Multidirectional Conjugate Gradients for Scalable Bundle Adjustment

Simon Weber, Nikolaus Demmel, and Daniel Cremers

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

Abstract. We revisit the problem of large-scale bundle adjustment and propose a technique called Multidirectional Conjugate Gradients that accelerates the solution of the normal equation by up to 61%. The key idea is that we enlarge the search space of classical preconditioned conjugate gradients to include multiple search directions. As a consequence, the resulting algorithm requires fewer iterations, leading to a significant speedup of large-scale reconstruction, in particular for denser problems where traditional approaches notoriously struggle. We provide a number of experimental ablation studies revealing the robustness to variations in the hyper-parameters and the speedup as a function of problem density.

Keywords: Large-scale reconstruction · bundle adjustment · preconditioned conjugate gradients.

1 Introduction

The classical challenge of image-based large scale reconstruction is witnessing renewed interest with the emergence of large-scale internet photo collections [2]. The computational bottleneck of 3D reconstruction and structure from motion methods is the problem of large-scale bundle adjustment (BA): Given a set of measured image feature locations and correspondences, BA aims to jointly estimate the 3D landmark positions and camera parameters by minimizing a non-linear least squares reprojection error. More specifically, the most time-consuming step is the solution of the normal equation in the popular Levenberg-Marquardt (LM) algorithm that is typically solved by Preconditioned Conjugate Gradients (PCG).

In this paper, we propose a new iterative solver for the normal equation that relies on the decomposable structure of the competitive block Jacobi preconditioner. Inspired by respective approaches in the domain-decomposition literature, we exploit the specificities of the Schur complement matrix to enlarge the search-space of the traditional PCG approach leading to what we call *Multidirectional Conjugate Gradients* (MCG). In particular our contributions are as follows:

- We design an extension of the popular PCG by using local contributions of the poses to augment the space in which a solution is sought for.
- We experimentally demonstrate the robustness of MCG with respect to the relevant hyper-parameters.

(a) *Final-1936* from BAL dataset   (b) *Alamo* from 1dSfM dataset
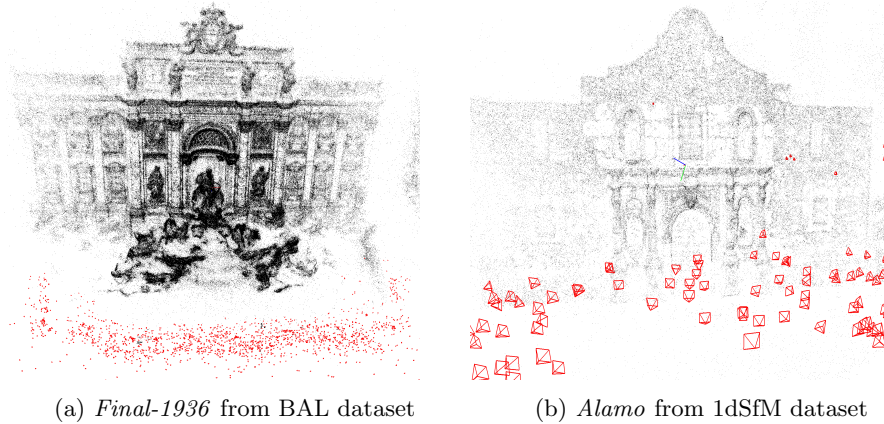
Fig. 1: (a) Optimized 3D reconstruction of a *final* BAL dataset with 1936 poses and more than five million observations. For this problem MCG is 39% faster than PCG and the overall BA resolution is 16% faster. (b) Optimized 3D reconstruction of Alamo dataset from 1dSfM with 571 poses and 900000 observations. For this problem MCG is 56% faster than PCG and the overall BA resolution is 22% faster.

- We evaluate MCG on a multitude of BA problems from BAL [1] and 1dSfM [20] datasets with different sizes and show that it is a promising alternative to PCG.
- We experimentally confirm that the performance gain of our method increases with the density of the Schur complement matrix leading to a speedup for solving the normal equation of up to 61%.

2   Related Work

Since we propose a way to solve medium to large-scale BA using a new iterative solver that enlarges the search-space of the traditional PCG, in the following we will review both scalable BA and recent CG literature.

