A Deflation based Fast and Robust Preconditioner for Bundle Adjustment

Shrutimoy Das  Siddhant Katyan  Pawan Kumar
International Institute of Information Technology, Hyderabad
{shrutimoy.das, siddhant.katyan}@research.iiit.ac.in, pawan.kumar@iiit.ac.in

Abstract

The bundle adjustment (BA) problem is formulated as a non-linear least squares problem which requires the solution of a linear system. For solving this system, we present the design and implementation of a fast preconditioned solver. The proposed preconditioner is based on the deflation of the largest eigenvalues of the Hessian. We also derive an estimate of the condition number of the preconditioned system. Numerical experiments on problems from the BAL dataset [3] suggest that our solver is the fastest, sometimes, by a factor of five, when compared to the current state-of-the-art solvers for bundle adjustment.

1. Introduction

Bundle adjustment (BA)[25] is one of the most important steps in the three dimensional (3D) reconstruction systems. In recent years, the focus has shifted to developing 3D reconstruction systems that can handle millions of images[1, 10, 24]. However, as the size and scale of these systems increase, BA becomes computationally very expensive. This has led to a renewed interest in developing large scale bundle adjustment solvers[3, 6, 11, 26, 27].

The BA problem can be formulated as a non-linear least squares problem, which can be solved using classical optimization algorithms. The most commonly used algorithms for these problems is the Levenberg-Marquardt (LM) algorithm. Each iteration of the LM algorithm requires solving a system of equations, which is the most expensive step. Thus, a lot of research has gone into making this step computationally efficient both in terms of time as well as space complexity.

In SBA [17], direct methods using dense Cholesky factorization of the reduced camera matrix has been explored. However, Cholesky factorization has quadratic space complexity, and cubic time complexity in the number of unknowns. This makes it inefficient for large scale problems. Thus, for large problems, the focus has shifted from using direct linear solvers to iterative linear solvers.

One of the most commonly used iterative solvers used in recent works is the Conjugate Gradient(CG) method [3, 6, 7, 11, 26, 27]. The CG based method have a fraction of the memory usage compared to factorization based methods. However, the rate of convergence of these methods depend on how well-conditioned the problem is, and the BA problems are known to be notoriously ill-conditioned with condition number going up to the order of $10^{20}$. To overcome this problem, a suitable preconditioner is used, where preconditioner is an approximation to the inverse of the matrix, which is easy to solve with, and cheap to construct. Using preconditioner often improves the condition number of the coefficient matrix [23]. Applying a preconditioner to the CG method results in the Preconditioned Conjugate Gradients algorithm.

In this paper, we propose a deflation preconditioner that explicitly removes or “deflates” the error components corresponding to the largest eigenvalues of the Hessian. We find that for bundle adjustment, there are very large eigenvectors, that slows down the convergence of an iterative method such as CG or GMRES [23]. We also show that the condition number is reduced as a result of deflation, thus providing an estimate of the condition number. Also, a modified version of the deflated two-grid [12] is presented. In the experiments, we show that the proposed solver is better than the state-of-the-art solvers implemented in popular ceres-solver [2]. In some cases, it is up to five times faster. The memory requirement for the preconditioner is approximately $kn$, where $n$ is the number of rows of the Hessian and $k$ is the number of eigenvalues deflated. Hence, it has the least memory requirement of the existing solvers.

In Section 2, we have a brief discussion on the problem of bundle adjustment. We also discuss the LM method, which is used for solving the optimization problem in bundle adjustment. In Section 3, we propose a deflation based preconditioner for solving the linear system in each iteration of the LM algorithm. Additionally, we derive some useful properties of the proposed preconditioner. We explain the implementation details of the proposed method in Section 4. In Section 5, we discuss the results obtained with the deflation preconditioner and compare it with state-of-the-art solvers. We conclude our paper in Section 6.
2. Bundle Adjustment

Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameters (camera pose and/or calibration) estimates. For more details, see Triggs et al. [25]

