# Domain Decomposition Based Preconditioned Solver for Bundle Adjustment

Shrutimoy Das, Siddhant Katyan, and Pawan Kumar

International Institute of Information Technology, Hyderabad, India - 500032
{shrutimoy.das@research.,siddhant.katyan@research.,
pawan.kumar@}iiit.ac.in

**Abstract.** We propose Domain Decomposed Bundle Adjustment (DDBA), a robust and efficient solver for the bundle adjustment problem. Bundle adjustment (BA) is generally formulated as a non-linear least squares problem and is solved by some variant of the Levenberg-Marquardt (LM) algorithm. Each iteration of the LM algorithm requires solving a system of normal equations, which becomes computationally expensive with the increase in problem size. The coefficient matrix of this system has a sparse structure which can be exploited for simplifying the computations in this step. We propose a technique for approximating the Schur complement of the matrix, and use this approximation to construct a preconditioner, that can be used with the Generalized Minimal Residual (GMRES) algorithm for solving the system of equations. Our experiments on the `BAL` [2] dataset show that the proposed method for solving the system is faster than GMRES solve preconditioned with block Jacobi and more memory efficient than direct solve.

**Keywords:** Computer Vision · Bundle Adjustment · Structure from Motion · Generalized Minimal Residual · Preconditioning · Domain Decomposition.

## 1 Introduction

Many recent works in three dimensional ($3D$) reconstruction using Structure-from-Motion (SfM) algorithms has focused on building systems [1, 9, 19] that are capable of handling millions of images from unstructured internet photo collections. Given the feature matches between images, bundle adjustment (BA) [21] is a key component in most SfM systems. It is typically used as the last step in a $3D$ reconstruction pipeline. For large scale problems however, BA becomes very expensive computationally and thus, creates a bottleneck in the SfM systems. As a result, there has been a lot of interest in developing scalable large scale bundle adjustment algorithms [2, 6, 7, 11, 22].

The BA problem is typically formulated as the minimization of a nonlinear least squares problem, which can be done by using a classical algorithm such as the Levenberg-Marquardt(LM) algorithm. In each iteration of the LM algorithm, a solution to a linear system is required, which is the most computationally expensive step. A lot of research has been focused on making this step cheaper.

In [16], a direct method using Dense Cholesky factorization has been proposed. However, these methods do not scale well as the problem size increases. This has led to the application of iterative methods, specifically the Conjugate Gradient(CG)[18] method, for solving these systems. The convergence of these methods depend upon the condition number of the coefficient matrix. However, it has been observed that BA problems are very ill-conditioned. To solve this problem, preconditioning matrices or *preconditioners* [18] are applied to the system. They lower the condition number of the systems, which in turn speeds up the convergence of the iterative methods.

In this paper, we propose a new preconditioner which is based on the domain decomposition of the coefficient matrix. As it has been pointed out in [2]: "each point in the SfM problem is a domain, and the cameras form the interface between these domains". Using the domain decomposition method, we present a technique that can be used for approximating the global Schur complement of the matrix. We name this preconditioner as Mini Schur Complement (MSC) preconditioner. One of the advantages of using this preconditioner is that it is sparse, highly parallelizable and can be scaled up for very large problems. However, the preconditioned operator is unsymmetric. Thus, we use another iterative method known as restarted generalized minimal residual (GMRES). As the results show, solving the normal equations using GMRES preconditioned with MSC gives state-of-the-art performance on the `BAL` dataset.

The remaining part of the paper is organized as follows. Section 2 introduces the BA problem and also gives a review of the recent work on the use of preconditioned iterative methods. Section 3 gives a brief overview of domain decomposition methods and describes the design and implementation of the MSC preconditioner. Section 4 compares the results of our technique with direct solver and block Jacobi preconditioned `GMRES` solver. In section 5, we conclude with a discussion.

## 2 Bundle Adjustment

Bundle adjustment tries to minimize the sum of reprojection errors between the 2D observations and the reprojected 2D points which are determined by the point and camera parameters. More information about this process can be found in [21].