Scalable bundle adjustment

A detailed survey of the theory and methods in BA literature can be found in [17]. Sparsity of the BA problem is commonly exploited with the Schur complement matrix [5]. As the performance of BA methods is closely linked to the resolution of the normal equations, speed up the solve step is a challenging task. Traditional direct solvers such as sparse or dense Cholesky factorization [11] have been outperformed by inexact solvers as the problem size increases and are therefore frequently replaced by Conjugate Gradients (CG) based methods

[1,6,18]. As its convergence rate depends on the condition number of the linear system a preconditioner is used to correct ill-conditioned BA problems [14]. Several works tackle the design of performant preconditioners for BA: [9] proposed the band block diagonals of the Schur complement matrix, [10] exploited the strength of the coupling between two poses to construct cluster-Jacobi and block-tridiagonal preconditioners, [7] built on the combinatorial structure of BA. However, despite these advances in the design of preconditioners, the iterative solver itself has rarely been challenged.

(Multi-preconditioned) conjugate gradients

Although CG has been a popular iterative solver for decades [8] there exist some interesting recent innovations, e.g. flexible methods with a preconditioner that changes throughout the iteration [13]. The case of a preconditioner that can be decomposed into a sum of preconditioners has been exploited by using Multi-Preconditioned Conjugate Gradients (MPCG) [4]. Unfortunately, with increasing system size MPCG rapidly becomes inefficient. As a remedy, Adaptive Multi-Preconditioned Conjugate Gradients have recently been proposed [15,3]. This approach is particularly well adapted for domain-decomposable problems [12]. While decomposition of the reduced camera system in BA has already been tackled e.g. with stochastic clusters in [19], to our knowledge the decomposition *inside* the iterative solver has never been explored. As we will show in the following, this modification gives rise to a significant boost in performance.

3 Bundle Adjustment and Multidirectional Conjugate Gradients

We consider the general form of bundle adjustment with n_p poses and n_l landmarks. Let x be the state vector containing all the optimization variables. It is divided into a pose part x_p of length $d_p n_p$ containing extrinsic and eventually intrinsic camera parameters for all poses (generally $d_p = 6$ if only extrinsic parameters are unknown and $d_p = 9$ if intrinsic parameters also need to be estimated) and a landmark part x_l of length $3n_l$ containing the 3D coordinates of all landmarks. Let $r(x) = [r_1(x), ..., r_k(x)]$ be the vector of residuals for a 3D reconstruction. The objective is to minimize the sum of squared residuals

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

3.1 Least Squares Problem and Schur Complement

This minimization problem is usually solved with the Levenberg Marquardt algorithm, which is based on the first-order Taylor approximation of $r(x)$ around the current state estimate $x^0 = \left(x_p^0, x_l^0\right)$:

$$r(x) \approx r^0 + J\Delta x \tag{2}$$

where

$$r^0 = r\left(x^0\right), \tag{3}$$

$$\Delta x = x - x^0, \tag{4}$$

$$J = \frac{\partial r}{\partial x}\Big|_{x=x^0} \tag{5}$$

and J is the Jacobian of r that is decomposed into a pose part J_p and a landmark part J_l. An added regularization term that improves convergence gives the damped linear least squares problem

$$\min_{\Delta x_p, \Delta x_l} \left(\|r^0 + \begin{pmatrix} J_p & J_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix}\|^2 + \lambda \|\begin{pmatrix} D_p & D_l \end{pmatrix} \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix}\|^2 \right) \tag{6}$$

with 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

$$H \begin{pmatrix} \Delta x_p \\ \Delta x_l \end{pmatrix} = - \begin{pmatrix} b_p \\ b_l \end{pmatrix} \tag{7}$$

where

$$H = \begin{pmatrix} U_\lambda & W \\ W^\top & V_\lambda \end{pmatrix}, \tag{8}$$

$$U_\lambda = J_p^\top J_p + \lambda D_p^\top D_p, \tag{9}$$

$$V_\lambda = J_l^\top J_l + \lambda D_l^\top D_l, \tag{10}$$

$$W = J_p^\top J_l, \ b_p = J_p^\top r^0, \tag{11}$$

$$b_l = J_l^\top r^0 \tag{12}$$

As the system matrix H is of size $(d_p n_p + 3n_l)^2$ and tends to be excessively costly for large-scale problems [1], it is common to reduce it by using the Schur complement trick and forming the reduced camera system

