# Power Bundle Adjustment for Large-Scale 3D Reconstruction

Simon Weber[1,2] sim.weber@tum.de
Nikolaus Demmel[1,2] nikolaus.demmel@tum.de
Tin Chon Chan[1,2] tin-1254@hotmail.com
Daniel Cremers[1,2,3] cremers@tum.de

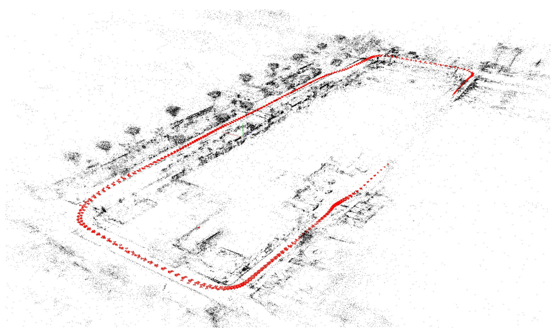
## Abstract

*We introduce Power Bundle Adjustment as an expansion type algorithm for solving large-scale bundle adjustment problems. It is based on the power series expansion of the inverse Schur complement and constitutes a new family of solvers that we call inverse expansion methods. We theoretically justify the use of power series and we prove the convergence of our approach. Using the real-world BAL dataset we show that the proposed solver challenges the state-of-the-art iterative methods and significantly accelerates the solution of the normal equation, even for reaching a very high accuracy. This easy-to-implement 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 significantly improves speed and accuracy of the distributed optimization.*

## 1. Introduction

Bundle adjustment (BA) is a classical computer vision problem that forms the core component of many 3D reconstruction and Structure from Motion (SfM) algorithms. It refers to the joint estimation of camera parameters and 3D landmark positions by minimization of a non-linear reprojection error. The recent emergence of large-scale internet photo collections [1] raises the need for BA methods that are scalable with respect to both runtime and memory. And building accurate city-scale maps for applications such as augmented reality or autonomous driving brings current BA approaches to their limits.

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 involves only the pose parameters and is significantly smaller. Its size can be reduced even more by using a QR factorization, deriving only a matrix square root of the RCS, and then solving an algebraically equivalent


(a) *Ladybug-1197*

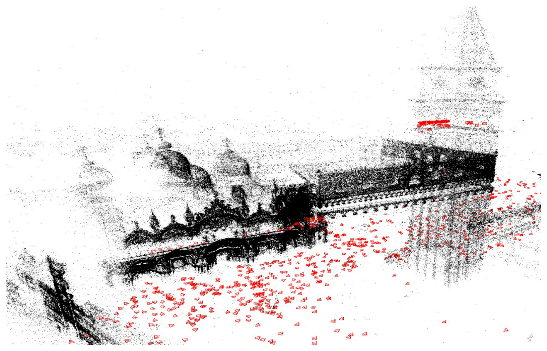

(b) *Venice-1102*

Figure 1. Power Bundle Adjustment (PoBA) is a novel solver for large-scale BA problems that is significantly faster and more memory-efficient than existing solvers. (a) Optimized 3D reconstruction of a *Ladybug* BAL problem with 1197 poses. *PoBA*-32 (resp. *PoBA*-64) is 41% (resp. 36%) faster than the best competing solver to reach a cost tolerance of 1%. (b) Optimized 3D reconstruction of a *Venice* BAL problem with 1102 poses. *PoBA*-32 (resp. *PoBA*-64) is 71% (resp. 69%) faster than the best competing solver to reach a cost tolerance of 1%. *PoBA* is five times (resp. twice) less memory-consuming than $\sqrt{BA}$ (resp. Ceres).

problem [4]. Both the RCS and its square root formulation are commonly solved by iterative methods such as the popular preconditioned conjugate gradients algorithm for large-scale problems or by direct methods such as Cholesky factorization for small-scale problems.

In the following, we will challenge these two families of solvers by relying on an iterative approximation of the inverse Schur complement. In particular, our contributions

[1]Technical University of Munich
[2]Munich Center for Machine Learning
[3]University of Oxford

are as follows:

- We introduce Power Bundle Adjustment (*PoBA*) for efficient large-scale BA. This new family of techniques that we call *inverse expansion methods* challenges the state-of-the-art methods which are built on iterative and direct solvers.
- We link the bundle adjustment problem to the theory of power series and we provide theoretical proofs that justify this expansion and establish the convergence of our solver.
- We perform extensive evaluation of the proposed approach on the BAL dataset and compare to several state-of-the-art solvers. We highlight the benefits of *PoBA* in terms of speed, accuracy, and memory-consumption. Figure 1 shows reconstructions for 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: `https://github.com/simonwebertum/poba`

## 2. Related Work

Since we propose a new way to solve large-scale bundle adjustment problems, we will review works on bundle adjustment and on traditional solving methods, that is, direct and iterative methods. We also provide some background on power series. For a general introduction to series expansion we refer the reader to [14].

### Scalable bundle adjustment.

A detailed survey of bundle adjustment can be found in [16]. The Schur complement [20] 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 [15] are outperformed by iterative methods such as inexact Newton algorithms. Large-scale bundle adjustment problems with tens of thousands of images are typically solved by the conjugate gradient method [1, 2, 8]. Some variants have been designed, for instance the search-space can be enlarged [17] or a visibility-based preconditioner can be used [9]. A recent line of works on square root bundle adjustment proposes to replace the Schur complement for eliminating landmarks with nullspace projection [4, 5]. It leads to significant performance improvements and to one of the most performant solver for the bundle adjustment problem in term of speed and accuracy. Nevertheless these methods still rely on traditional solvers for the reduced camera system, i.e. preconditioned conjugate gradient method (PCG) for large-scale [4] and Cholesky decomposition for small-scale [5] problems, besides an important cost in term of memory-consumption. Even with PCG, solving the normal equation remains the bottleneck and finding thousands of unknown parameters requires a large number of inner iterations. Other authors try to improve the runtime of BA with PCG by focusing on efficient parallelization [13]. Recently, Stochastic BA [22] was introduced to stochastically decompose the reduced camera system into subproblems and solve the smaller normal equation by dense factorization. This leads to a distributed optimization framework with improved speed and scalability. By encapsulating the general power series theory into a linear solver we propose to simultaneously improve the speed, the accuracy and the memory-consumption of these existing methods.