Let us assume that the parameter vector \( x \) for the 3D reconstruction problem has the block structure \( x = [y_1, \ldots, y_p, z_1, \ldots, z_q] \), where \( y \) and \( z \) correspond to the point (structure) and camera (viewing) parameters, respectively, such that there are \( p \) points and \( q \) cameras. Let \( r_k(x) \), where \( k \in \{1, \ldots, q\} \), be the measurement function of a 3D point in a camera \( k \). Let \( m_k \) be the observed measurement of a 3D point in camera \( k \). Let \( f_k(x) = r_k(x) - m_k \) be a cost function for the error in the computed measurement, \( r_k(x) \), and the observed measurement, \( m_k \). Define \( F(x) = [f_1(x), \ldots, f_q(x)]^T \). Then, the objective for the bundle adjustment problem can be stated as

\[
x^* = \arg\min_x \frac{1}{2} ||F(x)||^2. \tag{1}
\]

The objective function (1) is non-linear and is usually optimized by trust region methods [8], one of which is the Levenberg-Marquardt algorithm described below.

2.1. Levenberg Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm [4, 16, 19] is one of the most commonly used algorithms for solving non-linear least squares problems. Ceres\cite{Ceres}, which is an open-source C++ library for modelling and solving optimization problems, implements both an exact step\cite{exactstep}, and an inexact step [21, 22] variant of the LM algorithm.

Define \( J = \frac{\partial F}{\partial x} \big|_{x=x_t} \) as the Jacobian of \( F(x) \) at the current iterate \( x_t \). The LM algorithm forms an affine approximation \( F(x_t + \Delta x) \approx F(x_t) + J \Delta x, \) of the objective (1) at \( x_t \) in every iteration. The next iterate is then taken to be the minimizer of \( ||F(x_t + \Delta x)||_2^2 \). To penalize choices of \( x_{t+1} \) that may not be in the neighbourhood of \( x_t \), a trust penalty term \( \lambda^T \text{diag}(J^T J) \Delta x \) is introduced. The term \( \lambda \) is the trust parameter which is updated after each LM iteration. Then, the next iterate of the LM algorithm is obtained as shown below.

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

The quality of this minimizer is measured by the ratio of the actual decrease in the objective function to the decrease in the value of the affine approximation. Now, from (2), let \( H_{LM} = J^T J + \lambda^T \text{diag}(J^T J) \) be the approximated Hessian of \( F(x) \) at \( x_t \). Also, let \( g = J^T F(x_t) \) and \( \Delta x = x_{t+1} - x_t \). Then, rearranging the terms in equation (2), we have

\[
H_{LM} \Delta x = -g. \tag{3}
\]

Solving the system of equations in (3) is the most expensive computational step in each iteration of the LM algorithm. In general, for solving small to medium scale problems, direct linear solvers can be used for (3). However, these methods are not scalable for large problems. As the problem size increases, iterative solvers become more efficient, and often the only feasible choice for solving this system. By exploiting the special (sparse) structure and properties of the Hessian in BA problems, we design an efficient and scalable scheme for solving (3).

2.2. Structure of the Hessian

As mentioned before, suppose the reconstruction problem consists of \( p \) points and \( q \) cameras, such that the parameter vector \( x \) has the block structure as mentioned in the beginning of section 2. Let each point block be of size \( s (s = 3 \) for most problems), and each camera block be of size \( c \) (usually \( c \in \{6, \ldots, 9\} \)). Given these block sizes, the Jacobian \( J \) can be partitioned into a point part \( J_p \) and camera part \( J_c \) as \( J = [J_p; J_c] \). Using this partitioning scheme, the Hessian \( H_{LM} \) can be represented as

\[
H_{LM} = \begin{bmatrix} J_p^T J_p & J_p^T J_c \\ J_c^T J_p & J_c^T J_c \end{bmatrix} = \begin{bmatrix} D & L^T \\ L & G \end{bmatrix}, \tag{4}
\]

where \( D \in \mathbb{R}^{ps \times ps} \) is a block diagonal matrix with \( p \) blocks of size \( s \times s \) and \( G \in \mathbb{R}^{qc \times qc} \) is a block diagonal matrix with \( q \) blocks of size \( c \times c \). The matrix \( L \in \mathbb{R}^{qc \times ps} \) is a general block sparse matrix. Then, the system in (3) can be written as a block structured linear system as

\[
\begin{bmatrix} D & L^T \\ L & G \end{bmatrix} \begin{bmatrix} \Delta x_p \\ \Delta x_c \end{bmatrix} = \begin{bmatrix} g_p \\ g_c \end{bmatrix}, \tag{5}
\]

where \( \Delta x = [\Delta x_p; \Delta x_c] \), with \( \Delta x_p \) and \( \Delta x_c \) corresponding to point and camera parameter blocks of \( \Delta x \), respectively, and \( g = [g_p; g_c] \) with \( g_p \) and \( g_c \) the corresponding point and camera parameter blocks of \( g \) respectively. The structure of a Hessian of the Ladybug 49 problem from BAL dataset is shown in Figure 1.