$$S\Delta x_p = -\widetilde{b} \tag{13}$$

with

$$S = U_\lambda - WV_\lambda^{-1}W^\top, \tag{14}$$

$$\widetilde{b} = b_p - WV_\lambda^{-1}b_l \tag{15}$$

and then solving (13) for Δx_p and backsubstituting Δx_p in

$$\Delta x_l = -V_\lambda^{-1}\left(-b_l + W^\top \Delta x_p\right) \tag{16}$$

3.2 Multidirectional Conjugate Gradients

Direct methods such as Cholesky decomposition [17] have been studied for solving (13) for small-size problems, but this approach implies a high computational cost whenever problems become too large.

A very popular iterative solver for large symmetric positive-definite system is the CG algorithm [16]. Since its convergence rate depends on the distribution of eigenvalues of S it is common to replace (13) by a preconditioned system. Given a preconditioner M the preconditioned linear system associated to

$$S\Delta x_p = -\widetilde{b} \tag{17}$$

is

$$M^{-1}S\Delta x_p = -M^{-1}\widetilde{b} \tag{18}$$

and the resulting algorithm is called Preconditioned Conjugate Gradients (PCG) (see Algorithm 1). For block structured matrices as S a competitive preconditioner is the block diagonal matrix $D(S)$, also called block Jacobi preconditioner [1]. It is composed of the block diagonal elements of S. Since the block S_{mj} of S is nonzero if and only if cameras m and j share at least one common point, each diagonal block depends on a unique pose and is applied to the part of conjugate gradients residual r_i^j that is associated to this pose. The motivation of this section is to enlarge the conjugate gradients search space by using several local contributions instead of a unique global contribution.
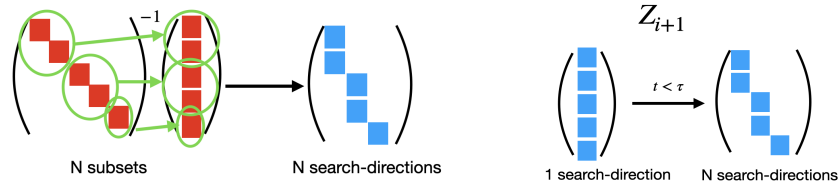
Adaptive multidirections

Local preconditioners. We propose to decompose the set of poses into N subsets of sizes $l_1,...,l_N$ and to take into consideration the block-diagonal matrix $D_p(S)$ of the block-jacobi preconditioner and the associated residual r^p that correspond to the l_p poses of subset p (see Figure 2(a)). All direct solves are performed inside these subsets and not in the global set. Each local solve is treated

---

Algorithm 1 Preconditioned Conjugate Gradients

1: x_0, $r_0 = -\tilde{b} - Sx_0$, $Z_0 = D(S)^{-1}r_0$, $P_0 = Z_0$, ϵ;
2: **while** $i < \text{imax}$ **do**
3:     $Q_i = SP_i$;
4:     $\Delta_i = Q_i^\top P_i$; $\gamma_i = P_i^\top r_i$; $\alpha_i = \frac{\gamma_i}{\Delta_i}$;
5:     $x_{i+1} = x_i + \alpha_i P_i$;
6:     $r_{i+1} = r_i - \alpha_i Q_i$;
7:     **if** $r_{i+1} < \epsilon * r_0$ **then**
8:         break
9:     **end if**
10:     $Z_{i+1} = D(S)^{-1}r_{i+1}$;
11:     $\Phi_i = Q_i^\top Z_{i+1}$; $\beta_i = \frac{\Phi_i}{\Delta_i}$;
12:     $P_{i+1} = Z_{i+1} - \beta_i P_i$;
13: **end while**
14: return x_{i+1};

(a) Decomposed preconditioned CG residuals

(b) Enlarged search-space

Fig. 2: (a) Block-Jacobi preconditioner $D(S)$ is divided into N submatrices $D_p(S)$ and each of them is directly applied to the associated block-row r^p in the CG residual. (b) Up to a τ-test the search-space is enlarged. Each iteration provides N times more search-directions than PCG.