### Power series solver.

While power series expansion is common to solve differential equations [3], to the best of our knowledge it has never been employed for solving the bundle adjustment problem. A recent work [21] links the Schur complement to Neumann polynomial expansion to build a new preconditioner. Although this method presents interesting results for some physics problems such as convection-diffusion or atmospheric equations, it remains unsatisfactory for the bundle adjustment problem (see Figure 2). In contrast, we propose to directly apply the power series expansion of the inverse Schur complement for solving the BA problem. Our solver therefore falls in the category of expansion methods that – to our knowledge – have never been applied to the BA problem. In addition to being an easy-to-implement solver it leverages the special structure of the BA problem to simultaneously improve the trade-off speed-accuracy and the memory-consumption of the existing methods.

## 3. Power Series

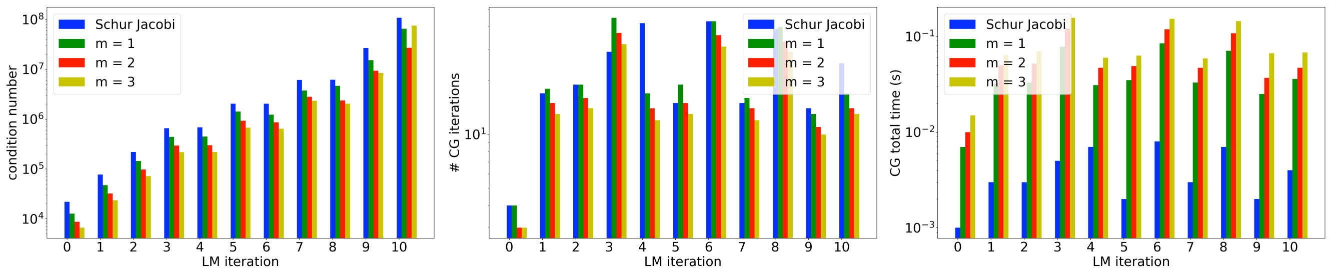
We briefly introduce power series expansion of a matrix. Let $\rho(A)$ denote the spectral radius of a square matrix $A$, i.e. the largest absolute eigenvalue and denote the spectral norm by $\|A\| = \rho(A)$. The following proposition holds:

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

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

*where the error matrix*

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



(a) Condition number (b) Number of CG iterations (c) Total runtime of the CG algorithm

Figure 2. Although [21] explores the use of power series as a preconditioner for some physics problems it suffers from the special structure of the BA formulation. Given a preconditioner $M^{-1}$ and the Schur complement $S$, the condition number $\kappa(M^{-1}S)$ is linked to the convergence of the conjugate gradients algorithm. (a) illustrates the behaviour of $\kappa$ for the ten first iterations of the LM algorithm for the real problem Ladybug-49 with 49 poses from BAL dataset and for different orders $m$ of the power series expansion (22) used as preconditioner for the CG algorithm. The condition number associated to the popular Schur-Jacobi preconditioner is reduced with this power series preconditioner, that is illustrated by a better convergence of the CG algorithm and then a smaller number of CG iterations (b). Nevertheless each supplementary order $m$ is more costly in terms of runtime as the application of the power series preconditioner involves $4m$ matrix-vector product, whereas the Schur-Jacobi preconditioner can be efficiently stored and applied. (c) It leads to an increase of the overall runtime when solving the normal equation (6).

*satisfies*

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

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

## 4. Power Bundle Adjustment

We consider a 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, where the vector $x_p$ of length $d_p n_p$ is associated to the extrinsic and (possibly) intrinsic camera parameters for all poses and the vector $x_l$ of length $3n_l$ is associated to the 3D coordinates of all landmarks. In case only the extrinsic parameters are unknown then $d_p = 6$ for rotation and translation of each camera. 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 the vector $r(x) = [r_1(x)^\top, ..., r_k(x)^\top]^\top$ comprises all residuals capturing the discrepancy between model and observation.

### 4.1. Least Squares Problem

This nonlinear least squares 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

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

with $r^0 = r(x^0)$, $J_p = \frac{\partial r}{\partial x_p}|_{x^0}$, $J_l = \frac{\partial r}{\partial x_l}|_{x^0}$, $\lambda$ 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

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

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

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

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

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

$U_\lambda$, $V_\lambda$ and $H$ are symmetric positive-definite [16].

### 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 $\Delta x_p$. The optimal $\Delta x_l$ is obtained by back-substitution:

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

### 4.3. Power Bundle Adjustment

Factorizing (13) with the block-matrix $U_\lambda$

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

leads to formulate the inverse Schur complement as

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

In order to expand (17) into a power series as detailed in Proposition 1, we require to bound the spectral radius of $U_\lambda^{-1} W V_\lambda^{-1} W^\top$ by 1.

