SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

June 2025, Vol. 68, Iss. 6, 162103:1–162103:15
https://doi.org/10.1007/s11432-023-4281-3

Alternating optimization for bundle adjustment with closed form solutions

Chengzhe MENG, Yiwen JIANG & Weiwei XU*

State Key Lab of Computer-Aided Design & Computer Graphics, Zhejiang University, Hangzhou 310058, China

Received 11 December 2023/Revised 18 March 2024/Accepted 20 May 2024/Published online 15 May 2025

Abstract This study proposes a novel alternating optimization algorithm for bundle adjustment, a critical process in structure from motion methods. We introduce the inverse depth of each three-dimensional (3D) point as an augmented independent variable and develop a low-order polynomial error metric. Theoretically, the error can be adjusted to align closely with the re-projection error since it essentially acts as a re-weighted re-projection error. We decouple the bundle adjustment problem by breaking it down into three separate tasks: estimating the camera's pose, determining the 3D structure, and optionally estimating the camera's intrinsic parameter. Each task can be handled independently, either by camera or by point, allowing for easy distribution of computation. Camera pose estimation is a case of the absolute orientation problem, which can be globally solved in closed form. A linearization scheme is proposed for 3D point estimation, which allows the computation of the update direction and line search in closed form. Our algorithm proves to be efficient, reliable, and accurate, as demonstrated by experimental results that confirm its superiority over recent alternatives.

Keywords bundle adjustment, alternating optimization, 3D reconstruction, vision localization

Citation Meng C Z, Jiang Y W, Xu W W. Alternating optimization for bundle adjustment with closed form solutions. Sci China Inf Sci, 2025, 68(6): 162103, https://doi.org/10.1007/s11432-023-4281-3

1 Introduction

Bundle adjustment (BA) is a crucial component in structure from motion (SfM) and simultaneous localization and mapping (SLAM) [1–3] applications. It is formulated as a least-squares optimization problem concerning the re-projection or photometric error of a set of matched 2D features [4–7].

Traditionally, methods like the Gauss-Newton algorithm and its variants are employed to tackle the BA problem. A significant challenge is the time required to solve the normal equation system in each iteration. To expedite this process, techniques such as the Schur complement [8] or preconditioned conjugate gradient (PCG) [9] methods are often utilized.

In large-scale BA scenarios, such as city-scale problems where data is too extensive for a single machine, alternating optimization (AO) methods are explored. These methods distribute the computation by dividing variables into subproblems concerning either cameras or 3D points. By alternately updating these subproblems, they aim to reduce the re-projection error. Some approaches, like proximal splitting and the alternating direction method of multipliers (ADMMs) [10], first partition 3D points or cameras into disjoint sets and then solve each in turn. However, these methods tend to achieve consensus through variable aggregation and multiplier adjustment, which can lead to a slower convergence rate [10].

This study introduces a new algorithm for BA that is fast enough for real-time visual SLAM systems and large-scale BA problems. Our algorithm also alternately refines camera poses, 3D points, and the camera's intrinsic parameters, leveraging key technical innovations to achieve fast convergence speed.

• We introduce the inverse depth of each 3D point as an augmented variable to modify the re-projection error, significantly reducing non-linearity in the objective functions. This adjustment allows all subproblems to be optimally refined with closed form solutions.

• We propose a linearization scheme to derive a closed form solution for refining 3D points. By treating changes in inverse depths and 3D positions as variables and discarding second-order terms, we achieve a linear objective function. This step enables the simultaneous solution of update directions for inverse

* Corresponding author (email: xww@cad.zju.edu.cn)

depth and 3D positions. The original objective function is a fourth-order polynomial, and line search also has a closed form solution.

The augmented inverse depth variable in our formulation aims to replace division with multiplication, significantly reducing the nonlinearity of the objective function. While similar parameters are used in photometric BA, where inverse depth parameterizes the 3D structure [6], here it is applied differently. Our modified error measurement, involving inverse depth, is effectively a re-weighted re-projection error. This allows us to seamlessly adjust the modified error back to the original error using the cosine of the angle between the optical axis and the observation ray connecting the optical center and the 2D feature point. Our objective function is not only easier to optimize compared to the re-projection error, but also achieves the same accuracy.

Our framework permits independent update of each 3D point with fixed camera poses and vice versa, facilitating straightforward distributed implementation. In our current setup, we partition captured images into disjoint sets, requiring only transmission of data related to overlapping 3D points visible across multiple sets. However, the transmitted information is utilized to solve the 3D-point updating subproblem, which is different from the parameter or consensus steps in [10, 11].

Our method introduces a stable, efficient, and accurate solver for BA, which is also easy to implement. We evaluated our method against leading methods across various problem sizes, ranging from small real-time SLAM to city-scale SfM tasks. The experimental results confirm the advantages of our method, achieving the highest accuracy on datasets with ground truths among state-of-the-art methods. Furthermore, it can solve very large BA problems on standard computers with a maximum of 8 GB of memory.

The rest of this paper is organized as follows. Section 2 summarizes related work, Section 3 presents the mathematical framework of our method, Section 4 compares our method with state-of-the-art approaches, and Section 5 concludes the paper.

2 Related work

2.1 BA

BA involves solving a vision model that accounts for a set of matched 2D features. This model includes a set of 3D points, camera poses, and optionally the camera's intrinsic parameters, all of which are considered unknown variables. BA is typically defined as an optimization problem $\min J(\boldsymbol{\theta}) = \sum_i^n \|e_i^r(\boldsymbol{\theta})\|^2$, where e_i^r represents the re-projection error of the ith observation and $\boldsymbol{\theta}$ the vector of unknowns.

Existing BA implementations belong to the family of Gauss-Newton methods [4]. A normal equation system $\boldsymbol{H}\Delta\boldsymbol{\theta} = \boldsymbol{g}$ is solved in each iteration. The number of unknowns is huge as it contains every 3D point and camera pose. Therefore, the matrix $\boldsymbol{H}$ becomes considerably large, making the computational process time-consuming as significant resources are devoted to solving the normal equation system.

To reduce computational complexity, the sparsity of the BA problem can be explored [4, 8]. This sparsity arises because each term of the objective function involves only a single camera and a single 3D point. This implies that the normal equation system can be represented in a sparse form [12]

$$\begin{bmatrix} \boldsymbol{B} & \boldsymbol{E} \\ \boldsymbol{E}^{\mathrm{T}} & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{y} \\ \Delta\boldsymbol{z} \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}_1 \\ \boldsymbol{g}_2 \end{bmatrix},$$

where $\boldsymbol{B}$ and $\boldsymbol{C}$ represent block diagonal matrices, the inverses of which are easy to compute. Then, one can eliminate $\Delta\boldsymbol{z}$ to get $[\boldsymbol{B} - \boldsymbol{E}\boldsymbol{C}^{-1}\boldsymbol{E}^{\mathrm{T}}]\Delta\boldsymbol{y} = \boldsymbol{g}_1 - \boldsymbol{E}\boldsymbol{C}^{-1}\boldsymbol{g}_2$, forming a smaller equation system, named the reduced camera system (RCS). This technology is also known as the Schur complement trick.

Some variants can further accelerate BA computation. One such method, RootBA [13], uses a memory-efficient nullspace projection of the Jacobian, offering a more sparse-efficient alternative to the Schur complement. This technique allows for solving large-scale BA problems using only single-precision floating-point numbers. Another approach, PowerBA [14], speeds up computation by expanding the inverse Schur complement using a power series. Furthermore, STBA [15], uses a stochastic algorithm to decompose the RCS during Levenberg-Marquardt (LM) iterations, thus accelerating computation.

However, the complexity of RCS remains as high as $O(n^2)$. Therefore, more efficient methods are necessary for handling large-scale BA problems, such as city-scale reconstruction [10,16,17]. Most existing methods often rely on PCGs or the ADMMs. Details of the methods are discussed below.

2.2 Inexact method

Researchers in [9, 12, 18, 19] have explored the use of inexact PCGs to solve the normal equation system. The CG method solves a system of linear equations using non-expensive matrix-vector production, which eliminates the need to explicitly construct the matrix in memory [9, 12]. Thus, time and space complexity are reduced.

Multicore [9] is likely the most efficient PCG-based method. It exploits the sparsity of both the Jacobians and the preconditioners, thereby breaking down the computation into a series of simple matrix-vector products, which is parallelized by dividing the computation into camera-, point-, and observation-wise components.

