Two-Grid Preconditioned Solver for Bundle Adjustment

by

Pawan Kumar, Shrutimoy Das, Siddhant Katyan

in

IEEE

Report No: IIIT/TR/2020/-1

Centre for Security, Theory and Algorithms
International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2020

Two-Grid Preconditioned Solver for Bundle Adjustment

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

Abstract

We present the design and implementation of Two-Grid Preconditioned Bundle Adjustment (TPBA), a robust and efficient technique for solving the non-linear least squares problem that arises in bundle adjustment. Bundle adjustment (BA) methods for multi-view reconstruction formulate the BA problem as a non-linear least squares problem which is solved by some variant of the traditional Levenberg-Marquardt (LM) algorithm. Most of the computation in LM goes into repeatedly solving the normal equations that arise as a result of linearizing the objective function. To solve these system of equations we use the Generalized Minimal Residual (GMRES) method, which is preconditioned using a deflated algebraic two-grid method. To the best of our knowledge this is the first time that a deflated algebraic two-grid preconditioner has been used along with GMRES, for solving a problem in the computer vision domain. We show that the proposed method is several times faster than the direct method and block Jacobi preconditioned GMRES.

1. Introduction

Recent work in Structure-from-Motion (SfM) problem has moved towards three-dimensional (3D) reconstruction from large-scale photo collections of high quality and high resolution images [2, 8, 12]. Given that the feature matching is already done by the existing bundlers [6], bundle adjustment (BA) is the key component in most of the SfM systems. The bottleneck in SfM systems is the BA process, which is the joint non-linear optimization of camera parameters, and the 3D points to minimize the mean reprojection error.

By formulating the BA problem as a non-linear least squares problem, classical algorithms can be used for solving these problems. One of the most commonly used algorithm is the Levenberg-Marquardt (LM) algorithm, with some variations. Each iteration of the LM algorithm involves solving a system of normal equations, which is the most expensive step. Thus, making this computation cheaper has become a very important problem.

In this paper, we use the restarted generalized minimal residual (GMRES) method as the iterative solver. Similar to Conjugate Gradient (CG), GMRES alone does not converge fast for these problems, and thus, requires a suitable preconditioner. The reason for using preconditioned GMRES instead of Preconditioned Conjugate Gradient (PCG) is that the preconditioned operator is unsymmetric. We propose a deflated algebraic two-grid method [35] for constructing the preconditioner. A two-grid method consists of two components: a coarse grid correction and a smoother. The coarse grid correction damps the low frequency part of the error and the smoother damps the high frequency part of the error.

We replace the coarse grid correction with a deflation preconditioner which explicitly removes the high frequency components from the error. Furthermore, the smoother damps the remaining high frequency components from the error. Hence our two-grid approach is different from the ones used in the scientific computing domain. When coupled with an inexact Levenberg-Marquardt algorithm [22], our preconditioned solver gives state of the art performance on the `BAL` dataset [1].

The rest of the paper is organized as follows. Section 2 presents a brief overview of the bundle adjustment problem and recent work on the use of preconditioned iterative methods for solving it. Section 3 describes the design and implementation of deflated algebraic two-grid preconditioner for GMRES. Section 4 compares our new preconditioning technique to the state of the art methods using problems from the `BAL` dataset. We conclude with a discussion in section 5.

2. Bundle Adjustment

Given a set of measured image feature locations and correspondences, bundle adjustment aims to find the $3D$ point positions and camera parameters that minimize the reprojection errors. For more details, see Triggs et al. [3].

Let us assume that the SfM problem consists of p points and q cameras and the parameter vector x has the block structure $x = [y_1 \cdots y_p, z_1 \cdots z_q]$ where y and z correspond to the point and camera parameters, respectively. Let $r_k(x)$, where $k \in 1 \cdots q$, be the measurement function of a

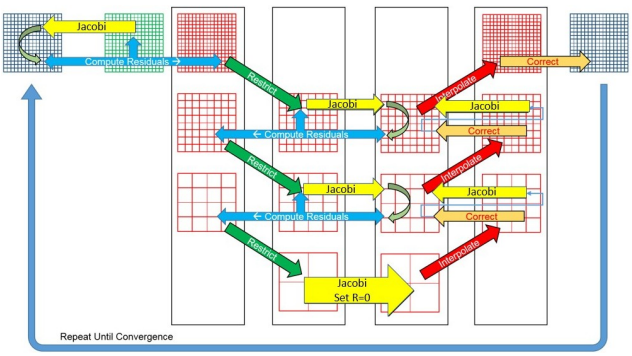

Figure 1. Multigrid preconditioning with Jacobi as smoother

$3D$ point in a camera k. Let m_k be the measurement of a $3D$ point in camera k. Define a cost function $f_k(x)$ as

$$f_k(x) = r_k(x) - m_k, k = 1 \cdots q$$

Define $F(x) = [f_1(x), \cdots, f_q(x)]^T$. Then, the bundle adjustment problem can be stated as

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

The objective function in (1) is non-linear and can be minimized by the standard non-linear least squares algorithm described below.