Suppose that the scene to be reconstructed consists of $p$ 3D points(or features), individually denoted as $y_i, i = 1, \ldots, p$, and these points are imaged in $q$ cameras, whose individual parameters are denoted as $z_k, k = 1, \ldots, q$. Assume that the structure(point) and camera parameters to be estimated are taken in a large state vector $x \in \mathbb{R}^{(p+q)}$ which has the block structure $x = [y_1, \ldots, y_p, z_1, \ldots, z_q]^T$. Then, the reprojection error is defined as $f_k(x) = r_k(x) - m_{xk}$, for k = 1, ..., q. Given the mean reprojection error for each camera, the unknown 3D point and camera parameters can be estimated by minimizing the total reprojection error. Define $F(x) = [f_1(x), \ldots, f_q(x)]^T$ to be a $q$-dimensional

function of the given parameter vector $x$. Then, the bundle adjustment problem can be stated as

$$x^* = \underset{x}{argmin} \frac{1}{2}||F(x)||_2^2. \tag{1}$$

In (1), the objective function is non-linear. For solving non-linear least squares problems of this form, the Levenberg-Marquardt (LM) algorithm [3, 15, 17] is applied. It is an iterative method where, in each iteration, an affine approximation of the cost function $F(x)$ in a neighbourhood of the current iterate $x_t$, is minimized. It is shown in [3] that the next iterate $x_{t+1}$ can be computed as

$$x_{t+1} = x_t - (J^T J + \lambda^t \texttt{diag}(J^T J))^{-1} J^T F(x_t). \tag{2}$$

Let $H_{LM} = J^T J + \lambda^t \texttt{diag}(J^T J)$ and $g = J^T F(x_t)$. Here, $J$ is the Jacobian of $F(x)$ at $x_t$ and $\lambda^t > 0$ is a damping parameter which ensures that $x_{t+1}$ lies in a neighbourhood of $x_t$. It should be noted that the definition of $H_{LM}$ results in an approximation of the Hessian. Then, rearranging the terms in (2) gives

$$H_{LM} \Delta x = -g, \text{ where } \Delta x = x_{t+1} - x_t. \tag{3}$$

The Hessian $H_{LM}$ is symmetric and positive definite(SPD). Thus, (3) can be solved as $\Delta x = -H_{LM}^{-1} g$ to get the exact solution. However, when the problem size becomes large, computing the inverse of $H_{LM}$ becomes expensive. In these cases, an inexact solution of the system (3) can be computed by using an iterative method, such as GMRES. The convergence of the iterative methods depend on the condition number of the coefficient matrix, the Hessian in this case. For badly conditioned problems, such as bundle adjustment, the condition number can be improved with the help of a preconditioner [18]. This paper proposes the design and implementation of such a preconditioner, which exploits the special structure of the Hessian $H_{LM}$.

### 2.1 Structure of the Hessian

In the state vector $x$ defined in Section 2, let $s$ be the size of each point block and $c$ be the size of each camera block. For the `BAL` dataset used in this paper, $c = 9$ and $s = 3$. Given these block sizes, the Jacobian $J$ can be partitioned into a point part $J_s$ and camera part $J_c$ as $J = [J_s; J_c]$, which gives

$$H_{LM} = \begin{bmatrix} J_s^T J_s & J_s^T J_c \\ J_c^T J_s & J_c^T J_c \end{bmatrix} = \begin{bmatrix} D & L^T \\ L & G \end{bmatrix}. \tag{4}$$

Here, $D \in \mathbb{R}^{ps \times ps}$ is a block diagonal matrix with $p$ blocks such that each block is of size $s \times s$ and $G \in \mathbb{R}^{qc \times qc}$ is a block diagonal matrix with $q$ blocks such that each block is of size $c \times c$. The matrix $L \in \mathbb{R}^{qc \times ps}$ is a general block sparse matrix. Thus, we can rewrite (3) as a block structured linear system as follows

$$\begin{bmatrix} D & L^T \\ L & G \end{bmatrix} \begin{bmatrix} \Delta x_s \\ \Delta x_c \end{bmatrix} = \begin{bmatrix} g_s \\ g_c \end{bmatrix}, \tag{5}$$

where $\Delta x = [\Delta x_s; \Delta x_c]$, $\Delta x_s$and$\Delta x_c$ correspond to point parameter blocks and camera parameter blocks of $\Delta x$, respectively, and $g = [g_s; g_c]$, $g_s$ and $g_c$ correspond to point and camera parameter blocks of $g$, respectively. Different approaches have been proposed for solving (5), which exploits the special structure of the Hessian.