Inexact methods, while efficient, may compromise accuracy. While the CG method requires a large number of iterations to achieve satisfactory accuracy, most PCG-based BA methods employ a fixed, limited number of iterations. Consequently, exact BA algorithms are more common in scenarios where high accuracy is paramount, with most commercial 3D vision software opting for exact BA libraries [20].

2.3 ADMM based method

To solve BA problems in a distributed way, ADMM-based methods are proposed in [10, 21]. These methods distribute the objective function and its variables across multiple machines, solving problems iteratively in an alternating manner. In each iteration, individual machines independently optimize their local variables. The newly computed local variables are sent to a master machine to compute the average, which in turn broadcast back to the local machines. This process ensures the consensus of split variables by using an additional penalty term in the objective function. Notable examples of these methods includes RPBA [22], DPBA [23] and STBA [15].

However, the convergence rate of ADMM-based methods can be slow. This is because the ADMM method is inherently designed for convex problems, which do not directly align with the nature of the BA problem. To adapt ADMM for BA problems, approach [10] suggests using sufficiently large penalty parameters, which unfortunately can further slow down the convergence rate.

2.4 Error measurement

The re-projection error is the gold-standard metric for building the objective function [24, 25]. It is typically expressed as $\|\boldsymbol{v}-\frac{\boldsymbol{p}(\boldsymbol{\theta})}{\lambda(\boldsymbol{\theta})}\|$, where $\boldsymbol{v}$ is an observation, $\boldsymbol{p}$ denotes an estimated 3D point, λ represents the point depth, and $\boldsymbol{\theta}$ indicates the parameter vector of the vision system. The re-projection error is a fraction because λ depends on $\boldsymbol{\theta}$.

Minimizing the objective function based on re-projection error can be challenging owing to its numerous fractions. A popular treatment is taking λ as an independent variable and moving it out of the denominator [26, 27]. Then, the error becomes $\|\boldsymbol{v}\lambda - \boldsymbol{p}(\boldsymbol{\theta})\|$. The term $\boldsymbol{v}\lambda$ can be viewed as the back-projection of an observation into the 3D space. This error measurement is sometimes referred to as the space error.

The space error is easier to minimize but less accurate than the re-projection error because it does not align with the distribution of vision noise, often re-weighting 3D points by using the point depth λ.

To simplify the error metric and preserve accuracy, we follow [27] and take the inverse point depth, $s = \frac{1}{\lambda}$, as an independent variable. This transforms our error expression to $\|\boldsymbol{v} - \boldsymbol{p}(\boldsymbol{\theta})s\|$. The benefits of our error definition are twofold. First, the resulting objective function becomes a low-order polynomial, making it easier to minimize. Second, the error measurement is conducted from the camera side, ensuring that the observation term $\boldsymbol{v}$ remains unaffected in our error expression. Moreover, our error can be mapped back to the re-projection error with a negligible difference using a known constant factor. Further details are provided in Section 3.

The concept of inverse point depth has been explored in extended Kalman filter (EKF)-based SLAM approaches [28, 29]. However, their application of inverse point depth differs significantly from ours. In EKF methods, inverse depths serve as intermediate variables to parameterize 3D points but are not treated as independent. The error measurement in these methods remains based on the re-projection error. Conversely, our approach treats inverse point depths as independent variables, allowing us to define a new objective function.

2.5 Learning-based method

Recently, learning-based methods [30–33] have been introduced to solve the SLAM problem. These approaches often utilize a multilayer perceptron (MLP) as a scene encoder, to query scene density and color using view coordinates. A differential rendering process then synthesizes images from this data. By comparing these rendered images with a set of input images, gradient descent is used to optimize the MLP and the view coordinates. Gradients are computed on a set of keyframes or sampled pixels to manage computational demands and prevent catastrophic forgetting.

While learning-based methods enable joint camera tracking and dense surface reconstruction, they require significant computational power. Currently, only room-size problems can be processed in real-time using high-performance GPUs. Importantly, the issue of catastrophic forgetting still prevents these learning-based methods from achieving a global optimum in large-scale BA problems. Consequently, the traditional BA problem, which focuses on spare-matched 2D features, still needs investigation and remains the primary focus of this study.

3 Our algorithm

In this section, we will first describe how to formulate the modified re-projection error using inverse depth for pinhole cameras. We will then delve into the specifics of three tasks: refining camera pose, refining 3D points, and refining the camera's intrinsic parameters.

Notations. We suppose a BA problem involving n 3D points observed in m images. The ith 3D point in the world coordinate frame is denoted as $\boldsymbol{p}_i = [X, Y, Z]_i^{\mathrm{T}}, i \in \{1, \ldots, n\}$. The camera pose of the jth image is represented by a rotation matrix $\boldsymbol{R}_j$ and a translation vector $\boldsymbol{t}_j$, $j \in \{1, \ldots, m\}$. The local coordinates of the ith 3D point under the jth camera pose are $\boldsymbol{R}_j \boldsymbol{p}_i + \boldsymbol{t}_j$. Since cameras might share intrinsic parameters, we denote the set of focal length parameters as $f_k, k \in \{1, \ldots, K\}$.

3.1 Modified re-projection error

The re-projection of the ith 3D point onto the jth image using a pinhole camera model can be formulated as follows:

$$\hat{\boldsymbol{v}}_{i,j} = \left[\frac{(\boldsymbol{p}_i^{\mathrm{T}} r_j^1 + \boldsymbol{t}_j^1) f_k}{\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{r}_j^3 + \boldsymbol{t}_i^3}, \frac{(\boldsymbol{p}_i^{\mathrm{T}} r_j^2 + \boldsymbol{t}_j^2) f_k}{\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{r}_j^3 + t_j^3}, f_k \right]^{\mathrm{T}}, \tag{1}$$

where $\boldsymbol{r}_j^3$ is the 3rd row of $\boldsymbol{R}_j$, t_j^3 the 3rd component of $\boldsymbol{t}_j$. The 3rd term of $\hat{\boldsymbol{v}}_{i,j}$ is f_k because it is on the image plane, as shown in Figure 1.

We denote the observation of the ith point on the jth image plane as follows:

$$\boldsymbol{v}_{i,j} = [u_{i,j}, v_{i,j}, f_k]^{\mathrm{T}}, \tag{2}$$

where $\boldsymbol{v}_{i,j}$ is computed from the pixel position of the corresponding feature using the camera's intrinsic parameters, which are obtained by either calibration or initialization. For simplicity, we consider $\boldsymbol{v}_{i,j}$ as known measurements when estimating camera poses and 3D points. In most SLAM and SfM tasks, the cameras are already calibrated. We will also show how to refine the camera's intrinsic parameters in Subsection 3.2.3.

The re-projection error is the difference between the observation of a 3D point and its re-projection [24]. To facilitate the formulation in our paper, we write the re-projection error $e_{i,j}^r$ as follows:

$$e_{i,j}^r = \|\boldsymbol{v}_{i,j} - \hat{\boldsymbol{v}}_{i,j}\|_2 = \left\| \boldsymbol{v}_{i,j} - (\boldsymbol{R}_j \boldsymbol{p}_i + \boldsymbol{t}_j) \frac{f_k}{\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{r}_j^3 + t_j^3} \right\|_2, \tag{3}$$

where $\|\cdot\|_2$ denotes the Euclidean distance.

