Supplementary materials

1. Accessing efficiency of the BSMC and CSR format

To evaluate the accessing efficiency of a compression format, we consider two operations. (1) Given an element or a block in the compressed structure, find its column and row identities in the full matrix; (2) Given the column and row identities of an element or a block in the full matrix, find the corresponding block in the compressed structure.

1.1. The first operation

Given the ID of an element in the first array of the CSR structure, the column ID in the full matrix can be identified immediately by the second array as shown in Figure 1 (left). If the elements are continuously accessed, the row ID in the full matrix can be obtained by row counting. Otherwise, the row ID must be interpolated from the third array. Use a dichotomy algorithm, the maximum searching times is $\log_2 nc$ since the size of the third array is nc where n is the block number and c is the block width.

Given the identity of a block in the first array of the BSMC structure, the column ID in the full matrix is firstly identified by the second array as shown in Figure 1 (right). Similar to CSR format, the row identity can be obtained by row counting if the blocks are continuously accessed, or the row identity must be searched in the third array with the maximum searching times of $\log_2 n$ since the size of the third array is the block number n. Note that one access in CSR structure only gets one element but it gets a block which includes $c \times c$ elements in BSMC structure.

1.2. The second operation

Given the row and column IDs of a element in full matrix, to find its corresponding position in the CSR structure, we first locate the start and end column IDs in the second array from the third array by the row ID in the full matrix as shown in Figure 1 (left), then the column ID of the element in the first array can be searched between the start and end column IDs. The maximum searching times for each element is $\log_2 cs_i$. To find a block, we only have to find the beginning element for each row of this block. Therefore the total searching times for finding a block is equal to $\sum_{i=0}^{c} \log_2 cs_i$ where s_i is the number of non-zero blocks in row i.

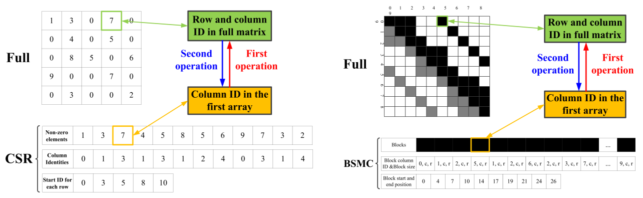
Full
Row and column ID in full matrix
Second operation
First operation
Column ID in the first array
CSR
Non-zero elements
Column Identities
Start ID for each row
BSMC
Blocks
Block column ID &Block size
Block start and end position

Figure 1. Accessing the data in CSR (left) and BSMC (right) format

Format	First Operation	Second Operation
CSR	1 or $\log_2 nc$	$\sum_{i=0}^{c} \log_2 cs_i$
BSMC	1 or $\log_2 c$	$\log_2 s_i$

Table 1. Accessing efficiency of the BSMC and CSR formats

Given the row and column identities of a block in the full matrix, to find its corresponding position in the BSMC structure, we first locate the start and end column IDs of the block in the second array from the third array by the row identity as shown in Figure 1 (right), and then the column ID in the first array is searched between the start and end column IDs with maximum searching times equal to $\log_2 s_i$. Once the first element of the block is determined, the element of the whole block is identified. Apparently, for this operation, the BSMC is more efficient than the CSR.

Table 1 shows the searching times of these two operations for the BSMC and CSR formats. As shown in Table 1, if the elements or blocks in the compressed structure are accessed continuously, the first operation will be immediately performed for the two formats. In the real case, for matrix-vector product, the elements or blocks in the compressed structure are continuously accessed. However, for matrix updating or aggregation, the second operation is frequently applied. The accessing efficiency of BSMC for both the first and second operations is better than the CSR; thus, it's strongly recommended for the compression of block-based sparse matrices, such as the Hessian and RCS.

