Computers & Geosciences 92 (2016) 70–78

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Research paper

Bundle block adjustment of large-scale remote sensing data with Block-based Sparse Matrix Compression combined with Preconditioned Conjugate Gradient

Maoteng Zheng [a,*], Yongjun Zhang [b], Shunping Zhou [a], Junfeng Zhu [c], Xiaodong Xiong [c]

[a] *National Engineering Research Center for Geographic Information System, China University of Geosciences, Wuhan, China*
[b] *School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China*
[c] *Chinese Academy of Surveying and Mapping, Beijing, China*

ARTICLE INFO

Article history:
Received 26 November 2015
Received in revised form
9 April 2016
Accepted 15 April 2016
Available online 16 April 2016

Keywords:
Bundle block adjustment
Large-scale remote sensing data
Block-based Sparse Matrix Compression
Preconditioned Conjugate Gradient

ABSTRACT

In recent years, new platforms and sensors in photogrammetry, remote sensing and computer vision areas have become available, such as Unmanned Aircraft Vehicles (UAV), oblique camera systems, common digital cameras and even mobile phone cameras. Images collected by all these kinds of sensors could be used as remote sensing data sources. These sensors can obtain large-scale remote sensing data which consist of a great number of images. Bundle block adjustment of large-scale data with conventional algorithm is very time and space (memory) consuming due to the super large normal matrix arising from large-scale data. In this paper, an efficient Block-based Sparse Matrix Compression (BSMC) method combined with the Preconditioned Conjugate Gradient (PCG) algorithm is chosen to develop a stable and efficient bundle block adjustment system in order to deal with the large-scale remote sensing data. The main contribution of this work is the BSMC-based PCG algorithm which is more efficient in time and memory than the traditional algorithm without compromising the accuracy. Totally 8 datasets of real data are used to test our proposed method. Preliminary results have shown that the BSMC method can efficiently decrease the time and memory requirement of large-scale data.

1. Introduction

Bundle block adjustment is an inevitable and crucial procedure in the photogrammetry, remote sensing and computer vision area. Especially in the 3D model reconstruction with multiple sensors such as UAV camera, oblique camera system and even cell phone cameras. A lot of research works have been done for UAV camera systems (Mathews and Jensen, 2013; Ai et al., 2015; Tong et al., 2015; Frueh et al., 2004; Dahlke and Wieden, 2013). Oblique images (Besnerais et al., 2008), ordinary digital cameras (Dandois and Ellis, 2010), and even internet images (Snavely et al., 2008; Agarwal et al., 2010, 2011). Those images are always large-scale data. They bring us more redundant observations, but in the mean time, they need more computation and memory resources. The main challenge of bundle block adjustment of these large-scale data is storing and computing the super large normal equation produced by a large number of images and tie points. Thus how to solve the big normal equation is a key issue. There are usually two categories of solutions, direct method and iterative method. Direct method is always referred to the conventional Levenberg–Marquardt (LM) algorithm. It was a most popular algorithm in the last few decades for solving non-linear least square problems. Conventional aerial photogrammetry procedure acquired images on an airborne platform and the images are regularly arranged. Thus the normal matrix has a sparse band structure which can be easily solved with memory space equal to the bandwidth of the sparse band structure. But the UAV images, oblique images, cell phone image and especially the internet images are mostly arranged irregularly. So the normal matrix of these data has no band characteristics. The LM method has to invert the full sparse matrix. It is no longer suitable for processing the large-scale remote sensing data. The iterative method includes a lot of algorithms. The most widely used iterative method is Conjugate Gradient (CG) algorithm which was firstly proposed in 1952 (Hestenes and Stiefel, 1952), but it's not widely used due to its drawbacks in precision and stability until recently. This method multiplies the normal matrix and the residuals vector of normal equation, forming a Krylov subspace (Saad, 1981), and then iteratively computes the answer of normal equation, and eventually end up to a convergence value which is close enough to the true answer (Hestenes and Stiefel, 1952). CG was further extended to an advanced method called Preconditioned Conjugate Gradient (PCG) which uses a preconditioner to reduce the condition

* Corresponding author.
E-mail address: tengve@163.com (M. Zheng).

http://dx.doi.org/10.1016/j.cageo.2016.04.006
0098-3004/© 2016 Elsevier Ltd. All rights reserved.

of the normal matrix so as to improve the converging speed (Bru et al., 2008; Byröd and Åström, 2009, 2010; Jian et al., 2011; Li and Saad, 2013).

The main advantage of the iterative method is that the normal equation can be solved without explicitly forming the full normal matrix which is a relatively high cost in both computation and storage phase. It gives us a chance to solve the super large normal equation produced by large-scale data consisting of more than hundreds of thousands even millions of images on a common personal computer. The normal matrix is always sparse and includes a lot of zero elements. So only non-zero elements and their position indexes in the sparse matrix need to be stored, this strategy can largely compress the normal matrix. A widely used matrix compress method called Compressed Sparse Row (CSR) uses three one dimension data arrays to store non-zero elements and the corresponding necessary information. But we found in practice that this storage method is not suitable for the normal matrix update since that the normal matrix are formed and updated point by point in bundle block adjustment. After the sub-normal-matrix of each point is calculated, the whole normal matrix needs to be updated. When a CSR storage method is adopted, finding the position of the sub-normal-matrix of current point in the full normal matrix is very time consuming and complicated. Because the CSR stores matrix elements one by one while the sub-normal-matrix is an $n*n$ block (n is the number of unknown parameters related to the current point), all the elements of the sub-normal-matrix have to be updated one by one. So we propose a Block-based Sparse Matrix Compression (BSMC) format to compress the whole normal matrix in order to decrease its memory requirement while making the normal matrix easy to be updated.

In this paper, PCG algorithm is applied to solve the large normal equation. The Jacobi preconditioner is chosen to decrease the iteration times of PCG process. The BSMC method is introduced to combine with PCG aiming to decrease the memory requirement. The main purpose of this work is to build a stable and efficient bundle block adjustment system to deal with large-scale remote sensing data. Part of the test data are downloaded from a public data source website which was built and shared by Sameer Agarwal in University of Washington (Agarwal et al., 2010). UAV images, oblique images and cell phone images are also used as test data. We have analyzed and compared the memory and time requirement of different methods including the conventional LM algorithm and proposed BSMC method with PCG. A final summary of this work is given in the last section.