2.3. Related Work

The block structured linear system in (5) can be solved either via direct methods or via iterative methods. In [5], Brown introduced the method of the reduced bundle system, which utilizes the structure of the linear system (5). In this method, the linear system is split into a reduced camera system, which involves computing the Schur complement of \( H_{LM} \), and a reduced structure system. For solving the reduced camera system, [17] uses a Cholesky factorization of the coefficient matrix of the reduced system, and then solves the reduced structure system using back-substitution. However, this method does not scale satisfactorily as the problem size increases. Thus, a better alternative is to use iterative methods for these problems.

The most popular iterative method used for solving (5) is the Conjugate Gradients (CG) method. Since this method requires only matrix-vector products, it has a much less memory requirement compared to direct methods. However, the convergence of
these methods depend on how well-conditioned the original problem is. This has led to recent research for obtaining efficient preconditioners for these methods. In [3], the camera submatrix has directly with a preconditioner based on incomplete QR factorization. The band block diagonal of the Schur complement of \( H_{LM} \) is used as a preconditioner in [27]. In [15], the visibility information in the scene is used to form block diagonal and block tridiagonal preconditioners. A generalized subgraph preconditioning (GSP) technique based on the combinatorial structure of the BA problem is explored in [11].

Yet another iterative method, the Generalized Minimal Residual (GMRES) method, has been used for solving (5). Inspired from [13], in [9], a preconditioner based on the approximation of the Schur complement of \( H_{LM} \) is explored. Recently, inspired from multigrid, another preconditioner based on the two-grid method has been explored in [12, 14]. A modified version of the preconditioner proposed in [12] is also compared in the experiments.

The direct methods are usually faster than iterative methods for small to medium sized problems. However, the iterative methods become faster and are more memory efficient as the problem sizes increase. In the following we explore preconditioners suitable for very large problems.

### 3. Deflation Preconditioner

In this paper, a purely deflation based preconditioner [20] is proposed for solving (3). It is motivated by the fact that for bundle adjustment problems, a few of the largest eigenvalues contribute to the very high condition numbers typically associated with these problems. Also, a modified version of the deflated two-grid [12] is presented. A condition number estimate of the proposed solver is also presented in this work. In the remainder of this section, the notations \( A \) and \( H_{LM} \) will be used interchangeably for the Hessian.

A good choice for the deflation preconditioner is the \( B_{def} \) defined in [12], which is given as

\[
B_{def}^{-1} = PA_c^{-1}R_c
\]

where \( P \in \mathbb{R}^{n \times k} \) is prolongation operator and the restriction operator is defined as \( R = P^T \). The prolongation operator is constructed by taking the eigenvectors corresponding to the \( k \) largest eigenvalues of \( A \) such that \( \lambda_1 > \lambda_2 > \ldots > \lambda_k \). Also, \( A_c \) is the coarse grid matrix and is defined as

\[
A_c = RAP = P^T AP = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{bmatrix}
\]

In [12], \( B_{def} \) is used with a smoother as a multiplicative combination. It is then used as a preconditioner with GMRES. However, as the following lemma shows, \( B_{def}^{-1} \) is symmetric semidefinite and rank deficient. Thus, it cannot be used as a standalone preconditioner. In [12], the Jacobi preconditioner is used in a multiplicative combination with \( B_{def}^{-1} \) due to which the resulting preconditioner is nonsingular.

In the following, \( \langle x, y \rangle = x^T y \) denotes an inner product on the vector space \( \mathbb{R}^n \), and SPD denotes symmetric positive definite.

**Lemma 3.1.** Let the given matrix \( A \) be SPD. Then, the deflation preconditioner defined in (6) is symmetric and positive semidefinite. Also, it is rank deficient, i.e., \( \text{rank}(B_{def}^{-1}) \leq k \), where \( k \) is the number of deflated eigenvectors.