By leveraging the special structure of the BA problem we prove an even stronger result:

**Lemma 1.** *Let $\mu$ be an eigenvalue of $U_\lambda^{-1} W V_\lambda^{-1} W^\top$. Then*

$$\mu \in [0, 1[\,. \tag{18}$$

*Proof.* On the one hand $U_\lambda^{-\frac{1}{2}} W V_\lambda^{-1} W^\top U_\lambda^{-\frac{1}{2}}$ is symmetric positive semi-definite, as $U_\lambda$ and $V_\lambda$ are symmetric positive definite. Then its eigenvalues are greater than 0. As $U_\lambda^{-\frac{1}{2}} W V_\lambda^{-1} W^\top U_\lambda^{-\frac{1}{2}}$ and $U_\lambda^{-1} W V_\lambda^{-1} W^\top$ are similar,

$$\mu \geq 0\,. \tag{19}$$

On the other hand $U_\lambda^{-\frac{1}{2}} S U_\lambda^{-\frac{1}{2}}$ is symmetric positive definite as $S$ and $U_\lambda$ are. It follows that the eigenvalues of $U_\lambda^{-1} S$ are all strictly positive due to its similarity with $U_\lambda^{-\frac{1}{2}} S U_\lambda^{-\frac{1}{2}}$. As

$$U_\lambda^{-1} W V_\lambda^{-1} W^\top = I - U_\lambda^{-1} S\,, \tag{20}$$

it follows that

$$\mu < 1\,, \tag{21}$$

that concludes the proof. □

Let be

$$\tilde{S}_{-1}(m) = \sum_{i=0}^{m} (U_\lambda^{-1} W V_\lambda^{-1} W^\top)^i U_\lambda^{-1}\,, \tag{22}$$

and

$$x(m) = -\tilde{S}_{-1}(m)\tilde{b}\,, \tag{23}$$

for $m \geq 0$. The following proposition confirms that the approximation indeed converges with increasing order of $m$:

**Proposition 2.** $\|x(m) - \Delta x_p\|_2 \underset{m \to +\infty}{\longrightarrow} 0\,.$

*Proof.* We denote $P = U_\lambda^{-1} W V_\lambda^{-1} W^\top$. Due to Lemma 1

$$\|P\| < 1\,. \tag{24}$$

The inverse Schur complement associated to (6) admits a power series expansion:

$$S^{-1} = \tilde{S}_{-1}(m) + R_m\,, \tag{25}$$

where

$$R_m = \sum_{i=m+1}^{\infty} P^i U_\lambda^{-1} \tag{26}$$

satisfies

$$\|R_m\| \leq \frac{\|P\|^{m+1}}{1 - \|P\|} \|U_\lambda^{-1}\|\,. \tag{27}$$

It follows that:

$$x(m) - \Delta x_p = R_m \tilde{b}\,. \tag{28}$$

The consistency of the spectral norm with respect to the vector norm implies:

$$\|R_m \tilde{b}\|_2 \leq \|R_m\| \|\tilde{b}\|_2\,. \tag{29}$$

From (24), (27) and (29) we conclude the proof:

$$\|R_m \tilde{b}\|_2 \underset{m \to +\infty}{\longrightarrow} 0\,, \tag{30}$$

and then

$$\|x(m) - \Delta x_p\|_2 \underset{m \to +\infty}{\longrightarrow} 0\,. \tag{31}$$

□

This convergence result proves that

- an approximation of $\Delta x_p$ can be directly obtained by applying (22) to the right-hand side of (12);
- the quality of this approximation depends on the order $m$ and can be as small as desired.

The power series expansion being iteratively derived, a termination rule is necessary.

By analogy with inexact Newton methods [11, 12, 18] such that the conjugate gradients algorithm we set a stop criterion

$$(i+1) * \|(x(i) - x(i-1))\|_2 / \|x(i)\|_2 < \epsilon\,, \tag{32}$$

for a given $\epsilon$. This criterion ensures that the power series expansion stops when the refinement of the pose update by expanding the inverse Schur complement into a supplementary order

$$\|(x(i) - x(i-1))\|_2 \tag{33}$$

is much smaller than the average refinement when reaching the same order

$$\frac{\|\sum_{j=1}^{i} (x(j) - x(j-1)) + x(0)\|_2}{i+1} = \frac{\|x(i)\|_2}{i+1}\,. \tag{34}$$