as a separate preconditioned equation and provides a unique search-direction. Consequently the conjugate vectors $Z_{i+1} \in \mathbb{R}^{d_p n_p}$ in the preconditioned conjugate gradients (line 10 in Algorithm 1) are now replaced by conjugate matrices $Z_{i+1} \in \mathbb{R}^{d_p n_p \times N}$ whose each column corresponds to a local preconditioned solve. The search-space is then significantly enlarged: N search directions are generated at each inner iteration instead of only one. An important drawback is that matrix-vector products are replaced by matrix-matrix products which can lead to a significant additional cost. A trade-off between convergence improvement and computational cost needs to be designed.

Adaptive τ-test. Following a similar approach as in [15] we propose to use an adaptive multidirectional conjugate gradients algorithm (MCG, see Algorithm 2) that adapts automatically if the convergence is too slow. Given a threshold $\tau \in \mathbb{R}^+$ chosen by the user, a τ-test determines whether the algorithm sufficiently reduces the error (case $t_i > \tau$) or not (case $t_i < \tau$). In the first case a global block

---

Algorithm 2 Multidirectional Conjugate Gradients

1: x_0, $r_0 = -\widetilde{b} - Sx_0$, $Z_0 = D(S)^{-1}r_0$, $P_0 = Z_0$, ϵ;
2: **while** $i < \text{imax}$ **do**
3:     $Q_i = SP_i$;
4:     $\Delta_i = Q_i^\top P_i$; $\gamma_i = P_i^\top r_i$; $\alpha_i = \Delta_i^\dagger \gamma_i$;
5:     $x_{i+1} = x_i + P_i\alpha_i$;
6:     $r_{i+1} = r_i - Q_i\alpha_i$;
7:     **if** $r_{i+1} < \epsilon * r_0$ **then**
8:         break
9:     **end if**
10:     $t_i = \frac{\gamma_i^\top \alpha_i}{r_{i+1}^\top D(S)^{-1}r_{i+1}}$;
11:     **if** $t_i < \tau$ **then**
12:         $Z_{i+1} = \begin{pmatrix} D_1(S)^{-1}r_{i+1}^1 & & 0 \\ 0 & \dots & \dots \\ \dots & \dots & 0 \\ 0 & & D_N(S)^{-1}r_{i+1}^N \end{pmatrix}$;
13:     **else**
14:         $Z_{i+1} = D(S)^{-1}r_{i+1}$;
15:     **end if**
16:     $\Phi_{i,j} = Q_j^\top Z_{i+1}$; $\beta_{i,j} = \Delta_j^\dagger \Phi_{i,j}$ for $j = 0, ..., i$;
17:     $P_{i+1} = Z_{i+1} - \sum_{j=0}^{i} P_j\beta_{i,j}$;
18: **end while**
19: return x_{i+1};

Jacobi preconditioner is used and the algorithm performs a step of PCG; in the second case local block Jacobi preconditioners are used and the search-space is enlarged (see Figure 2(b)).

Optimized implementation Besides matrix-matrix products two other changes appear. Firstly an $N \times N$ matrix Δ_i must be inverted (or pseudo-inverted if Δ_i is not full-rank) each time $t_i < \tau$ (line 4 in Algorithm 2). Secondly a full reorthogonalization is now necessary (line 16 in Algorithm 2) because of numerical errors while $\beta_{i,j} = 0$ as soon as $i \neq j$ in PCG.

To improve the efficiency of MCG we do not directly apply S to P_i (line 3 in Algorithm 2) when the search-space is enlarged. By construction the block S_{kj} is nonzero if and only if cameras k and j observe at least one common point. The trick is to use the construction of Z_i and to directly apply the non-zero blocks S_{jk}, i.e. consider only poses j observing a common point with k, to the column in Z_i associated to the subset containing pose k and then to compute

$$Q_i = SZ_i - \sum_{j=0}^{i-1} Q_j\beta_{i,j} \tag{19}$$

To get t_i we need to use a global solve (line 10 in Algorithm 2). As the local block Jacobi preconditioners $\{D_p(S)\}_{p=1,...,N}$ and the global block Jacobi

preconditioner $D(S)$ share the same blocks it is not necessary to derive all local solves to construct the conjugates matrix (line 12 in Algorithm 2); instead it is more efficient to fill this matrix with block-row elements of the preconditioned residual $D(S)^{-1} r_{i+1}$.