2. Related works

LM algorithm has been well studied for a long time. The mathematic theory and equation derivation are well-defined. Recently, the structure from motion is widely discussed in computer vision community. Most of the researchers working on structure from motion applied iterative methods to deal with bundle block adjustment problems of large-scale data (Snavely et al., 2008; Jian et al., 2011). The most famous and widely used method is PCG which is an extension version of the CG algorithm. CG belongs to the algorithms family called Krylov method (Saad, 1981). It multiplies the normal matrix and the residuals vector, forming a Krylov subspace which is used to iteratively solve the normal equations. It's been reported that the iteration times of convergence is related to the condition number of the normal matrix. Thus a PCG algorithm uses a preconditioner to decrease the condition of the normal matrix, so as to improve the converging speed (Byröd and Åström, 2010; Jian et al., 2011). A lot of works have focused on how to choose a proper preconditioner. Some efficient and stable preconditioners have been introduced, such as Jacobi preconditioner (Agarwal et al., 2010, 2011) (Byröd and Åström, 2010; Jian et al., 2011), Symmetric Successive Over-relaxation (SSOR) preconditioner (Byröd and Åström, 2010; Jian et al., 2011), QR factorization preconditioner (Byröd and Åström, 2010), Balanced Incomplete Factorization based preconditioner (Bru et al., 2008), multiscale preconditioner (Byröd and Åström, 2009), subgraph preconditioner (Jian et al., 2011) and so on. Among the above preconditioners, Jacobi is simplest and most widely used in the real case. Iterative methods can be also explored in remote sensing community since more and more large-scale remote sensing data have emerged, such as UAV images, Oblique images, and mobile phone images even internet images. When the images increased to a certain large number, the conventional LM method is no longer suitable for solving such big normal equations. Then, the iterative method is considered to be a necessary alternative.

Matrix-vector product is the most frequently computed procedure in the PCG iteration. The multiplications of normal matrix and residuals vector need to be calculated during each iteration of PCG. It means that the normal matrix will be frequently read and used during each iteration of PCG. Thus it has to be stored whatever in RAM or external memory. Some methods use mathematic trick to avoid storing the normal matrix (Agarwal et al., 2010; Byröd and Åström, 2010). However, this will take even more computational cost which will largely slow down the iteration speed. But to store the whole normal equation will need a very large memory space especially for the large-scale data. So the normal matrix should be compressed. As mentioned before, the normal matrix is often sparse and includes a lot of zero elements. Only non-zero elements and their position indexes in the sparse matrix need be stored. A famous and widely used normal matrix compression method is called CSR. This method only needs three one dimensional data arrays to store the non-zero elements and their position indexes while abandoning all the zero elements (Bell and Garland, 2009). But we found in practice that this storage method is not suitable for the normal matrix update. So the BSMC method is introduced to decrease its memory requirement while making the normal matrix easy to be updated.

3. Methodology

3.1. Imaging geometry

A ground point $P(X, Y, Z)$ is imaged by a camera with parameters (Xs, Ys, Zs, phi, omega, kappa) known as Exterior Orientation Parameters (EOPs) and (f, $x0$, $y0$, $k1$, $k2$) known as Interior Orientation Parameters (IOPs). Then an image point $p(x, y)$ corresponding to the ground point P can be obtained in the image as shown in Fig. 1. The camera lens center is defined as the perspective center S. The ground point P, it's corresponding image point p and the perspective center S is on a line, the relationship can be described by formulae as Eqs. (1), (2), (3) and (4).

$$\begin{bmatrix} x-\Delta x \\ y-\Delta y \\ -f \end{bmatrix} = \mathbf{R}^{\mathrm{T}} \begin{bmatrix} X-Xs \\ Y-Ys \\ Z-Zs \end{bmatrix} \tag{1}$$

$$\mathbf{R} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \tag{2}$$

where **R** is a rotation matrix consisting of three rotation angles: phi, omega, and kappa. Δx, Δy is the correction terms for image point coordinates.

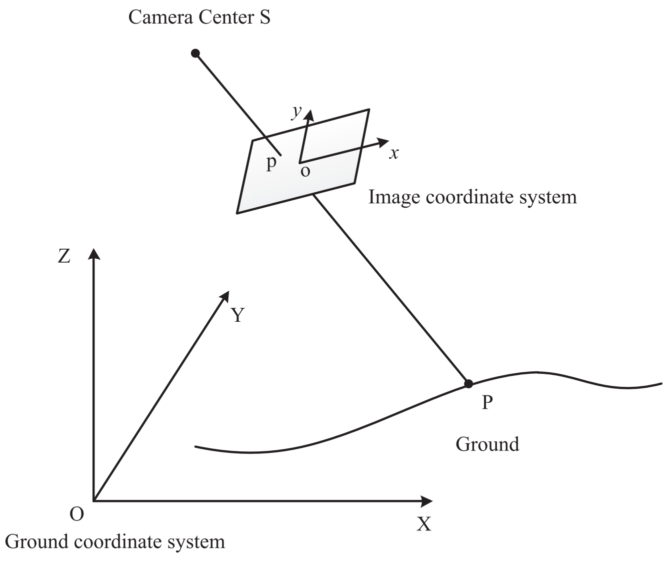

Fig. 1. The Geometry of image to the ground. O-*XYZ* is the ground coordinate system, o-*xy* is the image coordinate system.

$$\begin{cases} \Delta x = x_0 + k_1(x - x_0)r^2 + k_2(x - x_0)r^4 \\ \Delta y = y_0 + k_1(y - y_0)r^2 + k_2(y - y_0)r^4 \end{cases} \tag{3}$$

$$r = \sqrt{(x - x_0)^2 + (y - y_0)^2} \tag{4}$$

Combine (1) and (2) we have

$$\begin{bmatrix} x - \Delta x \\ y - \Delta y \\ -f \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} X - Xs \\ Y - Ys \\ Z - Zs \end{bmatrix} \tag{5}$$

In a common bundle block adjustment problem, the EOPs (*Xs*, *Ys*, *Zs*, phi, omega, kappa) and ground point *P*(*X*, *Y*, *Z*) are unknowns. If we use an uncalibrated camera, then the IOPs (*f*, *x*0, *y*0, *k*1, *k*2) are also unknowns. The given parameters are the observations: image point coordinates (*x*, *y*) extracted from images.

If the EOPs and IOPs are all given, we can calculate the ground point coordinates by intersection with conjugate image points, the intersection geometry is demonstrated in Fig. 2. If there are more than two conjugate image points, the intersection should be more robust.