### 2.2 Previous Work

For solving (5), direct methods have been well studied in literature [16, 21]. In [4], the special structure of the Hessian is exploited to solve the system using a reduced camera system and a reduced structure system. A survey of various direct and iterative methods as well the use of various preconditioners can be found in [21]. Cholesky factorization is used for solving the reduced camera system in [16]. However, for large scale problems, this method does not scale satisfactorily.

An advantage of using iterative methods, such as CG, is that these methods require less memory compared to direct methods. This is because these methods require only matrix-vector products. However, since BA problems are very badly conditioned, recent research has focused on obtaining efficient preconditioners to speed up the convergence of these methods. In [2], several classical preconditioners have been implemented and their impact on large scale problems is shown. In [5],Preconditioned Conjugate Gradients(PCG) is used for solving (5) with an incomplete QR factorization based preconditioner. In [22], PCG is used for solving the reduced camera system with the bandwidth limited block diagonal of the Schur complement as the preconditioner. [7] exploits hardware parallelism on multicore CPUs as well as multicore GPUs to solve the BA problem by a new inexact Newton type method. Avanish et al. [14] utilizes the camera-point visibility structure in the scene to form block diagonal and block tridiagonal preconditioners. [11] explores a generalized subgraph preconditioning (GSP) technique which is based on the combinatorial structure of the BA problem. In [12], a preconditioner based on a deflated two grid methods is used with GMRES as the iterative method.

Usually for small to medium sized problems, direct methods converge faster than iterative methods . In this paper, we show that our method is more memory efficient than direct methods and faster than iterative methods preconditioned with block Jacobi, for small to medium problems, to converge to a comparable mean reprojection error. Also, it has been observed that the construction of the MSC preconditioner does not take much time.

## 3 Domain Decomposition Method

Domain decomposition (DD) methods refer to a class of divide-and-conquer techniques, that have been primarily developed for solving Partial Differential Equations over regions in two or three dimensions. However, the principles used in this techniques have also been exploited in other fields of scientific and engineering computational problems. The DD methods attempt to solve the problem on the entire domain from problem solutions to the subdomains. For more details, see [18, 20]. One of the most widely used non-overlapping DD methods is the Schur complement method, which is described below.

Consider the following block triangular factorization of $H_{LM}$:

$$H_{LM} = \begin{bmatrix} D & L^T \\ L & G \end{bmatrix} = \begin{bmatrix} I_D & 0 \\ LD^{-1} & I_G \end{bmatrix} \begin{bmatrix} D & L^T \\ 0 & S \end{bmatrix}$$

where $I_D \in \mathbb{R}^{ps \times ps}$ and $I_G \in \mathbb{R}^{qc \times qc}$, are identity matrices. Here, $S = G - LD^{-1}L^T$ is the *Schur complement* of $D$ in $H_{LM}$. It has been observed that the construction of $S$ for large problems becomes computationally expensive. Also, the Cholesky decomposition of $S$ leads to dense factors, even though $S$ remains sparse. Here, we present a technique for approximating the Schur complement and design a preconditioner using this approximation.

### 3.1 The Mini Schur Complement Preconditioner

In [13], several methods for approximation of the global Schur complement $S$ have been mentioned. Here, we construct the Mini Schur Complements (MSC) using the MSC based on Numbering (MSCN) scheme, which is described below.
We consider the block $2 \times 2$ partitioned system in (4). The matrix $H_{LM}$ is further partitioned as follows.

$$H_{LM} = \left[\begin{array}{cc|cc} D_{11} & D_{12} & L_{11}^T & L_{12}^T \\ D_{21} & D_{22} & L_{21}^T & L_{22}^T \\ \hline L_{11} & L_{12} & G_{11} & G_{12} \\ L_{21} & L_{22} & G_{21} & G_{22} \end{array}\right] \tag{6}$$

Now, a further approximation of the matrix in (6) is constructed by dropping the blocks $D_{ij}, L_{ij}^T, L_{ij}$ and $G_{ij}$ for which $i \neq j$. Thus the following approximation $\hat{H}_2$ is obtained.

