Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A low-overhead asynchronous consensus framework for distributed bundle adjustment*

Zhuo-hao LIU†1, Chang-yu DIAO†‡2,3, Wei XING[1], Dong-ming LU[1,3]

[1]*College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*

[2]*Cultural Heritage Institute, Zhejiang University, Hangzhou 310027, China*

[3]*Key Scientific Research Base for Digital Conservation of Cave Temples, Zhejiang University, Hangzhou 310027, China*

†E-mail: roadliu@zju.edu.cn; dcy@zju.edu.cn

Received Aug. 30, 2019; Revision accepted Jan. 5, 2020; Crosschecked June 4, 2020; Published online Aug. 26, 2020

Abstract: Generally, the distributed bundle adjustment (DBA) method uses multiple worker nodes to solve the bundle adjustment problems and overcomes the computation and memory storage limitations of a single computer. However, the performance considerably degrades owing to the overhead introduced by the additional block partitioning step and synchronous waiting. Therefore, we propose a low-overhead consensus framework. A partial barrier based asynchronous method is proposed to early achieve consensus with respect to the faster worker nodes to avoid waiting for the slower ones. A scene summarization procedure is designed and integrated into the block partitioning step to ensure that clustering can be performed on the small summarized scene. Experiments conducted on public datasets show that our method can improve the worker node utilization rate and reduce the block partitioning time. Also, sample applications are demonstrated using our large-scale culture heritage datasets.

Key words: Structure-from-motion; Distributed bundle adjustment; Overhead; Asynchronous consensus; Partial barrier; Bipartite graph summarization

https://doi.org/10.1631/FITEE.1900451 **CLC number:** TP391.41

1 Introduction

The digitized three-dimensional (3D) model is an important medium for analyzing real-world scenes. It can be used in various fields, including cultural heritage protection (Dhonju et al., 2018; Rahaman and Champion, 2019), city reconstruction (Zhu et al., 2018; Han and Shen, 2019), and cross-media retrieval (Zhai et al., 2014; Peng et al., 2016, 2018; Huang et al., 2020).

Large-scale data pose great challenges to current multi-view 3D reconstruction methods (Fuhrmann et al., 2014; Schönberger and Frahm, 2016; Schönberger et al., 2016). A massive number of images can be captured by unmanned aerial vehicles or digital cameras in a short period of time for reconstruction. However, the reconstruction pipeline exhibits a high computational complexity. With an increase in the number of images, the demand for computation time and memory space drastically increases.

Bundle adjustment (BA) (Triggs et al., 1999), which is a nonlinear least-squares (NLS) problem, is a considerably time-consuming and memory-demanding step in the reconstruction pipeline. The classical Levenberg-Marquardt solver for BA exhibits a time complexity of $O(n^3)$ and a space complexity of $O(n^2)$, where n denotes the number of images. Furthermore, the prevalent incremental structure-from-motion (SfM) method runs BA every

‡ Corresponding author

* Project supported by the Key R&D Program of Zhejiang Province, China (No. 2018C03051) and the Key Scientific Research Base for Digital Conservation of Cave Temples of the National Cultural Heritage Administration, China

ORCID: Zhuo-hao LIU, https://orcid.org/0000-0001-7093-6267; Chang-yu DIAO, https://orcid.org/0000-0001-7744-0889

time new images are registered, adding another factor n to the complexity of the reconstruction pipeline. The efficiency of BA has received significant research attention in the field of computer vision. In-core parallelization (Wu et al., 2011) and specialized algorithms (Lourakis and Argyros, 2005; Konolige, 2010) were proposed to alleviate the aforementioned situation. However, various problems are still observed with a further increase in the scale of reconstruction.

Traditional distributed bundle adjustment (DBA) methods use separate memory and break the computation power bottleneck associated with a single computer. They convert the original problem into multiple smaller BA subproblems. However, we cannot obtain an optimal solution to the original problem by separately solving the subproblems (Zhu et al., 2014). Consensus-based distributed bundle adjustment (CDBA) methods (Eriksson et al., 2016; Liu et al., 2019; Zhang RZ et al., 2020) assign overlapping parameters to the local subproblems. The information of subproblems is passed to each other with the overlapping parameters being fused and distributed via an iterative process, which can result in an optimal solution to the original BA problem. However, considerable computation cost is associated with the manner in which the CDBA methods handle the large-scale BA problem. They introduce an additional overhead when compared with directly solving the BA problem. The actual increase in convergence speed obtained using multiple computation nodes barely meets the expectations. The overhead originates mainly from three sources, network transmission, synchronous waiting for consensus, and block partitioning. Network transmission is required to achieve the initial deployment of subproblems and consensus with respect to communication in each iteration. The time is related to the transmission bandwidth and the amount of data to be transferred. In BA problems, the number of cameras is considerably smaller than the number of points. Therefore, camera consensus (Zhang RZ et al., 2020) that uses only overlapping cameras minimizes the transmission overhead when the bandwidth is kept unchanged. However, the method in Zhang RZ et al. (2020) does not fully take the latter two sources of computation overhead into consideration.

Synchronous waiting occurs because the existing CDBA methods achieve consensus only after the optimal results are obtained by solving all the subproblems. The optimization time of the subproblems may vary considerably. Owing to the difference between the subproblem sizes and the properties of the local solvers, it is difficult to practically determine the time required to solve the local subproblems. Also, the differences between worker nodes in computational power and network transmission rates lead to different times for solving local subproblems. Multiple iterations are required from an initial guess for CDBA, and the typical straggler problem will arise in synchronously distributed systems. Then, all worker nodes must wait for the slowest one in each iteration. The computing resource is wasted while many worker nodes are idly waiting.

Block partitioning in CDBA divides the original BA problem into multiple BA subproblems with overlapping parameters for consensus. It is a nontrivial task because the cameras and points are interdependent. Also, to solve the local problems, the visibility constraints of BA should be satisfied as much as possible. When using only the spatial coordinates of points and cameras, visibility information is lost because spatial adjacency does not necessarily correspond to associations in the visibility graph. Alternatively, the problem can be solved using graph-cut algorithms. Performing graph-cut directly requires eigendecomposition on a large matrix of the bipartite visibility graph, and the computation time can be even longer than that for running consensus iterations.

2 Related works

2.1 Bundle adjustment in SfM

In the multi-view 3D reconstruction pipeline, accurate camera pose is the basis for the subsequent high-quality dense stereo and texture mapping step. SfM recovers the camera parameters of images that are taken from multiple angles, and BA serves as the core optimization step in various SfM methods. In the prevalent incremental SfM method (Schönberger and Frahm, 2016), images are sequentially added to the scene, and the noise and outliers can be handled during the reconstruction process. BA is employed to obtain an optimized estimate each time new images are registered to the scene. In the global SfM method

(Cui and Tan, 2015), a rough estimate of the absolute camera pose can be obtained from the relative pose. Thus, BA refines the camera parameters and the triangulated 3D points. The hierarchical SfM combines the global and incremental methods, in which the local reconstructions and the final merged scene are optimized using BA. For a detailed review and comparison of various SfM methods, refer to Özyeşil et al. (2017).

