitute Δx_p^{GN} in

$$\Delta x_l^{GN} = -V^{-1}(-b_l + W^\top \Delta x_p^{GN}) \,, \tag{15}$$

where W and b_l are defined as in the LM algorithm, and

$$V = J_l^\top J_l \,. \tag{16}$$

The effective update Δx is then:



(a) The GN step is inside the trust-region.

(b) The GN step and the Cauchy point are outside the trust-region.

(c) The GN step and the Cauchy point are respectively outside and inside the trust-region.

Fig. 1: DL algorithm solves the minimal least squares problem by introducing an explicit trust-region. The pose and landmark updates, plotted in red colour, depend on the radius α of the considered trust-region. α is then updated depending on the success or failure of the outer iteration.

- $\Delta x_{DL} = \Delta x_{GN}$ if the GN step is inside the trust-region ($\|\Delta x_{GN}\| \leqslant \alpha$);
- $\Delta x_{DL} = \frac{\alpha}{\|\Delta x_{CP}\|}\Delta x_{CP}$ if both Δx_{CP} and Δx_{GN} are outside the trust-region;
- $\Delta x_{DL} = \Delta x_{CP} + \beta(\Delta x_{GN} - \Delta x_{CP})$ with β such that $\|\Delta x_{DL}\| = \alpha$ if Δx_{CP} and Δx_{GN} are respectively inside and outside the trust-region.

The radius α is then updated depending on the success or failure of the outer iteration. Fig. 1 illustrates the three different cases for the effective update Δx_{DL}.

B.2 PoBA Algorithm

Algorithm 1 gives the step-by-step description of the linear solver of PoBA. Notice how it can be efficiently implemented by requiring only (sparse) matrix-vector products and no matrix-matrix products, inside the loop. Inversion and multiplication with U_λ is efficient, since it is block-diagonal (same as V_λ).

Right-Factorization The right-factorization of the Schur complement by U_λ

$$S = (I - WV_\lambda^{-1}W^\top U_\lambda^{-1})U_\lambda \,, \tag{17}$$

is mathematically equivalent to the left-factorization (Equation (16) in the main paper):

$$S = U_\lambda(I - U_\lambda^{-1}WV_\lambda^{-1}W^\top) \,. \tag{18}$$

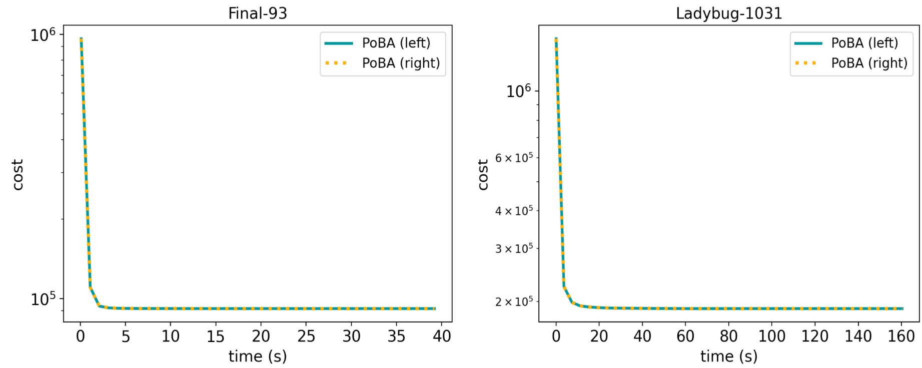
The corresponding Algorithm 2 differs from Algorithm 1 in the number of matrix-vector products. Nevertheless due to the sparsity of U_λ^{-1} the effect on the runtime is negligible. Fig. 2 shows the convergence plot with respect to runtime for two differently sized BAL problems and illustrates the equivalence of right- and left-factorization in terms of runtime and convergence.

Algorithm 1 Linear solver of PoBA (Left-factorization)

Require: b, U_λ, V_λ, W, ϵ, it_{max}
Ensure: Pose step Δx_p
$b^{init} = U_\lambda^{-1} b$
$b^{temp} = b^{init}$
$\Delta x_p = b^{init}$
$Z = W V_\lambda^{-1} W^\top$
$it = 0$
while $\|b^{temp}\|_2 \geqslant \epsilon \|b^{init}\|_2$ and $it < it_{max}$ **do**
  $b^{temp} \leftarrow U_\lambda^{-1}(Z(b^{temp}))$
  $\Delta x_p \leftarrow \Delta x_p + b^{temp}$
  $it \leftarrow it + 1$
end while