Reformulation with inverse depth. The term $\frac{f_k}{\boldsymbol{p}_i^{\mathrm{T}} \boldsymbol{r}_j^3 + t_j^3}$ in (3) can be viewed as the inverse depth of $\boldsymbol{p}_i$ in the jth local frame scaled by the focal length f_k. We propose to take this term as an augmented variable and denote it as $s_{i,j}$. Therefore, the modified re-projection error function is expressed as follows:

$$e_{i,j} = \|\boldsymbol{v}_{i,j} - (\boldsymbol{R}_j \boldsymbol{p}_i + \boldsymbol{t}_j) s_{i,j}\|_2. \tag{4}$$

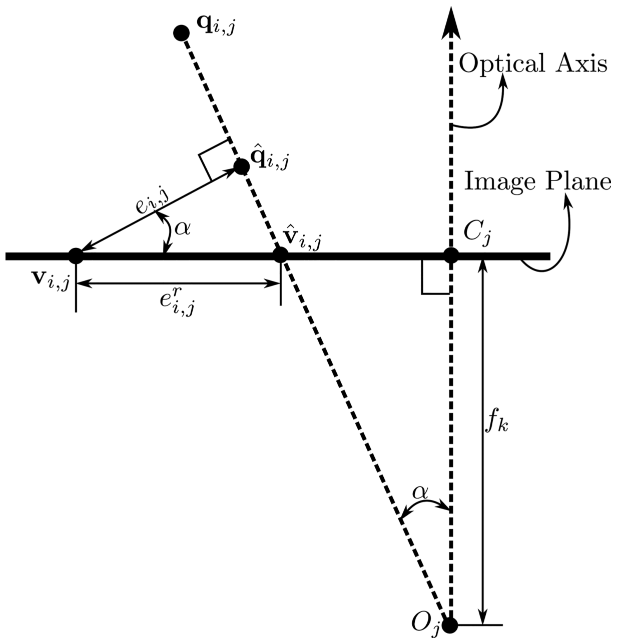

Figure 1 Pinhole camera projection of the ith point on the jth image. $\boldsymbol{q}_{i,j} = \boldsymbol{R}_j \boldsymbol{p}_i + \boldsymbol{t}_j$ represents the position of the point within the local camera coordinate frame. $\hat{\boldsymbol{v}}_{i,j}$ denotes the re-projection position of the point on the image plane. $\boldsymbol{v}_{i,j}$ indicates the observed position of the point. $\hat{\boldsymbol{q}}_{i,j} = \boldsymbol{q}_{i,j} s_{i,j}$ is the modified re-projection position of the point by our method. $e^r_{i,j}$ signifies the original re-projection error, and $e_{i,j}$ is the modified error. O_j is the principal point of the camera, where C_j is the intersection point between the image plane and the camera's optical axis.

By introducing $s_{i,j}$, we replace division operations with multiplication. This reduces the nonlinearity of the objective function, simplifying the derivation of closed form optimal solutions (please refer to the details in Subsections 3.2.1 and 3.2.2).

The accuracy of the modified error $e_{i,j}$ presented in (4) can be studied by comparing it with the original re-projection error $e^r_{i,j}$ in (3). We now show that $e_{i,j}$ can be explained as a re-weighted $e^r_{i,j}$. Specifically, $e_{i,j} = e^r_{i,j}\cos(\alpha_{i,j})$, where $\alpha_{i,j}$ is the angle between the optical axis and the re-projection line of the 3D point (see Figure 1).

As $s_{i,j}$ is an independent variable, the solution for $s_{i,j}$ that minimizes (4) must be optimal concerning the other variables. When these other variables are fixed, the problem about $s_{i,j}$ is a simple linear system. Consequently, the solution to $s_{i,j}$ is as follows:

$$s_{i,j} = \frac{\boldsymbol{q}_{i,j}^{\mathrm{T}} \boldsymbol{v}_{i,j}}{\boldsymbol{q}_{i,j}^{\mathrm{T}} \boldsymbol{q}_{i,j}},$$

where $\boldsymbol{q}_{i,j} = \boldsymbol{R}_j \boldsymbol{p}_i + \boldsymbol{t}_j$ is the position of the ith 3D point under the jth camera pose. The error expression in (4) can be written as $\|\boldsymbol{v}_{i,j} - \boldsymbol{q}_{i,j} s_{i,j}\|$. The role of $s_{i,j}$ is evident, as seen in Figure 1. Specifically, it finds a point $\hat{\boldsymbol{q}}_{i,j} = \boldsymbol{q}_{i,j} s_{i,j}$ on the line between $\boldsymbol{q}_{i,j}$ and O_j, so that $\hat{\boldsymbol{q}}_{i,j}$ closely matches the observation $\boldsymbol{v}_{i,j}$. Therefore, the three points $\langle \boldsymbol{v}_{i,j}, \hat{\boldsymbol{q}}_{i,j}, \hat{\boldsymbol{v}}_{i,j} \rangle$ form a perpendicular triangle. This triangle is similar to the triangle $\langle O_j, C_j, \hat{\boldsymbol{v}}_{i,j} \rangle$, as shown in Figure 1. Therefore, we can deduce that $e_{i,j} = e^r_{i,j}\cos(\alpha_{i,j})$, with an optimal inverse depth value. Thus, we can use $\frac{1}{\cos(\alpha_{i,j})} e_{i,j}$ to re-weight $e_{i,j}$ back to the original re-projection error $e^r_{i,j}$.

Moreover, the re-weighting factor $\frac{1}{\cos(\alpha_{i,j})} = \frac{\|\hat{\boldsymbol{v}}_{i,j}\|}{f_k}$ can be further approximated by a known constant coefficient. As the magnitude of $\hat{\boldsymbol{v}}_{i,j} - \boldsymbol{v}_{i,j}$ is usually much smaller than f_k in BA, we could safely ignore the difference such that $\phi_{i,j} := \frac{\|\boldsymbol{v}_{i,j}\|}{f_k} \approx \frac{\|\hat{\boldsymbol{v}}_{i,j}\|}{f_k} = \frac{1}{\cos(\alpha_{i,j})}$. Since $\boldsymbol{v}_{i,j}$ and f_k are known, the re-weighting factor can be fixed.

In summary, we use $\phi_{i,j} e_{i,j}$ as our error measurement. It well approximates the re-projection error while still maintaining its properties when deriving closed form solutions.

Remark 1. The previous discussion points out that the original re-projection error tends to assign greater weight to pixels with larger projection angles. These pixels, typically located further from the image center, are more prone to distortion, blur, and brightness decay. Therefore assigning significant weight to them is not practical. Another error measurement version, where $\phi_{i,j} = 1$, changes this unbalanced weighting. Hence, we reserve $\phi_{i,j} = 1$ as an option to build the objective function. To clarify notations, we call "V-metric" for $\phi_{i,j} = 1$ and "Z-metric" for $\phi_{i,j} = \frac{\|\boldsymbol{v}_{i,j}\|}{f_k}$. In our experiments, both metrics demonstrate similar performance, each achieving higher accuracy compared to methods relying solely on the re-projection error.

3.2 Alternative optimization

According to our error metric, the objective function in our algorithm is expressed as follows:

$$J = \sum_{i,j\in\Omega} \mu_{i,j} e_{i,j}^2, \tag{5}$$

where $(i,j) \in \Omega$ indicates that the ith point is visible on the jth image, and $\mu_{i,j}$ is a mixed weight to account for our correcting factor $\phi_{i,j}$, uncertainty [2] and the re-weighting factor of robustification [4], to control the influence of a specific error term. Note that $\mu_{i,j}$ is a known and constant number in each iteration, where $e_{i,j}$ is defined in (4).

3.2.1 *Camera pose refinement*

When refining camera poses, we keep the 3D point variables and the camera's intrinsic parameters constant. This means that each error term involves only one camera pose at a time. Thus, we can update each camera pose independently and omit the subscript j when referring to the jth camera pose.

The objective function about the jth camera pose is as follows:

$$J_j(\boldsymbol{R},\boldsymbol{t}) = \sum_{i\in\Omega_j} \mu_i \|\boldsymbol{v}_i - (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t})s_i\|_2^2, \tag{6}$$

where $i \in \Omega_j$ means that the ith point is captured on the jth image. Setting the derivative with respect to $\boldsymbol{t}$ to zero yields the solution to $\boldsymbol{t}$:

$$\boldsymbol{t}^* = \frac{\sum_{i\in\Omega_j} (\boldsymbol{v}_i s_i - \boldsymbol{R}\boldsymbol{p}_i s_i^2)\mu_i^2}{\sum_{i\in\Omega_j} s_i^2 \mu_i^2}. \tag{7}$$

Substituting (7) to (6) yields the following:

$$\begin{aligned} J_j &= \sum_{i\in\Omega_j} \mu_i \left\| \boldsymbol{v}_i - \left(\boldsymbol{R}\boldsymbol{p}_i + \frac{\sum_{i\in\Omega_j} \boldsymbol{v}_i s_i + \boldsymbol{R}\sum_{i\in\Omega_j} \boldsymbol{p}_i s_i^2}{\sum_{i\in\Omega_j} s_i^2} \right) s_i \right\|_2^2 \\ &= \sum_{i\in\Omega_j} \mu_i \left\| \boldsymbol{x}_i - \boldsymbol{R}\boldsymbol{y}_i \right\|_2^2, \end{aligned} \tag{8}$$

where

$$\boldsymbol{x}_i = \boldsymbol{v}_i - \frac{\sum \boldsymbol{v}_i s_i}{\sum s_i^2} s_i, \quad \boldsymbol{y}_i = \left(\boldsymbol{p}_i + \frac{\sum \boldsymbol{p}_i s_i^2}{\sum s_i^2} \right) s_i.$$

Globally minimizing (8) constitutes a special case of the absolute orientation problem, for which closed form solutions are available [34].