$$\hat{H}_2 = \left[\begin{array}{cc|cc} D_{11} & & L_{11}^T & \\ & D_{22} & & L_{22}^T \\ \hline L_{11} & & G_{11} & \\ & L_{22} & & G_{22} \end{array}\right] \tag{7}$$

Here, the subscript 2 in $\hat{H}_2$ denotes the number of principal sub-matrices of the matrix $G$, namely, $G_{11}$ and $G_{22}$. The matrix in (7) is further partitioned to get the following matrix

$$\hat{H}_2 = \left[\begin{array}{cc|cc|cc|cc} \hat{D}_{11} & \hat{D}_{12} & & & \hat{L}_{11}^T & \hat{L}_{12}^T & & \\ \hat{D}_{21} & \hat{D}_{22} & & & \hat{L}_{21}^T & \hat{L}_{22}^T & & \\ \hline & & \hat{D}_{33} & \hat{D}_{34} & & & \hat{L}_{33}^T & \hat{L}_{34}^T \\ & & \hat{D}_{43} & \hat{D}_{44} & & & \hat{L}_{43}^T & \hat{L}_{44}^T \\ \hline \hat{L}_{11} & \hat{L}_{12} & & & \hat{G}_{11} & \hat{G}_{12} & & \\ \hat{L}_{21} & \hat{L}_{22} & & & \hat{G}_{21} & \hat{G}_{22} & & \\ \hline & & \hat{L}_{33} & \hat{L}_{34} & & & \hat{G}_{33} & \hat{G}_{34} \\ & & \hat{L}_{43} & \hat{L}_{44} & & & \hat{G}_{43} & \hat{G}_{44} \end{array}\right] \tag{8}$$

Again, a sparse approximation of (8) is done by dropping the blocks $\hat{D}_{ij}, \hat{L}_{ij}^T, \hat{L}_{ij}$ and $\hat{G}_{ij}$ for which $i \neq j$, to obtain the following matrix

$$\hat{H}_4 = \left[\begin{array}{cc|cc|cc|cc} \hat{D}_{11} & & & & \hat{L}_{11}^T & & & \\ & \hat{D}_{22} & & & & \hat{L}_{22}^T & & \\ \hline & & \hat{D}_{33} & & & & \hat{L}_{33}^T & \\ & & & \hat{D}_{44} & & & & \hat{L}_{44}^T \\ \hline \hat{L}_{11} & & & & \hat{G}_{11} & & & \\ & \hat{L}_{22} & & & & \hat{G}_{22} & & \\ \hline & & \hat{L}_{33} & & & & \hat{G}_{33} & \\ & & & \hat{L}_{44} & & & & \hat{G}_{44} \end{array}\right]$$

Here the subscript 4 in $\hat{H}_4$ denotes the number of principal sub-matrices of matrix $G$. Eliminating the blocks $\hat{L}_{ii}$ by using $\hat{D}_{ii}$ as a pivot, we obtain an approximation to the global Schur complement $S$ by $\hat{S}_4 = \texttt{blkDiag}(S_{ii})$ where $S_{ii} = \hat{G}_{ii} - \hat{L}_{ii}\hat{D}_{ii}^{-1}\hat{L}_{ii}^T$.

The matrix $S_{ii}$ is called a Mini Schur Complement (MSC). Here, for simplicity, we have partitioned the matrix recursively into a block $2 \times 2$ matrix. During implementation, by taking advantage of the sparsity structure of the Hessian $H_{LM}$ and the information about the size of the blocks, we could directly identify the blocks $\hat{G}_{ii}, \hat{D}_{ii}, \hat{L}_{ii}^T$ and $\hat{L}_{ii}$, such that $S_{ii}$ is computed as $S_{ii} = G_{ii} - L_{ii}D_{ii}^{-1}L_{ii}^T$, where $i = 1 : m$ and $m$ is the number of MSCs desired.

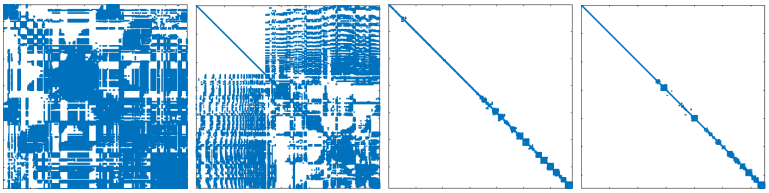

**Fig. 1.** The first two plots show the Schur complement of `ladybug-372` and `ladybug-885` respectively. The third and fourth plots show their respective MSC approximations ($m = 30$ blocks).