2.1. Levenberg Marquardt Algorithm

The Levenberg-Marquardt (LM) [10, 11] method has become a standard algorithm for solving the nonlinear least-squares problem, and widely adopted in various disciplines. It has become very popular due to its relative ease of implementation, and its use of an effective damping strategy that lends it the ability to converge promptly from a wide range of initial guesses.

For solving the non-linear least squares problem in (1), each iteration of the LM algorithm forms an affine approximation of the cost function $F(x)$ at the current iterate x_t and takes a step towards the minimizer of this approximation. The first order Taylor approximation of $F(x)$ in a neighbourhood Δx around x_t is given by

$$F(x_t + \Delta x) \approx F(x_t) + J\Delta x,$$

where $J = \frac{\partial F}{\partial x}|_{x=x_t}$ is the Jacobian of $F(x)$. The next iterate x_{t+1} is then taken to be the minimizer of $||F(x_t + \Delta x)||^2$. However, this minimizer may be far from the current iterate x_t, in which case the approximation $F(x_t) \approx F(x_t + \Delta x)$ does not hold. Thus, for minimizing $||F(x_t + \Delta x)||^2$ as well as keeping Δx small, the next iterate is taken to be the minimizer of $\frac{1}{2}||F(x_t + \Delta x)||^2 + \mu^t||\mathtt{diag}(J^T J)\Delta x||^2$, where μ^t is a non-negative damping parameter for the t^{th} iteration. The μ^t term is updated after each LM iteration based on how well J approximates $F(x)$. Then, the next iterate of the LM algorithm is updated as shown below.

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

The quality of this fit is measured by the ratio of the actual decrease in the objective function to the decrease in the value of the linearized model $l(\Delta x) = \frac{1}{2}||F(x_t + \Delta x)||^2$.

Further, from (2), let

$$H_{LM} = J^T J + \mu^t \mathtt{diag}(J^T J) \tag{3}$$

and

$$g = J^T F(x_t)$$

Here, the construction of H_{LM} approximates the Hessian of $F(x)$. Then, it can be seen that solving (2) is equivalent to solving the following system of *normal* equations.

$$H_{LM}\Delta x = -g, \text{ where } \Delta x = x_{t+1} - x_t. \tag{4}$$

Solving (4) is the dominant computational cost in each iteration of the LM algorithm. In general, for small to medium scale problems it is recommended to use QR factorization for solving (4). But it is not scalable for large problems. As the Hessian in the BA problem has a sparse block structure, it can be exploited to construct a more efficient and scalable scheme for solving (4).

2.2. Structure of the Hessian

As mentioned at the beginning of section 2, it is assumed that the SfM problem consists of p points and q cameras, and the parameter vector x has the block structure $x = [y_1, \cdots, y_p, z_1 \cdots z_q]$ where y and z are the point and camera parameter vectors, respectively. Each point block is of size s and each camera block is of size c (where $c \in \{6, 7, 8, 9\}$ and $s = 3$ for most problems). Using these block sizes, it is possible to partition the Jacobian J into a point part J_s and camera part J_c as $J = [J_s; J_c]$, which gives

$$H_{LM} = \begin{bmatrix} J_s^T J_s & J_s^T J_c \\ J_c^T J_s & J_c^T J_c \end{bmatrix} = \begin{bmatrix} D & L^T \\ L & G \end{bmatrix}, \tag{5}$$

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, (4) can be re-written as a block structured linear system as follows

$$\begin{bmatrix} D & L^T \\ L & G \end{bmatrix} \begin{bmatrix} \Delta x_s \\ \Delta x_c \end{bmatrix} = \begin{bmatrix} g_s \\ g_c \end{bmatrix}, \tag{6}$$

where $\Delta x = [\Delta x_s; \Delta x_c], \Delta x_s \text{and} \Delta x_c$ correspond to point parameter blocks and camera parameter blocks of Δx, respectively, and $g = [g_s; g_c]$, g_s and g_c correspond to point parameter blocks and camera parameter blocks of g, respectively. Various approaches have been proposed for solving (6), by exploiting the special structure of the Hessian as described below.

2.3. Related Work

Direct methods for solving (6) has been well studied in literature [3, 4]. Brown, in [20], introduced the method of the reduced bundle system, which is motivated by the special structure of the

Hessian, as shown in (5). Using this method, the system in (6) is split into a *reduced camera* system, which involves constructing the Schur complement of H_{LM}, and a *reduced structure* system. A common method [4] is to use Cholesky factorization to solve the reduced camera system and use back-substitution to solve the reduced structure system. However, this method does not scale satisfactorily for large scale problems.

In recent works, the use of iterative methods such as the conjugate gradient (CG) method for solving large scale bundle adjustment problems has been observed. It requires less memory than direct methods as it involves only matrix-vector multiplications. However the success of CG method depends on the number of iterations required to converge, which in turn depends on how well-conditioned the original problem is. Therefore, recent research has focused on obtaining efficient preconditioners for CG.