BA optimizes the coordinates of 3D points and the parameters of cameras based on two-dimensional (2D) observations and visibility information. It is formulated as an NLS problem to make the reprojection coincide with the 2D observations as much as possible. A direct method for solving BA is the Levenberg-Marquardt algorithm, which requires to solve the normal equation and has $O\big((m+n)^3\big)$ time complexity, where m and n are the numbers of points and cameras, respectively. The Schur complement technique (Triggs et al., 1999) can achieve $O(n^3)$ time complexity by solving the reduced equation system. Based on the fact that only a small portion of points are visible to each camera, Lourakis and Argyros (2009) exploited the sparsity of the Jacobian matrix and saved considerable computation time. Wu et al. (2011) implemented a parallel bundle adjustment (PBA) solver based on the inexact Newton-type algorithm to take advantage of in-core parallelism of CPU and GPU.

2.2 Distributed bundle adjustment

DBA seeks to parallel the BA solving process, which is similar to PBA. The difference is that DBA requires independent optimization of each subproblem, and thus the subproblems can be dispatched to multiple computing nodes that are physically separated.

Existing DBA methods fall mainly into two categories. One category decomposes the NLS solving process. Shen et al. (2018) proposed to compute the block matrix and solve the reduced equation system in NLS in multiple nodes. However, the method relies on the internal implementation of SBA (Lourakis and Argyros, 2009) and cannot take advantage of recent efficient solvers such as PBA (Wu et al., 2011) or Google's Ceres Solver (Agarwal and Mierle, 2012). In another category the original problem is transformed into multiple smaller BA subproblems. Different BA solvers can be plugged into the meta-algorithm without modification. The linear decrease in the number of cameras in subproblems can lead to a drastic computational complexity reduction, and the reduction depends on the optimization algorithms of local BA solvers. The local readjustment method (Zhu et al., 2014) achieves optimized local reconstruction but fails to obtain an optimal solution of the original problem with no interaction among the BA subproblems.

The CDBA framework passes information between subproblems by performing consensus in the master node. Eriksson et al. (2016) constructed the consensus framework as a Douglas-Rachford splitting problem. They also provided an initial proof that the algorithm can converge to the local minimum of the original BA problem. Then, Zhang RZ et al. (2020) formulated CDBA based on the alternating direction method of multipliers (ADMM). They further completed the proof of convergence with assumptions about projection depth and camera rotation representation. Multiple iterations are required for the consensus-based methods. All worker nodes must wait for the slowest node to complete the calculation in each iteration to perform a complete consensus. Due to the differences in the computation time and network transmission rates among the worker nodes, a significant amount of computation power is wasted.

Inspired by recent research on the straggler problem in distributed systems (Zhang RL and Kwok, 2014), we extend and generalize the existing consensus methods to perform partial consensus to reduce synchronous waiting overhead. The applicability of asynchronous ADMM to the nonlinear BA problem is studied both in theory and by experiments.

2.3 Block partitioning

The block partitioning problem has been widely studied in the multi-view 3D reconstruction pipeline. Kushal and Agarwal (2012) partitioned cameras based on the canonical views algorithm (Simon et al., 2007). A camera arrangement matrix can be obtained according to clustering results, and then used as a pre-conditioner to ensure the stability of the normal equation. The cluster method involves no overlapping parameters. Another method of rearranging cameras is to use the iterative bandwidth reduction partitioning (IBRP) algorithm (Lu et al.,

2019), which groups cameras and performs camera adjacency matrix bandwidth reduction iteratively. To achieve Internet-scale dense reconstruction, Furukawa et al. (2010) hierarchically divided images to ensure that the number of local images does not exceed a predefined value. In the case that an approximate geometry of the scene is known, image clustering and scene segmentation can be combined (Zhang RZ et al., 2015). In reconstruction methods that are based on the depth map, each image finds its adjacent images (Mostegel et al., 2016, 2018). The number of partitions in this case is equal to the number of images, and the partitions have a high degree of overlap.

According to the type of overlapping parameters in block partitioning, existing CDBA methods can be divided into point consensus (PC) (Eriksson et al., 2016), camera consensus (CC) (Zhang RZ et al., 2020), and critical parameter consensus (CPC) (Liu et al., 2019). PC introduces a large number of overlapping points because the number of points is generally much larger than that of images, which increases the network transmission overhead. Nearly all points will present in each block when images see a large portion of scene points. CC uses only overlapping cameras and has the minimal network transmission overhead, while CPC jointly partitions cameras and points to achieve a uniform partition of subproblems. One common problem of the three methods is that they all rely on spectral clustering, which requires eigendecomposition and can be computationally complex compared to the consensus process.

3 Asynchronous consensus

3.1 Framework overview

The proposed method has the same input and output as the conventional BA methods. It is an alternative of the BA solvers that can run in the distributed environment in large-scale multi-view 3D reconstruction problems. We illustrate how our method can be integrated into the prevalent incremental SfM pipeline in Fig. 1.

Incremental SfM (Schönberger and Frahm, 2016) begins with a two-view reconstruction, and other images are registered to the scene one by one. The initial small-scale BA problem can be handled by conventional BA solvers in a single computing node. The demand for time and memory storage grows fast because BA has high computational complexity. DBA starts to work when the BA problem is too large to handle for a single node. Though the actual starting condition is determined in specific applications, DBA is effective when there are thousands of images in the scene.

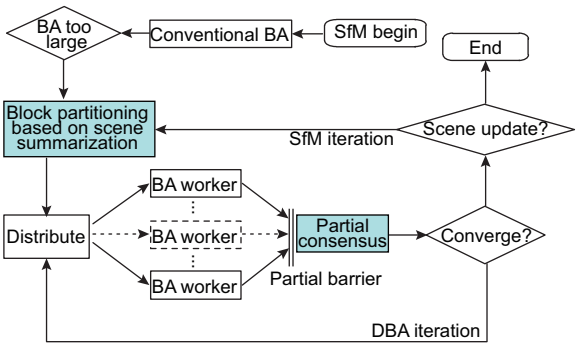

Fig. 1 Low-overhead consensus framework integrated into the incremental structure-from-motion pipeline

The original problem is divided into multiple smaller BA subproblems for optimization in worker nodes. The block partitioning step is performed each time before consensus when the scene is updated. Therefore, the efficiency of block partitioning is critical for the reconstruction pipeline. We summarize similar points and images and then perform clustering on the summarized scene, which is described in detail in Section 4.1.

The subproblems are then distributed to multiple computing nodes that are not physically connected because the local optimizations are independent BA subproblems. Instead of waiting for all the worker nodes to complete the computation, we perform partial consensus to fuse results from faster nodes and reduce waiting time. The improvement is supported by the asynchronous ADMM theory, which is explained in detail in later subsections. In each DBA iteration, block partitioning, local optimization, and consensus are performed in turn. The DBA iteration is terminated when the residual is sufficiently small or the iteration number threshold is reached. Finally, the SfM pipeline ends when no more images can be registered to the scene.