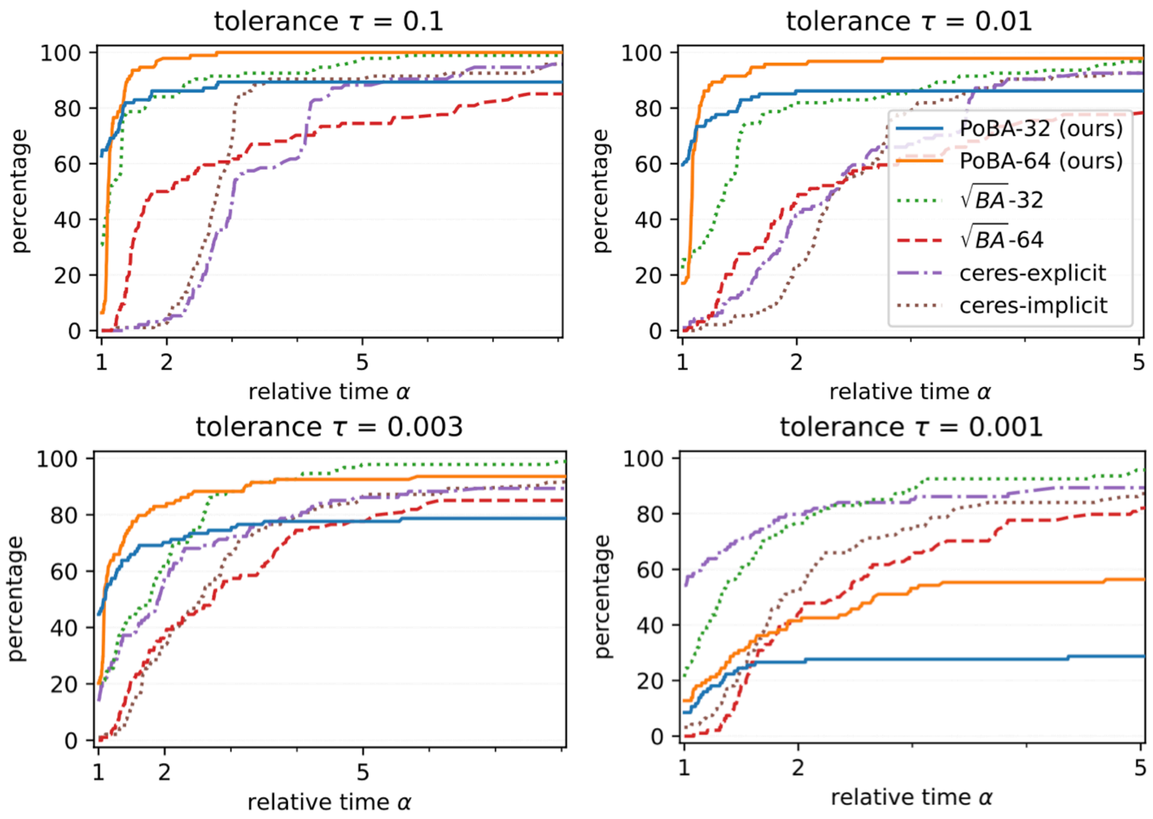


Figure 3. Performance profiles for all BAL problems show the percentage of problems solved to a given accuracy tolerance $\tau \in \{0.1, 0.01, 0.003, 0.001\}$ with relative runtime $\alpha$. Our proposed solver *PoBA* using series expansion of the Schur complement significantly outperforms all the competing solvers up to the high accuracy $\tau = 0.003$.

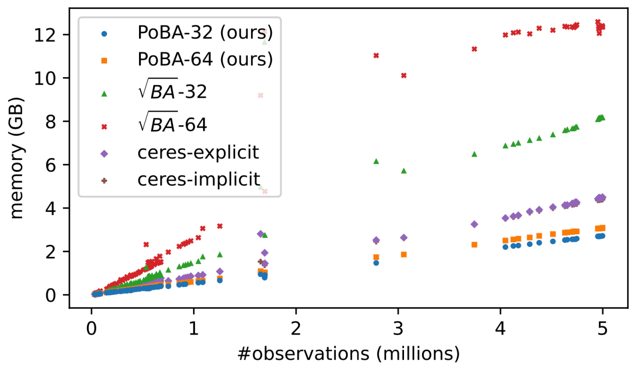
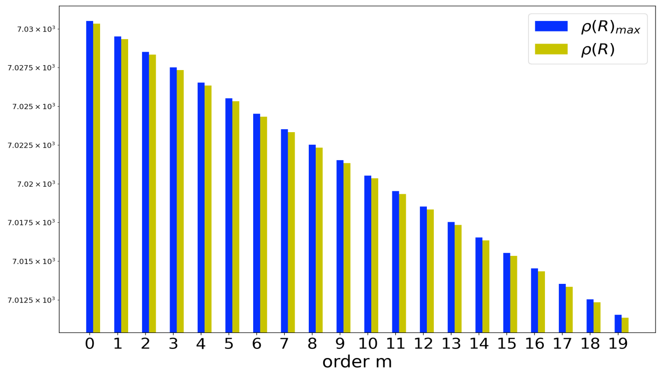


Figure 4. Memory consumption for all BAL problems. The proposed *PoBA* solver (orange and blue points) is five times less memory-consuming than $\sqrt{BA}$ solvers.

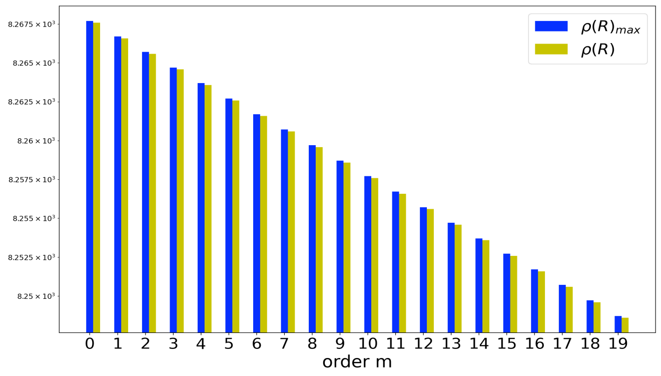
## 5. Implementation

We implement our *PoBA* solver in C++ in single (*PoBA*-32) and double (*PoBA*-64) floating-point precision, directly on the publicly available implementation[1] of [4]. This recent solver presents excellent performance to solve the bundle adjustment by using a QR factorization of the landmark Jacobians. It notably competes the popular Ceres solver. We additionally add a comparison with Ceres' sparse Schur complement solvers, similarly as in [4]. *Ceres-explicit* and *Ceres-implicit* iteratively solve (12) with the conjugate gradients algorithm preconditioned by the Schur-Jacobi preconditioner. The first one saves $S$ in memory as a block-sparse matrix, the second one computes $S$ on-the-fly during iterations. $\sqrt{BA}$ and Ceres offer very competitive performance to solve the bundle adjustment problem, that makes them very challenging baselines to compare *PoBA* to. We run experiments on MacOS 11.2 with an Intel Core i5 and 4 cores at 2GHz.

[1] https://github.com/NikolausDemmel/rootba



(a) *Ladybug-49*



(b) *Trafalgar-193*

Figure 5. Illustration of the inequality (3) in Proposition 1 for the first LM iteration of two BAL problems: (a) *Ladybug* with 49 poses and (b) *Trafalgar* with 193 poses. The spectral norm of the error matrix $R$ is plotted in green for $m < 20$. The right-side of the inequality plotted in blue 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 [23] $\rho(M)$ takes the values (a) 0.999858 for *L-49* and (b) 0.999879 for *T-193*. Both values are smaller than 1 and $\rho(R)$ is always smaller than $\rho(M)^{m+1}/(1-\rho(M))$, as stated in Lemma 1.

**Efficient storage.**

We leverage the special structure of BA problem and design a memory-efficient storage. We group the Jacobian matrices and residuals by landmarks and store them in separate dense memory blocks. For a landmark with $k$ observations, all pose Jacobian blocks of size $2 \times d_p$ that correspond to the poses where the landmark was observed, are stacked and stored in a memory block of size $2k \times d_p$. Together with the landmark Jacobian block of size $2k \times 3$ and the residuals of length $2k$ that are also associated to the landmark, all information of a single landmark is efficiently stored in a memory block of size $2k \times (d_p + 4)$. Furthermore, operations involved in (15) and (23) are parallelized using the memory blocks.