3.2.2 *3D point refinement*

Similarly, we refine 3D points and their inverse depth on different images while keeping camera poses and intrinsic parameters fixed. For clarity, the subscript i is omitted when referring to the ith point. The objective function about the ith point can be formulated as follows:

$$J_i(\boldsymbol{p}, s_j) = \sum_{j\in\Omega_i} \mu_j \|\boldsymbol{v}_j - (\boldsymbol{R}_j\boldsymbol{p} + \boldsymbol{t}_j)s_j\|_2^2, \tag{9}$$

where Ω_i represents the set of images on which the ith 3D point is detected.

The objective function in (9) is nonlinear because the term $(\boldsymbol{R}_j\boldsymbol{p}+\boldsymbol{t}_j)s_j$ involves the product of optimization variables. Therefore, the closed form solutions for $\boldsymbol{p}$ and s_j are not readily accessible. Fortunately, since Eq. (9) is a low-order polynomial, it can be efficiently minimized using conventional optimization methods, such as Newton's method. In this paper, we introduce an alternative scheme that simplifies the derivation of closed form solutions.

Specifically, we replace $\boldsymbol{p}$ and s_j with $\boldsymbol{p}+\delta\boldsymbol{p}$ and $s_j+\delta s_j$, respectively. By fixing the current values of $\boldsymbol{p}$ and s_j, we treat $\delta\boldsymbol{p}$ and δs_j as new variables. With simple variable substitution and combination, Eq. (9) can be reformulated as follows:

$$J_i(\delta\boldsymbol{p},\delta s_j)=\sum_{j\in\Omega_i}\mu_j\|\hat{\boldsymbol{e}}_j-\delta\boldsymbol{p}s_j-\hat{\boldsymbol{q}}_j\delta s_j-\delta\boldsymbol{p}\delta s_j\|_2^2,$$

where $\hat{\boldsymbol{e}}_j=\boldsymbol{R}_j^{\mathrm{T}}(\boldsymbol{v}_j-(\boldsymbol{R}_j\boldsymbol{p}+\boldsymbol{t}_j)s_j)$ and $\hat{\boldsymbol{q}}_j=\boldsymbol{R}_j^{\mathrm{T}}(\boldsymbol{R}_j\boldsymbol{p}+\boldsymbol{t}_j)$. Since the term $\delta\boldsymbol{p}\delta s_j$ is of second order, we propose to drop it, thereby obtaining a linearized objective function:

$$\bar{J}_i(\delta\boldsymbol{p},\delta s_j)=\sum_{j\in\Omega_i}\mu_j\|\hat{\boldsymbol{e}}_j-\delta\boldsymbol{p}s_j-\hat{\boldsymbol{q}}_j\delta s_j\|_2^2. \tag{10}$$

The analytical solution to minimize the linear (10) can be easily derived. Specifically, the solution of δs_j is as follows:

$$\delta s_j^*=\frac{\hat{\boldsymbol{q}}_j^{\mathrm{T}}(\hat{\boldsymbol{e}}_j-\delta\boldsymbol{p}s_j)}{\hat{\boldsymbol{q}}_j^{\mathrm{T}}\hat{\boldsymbol{q}}_j}. \tag{11}$$

Substituting (11) to (10) yields the following:

$$\begin{aligned}\bar{J}_i(\delta\boldsymbol{p})&=\sum_{j\in\Omega_i}\mu_j\left\|\hat{\boldsymbol{e}}_j-\delta\boldsymbol{p}s_j-\hat{\boldsymbol{q}}_j\frac{\hat{\boldsymbol{q}}_j^{\mathrm{T}}(\hat{\boldsymbol{e}}_j-\delta\boldsymbol{p}s_j)}{\hat{\boldsymbol{q}}_j^{\mathrm{T}}\hat{\boldsymbol{q}}_j}\right\|_2^2\\&=\sum_{j\in\Omega_i}\mu_j\left\|\underbrace{\left(I-\frac{\hat{\boldsymbol{q}}_j\hat{\boldsymbol{q}}_j^{\mathrm{T}}}{\hat{\boldsymbol{q}}_j^{\mathrm{T}}\hat{\boldsymbol{q}}_j}\right)}_{\boldsymbol{A}_j}(\hat{\boldsymbol{e}}_j-\delta\boldsymbol{p}s_j)\right\|_2^2,\end{aligned} \tag{12}$$

where I is an identity matrix. It is worth noting that $\boldsymbol{A}_j^{\mathrm{T}}\boldsymbol{A}_j=\boldsymbol{A}_j$. Then, the solution of $\delta\boldsymbol{p}$ is as follows:

$$\delta\boldsymbol{p}^*=\boldsymbol{A}_p^{-1}\boldsymbol{y}_p, \tag{13}$$

where $\boldsymbol{A}_p=\sum_{j\in\Omega_i}\boldsymbol{A}_js_j^2\mu_j$ and $\boldsymbol{y}_p=\sum_{j\in\Omega_i}\boldsymbol{A}_j\boldsymbol{e}_j\mu_j$.Importantly, $\boldsymbol{A}_p$ and $\boldsymbol{y}_p$ are accumulations, as seen in their formulas. This means that $\boldsymbol{A}_p$ and $\boldsymbol{y}_p$ can be computed in a distributed way. Details are seen in Subsection 4.2.5.

Our method offers a more computationally efficient alternative compared to the conventional Newton's method. Let $d_n=3+|\Omega_i|$, where $|\Omega_i|$ represents the number of the images on which the 3D point is visible. The naive Newton's method is employed, and a $d_n\times d_n$ linear equation system needs to be solved for $\delta\boldsymbol{p}^*$. By contrast, $\boldsymbol{A}_p$ in our method is only 3×3.

Theorem 1. The optimal solution to (10) is a descent direction about the original objective function.

Proof. Let the optimal solution that minimizes $\bar{J}_i$ in (10) be $\delta\boldsymbol{p}^*,\delta s_j^*$. We denote $\bar{\boldsymbol{e}}=\delta\boldsymbol{p}^*s_j+\hat{\boldsymbol{q}}_j\delta s_j^*$. Then,

$$\bar{J}_i(\delta\boldsymbol{p}^*,\delta s_j^*)=\sum_{j\in\Omega_i}\mu_j\hat{\boldsymbol{e}}_j^{\mathrm{T}}\hat{\boldsymbol{e}}_j-2\sum_{j\in\Omega_i}\mu_j\hat{\boldsymbol{e}}_j^{\mathrm{T}}\bar{\boldsymbol{e}}_j+\sum_{j\in\Omega_i}\mu_j\bar{\boldsymbol{e}}_j^{\mathrm{T}}\bar{\boldsymbol{e}}_j.$$

We suppose $\sum_{j\in\Omega_i}\mu_j\hat{\boldsymbol{e}}_j^{\mathrm{T}}\bar{\boldsymbol{e}}_j<0$, and then

$$\bar{J}_i(-\delta\boldsymbol{p}^*,-\delta s_j^*)=\bar{J}_i(\delta\boldsymbol{p}^*,\delta s_j^*)+4\sum_{j\in\Omega_i}\mu_j\hat{\boldsymbol{e}}_j^{\mathrm{T}}\bar{\boldsymbol{e}}_j\leqslant\bar{J}_i(\delta\boldsymbol{p}^*,\delta s_j^*).$$

However, this is not possible since Eq. (10) is a linear function, and its optimum is unique and global. Thus, $\sum_{j\in\Omega_i}\mu_j\hat{\boldsymbol{e}}_j^{\mathrm{T}}\bar{\boldsymbol{e}}_j \geqslant 0$.