**MSC Preconditioner :** Let $\hat{S}_m$ denote the Schur complement approximation computed from $m$ MSCs. Then we construct the MSC preconditioner as

$$P_{msc} = \left[\begin{array}{c|c} D & 0 \\ \hline L & \hat{S}_m \end{array}\right]$$

Here $D$ and $\hat{S}_m$ have a block diagonal structure and also, $D$ and $L$ blocks are already available from the coefficient matrix $H_{LM}$. Thus, storing the required blocks of the MSC preconditioner does not require much extra memory. In figure 1, the Schur complement and the Mini Schur Complement approximation of two problems are shown. It can be seen that the MSC approximation has a lot more sparsity than the global Schur complement.

## 4 Experimental Evaluation

### 4.1 Implementation Details

For performing the experiments, we select block $D$ such that the number of rows (and columns) of $D$ is given by $3\times$ (number of points) and block $G$ such that the number of rows (and columns) is given by $9\times$ (number of cameras). The information about the number of points and the number of cameras are available in the dataset. We construct the block Jacobi preconditioner as $P_{jac} = \texttt{blkdiag}(D, G)$ (as shown in [2]), where `blkdiag` forms a block diagonal matrix using the given blocks. For constructing the MSC preconditioner, the number of MSC blocks is taken as 30 blocks.

For the Levenberg-Marquardt algorithm, we use a freely available sparse C++ implementation (SSBA)[1], which has several cost functions that are used by the LM algorithm for the BA step. Out of these, we choose the bundle_large_lifted_schur cost function implemented in the SSBA package, which is discussed in detail in [23]. The LM algorithm runs for 100 iterations, or till the difference in norms of two consecutive residuals is not less than $10^{-12}$ in magnitude, whichever criterion is met first. For solving the *normal* equations in (3) using direct method, SSBA uses LDL factorization [10], which is a Cholesky like factorization method for sparse symmetric positive definite matrices. COLAMD is applied for appropriate column reordering. Both LDL and COLAMD have been adopted from the SuiteSparse package [8].

We experimented with Preconditioned Conjugate Gradient(PCG) as the iterative solver but the results we got using $P_{jac}$ as a preconditioner were not encouraging. Hence, we implement an MSC preconditioned GMRES with restarts and warm starts [3, p.393] as an iterative solver in the SSBA package, for solving (3). The restart parameter is taken as 40, thus forming a Krylov subspace of 40 vectors. The GMRES algorithm runs as long as the number of iterations is less than 100 or the norm of the relative residual is not less than $10^{-2}$ (as taken in [2]), whichever comes first. The GMRES method is implemented using the dfgmres routine available in the INTEL MKL library, version 2019.4.243. All of the experiments are performed on a subset of problems from the BAL dataset [2]. We run all of the experiments on a machine with Intel Pentium(R) processor and 8 GB of RAM. As all the problems from the BAL dataset cannot fit into memory, we select 8 problems for which the number of points varies from $7K$ to $226K$.

### 4.2 Results

We compare the direct solve, specifically, the LDL factorization method and the restarted GMRES preconditioned with two preconditioners : (1) block Jacobi preconditioner and (2) MSC preconditioner. The problems have been selected from the BAL dataset. We experimented with different number of MSC blocks and found that taking 30 blocks gave optimal results.

In Table 1, the time per LM iteration and the mean reprojection error for various methods are shown. As we have tested mostly on small to medium sized problems, we observe that direct solve is faster than iterative solve for these problems. However, as can be seen from figure 3, the memory requirement for direct solver increases with increase in problem size. Thus, iterative solvers are essential for very large problems. In Table 1, it can be seen that using MSC as a preconditioner results in faster computation time than using block Jacobi as a preconditioner.

Also, from figure 3 it can be seen that the MSC preconditioned GMRES is the most memory efficient of the three methods. In figure 3, as the number of cameras increases, the size of the L, D factors of block G also increases. Thus, doing an LDL factorization of block G during block Jacobi preconditioner solve requires more time compared to that of MSC, as seen in figure 4. Thus, for larger problems, using MSC as a preconditioner is a much better option for a memory constrained system.