**Proof.** From the construction of \( A \) given in (7), it can be seen that \( A_c = RAP = P^T AP = A_c^T \), i.e., \( A_c \) is symmetric. Now, \( (B_{def}^{-1})^T = (PA_c^{-1}P^T)^T = P(A_c^{-1})^TP^T = PA_c^{-1}P^T = B_{def}^{-1} \), since \( A_c \) is symmetric. Thus, \( B_{def}^{-1} \) is symmetric.

It can be shown that for \( z \in \mathbb{R}^k \) and \( z \neq 0 \), \( A_c \) is positive definite

\[
z^T A_c z = \langle A_c z, z \rangle = \langle P^T AP z, z \rangle = \langle A(Pz), Pz \rangle = y^T Ay > 0, \text{ since } y = Pz \neq 0.
\]

The prolongation operator, \( P \), is full rank, hence, for \( z \neq 0 \), we have \( Pz \neq 0 \). Since, \( A_c \) is SPD, \( A_c^{-1} \) is also SPD.

Now, for \( x \in \mathbb{R}^n \) and \( x \neq 0 \).
\[ x^T B^{-1}_{de} x = \langle B^{-1}_{de} x, x \rangle = \langle P A^{-1}_c P^T x, x \rangle = \langle A^{-1}_c (P^T x), P^T x \rangle. \]

Since \( A \) is SPD, it has \( n \) independent and orthogonal eigenvectors. Now, if \( x \) is some eigenvector of \( A \) different from the \( k \) eigenvectors in \( P \), then \( P^T x = 0 \) even if \( x \neq 0 \). Thus, we can at least conclude that \( x^T B^{-1}_{de} x \geq 0 \), i.e., \( B^{-1}_{de} \) is positive semi-definite. Also, \( \text{rank}(PA^{-1}_c P^T) \leq \min(\text{rank}(P), \text{rank}(A^{-1}_c P^T)) \leq k \), where \( k \) is the rank of \( P \), since all \( k \) eigenvectors (which are columns of \( P \)) are linearly independent.

In this paper, a new standalone deflation preconditioner \( B_D \) is defined as shown below

\[ B_D = I - AP (P^T AP)^{-1} P^T, \]  \hspace{1cm} (10)

where the columns of \( P \in \mathbb{R}^{n \times k} \) span the deflation space and \( I \in \mathbb{R}^{n \times n} \) is the identity matrix. It should be noted that the matrix \( P \) in this case is not the prolongation operator used for \( B_{def} \).

**Lemma 3.2.** The preconditioner \( B_D \) defined in (10) is a projection matrix, i.e.,

\[ B_D^2 = B_D \]  \hspace{1cm} (11)

**Proof.**

\[
\begin{align*}
B_D^2 &= B_D B_D \\
&= (I - AP (P^T AP)^{-1} P^T)(I - AP (P^T AP)^{-1} P^T) \\
&= I - 2AP (P^T AP)^{-1} P^T + AP (P^T AP)^{-1} P^T \\
&= I - AP (P^T AP)^{-1} P^T \\
&= B_D
\end{align*}
\]

\[ \square \]

**Lemma 3.3.** For \( B_D \) defined in (10), the following identity holds

\[ AB_D^T = B_D A. \]  \hspace{1cm} (13)

**Proof.**

\[
\begin{align*}
AB_D^T &= A(I - AP (P^T AP)^{-1} P^T)^T \\
&= [I - AP (P^T AP)^{-1} P^T] A \\
&= B_D A
\end{align*}
\]

\[ \square \]

Similar to [12], the columns of \( P \) contain the eigenvectors of \( A \). Thus, rank of \( P \) is exactly \( k \), which is much less than \( n \). Given this definition of \( P \) and \( A_c = P^T AP \), \( A_c \) is SPD, as seen in Lemma 3.1. Again, the matrix \( A_c \) defined with respect to the deflation preconditioner has different significance from the coarse grid matrix defined in (7).