As the behaviour of CG residuals is *a priori* unknown the best decomposition is not obvious. We decompose the set of poses into $N - 1$ subsets of same size and the last subset is filled by the few remaining poses. This structure presents the practical advantage to be very easily fashionable and the parallelizable block operations are balanced.

4   Experimental Evaluations

4.1   Algorithm and Datasets

Levenberg-Marquardt (LM) loop. Starting with damped parameter 10^{-4} we update λ according to the success or failure of the LM loop. Our implementation runs for at most 25 iterations, terminating early if a relative function tolerance of 10^{-6} is reached. Our evaluation is built on the LM loop implemented in [19] and we also estimate intrinsics parameters for each pose.

Iterative solver step. For a direct performance comparison we implement our own MCG and PCG solvers in C++ by using Eigen 3.3 library. All row-major-sparse matrix-vector and matrix-matrix products are multi-threaded by using 4 cores. The tolerance ϵ and the maximum number of iterations are set to 10^{-6} and 1000 respectively. Pseudo-inversion is derived with the pseudo-inverse function from Eigen.

Datasets. For our evaluation we use 9 datasets with different sizes and heterogeneous Schur complement matrix densities d from BAL [1] and 1dSfM [20] datasets (see Table 1). The values of N and τ are arbitrarily chosen and the robustness of our algorithm to these parameters is discussed in the next subsection.

We run experiments on MacOS 11.2 with Intel Core i5 and 4 cores at 2GHz.

4.2   Sensitivity with τ and N

In this subsection we are interested in the solver runtime ratio that is defined as $\frac{t_{MCG}}{t_{PCG}}$ where t_{MCG} (resp. t_{PCG}) is the total runtime to solve all the linear systems (12) with MCG (resp. PCG) until a given BA problem converges. We investigate the influence of τ and N on this ratio.

Sensitivity with τ. We solve BA problem for different values of τ and for a fixed number of subsets N given in Table 1. For each problem a wide range of values supplies a good trade-off between the augmented search-space and the additional computational cost (see Figure 3). Although the choice of τ is crucial it does not require a high accuracy. That confirms the tractability of our solver with τ.

Table 1: Details of the problems from BAL (prefixed as: F for *final*, L for *Lady-bug*) and 1dSfM used in our experiments. d is the density of the associated Schur complement matrix, N is the number of subsets, τ is the adaptive threshold that enlarges the search-direction space.

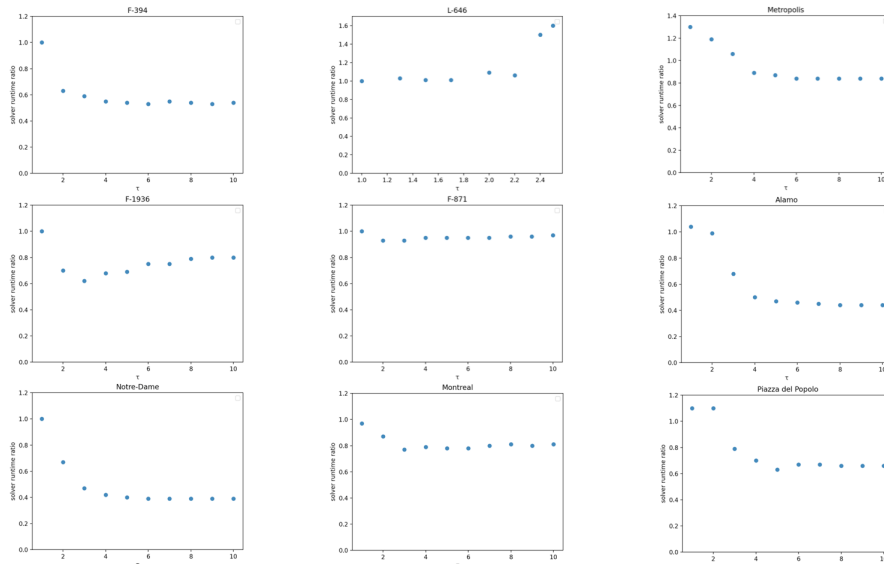| Names	poses	points	projections	d	N	τ
Piazza del Popolo	335	37,609	195,016	0.57	33	10
Metropolis	346	55,679	255,987	0.50	34	6
F-394	394	100,368	534,408	0.94	131	6
Montreal	459	158,005	860,116	0.60	22	3
Notre-Dame	547	273,590	1,534,747	0.77	45	10
Alamo	571	151,085	891,301	0.77	47	10
L-646	646	73,484	327,297	0.25	64	2
F-871	871	527,480	2,785,977	0.40	43	2.5
F-1936	1,936	649,673	5,213,733	0.91	121	3.3