Let h be a step size. The original objective function along the direction of $(\delta\boldsymbol{p}^*, \delta s_j^*)$ is as follows:

$$J_i(h) = \sum_{j\in\Omega_i}\mu_j\|\hat{\boldsymbol{e}}_j - (\delta\boldsymbol{p}^* s_j + \hat{\boldsymbol{q}}_j\delta s_j^*)h - (\delta\boldsymbol{p}^*\delta s_j^*)h^2\|_2^2. \tag{14}$$

Then,

$$\left.\frac{\mathrm{d}J_i}{\mathrm{d}h}\right|_{h=0} = -\sum_{j\in\Omega_i}\mu_j\hat{\boldsymbol{e}}_j^{\mathrm{T}}\bar{\boldsymbol{e}}_j \leqslant 0, \tag{15}$$

where J_i is a smooth and continuous polynomial. Since the directional derivative along $(\delta\boldsymbol{p}^*, \delta s_j^*)$ is non-positive, we conclude that this direction is a robust updating direction.

$J_i(h)$ is a fourth-order polynomial. To find the optimal value for h, we solve a third-order polynomial equation, for which well-known closed form solutions exist. Once the optimal step size h^* is determined, the 3D point is updated by $\boldsymbol{p} = \boldsymbol{p} + \delta\boldsymbol{p}^*h^*$ and $s_j = s_j + \delta s^*h^*$.

3.2.3 *Camera intrinsic parameter update*

This subsection presents the formulas for updating the camera's intrinsic parameters, which is unnecessary for cameras that are precalibrated.

The intrinsic parameters are encoded within the variables $\boldsymbol{v}_{i,j}$. When these parameters are known or fixed, one can treat $\boldsymbol{v}_{i,j}$ as constant coefficients. In this subsection, however, we focus on decoding intrinsic parameters from $\boldsymbol{v}_{i,j}$, while keeping other variables fixed.

Following [12], we assume that the optical center $[c_x, c_y]$ is positioned at the image center, as estimating these in BA is challenging. Moreover, we assume the horizontal and vertical focal lengths, f_x and f_y, are equal, denoted as $f_x = f_y = f$.

The focal length f already appears in $\boldsymbol{v}_{i,j}$. Furthermore, we account for radical distortion, which is usually modeled using the polynomial $\hat{r} = rP(r) = r + c_1r^2 + c_2r^4$. Here, $r = \sqrt{u^2+v^2}$ represents the distorted radius of a pixel, and $\hat{r}$ is the corrected radius. It is important to note that we use back-projection distortion coefficients, which correct the back-projection ray of a distorted pixel. This approach contrasts with the more widely used distortion parameters that calculate the distorted pixel position of an input ray.

Given the fixed 3D points and camera poses, the right side of the projection equation is a fixed 3D point. Then, the objective function of the kth camera is as follows:

$$J_k = \sum_{i\in\Omega_k}\mu_i\left\|\begin{bmatrix}P(r_i) & 0 & 0\\ 0 & P(r_i) & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u_i\\ v_i\\ f\end{bmatrix} - \begin{bmatrix}x_i\\ y_i\\ z_i\end{bmatrix}\right\|^2, \tag{16}$$

where $r_i = \sqrt{u_i^2+v_i^2}$, and $i \in \Omega_k$ indicates that the ith projection is made by the kth camera. The optimal solution to the focal length f is straightforward, as follows:

$$f^* = \frac{\sum_{i\in\Omega_k} z_k\mu_i^2}{\sum_{i\in\Omega_k}\mu_i^2}. \tag{17}$$

Let $\hat{r}_i = \sqrt{x_i^2+y_i^2}$. The objective function about c_1 and c_2 is also linear.

$$J_k(c_1, c_2) = \sum_{i\in\Omega_k}\mu_i\|r_i + c_1r_i^2 + c_2r_i^4 - \hat{r}_i\|^2. \tag{18}$$

Let $C_i = \left[\begin{smallmatrix}r_i^4 & r_i^6\\ r_i^6 & r_i^8\end{smallmatrix}\right]$. The solutions to c_1 and c_2 are

$$\begin{bmatrix}c_1\\ c_2\end{bmatrix}^* = \left(\sum_{i\in\Omega_k} C_i\mu_i^2\right)^{-1}\left(\sum_{i\in\Omega_k} C_i\begin{bmatrix}r_i^2\\ r_i^4\end{bmatrix}(\hat{r}_i - r_i)\mu_i^2\right). \tag{19}$$

Figure 2 (Color online) Examples of colored images from AirSim.

4 Experiments

To the best of our knowledge, there does not yet exist a very large BA dataset equipped with ground truth. For a comprehensive evaluation, we use public and synthetic datasets, structuring our evaluation as follows.

- We assess accuracy against ground truth using a dataset named ETHBA, which we derived from the ETH3D SLAM Benchmark [35].
- We evaluate robustness in BAL [12] in the presence of significant outliers. A synthetic dataset named ASBA, generated from AirSim [36], helps us evaluate robustness under conditions of inaccurate initialization.
- We compare the efficiency of our method across all datasets.
- We conduct SLAM experiments using the EuRoC [37] dataset.
- We perform distributed experiments using the ASBA dataset.

As usual in the field, translation errors are measured using Euclidean distance. Rotation measures are determined by the norm of the exponential coordinate of $R_g\boldsymbol{R}_t^{\mathrm{T}}$, where $\boldsymbol{R}_t$ is the estimated rotation matrix and R_g is its ground truth. All experiments ran on an Intel Core i5-4460 CPU at 3.20 GHz, with 8 GB of memory.

We compare our proposed alternating BA (ABA) with state-of-the-art methods, including CeresBA [38], Multicore (MBA) [9], RootBA [13], and ORB-SLAM3 [2]. The open-source codes for these methods were sourced online. We strictly adhere to the manuals of those codes and use their default control parameters to ensure a fair comparison. Unfortunately, we could not find any open-source code for ADMM-based methods.

Our approach is implemented in C++ and parallelized using the TBB library [39]. We plan to make both the code and dataset publicly accessible online.

4.1 Dataset

The BAL dataset [12] is probably the most popular dataset for BA in the literature and is employed in this study for evaluating efficiency and robustness. The images in BAL are either captured at a regular rate using a Ladybug camera or sourced from the internet. This makes BAL a real-world, extensive dataset that includes significant outliers, rendering it ideal for assessing efficiency and robustness. However, BAL lacks ground truth data for camera poses and 3D points, making it unsuitable for accuracy evaluation.

Accuracy evaluation is conducted on a synthetic dataset called ETHBA, which we created from the ETH3D SLAM benchmark [35]. This benchmark provides camera calibration, RGB-D images, and ground truth camera poses. We detect and match 2D features in the color images using SURF features [40], filtering out matches with a re-projection error larger than 4 pixels to prevent outlier interference in accuracy evaluations.

The 3D scenes in ETHBA dataset is small, so we turned to AirSim [36] to generate a large dataset. AirSim is a simulator designed for drones and cars, serving as a platform for AI research to experiment with deep learning, computer vision, and reinforcement learning algorithms. We manually control a drone to fly over a virtual city map called CityEnviron, collecting camera calibration data, camera poses, and RGB-D images by frame. Some examples of these color images are shown in Figure 2. The data format mirrors that of ETH3D, and we again use SURF feature to convert this data into a BA dataset, which we refer to as ASBA in this study.

ETHBA and ASBA use the same format as BAL and are readily readable by all comparison methods. Some basic information about the synthetic datasets is given in Table 1.

Meanwhile, we use the public dataset EuRoC [37] for the SLAM experiment as EuROC is readily readable by the ORB-SLAM3 code.

