This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

Distributed Very Large Scale Bundle Adjustment by Global Camera Consensus

Runze Zhang
rzhangaj@cse.ust.hk

Siyu Zhu[1]
szhu@cse.ust.hk

Tian Fang[2]
fangtian@altizure.com

Long Quan
quan@cse.ust.hk

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology

Abstract

The increasing scale of Structure-from-Motion is fundamentally limited by the conventional optimization framework for the all-in-one global bundle adjustment. In this paper, we propose a distributed approach to coping with this global bundle adjustment for very large scale Structure-from-Motion computation. First, we derive the distributed formulation from the classical optimization algorithm ADMM, Alternating Direction Method of Multipliers, based on the global camera consensus. Then, we analyze the conditions under which the convergence of this distributed optimization would be guaranteed. In particular, we adopt over-relaxation and self-adaption schemes to improve the convergence rate. After that, we propose to split the large scale camera-point visibility graph in order to reduce the communication overheads of the distributed computing. The experiments on both public large scale SfM data-sets and our very large scale aerial photo sets demonstrate that the proposed distributed method clearly outperforms the state-of-the-art method in efficiency and accuracy.

1. Introduction

With the popularization of smartphones and unmanned aerial vehicles, larger collection of images with high quality and resolution are available, which give rise to dramatic increment of the scale of Structure-from-Motion. Some works try to tackle city level and even world level SfM, such as works in [21, 13, 1, 19, 16, 25, 26, 24]. But, these works do not globally optimize the results in the end, since the global optimization called bundle adjustment has too large scale cost to be completed by limited memory and time.

The bundle adjustment is the non-linear optimization to refine the camera parameters and 3D points in the end of Structure-from-Motion. The Levenberg-Marquardt algorithm [31] with Schur complement is the most common method to optimize the bundle adjustment formulation, which takes advantages of the sparsity in the multiple-view geometry problem. The work [3] tries to improve the efficiency of the large scale bundle adjustment problem by factorization and precondition method, and [7] proposes a method to group factors for the bundle adjustment problem. The preconditioner based on the camera visibility graph is also adopted in work [17, 20] to improve the efficiency of LM algorithm. The work [33] implements the optimization algorithm of bundle adjustment in parallel for acceleration by improving the CPU utilization. However, those methods are still single-core methods which cannot solve the problem with too large scale to be loaded into memory.

Some works try to tackle the large scale bundle adjustment problem by out-of-core algorithms [23, 22] and the distributed system [12] to break through the memory limitation of the single-core algorithm. The out-of-core method [23] splits the large scale bundle adjustment problem into several small problems, solves the small problems in parallel and merges them iteratively by the optimization of overlapping region of those small problems. However, for densely captured data-sets, the splitting will yield too large overlapping regions to be loaded into memory to do optimization. Besides, the I/O overhead induced by overlapping regions is also too high. In the following work [22], the author utilizes a hierarchical framework to split the large problem and merge small problems in each level by the smooth method [8]. Nevertheless, the hierarchical framework sacrifices the degree of parallelism and is difficult to implement in a distributed system. The smooth method to merge small problems cannot guarantee the bundle adjustment result is optimized.

The work in [12] proposes a consensus framework to deal with large scale bundle adjustment in distributed system. Instead of merging small problems by the optimization

[1]Siyu Zhu is the corresponding author.
[2]Tian Fang is with Shenzhen Zhuke Innovation Technology since 2017.

of overlapping regions of small problems, the consensus framework utilizes the proximal splitting method to formulate the bundle adjustment problem, in which the small problems are merged by averaging points in fact, decreasing the cost of merging. The merging process for the same parameters guarantees the consensus of points in different nodes. Thus, we call this method point consensus based distributed bundle adjustment. However, the consensus framework based on point consensus and splitting by cameras in [12] still has some problems in practice. Firstly, in each iteration, each node in the distributed system has to broadcast all overlapping points to the master node to complete the merging process, which is a huge overhead for large scale data-sets. Secondly, parameters of each camera are independent of parameters of other cameras. However, in practice, some cameras may share the same intrinsic parameters. Thirdly, the method by merging points converges a little slowly in very large scale data-sets and may converge in a local minimum early.

To solve above problems, referring to space division based methods [35, 34], we propose an improved consensus framework by camera consensus and splitting points. In summary, our contributions primarily are as follows:

1. We propose a general consensus framework regardless of the number of parameters of camera and take consideration of the common intrinsic parameters;
2. We analyze the conditions of convergence in detail and adopt the over-relaxation and self-adaption scheme to improve the convergence rate;
3. We propose a splitting method to minimize the overhead in the distributed system.

The rest of the paper is organized as follows. We will first review the bundle adjustment problem and the camera model in section 2.1. Then, we will introduce the ADMM algorithm and the consensus method based on the ADMM in section 2.2. In section 2.3, we derive the iteration Equations for camera consensus based distributed bundle adjustment from the ADMM algorithm. After that, we will analyze the conditions to guarantee the convergence in 3.1. The relaxation and self-adaption scheme are introduced in section 3.2 to improve the convergence rate. In section 4, we propose a scalable splitting method to minimize the overhead in the distributed system and describe the implementation detail. At last, we will show the experimental results in section 5, demonstrating that our method can solve the large scale bundle adjustment problem in a distributed system efficiently and accurately.

2. The Distributed Formulation

In this section, we will first introduce the bundle adjustment problem and the camera model. Then we will describe the global consensus problem solved by the Alternating Direction Method of Multipliers(ADMM) and then apply the solution to the bundle adjustment problem by camera consensus.

2.1. Bundle adjustment

Generally, camera parameters consist of extrinsic parameters and intrinsic parameters. Extrinsic parameters are independent for different cameras but intrinsic parameters may be shared by different cameras. Suppose we have M observed 3D points, N cameras sharing L different intrinsic parameters, note camera extrinsic parameter set $\mathcal{E} = \{\mathbf{e}_i \in \mathbb{R}^C | i = 1, ..., N\}$, camera intrinsic parameter set $\mathcal{D} = \{\mathbf{d}_l \in \mathbb{R}^I | l = 1, ..., L\}$, observed 3D point set $\mathcal{P} = \{\mathbf{p}_j \in \mathbb{R}^3 | j = 1, ..., M\}$ and detected observation set $\mathcal{Q} = \{\mathbf{q}_{ij} \in \mathbb{R}^2 | \mathbf{p}_j \in \mathcal{Q}_i\}$, where $\mathcal{Q}_i$ is the set of points viewed by the ith camera. Please note that $\mathbf{e}_i$ is not the extrinsic matrix of the ith camera, but the parameterization of extrinsic parameters. The bundle adjustment is a non-linear least square optimization problem to obtain the optimized cameras and 3D points with the following objective function:

$$f(\mathcal{E}, \mathcal{D}, \mathcal{P}) = \sum_{i=1}^{n} \sum_{\mathbf{p}_j \in \mathcal{Q}_i} ||\mathbf{\Pi}(\mathbf{e}_i, \mathbf{d}_{l_i}, \mathbf{p}_j) - \mathbf{q}_{ij}||^2, \quad (1)$$

where $\mathbf{d}_{l_i}$ is the intrinsic parameters of the ith camera and $\Pi(\mathbf{e}_i, \mathbf{d}_{l_i}, \mathbf{p}_j)$ computes the reprojection of the jth point in the ith camera, which is non-linear and non-convex. The computation $\Pi(\mathbf{e}_i, \mathbf{d}_{l_i}, \mathbf{p}_j)$ depends on the camera model and we use the pinhole camera model with radial distortion. Here, camera center $\mathbf{c}_i$ and parameterization of rotation $\mathbf{r}_{Ri}$ are used to express the extrinsic parameter of camera $\mathbf{e}_i$, namely $\mathbf{e}_i = (\mathbf{r}_{Ri}, \mathbf{c}_i)$. The parameterization of rotation will be discussed in section 3.1.

2.2. The global consensus based on ADMM

The ADMM algorithm [5] aims to solve the problem in the form:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{w} \end{array} \quad (2)$$

By the augmented Lagrangian multipliers method, we form

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{w}) + \frac{\rho}{2}||\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{w}||_2^2 \quad (3)$$

The ADMM algorithm consists of the iterations:

$$\mathbf{x}^{t+1} = \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}^t, \mathbf{y}^t) \quad (4)$$

$$\mathbf{z}^{t+1} = \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}^{t+1}, \mathbf{z}, \mathbf{y}^t) \quad (5)$$

$$\mathbf{y}^{t+1} = \mathbf{y}^t + \rho(\mathbf{A}\mathbf{x}^{t+1} + \mathbf{B}\mathbf{z}^{t+1} - \mathbf{w}) \quad (6)$$

In the implementation, $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$ are updated by an alternating fashion.

Now, we want to solve this global consensus problem:

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{n} f_i(\mathbf{x}_i) \\ &\text{subject to } \mathbf{x}_i = \mathbf{z}, i = 1, ..., n \end{aligned} \tag{7}$$

Using ADMM, we can derive the iteration expression as

$$\mathbf{x}_i^{t+1} = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \left(\mathbf{y}_i^t\right)^T (\mathbf{x}_i - \mathbf{z}^t) + \frac{\rho}{2} ||\mathbf{x}_i - \mathbf{z}^t||_2^2 \right) \tag{8}$$

$$\mathbf{z}^{t+1} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i^{t+1} \tag{9}$$

$$\mathbf{y}_i^{t+1} = \mathbf{y}_i^t + \rho(\mathbf{x}_i^{t+1} - \mathbf{z}^{t+1}), i = 1, ..., n \tag{10}$$

Substituting $\mathbf{y}_i = \rho \mathbf{u}_i$, we can express equations 8 and 10 as

$$\mathbf{x}_i^{t+1} = \arg\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\rho}{2} \left\| \mathbf{x}_i - (\mathbf{z}^t - \mathbf{u}_i^t) \right\|_2^2 \tag{11}$$

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t + \mathbf{x}_i^{t+1} - \mathbf{z}^{t+1}, \tag{12}$$

Defining the proximity operator $\mathbf{prox}_{f/\rho}$ of one function $f(x)$ with $\rho > 0$ as $\mathbf{prox}_{f/\rho}(a) = f(x) + \frac{\rho}{2} \|x - a\|_2^2$, we can express the right of Eqn. 11 as $\mathbf{prox}_{f_i/\rho}(\mathbf{z}^t - \mathbf{u}_i^t)$. The computation of equations 11 and 12 can be performed easily in a distributed system. Eqn. 9 actually averages the results of optimization results in different nodes.

2.3. Camera Consensus

Actually, the iterations of equations 8, 9, and 10 are indeed the Douglas-Rachford splitting method in [12]. By clarifying the derivation of the method, we can explain the stop criterion, over-relaxation and self-adaption scheme in section 3. The work [12] performs the consensus framework based on points. However, very large scale data-sets always contain so many points that the distributed system has huge overhead to broadcast those points. Besides, intuitively, the averaging of too many variables in Eqn. 9 leads to slow convergence rate. Hence, we try to reduce the overhead of broadcasting of $\mathbf{x}_i$ and $\mathbf{y}_i$ in the averaging for very large scale bundle adjustment problem by camera consensus instead of points.

In order to broadcast cameras instead points in [12], we split the whole bundle adjustment problem by points. Note each block as $\mathcal{B}_k = (\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k), k = 1, ..., K$, where $\mathcal{P}_k$ is the set of points in the block so that $\mathcal{P}_{k_1} \cap \mathcal{P}_{k_2} = \phi$ and $\bigcup \mathcal{P}_k = \mathcal{P}$. Since one camera may view points in different block, extrinsic parameters of one camera can appear in different block. We define $\mathbf{e}_i^k$ as the extrinsic parameter of the ith camera in block k and $\mathcal{E}_k = \{\mathbf{e}_i^k | \text{the } i\text{th camera views points in the } k\text{th block}\}$. Similar to extrinsic parameters, one intrinsic parameter may be shared

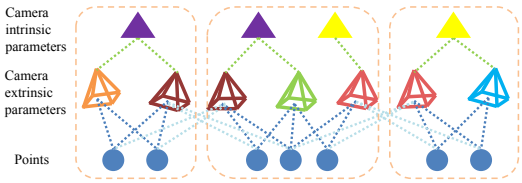


Figure 1: The camera extrinsic and intrinsic parameters with the same color are the component of the same parameter in different blocks. The links between cameras and points with the light color are the lost links due to the splitting.

by cameras in different blocks, so we define $\mathbf{d}_l^k$ as the lth intrinsic parameter in the kth block and $\mathcal{D}_k = \{\mathbf{d}_l^k | \text{the } l\text{th intrinsic parameter is shared by cameras in the } k\text{th block}\}$. Note n_i and m_l are the number of blocks where the ith extrinsic or the lth intrinsic parameter appear. The relationship of those parameters are shown in Fig. 1.