2. Formation of the RCS

The formation of RCS is a major step in BA. It dominates the most part of the computation complexity. In this section, we look into the details of the formation of a RCS and try to parallelize it. To be specific, given a dataset with 5 cameras

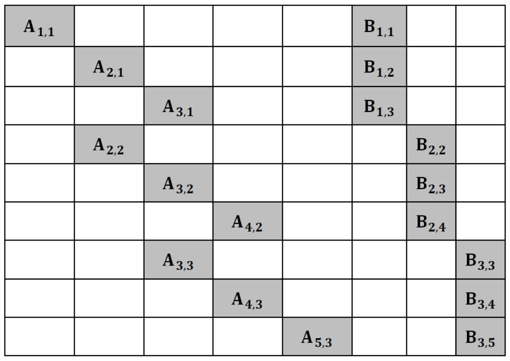

Figure 2. The jacobian matrix where $\mathbf{A}(i,j)$ and $\mathbf{B}(j,i)$ are the camera and landmark part of the jacobian generated by the edge of camera i and landmark j.

and 3 landmarks, the Jacobian is shown in Figure 2, and the Hessian is obtained by normalizing the Jacobian as shown in Figure 3. The damping is not shown in the figure, but it does not affect the structure of the Hessian.

The computation of non-zero blocks of the Hessian are shown in equation (1), (2) and (3).

$$\mathbf{A}_i^T\mathbf{A}_i = \sum_{j=0}^{p_i} \mathbf{A}_{i,j}^T \mathbf{A}_{i,j} \tag{1}$$

Where p_i is the number of landmarks seen by camera i.

$$\mathbf{B}_i^T\mathbf{B}_i = \sum_{i=0}^{q_j} \mathbf{B}_{j,i}^T \mathbf{B}_{j,i} \tag{2}$$

Where q_j is the number of cameras seeing landmark j.

$$\mathbf{P}_{i,j} = \mathbf{A}_{i,j}^T \mathbf{B}_{j,i} \tag{3}$$

According to equation in the main text, conduct Shur complement to the Hessian, we get a matrix $\mathbf{Q} = \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^T$ as shown in Figure 5 (left). Then the RCS is computed by equation $\mathbf{R} = \mathbf{U} - \mathbf{Q}$ as shown in Figure 5 (right). So, to compute Shur complement is actually to compute the matrix $\mathbf{Q}$. Each block $\mathbf{Q}_{i,j}$ in matrix $\mathbf{Q}$ is computed according to equation (4). Note that the block $\mathbf{Q}_{i,j}$ could be zero blocks if there is none common point between camera i and camera j.

$$\mathbf{Q}_{i,j} = \sum_{k=0}^{t_{i,j}} \mathbf{Q}_{i,j,k} \tag{4}$$

$$\mathbf{Q}_{i,j,k} = \mathbf{P}_{i,k}(\mathbf{B}_k^T\mathbf{B}_k)^{-1}\mathbf{P}_{j,k}^T \tag{5}$$

Where k is the landmark ID, $t_{i,j}$ is the number of common points between camera i and camera j.

$$\mathbf{Q} = \sum_{k=0}^{L} \mathbf{Q}_k \tag{6}$$

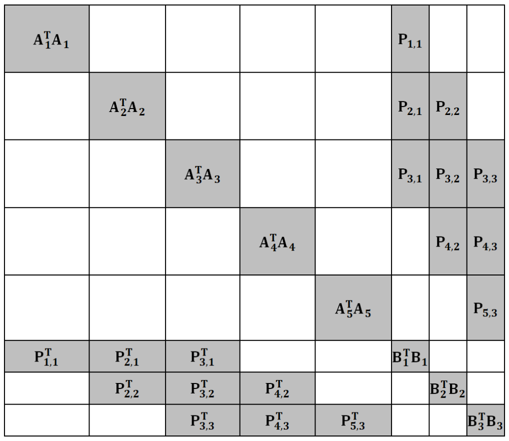

Figure 3. The Hessian matrix.