Fig. 3: Robustness to τ. The plots show the performance ratio as a function of τ for a number of subsets given in Table 1. The wide range of values that give similar performance confirms the tractability of MCG with τ.

Sensitivity with N. Similarly we solve BA problem for different values of N and for a fixed τ given in Table 1. For each problem a wide range of values supplies a good trade-off between the augmented search-space and the additional computational cost (see Figure 4). That confirms the tractability of our solver with N.
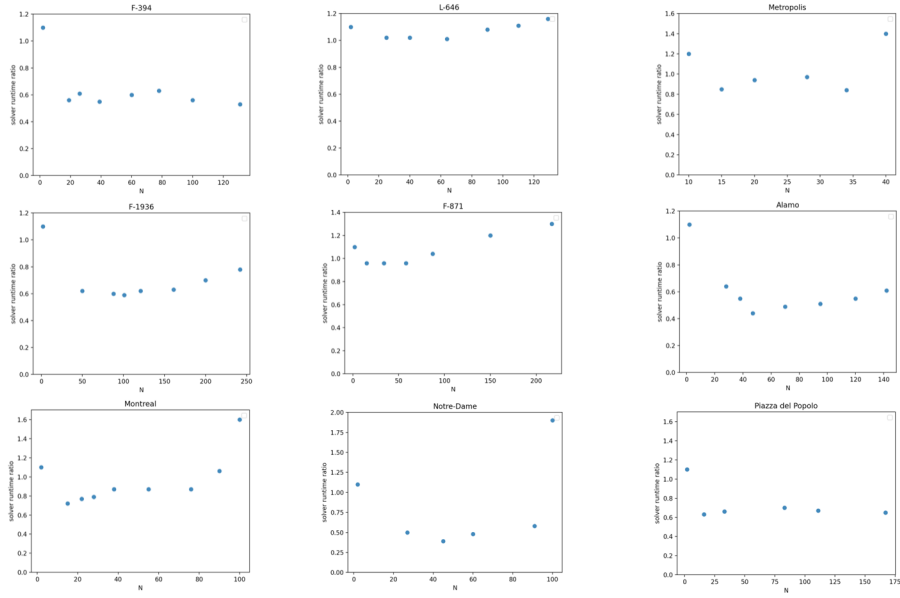
Fig. 4: Robustness to the number of subsets N. The plots show the performance ratio as a function of N for τ given in Table 1. The wide range of values that give similar performance confirms the tractability of MCG with N.
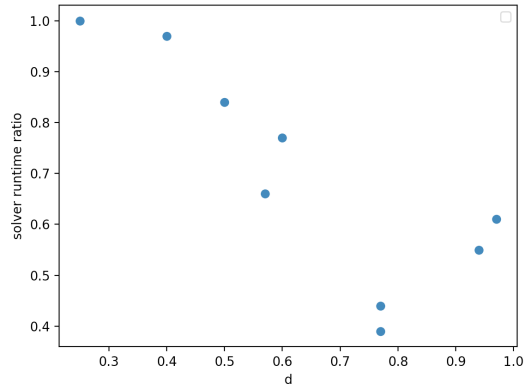

Fig. 5: Density effect on the relative performance. Each point represents a BA problem from Table 1 and d is the density of the Schur complement matrix. Our solver competes PCG for sparse Schur matrix and leads to a significant speed-up for dense Schur matrix.