Algorithm 2 Linear solver of PoBA (Right-factorization)

Require: b, U_λ, V_λ, W, ϵ, it_{max}
Ensure: Pose step Δx_p
$b^{init} = U_\lambda^{-1} b$
$b^{temp} = b^{init}$
$b^{right} = b$
$\Delta x_p = b^{init}$
$Z = W V_\lambda^{-1} W^\top$
$it = 0$
while $\|b^{temp}\|_2 \geqslant \epsilon \|b^{init}\|_2$ and $it < it_{max}$ **do**
  $b^{right} \leftarrow Z(U_\lambda^{-1}(b^{right}))$
  $b^{temp} = U_\lambda^{-1} b^{right}$
  $\Delta x_p \leftarrow \Delta x_p + b^{temp}$
  $it \leftarrow it + 1$
end while



(a) *Final-93* (b) *Ladybug-1031*

Fig. 2: Convergence plots for (a) *Final-93* from BAL dataset with 93 poses and (b) *Ladybug-1031* from BAL dataset with 1031 poses. PoBA (left) corresponds to Algorithm 1 and PoBA (right) to Algorithm 2. They only differ by one sparse matrix-vector product per inner iteration. The difference of runtime is negligible.

B.3 PoST/STBA Algorithm

Algorithm 3 shows the full step-by-step description for the reference STBA algorithm as presented in [20] and our proposed PoST algorithm. We keep the same notation as in [20]. STBA builds the local Schur complements and then solves the associated normal equations per-cluster with a dense Cholesky factorization. PoST on the other hand uses Algorithm 1 for the local sub-problems.

Algorithm 3 STBA/PoST

Require: Visibility graph, Initial pose and point parameters
Ensure: x^* minimizing $F(x)$

```
  it_max = 50, λ = 10^-4, Γ = 100, ε = 10^-6, thres = 0.01, m_max = 50
  it = 0, stop = False
  Build camera graph G_c
  while not stop and it < it_max do
      {φ_i}_{i=1}^l = StochasticGraphClustering(G_c, Γ)   ▷ Γ is the maximum cluster size
      Build the equality constraint matrix A according to {φ_i}_{i=1}^l
      Evaluate reprojection errors f
      Evaluate Jacobians J_c, J_p, J'_p, J' = (J_c, J'_p)
      Evaluate g = −J' f
      Evaluate U_λ = {U_λ^i}_{i=1}^l, V_λ, W
      Evaluate V'_λ = J'_p^⊤ J'_p + λ D'_p^⊤ D'_p = {V_λ^i}_{i=1}^l
      Evaluate W' = J'_p^⊤ J_l = {W^i}_{i=1}^l
      if λ > 0.1 then                                   ▷ Steepest descent correction
          H_λ = J' J'^⊤ + λ D'^⊤ D'
          H̃_λ = diag(H_λ)
          ν = (A H̃_λ^-1 A^⊤)^-1 A H̃_λ^-1 b
          g = g − A^⊤ ν
      end if
      g = (v'^⊤, w'^⊤)^⊤
      b' = v' − W' V'_λ^-1 w' = {b_i}_{i=1}^l
      if STBA then
          S' = U_λ − W' V'_λ^-1 W'^⊤ = {S_i}_{i=1}^l
          for i = 1, ..., l do
              Δx_{p,i} = Cholesky(S_i, b_i)
          end for
      else if PoST then                                 ▷ Our proposed solver
          for i = 1, ..., l do
              Δx_{p,i} = PoBA(b_i, U_λ^i, V_λ^i, W^i, thres, m_max)
          end for
      end if
      Δx_p = (Δx_{p,1}, ..., Δx_{p,n_p})
      Δx_l = V_λ^-1 (w − W^⊤ Δx_p)
      Δx = (Δx_p, Δx_l)
      if Cost tolerance < ε or Gradient tolerance < ε then
          stop=True
      end if
      if F(x) > F(x + Δx) then
          λ = λ/3, x_p ← x_p + Δx_p, x_l ← x_l + Δx_l
      else
          λ ← λ * 3
      end if
      it ← it + 1
  end while
  x* = (x_p^⊤, x_l^⊤)
```