3.2. Levenberg–Marquardt model

The purpose of the bundle block adjustment is to obtain the EOPs of every camera and every unknown Ground Point Coordinates (GPC) using a large number of conjugate image points and the collinearity condition. The IOPs are often calibrated before bundle block adjustment. IOPs are set as given values in this paper, thus only EOPs and GPCs are the unknown parameters. We can build error equations according to Eq. (5) based on Gauss–Newton model (Byröd and Åström, 2010).

$$\mathbf{A}^T\mathbf{A}\mathbf{u} = \mathbf{A}^T\mathbf{b} \tag{6}$$

where **A** is a matrix consisting of the first order derivatives of Eq. (5) to the unknowns (EOPs and GPC), it's also called Jacobi matrix. **u** is the vector of unknowns. **b** is a vector of the differences between calculated and observed image point coordinates. A damping term $\lambda\mathbf{D}$ is often used in case that the rank of $\mathbf{A}^T\mathbf{A}$ is not full and makes Eq. (6) irresolvable.

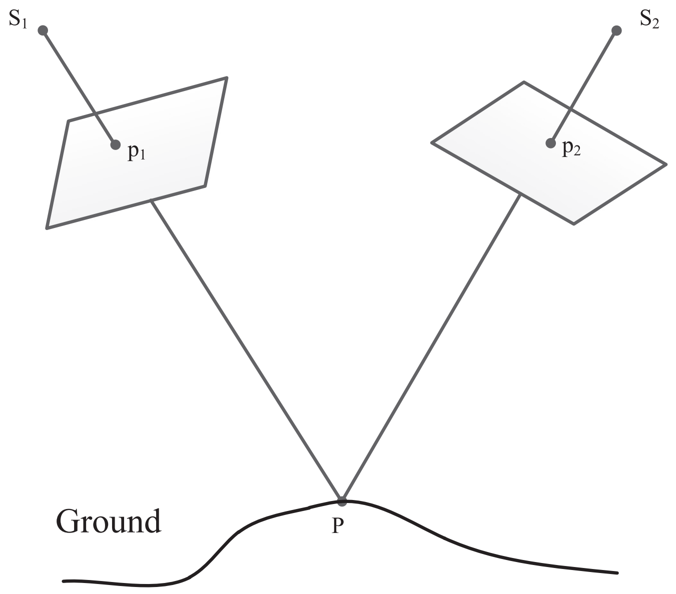

Fig. 2. S_1, S_2 are left and right camera centers, P is a ground point which is imaged by left and right camera at S_1 and S_2 producing two corresponding image points p_1, p_2.

$$(\mathbf{A}^T\mathbf{A} + \lambda\mathbf{D})\mathbf{u} = \mathbf{A}^T\mathbf{b} \tag{7}$$

Matrix **D** is usually the diagonal of matrix $\mathbf{A}^T\mathbf{A}$, λ is a damping value between (0, 1). It should be changed according to the results of each iteration.

3.3. Schur complement

The Jacobi matrix **A** can be partitioned into two parts, camera part and ground point part, so the matrix **A** can be rewritten as $\mathbf{A} = [\mathbf{A}_C \ \ \mathbf{A}_P]$ where $\mathbf{A}_C$ consists of the derivatives of Eq. (5) to EOPs and $\mathbf{A}_P$ consists of the derivatives of Eq. (5) to GPCs, the same can be done to $\mathbf{D} = [\mathbf{D}_C \ \ \mathbf{D}_P]$ and $\mathbf{u} = [\mathbf{u}_c \ \ \mathbf{u}_p]$. Then the Eq. (7) can be rewritten as follows:

$$\begin{bmatrix} \mathbf{A}_C^T\mathbf{A}_C + \lambda\mathbf{D}_C & \mathbf{A}_C^T\mathbf{A}_P \\ \mathbf{A}_P^T\mathbf{A}_C & \mathbf{A}_P^T\mathbf{A}_P + \lambda\mathbf{D}_P \end{bmatrix} \begin{bmatrix} \mathbf{u}_c \\ \mathbf{u}_p \end{bmatrix} = \begin{bmatrix} \mathbf{A}_C^T\mathbf{b} \\ \mathbf{A}_P^T\mathbf{b} \end{bmatrix} \tag{8}$$

For simplicity, denote $\mathbf{V}_C$ as $\mathbf{A}_C^T\mathbf{A}_C + \lambda\mathbf{D}_C$, $\mathbf{V}_P$ as $\mathbf{A}_P^T\mathbf{A}_P + \lambda\mathbf{D}_P$, **W** as $\mathbf{A}_C^T\mathbf{A}_P$, $\mathbf{b}_c$ as $\mathbf{A}_C^T\mathbf{b}$, $\mathbf{b}_p$ as $\mathbf{A}_P^T\mathbf{b}$, we get:

$$\begin{bmatrix} \mathbf{V}_C & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}_P \end{bmatrix} \begin{bmatrix} \mathbf{u}_c \\ \mathbf{u}_p \end{bmatrix} = \begin{bmatrix} \mathbf{b}_c \\ \mathbf{b}_p \end{bmatrix} \tag{9}$$

As can be seen in Eqs. (8) and (9), $\mathbf{V}_C$ and $\mathbf{V}_P$ are both block diagonal, the number of ground point unknowns are always much more than that of the camera unknowns. The structure of normal matrix is shown in Fig. 3.

So we can eliminate the ground point unknowns using block-wise Gauss elimination method and obtain the reduced normal equation.

$$(\mathbf{V}_C - \mathbf{W}\mathbf{V}_P^{-1}\mathbf{W}^T)\mathbf{u}_c = (\mathbf{b}_c - \mathbf{W}\mathbf{V}_P^{-1}\mathbf{b}_p) \tag{10}$$

$$\mathbf{V}_P\mathbf{u}_p = \mathbf{b}_p - \mathbf{W}^T\mathbf{u}_c \tag{11}$$

Unknown parameters $\mathbf{u}_c$ can be calculated by Eq. (10), and $\mathbf{u}_p$ can be then substituted by Eq. (11). As mentioned before, $\mathbf{V}_P$ is a block diagonal matrix which can be inverted block by block. If there are m cameras and n points, the size of normal matrix is now reduced from $(6m+3n)$*$(6m+3n)$ to $6m$*$6m$. This process is the so called Schur complement trick. Matrix $\mathbf{V}_C - \mathbf{W}\mathbf{V}_P^{-1}\mathbf{W}^T$ is known as