Then, we can modify the original bundle adjustment problem in Eqn. 1 as

$$\begin{aligned} &\text{minimize} \sum_{k=1}^{K} f(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k) \\ &\text{subject to } \mathbf{e}_i^k = \mathbf{e}_i, i = 1, ..., N, k = 1, ..., K \\ &\qquad\qquad \mathbf{d}_l^k = \mathbf{d}_l, l = 1, ..., L, k = 1, ..., K \end{aligned} \tag{13}$$

Using the ADMM algorithm, we can obtain the iterations for the above problem:

$$\left(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k\right)^{t+1} = \arg\min f\left(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k\right) + h\left(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k\right), \forall k \tag{14}$$

$$\mathbf{e}_i^{t+1} = \frac{\sum_{\mathcal{E}_k \ni \mathbf{e}_i^k} \left(\mathbf{e}_i^k\right)^{t+1}}{n_i}, \forall i, \mathbf{d}_l^{t+1} = \frac{\sum_{\mathcal{D}_k \ni \mathbf{d}_l^k} \left(\mathbf{d}_l^k\right)^{t+1}}{m_l}, \forall l, \tag{15}$$

$$\begin{aligned} \left(\tilde{\mathbf{e}}_i^k\right)^{t+1} &= \left(\tilde{\mathbf{e}}_i^k\right)^t + \left(\mathbf{e}_i^k\right)^{t+1} - \mathbf{e}_i^{t+1}, \forall i, k \\ \left(\tilde{\mathbf{d}}_l^k\right)^{t+1} &= \left(\tilde{\mathbf{d}}_l^k\right)^t + \left(\mathbf{d}_l^k\right)^{t+1} - \mathbf{d}_l^{t+1}, \forall l, k, \end{aligned} \tag{16}$$

where $f\left(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k\right)$ is the bundle adjustment objective function on variables in the kth block and

$$\begin{aligned} h\left(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k\right) &= \frac{1}{2} \sum\nolimits_{\mathbf{e}_i^k \in \mathcal{E}_k} \left\| \mathbf{e}_i^k - \mathbf{e}_i^t + \left(\tilde{\mathbf{e}}_i^k\right)^t \right\|_{\mathbf{\Sigma}_e}^2 \\ &+ \frac{1}{2} \sum\nolimits_{\mathbf{d}_l^k \in \mathcal{D}_k} \left\| \mathbf{d}_l^k - \mathbf{d}_l^t + \left(\tilde{\mathbf{d}}_l^k\right)^t \right\|_{\mathbf{\Sigma}_d}^2 + \frac{\rho_p}{2} \sum\nolimits_{\mathbf{p}_j \in \mathcal{P}_k} \left\| \mathbf{p}_j - \mathbf{p}_j^t \right\|_2^2 \end{aligned} \tag{17}$$

In Eqn. 17, unlike the method in [12] that the proximity operator does not work on the non-consensus items, the proximity operator here also works on points which do not attend the consensus in Eqn. 9. The reason will be introduced in section 3.1. For the term on camera parameters in Eqn. 17, we use $\|\cdot\|_{\mathbf{\Sigma}_e}$ and $\|\cdot\|_{\mathbf{\Sigma}_d}$ to replace the $l2$ norm, where $\mathbf{\Sigma}_e$ and $\mathbf{\Sigma}_d$ are a diagonal matrix whose diagonal elements are formed by penalty parameters $\boldsymbol{\rho}_e$ and $\boldsymbol{\rho}_d$. Since each parameter has its domain, each parameter should have their own penalty parameters according to its range. In above iterations, equations 14 and 16 can be implemented distributedly and camera parameters have to be broadcasted to complete the computing in Eqn. 15. The

Algorithm 1 Distributed Bundle Adjustment based on Camera Consensus

function DBACC($\mathcal{E}$, $\mathcal{P}$, $\mathcal{D}$, $\mathcal{Q}$, K)
  Initialize all $\widetilde{\mathbf{e}}_i^k$ and $\widetilde{\mathbf{d}}_l^k$ as 0
  Initialize all $\mathbf{e}_i^k$ and $\mathbf{d}_l^k$ as the corresponding initial values of $\mathbf{e}_i$ and $\mathbf{d}_l$
  while the criterion in Eqn. 19 is not satisfied **do**
    for each block $k \in [1, K]$ distributedly **do**
      for each extrinsic parameter $\mathbf{e}_i$ in block k in parallel **do**
        $\widetilde{\mathbf{e}}_i^k \leftarrow \widetilde{\mathbf{e}}_i^k + \mathbf{e}_i^k - \mathbf{e}_i$
      for each intrinsic parameter $\mathbf{d}_l$ in block k in parallel **do**
        $\widetilde{\mathbf{d}}_l^k \leftarrow \widetilde{\mathbf{d}}_l^k + \mathbf{d}_l^k - \mathbf{d}_l$
      $(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k) \leftarrow \arg\min f(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k) + h(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k)$
    Broadcast $\mathcal{E}_k$ and $\mathcal{D}_k$ of all nodes to the master node
    Average $\mathcal{E}_k$ and $\mathcal{D}_k$ in the master node by Eqn. 15 to get $\mathcal{E}$ and $\mathcal{D}$

distributed bundle adjustment algorithm based on camera consensus is concluded in Algorithm. 1.

Stopping criterion Referring to [6], we define the primal residual $\mathbf{r}^t$ and the dual residual $\mathbf{s}^t$ as

$$\begin{aligned}\left\|\mathbf{r}^t\right\|_2^2 &= \sum \left\|\left(\mathbf{e}_i^k\right)^t - \mathbf{e}_i^t\right\|_2^2 + \sum \left\|\left(\mathbf{d}_l^k\right)^t - \mathbf{d}_l^t\right\|_2^2 \\ \left\|\mathbf{s}^t\right\|_2^2 &= \sum \left\|\mathbf{e}_i^{t+1} - \mathbf{e}_i^t\right\|_{\mathbf{\Sigma}_e}^2 + \sum \left\|\mathbf{d}_l^{t+1} - \mathbf{d}_l^t\right\|_{\mathbf{\Sigma}_d}^2 \\ &\quad + \rho_p^2 \sum \left\|\mathbf{p}_j^{t+1} - \mathbf{p}_j^t\right\|_2^2\end{aligned} \tag{18}$$

Then, the stopping criterion is that $l2$ norms of the primal residual and dual residual are less than their thresholds

$$\left\|\mathbf{r}^t\right\|_2 < \epsilon^{pri}, \left\|\mathbf{s}^t\right\|_2 < \epsilon^{dual} \tag{19}$$

3. Convergence of Camera Consensus

In this section, we will first analyze the convergence conditions for the proposed algorithm. Then, we will discuss on some extensions to accelerate the algorithm.