**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 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

$$f_\tau(p) = f^*(p) + \tau(f^0(p) - f^*(p)) \tag{35}$$

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 $\alpha$ is defined as

$$\rho(s,\alpha) = \frac{100}{|P|} |\{p \in P | T_\tau(p,s) \leq \alpha \min_{s \in S} T_\tau(p,s)\}| \tag{36}$$

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

## 5.1. 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. *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 [4] and with Ceres. Starting with damping parameter $10^{-4}$ we update $\lambda$ 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. Concerning (23) and (32) we set the maximal number of inner iterations to 20 and a threshold $\epsilon = 0.01$. Ceres and $\sqrt{BA}$ use same forcing sequence for the inner CG loop, where the maximal number of iterations is set to 500. We add a small Gaussian noise to disturb initial landmark and camera positions.

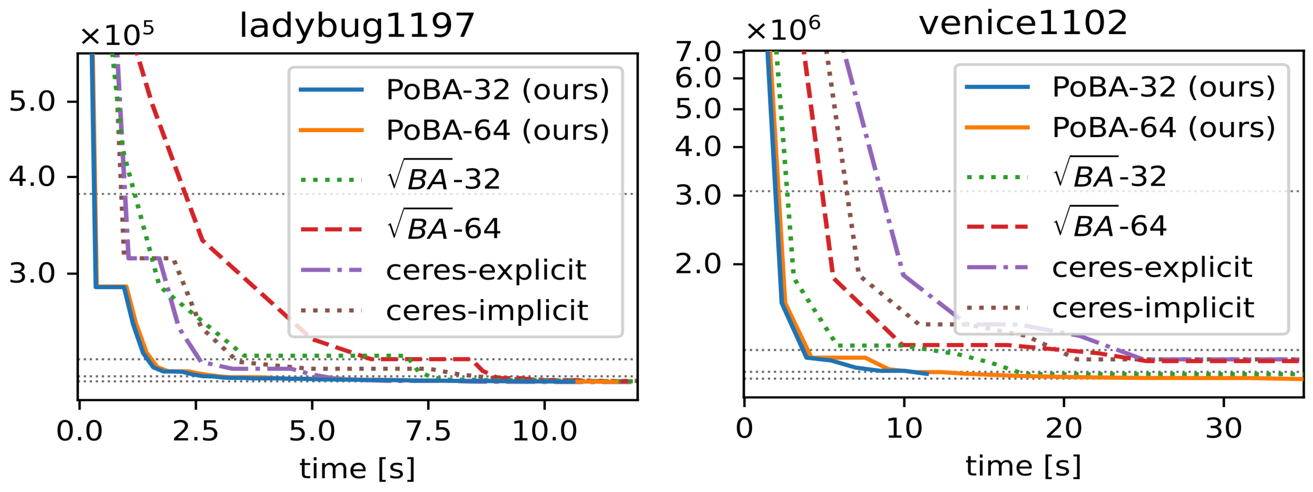


Figure 6. Convergence plots of *Ladybug-1197* (left) from BAL dataset with 1197 poses and *Venice-1102* (right) from BAL dataset with 1102 poses. Fig. 1 shows a visualization of 3D landmarks and camera poses for these problems. The dotted lines correspond to cost thresholds for the tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$.

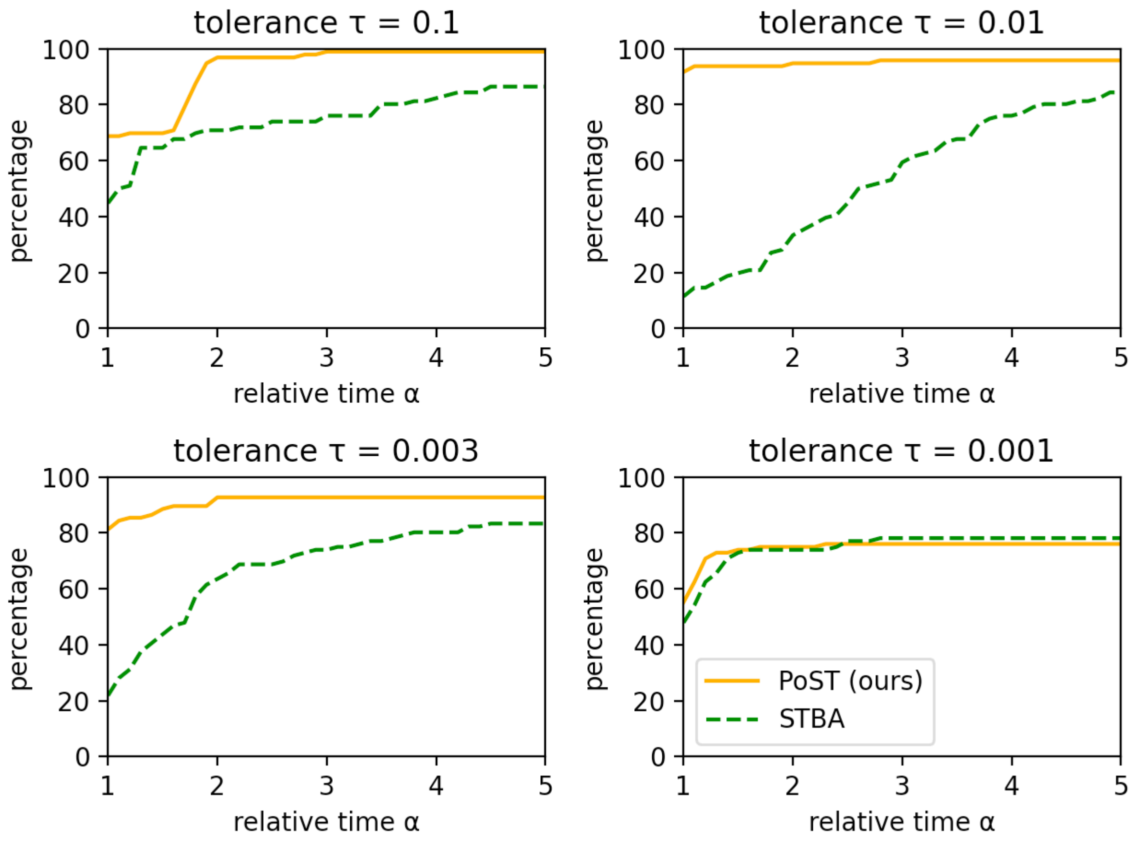


Figure 7. Performance profiles for all BAL problems with stochastic framework. Our proposed solver PoST outperforms the challenging STBA across all accuracy tolerances $\tau \in \{0.1, 0.01, 0.003\}$, both in terms of speed and precision, and rivals STBA for $\tau = 0.001$.