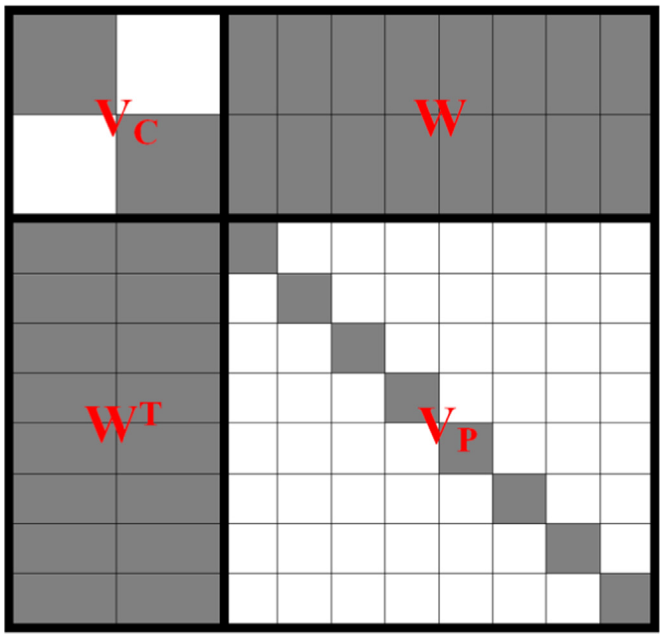

Fig. 3. The structure of a normal matrix in a simple bundle block adjustment problem where only EOPs and GPCs are unknowns, the gray squares or rectangles are blocks with non-zero elements. Others are zero elements. Each block in $\mathbf{V}_C$ consists of 6*6 elements, each block in $\mathbf{V}_P$ consists of 3*3 elements, each block in **W** consists of 6*3 elements.

Schur complement matrix **S**. Note that $n \gg m$ is the usual case, thus the Schur complement can efficiently reduce the size of the normal matrix. The condition of the reduced normal matrix is also largely decreased. After Schur complement, the normal equation can be rewritten as follows:

$$\mathbf{S}\mathbf{u}_c = \mathbf{l} \tag{12}$$

where

$$\mathbf{S} = \mathbf{V}_C - \mathbf{W}\mathbf{V}_P^{-1}\mathbf{W}^T \tag{13}$$

$$\mathbf{l} = \mathbf{b}_c - \mathbf{W}\mathbf{V}_P^{-1}\mathbf{b}_p \tag{14}$$

Schur complement matrix consists of only EOP unknowns corresponded to images. The structure of matrix **S** can express the relationship among those images. As demonstrated in Fig. 4, for a dataset with 12 images, the Schur matrix consists of 12*12 blocks each of which containing 6*6 elements. If block (i, j) is a non-zero block, it means that image i and image j have overlap area, on the contrary, if block (i, j) is a zero block, it means that image i and image j have no overlap. One can also see that the Schur matrix is a sparse matrix, as the image number is increasing, the sparsity of the matrix is getting bigger.

3.4. Preconditioned Conjugate Gradients

The PCG method is to apply a preconditioner $\mathbf{M}^{-1}$ to the normal matrix while using the conjugate gradient algorithm, so as to decrease the condition of the normal matrix, and thus accelerate the iteration process. After using a preconditioner, Eq. (12) can be rewritten as follows:

$$\mathbf{M}^{-1}\mathbf{S}\mathbf{u}_c = \mathbf{M}^{-1}\mathbf{l} \tag{15}$$

The iteration times now should be no more than the condition of matrix $\mathbf{M}^{-1}\mathbf{S}$. The main task is shifted to finding a proper precondition matrix which can decrease the condition of the normal matrix and is easy to be inverted. The simplest and most widely used preconditioner is Jacobi preconditioner which uses a diagonal of the normal matrix as preconditioner. Other preconditioners such as SSOR, QR factorization preconditioner and so on, could be more efficient but might be more complicated and less stable.

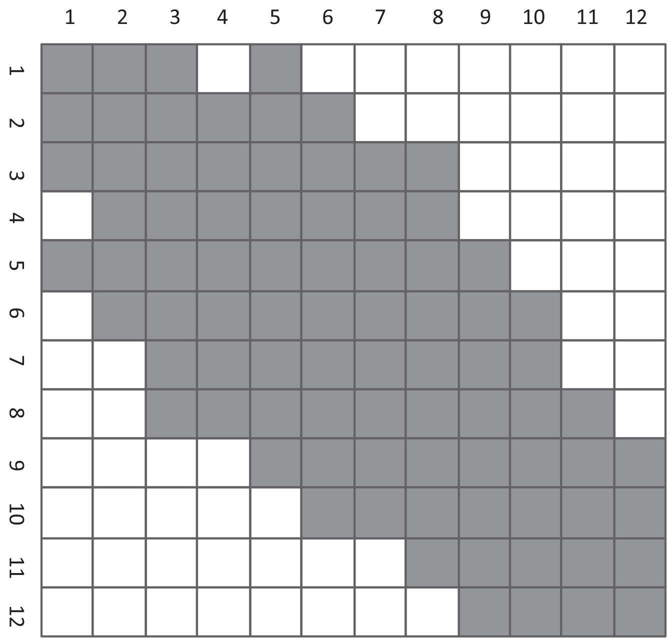

Fig. 4. The structure of a Schur complement matrix produced by dataset with 12 images, each square is a block containing 6*6 elements. The gray squares represent the blocks with non-zero elements, the blank squares represent blocks with zero elements.

3.5. Compressed sparse row format

Matrix-vector multiplications need to be calculated during each PCG iteration. So the Schur complement normal matrix **S** must be stored, otherwise it has to be calculated by observations data in each iteration of PCG. Some methods do not store normal matrix (Byröd and Åström, 2010), they have to calculate the whole normal matrix in each iteration of PCG (usually hundreds times to be converged) which is extremely time consuming. Agarwal did not store the Schur complement normal matrix but stored matrix $\mathbf{V}_C$, **W** and $\mathbf{V}_P$ in Eq. (9) instead (Agarwal et al., 2010). In that case, $\mathbf{V}_C$ and $\mathbf{V}_P$ are both block diagonal matrices which are easy to be stored, but the size of matrix **W** is $6m$*$3n$ which is still much bigger than that of the Schur complement matrix S ($6m$*$6m$) since that $n \gg m$.