3.2 Consensus

For a camera model with projection operator $\Pi(\cdot)$, the re-projected point is $\hat{\boldsymbol{q}}_{ij} = \Pi(\boldsymbol{c}_i, \boldsymbol{p}_j) \in \mathbb{R}^2$, where $\boldsymbol{p}_j \in \mathbb{R}^3$ is the scene point and $\boldsymbol{c}_i$ represents

the camera parameter. The camera parameters consist of the intrinsic parameters, camera center, and camera orientation. Let $d(\cdot)$ be the Euclidean distance between two 2D points and $\{\boldsymbol{q}_{ij}\}$ be the set of 2D feature points. The reprojection error e_{ij} can be computed by $f_{\mathrm{e}}(\boldsymbol{c}_i, \boldsymbol{p}_j) = d(\hat{\boldsymbol{q}}_{ij}, \boldsymbol{q}_{ij})$. BA jointly optimizes all the parameters of cameras $C = \{\boldsymbol{c}_i\}$ $(i = 1, 2, \ldots, m)$ and the coordinates of sparse 3D scene points $P = \{\boldsymbol{p}_j\}$ $(j = 1, 2, \ldots, n)$ by minimizing the sum of reprojection errors as follows:

$$f(C, P) = \sum_{i=1}^{m} \sum_{j=1}^{n} v_{ij} f_{\mathrm{e}}(\boldsymbol{c}_i, \boldsymbol{p}_j), \tag{1}$$

where $v_{ij} \in \{0, 1\}$ is the visibility between camera c_i and 3D point $\boldsymbol{p}_j$. $v_{ij} = 1$ if $\boldsymbol{p}_j$ is visible in $\boldsymbol{c}_i$; otherwise, $v_{ij} = 0$. The visibility graph $G = (C, P, V)$ consists of two types of vertices, camera vertices C and point vertices P. The visibility information $V = \{v_{ij}\}$ connects the two types of vertices.

Within the consensus framework (Boyd et al., 2011), the unconstrained minimization problem can be converted to the standard form of the global consensus problem. Let the local copies of C and P in the k^{th} local block be $C_k = \{\boldsymbol{c}_i^k\}$ and $P_k = \{\boldsymbol{p}_j^k\}$, respectively, where $1 \le k \le K$ and K is the number of local blocks. The original BA problem (1) can be written as

$$\begin{aligned} \min \quad & \sum_{k=1}^{K} f(C_k, P_k) \\ \text{s.t.} \quad & \boldsymbol{c}_i^k = \boldsymbol{c}_i, \boldsymbol{p}_j^k = \boldsymbol{p}_j. \end{aligned} \tag{2}$$

The solution to problem (2) is equivalent to that to the original problem when all reprojection costs present in the local subproblems. Note that the parameters of C_k and P_k not involved in the local optimization are always identical to their corresponding global value. Therefore, each local subproblem holds only parameters slightly more than $1/K$ of the total number of parameters.

The above constrained problem with equality constraints can be solved using the augmented Lagrangian method. We simplify the notation by stacking the variables from (C_k, P_k) to x_k and denote $(x_1, x_2, \ldots, x_K)$ as $\boldsymbol{x}$. $\bar{\boldsymbol{x}}$ is the corresponding consensus value for $\boldsymbol{x}$. Then the Lagrangian L_ρ is

$$\begin{aligned} L_\rho(\boldsymbol{x}, \boldsymbol{y}, \bar{\boldsymbol{x}}) &= f(\boldsymbol{x}) + \boldsymbol{y}^{\mathrm{T}}(\boldsymbol{x} - \bar{\boldsymbol{x}}) + \frac{\rho}{2} \|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_2^2 \\ &= f(\boldsymbol{x}) + \frac{\rho}{2} \left\| \boldsymbol{x} - \left(\bar{\boldsymbol{x}} - \frac{\boldsymbol{y}}{\rho} \right) \right\|_2^2 + C_{\mathrm{L}}, \end{aligned} \tag{3}$$

where C_{L} is a term not related to $\boldsymbol{x}$, $\rho > 0$ is the penalty parameter, and $\boldsymbol{y}$ is the Lagrange multiplier. When $\rho = 0$, Eq. (3) becomes the standard Lagrangian form.

Employing the augmented Lagrangian method and ADMM, the consensus problem can be solved by alternatively optimizing local parameters $\{x_k\}$, consensus value $\bar{\boldsymbol{x}}$, and the Lagrange multiplier $\boldsymbol{y}$. In the alternative optimization process, all variables are fixed as constants except for the target variables. In the $(t+1)^{\mathrm{th}}$ iteration, $\boldsymbol{x}$, $\bar{\boldsymbol{x}}$, and $\boldsymbol{y}$ are optimized as follows:

$$x_k^{t+1} = \arg\min_{x_k} L_\rho(\boldsymbol{x}, \boldsymbol{y}^t, \bar{\boldsymbol{x}}^t), \tag{4}$$

$$\begin{aligned} \bar{\boldsymbol{x}}^{t+1} &= \arg\min_{\bar{\boldsymbol{x}}} L(\boldsymbol{x}^{t+1}, \boldsymbol{y}^t, \bar{\boldsymbol{x}}^t) \\ &= \frac{1}{K} \sum_{k=1}^{K} x_k^{t+1}, \end{aligned} \tag{5}$$

$$\boldsymbol{y}^{t+1} = \boldsymbol{y}^t + \rho(\boldsymbol{x}^{t+1} - \bar{\boldsymbol{x}}^{t+1}). \tag{6}$$

Eq. (4) is performed in each block to update the local copies of parameters x_k. Combining Eq. (3), it can be found that the local optimization is a BA problem with a regularization term, which can be solved by existing BA solvers. Eq. (5) is a simple algebraic averaging and updates the consensus value. Eq. (6) updates the Lagrangian multipliers $\boldsymbol{y}$ by accumulating the difference between local parameters and its consensus value with a ratio factor ρ.

3.3 Partial consensus and asynchronization

Local optimization tasks in Eq. (4) can be assigned to different worker nodes because each of them is an independent local BA problem. The processing in the worker nodes is shown in Algorithm 1. The master node waits until all of the local updates are gathered to perform consensus and update Lagrangian multipliers. Due to the differences in local optimization time and the delay in computation and network transmission, all the worker nodes must wait for the slowest one to run the next iteration.

We exploit the redundancy of the BA problem and propose to perform partial consensus to fully use available computation resources. The partial consensus is performed on S local updates in each iteration, where $1 \le S \le K$. Let the updates from the S local blocks be $\{k_s\}$ and $1 \le s \le S$. The consensus step

Algorithm 1 Processing in worker node k

repeat
  wait
until receive block data from the master node
repeat
  update x_k as Eq. (4)
  send updated x_k to the master node
  repeat
    wait
  until receive updated y_k and $\bar{x}_k$ from the master node
until the master node terminates

in Eq. (5) becomes

$$\bar{\boldsymbol{x}} = \frac{1}{S}\sum_{s=1}^{S}\left(\boldsymbol{x}^{k_s} + \frac{\boldsymbol{y}^{k_s}}{\rho}\right), \tag{7}$$

where $\boldsymbol{x}^{k_s}$ represents the parameters from local worker k_s, $\boldsymbol{y}^{k_s}$ is the corresponding Lagrange multiplier, and ρ is the penalty factor. When $S = K$, the partial consensus degenerates to the synchronous version. The $\boldsymbol{y}^{k_s}/\rho$ term will be canceled when all the local blocks participate in the consensus; then it will be the same as Eq. (5). Therefore, the synchronous consensus can be regarded as a special case of partial consensus.