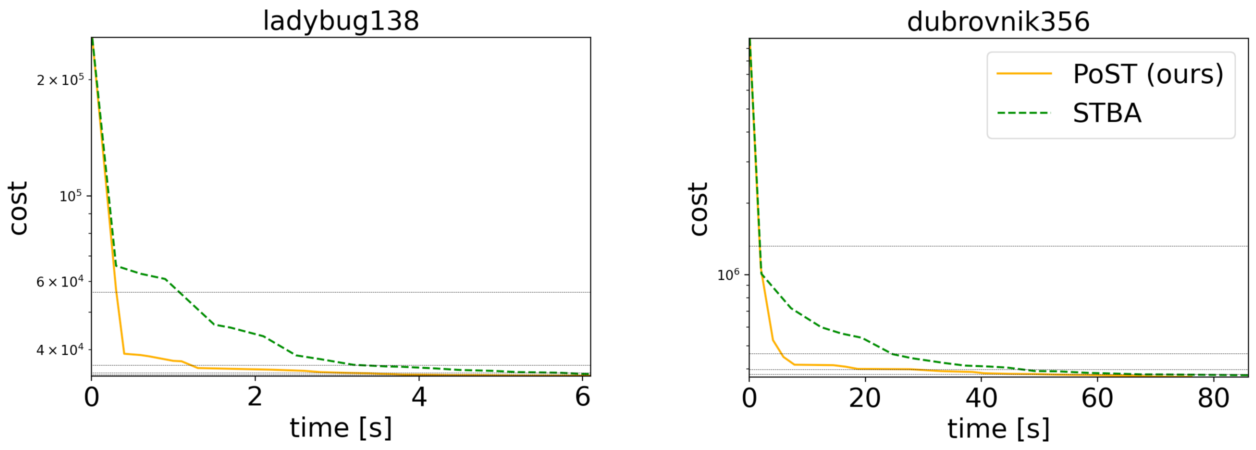


Figure 8. Convergence plots of *Ladybug-138* (left) from BAL dataset with 138 poses and *Dubrovnik-356* (right) from BAL dataset with 356 poses. The dotted lines correspond to cost thresholds for the tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$.

## 5.2. Analysis

Figure 3 shows the performance profiles for all BAL datasets with tolerances $\tau \in \{0.1, 0.01, 0.003, 0.001\}$. For $\tau = 0.1$ and $\tau = 0.01$ *PoBA*-64 clearly outperforms all challengers both in terms of runtime and accuracy. *PoBA*-64 remains clearly the best solver for the excellent accuracy $\tau = 0.003$ until a high relative time $\alpha = 4$. For higher relative time it is competitive with $\sqrt{BA} - 32$ and still outperforms all other challengers. Same conclusion can be drawn from the convergence plot of two differently sized BAL problems (see Figure 6). Figure 4 highlights the low memory consumption of *PoBA* with respect to its challengers for all BAL problems. Whatever the size of the problem *PoBA* is much less memory-consuming than $\sqrt{BA}$ and Ceres. Notably it requires almost five times less memory than $\sqrt{BA}$ and almost twice less memory than Ceres-implicit and Ceres-explicit.

## 5.3. Power Stochastic Bundle Adjustment (PoST)

**Stochastic Bundle Adjustment.**

STBA decomposes the reduced camera system into clusters inside the Levenberg-Marquardt iterations. The per-cluster linear sub-problems are then solved in parallel with dense $LL^\top$ factorization due to the dense connectivity inside camera clusters. As shown in [22] 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. In the following we show that replacing the subproblem solver with our Power Bundle Adjustment can significantly boost runtime even further.

We extend STBA[2] 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 with the same parameters as in Section 5.1. In accordance to [22] we set the maximal cluster size to 100 and the implementation is written in double in C++.

[2]https://github.com/zlthinker/STBA

**Analysis.**

Figure 7 presents the performance profiles with all BAL problems for different tolerances $\tau$. Both solvers have similar accuracy for $\tau = 0.001$. For $\tau \in \{0.1, 0.01, 0.003\}$, PoST clearly outperforms STBA both in terms of runtime and accuracy, most notably for $\tau = 0.01$. Same observations are done when we plot the convergence for differently sized BAL problems (see Figure 8).

## 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 theoretical validity of the proposed approximation and the convergence of this solver. Moreover, we experimentally confirm that the proposed power series representation of the inverse Schur complement outperforms competitive iterative solvers in terms of speed, accuracy, and memory-consumption. Last but not least, we show that the power series representation can complement distributed bundle adjustment methods to significantly boost its performance for large-scale 3D reconstruction.

## Acknowledgement

This work was supported by the ERC Advanced Grant SIMULACRON, the Munich Center for Machine Learning, the EPSRC Programme Grant VisualAI EP/T028572/1, and the DFG projects WU 959/1-1 and CR 250 20-1 “Splitting Methods for 3D Reconstruction and SLAM”.