Agarwal et al. [1] examined the performance of several standard preconditioners and implementation strategies on large-scale datasets. Bryöd and Aström avoided the computation of H_{LM} and S, and instead ran CG on J directly with an incomplete QR factorization based preconditioner [7]. Jeong et al. proposed using the band block diagonals of the Schur complement matrix S as preconditioners. They observed that amongst the various banded preconditioners, the block-Jacobi preconditioner was a cheap and robust choice [14]. Wu et al. [24] extended the work in [1] to a multicore Jacobian free bundle adjustment method.

More recently, Avanish et al. [18], using the visibility information in the scene, cluster the cameras into tightly interacting clusters. These clusters formed the basis of block diagonal and block tridiagonal preconditioners. Dellaert et al. [19] explored generalized subgraph preconditioning (GSP) technique which is based on the combinatorial structure of the bundle adjustment problem. Motivated by the work in combinatorial preconditioning, the authors have proposed using a low-stretch spanning tree approximations to H_{LM} as preconditioners for (6). However, as pointed out in [19]: "when the additional edges are added to the subgraph, not only the subgraph takes longer time to build, but also the preconditioner becomes more expensive to apply in the CG method." Moreover, domain decomposition based methods [44] have been tried recently for such problems in [43].

Usually direct methods converge faster than these methods for small to medium size problems. In this paper, we show that our method converges faster than direct methods and traditional preconditioned iterative methods even for these sizes, to achieve a comparable mean reprojection error.

3. Two-Grid Preconditioned GMRES

In the following sections, sometimes, we use the notation A to denote H_{LM}. The Generalized minimal residual method [25, 26] is very popular for solving large linear system of equations, $Ax = b, A \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$. The kth GMRES iterate x_k minimizes, with $x_0 = 0$, the norm of the kth residual vector $r_k = b - Ax_k$ over all vectors in the Krylov subspace $K_k(A, b) \equiv \text{span}\{b, Ab, ..., A^{k-1}b\}$. Therefore, residual norms are non-increasing and satisfy

$$\|r_k\| = \min_{p\in\pi_k} \|p(A)b\|^2,$$

where π_k is the set of polynomials of degree k with the value one at origin.

3.1. Deflated Two-Grid Preconditioner

Two-grid or multigrid preconditioning has been known for a long time, dating back at least to the 1930s. Its potential was first exploited by Fedorenko and Bakhalov in the 1960s, and later by Brandt [30] and Hackbusch [33], which paved the way to the birth of multigrid methods. We refer to [31, 32] and references therein for more details. It was shown that adding a coarse-grid correction or a second level correction can lead to a significant improvement in the convergence rate.

In multigrid there exist several ways of incorporating the coarse-grid correction [36]. We propose a new deflation based two-grid method, where the coarse grid correction is achieved by a deflation preconditioner. In our deflation based two-grid method, eigenvectors or approximations to eigenvectors associated with unfavorable eigenvalues are used as interpolation and restriction operators between the coarse to fine grid and vice-versa.

Let the coarse grid matrix be defined as follows

$$A_c = RAP, \quad A_c \in \mathbb{R}^{n_c\times n_c},$$

where n_c is the number of eigenvectors of A to be deflated, here $P \in \mathbb{R}^{n\times n_c}$ is the prolongation operator, and $R \in \mathbb{R}^{n_c\times n}$ is the restriction operator.

Our proposed two-grid preconditioner as combination of Jacobi and deflated preconditioner is given as follows

$$B_{\text{tg}}^{-1} = B_{\text{jac}}^{-1} + B_{\text{def}}^{-1} - B_{\text{def}}^{-1} A B_{\text{jac}}^{-1}, \tag{7}$$

where the smoother B_{jac} is defined as follows

$$B_{\text{jac}}^{-1} = \left[(\texttt{BlockDiag}(A)\right]^{-1}, \tag{8}$$

where the inverse of the blocks of Jacobi can be computed using well knownLU factorization [26], and the deflation preconditioner is defined as follows

$$B_{\text{def}}^{-1} = PA_c^{-1}R, \quad \text{where we define} \quad R = P^T. \tag{9}$$

From (7), we immediately observe that the iteration matrix corresponding to fixed point iteration $I - B_{\text{tg}}^{-1}A$ is given as follows

$$I - B_{\text{tg}}^{-1}A = (I - B_{\text{jac}}^{-1}A)(I - B_{\text{def}}^{-1}A), \tag{10}$$

which is a product of the iteration matrices corresponding to B_{jac} and B_{def}. Hence the preconditioner B_{tg} is a multiplicative combination of the smoother B_{jac} and the coarse grid correction B_{def}.

In the following lemma, we show that H_{LM} is Symmetric and Positive Definite (SPD), hence all its eigenvalues will be real and positive.

Lemma 1. *If the Jacobian J is full rank, and $\mu > 0$ in (3) according to LM method, then the coefficient matrix $A = H_{LM}$ is SPD.*

Proof. We have from (3),

$$A = J^TJ + \mu\,\texttt{diag}(J^TJ)$$

J being full rank implies that J^TJ is full rank, and SPD. Also $\text{diag}(J^TJ)$ being the diagonal of a SPD matrix J^TJ is also SPD, hence,

$$\begin{aligned} x^TAx &= x^T(J^TJ + \mu\texttt{diag}(J^TJ))x \\ &= x^T(J^TJ)x + x^T(\mu\,\texttt{diag}(J^TJ))x \\ &> 0 \end{aligned}$$

□

In the following Lemma, we show that if B_{def} is constructed from the $k-$largest eigenvectors, then those eigenvectors becomes the eigenvectors corresponding to eigenvalue one for the preconditioned operator $B_{def}^{-1}A$, i.e., B_{def} "deflates" the largest eigenvalue to one.