Table 1 Basic statistics of the ETHBA and the ASBA datasets.

	cables1	table7	repetitive	ASBA
Images	1158	1484	1015	32739
3D points	68824	11189	11639	3446776
Projections	860608	126835	167178	29738513

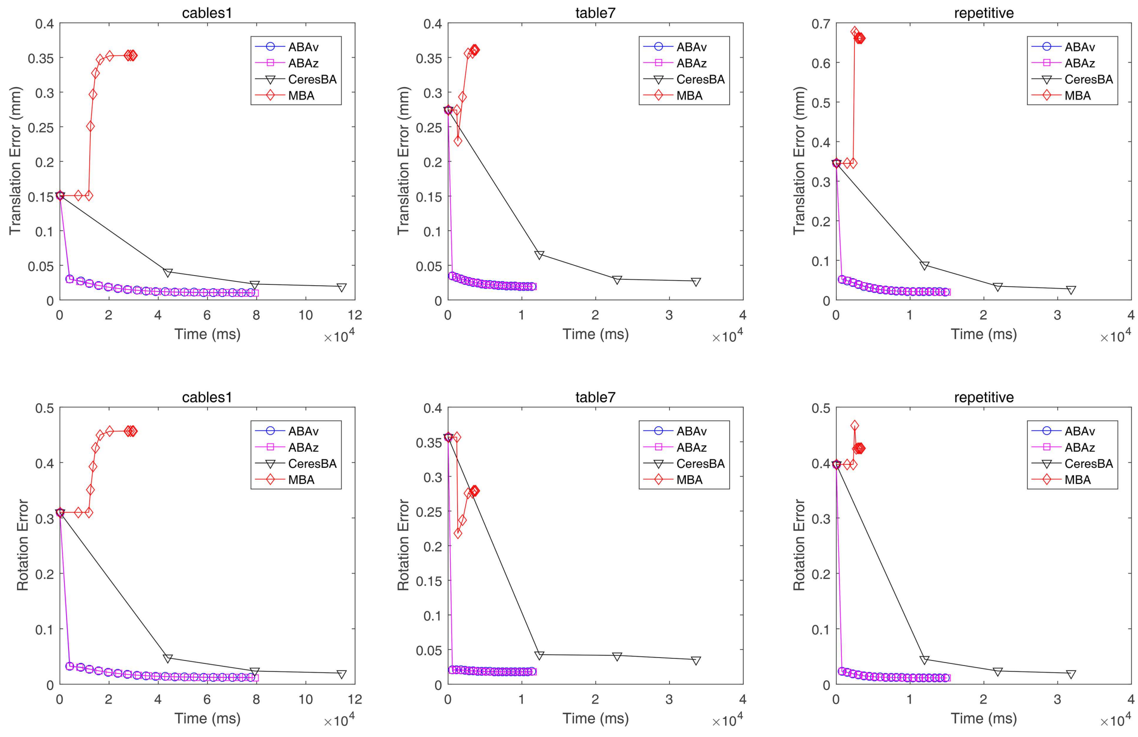

Figure 3 (Color online) Convergence curve concerning computational time on the ETHBA dataset.

Table 2 Camera pose errors of the compared algorithms on the ETHBA and the ASBA datasets.

		cables1	table7	repetitive	ASBA
Translation (mm)	ABA_v	0.0104	0.0196	0.0204	1.30
	ABA_z	0.0101	0.0194	0.0203	1.296
	CeresBA	0.0196	0.0274	0.0279	–
	MBA	0.3529	0.3608	0.6609	–
Rotation	ABA_v	0.0122	0.0183	0.0114	0.0042
	ABA_z	0.0118	0.0179	0.0113	0.0041
	CeresBA	0.0201	0.0356	0.0200	–
	MBA	0.4570	0.2790	0.4252	–

4.2 Experimental results

4.2.1 *Accuracy*

In our accuracy comparison using the ETHBA dataset, we benchmarked our method against the ground truth. Notably, RootBA is excluded in this experiment because the available RootBA code does not provide camera pose outputs. We denote our method as ABA_z for "Z-metric" and ABA_v for "V-metric".

The experimental results show that our method achieves the highest accuracy. The convergence curves of the evaluated algorithms are presented in Figure 3, with errors being summarized in Table 2. MBA fails to converge to the ground truth, probably due to its intrinsic inexact PCGs solver. The solutions of CeresBA are slightly less accurate than our method on every image sequence. Generally, the accuracy heavily depends on the objective function's design. Therefore, we can confidently conclude that our modified error is at least as accurate as the re-projection error used by CerasBA. Our experiments reveal no notable performance difference between the "Z-metric" and the "V-metric", indicating that the choice

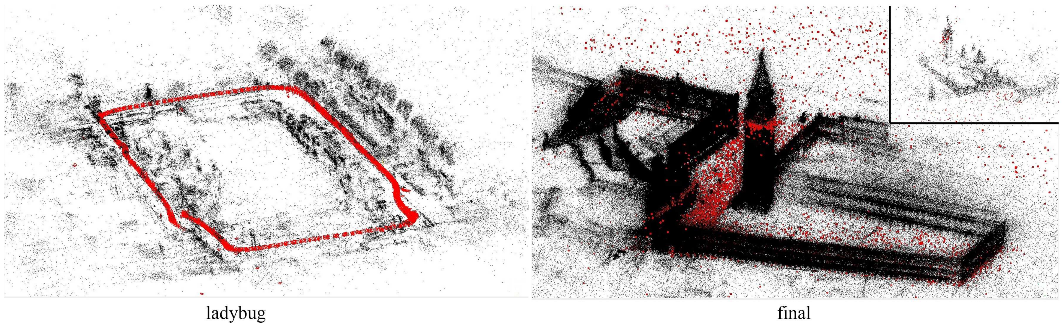

Figure 4 (Color online) Generated 3D structures and camera poses of our ABA$_v$ method on the "ladybug" and the "final" problem from BAL. The black dots correspond to 3D points, while the red blocks correspond to camera poses. The 3D structure is consistent with the scene, indicating that our algorithm works well on the two problems.

of re-weighting factors may not be crucial.

Further validation of our method's high accuracy is provided on the ASBA dataset, as shown in Table 2. Our method achieves an average camera translation error of approximately 1 mm, which is small compared to the dimensions of the city-size 3D structure.

4.2.2 *Robustness*

In our evaluation of robustness against significant outliers, we use the BAL dataset. Since there is no available ground truth, we assessed each algorithm's correctness by visually inspecting their output 3D structures. Our method appears accurate as its 3D structures align well with the target scene, as illustrated in Figure 4. All compared methods generated similar point clouds.

Robustness was also evaluated by examining convergence curves, using the objective function's value to represent the convergence behavior owing to the absence of ground truth for BAL. These convergence curves, presented in Figure 5, indicate that RootBA and our approach converge steadily. By contrast, rises are observed on the curves of MBA and CeresBA. It is worth noting that our method rigorously decreases the objective function in every iteration. Conversely, methods based on Newton's approach do not guarantee such consistent converge behavior without an exact line search. However, an exact line search is challenging to achieve with re-projection error owing to the complexity introduced by division operations.

We further evaluate the robustness of our method under conditions of heavily inaccurate initializations, using the ASBA dataset, which contains more than 2.9×10^7 images. Successfully solving ASBA demonstrates great reliability.

Our approach involved selecting two frames from ASBA, and initializing their poses using essential matrix estimation and decomposition, a common technique in SFM tasks. Subsequently, additional camera poses and 3D points are incrementally initialized by minimizing their corresponding space errors. The already initialized camera poses, and 3D points constitute a temporary BA problem, which we refine using our solver to mitigate cumulative errors.

The experimental results confirm the reliability of our method to handle very large BA problems despite inaccurate initializations. The resulting 3D structures and camera poses align well with the city landscape, as illustrated in Figure 6. The error of our method against ground truth is small, as shown in Table 2. Unfortunately, no other methods support incremental BA solving like ours. Thus, only the outcome of our method is presented.

4.2.3 *Efficiency*

Our experiments also highlight the efficiency of our method. As shown in Figure 3, where both CeresBA and ABA converge, our method needs only one-tenth of the computational time needed by CeresBA. In Figure 5, CeresBA's performance significantly slows as the problem scale increases to that of the BAL dataset. While PCG-based MBA achieves significant improvements in efficiency, RootBA and ABA are slightly faster than MBA.

Beyond computational speed, our method excels in memory efficiency. The baseline methods struggle with large-scale problems like the BAL final problem and ASBA owing to the memory limitations of