## 5 Conclusions and Future Work

We proposed a technique for the approximation of the global Schur complement and used this approximation to design a preconditioner. We showed some preliminary re-

[1] **www.cvg.ethz.ch/research/chzach/opensource.html**

**Table 1.** Average time (in seconds) per iteration for the LM solver using the three methods on 8 problems from the `BAL` datasets using the `bundle_large_lifted_schur` cost function routine in `SSBA`. The time in bold represents the faster of the two preconditioners for iterative solve. The problems are prefixed as : L for `LadyBug`, TS for `Trafalgar Square` and D for `Dubrovnik`.

| `BAL` Dataset Parameters | | | Direct Solve | | Block Jacobi | | MSC(30) | |
|---|---|---|---|---|---|---|---|---|
| Cameras | Points | Observations | Time | Error | Time | Error | Time | Error |
| L-49 | 7776 | 31843 | 0.1 | 0.73 | **0.6** | 0.96 | **0.6** | 0.65 |
| L-138 | 19878 | 85217 | 0.4 | 0.89 | **1.1** | 0.82 | 1.6 | 0.88 |
| TS-225 | 57665 | 208622 | 0.9 | 0.87 | 10.4 | 0.79 | **9.5** | 0.67 |
| D-308 | 195089 | 1045197 | 8.2 | 0.71 | 29.4 | 0.71 | **26.2** | 0.72 |
| D-356 | 226730 | 1255268 | 11 | 0.80 | 41.5 | 0.77 | **30.6** | 0.79 |
| L-372 | 47423 | 204472 | 3.0 | 0.70 | **11.9** | 0.71 | 13.1 | 0.71 |
| L-539 | 65220 | 277273 | 5.4 | 0.74 | 9.5 | 0.80 | **6.7** | 0.77 |
| L-885 | 97473 | 434905 | 11.5 | 0.69 | 25.7 | 0.69 | **17** | 0.71 |

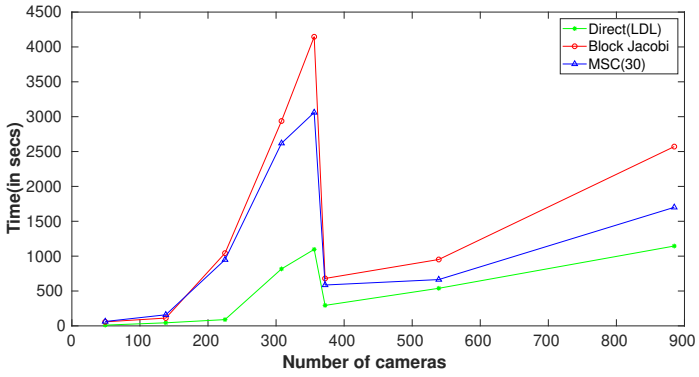


**Fig. 2.** Plot showing the total time taken for 100 iterations of the Levenberg Marquardt algorithm, using `direct` solve, `block Jacobi` preconditioned `GMRES` and `MSC` preconditioned `GMRES`.

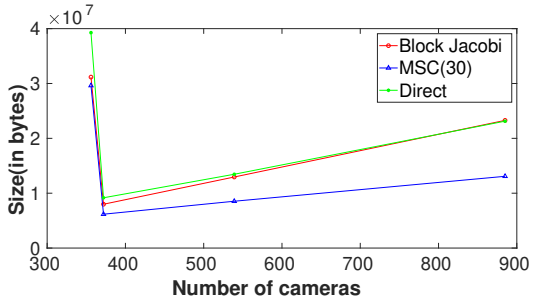


**Fig. 3.** Plot showing the memory requirement for the three methods for the 4 largest problems in our paper, as the number of cameras increases. We have plotted the memory for the three methods : L, D factors for direct solve, L, D factors of block G alongwith the Krylov subspace and the memory for storing the MSC block, its L, D factors as well as the Krylov subspace.