The processing in the master node is shown in Algorithm 2. In each iteration, the master node waits until S local updates are received. The consensus value is computed according to Eq. (7). Then, the Lagrangian $\boldsymbol{y}$ is updated and sent back to local worker nodes together with the consensus value. The processing in the worker nodes in partial consensus is the same as that in the total consensus. Note that we use the normalized number of iterations S/K to record the iteration counts. When K worker nodes are updated on average, the iteration count increases by one. The count is meaningful for the different numbers of local updates.

The local nodes may miss the consensus in the master node if $S < K$. Therefore, setting a maximum delay iteration τ can force the local updates being merged into the consensus value and $0 \leq \tau < +\infty$. $\tau = 0$ means that partial consensus is not allowed and a large τ tolerates the long-time absence of local workers in consensus. It has a similar effect to S, so we set τ to a relatively large value and use S to control the degree of asynchronization.

Algorithm 2 Processing in the master node

distribute block data to each worker node
initialize iteration count $t = 0$
repeat
  repeat
    wait
  until receive local parameter update x_k^{t+1} from all worker nodes $k \in \{k_s\}$
  compute consensus $\bar{\boldsymbol{x}}^{t+1}$ as Eq. (5)
  update $\boldsymbol{y}^{t+1}$ as Eq. (6)
  send y_k^{t+1} and $\bar{x}_k^{t+1}$ back to worker node k
  $t \leftarrow t + S/K$
until $t \geq t_\theta$
retrieve non-overlapping parameters from all workers

The master iterations terminate when the maximum iteration count $t_\theta = 20$ is reached. The error decreases fast in the first few iterations for consensus methods. Choosing a larger t_θ may result in a smaller final error, but the differences are very small. Therefore, it does not affect the final results about the utility rate of worker nodes and the convergence process.

The partial consensus in our framework is based on the asynchronous ADMM algorithm. For the analysis of convergence, He and Yuan (2012) gave a proof of the $O(1/t)$ convergence rate of the ADMM method for convex problems, where t is the iteration count. It can also be applied to the convex consensus problem because the consensus framework is a special case of ADMM. Zhang RL and Kwok (2014) further proved that the convergence rate of the asynchronous method is $O(\frac{1}{t}\cdot\frac{K}{S}\cdot\kappa)$ under the assumption that the results of local subproblems come with the same probability and $f(x)$ is a convex function, where κ is the average number of iterations for the information of all worker nodes to be incorporated in master nodes once.

However, this proof fails because the objective function in Eq. (1) is non-convex. Zhang RZ et al. (2020) used the constructive proof technique to show that CDBA would converge under some mild assumptions about the representation of the camera rotation and depth of projected points, which guarantee the local Lipschitz-continuous property of $\nabla f(x)$ and are reasonable for practical SfM problems. Combining the convergence theorem in the asynchronous consensus and the convergence of synchronous CDBA, our partial consensus framework for CDBA can be shown to converge to a local minimum of Eq. (1).

For convergence rate analysis, existing works (He and Yuan, 2012; Giselsson and Boyd, 2017; Boţ and Csetnek, 2019) require $f(x)$ at least be convex, which is not satisfied in our case. Empirically, we find that the cost in CDBA quickly decreases in the

first few iterations and then converges sub-linearly. The asynchronous consensus method has an additional factor $K\kappa/S$ compared with the synchronous method (Zhang RL and Kwok, 2014), in which the effect of the factor K/S is canceled out using the normalized iteration count. Furthermore, the factor κ is roughly equal to one because the consensus operation in Eq. (7) is simple and fast. Therefore, the convergence rate in terms of the number of iterations for the asynchronous method is roughly equal to that for the synchronous method. Because the optimization computations in worker nodes are the same, our partial consensus method works asynchronously to reduce the waiting overhead and improve the convergence speed.

4 Scene summarization based block partitioning

4.1 Block partitioning in CDBA

The observations in local blocks of the K worker nodes are $\{O_k\}$, where $1 \leq k \leq K$ and $\cup_{k=1}^{K} O_k = O$. Each observation o_{ij} corresponds to a reprojection cost $f_{\mathrm{e}}(\boldsymbol{c}_i, \boldsymbol{p}_j)$ that is computed using the camera parameter $\boldsymbol{c}_i$ and point coordinates $\boldsymbol{p}_j$. Therefore, the corresponding cameras and points must be present in the same subproblem.

In CDBA, the information of different local blocks is passed to each other with the overlapping cameras and points being fused and broadcast. The visibility graph G is a connected component in a single scene. For local blocks, the condition $C_{k_1} \cap C_{k_2} \neq \varnothing$ or $P_{k_1} \cap P_{k_2} \neq \varnothing$ is satisfied for at least one pair of (k_1, k_2) and $1 \leq k_1, k_2 \leq K$. The actual number of overlapping parameters N_{ol} varies for different block partitioning methods. The strategy of partitioning the scene points and then performing CC generally has the smallest number of N_{ol} compared with other strategies.

However, there are usually a large number of scene points in BA problems. Directly performing normalized-cut (Shi and Malik, 2000) or bi-clustering (Dhillon, 2001) consumes considerable time because it is not trivial to perform the eigendecomposition-based graph-cut. The visibility graph G is essentially a bipartite graph that has connections only between camera vertices and point vertices.

Merging vertices with identical visibility information does not change the property of the original graph, and can result in a more compact weighted graph. Based on the idea, we slightly relax the constraints and merge vertices that have similar visibility. The partitioning of the original scene elements is then recovered from the partitioning of the compact graph and the grouping information.

Let the set of observations that are not covered be O'. For each observation $o_{ij} \in O'$, $\boldsymbol{c}_i$ and $\boldsymbol{p}_j$ are not in the same block. Assuming that $\boldsymbol{c}_i$ is in block $\{k_c\}$ and $\boldsymbol{p}_j$ in block $\{k_p\}$, each covered observation o_{ij} satisfies $\{k_c\} \cap \{k_p\} \neq \varnothing$. To make o_{ij} be covered, one option is to assign $\boldsymbol{c}_i$ to one of the blocks in $\{k_p\}$, and the other option is to assign $\boldsymbol{p}_j$ to one of the blocks in $\{k_c\}$. For $o_{ij} \in O'$, we add overlapping points to cover o_{ij} when the points are not in the largest block. Otherwise, the overlapping cameras are assigned to the blocks where the corresponding points are located. In this way, most of the overlapping parameters are cameras and the network transmission overhead can be kept at a low level. The partitioned blocks are more uniform because the largest block does not grow in the process of covering observations in O'.

4.2 Scene summarization

Our block partitioning method takes the bottom-to-up approach, first summarizing similar points and cameras and then cutting the graph on the summarized graph to make the subsequent graph-cut computation more efficient (Fig. 2).