3.1. Convergence conditions

ADMM algorithm requires that the function $f_i(x)$ in Eqn. 7 should be convex. However, the bundle adjustment objective function in Eqn. 1 is non-convex. Some works such as [14, 18, 30] analyze the proximal splitting method on non-convex problems and the work [12] proposes a statement for the convergence of Douglas-Rachford splitting applied to the bundle adjustment problem with point consensus without proof. In the following, we will analyze the convergence conditions of the Algorithm 1 and provide the provable statement for the convergence. The convergence proof is provided in the supplementary material.

In the convergence proof of ADMM on convex functions in [6], the proof depends on the convexity of $L_\rho(\mathbf{x}, \mathbf{z}^t, \mathbf{y}^t)$ in Eqn. 4 guaranteed by the convexity of $f(\mathbf{x})$, so that $\mathbf{x}^{t+1}$ minimizes it. If $f(\mathbf{x})$ is not convex, according to the propositions and theorems in [18], $f(\mathbf{x})$ must satisfy $\nabla f(\mathbf{x})$ is local Lipschitz-continuous with Lipschitz constant ρ_{min}, which guarantees $L_\rho(\mathbf{x}, \mathbf{z}^t, \mathbf{y}^t)$ be convex when $\rho > \rho_{min}$. Since the Lipschitz-continuity is defined on all variables, the proximity operator should work on all variables in Eqn. 17, though points do not participate in the consensus. Based on the Lipschitz-continuity requirement, we then check the objective function of bundle adjustment.

Note the reprojecton function in Eqn. 1 $\mathbf{\Pi}(\mathbf{e}, \mathbf{d}, \mathbf{p}) = \mathbf{f}_d(\mathbf{f}_p(\mathbf{e}, \mathbf{K}, \mathbf{p}), \mathbf{o})$, where $\mathbf{K}$ is the intrinsic matrix of the camera, $\mathbf{o}$ is the distortion parameter, $\mathbf{f}_p$ is the reprojection function of pinhole camera and $\mathbf{f}_d$ is the distortion function. Note the camera matrix as $[\mathbf{M}|\mathbf{m}] = \mathbf{K}\mathbf{R}[\mathbf{I}| - \mathbf{c}]$. Then $\mathbf{f}_p(\mathbf{e}, \mathbf{K}, \mathbf{p}) = [(\mathbf{M}_1^T\mathbf{p} + m_1)/(\mathbf{M}_3^T\mathbf{p} + m_3), (\mathbf{M}_2^T\mathbf{p} + m_2)/(\mathbf{M}_3^T\mathbf{p} + m_3)]$, where $\mathbf{M}_i$ is the ith row of $\mathbf{M}$ and m_i is the ith element of $\mathbf{m}$. Therefore, excluding the parameterization of rotation, $(\mathbf{M}_3^T\mathbf{p} + m_3)^{-a}, a > 0$ is the only part of the gradient of $\mathbf{f}_p$ which may break the lipschitz-continuity. To avoid the situation where $\mathbf{M}_3^T\mathbf{p} + m_3 \to 0$, we must assume $\mathbf{M}_3^T\mathbf{p} + m_3 \geq d_{min} > 0$, namely depths of any points in any cameras are larger than d_{min}, which is a reasonable assumption for SfM problem. Besides, we must also guarantee the gradient of distortion function $\partial\mathbf{f}_d/\partial\mathbf{o}$ be local Lipschitz-continuous, which is valid for radial distortion.

Let's consider the parameterization of rotation for the Lipschitz-continuity of the objective function in Eqn. 1. Note the parameterization of rotation $\mathbf{R}$ as $\mathbf{r}_R$. We must guarantee $\partial\mathbf{R}/\partial\mathbf{r}_R$ be Lipschitz-continuous. The commonly used parameterization of rotation includes quaternion and angle-axis($\theta\mathbf{v} \in \mathbb{R}^3$). The map of quaternion to rotation matrix involves the normalization of quaternion, so the gradient is unbounded when quaternion approaches zero. Besides, quaternion is a over-parameterization for rotation, so it leads to communication redundancy. The gradient of exponential map of angle-axis to rotation matrix $\mathbb{R}^3 \to SO(3)$ also contains a part of $\|\theta\mathbf{v}\|^{-a}, a > 0$, but it can be proved that the Jacobian tensor of exponential map approaches a constant when $\|\theta\mathbf{v}\| \to 0$. Therefore, the gradient of the exponential map can be proved as Lipschitz-continuous and angle-axis is the minimum description for rotation. Hence, angle-axis can be adopted as the parameterization of rotation in the proposed algorithm. Based on above analysis, we can conclude the following statement.

Theorem *With the bundle adjustment objective function 1, let* $\{\mathcal{E}^t, \mathcal{D}^t, \mathcal{P}^t\} \subset \mathbb{R}^{6N} \times \mathbb{R}^{IL} \times \mathbb{P}^{3M}$ *denote a sequence generated by Algorithm 1, where I is the dimension of intrinsic parameter and each* $\mathcal{E}^t = \{(\mathbf{r}_R^t, \mathbf{c}^t)\} \in \mathbb{R}^{3N} \times \mathbb{R}^{3N}$ *where* $\mathbf{r}_R$ *is the angle-axis presentation of rotation and* $\mathbf{c}$ *is the camera center. Suppose the gradient of distortion function is Lipschitz-continuous and the scene depth* d *is bounded from below by* $\mathbf{M}_{i3}^T\mathbf{p}_j + m_{i3} \geq d_{min} > 0$ *for all cameras and points, then, there exists a* $\boldsymbol{\rho}_{min} = (\boldsymbol{\rho}_e^{min}, \boldsymbol{\rho}_d^{min}, \rho_p^{min})$ *such that if each element in* $\boldsymbol{\rho}$ *of Algorithm 1 is larger than the corresponding one in* $\boldsymbol{\rho}_{min}$*, Algorithm 1 is guaranteed to converge to a local minimum of Eqn. 1.*

There are three differences of the statement from the one in [12]. Firstly, the rotation parameterization is required as angle-axis to satisfy the Lipschitz-continuity since the proximity operator works on camera parameters. Secondly, we consider distortions of the camera model and give the requirement of distortion function. Thirdly, we also consider the non-consensus term and its penalty to guarantee the Lipschitz-continuity more validly.

3.2. Improving convergence rate

In this section, we will introduce two extensions of ADMM algorithm to improve the convergence rate.