If we store full matrix **S** directly, a large memory space is needed when the image number is very big, for an instance, ten thousands even hundreds of thousand more. Thus the matrix compression is crucial. Note that the matrix **S** is sparse. The sparsity can be calculated by the ratio of the number of non-zero elements and the number of all elements in the matrix. We can compress the sparse matrix by abandoning the zero elements. The CSR format is a standard and general sparse matrix compression format, the details can be found in references (Bell and Garland, 2009; Wu, 2011.).

3.6. Block-based sparse matrix compression storage

In a structure from motion problem, the normal matrix is a block-wise sparse matrix as shown in Fig. 4. It can be divided into a number of sub-matrices. The size of sub-matrices is related to the number of unknown parameters. In our case, for an instance, the size of sub-matrix is 6*6. If one uses CSR format, block

Table 1
A block-wise sparse matrix with 10*10 sub-matrix, where the shadow blocks are non-zero sub-normal-matrix, the others are zero blocks.

ID	0	1	2	3	4	5	6	7	8	9
0	■					■				
1		■					■			
2			■					■		
3				■					■	
4					■					■
5	▨					■				
6		▨					■			
7			▨					■		
8				▨					■	
9					▨					■

structure of the normal matrix will be destroyed. More importantly, during the bundle block adjustment, the normal matrix needs to be updated after normalization of each point. Positions of the sub-normal-matrix of each point in the whole normal matrix need to be calculated, and the block in that position should be read out and added by the newly computed sub-normal-matrix of the current point, then the result is written back into that block of the whole normal matrix. If the normal matrix was stored in the CSR format, every elements of the current sub-normal-matrix should be updated one by one. Thus it's a very complicated and time consuming procedure.

A Block-based Sparse Matrix Compression (BSMC) format was introduced according to the special block structure of the normal matrix in a photogrammetry, remote sensing and computer vision problem. In this format, only the non-zero sub-block-matrix and its position indices which includes four elements: beginning row ID, beginning column ID, row number and column number in the full matrix are stored.

In Table 1, the normal matrix is a block-wise sparse matrix consisting of 10*10 sub-matrix which have a size of $m*m$, where m is the number of unknown parameters. Here, m is 6. So the size of every sub-matrix is 36 doubles. The size of the block info is 4 integers. Every sub-block data is supplemented with a block info structure which identifies the position and size of the sub-matrix in the full matrix as shown in Table 2. Besides, the normal matrix is usually a symmetric matrix, so only the diagonal sub-matrix and the upper triangle part (blocks filled with black color) need to be stored.

As can be seen in Table 2, BSMC storage can be divided into block data part and block info parts. The size of the block data part is smaller than the first array in the CSR format since only the diagonal blocks and the upper triangle part of blocks are stored. The block info part is also much smaller than the rest of CSR. So the BSMC storage format is overall more efficient than CSR while dealing with the block-based sparse matrix. To further investigate the compression efficiency of the two storage formats, take the sparse matrix in Table 1 for an instance, we assume that all elements in the sub-normal-matrix is non-zero, the required memory of the sparse matrix in different formats are listed in Table 3.

One can easily tell that the proposed BSMC format only need about half of the required memory as the CSR format does. As the whole sparse matrix size is getting bigger, the BSMC format will be more efficient than that of the CSR format.

3.7. Sparse normal matrix update

As mentioned in the previous section, the normal matrix needs to be updated point by point. Each point will produce a sub-normal-matrix which will be added in the whole normal matrix. The procedure of updating normal matrix is given as in Fig. 5.

4. Experiments and analysis

To further investigate the performance of the proposed method, a series of test datasets with different sizes are tested with the conventional method and our proposed method. The tie point identification was performed before the bundle block adjustment. In this paper, we focused on algorithms and technical details of the bundle block adjustment. The IOPs are also calibrated in the previous procedure, and only EOPs and GPC are set as unknowns in the bundle block adjustment. First, we compared the memory requirement of three methods, the uncompressed normal matrix, the compressed normal matrix in the CSR format, the compressed normal matrix in the proposed BSMC format. Then, the conventional LM algorithm and PCG algorithm combined with BSMC are tested. The simple and effective Jacobi preconditioner are adopted in PCG. The memory requirements, total run time and the overall precisions were compared and analyzed. All the experiments are performed on a common laptop computer equipped with the Inter (R) Core(TM) i5-33320M CPU 2.60 GHz, 8.00GB RAM, and 64 bit Window 7 Operating System. Note that this is a relatively low-performance machine.

4.1. Datasets

The first three datasets of the test data include cell phone images and UAV images. Other datasets are all consist of internet images which were downloaded from a public source website which was built and shared by Sameer Agarwal in University of Washington (Agarwal et al., 2010). These test datasets have different sizes from the smallest one with 40 images to the biggest one with 13682 images. Those images are mostly collected by uncalibrated camera such as cell phones, automatic digital cameras and digital single lens reflex cameras and so on. They are mostly not regularly arranged. Thus the corresponding normal matrices have no band characteristic, they have to be inverted

Table 2
The Block-based Sparse Matrix Compression (BSMC) storage format.

Block ID	0	1	2	3	…	13	14
Block info	(0,0,6,6)	(0,5,6,6)	(1,1,6,6)	(1,6,6,6)	…	(8,8,6,6)	(9,9,6,6)
Block data	Sub-block1	Sub-block2	Sub-block3	Sub-block4	…	Sub-block13	Sub-block14

Table 3
Comparison of the required memory of sparse matrix in Table 1 in CSR format and BSMC format.

Method	Full size	CSR format				BSMC format		
		Array1	Array2	Array3	Total	Block info	Block data	Total
Size	10*10*36	20*36	20*36	10	8680	15*4	15*36	4560
Data type	Double	Double	Integer	Integer		Integer	Double	
Memory (Bytes)	28,800	5760	2880	40		240	4320	