By adding edges to or removing edges from the original visibility graph G=(C, P, V), the modified graph $G' = (C, P, V')$ is obtained by finding the vertices with identical visibility. The vertices are then merged to obtain the summarized graph

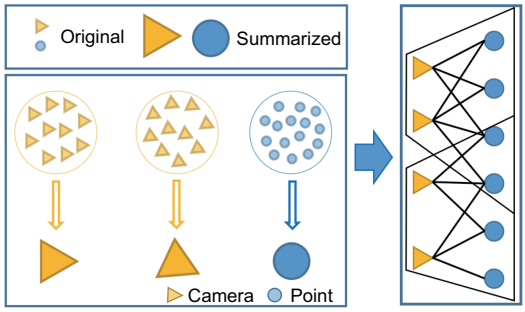

Fig. 2 Illustration of our scene summarization based joint block partitioning method

$G_{\mathrm{M}} = (C_{\mathrm{M}}, P_{\mathrm{M}}, V_{\mathrm{M}})$. If vertices with similar but not identical visibility are merged, G_{A} will lose some information on the original visibility graph.

The notion of the additional graph $G_{\mathrm{A}} = (C, P, V_{\mathrm{A}})$ (Feng et al., 2012) is introduced to keep track of the modified edges from G to G'. G_{A}, G, and G' have the same vertices by definition. All of the edges in G_{A} are initialized as zeros, $v_{ij}^{\mathrm{A}} = 0$. Then the edge in G_{A} is set to $v_{ij}^{\mathrm{A}} = 1$ if an edge is removed from G or set to $v_{ij}^{\mathrm{A}} = -1$ if the edge is added to G. G_{A} can be used to recover G with recovery function f_{r}, where $f_{\mathrm{r}}(G_{\mathrm{M}}, G_{\mathrm{A}}) = G$. In our case, f_{r} is the element-wise addition operation for the two visibility matrices.

The scene summarization step is formulated to minimize the following objective function:

$$g(G_{\mathrm{M}}) = f_{\mathrm{c}}(G_{\mathrm{M}}) + w_{\mathrm{a}} f_{\mathrm{a}}(G_{\mathrm{A}}), \tag{8}$$

where the summarization cost $f_{\mathrm{c}}(G_{\mathrm{M}})$ is proportional to the time for performing the subsequent eigendecomposition step and the additional graph cost $f_{\mathrm{a}}(G_{\mathrm{A}})$ corresponds to the information loss. Both f_{c} and f_{a} are computed as the number of edges in the graph, $f_{\mathrm{c}}(G) = f_{\mathrm{a}}(G) = \sum_{i,j} |v_{ij}|$. The weight factor $w_{\mathrm{a}} = 1$ controls the degree of summarization.

Merging the points and cameras that have the same visibility is favorable because the operation reduces only the summarization cost $f_{\mathrm{c}}(G_{\mathrm{M}})$ and leaves the additional graph cost $f_{\mathrm{a}}(G_{\mathrm{A}})$ unchanged. However, due to the sparsity of detected feature points and the noise in feature matching, the visibility information is usually disturbed and points with many visible images can hardly have identical visibility. Therefore, we relax the constraints and compute vertices similarity based on their visibility information. Let $\mathcal{V}(n)$ denote the set of linking neighbors of vertex n in a bipartite graph. The similarity of two nodes n_1 and n_2 is defined as the intersection-over-union (IoU) of the set of visible vertices:

$$f_{\mathrm{s}}(n_1, n_2) = \frac{\mathcal{V}(n_1) \cap \mathcal{V}(n_2)}{\mathcal{V}(n_1) \cup \mathcal{V}(n_2)}, \tag{9}$$

where $0 \le f_s(n_1, n_2) \le 1$. The two vertices with similarities f_{s} above the threshold $s_\theta = 0.5$ are merged.

Vertices in the bipartite graph are the cameras or the 3D points in the context of the BA problem. The adjacency information in the 3D space can be used to find similar vertices to reduce the candidates for similarity computation. We use the k-dimensional tree model to find the neighbor 3D points and use the camera's position and viewing angle to find neighbor cameras. Points and cameras that meet the requirements are merged as long as the cost is monotonously decreasing. Our method can be regarded as a bipartite graph summarization procedure in which scene information is also used. The candidate neighbors are found based on scene information and the similarity score is computed only for neighbor vertices.

A compact bipartite graph G_{M} is obtained after the scene summarization step. The vertices are the summarized cameras and points, and the weight of each edge is the product of the number of elements included. Biclustering is performed on G_{M} to simultaneously cluster cameras and points to obtain a non-overlapping partition that will not cover all observations. Then the clustering information can be easily propagated to G' to obtain the clustering results of the original vertices.

5 Experimental results

5.1 Settings

In this section, we evaluate our method with existing methods on the public BAL (Bundle Adjustment in the Large) datasets. Example applications of large-scale SfM with DBA for multi-view reconstruction of cultural relics are also presented.

Our method runs on a distributed system with five workstations that have the same configurations. Each workstation has a 2.40 GHz E5-2360 CPU with 32 cores and 64 GB memory. The operating system (OS) is the 64-bit Linux OS. They are connected using a star topology, in which the workstation in the center runs the master node and multiple worker nodes can be assigned to each computer. Note that we also assign worker nodes to the center workstation because the consensus operation is simple and fast. The actual network transmission rate is about 10 MB/s in our experiments.

The local optimization and block partitioning modules are written in C++. The two modules are wrapped in the framework implemented using Python. Local subproblems are solved using Google's Ceres Solver (Agarwal and Mierle, 2012).

The network data transmission is directly based on network sockets. Each worker node runs as a separate server and waits for the master node's data and instructions.

The workstations also serve as remote workers for feature matching and dense stereo in the practical distributed multi-view 3D reconstruction system. Therefore, network data transmission and computation may be delayed. To experiment in a controlled environment, we assume that the delay is proportional to the actual processing time with a delay factor $\gamma = 1$ and occurs randomly with a probability factor $\eta = 0.2$.

5.2 Results

The experiments are conducted on the BAL public datasets (Agarwal et al., 2010) and our cultural heritage (CH) datasets. The statistics of the two datasets are shown in Table 1. The images in BAL are collected from the Internet photo-sharing site, Flickr, and the sparse scene model is reconstructed using the skeleton method (Snavely et al., 2008). In the skeleton method, all images are divided into a skeleton image set and a leaf image set and then reconstructed in turn. BA serves in the final optimization step to refine camera parameters and 3D point coordinates. The BAL dataset provides the camera parameters and point parameters before optimization, as well as the visibility information and 2D observations.

Also, we integrate our method into the state-of-the-art SfM pipeline. We capture the images of our CH dataset using the digital single lens reflex (DSLR) camera in the Yungang Grottoes, the ancient Chinese Buddhist temple grottoes in Shanxi Province, China. For the yg_cave3 dataset, we use incremental SfM and start DBA when the number of images $m_\theta > 3500$. For the larger yg_cave12 dataset, the hierarchical SfM is performed. Both of the incremental and hierarchical SfM pipelines are implemented in Colmap software.