## 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. 1, 2, 6

[2] M. Byröd, K. Åström. Conjugate gradient bundle adjustment. In *European Conference on Computer Vision (ECCV)*, 2010. 2

[3] E. A. Coddington, N. Levinson. Theory of Ordinary Differential Equations. McGraw–Hill, 1955. 2

[4] N. Demmel, C. Sommer, D. Cremers, V. Usenko. Square Root Bundle Adjustment for Large-Scale Reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 5, 6

[5] N. Demmel, D. Schubert, C. Sommer, D. Cremers, V. Usenko. Square Root Marginalization for Sliding-Window Bundle Adjustment. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[6] E. D. Dolan, and J. J. More. Benchmarking optimization software with performance profiles. In *Mathematical programming* 91(2), pages 201–213, 2002. 6

[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. 2

[9] A. Kushal, and S. Agarwal. Visibility based preconditioning for bundle adjustment. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 2

[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] S. G. Nash, A Survey of Truncated Newton Methods, Journal of Computational and Applied Mathematics, 124(1-2), 45-59, 2000. 4

[12] S. G. Nash, A. Sofer, Assessing A Search Direction Within A Truncated Newton Method, Operation Research Letters 9(1990) 219-221. 4

[13] J. Ren, W. Liang, R. Yan, L. Mai, X. Liu . MegBA: A High-Performance and Distributed Library for Large-Scale Bundle Adjustment. In *European Conference on Computer Vision (ECCV)*, 2022. 2

[14] Y. Saad. Itervative methods for sparse linear systems, 2nd ed. In *SIAM*, Philadelpha, PA, 2003. 2

[15] L. Trefethen, D. Bau. Numerical linear algebra. SIAM, 1997. 2

[16] 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. 2, 3

[17] 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. 2

[18] 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. 4

[19] C. Zach. Robust bundle adjustment revisited. In *European Conference on Computer Vision (ECCV)*, 2014.

[20] F. Zhang. The Schur complement and its applications. In Numerical Methods and Algorithms. Vol. 4, Springer, 2005. 2

[21] 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. 2, 3

[22] 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. 2, 8

[23] https://spectralib.org 6

# 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. In Section B we evaluate different levels of noises to highlight the consistence of our solver. In Section C we tabulate the percentage of solved problems of the performance profiles (Sec. 5.2.) for each tolerance $\tau \in \{0.1, 0.01, 0.03, 0.001\}$ and for each solver. In Section D we list the evaluated problems from the BAL dataset.

## A. Proof of Proposition 1

Firstly, simple product expansion gives

$$(I - M)(I + ... + M^i) = I - M^{i+1}\,. \tag{37}$$

Since the spectral norm is sub-multiplicative and

$$\|M\| < 1\,, \tag{38}$$

it is straightforward that

$$\|M^i\| \leq \|M\|^i \underset{i\to\infty}{\longrightarrow} 0\,. \tag{39}$$

Thus,

$$M^i \underset{i\to\infty}{\longrightarrow} \mathbf{0}\,. \tag{40}$$

Taking the limit of both sides in (37) gives (1).

Secondly,

$$R = \sum_{i=m+1}^{\infty} M^i = M^{m+1} \sum_{i=0}^{\infty} M^i = M^{m+1}(I - M)^{-1}\,. \tag{41}$$

It follows that

$$\|R\| = \|M^{m+1} \sum_{i=0}^{\infty} M^i\| \leq \|M\|^{m+1} \sum_{i=0}^{\infty} \|M\|^i\,. \tag{42}$$

Since $\|M\| < 1$ we have

$$\sum_{i=0}^{\infty} \|M\|^i = \frac{1}{1 - \|M\|}\,, \tag{43}$$

which directly leads to the inequality

$$\|R\| \leq \frac{\|M\|^{m+1}}{1 - \|M\|}\,. \tag{44}$$

## B. Consistence

In Sec. 5.2. initial landmark and camera positions are perturbated with a small Gaussian noise $(m, \sigma) = (0, 0.01)$. We observe that the relative performance of solvers is similar for different noise levels. Fig. 9 and 10 illustrate the consistence of our results with different initial noises $\sigma = 0.05$ and $\sigma = 0.1$.

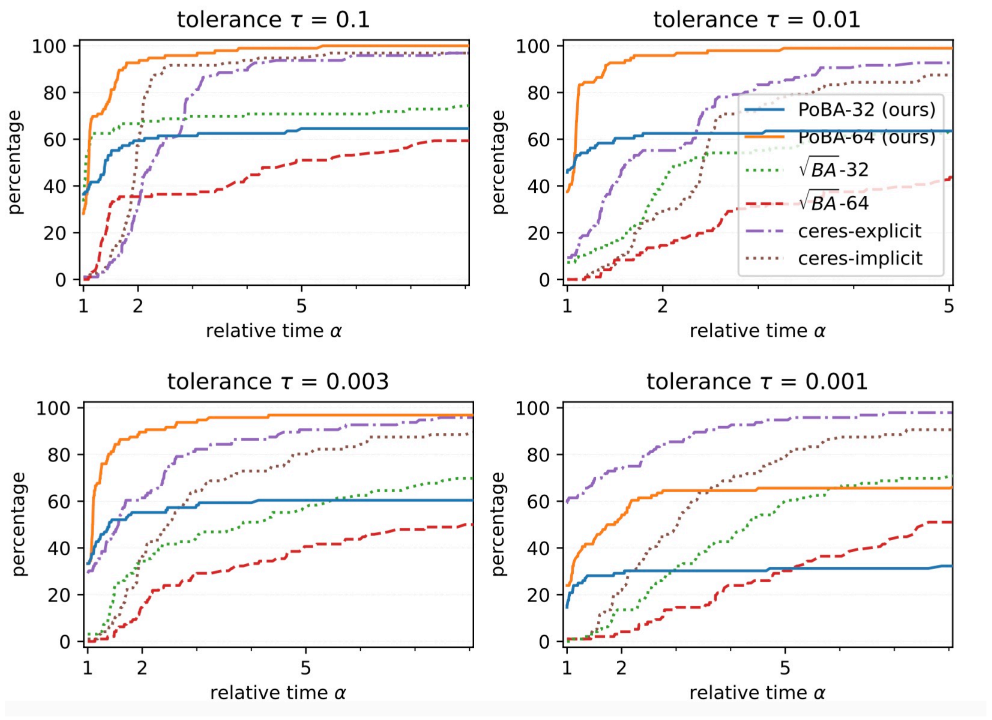


Figure 9. Performance profiles for all BAL problems show the percentage of problems solved to a given accuracy tolerance $\tau \in \{0.1, 0.01, 0.003, 0.001\}$ with relative runtime $\alpha$. Initial landmark and camera positions are disturbed with a Gaussian noise $(0, 0.05)$.