Lemma 2. *Let $A \in \mathbb{R}^{n\times n}$ be SPD, then it has n linearly independent orthogonal eigenvectors. Let those eigenvectors be $\{v_1, \ldots, v_n\}$ corresponding to eigenvalues $\lambda_1, \ldots, \lambda_n$, where*

$$\lambda_1 > \lambda_2 > \cdots > \lambda_n.$$

Let the deflation preconditioner B_{def} be defined as in (9)*. Let $P = [v_1, v_2, \ldots, v_k]$, where $v_i \in \mathbb{R}^n, i = 1, \ldots, k$ are eigenvectors corresponding to k-largest eigenvalues to be deflated. Then the following holds*

$$B_{def}^{-1}A\,v_i = v_i, \quad i = 1, \ldots, k.$$

and

$$B_{def}^{-1}A\,v_i = 0, \quad i = k+1, \ldots, n.$$

Proof. We have

$$\begin{aligned} (I - B_{def}^{-1}A)v_i &= v_i - B_{def}^{-1}Av_i \\ &= v_i - PA_c^{-1}P^TAv_i \\ &= v_i - PA_c^{-1}P^T\lambda_i v_i \\ &= v_i - P\begin{bmatrix} \lambda_1^{-1} & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2^{-1} & 0 & \cdots & 0 \\ \vdots & \ddots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & \cdots & \lambda_k^{-1} \end{bmatrix}\begin{bmatrix} 0 \\ \vdots \\ \lambda_i \\ \vdots \\ 0 \end{bmatrix} \\ &= v_i - v_i = 0 \end{aligned}$$

Similarly, from above

$$\begin{aligned} (I - B_{def}^{-1}A)v_i &= v_i - PA_c^{-1}P^T\lambda_i v_i \\ &= v_i - PA_c^{-1}\mathbf{0}, \quad \text{since } P^Tv_i = \mathbf{0},\ i > k, \end{aligned}$$

where $\mathbf{0} \in \mathbb{R}^n$ is a zero vector. □

Theorem 3. *If A is SPD, then B_{jac} defined in* (8) *exists.*

Proof. Since A is SPD, the principal blocks of A is also SPD, hence B_{jac} being the block diagonal of the principal diagonal blocks of A remains SPD. □

As a consequence of Lemma 2, we now show that the largest eigenvector components of the error will be damped or removed. This in turn helps in faster convergence of iterative scheme.

Theorem 4. *Let A be SPD. Let B_{jac} and B_{def} be defined as in* (8) *and* (9) *respectively. Also, assume that P in B_{def} is defined as $P = [v_1, \ldots, v_k]$, where $v_1, \ldots, v_k$ are the eigenvectors corresponding to k largest eigenvalues of A, then during the fixed point iteration, the deflation preconditioner B_{def} damps components $v_1, \ldots, v_k$ of the error.*

Proof. For fixed point iteration, i.e., Richardson iteration, we have the following error iteration

$$e^{t+1} = (I - B_{tg}^{-1}A)e^t, \text{ where } \quad e^t \in R^n.$$

Here, e^t is the error at iteration t. Since from Lemma 1, $A = H_{LM}$ is symmetric positive definite matrix, A has n linearly independent eigenvectors which forms a basis for $\mathbb{R}^n$. Thus, we can write e^t as a linear combination of eigenvectors $v_i, i = 1, \ldots, n$ as follows

$$e^t = \sum_{i=1}^{n} c_i v_i,$$

where c_i's are scalars. From (10), we have

$$I - B_{\text{tg}}^{-1}A = (I - B_{\text{jac}}^{-1}A)(I - B_{\text{def}}^{-1}A).$$

Multiplying both sides by e^t, we get

$$(I - B_{\text{tg}}^{-1}A)e^t = (I - B_{\text{jac}}^{-1}A)(I - B_{\text{def}}^{-1}A)e^t \tag{11}$$

We have

$$(I - B_{def}^{-1}A)e^t = (I - B_{def}^{-1}A)\sum_{i=1}^{n} c_i v_i.$$

From Lemma 2, we have

$$\begin{aligned} B_{def}^{-1}Av_i &= v_i, \quad i = 1, \ldots, k, \\ B_{def}^{-1}Av_i &= 0, \quad i = k+1, \ldots, n, \end{aligned}$$

Hence,

$$(I - B_{def}^{-1}A)e^t = \sum_{i=k+1}^{n} c_i v_i$$

From (11), we have

$$(I - B_{tg}^{-1}A)e^t = (I - B_{jac}^{-1}A)\sum_{i=k+1}^{n} c_i v_i \tag{12}$$

□

Remark 5. *In theorem above, in* (12)*, the remaining components of the error are efficiently damped by Jacobi [31, p30], which is a well known smoother.*

Contrary to typical two-grid method, where coarse grid correction damps the low frequency components of the error, the deflation preconditioner explicitly removes the error corresponding to very high frequency components (large eigenvectors) of the error as proved above. This is achieved by defining P such that it contains the eigenvectors corresponding to large eigenvalues. The Jacobi preconditioner further smoothens the remaining high frequency components of the error. We remark here that computing eigenvectors corresponding to largest eigenvalues is much cheaper compared to computing eigenvectors corresponding to smallest eigenvalues. Consequently, as we will show in the numerical experiments section, this combination of deflation with Jacobi smoothing proves to be very effective.