Table 1 Statistics of the public BAL and our CH datasets

Dataset		m	n ($\times 10^5$)	N_o ($\times 10^6$)
BAL	trafalgar	257	0.65	0.23
	ladybug	1723	1.57	0.68
	venice	1778	9.94	5.00
	final	13 682	44.60	28.99
CH	yg_cave3	4217	22.80	25.21
	yg_cave12	13 453	50.30	34.74

m: number of images; n: number of 3D points; N_o: number of 2D observations

To examine the relationship between the utilization rate of the worker nodes and the partial barrier S, we visualize the workload of all the computing nodes. The experiment is performed on the ladybug dataset. For easy illustration, we set the number of local blocks $K = 8$. In this case, there are eight worker nodes (W0–W7) and one master node (M). Fig. 3 shows how the workload of the computing nodes changes with the asynchronous parameter S. On the workload graph, we use two different colors to indicate the time taken for data transfer and for local optimization. In the row corresponding to the master node, black blocks represent the time for the consensus. The vertical red lines represent the time to start the consensus, and each consensus requires at least S worker nodes perform local optimization.

When $S = K = 8$, the asynchronous method degenerates to the existing synchronous method, in which all nodes must wait for the slowest node. $S = 6$ means that there can be two slowest nodes deferred to the next iteration. For example, worker nodes W4 and W7 are not required to participate in the third consensus, and the consensuses for the two nodes are deferred to later iterations. Other nodes can send the optimized parameters to the master node immediately without waiting for the two slower nodes. However, when there are more than two slower nodes, the waiting still happens as before the second consensus. Continuing lowering S results in less time for nodes to be idly waiting. A small asynchronous parameter S close to one will keep all worker nodes busy when the consensus is fast.

The experiment in Fig. 4 shows how existing methods, iterative bandwidth reduction procedure (IBRP) (Lu et al., 2019), CPC (Liu et al., 2019), CC (Zhang RZ et al., 2020), and PC (Eriksson et al., 2016), as well as our method with different S, converge in terms of the running time. For the four reference methods, CPC balances cameras and points in block partitioning and is faster than the other three methods. When $S = K$, our method degenerates to synchronous processing. The convergence rate of our method is close to that of CPC because joint partitioning is also performed in our method, and the local space adjacency information is incorporated in the partitioning step. Using the partial

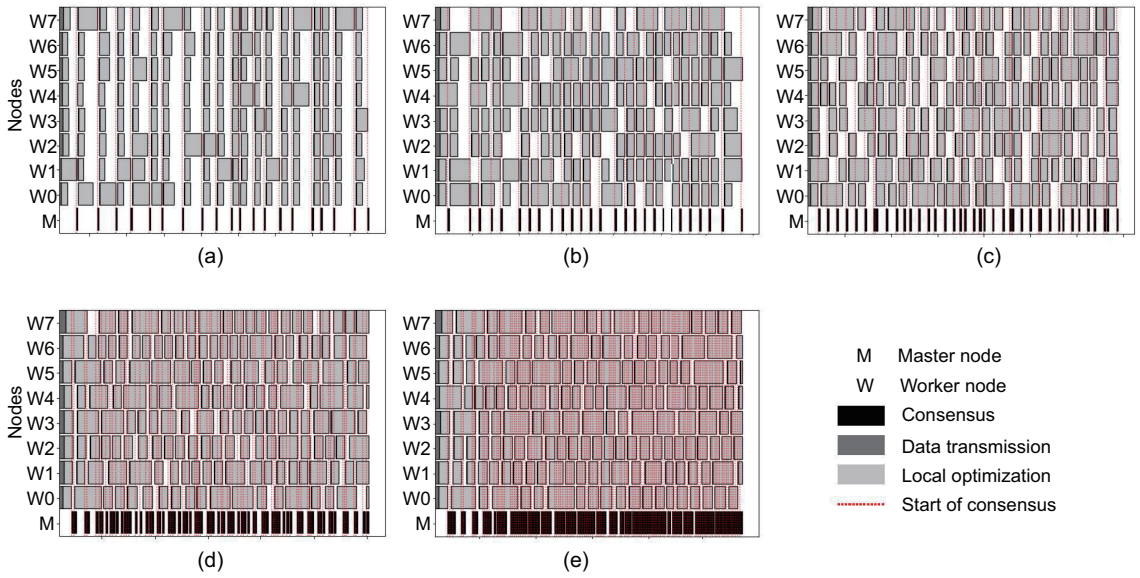

Fig. 3 Workload of the computing nodes in our asynchronous CDBA method: (a) $S=8$; (b) $S=6$; (c) $S=4$; (d) $S=2$; (e) $S=1$

The experiment is run on the ladybug dataset with $K=8$. References to color refer to the online version of this figure

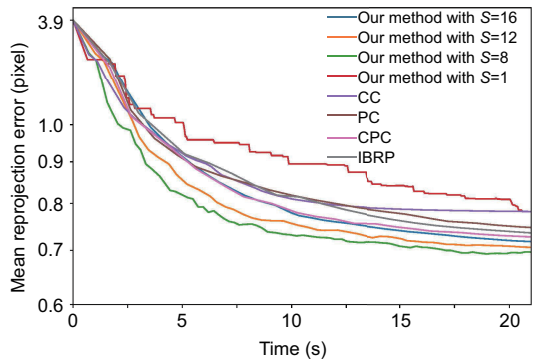

Fig. 4 Comparison of the convergence speed for different DBA methods with $K=16$

References to color refer to the online version of this figure

barrier and setting a smaller S may further improve the convergence speed. For $S \geq K/2$, a smaller S leads to a higher convergence speed though the convergence curve is less smooth in the local area.

The utilization rate can be substantially improved with partial consensus. However, an extremely low S may result in worse results. The reason is that in each consensus of the asynchronous method, only overlapping parameters from the S subproblems are fused. When S is too small, the consensus value will be biased to the bad value of some local parameters and is hard to recover in the subsequent local nonlinear optimization. A large S can succeed because of the large amount of redundancy in the BA problem. In theory, the coordinates of a 3D point can be determined by two viewing cameras. In BA, however, there are usually more cameras seeing the same 3D point. The iterative process still converges when S is properly chosen though there are some local disturbances.

Table 2 shows that the worker node utilization rate u is close to those of existing synchronous methods. We compare the results of different DBA methods that run in the same period. The datasets with a larger partition number K have a lower u because more worker nodes are more likely to have delayed nodes. The running time is set to $10\bar{t}$, and all the methods have the same running time. The mean reprojection error e of our asynchronous method is lower than those of the synchronous methods. Further reduction of S will increase the utilization rate, but the convergence performance is worse as suggested by Fig. 4.