Self-adaption penalty It is obviously that if the penalty parameter $\boldsymbol{\rho}$ is too large, the algorithm will converge slowly. Otherwise, the algorithm will yield diverged results according to the analysis in section 3.1. According to the iterations in Eqn. 17, large penalty on violations of primal feasibility results in small primal residual $\mathbf{r}$. Conversely, small penalty leads to small dual residual according to the definition of dual residual $\mathbf{s}$ in Eqn. 18. Therefore, we adopt the scheme introduced in [15].

$$\rho_x^{t+1} = \begin{cases} \tau^{incr}\rho_x^t & \text{if} \|\mathbf{r}_x^t\|_2 > \mu_1 \|\mathbf{s}_x^t\|_2 \\ \rho_x^t / \tau^{decr} & \text{if} \|\mathbf{s}_x^t\|_2 > \mu_2 \|\mathbf{r}_x^t\|_2 \\ \rho_x^t & \text{otherwise} \end{cases} \tag{20}$$

where $_x$, $\mathbf{s}_x$ and $\mathbf{r}_x$ means the different components of $\boldsymbol{\rho}$, $\mathbf{s}$ and $\mathbf{r}$ corresponding to the parameters of different cameras and points. Since each parameter is independent, we adopt different penalty parameters for different parameters of different cameras and points. Since $\mathbf{u}_i = \mathbf{y}_i/\rho$ in Eqn. 12, when ρ multiplies τ, the corresponding $\widetilde{\mathbf{e}}_k^t$ and $\widetilde{\mathbf{d}}_k^t$ should be divided by τ.

Over-relaxation Over-relaxation scheme can be adopted in the ADMM iteration of Eqn. 6 according to the analysis in [10], where $\mathbf{A}\mathbf{x}^{t+1}$ can be replaced with $\alpha^k \mathbf{A}\mathbf{x}^{t+1} - (1-\alpha^t)(\mathbf{B}\mathbf{z}^t - \mathbf{w})$ in the iteration of Eqn. 6. Substituting the bundle adjustment configuration to the over-relaxation scheme, we can obtain the over-relaxed iteration of Eqn. 16:

$$\begin{aligned} \left(\widetilde{\mathbf{e}}_i^k\right)^{t+1} &= \left(\widetilde{\mathbf{e}}_i^k\right)^t + (1+\alpha^t)\left(\left(\mathbf{e}_i^k\right)^{t+1} - \mathbf{e}_i^{t+1}\right), \forall i,k \\ \left(\widetilde{\mathbf{d}}_l^k\right)^{t+1} &= \left(\widetilde{\mathbf{d}}_l^k\right)^t + (1+\alpha^t)\left(\left(\mathbf{d}_l^k\right)^{t+1} - \mathbf{d}_l^{t+1}\right), \forall l,k, \end{aligned} \tag{21}$$

where $\alpha^t \in (0,1)$ and $\lim_{t\to\infty} \sum \alpha^t (1-\alpha^t) = 0$. The experiments in [9, 11] suggest that $\alpha^t \in [0.5, 0.8]$ should improve the convergence rate.

4. The Distributed Implementation

In this section, we will describe the implementation details to reduce the overhead further more and improve the convergence rate.

4.1. Block splitting

Few previous works on the parallel or distributed bundle adjustment discuss how to split blocks. [23] adopts graph cut to get a partition minimizing the edges that span the visibility graph, so that the overlapped region including cameras and points is minimized. Our method only broadcast all the cameras in different nodes to the master node to compute Eqn. 15. Suppose parameters of each camera are independent, the total overhead of one iteration in our distributed method is proportional to

$$\sum\nolimits_{i=1}^{K} |\mathcal{E}_k| = \sum\nolimits_{i=1}^{N} n_i, \tag{22}$$

However, it is NP-hard to minimize the above overhead. Referring to [23], we transform the problem to a graph cut problem on the camera-point visibility graph.

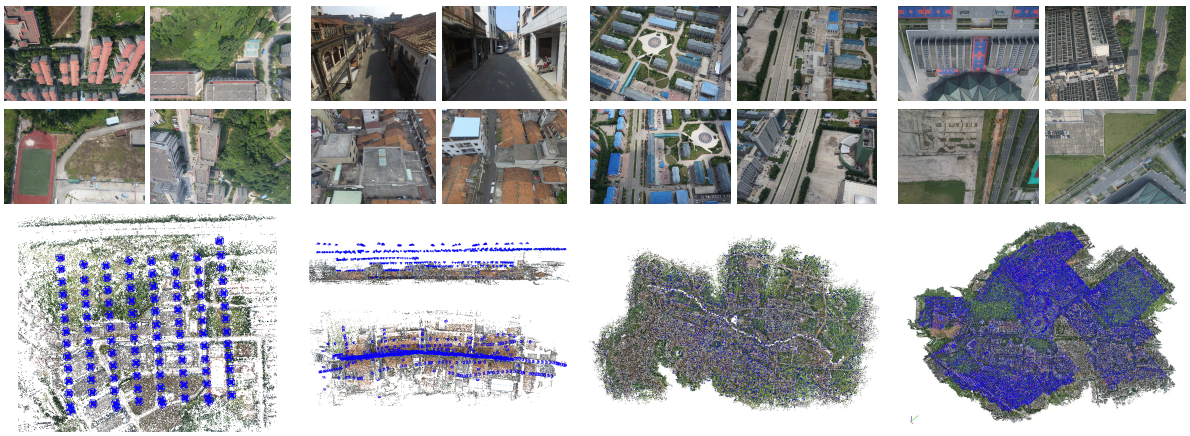
If the ith camera appears in n_i blocks, the number of links between the camera and its visible points which will be cut(the link with the light color shown in Fig.1) is at least $n_i - 1$. Hence, $\sum_{i=1}^{N} n_i \leq \sum_{i=1}^{N} E_i + n$, where E_i is the number of cut edges on the ith camera. Therefore, we can minimize the upper bound of Eqn. 22 by graph-cut in the visibility graph between cameras and points to obtain the sub-optimization of Eqn. 22. Meanwhile, it is better that each block has unbiased number of cameras and points to balance the load of each node. Therefore, Normalized-Cut [27] is a proper algorithm to implement the graph cut. After the graph cut, we collect points and cameras viewing those points in different blocks.

However, for very large scale data-sets with high capturing density, cameras, points and edges in visibility graph are so many that the graph cut algorithm cannot work on one machine. To tackle this problem, we first divide the points just by KD-Tree [4] into the first-level blocks which can be split by graph cut and also collect all the cameras viewing those points in each first-level block to construct the sub-graphs. After that, we perform graph cut on each sub-graph to get the blocks which will be used in the distributed bundle adjustment algorithms.

4.2. Parameter setting

Although we adopt the self-adaption scheme for the penalty parameters $\boldsymbol{\rho}$ in section 3.2, we still need set a proper initial value for the penalty parameters.