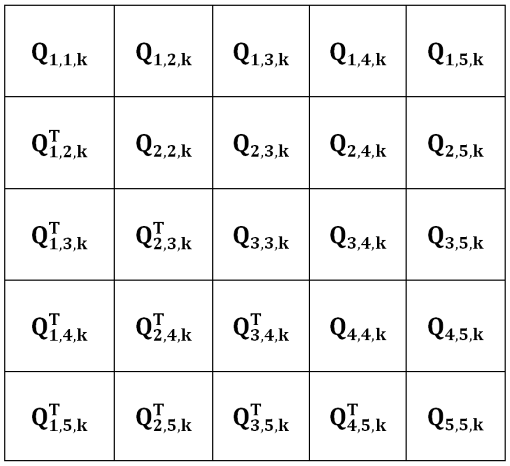

Figure 4. Structure of matrix $\mathbf{Q}_k$. The non-zero blocks are not identified since k is unknown.

Where the matrix $\mathbf{Q}_k$ is a component of the RCS, L is the number of 3D points, and k is the ID of 3D points.

Conduct Shur complement to the 3D points, each of which generates a matrix $\mathbf{Q}_k$ where k is the ID of the landmark, as shown in Figure 4. The matrix $\mathbf{Q}$ is the sum of $\mathbf{Q}_k$ as shown in equation (6). This indicates that the Shur complement trick can be done point by point.

According to the Jacobian shown in figure 2, the 3D point 1 is seen in the camera 1, 2 and 3, the 3D point 2 is seen in the camera 2, 3 and 4, and the 3D point 3 is see in camera 3, 4 and 5, thus the actual structure of matrix $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$ are shown in Figure 6. The number and positions of non-zero blocks are determined by the number and identities of cameras which see this 3D point. For an instance, if a landmark k is seen in m cameras, then the number of non-zero blocks in $\mathbf{Q}_k$ is m^2, and positions of non-zero blocks are determined by the identities of these cameras.

The formation of an RCS can be summarized as follows:

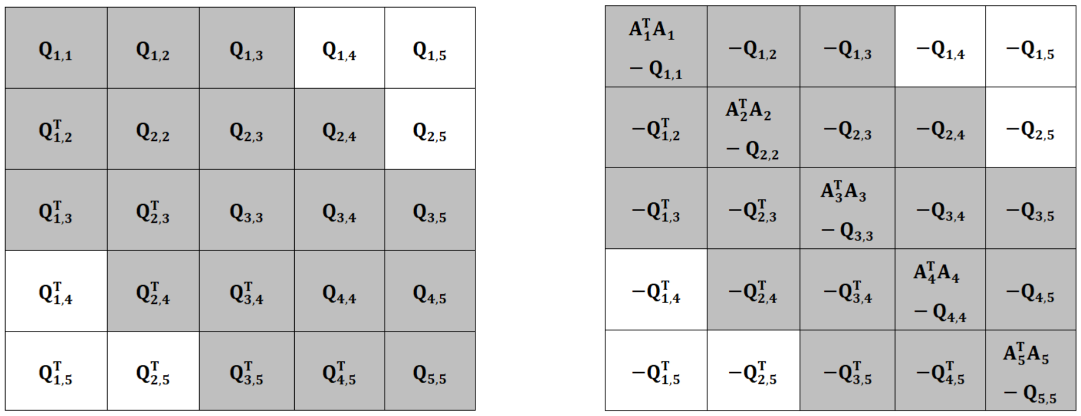
$Q_{1,1}$ $Q_{1,2}$ $Q_{1,3}$ $Q_{1,4}$ $Q_{1,5}$
$Q_{1,2}^T$ $Q_{2,2}$ $Q_{2,3}$ $Q_{2,4}$ $Q_{2,5}$
$Q_{1,3}^T$ $Q_{2,3}^T$ $Q_{3,3}$ $Q_{3,4}$ $Q_{3,5}$
$Q_{1,4}^T$ $Q_{2,4}^T$ $Q_{3,4}^T$ $Q_{4,4}$ $Q_{4,5}$
$Q_{1,5}^T$ $Q_{2,5}^T$ $Q_{3,5}^T$ $Q_{4,5}^T$ $Q_{5,5}$
$A_1^TA_1 - Q_{1,1}$ $-Q_{1,2}$ $-Q_{1,3}$ $-Q_{1,4}$ $-Q_{1,5}$
$-Q_{1,2}^T$ $A_2^TA_2 - Q_{2,2}$ $-Q_{2,3}$ $-Q_{2,4}$ $-Q_{2,5}$
$-Q_{1,3}^T$ $-Q_{2,3}^T$ $A_3^TA_3 - Q_{3,3}$ $-Q_{3,4}$ $-Q_{3,5}$
$-Q_{1,4}^T$ $-Q_{2,4}^T$ $-Q_{3,4}^T$ $A_4^TA_4 - Q_{4,4}$ $-Q_{4,5}$
$-Q_{1,5}^T$ $-Q_{2,5}^T$ $-Q_{3,5}^T$ $-Q_{4,5}^T$ $A_5^TA_5 - Q_{5,5}$