Fig. 5 shows the influence of the number of worker nodes. As the number of nodes increases, the computation time and waiting overhead decrease. However, due to the time required for initializing the

Table 2 Comparison of our method with existing methods

Dataset	K	$\bar{t}_{\mathrm{BA}}$ (s)	$\bar{t}$ (s)	Utilization rate					Mean reprojection error (pixel)				
				Ours	CPC	CC	PC	IBRP	Ours	CPC	CC	PC	IBRP
trafalgar	8	1.4	0.45	0.82	0.36	0.35	0.38	0.38	0.62	0.75	0.81	0.79	0.77
ladybug	16	4.9	0.64	0.71	0.36	0.35	0.36	0.34	0.78	0.88	0.91	0.89	0.90
venice	16	31	2.60	0.53	0.23	0.20	0.18	0.22	0.81	0.85	0.96	0.95	0.91
final	64	179	4.30	0.55	0.33	0.28	0.24	0.34	1.02	1.25	1.18	1.14	1.14
yg_cave3	64	159	3.80	0.63	0.36	0.35	0.43	0.39	1.81	1.90	2.04	1.98	1.94
yg_cave12	128	265	3.00	0.55	0.35	0.32	0.31	0.35	1.83	1.99	1.94	2.05	1.92

K: number of worker nodes; $\bar{t}_{\mathrm{BA}}$: average time for running one iteration of the conventional BA solver; $\bar{t}$: average time for running one iteration of our method

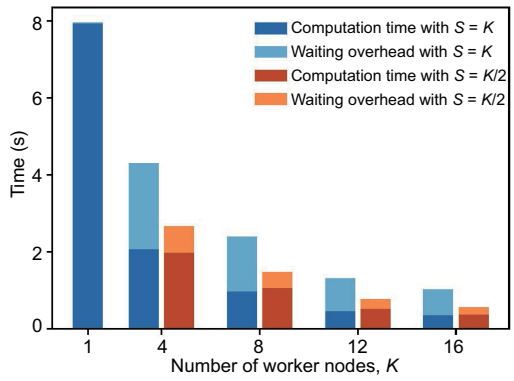

Fig. 5 Computation time and waiting overhead with different numbers of worker nodes
When $S = K$, our method runs synchronously. We set $S = K/2$ to perform asynchronous processing for our method (When $K = 1$, $S = 1$, as $S \geq 1$)

local updates and the data transmission, the performance improvement will be less significant for a larger number of worker nodes. Compared with synchronous processing, using the partial barrier to perform asynchronous consensus requires similar computing time, but the waiting overhead is much less.

In DBA, an additional block partitioning step is introduced, which does not exist in the conventional BA method. Our block partitioning method summarizes the scene before performing biclustering. We compare the block partitioning with or without scene summarization (SS). Let t'_{bc} and t_{bc} represent the time of biclustering-based block partitioning with and without SS, respectively. The total time required for our partitioning method is recorded as t_{ssbp}.

Table 3 shows the comparison on the BAL datasets. With SS, the time required for biclustering can be greatly reduced. The computation time t'_{bc} roughly equals the iteration time of consensus computation, and the method without SS takes about 10 iterations.

Fig. 6 visualizes the results of SfM reconstructions of the two scenes in the CH datasets. For the yg_cave12 dataset, DBA is performed as the final optimization step in the hierarchical SfM pipeline. The global BA is executed 61 times and about 4 min is saved each time compared with the conventional BA solvers.

Table 3 Comparison of block partitioning time with and without scene summarization

Dataset	t_{bc} (s)	t_{ssbp} (s)	t'_{bc} (s)
trafalgar	1.2	0.46	0.35
ladybug	4.3	0.47	0.31
venice	53	3.4	2.2
final	92	4.5	2.4
yg_cave3	80	4.2	1.8
yg_cave12	175	7.9	3.8

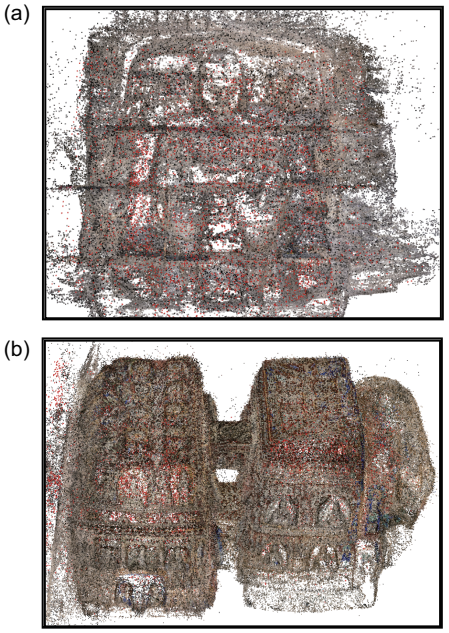

Fig. 6 Visualization of SfM results with our method as the global BA solver: (a) yg_cave3; (b) yg_cave12

6 Conclusions and future work

This study explores to reduce the additional overhead in consensus processing and block partitioning to improve the efficiency of CDBA. We analyze the asynchronous consensus method for DBA and introduce the partial barrier based method to improve the worker node utilization rate. Moreover, using the bipartite graph summarization framework, we propose an efficient scene-summarization method for block partitioning. Experiments show that our method outperforms existing methods in terms of the worker node utilization rate and convergence speed. Also, the applications in large-scale heritage reconstruction show that our method can be readily plugged into the modern SfM pipeline.

In the future, we hope to experiment on larger datasets to further exploit the potential of the proposed method. Besides, we plan to design centerless architectures to make the consensus computation more robust under extreme network conditions.

Contributors

Chang-yu DIAO, Wei XING, and Dong-ming LU guided the research. Chang-yu DIAO and Zhuo-hao LIU designed the research. Zhuo-hao LIU drafted the manuscript. Chang-yu DIAO helped organize the manuscript. Zhuo-hao LIU revised and finalized the paper.

Compliance with ethics guidelines

Zhuo-hao LIU, Chang-yu DIAO, Wei XING, and Dong-ming LU declare that they have no conflict of interest.

References

Agarwal S, Mierle K, 2012. Ceres Solver. https://ceres-solver.org [Accessed on Feb. 23, 2020].

Agarwal S, Snavely N, Seitz SM, et al., 2010. Bundle adjustment in the large. Proc 11th European Conf on Computer Vision, p.29-42. https://doi.org/10.1007/978-3-642-15552-9_3

Boţ RI, Csetnek ER, 2019. ADMM for monotone operators: convergence analysis and rates. *Adv Comput Math*, 45(1):327-359. https://doi.org/10.1007/s10444-018-9619-3

Boyd S, Parikh N, Chu E, et al., 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found Trends Mach Learn*, 3(1):1-122. https://doi.org/10.1561/2200000016

Cui ZP, Tan P, 2015. Global structure-from-motion by similarity averaging. IEEE Int Conf on Computer Vision, p.864-872. https://doi.org/10.1109/ICCV.2015.105

Dhillon IS, 2001. Co-clustering documents and words using bipartite spectral graph partitioning. ACM Int Conf on Knowledge Discovery and Data Mining, p.269-274.