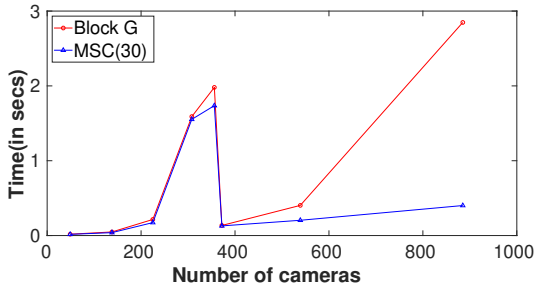


**Fig. 4.** Plot showing the time taken for `LDL` factorization of the G block and the MSC block.

sults which were obtained by implementing our technique as a sequential code. As seen in figure 3, this solver has much less memory requirement than the other methods mentioned in the paper, and is also faster than using block Jacobi as a preconditioner. This makes using MSC as a preconditioner a much better choice. Also, since the MSC blocks are non-overlapping, they are independent of each other. Thus, one possible direction of future work is a parallel implementation of the proposed solver. Another direction would be to assess the robustness of the solver for very large scale problems.

## References

1. Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S.M., Szeliski, R.: Building rome in a day. Commun. ACM **54**(10), 105–112 (Oct 2011)
2. Agarwal, S., Snavely, N., Seitz, S.M., Szeliski, R.: Bundle adjustment in the large. In: ECCV 2010. pp. 29–42. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
3. Boyd, S., Vandenberghe, L.: Introduction to Applied Linear Algebra - Vectors, Matrices, and Least Squares. Cambridge University Press (2018)

4. Brown, D.C.: The bundle adjustment - progress and prospects (1976)
5. Byröd, M., Åström, K.: Conjugate gradient bundle adjustment. In: ECCV 2010. pp. 114–127. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
6. Byröd, M., Åström, K.: Bundle adjustment using conjugate gradients with multiscale preconditioning (01 2009)
7. C.Wu, S.Agarwal, B.Curless, Seitz, S.M.: Multicore bundle adjustment. In: CVPR 2011. pp. 3057–3064 (June 2011)
8. Davis, T.: Suitesparse, http://faculty.cse.tamu.edu/davis/suitesparse.html
9. Frahm, J.M., Fite-Georgel, P., Gallup, D., Johnson, T., Raguram, R., Wu, C., Jen, Y.H., Dunn, E., Clipp, B., Lazebnik, S., Pollefeys, M.: Building rome on a cloudless day. In: ECCV 2010. pp. 368–381. Springer Berlin Heidelberg, Berlin, Heidelberg (2010)
10. Golub, G.H., Van Loan, C.F.: Matrix Computations (3rd Ed.). Johns Hopkins University Press, Baltimore, MD, USA (1996)
11. Jian, Y.D., Balcan, D.C., Dellaert, F.: Generalized subgraph preconditioners for large-scale bundle adjustment. In: Outdoor and Large-Scale Real-World Scene Analysis. pp. 131–150. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
12. Katyan, S., Das, S., Kumar, P.: Two-grid preconditioned solver for bundle adjustment. In: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 3588–3595 (2020)
13. Kumar, P.: Purely algebraic domain decomposition methods for the incompressible navier-stokes equations (2011)
14. Kushal, A.: Visibility based preconditioning for bundle adjustment. In: CVPR 2012 Proceedings. pp. 1442–1449. CVPR '12, IEEE Computer Society, Washington, DC, USA (2012)
15. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. Quarterly of Applied Mathematics **2**(2), 164–168 (1944), http://www.jstor.org/stable/43633451
16. Lourakis, M.I.A., Argyros, A.A.: Sba: A software package for generic sparse bundle adjustment. ACM Trans. Math. Softw. **36**(1), 2:1–2:30 (Mar 2009)
17. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the Society for Industrial and Applied Mathematics **11**(2), 431–441 (1963)
18. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM, second edn. (2003)
19. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. International Journal of Computer Vision **80**(2), 189–210 (Nov 2008)
20. Toselli, A., B. Widlund, O.: Domain Decomposition Methods- Algorithms and Theory. Cambridge University Press (2005)
21. Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment — a modern synthesis. In: Vision Algorithms: Theory and Practice. pp. 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
22. Y.Jeong, D.Nister, D.Steedly, Szeliski, R., I.Kweon: Pushing the envelope of modern methods for bundle adjustment. IEEE TPAMI **34**(8), 1605–1617 (Aug 2012)
23. Zach, C.: Robust bundle adjustment revisited. In: Computer Vision – ECCV 2014. pp. 772–787. Springer International Publishing, Cham (2014)