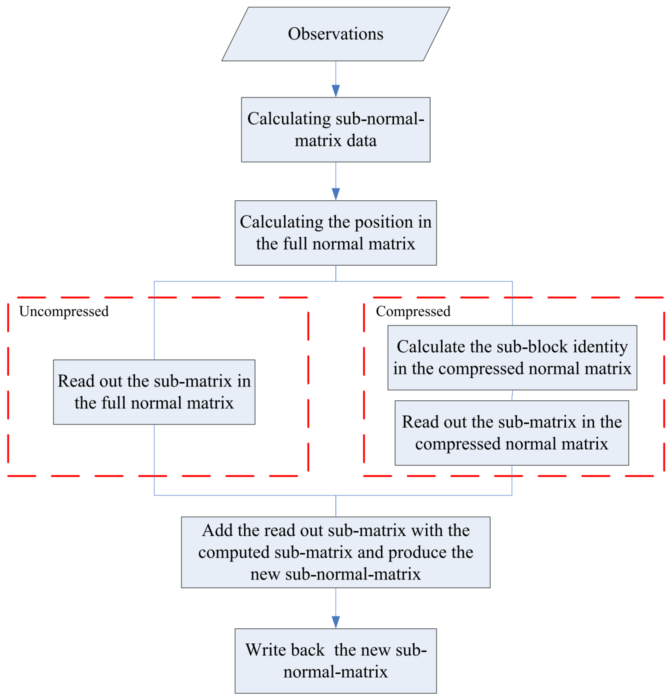
Observations
Calculating sub-normal-matrix data
Calculating the position in the full normal matrix
Uncompressed
Read out the sub-matrix in the full normal matrix
Compressed
Calculate the sub-block identity in the compressed normal matrix
Read out the sub-matrix in the compressed normal matrix
Add the read out sub-matrix with the computed sub-matrix and produce the new sub-normal-matrix
Write back the new sub-normal-matrix

Fig. 5. The procedure of updating the normal matrix with both the uncompressed and compressed normal matrix.

directly in the bundle block adjustments in the LM algorithm. The IOPs are all calibrated before the bundle block adjustment. So we have not introduced self-calibration here. As mentioned before, this paper mainly focuses on the efficiency and capability of the bundle block adjustment. Image qualities are different from each other especially for internet images. There are a lot of gross errors in the dataset. A gross point detection and elimination process is applied both before and during the bundle block adjustment.

4.2. Comparison of memory requirement of the normal matrix

In this section, normal matrices are stored in three ways, the uncompressed normal matrix, the compressed normal matrix in the CSR format and the compressed normal matrix in the proposed BSMC format. All 8 datasets of normal matrix are tested with these three methods. The compression ratio is calculated as the original-size divided by the compressed-size. The final results are listed in Table 4.

As can be seen in Table 4 and Fig. 6, Both CSR format and BSMC format can decrease the memory requirements compared to the conventional LM algorithm without any compression. Memory requirement of the normal matrix in BSMC format is smallest while dealing with the same dataset. The proposed BSMC method is more efficient than that of CSR format. As shown in Figs. 7 and 8, the compression ratios of the two methods are both highly related to the sparsities of the normal matrix.

4.3. Efficiency and accuracy assessment

To further investigate the efficiency and accuracy of the proposed method, total 8 datasets are tested with both LM method and PCG-BSMC method. The time and accuracies of bundle block adjustment are compared and analyzed. The detail data can be found in Table 5.

As demonstrated in Table 5 and Fig. 9, time of building normal equation of the same dataset with both LM and PCG-BSMC are almost the same while the time of solving normal equation is different. So the total time is highly depended on the time of solving normal equation. The time of solving normal equation with conventional LM algorithm is growing exponentially as the image number is increasing while that of our proposed method PCG-BSMC is almost linear. Besides, when the image number continues to increase, the conventional LM algorithm can not work any more due to the memory constraint.

The accuracy performance can be found in Table 5 and Fig. 10. The overall accuracies of both LM and PCG_BSMC algorithm are at the same level. That is to say, the proposed PCG-BSMC method can achieve the same accuracy as the conventional LM method while it is more efficient than LM when dealing with large-scale data. It should be also noted that when image number is more than 3000, PCG-BSMC is still available while LM is not.

Table 4
The memory requirements of the normal matrix and look up table with different algorithms (Memory Unit: MB).

Dataset	Images	Sparsity	LM	PCG-CSR				PCG-BSMC			
			Total	Look up table	Normal matrix	Total	Compress ratio	Look up table	Normal matrix	Total	Compress ratio
1	40	0.380	0.439	0.109	0.142	0.253	1.735	0.003	0.099	0.103	4.262
2	84	0.106	1.938	0.484	0.191	0.676	2.867	0.013	0.133	0.147	13.184
3	218	0.126	13.05	3.261	1.331	4.597	2.839	0.090	0.933	1.025	12.732
4	394	0.867	42.63	10.65	27.90	38.57	1.105	0.295	19.63	19.93	2.139
5	961	0.989	253.6	63.40	188.7	252.1	1.006	1.760	132.7	134.5	1.886
6	1936	0.923	1029.4	257.3	714.1	971.4	1.060	7.145	502.4	509.6	2.020
7	4585	0.105	5773.9	1443.4	459.3	1902.9	3.034	40.08	323.1	363.2	15.897
8	13682	0.104	51415.2	12853.6	4043.9	16897.9	3.043	357.0	2845.5	3202.6	16.054

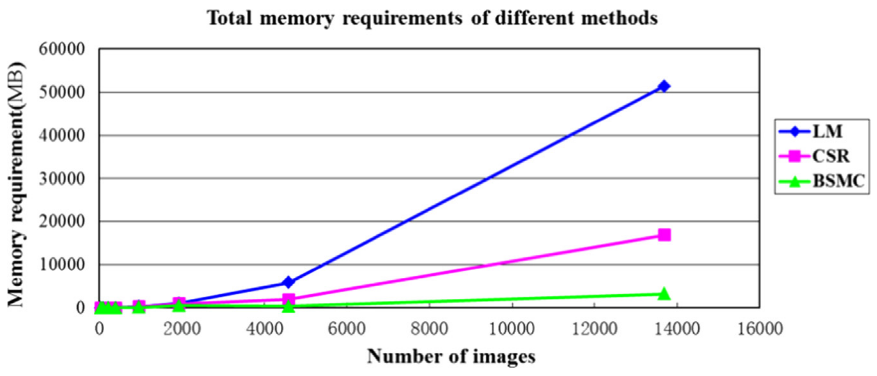

Fig. 6. Total memory requirements of different algorithms: LM, PCG_CSR, PCG_BSMC.