The analysis in section 3.1 shows that the penalty parameters should be larger than the Lipschitz constant of the objective function gradient. However, too large penalty parameters may lead to low convergence rate. The Lipschitz constant is difficult to estimate, so we set the initial penalty parameters intuitively according to the estimated range of parameters and the ratio of observations, cameras and points. Since the penalty parameters should balance the errors of $f(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k)$ and $h(\mathcal{E}_k, \mathcal{D}_k, \mathcal{P}_k)$ in Eqn. 14, the



Buildings Street Town City

Figure 2: Selected images and screenshots of the Structure-from-Motion results of **Buildings**, **Street**, **Town** and **City**. The first row is the selected images and the second row is the screenshots of SfM results, where the blue points are cameras. For **Street**, the top view and side view of SfM results are provided to show the data-set contains both aerial view and street view photos.

initial penalty parameters are set proportional to the ratio of observations, cameras and points as $\rho_x = \alpha_x|\mathcal{Q}|/N$ and $\rho_p = \alpha_p|\mathcal{Q}|/M$. ρ_x and α_x correspond to different kinds of camera parameters.

To unify the scale of input data, we first normalize the input SfM results so that the coordinates of camera centers are in $[-1, 1]^3$. Then we set the penalty coefficient α_c on camera centers and α_p on points as 10^5. The angle-axis $\mathbf{r}_R$ of rotation are periodic and $\|\mathbf{r}_R - \mathbf{r}_R^0\| < 2\pi$, so it has the similar range to the normalized camera centers. Thus, the penalty coefficient α_r is also set as 10^5 for all cameras. The intrinsic parameters have their own ranges. In our experiment, we will optimize focal lengths, principle points and radial distortions simultaneously. Focal lengths and principle points are in the same order of magnitude as the image resolution and we set the penalty coefficient α_f and α_{uv} as 10^{-3}. Radial distortions are always less than 10^{-2}, we set the the penalty coefficient α_d on them as 10^4. The convergence threshold ϵ^{pri} and ϵ^{dual} in Eqn. 19 are set according to the initial penalty parameters. ϵ^{pri} is set as $10^{-5} \times N$ and ϵ^{dual} is set as $10^{-5} \times (2N\rho_r^0 + M\rho_p^0 + L(\rho_d^0 + 3\rho_f^0))$.

In the self-adaption scheme in section 3.2, we need to set four parameters μ_1, μ_2, τ^{incr} and τ^{decr}. Since the dual residuals are proportion with the penalty parameters and primal residuals are independent of the penalty parameters, generally, with large penalty parameters, the dual residuals are always much larger than the primal residuals in the iterations in Eqn. 20. Thus, we set μ according to the initial penalty parameters as $\mu_{1x} = 10/\rho_x^0$ and $\mu_{2x} = 10\rho_x^0$. τ^{incr} and τ^{decr} are both set as 2. By this way, the penalty parameters fluctuate on enough large values to guarantee the convergence. The over-relaxation coefficient in our experiments is set as 0.5.

5. Experimental Results

The proposed algorithm is implemented in C++ and run on PCs with Intel(R) Core(TM) i7-4770K 3.50GHz processors with 8 threads and 32GB memory. The communication speed is about 10MB/s among different nodes. We assign each thread one block and the number of used computers depends on the number of blocks according to the data scale. The method is a meta-algorithm independent of the specific optimization method for the iteration in Eqn. 14. Ceres Solver [2] with preconditioners in [20] is used as the optimization tool for each block's optimization in our experiments. We test our algorithm on the public online data-sets and our aerial photo data-sets. We choose the public online data-sets with number of images larger than 900, among which **Roman Forum**, **Piccadilly**, **Trafalgar** [32] are initialized by Bundler [28] and **Ladybug**, **Venice**, **Final 961**, **Final 13682** [3] are initialized by the methods in [1], [29] and Bundler [28]. Our aerial photo data-sets include **Buildings**, **Street**, **Town** and **City** with high resolution 6000×4000 captured by DJI Phantom 4. **Buildings**, **Town** and **City** are captured in the aerial view and the cameras look towards the ground. **Street** is captured in both aerial and street view and some of cameras look forward in the street view. Selected images and screenshots of the Structure-from-Motion results of those four aerial photo data-sets are shown in Fig. 2. These 4 aerial photo data-sets are also initialized by the methods in [1], [29] and Bundler [28]. The number of intrinsic parameters shared by all cameras are one in **Buildings**, five in **Town** and 45 in **City**. Intrinsic parameters of cameras in other data-sets are independent. Since overlapping regions of the method of [23] for most of our data-sets account for at least half of the whole regions, it is nearly equivalent to do bundle adjustment in one machine. Therefore, we mainly compare our algorithm with the point consensus

Dataset	N	M	$\|\mathcal{Q}\|$	K	Error	KDTree		NCut				PC			
						N_c	B(MB)	N_c	B(MB)	T(s)	r_o(%)	N_p	B(MB)	T(s)	r_o(%)
Buildings	510	260k	1.40M	32	1.21	5.12k	0.236	**4.57k**	**0.211**	**1.41**	**2.94**	190k	4.35	2.22	38.7
Final 961	961	187k	1.69M	32	0.760	26.4k	2.42	**25.4k**	**2.33**	**5.22**	**10.4**	155k	3.56	5.30	13.1
Roman Forum	1084	158k	1.12M	32	0.531	12.0k	1.10	**9.66k**	**0.884**	**7.16**	**2.89**	92.3k	2.11	7.18	4.61
Street	1130	347k	2.00M	64	1.04	22.6k	2.07	**13.6k**	**1.25**	**2.66**	**13.5**	156k	3.57	3.00	23.5
Ladybug	1723	157k	679k	64	0.739	19.9k	1.82	**10.2k**	**0.93**	**4.07**	**6.66**	66.9k	1.53	4.19	8.10
Venice	1778	994k	5.00M	64	1.11	21.6k	1.98	**20.6k**	**1.89**	**4.86**	**8.44**	352k	8.05	6.18	28.2
Piccadilly	2152	136k	9.20M	64	0.613	30.9k	2.83	**23.5k**	**2.15**	**8.89**	**5.61**	156k	3.56	9.06	9.63
Trafalgar	5288	214k	1.82M	128	-	92.7k	8.49	**53.4k**	**4.89**	**19.3**	**6.05**	354k	8.10	20.3	10.4
Final 13682	13682	4.46M	29.0M	256	-	511k	46.8	**282k**	**25.8**	**40.2**	**17.1**	4.65M	106	54.0	39.8
Town	36428	27.8M	3512M	1024	-	412k	18.9	**365k**	**17.3**	**22.8**	**12.2**	15.0M	342	105.6	80.0
City	138193	100.2M	10088M	2048	-	910k	45.9	**693k**	**35.9**	**73.1**	**10.5**	33.2M	760	167.4	70.9