Let the right hand side of the linear system to be solved be \( b \). Consider the deflated system

\[ B_D Ax = B_Db. \]  \hspace{1cm} (15)

The system (15) can be solved using CG or GMRES to get an approximate solution \( \hat{x} \). Using the identity given in Lemma 3.3, this can be written as

\[
\begin{align*}
AB_D^T \hat{x} &= B_Db \\
\Rightarrow B_D^T \hat{x} &= A^{-1} B_Db \\
\Rightarrow B_D^T \hat{x} &= A^{-1}(I - AP (P^T AP)^{-1} P^T)b \\
\Rightarrow B_D^T \hat{x} &= A^{-1}b - P (P^T AP)^{-1} P^T b \\
\Rightarrow A^{-1}b &= PA^{-1}_c P^T b + B_D^T \hat{x} \\
\Rightarrow \hat{x} &= PA^{-1}_c P^T b + B_D^T \hat{x}.
\end{align*}
\]

Thus, the solution of linear system of the form \( Ax = b \) with the deflation preconditioner \( B_D \), is given by

\[ x = PA^{-1}_c P^T b + B_D^T \hat{x}. \]  \hspace{1cm} (17)

The algorithm for solving a linear system with the deflated preconditioner \( B_D \) is given in Algorithm 1.

**Algorithm 1** Solving \( Ax = b \) with \( B_D \) as a preconditioner

**Input:** \( A, b \)

**Output:** \( x \)

1: Set \( \bar{b} = B_Db \)
2: Solve \( B_D \bar{a} = \bar{b} \) using GMRES
3: Set \( \hat{x} = PA^{-1}_c P^T b \), where \( A_c = P^T AP \)
4: Set \( x = B_D^T \hat{x} \)
5: \( x = \hat{x} + \bar{x} \)

Contrary to the typical assumption of deflating eigenvectors corresponding to the smallest eigenvalues, the eigenvectors corresponding to the largest eigenvalues are deflated in this work. As already seen in previous sections, the matrices related to the BA problem have some eigenvectors which correspond to very large eigenvalues and are well spread out. Hence, the largest eigenvalues are deflated. An analysis of the largest eigenvalues for a given matrix is given in Section 5.

For iterative methods, the speed of convergence of the method depends upon the condition number of the coefficient matrix. The following theorem shows the condition number bounds on the deflation based preconditioned matrix. Assume that the eigenvectors of \( A \), \( \{v_1, v_2, \ldots, v_k, v_{k+1}, \ldots, v_n\} \) are arranged in increasing order of their corresponding eigenvalues \( \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \). The eigenvalues corresponding to the \( k \) largest eigenvalues are taken as the columns of the matrix \( P \), i.e., \( P = [v_{n-k+1}, \ldots, v_n] \). The span of these columns form the deflation space. Since, the Hessian in BA problems is SPD, the eigenvectors \( v_i \) are orthogonal, i.e,

\[ v_i^T v_j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases} \]  \hspace{1cm} (18)

Given the definition of \( P \), the effective condition number is defined as
\[ \kappa_{eff} = \frac{\lambda_{n-k}}{\lambda_1} \]  

(19)

where \( \lambda_{\text{large}} \) and \( \lambda_{\text{small}} \) are the largest and smallest non-zero eigenvalues of \( A \) respectively.

Then, the following theorem can be stated.

**Theorem 3.1.** For the deflation precondioner \( B_D \) defined in (10) with \( P \in \mathbb{R}^{n \times n} \), the deflation space contains the eigenvectors corresponding to the \( k \) largest eigenvalues of \( A \in \mathbb{R}^{n \times n} \) and the spectrum of \( B_D A \) is \( \{ \lambda_1, \ldots, \lambda_{n-k}, 0, \ldots, 0 \} \). Then, the effective condition number of \( B_D A \) is

\[ \kappa_{eff} = \frac{\lambda_{n-k}}{\lambda'_1} \]  

(20)

**Proof.** As mentioned previously,

\[ A_c = P^T AP = \begin{bmatrix} \lambda_{n-k+1} & \lambda_{n-k+2} & \cdots & \lambda_n \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{n-k} & \lambda_{n-k+1} & \cdots & \lambda_n \end{bmatrix} . \]  

(21)