C Problems Table

	cameras	landmarks	observations
ladybug-49	49	7,766	31,812
ladybug-73	73	11,022	46,091
ladybug-138	138	19,867	85,184
ladybug-318	318	41,616	179,883
ladybug-372	372	47,410	204,434
ladybug-412	412	52,202	224,205
ladybug-460	460	56,799	241,842
ladybug-539	539	65,208	277,238
ladybug-598	598	69,193	304,108
ladybug-646	646	73,541	327,199
ladybug-707	707	78,410	349,753
ladybug-783	783	84,384	376,835
ladybug-810	810	88,754	393,557
ladybug-856	856	93,284	415,551
ladybug-885	885	97,410	434,681
ladybug-931	931	102,633	457,231
ladybug-969	969	105,759	474,396
ladybug-1064	1,064	113,589	509,982
ladybug-1118	1,118	118,316	528,693
ladybug-1152	1,152	122,200	545,584
ladybug-1197	1,197	126,257	563,496
ladybug-1235	1,235	129,562	576,045
ladybug-1266	1,266	132,521	587,701
ladybug-1340	1,340	137,003	612,344
ladybug-1469	1,469	145,116	641,383
ladybug-1514	1,514	147,235	651,217
ladybug-1587	1,587	150,760	663,019
ladybug-1642	1,642	153,735	670,999
ladybug-1695	1,695	155,621	676,317
ladybug-1723	1,723	156,410	678,421
	cameras	landmarks	observations
trafalgar-21	21	11,315	36,455
trafalgar-39	39	18,060	63,551
trafalgar-50	50	20,431	73,967
trafalgar-126	126	40,037	148,117
trafalgar-138	138	44,033	165,688
trafalgar-161	161	48,126	181,861
trafalgar-170	170	49,267	185,604
trafalgar-174	174	50,489	188,598
trafalgar-193	193	53,101	196,315

trafalgar-201	201	54,427	199,727
trafalgar-206	206	54,562	200,504
trafalgar-215	215	55,910	203,991
trafalgar-225	225	57,665	208,411
trafalgar-257	257	65,131	225,698
	cameras	landmarks	observations
dubrovnik-16	16	22,106	83,718
dubrovnik-88	88	64,298	383,937
dubrovnik-135	135	90,642	552,949
dubrovnik-142	142	93,602	565,223
dubrovnik-150	150	95,821	567,738
dubrovnik-161	161	103,832	591,343
dubrovnik-173	173	111,908	633,894
dubrovnik-182	182	116,770	668,030
dubrovnik-202	202	132,796	750,977
dubrovnik-237	237	154,414	857,656
dubrovnik-253	253	163,691	898,485
dubrovnik-262	262	169,354	919,020
dubrovnik-273	273	176,305	942,302
dubrovnik-287	287	182,023	970,624
dubrovnik-308	308	195,089	1,044,529
dubrovnik-356	356	226,729	1,254,598
	cameras	landmarks	observations
venice-52	52	64,053	347,173
venice-89	89	110,973	562,976
venice-245	245	197,919	1,087,436
venice-427	427	309,567	1,695,237
venice-744	744	542,742	3,054,949
venice-951	951	707,453	3,744,975
venice-1102	1,102	779,640	4,048,424
venice-1158	1,158	802,093	4,126,104
venice-1184	1,184	815,761	4,174,654
venice-1238	1,238	842,712	4,286,111
venice-1288	1,288	865,630	4,378,614
venice-1350	1,350	893,894	4,512,735
venice-1408	1,408	911,407	4,630,139
venice-1425	1,425	916,072	4,652,920
venice-1473	1,473	929,522	4,701,478
venice-1490	1,490	934,449	4,717,420
venice-1521	1,521	938,727	4,734,634
venice-1544	1,544	941,585	4,745,797
venice-1638	1,638	975,980	4,952,422
venice-1666	1,666	983,088	4,982,752

venice-1672	1,672	986,140	4,995,719
venice-1681	1,681	982,593	4,962,448
venice-1682	1,682	982,446	4,960,627
venice-1684	1,684	982,447	4,961,337
venice-1695	1,695	983,867	4,966,552
venice-1696	1,696	983,994	4,966,505
venice-1706	1,706	984,707	4,970,241
venice-1776	1,776	993,087	4,997,468
venice-1778	1,778	993,101	4,997,555
	cameras	landmarks	observations
final-93	93	61,203	287,451
final-394	394	100,368	534,408
final-871	871	527,480	2,785,016
final-961	961	187,103	1,692,975
final-1936	1,936	649,672	5,213,731
final-3068	3,068	310,846	1,653,045
final-4585	4,585	1,324,548	9,124,880
final-13682	13,682	4,455,575	28,973,703

Table 1: List of all 97 BAL problems including number of cameras, landmarks and observations. These are the problems evaluated for performance profiles in the main paper.