Incremental 3-D pose graph optimization for SLAM algorithm without marginalization

Feng Youyang, Wang Qing and Yang Gaochao

Abstract
Pose graph optimization algorithm is a classic nonconvex problem which is widely used in simultaneous localization and mapping algorithm. First, we investigate previous contributions and evaluate their performances using KITTI, Technische Universität München (TUM), and New College data sets. In practical scenario, pose graph optimization starts optimizing when loop closure happens. An estimated robot pose meets more than one loop closures; Schur complement is the common method to obtain sequential pose graph results. We put forward a new algorithm without managing complex Bayes factor graph and obtain more accurate pose graph result than state-of-art algorithms. In the proposed method, we transform the problem of estimating absolute poses to the problem of estimating relative poses. We name this incremental pose graph optimization algorithm as G-pose graph optimization algorithm. Another advantage of G-pose graph optimization algorithm is robust to outliers. We add loop closure metric to deal with outlier data. Previous experiments of pose graph optimization algorithm use simulated data, which do not conform to real world, to evaluate performances. We use KITTI, TUM, and New College data sets, which are obtained by real sensor in this study. Experimental results demonstrate that our proposed incremental pose graph algorithm model is stable and accurate in real-world scenario.

Keywords
Pose graph optimization, computer vision, visual simultaneous localization and mapping, optimization method

Introduction
A robot moves around in unknown environment; it constructs the world model and its trajectory simultaneously. This is the classic simultaneous localization and mapping (SLAM) problem, which has been widely used in robotic areas. Accumulation of errors cannot be avoided if only odometry is used without loop closure part. Pose graph optimization (PGO) is used to estimate robot poses by solving nonconvex problem when loop closure is detected. Thus PGO is an efficient and necessary step to eliminate accumulated pose errors. PGO algorithm is distinguished into two categories: batch algorithm and incremental algorithm. Batch algorithm takes all of measurements into consideration and optimizes all of robot poses at the same time. This algorithm is often used when we have finished collecting sensor data, but batch algorithm only can provide off-line results. Incremental algorithm is more suitable to real-world scenario which incrementally takes the measurements into optimization and obtains real-time robot poses. In this algorithm, Schur complement is used to marginalize variables, which are outside of sliding window.

Corresponding author:
Wang Qing, Instrumental science and engineering, Southeast University, Nanjing, Jiang Su 210000, China.
Email: wq_seu@seu.edu.cn
The purpose of Schur complement in this article is different from bundle adjustment. In bundle adjustment, points in the world are marginalized using Schur complement, and only robot poses are optimized. The reason of this is to reduce amount of calculation. First estimation Jacobian (FEJ) problem should be taken into consideration during optimization. We analyze drawbacks of marginalization algorithm and put forward a new incremental PGO algorithm without using Schur complement algorithm like Georgia Tech Smoothing and Mapping (GTSAM) and SLAM++. In most of previous articles, simulated data are used to evaluate performance, which do not conform to real-world scenario. In this study, we evaluate previous PGO initialization algorithms and least squares solvers using real-world data. Then the best performance algorithm is used in the proposed incremental PGO algorithm.

In the second section, we review previous PGO algorithms and give their drawbacks. In the third section, we present formulation of PGO algorithm. In the fourth section, we present the proposed G-PGO algorithm. In the fifth section, details of experiments are provided. The experiment data sets are based on KITTI data set (sequence 00 and sequence 02), TUM data set, and New College data set. The main reason why we choose these data sets is that these data sets are generated by real sensor data, which are closer to the reality compared with synthetic data sets. In the sixth section, we make conclusion about our algorithm.

Related work

The PGO researches include least squares solvers, number of local minima, convex relaxations, Riemannian optimization, outlier detection, separability, and graph topology. In this article, we focus on solving outliers in loop closure and least squares solves. The loop closure detection is dependent on front end of SLAM, so the wrong place recognition often happens if there are many similar scenes in the world map. The outlier rejection is to reduce the influence of outliers on the cost function. The common used method is to replace quadratic error function with robust cost function (e.g. Huber function and Cauchy function). Another approach is to use loop closure constraints to obtain robot poses and then use subgroup results to correct robot poses. In real-world scenario, wrong place recognition often occurs if many similar scenes exit in this environment. In this article, we are inspired by Bai et al. and introduce loop closure constraint metric into our G-PGO algorithm, which is able to deal with wrong loop closure information.

Different initial nonlinear optimization values may affect final robot poses, so the initial values are important to global optimal solution. In closed-form PGO (COP)-SLAM, authors do not use the traditional nonlinear optimization but explore the COP result. The COP-SLAM method is a lightweight SLAM PGO back-end algorithm and consumes less time and can obtain accurate robot pose. Other initialization techniques for PGO are and LAGO. LAGO can only be used in planar scenarios, which cannot be used in three-dimensional (3-D) PGO initial value estimation. In Carlone et al., initialization method is divided into two steps. The orientations are firstly estimated using relative rotation measurements and then translations are estimated with orientations using nonlinear optimization algorithm. This initialization algorithm is called two-step method in this article. In “Initialization techniques” section, we compare PGO initialization algorithm between COP-SLAM and two-step algorithm. The experimental results show in the fifth section that two-step is more accurate than COP-SLAM algorithm using real-world data. Thus two-step initialization algorithm is used in our G-PGO algorithm.