Apart from Jacobi many other classical solvers such as Gauss-Seidel, SSOR, and ILU are typically used as smoothers in Multigrid Methods, on the other hand, for coarse grid correction, aggregation based schemes have been proposed in [36, 37].

Following the construction of the two-grid preconditioner above, we present a sketch of the algorithm (corresponds to a V-cycle) to solve with the preconditioner.

Algorithm 1 Preconditioner Solve (a V-cycle) $B_{\text{tg}}x = b$

Require: $B_{\text{jac}}, A_c, P, A, b$
// Relax with the smoother
$s = B_{\text{jac}}^{-1} b$
// Compute residual
$r = b - As$
// Restrict residual
$r_c = P^T r$
// Coarse grid correction
Solve the coarse grid system $A_c e_c = r_c$
// Prolongate the coarse grid correction
$x = s + Pe_c$
Ensure: x

4. Experimental Evaluation

4.1. Configurations

We use a freely available sparse Levenberg-Marquardt C++ implementation (SSBA) [1], which has several cost functions that are used by the LM algorithm for the bundle adjustment step. We use the following cost functions for our experiments: (1) `bundle_large` and (2) `bundle_large_lifted_schur` implemented in `SSBA` package. The details of these cost functions have been discussed in detail in [38].

The stopping criteria for the LM algorithm are either of the following

- the number of iterations of LM exceeds 100.
- the difference in two consecutive residuals is less than 10^{-12} in magnitude.

[1] We selected the simple sparse bundle adjustment (`SSBA`) package, `http://www.cvg.ethz.ch/research/chzach/opensource.html`

For solving the augmented normal equations (6) in each LM iteration, `SSBA` uses LDL factorization [39, p. 157], a Cholesky like factorization method for sparse symmetric positive definite matrices. The appropriate column reordering is done with `COLAMD`. Both LDL and COLAMD have been adopted from the `SuiteSparse` package [40].

We have implemented a two-grid preconditioned GMRES as an iterative solver in the `SSBA` package for the augmented normal equations arising in each LM iteration to solve (6). The stopping criteria for the GMRES method are (1) the number of iterations exceeds 500 with number of restarts equal to 20 (2) the norm of the relative residual is less than 10^{-3}. The GMRES method is implemented using the `dfgmres` routine available in the `INTEL MKL` library version `2019.4.243`.

All the experiments are performed on the `BAL` dataset released by Agarwal et al.[1]. Since `BAL` contains many datasets, and some of them cannot fit into the memory of a regular PC due to RAM constraints, we select ten appropriate datasets from `BAL` which have 7K to 700K points (see Table. 3). We run all of the experiments on a machine with Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz and 16GB RAM.

4.2. Implementation Details

As mentioned before, the implementation of the two-grid deflated preconditioner has two main ingredients (1) Construction of smoother, and (2) Coarse grid correction. For constructing the smoother B_{jac}, we explicitly combine the D and G blocks of (6) to form a block diagonal matrix as follows

$$B_{jac} = LU\left(\begin{bmatrix} D & \\ & G \end{bmatrix}\right).$$

On the other hand, coarse grid correction requires the setup of prolongation and restriction operators and the construction of a coarse grid matrix using these grid transfer operators. The columns of the prolongation operator P are the eigenvectors corresponding to the 5 largest eigenvalues of H_{LM}. These eigenvectors of A are computed using the double precision routine `mkl_sparse_d_ev`, which requires the input matrix to be in CSR format with one based indexing.

Usually the construction of the coarse grid matrix $A_c = RH_{\text{LM}}P$ requires two *matrix-matrix* products. In our case, since P contains eigenvectors of H_{LM}, the coarse grid matrix $A_c = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_k)$ is a $k \times k$ diagonal matrix. Here we fix $k = 5$ in all our experiments.

The application of B_{def}^{-1} in (7) for the preconditioner solve is implemented as

$$B_{\text{def}}^{-1}x = (P(A_c^{-1}(Rx))), \quad \text{where} \quad R = P^T,$$

for some vector $x \in \mathbb{R}^n$, where n is the number of rows in H_{LM}, generated by the GMRES algorithm. Here, Rx requires one *matrix-vector* product. Since the diagonal matrix A_c^{-1} is stored as a vector, $A_c^{-1}(Rx)$ is just a simple scaling of the elements of Rx. This resultant vector is then multiplied with P. Thus the construction of B_{def}^{-1} requires just two *matrix-vector* products.