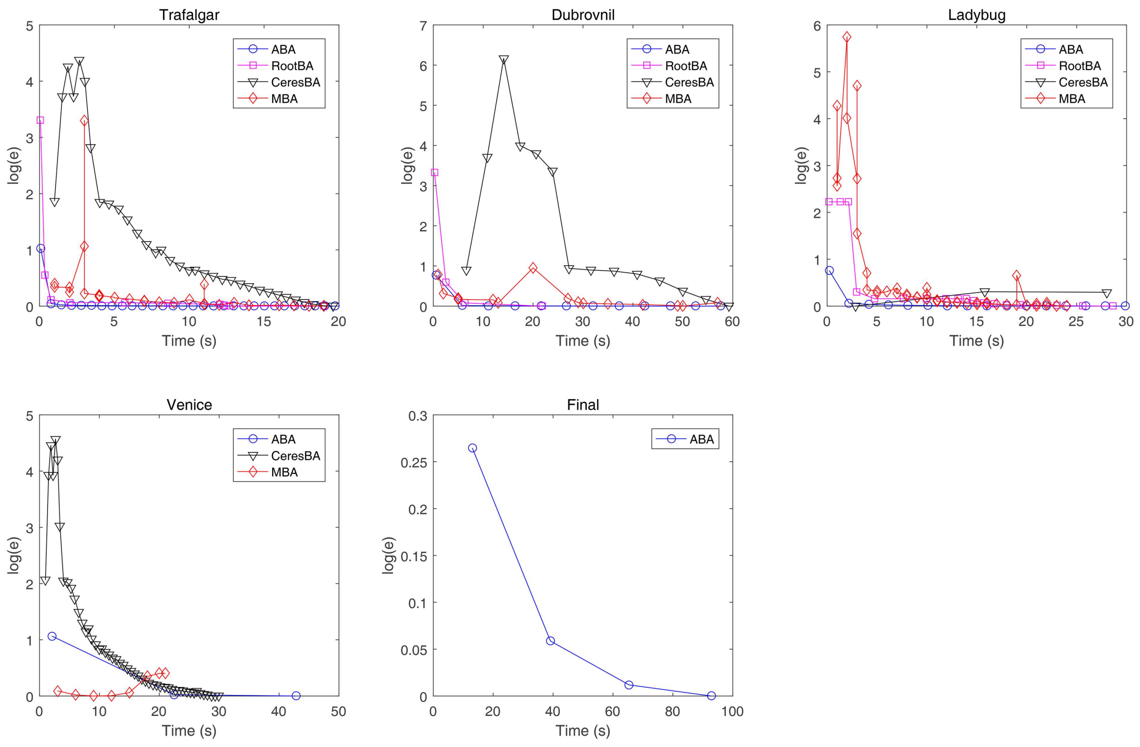

Figure 5 (Color online) Convergence curves on the BAL dataset. The absolute error value differs significantly among algorithms owing to different outlier rejection strategies or thresholds. To ensure a fair comparison, we use the logarithm of the relative error. Specifically, let $e(t)$ denote the error with respect to time, and let e^* be the minimum ever reached. Then, the convergence rate is measured by $\log(\frac{e(t)}{e^*})$. RootBA runs out of memory to solve the problem of Venice and final. CeresBA and MBA run out of memory on the problem of final. The maximum memory size of the experimental computer is 8 GB.

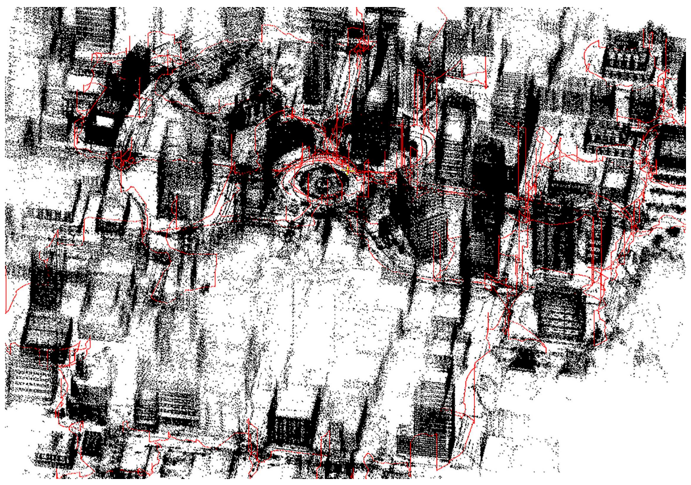

Figure 6 (Color online) 3D structure and camera trajectory of the output of our incremental BA on the ASBA dataset.

the experimental computer. By contrast, ABA only requires about 4 GB of memory to solve the two problems.

4.2.4 *Usefulness on SLAM*

We evaluate our method for SLAM using the open-source ORB-SLAM3 code [2]. In this experiment, we replaced the backstage BA solver with ABA$_v$, while keeping all other components, such as the tracking thread, key frame management, and frame rate, unchanged. Our method was then compared to the original ORB-SLAM3. The mean of the absolute translation error of the camera poses is measured using the evaluation script included in the ORB-SLAM3 source code.

Table 3 Translation error on the EuRoC dataset of SLAM experiments. The error is calculated by using the script in the ORB-SLAM3 source code. Both ABA and ORB-SLAM3 fail to solve the V103 and the V203 sequences.

	V101	V102	V201	V202	
ORB-SLAM3	0.031	0.012	0.021	0.017	
ABA_v	0.031	0.013	0.011	0.023	
	MH01	MH02	MH03	MH04	MH05
ORB-SLAM3	0.017	0.027	0.023	0.100	0.046
ABA_v	0.014	0.014	0.027	0.076	0.049

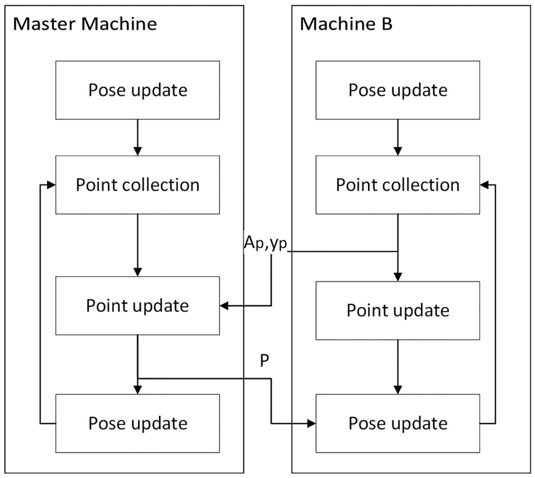

Figure 7 Workflow of distributed computation. $\boldsymbol{A}_p$ and $\boldsymbol{y}_p$ are defined in (13). $\boldsymbol{p}$ represents the updated value of the 3D point.

This experiment aims to evaluate the applicability of our method for SLAM rather than perform a quantitative comparison. Owing to the inherent randomness in dynamic processes, even minor interferences can lead to varying experimental results, potentially rendering quantitative comparisons meaningless. For SLAM task, balancing between efficiency and accuracy is crucial, as any inefficiencies can lead to error accumulation. Thus the focus is on whether an algorithm can complete SLAM tasks. CeresBA, MBA, and RootBA are not designed for small-scale, real-time tasks, and were not included in this evaluation.

The experimental results, presented in Table 3, confirm the effectiveness of our method for SLAM task. Our approach successfully accomplishes all tasks that ORB-SLAM3 does, achieving comparable accuracy.

4.2.5 *Distributed BA*

Our method is designed for efficient distributed computation. To illustrate this, we demonstrate a distributed implementation by partitioning camera pose variables. This setup results in certain 3D points, called overlap 3D points, which involve camera poses distributed across multiple machines. To update these overlapping 3D points, it is important to note that the matrix $\boldsymbol{A}_p$ and the vector $\boldsymbol{y}_p$ in (13) are accumulations. Each local machine computes its local $\boldsymbol{A}_p$ and $\boldsymbol{y}_p$. Then, the local $\boldsymbol{A}_p$ and $\boldsymbol{y}_p$ are transferred to a master machine to recover (13). Subsequently, the point is updated in the master machine and then broadcast back to the local machines. Figure 7 illustrates the workflow of our distributed computation.

The ASBA dataset is employed to evaluate our distributed method. We develop a simple trick to simulate distributed computation on a single machine by utilizing disk storage as supplementary memory. This allows the single computer to alternate roles, mimicking multiple machines. The original large ASBA dataset is partitioned into two blocks based on image indices: the first contains the initial 16369 images, while the remaining images form the second block. These blocks are loaded, processed, and saved alternately to solve the overall BA problem. A third file facilitates communication between the two blocks.