For any arbitrary vector \( v_r \), which is the \( r \)th eigenvector of \( A \), we have

\[ B_D A v_r = (I - A PA_c^{-1} P^T) A v_r \]

\[ = \lambda_r v_r - AP \text{diag} \left( \frac{1}{\lambda_{n-k+1}}, \ldots, \frac{1}{\lambda_n} \right) P^T \lambda_r v_r \]

\[ = \lambda_r v_r - \lambda_r \left[ \lambda_{n-k+1} v_{n-k+1} \ldots \lambda_n v_n \right] \begin{bmatrix} v_{n-k+1}^T v_r / \lambda_{n-k+1} \\ \vdots \\ v_n^T v_r / \lambda_n \end{bmatrix} \]

\[ = \lambda_r v_r - \lambda_r \left[ (v_{n-k+1}^T v_r) v_{n-k+1} + \ldots + (v_n^T v_r) v_n \right]_{n \times 1} . \]  

(22)

It can be observed that if \( v_r \) is not the deflated eigenvector, i.e., if \( r < n - k + 1 \), the second term is zero due to the orthogonality of these eigenvectors, as shown in (18). On the other hand, if \( v_r \) is one of the deflation vectors, i.e., if \( n - k + 1 \leq r \leq n \), then, by the construction of \( B_D \), we have \( B_D A v_r = 0 \). That is

\[ B_D A v_r = \begin{cases} \lambda_r v_r, & \text{for } r = 1, \ldots, n-k \\ 0, & \text{for } r = n-k+1, \ldots, n. \end{cases} \]  

(23)

Hence, the effective condition number of \( B_D A \) is

\[ \kappa_{eff} = \frac{\lambda_{n-k}}{\lambda_1} . \]

\[ \square \]

4. Implementation

The deflation space, i.e., the span of the columns of \( P \) are the eigenvectors corresponding to the 2 largest eigenvalues of \( H_{LM} \). These eigenvectors of \( H_{LM} \) are computed using the double precision Intel MKL routine mkl_sparse_d_ev, which requires the input matrix to be in CSR format with 1-based indexing. As observed before, since the columns of \( P \) are the eigenvectors of \( H_{LM} \), computing \( A_c = P^T AP \) is trivial because, \( A_c = \text{diag}(\lambda_1, \lambda_2) \) is a diagonal matrix of size \( 2 \times 2 \). Hence, any matrix-matrix operations for computing \( A_c \) are avoided.

For the LM iterations, the popular Ceres solver [2] is used. The proposed solver is integrated in the Ceres framework. The stopping criteria for the LM iterations is either of the following

- The number of iterations exceeds 100, or,
- The termination rule is satisfied

\[ \| H_{LM} \Delta x + g \| \leq \eta_t \| g \| \]  

(24)

where \( t \) is the LM iteration number and \( \eta_t = 10^{-8} \).

The upper bound for the total reprojection error is guaranteed by (24) for all the solvers compared in this paper.

The proposed solver is compared with the following solvers/preconditioners, which are already available in Ceres.

- **Sparse Normal Cholesky (SNC):** This solver uses the Cholesky factorization of \( H_{LM} \) for solving (3). It is a direct solver.
- **Dense Iterative Schur (DS):** This solver solves the reduced camera system using CG, and constructs the dense Schur complement.
- **Sparse Iterative Schur (SS):** This solver solves the reduced camera system using PCG, but using a sparse approximated Schur complement system. The diagonal blocks of the Schur complement, termed as SCHUR_JACOBI, is used as the preconditioner.
- **Sparse Schur (SS):** This solver solves the reduced camera system using sparse direct methods on a sparse approximation of the Schur complement.

In this work, the following solvers, along with GMRES, are integrated into the Ceres framework:

- **Modified Two Grid (MTG):** Rather than doing a fill-reducing reordering of the blocks, as done in SS, here the blocks are selected directly without applying any reordering.
- **Deflation (DEF):** This is the method proposed in this paper.

These preconditioners are used with restarted GMRES instead of CG. This is because for both DEF and MTG, GMRES is more robust compared to CG. The restart parameter for GMRES is 5. The stopping criterion for GMRES is either of the following

- The norm of the relative residual is below \( 10^{-2} \), or,
- The number of iterations exceeds 10.