4.3   Density effect

As the performance of PCG and MCG depends on matrix-vector product and matrix-matrix product respectively we expect a correlation with the density of the Schur matrix. Figure 5 investigates this intuition: MCG greatly outperfoms PCG for dense Schur matrix and is competitive for sparse Schur matrix.

4.4   Global performance

Figures 6 and 7 present the total runtime with respect to the number of BA iterations for each problem and the convergence plots of total BA cost for *F-1936* and *Alamo* datasets, respectively. MCG and PCG give the same error at each BA iteration but the first one is more efficient in terms of runtime. Table 2 summarizes our results and highlights the great performance of MCG for dense Schur matrices. In the best case BA resolution is more than 20% faster than using PCG. Even for sparser matrices MCG competes PCG: in the worst case MCG presents similar results as PCG. If we restrict our comparison to the linear system solve steps our relative results are even better: MCG is up to 60% faster than PCG and presents similar results as PCG in the worst case.
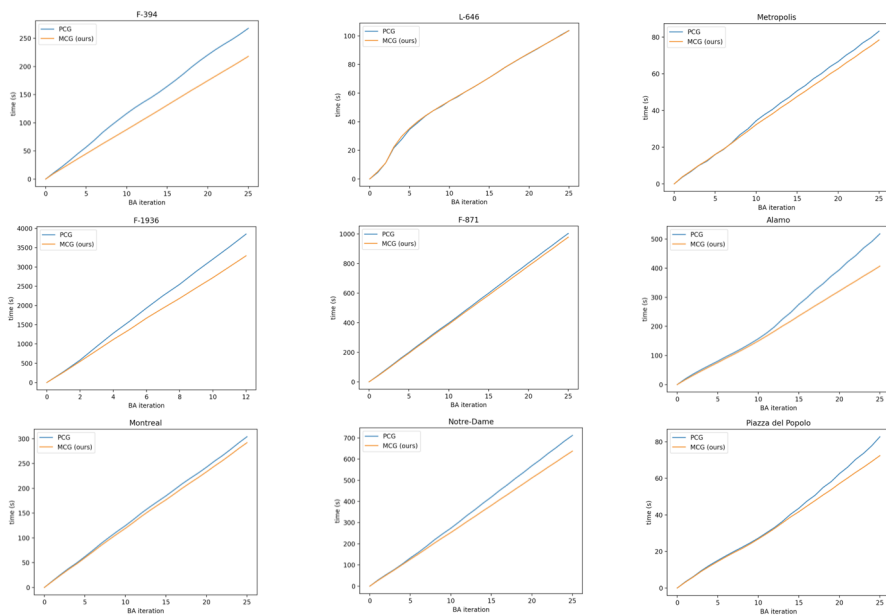

Fig. 6: Global runtime to solve BA problem. The plots represent the total time with respect to the number of BA iterations. For almost all problems the BA resolution with MCG (orange) is significantly faster than PCG (blue).
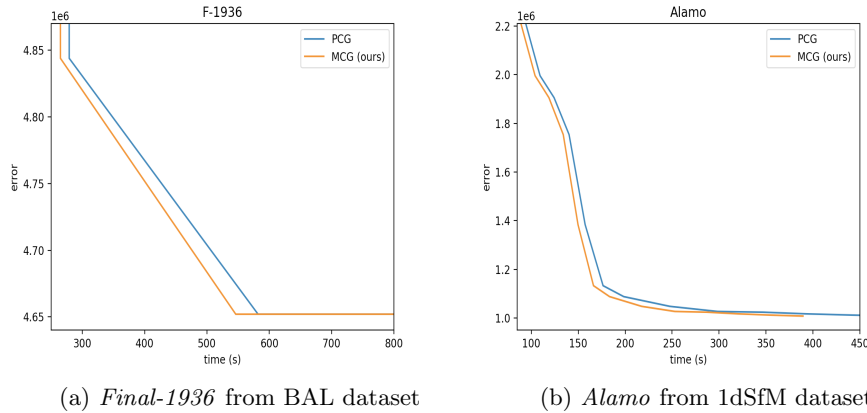
(a) *Final-1936* from BAL dataset   (b) *Alamo* from 1dSfM dataset

Fig. 7: Convergence plots of (a) *Final-1936* from BAL dataset and (b) *Alamo* from 1dSfm dataset. The y-axes show the total BA cost.