The experimental results confirm the effectiveness of our distributed method. As shown in Figure 8, the convergence curve aligns closely with the efficiency and robustness observed in Figure 3. Our distributed method mirrors the computational process of its nondistributed counterpart, whereas ADMM-based methods require an additional consensus strategy, potentially sacrificing efficiency. We believe that this

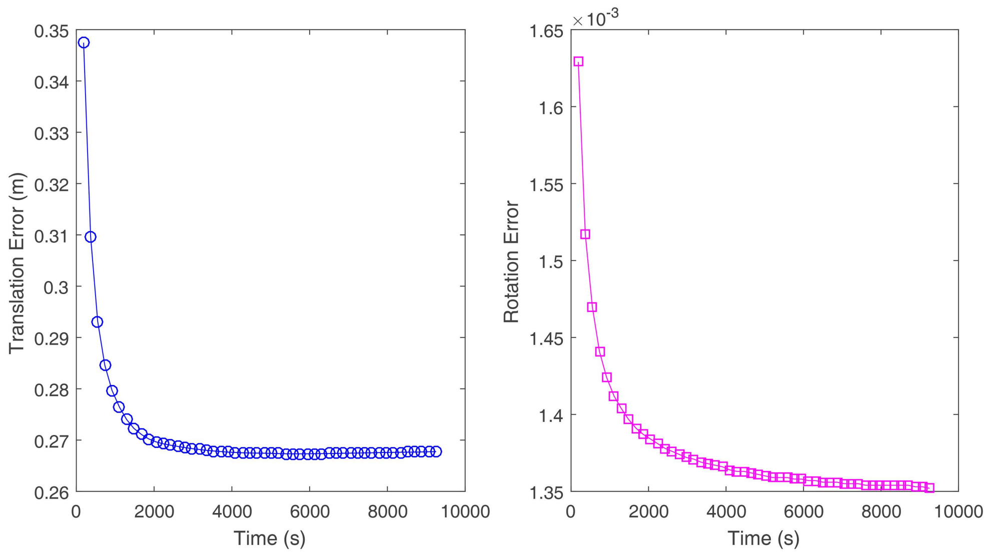

Figure 8 (Color online) Convergence curves of our distributed method on the ASBA dataset.

feature highlights the appeal of our approach.

This experiment also underscores the minimal communication load associated with our method. The size of the third file is 19.6 MB, which is relatively small compared to the 1.2 GB and the 1.4 GB block files.

5 Conclusion

This paper proposes a novel approach to solving the BA problem by utilizing AO. The inverse depth of each 3D point is introduced as an augmented independent variable. This results in a low-order polynomial error metric, which can be safely converted back to the traditional re-projection error using known constant factors. Consequently, we derived closed form formulas to optimally refine camera poses, 3D structures, and the camera's intrinsic parameters. Extensive experiments were conducted on various datasets, including small-scale, real-time SLAM, and large city-scale SfM cases. Experimental results demonstrate the advantages of our approach.

Our method is tailored to address fundamental BA problems and showcases impressive performance. However, it is not easily adaptable to generalized BA problems. Looking ahead, we plan to broaden our research to include more diverse feature types, such as lines and circles.

Acknowledgements Weiwei XU was partially supported by National Natural Science Foundation of China (Grant No. 62421003) and Key R&D Program of Zhejiang Province (Grant No. 2023C01181). This work was supported by Ant Group and Information Technology Center and State Key Lab of CAD&CG, Zhejiang University.

References

1 Mur-Artal R, Montiel J M M, Tardos J D. ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Trans Robot, 2015, 31: 1147–1163
2 Campos C, Elvira R, Rodriguez J J G, et al. ORB-SLAM3: an accurate open-source library for visual, visual-inertial, and multimap SLAM. IEEE Trans Robot, 2021, 37: 1874–1890
3 Schonberger J L, Frahm J M. Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 4104–4113
4 Triggs B, McLauchlan P F, Hartley R I, et al. Bundle adjustment — a modern synthesis. In: Proceedings of the International Workshop on Vision Algorithms, 1999. 298–372
5 Woodford O J, Rosten E. Large scale photometric bundle adjustment. 2020. ArXiv:2008.11762
6 Engel J, Koltun V, Cremers D. Direct sparse odometry. IEEE Trans Pattern Anal Mach Intell, 2018, 40: 611–625
7 Alismail H, Browning B, Lucey S. Photometric bundle adjustment for vision-based SLAM. In: Proceedings of Asian Conference on Computer Vision, 2017. 324–341
8 Konolige K, Garage W. Sparse sparse bundle adjustment. In: Proceedings of British Machine Vision Conference, 2010
9 Wu C, Agarwal S, Curless B, et al. Multicore bundle adjustment. In: Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR), 2011. 3057–3064
10 Zhang R, Zhu S, Fang T, et al. Distributed very large scale bundle adjustment by global camera consensus. In: Proceedings of the IEEE International Conference on Computer Vision, 2017. 29–38
11 Eriksson A, Bastian J, Chin T J, et al. A consensus-based framework for distributed bundle adjustment. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 1754–1762
12 Agarwal S, Snavely N, Seitz S M, et al. Bundle adjustment in the large. In: Proceedings of the European Conference on Computer Vision, 2010. 29–42

13 Demmel N, Sommer C, Cremers D, et al. Square root bundle adjustment for large-scale reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021. 11723–11732

14 Weber S, Demmel N, Chan T C, et al. Power bundle adjustment for large-scale 3D reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023. 281–289

15 Zhou L, Luo Z, Zhen M, et al. Stochastic bundle adjustment for efficient and scalable 3D reconstruction. In: Proceedings of the European Conference on Computer Vision, 2020. 364–379

16 Zhu S, Shen T, Zhou L, et al. Parallel structure from motion from local increment to global averaging. 2017. ArXiv:1702.08601

17 Zhu S, Zhang R, Zhou L, et al. Very large-scale global SFM by distributed motion averaging. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018. 4568–4577

18 Engels C, Stewénius H, Nistér D. Bundle adjustment rules. In: Proceedings of the Photogrammetric Computer Vision, 2006

19 Byröd M, Åström K. Conjugate gradient bundle adjustment. In: Proceedings of the European Conference on Computer Vision, 2010. 114–127

20 Ren J, Liang W, Yan R, et al. MegBA: a high-performance and distributed library for large-scale bundle adjustment. 2021. ArXiv:2112.01349

21 Ramamurthy K N, Lin C C, Aravkin A, et al. Distributed bundle adjustment. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshops, 2017

22 Mayer H. RPBA — robust parallel bundle adjustment based on covariance information. 2019. ArXiv:1910.08138

23 Demmel N, Gao M, Laude E, et al. Distributed photometric bundle adjustment. In: Proceedings of the International Conference on 3D Vision (3DV), 2020. 140–149

24 Hartley R, Zisserman A. Multiple View Geometry in Computer Vision. Cambridge: Cambridge University Press, 2003

25 Zheng Y, Kuang Y, Sugimoto S, et al. Revisiting the PnP problem: a fast, general and optimal solution. In: Proceedings of the IEEE International Conference on Computer Vision, 2013. 2344–2351

26 Hesch J A, Roumeliotis S I. A direct least-squares (DLS) method for PnP. In: Proceedings of the International Conference on Computer Vision, 2011. 383–390

27 Meng C, Xu W. ScPnP: a non-iterative scale compensation solution for PnP problems. Image Vision Comput, 2021, 106: 104085

28 Montiel J M M, Civera J, Davison A J. Unified inverse depth parametrization for monocular SLAM. In: Proceedings of the Robotics: Science and Systems, 2006

29 Civera J, Davison A J, Montiel J M M. Inverse depth to depth conversion for monocular SLAM. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2007. 2778–2783

30 Wang H, Wang J, Agapito L. Co-SLAM: joint coordinate and sparse parametric encodings for neural real-time SLAM. In: Proceedings of the Computer Vision and Pattern Recognition, 2023

31 Mildenhall B, Srinivasan P P, Tancik M, et al. NeRF: representing scenes as neural radiance fields for view synthesis. Commun ACM, 2021, 65: 99–106

32 Sucar E, Liu S, Ortiz J, et al. iMAP: implicit mapping and positioning in real-time. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021. 6209–6218

33 Zhu Z, Peng S, Larsson V, et al. NICE-SLAM: neural implicit scalable encoding for SLAM. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022

34 Horn B K P. Closed-form solution of absolute orientation using unit quaternions. J Opt Soc Am A, 1987, 4: 629–642

35 Schops T, Sattler T, Pollefeys M. BAD SLAM: bundle adjusted direct RGB-D SLAM. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), 2019

36 Shah S, Dey D, Lovett C, et al. AirSim: high-fidelity visual and physical simulation for autonomous vehicles. In: Proceedings of the Field and Service Robotics, 2017

37 Burri M, Nikolic J, Gohl P, et al. The EuRoC micro aerial vehicle datasets. Int J Robot Res, 2016, 35: 1157–1163

38 Sameer A, Keir M, Ceres Solver Team. Ceres Solver. 2023. http://ceres-solver.org

39 Pheatt C. Intel® threading building blocks. J Comput Sci Coll, 2008, 23: 298–298

40 Bay H, Ess A, Tuytelaars T, et al. Speeded-up robust features (SURF). Comput Vision Image Underst, 2008, 110: 346–359