Table 1: This table shows the data-scale of each data-set, containing the number of cameras(N), the number of points(M) and the number of observation($|\mathcal{Q}|$). K is the number of blocks, which are decided according to the number of cameras of each data-set. "Error" is the average reprojection error of each observation with unit pixel after the data is optimized by one single machine method implemented by Ceres Solver [2] with preconditioners in [20]. KDTree and NCut are different methods to split blocks for our algorithm. KDTree just splits blocks so that each block has the same number of points and NCut is the splitting algorithm described in section 4.1. PC is the point consensus method in [12]. Since [12] does not discuss the splitting method, we use the graph cut method similar to the method in section 4.1. N_c is the number of cameras to be broadcasted in one iteration in our algorithm and N_p is the number of points to be broadcasted in one iteration in method PC. B is the real size of data to be broadcasted in one iteration. T is the total time including computing and communication in one iteration. r_o is the ratio between the communication time and total time.

based method in [12]. To share intrinsic parameters in **Buildings**, **Town** and **City** in the experiment of the point consensus based method, we modify the method to average intrinsic parameters. Table 1 shows the numbers of cameras, points, observation, blocks of all data-sets and the optimized results by the traditional single machine bundle adjustment method [20] for above data-sets which can be optimized in one single machine.

In the following experiments, we first test how our splitting algorithm in section 4.1 optimizes the overhead and compare the overhead of our algorithm with the point consensus method [12]. Then, we compare the convergence of the two method. In the comparison, we will also show how the over-relaxation and self-adaption scheme affect the convergence rate.

5.1. Overhead

Table 1 shows the size of data to be broadcasted in one iteration, time used in one iteration and the ratio for which communication among different nodes accounts of total time in one iteration. Our method need broadcast all the camera parameters and each camera contains 6 extrinsic parameters and 6 camera intrinsic parameters according to our experiment configuration.

Firstly, we compare the splitting algorithm in section 4.1 with the method that just splits blocks equally by KD-Tree so that each block contains the same number of points. Table 1 demonstrates the splitting algorithm indeed reduces the total number of cameras to be broadcasted, especially for the densely captured data-set **Ladybug**, **Trafalgar**, **Final 13682** and **City**, for which the proposed method reduces nearly half of the data to be broadcasted. Then, we compare the proposed method with the point consensus method in [12]. Since the splitting method is not discussed in [12], in the comparison, we modify the method in

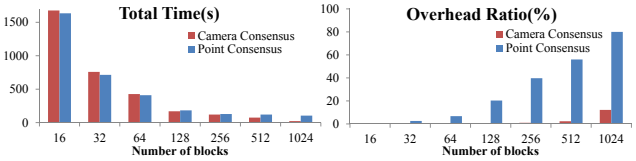


Figure 3: Left is the total time of each iteration with different numbers of blocks. Right is the ratio of communication time and total time. The experiments are performed on **Town**.

section 4.1 so that we collect the cameras and all the points viewed by those cameras after graph cut to minimize the points to be broadcasted in the point consensus method. Table 1 demonstrates that the cameras to be broadcasted in our method are less than the points to be broadcasted in the point consensus method by one or two orders of magnitude. Here, since we consider the optimization of camera intrinsic parameters in the bundle adjustment, each camera has to transfer 6 or 12 parameters, more than the parameters each point transfers. However, since the number of cameras to be broadcasted is much less than points, the total size of data to be broadcasted in our method is still less than the point consensus method. The more densely the data captured, the more data size reduced in broadcasting. On data-sets **Venice**, **Final 13682**, **Town** and **City**, the proposed method reduces the data to be broadcasted by 4 to 10 times in table 1.

Since our method splits blocks by points instead of cameras, the blocks in our methods have more cameras than points, while the blocks of points consensus method have more points than cameras. The number of cameras have more influence on the time used in local bundle adjustment of each block. Therefore, the local bundle adjustment of blocks in our method is a little slower than the compared method each iteration. However, with the increment of data scale and nodes used in the distributed system, the

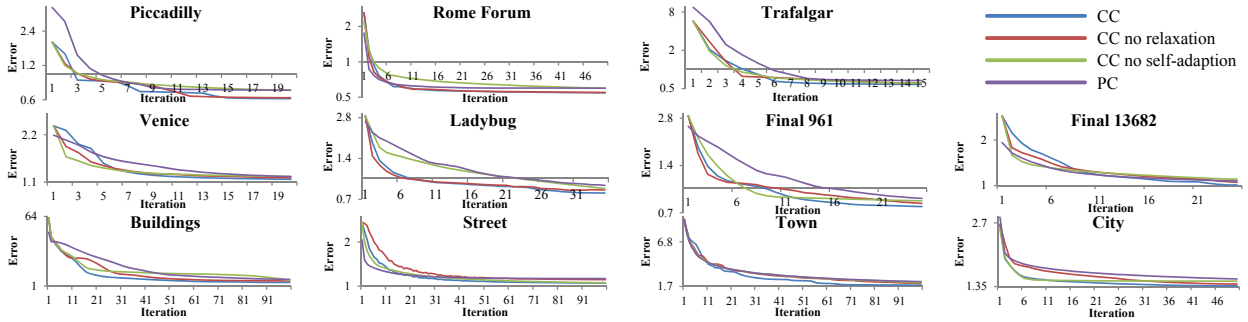


Figure 4: The convergence curves of each method for each data-set. The vertical axis of average reprojection errors uses the logarithmic scale to stand out the changes in the last iterations.

communication overhead accounts for more ratio among total time. Shown as Fig. 3, with the increment of number of used blocks, the total time reduces each iteration and the time cost by communication increases. The communication time in our method increases much slower than the compared method. For **Town** in Fig. 3, the total cost time each iteration in our method is less than the compared method after the number of blocks is larger than 128. On large scale data-sets **Final 13682**, **Town** and **City**, our method saves 20%-80% time, while the point consensus method spends more than half of time on communication. Therefore, our method has higher scalability when we use more blocks and deal with larger scale data-sets.

5.2. Convergence

Fig. 4 shows the convergence curves of each method for each data-set and table 2 provides the average reprojection error of each observation and the number of iteration when the algorithms satisfy the stop criterion in Eqn. 19.