Dhonju HK, Xiao W, Mills JP, et al., 2018. Share our cultural heritage (SOCH): worldwide 3D heritage reconstruction and visualization via web and mobile GIS. *ISPRS Int J Geo-Inform*, 7(9):360. https://doi.org/10.3390/ijgi7090360

Eriksson A, Bastian J, Chin TJ, et al., 2016. A consensus-based framework for distributed bundle adjustment. IEEE Conf on Computer Vision and Pattern Recognition, p.1754-1762. https://doi.org/10.1109/CVPR.2016.194

Feng J, He X, Konte B, et al., 2012. Summarization-based mining bipartite graphs. ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, p.1249-1257. https://doi.org/10.1145/2339530.2339725

Fuhrmann S, Langguth F, Goesele M, 2014. MVE: a multi-view reconstruction environment. EUROGRAPHICS Workshop on Graphics and Cultural Heritage, p.11-18. https://doi.org/10.2312/gch.20141299

Furukawa Y, Curless B, Seitz SM, et al., 2010. Towards Internet-scale multi-view stereo. IEEE Computer Society Conf on Computer Vision and Pattern Recognition, p.1434-1441. https://doi.org/10.1109/CVPR.2010.5539802

Giselsson P, Boyd S, 2017. Linear convergence and metric selection for Douglas-Rachford splitting and ADMM. *IEEE Trans Autom Contr*, 62(2):532-544. https://doi.org/10.1109/TAC.2016.2564160

Han JL, Shen SH, 2019. Distributed surface reconstruction from point cloud for city-scale scenes. Int Conf on 3D Vision, p.338-347. https://doi.org/10.1109/3DV.2019.00045

He BS, Yuan XM, 2012. On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method. *SIAM J Numer Anal*, 50(2):700-709. https://doi.org/10.1137/110836936

Huang X, Peng YX, Yuan MK, 2020. MHTN: modal-adversarial hybrid transfer network for cross-modal retrieval. *IEEE Trans Cybern*, 50(3):1047-1059. https://doi.org/10.1109/TCYB.2018.2879846

Konolige K, 2010. Sparse sparse bundle adjustment. Proc British Machine Vision Conf, p.102.1-102.11. https://doi.org/10.5244/C.24.102

Kushal A, Agarwal S, 2012. Visibility based preconditioning for bundle adjustment. IEEE Conf on Computer Vision and Pattern Recognition, p.1442-1449. https://doi.org/10.1109/CVPR.2012.6247832

Liu ZH, Diao CY, Xing W, et al., 2019. Critical parameter consensus for efficient distributed bundle adjustment. Proc 14th Int Joint Conf on Computer Vision, Imaging and Computer Graphics Theory and Applications, p.800-807. https://doi.org/10.5220/0007361108000807

Lourakis M, Argyros AA, 2005. Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment? IEEE Int Conf on Computer Vision, p.1526-1531. https://doi.org/10.1109/ICCV.2005.128

Lourakis M, Argyros AA, 2009. SBA: a software package for generic sparse bundle adjustment. *ACM Trans Math Softw*, 36(1):2. https://doi.org/10.1145/1486525.1486527

Lu LP, Zhang Y, Liu K, 2019. Block partitioning and merging for processing large-scale structure from motion problems in distributed manner. *IEEE Access*, 7:114400-114413. https://doi.org/10.1109/ACCESS.2019.2923667

Mostegel C, Rumpler M, Fraundorfer F, et al., 2016. Using self-contradiction to learn confidence measures in stereo vision. IEEE Conf on Computer Vision and Pattern Recognition, p.4067-4076. https://doi.org/10.1109/CVPR.2016.441

Mostegel C, Fraundorfer F, Bischof H, 2018. Prioritized multi-view stereo depth map generation using confidence prediction. *ISPRS J Photogr Remote Sens*, 143:167-180. https://doi.org/10.1016/j.isprsjprs.2018.03.022

Özyeşil O, Voroninski V, Basri R, et al., 2017. A survey of structure from motion. *Acta Numer*, 26:305-364. https://doi.org/10.1017/S096249291700006X

Peng YX, Zhai XH, Zhao YZ, et al., 2016. Semi-supervised cross-media feature learning with unified patch graph regularization. *IEEE Trans Circ Syst Video Technol*, 26(3):583-596. https://doi.org/10.1109/TCSVT.2015.2400779

Peng YX, Huang X, Zhao YZ, 2018. An overview of cross-media retrieval: concepts, methodologies, benchmarks, and challenges. *IEEE Trans Circ Syst Video Technol*, 28(9):2372-2385. https://doi.org/10.1109/TCSVT.2017.2705068

Rahaman H, Champion E, 2019. To 3D or not 3D: choosing a photogrammetry workflow for cultural heritage groups. *Heritage*, 2(3):1835-1851. https://doi.org/10.3390/heritage2030112

Schönberger JL, Frahm JM, 2016. Structure-from-motion revisited. IEEE Conf on Computer Vision and Pattern Recognition, p.4104-4113. https://doi.org/10.1109/CVPR.2016.445

Schönberger JL, Zheng EL, Frahm JM, et al., 2016. Pixelwise view selection for unstructured multi-view stereo. 14th European Conf on Computer Vision, p.501-518. https://doi.org/10.1007/978-3-319-46487-9_31

Shen XL, Dou Y, Mills S, et al., 2018. Distributed sparse bundle adjustment algorithm based on three-dimensional point partition and asynchronous communication. *Front Inform Technol Electron Eng*, 19(7):889-904. https://doi.org/10.1631/FITEE.1800173

Shi JB, Malik J, 2000. Normalized cuts and image segmentation. *IEEE Trans Patt Anal Mach Intell*, 22(8):888-905. https://doi.org/10.1109/34.868688

Simon I, Snavely N, Seitz SM, 2007. Scene summarization for online image collections. IEEE 11th Int Conf on Computer Vision, p.1-8. https://doi.org/10.1109/ICCV.2007.4408863

Snavely N, Seitz SM, Szeliski R, 2008. Skeletal graphs for efficient structure from motion. IEEE Conf on Computer Vision and Pattern Recognition, p.1-8. https://doi.org/10.1109/CVPR.2008.4587678

Triggs B, McLauchlan PF, Hartley RI, et al., 1999. Bundle adjustment—a modern synthesis. Int Workshop on Vision Algorithms, p.298-372. https://doi.org/10.1007/3-540-44480-7_21

Wu CC, Agarwal S, Curless B, et al., 2011. Multicore bundle adjustment. IEEE Conf on Computer Vision and Pattern Recognition, p.3057-3064. https://doi.org/10.1109/CVPR.2011.5995552

Zhai XH, Peng YX, Xiao JG, 2014. Learning cross-media joint representation with sparse and semisupervised regularization. *IEEE Trans Circ Syst Video Technol*, 24(6):965-978. https://doi.org/10.1109/TCSVT.2013.2276704

Zhang RL, Kwok JT, 2014. Asynchronous distributed ADMM for consensus optimization. Proc 31st Int Conf on Machine Learning, p.1701-1709.

Zhang RZ, Li SW, Fang T, et al., 2015. Joint camera clustering and surface segmentation for large-scale multi-view stereo. Proc IEEE Int Conf on Computer Vision, p.2084-2092. https://doi.org/10.1109/ICCV.2015.241

Zhang RZ, Zhu SY, Shen TW, et al., 2020. Distributed very large scale bundle adjustment by global camera consensus. *IEEE Trans Patt Anal Mach Intell*, 42(2):291-303. https://doi.org/10.1109/TPAMI.2018.2840719

Zhu SY, Fang T, Xiao JX, et al., 2014. Local readjustment for high-resolution 3D reconstruction. IEEE Conf on Computer Vision and Pattern Recognition, p.3938-3945. https://doi.org/10.1109/CVPR.2014.503

Zhu SY, Zhang RZ, Zhou L, et al., 2018. Very large-scale global SFM by distributed motion averaging. IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.4568-4577. https://doi.org/10.1109/CVPR.2018.00480