The GMRES method is implemented using the dfsgmres routine of Intel MKL. All the experiments are performed on a subset of problems from BAL dataset. The number of points in these problems varies from 65K to 993K. These tests were performed on a system with Intel Xeon CPU E5-2640 @ 2.40GHz and 20GB RAM. The details of the problems used for the experiments are given in Table 1.
5. Results

In Figure 2, the magnitude of the top 10 eigenvalues of the Hessian resulting from the Venice-1776 problem is shown. It can be seen that the first two eigenvalues are much larger than the other eigenvalues. Thus, deflating these two eigenvalues results in a reduction of the effective condition number, as proven in Theorem 3.1. The number of eigenvalues to be deflated, $k$, varies from problem to problem. For the problems considered in this paper, it was found that deflating the 2 largest eigenvalues is enough. If the number of eigenvalues to be deflated is kept small, then there is a significant gain in setup time as the time taken for computing the eigenvalues is reduced. Another reason for deflating the largest eigenvalues instead of the smallest eigenvalues is that computing the smallest eigenvalues takes more time than computing the largest eigenvalues.

We note that deflating smaller eigenvalues can also reduce the effective condition number, unfortunately, as shown in right of Figure 3, time for computing two smallest eigenvalues for the three problems ranges from 900 seconds to approximately 3000 seconds. Thus, the time for computing smallest eigenvectors does not amortize the total cost involved with LM iterations that require multiple linear solves and hence multiple eigenvector computations, thereby providing no benefits compared to other solvers. However, time for computing largest eigenvectors as shown on the left of Figure 3, for the three problems ranges from 0.7 seconds to 3.7 seconds. Hence, deflating largest eigenvectors is not much costly, and moreover, it helps in attaining overall faster convergence in time compared to existing solvers.

In Table 2, the time taken for each LM iterations using direct and iterative solvers for the linear system, are compared. It can be seen that, except for dataset 2, the deflation preconditioner, denoted by DEF is the best for all the datasets. In some cases, for example, for dataset 1, it is almost 4 times faster than the next best solver, MTG. Similarly, for dataset 4, DEF is almost 5 times faster than the next best solver, that is SNC. For the largest dataset, V-1776, DEF is almost 4 times faster than SNC. Also, it can be seen that the MTG solver is always faster than the TG solver proposed in [12]. In Figure 4, the bar plot of these time comparisons is shown for 100 iterations of the LM method.

6. Conclusion and Future Work

In this paper we proposed a deflation preconditioner for the efficient solution of the linear equations related with the LM method. To the best of our knowledge, this is the fastest solver proposed so far, moreover, this is the first time a purely deflation based solver have been investigated for the bundle adjustment problem. Given that the problems related to bundle adjustment are often very ill conditioned, with very high condition number, we establish that explicit deflation helps in designing efficient solvers for these problems. In particular, we observed that deflating some of the largest eigenvectors instead of the smallest ones is beneficial, as the largest eigenvectors tend to slow down the convergence of
Table 2. Average time (in seconds) per iteration of LM with various linear solvers. Here, \( \text{SNC} \) stands for Sparse Normal Cholesky, \( \text{DS} \) stands for Dense Schur, \( \text{IS} \) stands for Iterative Schur, \( \text{SS} \) stands for Sparse Schur, \( \text{TG} \) stands for Two Grid [12], \( \text{MTG} \) stands for Modified Two Grid, and \( \text{DEF} \) stands for the Deflation preconditioner proposed in this work. Also, \( > \) indicates that the LM iterations did not converge within 100 iterations.

Krylov subspace methods like GMRES. We also observe that deflating smallest eigenvectors is not feasible because it is too costly to compute compared to largest. In future, we would like to explore various deflation strategies to further improve the solver.

7. Acknowledgement

We would like to thank the reviewers for their constructive suggestions that helped improve the paper. This work was done at IIIT, Hyderabad; we acknowledge the facilities and grants provided to us that led to this work. We also thank Sameer Agarwal (Google research) for helpful discussions.

References


Figure 4. Bar plot of time comparisons for various solvers. Note that for dataset 2, the LM iterations do not converge using DEF solver, and for dataset 6, the LM iterations do converge using IS solver. Here, SNC stands for Sparse Normal Cholesky, DS stands for Dense Schur, IS stands for Iterative Schur, SS stands for Sparse Schur, TG stands for Two Grid [12], MTG stands for Modified Two Grid, and DEF stands for the Deflation preconditioner proposed in this work.