4.3. Results

Here we compare the direct solve, specifically, the sparse LDL factorization method and the generalized minimal residual method

(GMRES) with two preconditioners: (1) Block Jacobi preconditioner and (2) Multiplicative deflated two-grid preconditioner. All the preconditioners are implemented as part of the same SSBA package.

We also experimented using block SSOR preconditioner and standard coarse grid correction, where interpolations are constructed using strength of connection based aggregation method [37], but as shown in Table 2, they do not perform well compared to the results shown in Table 3. We note here that classical multigrid method such as AGMG were designed for scientific computing applications, where the coefficient matrices are usually either diagonally dominant or has so-called M-matrix property. For the problems considered in this paper, the matrix H_{LM} is found to be neither diagonally dominant nor an M-matrix. This explains why traditional multigrid methods fail.

In Table 1, we show the memory required by these methods. We observe that our method requires same order of memory as Jacobi method. On the other hand, as the problem size increases, the two-grid method requires only 54% of the memory required by the direct method.

In Table 3, we compare various methods. In the table, the best times are shown in bold. We observe that for smaller problems, the direct method is fastest. However, for larger problems, the two-grid method converges roughly two to three times faster than both the direct solve and block Jacobi methods.

One of the main reasons for fast convergence of our proposed two grid deflation technique is that we explicitly deflate the large eigenvalues instead of small eigenvalues. As shown in Table 4, we observe that deflating the smaller eigenvalues still leads to a very slow convergence of the linear solves in LM iterations. The largest eigenvalues of the H_{LM} as shown in Figure 2, contribute to its high condition number and therefore deflating the largest eigenvalues in our two-grid approach leads to faster convergence of the GMRES. We point out that some of the largest eigenvalues are well separated, and deflating them led to faster convergence. Thus our deflation strategy is different from the traditional deflation, where smallest eigenvalues are deflated. For these problems, we recommend deflating largest eigenvalues for faster convergence as is evident from our test results.

Moreover, in Figure 4, we show the total time of LM iteration for a cost function namely bundle_large_lifted_schur, implemented in the SSBA package, and for this cost function as well we observe that the two-grid method converges the fastest compared to other two methods for a comparable mean reprojection error. So our proposed preconditioning technique is robust and easily scalable to large scale bundle adjustment problems.

Table 1. Memory usage in bytes of the three methods on ten BAL datasets.

BAL dataset parameters			memory usage		
Images	Points	Obs.	Direct	Jacobi	Two-Grid
49	7776	31843	2.4e+07	2.1e+07	2.2e+07
73	11032	46122	3.8e+07	3.1e+07	3.2e+07
138	19878	85217	8.2e+07	6.0e+07	6.2e+07
372	47423	204472	3.0e+08	1.7e+07	2.2e+07
412	52215	224242	3.9e+08	1.9e+08	2.0e+08
460	56811	241877	4.2e+08	2.0e+08	2.1e+08
598	69218	304170	4.7e+08	2.6e+08	2.7e+08
707	78455	349940	5.7e+08	3.0e+08	3.1e+08
783	84444	377052	7.9e+08	3.3e+08	3.4e+08
969	105826	474627	7.4e+08	3.9e+08	4.0e+08

Table 2. Results for Aggregation based algebraic Multigrid

Dataset	Iterations	Time(s)	Tolerance
L372	1000	160.4	10^{-2}
L598	1000	247.093	10^{-2}
L1031	1000	379.718	10^{-2}
V427	1000	539.41	10^{-2}
D356	1000	427.03	10^{-2}

5. Conclusions and Future Work

We proposed a new two-grid approach. The proposed method is found to be fairly robust and fast compared to direct method and block Jacobi method proposed in literature. Moreover, the proposed method requires less memory compared to direct method as the problem size increases.

One direction of future work is to explore more sophisticated deflation techniques, and to do extensive comparison with other methods for solving bundle adjustment problems.

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.

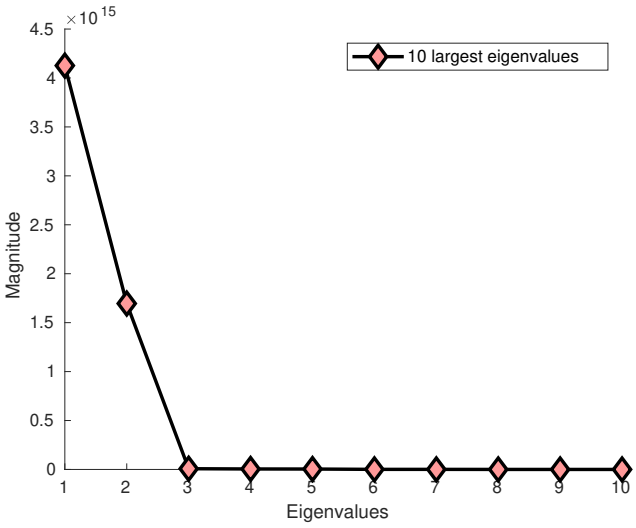

Figure 2. Largest 10 eigenvalues of LadyBug-372 Dataset

Table 3. Average time (in seconds) per iteration in the LM solver of the three methods on ten BAL datasets using bundle_large_lifted_schur cost function routine in SSBA. The first column corresponds to the name and index in the original bal: "L" - "LadyBug", "D" - "Dubrovnik", "V" - "Venice" and "F" - "Final". Here H_{LM} Size is the number of rows of H_{LM}.