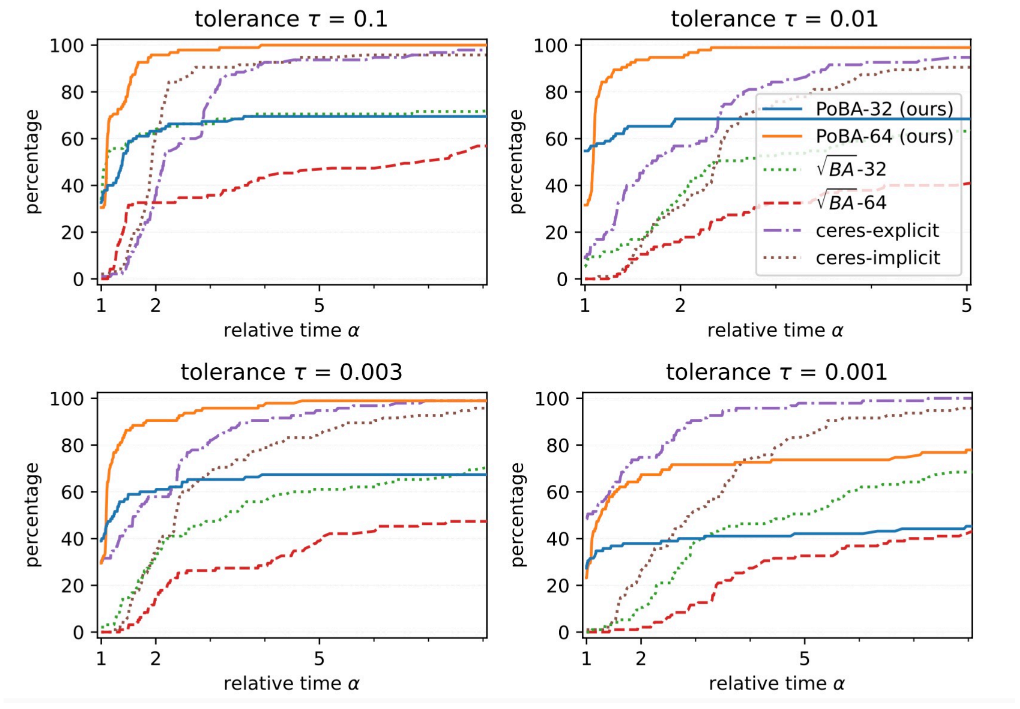


Figure 10. Performance profiles for all BAL problems show the percentage of problems solved to a given accuracy tolerance $\tau \in \{0.1, 0.01, 0.003, 0.001\}$ with relative runtime $\alpha$. Initial landmark and camera positions are disturbed with a Gaussian noise $(0, 0.1)$.

## C. Tables of solved problems associated to the performance profiles

| Solver | $\alpha = 1$ | $\alpha = 3$ | $\alpha = \infty$ |
|---|---|---|---|
| PoBA-64 (ours) | 7% | <u>100%</u> | <u>100%</u> |
| PoBA-32 (ours) | <u>62%</u> | 88% | 88% |
| $\sqrt{BA}$-64 | 0% | 60% | 81% |
| $\sqrt{BA}$-32 | 31% | 90% | 98% |
| ceres-explicit | 0% | 49% | 95% |
| ceres-implicit | 0% | 81% | 95% |

| Solver | $\alpha = 1$ | $\alpha = 3$ | $\alpha = \infty$ |
|---|---|---|---|
| PoBA-64 (ours) | 18% | <u>98%</u> | <u>98%</u> |
| PoBA-32 (ours) | <u>60%</u> | 84% | 84% |
| $\sqrt{BA}$-64 | 0% | 62% | 79% |
| $\sqrt{BA}$-32 | 22% | 83% | 97% |
| ceres-explicit | 2% | 66% | 90% |
| ceres-implicit | 0% | 80% | 90% |

| Solver | $\alpha = 1$ | $\alpha = 3$ | $\alpha = \infty$ |
|---|---|---|---|
| PoBA-64 (ours) | 20% | <u>90%</u> | 93% |
| PoBA-32 (ours) | <u>44%</u> | 75% | 79% |
| $\sqrt{BA}$-64 | 0% | 58% | 84% |
| $\sqrt{BA}$-32 | 22% | <u>90%</u> | <u>98%</u> |
| ceres-explicit | 14% | 71% | 90% |
| ceres-implicit | 0% | 66% | 91% |

| Solver | $\alpha = 1$ | $\alpha = 3$ | $\alpha = \infty$ |
|---|---|---|---|
| PoBA-64 (ours) | 13% | 52% | 58% |
| PoBA-32 (ours) | 8% | 26% | 27% |
| $\sqrt{BA}$-64 | 0% | 63% | 85% |
| $\sqrt{BA}$-32 | 21% | <u>88%</u> | <u>98%</u> |
| ceres-explicit | <u>54%</u> | 83% | 90% |
| ceres-implicit | 4% | 75% | 90% |

Table 1. Percentage of solved problems of the performance profiles (Sec. 5.2.) for each solver and for tolerance $\tau = 0.1$ (upper left), $\tau = 0.01$ (upper right), $\tau = 0.003$ (lower left) and $\tau = 0.001$ (lower right). We conclude that PoBA is particularly well suited for very fast/low-accurate ($\tau = 0.1$), fast/medium-accurate ($\tau = 0.01$) and slow/high-accurate ($\tau = 0.003$) applications.

## D. 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 |

| | cameras | landmarks | observations |
|---|---:|---:|---:|
| 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 2. 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.