Figure 5. The structure of the matrix $\mathbf{Q}$ (left) and $\mathbf{R}$ (right). The filled blocks are non-zero blocks and the unfilled ones are zero blocks.

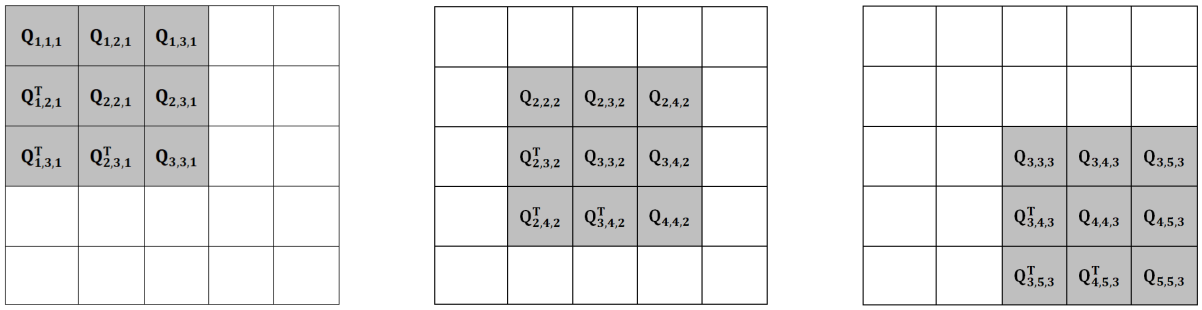
$Q_{1,1,1}$ $Q_{1,2,1}$ $Q_{1,3,1}$
$Q_{1,2,1}^T$ $Q_{2,2,1}$ $Q_{2,3,1}$
$Q_{1,3,1}^T$ $Q_{2,3,1}^T$ $Q_{3,3,1}$
$Q_{2,2,2}$ $Q_{2,3,2}$ $Q_{2,4,2}$
$Q_{2,3,2}^T$ $Q_{3,3,2}$ $Q_{3,4,2}$
$Q_{2,4,2}^T$ $Q_{3,4,2}^T$ $Q_{4,4,2}$
$Q_{3,3,3}$ $Q_{3,4,3}$ $Q_{3,5,3}$
$Q_{3,4,3}^T$ $Q_{4,4,3}$ $Q_{4,5,3}$
$Q_{3,5,3}^T$ $Q_{4,5,3}^T$ $Q_{5,5,3}$

Figure 6. The structure of matrix $\mathbf{Q}_1$, $\mathbf{Q}_2$ and $\mathbf{Q}_3$ generated by landmarks 1, 2 and 3 respectively.

(1) For each 3D point k, compute its Jacobian and Hessian, and then conduct the Shur complement to obtain $\mathbf{Q}_k$;

(2) Update the RCS with $\mathbf{Q}_k$;

(3) Stop until all the 3D points are completed.

According to the above procedures, the formation of RCS can be divided into parallel tasks, with each consisting of a group of 3D points. Those tasks are independent to each other; therefore, they can be executed in a distributed way.