BAL Dataset Parameters						Direct Solve		Block Jacobi		Two-Grid	
Set	Dataset	H_{LM} Size	Cameras	Points	Observations	Time	Error	Time	Error	Time	Error
1	L-49	23769	49	7776	31843	**0.10**	0.733	0.55	0.707	0.15	0.665
2	L-73	33753	73	11032	46122	**0.26**	0.686	2.44	0.642	0.32	0.675
3	L-138	60876	138	19878	85217	**0.45**	0.893	0.85	0.826	0.56	0.876
4	L-598	213036	598	69218	304170	8.23	0.761	20.9	0.784	**2.20**	0.769
5	L-969	326199	969	105826	474627	12.9	0.716	29.5	0.737	**4.73**	0.751
6	L-1723	485013	1723	156502	678718	50.4	0.758	72.6	0.758	**13.1**	0.758
7	D-356	683394	356	226730	1255268	11.3	0.801	41.5	0.794	**8.20**	0.796
8	V-427	934995	427	310384	1699145	16.6	0.754	37.0	0.762	**11.9**	0.752
9	F-871	1590279	871	527480	2785977	49.1	0.995	68.5	1.127	**38.2**	1.023
10	F-1936	1966443	1936	649673	5213733	129.4	0.956	114.7	0.947	**60.9**	0.958

Table 4. Time per iteration (secs) of LM with deflation of small eigenvalues.

Dataset	Condition No.	Eigen 1	Eigen 3	Eigen 5
L49	1.83e20	67.07	66.048	65.96
L138	3.47e19	245.75	247.82	246.03
V52	4.44e15	529.80	537.71	538.54
V89	2.46e16	800.32	809.41	813.02

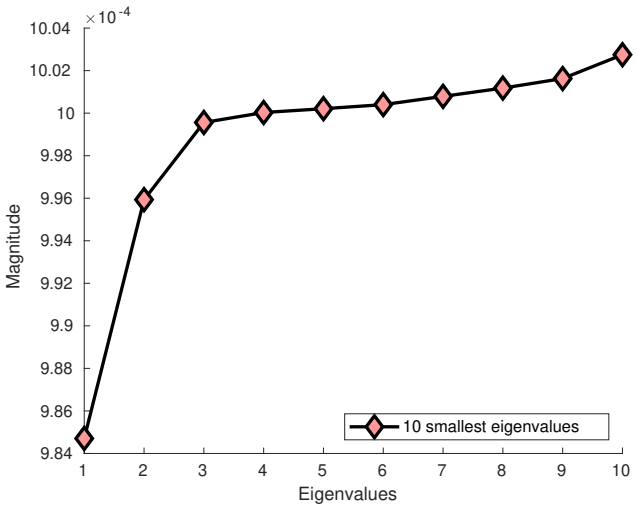

Figure 3. Smallest 10 eigenvalues of LadyBug-372 Dataset

References

[1] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In ECCV, pages 29–42, 2010.

[2] S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building rome in a day. In IEEE 12th International Conference on Computer Vision, pages 72–79, 2009.

[3] B. Triggs, P. McLauchlan, R. Hartley, A. Fitzgibbon. Bundle Adjustment - A modern synthesis. International Workshop on Vision Algorithms, Sep 2000, p.298-372

[4] M. Lourakis, A.A. Argyros. SBA: A software package for generic sparse bundle adjustment. TOMS 36 (2009).

[5] R. Hartley. In defense of the eight-point algorithm. IEEE Trans. Pattern Anal. Mach. Intell., vol. 19, no. 6, pp. 580–593, 1997.

[6] N. Snavely, S.M. Seitz, R. Szeliski. Photo Tourism: Exploring photo collections in 3D. TOG (2006) 835–846.

[7] M. Bryöd and K. Aström. Conjugate gradient bundle adjustment. In European Conf. on Computer Vision, 2010.

[8] J.-M. Frahm, P. F. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, and S. Lazebnik. Building Rome on a cloudless day. In European Conference on Computer Vision, 2010.

[9] A. Conn, N. Gould, and P. Toint. Trust Region Methods. MPS-SIAM Series On Optimization. SIAM, Philadelphia, PA, 2000.

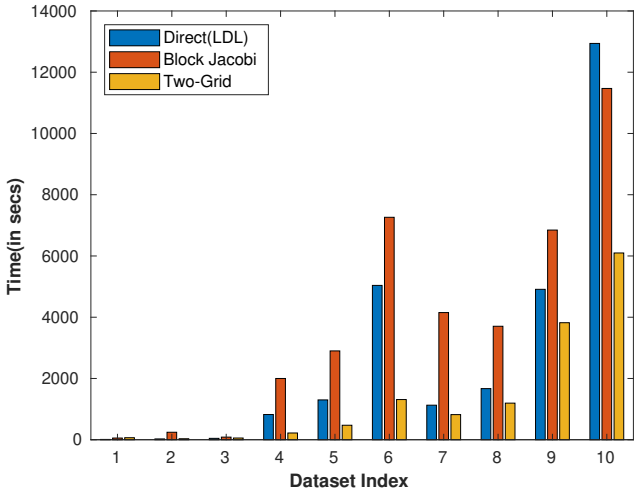

Figure 4. Time for convergence of full bundle adjustment problems for bundle_large_lifted_schur cost function of SSBA.