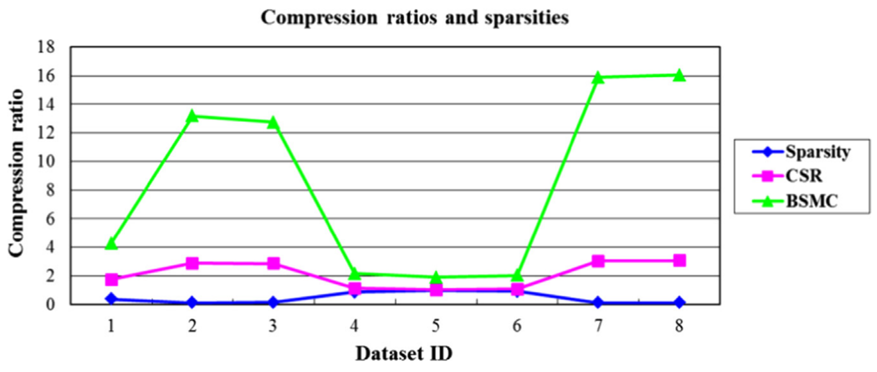

Fig. 7. Sparsity and compression ratios of different compression formats with all 8 datasets.

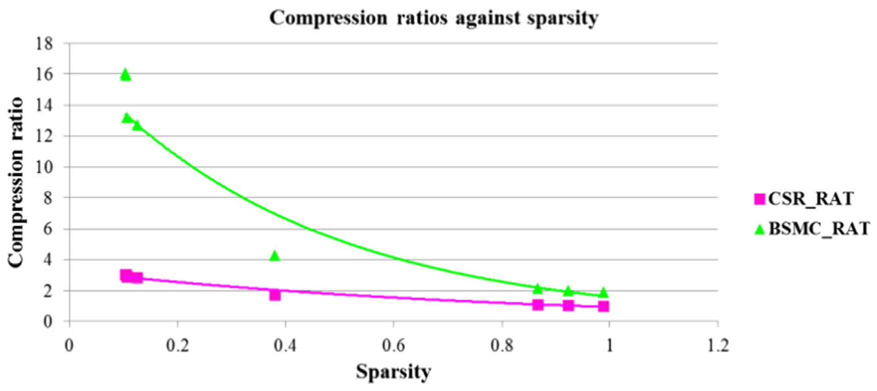

Fig. 8. The compression ratios against sparsity of matrix for CSR and BSMC format.

Table 5
Time and accuracies of bundle block adjustment of different dataset with LM and PCG-BSMC algorithm.

Data ID	Data source	Image number	Ground points	Image points	LM				PCG-BSMC			
					Time of building normal matrix	Time of solving normal matrix	Total time	RMSE	Time of building normal matrix	Time of solving normal matrix	Total time	RMSE
1	Cell phone	40	8272	23681	23 s	< 1 s	27 s	0.472	19 s	< 1 s	19 s	0.533
2	UAV	84	25589	56090	53 s	< 1 s	80 s	0.520	78 s	< 1 s	78 s	0.526
3	UAV	218	49966	179339	156 s	10 s	187 s	0.528	157 s	< 1 s	178 s	0.529
4	Internet	394	20074	105799	99 s	68 s	176 s	0.732	108 s	4 s	112 s	0.715
5	Internet	961	37421	337891	358 s	904 s	1289 s	0.879	388 s	24 s	412 s	0.868
6	Internet	1936	129935	1045645	1229 s	8593 s	9895 s	0.912	1247 s	79 s	1326 s	0.904
7	Internet	4585	264917	1825228	–	–	–	–	2491 s	107 s	2598 s	0.870
8	Internet	13682	891224	5801328	–	–	–	–	13351 s	15286 s	28637 s	0.907

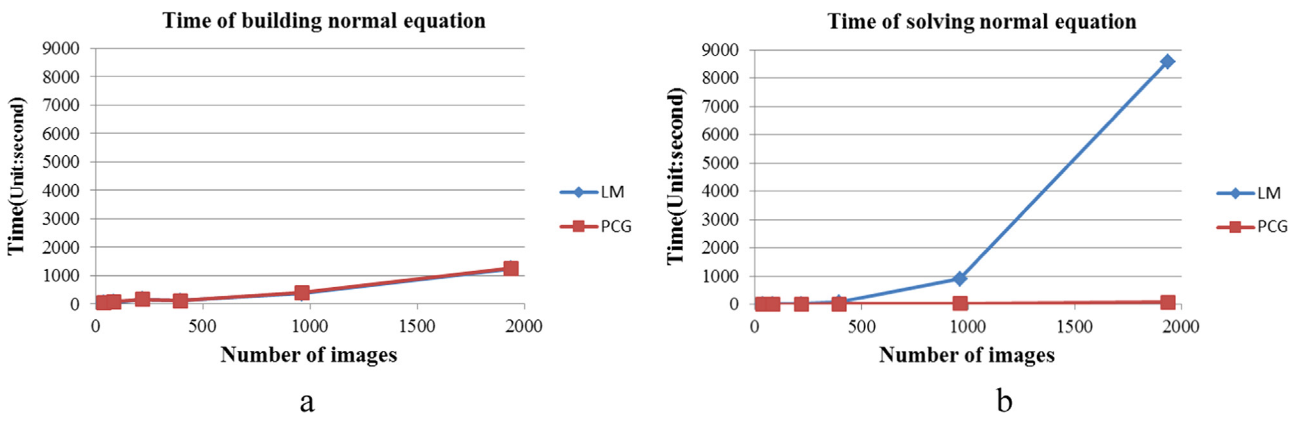

Fig. 9. Time of solving normal equation with LM and PCG-BSMC algorithm. Fig. 4a shows the building time, Fig. 4b shows the solving time.

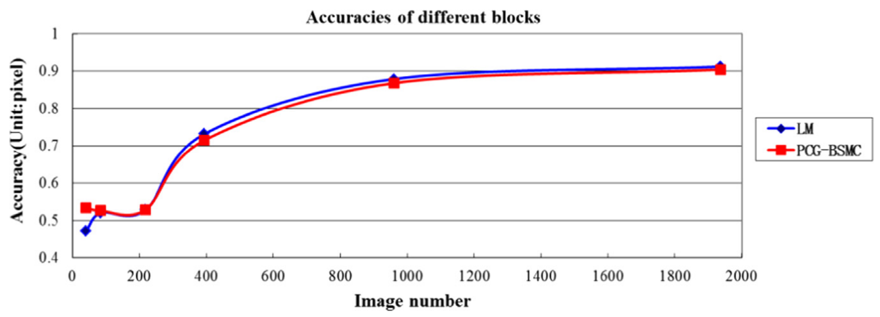

Fig. 10. Accuracies of bundle block adjustment with different datasets.

Table 6
Comparison on Accuracy of Wu et al.'s method and the proposed method.