5   Conclusion

We propose a novel iterative solver that accelerates the solution of the normal equation for large-scale BA problems. The proposed approach generalizes the traditional preconditioned conjugate gradients algorithm by enlarging its search-space leading to a convergence in much fewer iterations. Experimental validation on a multitude of large scale BA problems confirms a significant speedup in solving the normal equation of up to 61%, especially for dense Schur matrices where baseline techniques notoriously struggle. Moreover, detailed ablation studies demonstrate the robustness to variations in the hyper-parameters and increasing speedup as a function of the problem density.

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. S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In International Conference on Computer Vision (ICCV), 2009.
3. C. Bovet, A. Parret-Fréaud, N. Spillane and P. Gosselet. Adaptive Multiprecondi- tioned FETI: Scalability Results and Robustness Assessment. Computers & Struc- tures, 193 (2017), pp. 1-20.
4. R. Bridson and C. Greif. A multipreconditioned conjugate gradient algorithm. SIAM J. Matrix Anal. Appl., 27(4):1056–1068 (electronic), 2006.
5. D. C. Brown. A solution to the general problem of multiple station analytical stereo triangulation. RCA-MTP data reduction technical report no. 43 (or AFMTC TR 58-8), Patrick Airforce Base, Florida, 1958.
6. M. Byrod, K. Aström, and S. Lund. Bundle adjustment using conjugate gradients with multiscale preconditioning. In BMVC, 2009.

Table 2: Relative performances of MCG w.r.t. PCG. d is the density of the associated Schur complement matrix. MCG greatly outperforms PCG (up to 61% faster) for dense Schur matrix and competes PCG for sparse Schur matrix. The global BA resolution is up to 22% faster.

Name	Solver runtime ratio	Global runtime ratio	d
Notre-Dame	**0.39**	**0.86**	0.77
Alamo	**0.44**	**0.78**	0.77
F-394	**0.55**	**0.80**	0.94
F-1936	**0.61**	**0.84**	0.97
Piazza del Popolo	**0.66**	**0.87**	0.57
Montreal	**0.77**	0.96	0.60
Metropolis	**0.84**	0.92	0.50
F-871	0.93	0.97	0.40
L-646	1.01	1.00	0.25

7. F. Dellaert, J. Carlson, V. Ila, K. Ni, and C. E. Thorpe. Subgraph-preconditioned conjugate gradients for large scale slam. In IROS, pages 2566–2571, 2010.
8. M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. J. Research Nat. Bur. Standards, 49:409–436 (1953), 1952.
9. Y.Jeong, D. Nister,D. Steedly, R. Szeliski, and I.-S. Kweon. Pushing the envelope of modern methods for bundle adjustment. In CVPR, pages 1474–1481, 2010.
10. A. Kushal and S. Agarwal. Visibility Based Preconditioning for Bundle Adjustment. In CVPR, 2012.
11. M. Lourakis and A. Argyros. Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment. In International Conference on Computer Vision (ICCV), pages 1526–1531, 2005.
12. J. Mandel. Balancing domain decomposition. Comm. Numer. Methods Engrg., 9(3):233–241, 1993.
13. Y. Notay. Flexible conjugate gradients. SIAM J. Sci. Comput., 22(4):1444–1460, 2000.
14. Y. Saad. Iterative methods for sparse linear systems. SIAM, 2003.
15. N. Spillane. An Adaptive Multipreconditioned Conjugate Gradient Algorithm, SIAM J. Sci. Comput. 38 (3) (2016) A1896–A1918.
16. L. Trefethen and D. Bau, Numerical linear algebra, SIAM, 1997.
17. B. Triggs, P. McLauchlan, H.R.I, andA. Fitzgibbon. Bundle Adjustment - A modern synthesis. In Vision Algorithms'99, pages 298–372, 1999.
18. C. Wu, S. Agarwal, B. Curless, and S. Seitz. Multicore bundle adjustment. In CVPR, pages 3057–3064, 2011.
19. 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.
20. K. Wilson and N. Snavely. Robust Global Translations with 1DSfM. In Proceedings of the European Conference on Computer Vision (ECCV), 2014.