We first test the over-relaxation and self-adaption scheme for our algorithm. The curve in Fig. 4 shows that the over-relaxation facilitates the convergence after several iterations. To test the effect of the self-adaption scheme, we adopt the strategy in [12] for the algorithm without self-adaption as comparison. In that strategy, the penalty parameters increase 0.01 times in each iteration. Table 2 shows that the algorithm without self-adaption converges slowly or converges fast in a larger error. Fig. 4 shows that the algorithm with self-adaption always has a non-smooth curve. The reason is the penalty parameters decrease when the primary residual is much smaller than the dual residual and the curve will converge faster after the penalty is set smaller.

To compare with the point consensus method in [12], we modify the penalty parameter setting in [12] since we normalize all the data-sets. The initial penalty parameter on each point position is $10^5 \times |\mathcal{Q}|/M$ same as the one in our method. We also adopt the over-relaxation and the self-adaption scheme on the point consensus method. Table 2 shows the point consensus method always converges in little larger errors, though it converges faster for some

Dataset	Convergence							
	CC		CC_{nr}		CC_{na}		PC	
	N_{it}	Error	N_{it}	Error	N_{it}	Error	N_{it}	Error
Buildings	**64**	**1.23**	72	1.35	100	1.39	87	1.43
Final 961	22	**0.763**	30	0.801	**14**	0.820	25	0.864
Roman Forum	**22**	**0.536**	25	0.542	50	0.590	23	0.597
Street	85	**1.04**	87	1.10	100	1.05	**76**	1.08
Ladybug	**31**	**0.745**	32	0.801	34	0.810	32	0.837
Venice	16	**1.13**	17	1.15	**12**	1.18	20	1.17
Piccadilly	17	**0.614**	13	0.625	19	0.686	**11**	0.688
Trafalgar	**10**	**0.580**	15	0.601	12	0.597	**10**	0.619
Final 13682	24	**1.01**	30	1.04	23	1.08	**22**	1.07
Town	**83**	**1.76**	96	1.87	98	1.89	96	1.91
City	41	**1.36**	59	1.38	**34**	1.43	61	1.41

Table 2: This table shows the number of iterations when each method satisfies the stop criterion for each data-set and average reprojection errors of each observation when they converge. N_{it} is the number of iterations and "Error" is the average reprojection error of each observation with unit pixel. CC is our algorithm with over-relaxation and self-adaption scheme, which means camera consensus. CC_{nr} is our algorithm without the over-relaxation scheme and CC_{na} is the one without the self-adaption scheme. PC is the method in [12], meaning point consensus.

data-sets. Intuitively, single points influence weaker than cameras in the bundle adjustment problem and the more parameters to be averaged, the slower the ADMM algorithm converges. Thus, camera consensus method outperforms the point consensus one on convergence rate.

6. Conclusion

In this paper, we propose a distributed bundle adjustment algorithm based on camera consensus for very large scale data-sets. Our key contribution is that we distribute points in different nodes of the distributed system and broadcast cameras for consensus. The camera consensus reduces the size of data to be broadcasted in each iteration and thus saves much overhead in the distributed system. Besides, we adopt the over-relaxation and self-adaption scheme to improve the convergence rate. In the end, the experiments demonstrate our camera consensus method outperforms the state-of-the-art method in efficiency and accuracy.

Acknowledgement. This work is supported by Hong Kong RGC 16208614, T22-603/15N, Hong Kong ITC P-SKL12EG02, and China 973 program, 2012CB316300.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] S. Agarwal, K. Mierle, and Others. Ceres solver. `http://ceres-solver.org`.

[3] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *European Conference on Computer Vision*, pages 29–42. Springer, 2010.

[4] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[5] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[6] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[7] L. Carlone, P. Fernandez Alcantarilla, H.-P. Chiu, Z. Kira, and F. Dellaert. Mining structure fragments for smart bundle adjustment. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014.

[8] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[9] J. Eckstein. Parallel alternating direction multiplier decomposition of convex programs. *Journal of Optimization Theory and Applications*, 80(1):39–62, 1994.

[10] J. Eckstein and D. P. Bertsekas. On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[11] J. Eckstein and M. C. Ferris. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, 10(2):218–235, 1998.

[12] A. Eriksson, J. Bastian, T.-J. Chin, and M. Isaksson. A consensus-based framework for distributed bundle adjustment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[13] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, et al. Building rome on a cloudless day. In *European Conference on Computer Vision*, pages 368–381. Springer, 2010.

[14] M. Fukushima and H. Mine. A generalized proximal point algorithm for certain non-convex minimization problems. *International Journal of Systems Science*, 12(8):989–1000, 1981.

[15] B. He, H. Yang, and S. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications*, 106(2):337–356, 2000.

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

[17] Y.-D. Jian, D. C. Balcan, and F. Dellaert. Generalized subgraph preconditioners for large-scale bundle adjustment. In *Outdoor and Large-Scale Real-World Scene Analysis*, pages 131–150. Springer, 2012.

[18] A. Kaplan and R. Tichatschke. Proximal point methods and nonconvex optimization. *Journal of global Optimization*, 13(4):389–406, 1998.

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

[20] A. Kushal and S. Agarwal. Visibility based preconditioning for bundle adjustment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012.

[21] M. Lhuillier and L. Quan. A quasi-dense approach to surface reconstruction from uncalibrated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(3):418–433, 2005.

[22] K. Ni and F. Dellaert. Hypersfm. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 144–151. IEEE, 2012.

[23] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3d reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2007.

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

[25] J. L. Schonberger, F. Radenovic, O. Chum, and J.-M. Frahm. From single image query to detailed 3d reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[26] T. Shen, S. Zhu, T. Fang, R. Zhang, and L. Quan. Graph-based consistent matching for structure-from-motion. In *European Conference on Computer Vision*, pages 139–155. Springer, 2016.

[27] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[28] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics*, volume 25, pages 835–846. ACM, 2006.

[29] N. Snavely, S. M. Seitz, and R. Szeliski. Skeletal graphs for efficient structure from motion. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008.

[30] S. Sra. Scalable nonconvex inexact proximal splitting. In *Advances in Neural Information Processing Systems*, pages 530–538, 2012.

[31] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustmentła modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[32] K. Wilson and N. Snavely. Robust global translations with 1dsfm. In *European Conference on Computer Vision*, pages 61–75. Springer, 2014.

[33] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2011.

[34] R. Zhang, S. Li, T. Fang, S. Zhu, and L. Quan. Joint camera clustering and surface segmentation for large-scale multi-view stereo. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[35] S. Zhu, T. Fang, J. Xiao, and L. Quan. Local readjustment for high-resolution 3d reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.