Data ID	Source	Image number	Ground points	Image points	Reprojection error in pixels	
					Wu's method	Proposed method
1	Internet	52	64063	347143	>1.0	0.625
2	Internet	88	64298	383987	>1.8	0.536
3	Internet	13682	891224	5801328	>1.9	0.907

4.4. Comparison to other works

Wu et al. implemented bundle adjustment with GPU parallel computing. They did not store the normal matrix but compute it on the fly (Wu, 2011.). Their method is fast, but the accuracy is constrained since the GPU only provides single precision computation (Some high performance GPU can provide double precision but the computation speed is compromised, the memory requirement is also increased). Our method is based on double precision on CPU, we explore matrix compression format to decrease the memory requirement of large normal equation. We compared the accuracy of Wu's method and our proposed method. The results are shown in Table 6.

The three datasets are referred to “Figure 2(a)”, “Figure 2(b)”, and “Figure 1(a)” in Wu et al.'s paper respectively (Wu, 2011.). The third dataset is also referred to the eighth dataset of this paper as shown in Table 5. The speed of their method excels out method undoubtably since they adopted GPU parallel computing techniques. But the accuracy of our method is better than their method as listed in Table 6.

5. Conclusion

In this paper, a BSMC method is proposed to combine with PCG algorithm aiming to deal with large-scale remote sensing data. BSMC is compared with the well-known sparse matrix compression algorithm CSR. Conventional LM method and PCG method in bundle block adjustment are reviewed and discussed. The detail procedure of bundle block adjustment with PCG combined with BSMC is described. Total 8 datasets of different kinds of remote sensing data including UAV images, Cell Phone images and internet images are tested with both conventional LM and BSMC-PCG algorithm to compare the efficiency and accuracy performance of these two algorithms. After statistically analyzing all the experiment results, we can conclude that:

1) The proposed Block-based Sparse Matrix Compression method has better performance than CSR method when dealing with bundle block adjustment problem. The compression ratio of BSMC is smaller than that of CSR method, and BSMC is more suitable for bundle block adjustment due to its block-based characteristic.
2) The BSMC-PCG algorithm is more efficient than conventional LM algorithm especially dealing with large-scale remote sensing data. The memory requirement of BSMC-PCG is much smaller than that of LM method since that the BSMC algorithm can largely compress the normal matrix. When the image number is getting bigger (>3000), the conventional LM method can no longer work due to the memory constraint while the BSMC-PCG method is still available. The proposed method is available as long as the compressed normal matrix can be stored in the memory. As the image number continues to increase, when there was not enough memory even for the compressed normal matrix, the proposed method was disabled.

More real data should be tested to verify the robustness of this method. It is not parallelized yet. The parallel computing technology will be implemented in our future work.

Acknowledgment

Thanks to all those anonymous reviewers who gave a lot of valuable suggestions! This project is funded by China Postdocroral Science Foudation under Grant 2015M572224, the National Natural Science Foundation of China under Grant 41322010, the Key Laboratory for Aerial Remote Sensing Technology of National Administration of Surveying, Mapping and Geoinformation (NASG) under Grant 2014B01 and the Fundamental Research Funds for the Central Universities, China University of Geosciences (Wuhan) under Grant CUG160838.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.cageo.2016.04.006.

References

Agarwal, S., Snavely, N., Seitz, S.M., et al., 2010. Bundle adjustment in the large. Comput. Vision – ECCV, 29–42.
Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., et al., 2011. Building Rome in a day. Commun. ACM 54 (10), 105–112.
Ai, M., Hu, Q., Li, J., et al., 2015. A robust photogrammetric processing method of low-altitude UAV images. Remote Sens. 7 (3), 2302–2333.
Bell, N., Garland, M., 2009. Implementing sparse matrix-vector multiplication on throughput-oriented processors. In: Proceedings of the 2009 ACM/IEEE Conference on Supercomputing, pp. 1–11.
Besnerais, G.L., Sanfourche, M., Champagnat, F., 2008. Dense height map estimation from oblique aerial image sequences. Comput. Vision Image Underst. 109, 204–225.
Bru, R., MarÍn, J., Mas, J., Tůma, M., 2008. Balanced incomplete factorization. SIAM J. Sci. Comput. 30 (5), 2302–2318.
Byröd, M., Åström, K., 2009. Bundle adjustment using conjugate gradients with multiscale preconditioning. Br. Mach. Vision. Conf.
Byröd, M., Åström, K., 2010. Conjugate gradient bundle adjustment. Lect. Notes Comput. Sci. 6312, 114–127.
Dahlke, D., Wieden, A., 2013. True 3D-information from digital aerial images – edge detection and adjustment from an oblique viewing multi-camera system. In: Proceeding of the JURSE, April 21–23, 2013, Sao Paulo, Brazil.
Dandois, J.P., Ellis, E.C., 2010. Remote sensing of vegetation structure using computer vision. Remote Sens. 2 (4), 1157–1176.
Frueh, C., Sammon, R., Zakhor, A., 2004. Automated texture mapping of 3D city models with oblique aerial imagery. In: Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission, pp. 396–403.
Hestenes, M.R., Stiefel, E., 1952. Methods of conjugate gradients for solving linear system. J. Res. Natl. Bur. Stand. 49 (6), 409–436.
Jian, Y.D., Balcan, D.C., Dellaert, F., 2011. Generalized Subgraph preconditioners for large-scale bundle adjustment. In: Proceedings of the IEEE International Conference on Computer Vision, vol. 6669, pp. 295–302.
Li, R., Saad, Y., 2013. Gpu-accelerated preconditioned iterative linear solvers. J. Supercomput. 63 (2), 443–466.
Mathews, A.J., Jensen, J.L.R., 2013. Visualizing and quantifying vineyard canopy LAI Using an unmanned aerial vehicle (UAV) collected high density structure from motion point cloud. Remote Sens. 5, 2164–2183.
Saad, Y., 1981. Krylov subspace methods for solving large unsymmetric linear systems. Math. Comput. 37 (155), 105–126.
Snavely, N., Seitz, S.M., Szeliski, R., 2008. Modeling the world from internet photo collections. Int. J. Comput. Vision 80, 189–210.
Tong, X., Liu, X., Chen, P., et al., 2015. Integration of UAV-based photogrammetry and terrestrial laser scanning for the three-dimensional mapping and monitoring of open-pit mine areas. Remote Sens. 7, 6635–6662.
Wu, C., 2011. Multicore bundle adjustment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3057–3064, 2011.