[10] K. Levenberg. A Method for the Solution of Certain Non-Linear Problems in Least Squares. The Quarterly of Applied Mathematics, 2: 164–168 (1944).

[11] D. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics, 11, 431–441 (1963).

[12] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. International Journal of Computer Vision, 80(2):189–210, 2008.

[13] B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In The IEEE International Conference on Computer Vision (ICCV), December 2013.

[14] Y. Jeong, D. Nister, D. Steedly, R. Szeliski, and I. Kweon. Pushing the envelope of modern methods for bundle adjustment. In IEEE Conference on Computer Vision and Pattern Recognition, pages 1474–1481, 2010.

[15] J. Heinly, J. L. Schonberger, E. Dunn, and J.-M. Frahm. Reconstructing the world* in six days *(as captured by the yahoo 100 million image dataset). In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[16] M. Bryöd, K. Aström, and S. Lund. Bundle adjustment using conjugate gradients with multiscale preconditioning. In BMVC, 2009.

[17] 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.

[18] Avanish Kushal, Sameer Agarwal. Visibility Based Preconditioning for bundle adjustment. In CVPR, 2012.

[19] Y.-D. Jian, D. C. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In ICCV, 2011.

[20] D. C. Brown. The bundle adjustment progress and prospects. *Int. Archives Photogrammetry*, 21(3), 1976.

[21] M.I. Lourakis, A.A. Argyros. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment? In ICCV, vol. 2, pp. 1526–1531. IEEE (2005).

[22] J. Wright and J. N. Holt. An inexact Levenberg-Marquardt method for large sparse nonlinear least squares. J. Austral. Math. Soc. Ser. B, 26(4):387–403, 1985.

[23] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[24] C. Wu, S. Agarwal, B. Curless, and S. Seitz. Multicore bundle adjustment. In CVPR, pages 3057–3064, 2011.

[25] Y. Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., July 1986, Vol. 7, No. 3, pp. 856–69.

[26] Y. Saad. Iterative methods for sparse linear systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, second edition, 2003.

[27] A. Greenbaum and Z. Strakoš. Matrices that generate the same Krylov residual spaces. In Recent advances in iterative methods, volume 60 of IMA Vol. Math. Appl., pages 95–118. Springer, New York, 1994.

[28] A. Greenbaum, V. Pták, and Z. Strakoš. Any nonincreasing convergence curve is possible for GMRES. SIAM J. Matrix Anal. Appl., 17(3):465–469, 1996.

[29] M. Arioli, V. Pták, and Z. Strakoš. Krylov sequences of maximal length and convergence of GMRES. BIT, 38(4):636–643, 1998.

[30] Brandt, A. Multi-level adaptive solutions to boundary-value problems. Math. Comput. 31, 333–390 (1977).

[31] U. Trottenberg, C.W. Oosterlee, A. Schller. Multigrid, Academic Press, London (2001).

[32] J. Xu. Iterative methods by space decomposition and subspace correction. SIAM Rev. 34, 581–613 (1992).

[33] W. Hackbusch. Multigrid Methods and Applications. Springer, Berlin (1985).

[34] Y. Notay and A. Napov. Further comparison of additive and multiplicative coarse grid correction, Applied Numerical Mathematics, 65, (53), (2013).

[35] Y. Notay, A. Napov, When does two-grid optimality carry over to the V-cycle, NLAA, vol 17, issue 2-3, (2010).

[36] Y. Notay, An aggregation based algebraic multigrid method, Electronic transaction on Numerical analysis, 37 (2010), pages 123–146.

[37] K. Pawan, Aggregation based on graph matching and inexact coarse grid solve for algebraic two grid, International Journal of Computer Mathematics, 2014.

[38] C. Zach, Robust bundle adjustment revisited, ECCV, 2014.

[39] G. Golub, C. F. Van Loan, Matrix Computations, John Hopkins Press, 2013.

[40] T. Davis, SuiteSparse, `http://faculty.cse.tamu.edu/davis/suitesparse.html`

[41] L. Giraud, S. Gratton, X. Pinel, and X. Vasseur, Flexible GMRES with Deflated Restarting, SIAM Journal on Scientific Computing, 2010, Vol. 32, No. 4, pp. 1858-1878.

[42] R.B. Morgan, GMRES with Deflated Restarting, SIAM J. Sci. Comput., 24(1), 2037.

[43] S. Das, S. Katyan, P. Kumar, Domain Decomposition Based Preconditioned Solver for Bundle Adjustment, NCVPRIPG, 2019.

[44] A. Toselli, O. Widlund, Domain Decomposition Methods - Algorithms and Theory, doi: 10.1007/b137868, Springer-Verlag Berlin Heidelberg, 2005.