In the literature, authors discuss the nonlinear and convergence of PGO but do not give the tool to verify the estimate and show that Lagrangian duality is an effective tool to solve PGO problem and can evaluate the quality of the optimization result. This is indeed a pioneering research work to pave a new way to solve PGO problem. Nasiri et al. use angle-axis parameters to deduce analytical PGO final result, which transforms classic iterative optimization algorithm to analytical solution.

After obtaining the initial state guess, cost function is modeled and Gauss–Newton or Dog-Leg algorithm is used to converge to final result. Different cost function and estimated parameters produce different final result. In Grisetti et al., the residual error is Lie algebra and the estimated variable is se(3). In, the estimated state is quaternion, and the residual error of cost function is produced by quaternion and translation. The Jacobian matrix is obtained by automatic derivative. In GTSAM, the residual error of cost function is Lie algebra but the Jacobian matrix is different from Barfoot. In “PGO cost function” section, we compare different cost function and variables, and the best one is used in our G-PGO algorithm.

Incremental smoothing and mapping (ISAM) and SLAM++ provide the incremental PGO algorithm, which takes FEJ problem into consideration. In the fourth section, we give analysis about their drawbacks and propose our incremental G-PGO algorithm, which is more robust and accurate compared with ISAM and SLAM++.

We testify which PGO cost function is more suitable to be used in optimization process according to previous PGO works. ISAM is the often-used algorithm in traditional incremental PGO algorithm; we put forward a new incremental PGO algorithm without using FEJ algorithm. At the same time, we add outlier rejection algorithm based on our G-PGO algorithm to form a complete algorithm to solve PGO problem.

Formulation of PGO algorithm

Notations: SO(3) denotes the special-orthogonal group which is a set of valid rotation matrices. SO(3) is
corresponded with so(3) in Lie algebra. The homogeneous rigid transformation on robot poses can be described by the special Euclidean group SE(3). SE(3) is corresponded with se(3) in Lie algebra. “Log()” operator can transform SO(3)/SE(3) to so(3)/se(3). “√” operator can transform so(3)/se(3) to SO(3)/SE(3).

The PGO problem is shown in Figure 1. We have measured relative transformation between node i and j, which corresponds to symbol $E_{ij}$ in Figure 1. The target of PGO algorithm is to look for the best robot poses, which can make $\sum ||e_{ij}||^2$ minimum. $e_{ij}$ is the cost function and estimated variables are robot poses, which correspond with black triangles in Figure 1. But the measurement is affected by noise and some of measurements are outliers. We should use outlier rejection algorithm to exclude these data before optimization. In this section, formulation of PGO algorithm is described in details, which includes initialization technique, cost function, and parameters of robot pose.

**Initialization techniques**

We know that PGO problem is a nonconvex problem, so the initial vales of estimated variable affect final iterative result. Good initial values advance iterative process to the global optimal solution. To obtain global optimal solution, a good initial state is an important step. For the time being, there are two strategies initializing PGO. The first one is COP-SLAM, which uses analytical deduction to obtain initial pose of robot. The second one is based on two-step optimization. The main idea is to solve for the rotations first and then estimate translation. We use experimental data to testify that the two-step algorithm is more accurate than COP-SLAM algorithm and the result is shown in the fifth section. We give brief descriptions about two-step algorithm below:

We use Lie algebra to form the first cost function

$$e_{\phi, ij} = \log(\Delta R_{ij}^{-1} R_i^{-1} R_j)$$  \hspace{1cm} (1)

$\Delta R_{ij}$ is the relative rotation matrix between pose i and j. $e_{\phi, ij}$ is residual error which is in so(3) space.

After initialization of rotation matrix, we form translation cost function as follows

$$e_{t, ij} = R_i \Delta t_{ij} + t_i - t_j$$  \hspace{1cm} (2)

$R_i$ has been known and $\Delta t_{ij}$ is the relative pose measurement. $t_i$ and $t_j$ are estimated translation variables. It is convenient to calculate the Jacobian matrix with respect to $t_i$ and $t_j$ because they are linear in (2), which means that these two parameters can be solved directly without using iterative nonlinear optimization algorithm. The benefit of two-step algorithm is that it divides PGO problem into two easier optimization problems.

COP-SLAM produces analytical initial results, which are easily affected by noise data. But two-step initialization algorithm uses nonlinear optimization algorithm which can use M-estimator function to reduce effect of noise. So two-step initialization technique is more accurate than COP-SLAM algorithm and experimental results are shown in the fifth section. In our G-PGO algorithm, we use two-step algorithm to initialize PGO values.

**PGO cost function**

After initialization step, we have obtained initial values of robot pose. The next step is to form nonlinear cost function to polish initial result. Different types of cost function and the Jacobian matrix lead to different final results. There are three types of cost function, which have been given in previous articles. In this article, we compare their performances using real-world data and prove which is more accurate. Here we give three cost functions directly:

1. Equation (3) is from CERES library. $q$ is the unit quaternion and $q^*$ is the conjugation of unit quaternion. $e_{ij}$ is the residual error. $q_i$ is the unit quaternion of robot pose with index i. $t_i$ is the translation of robot pose with index i. $\Delta T_{ij}$ is the relative pose between robot pose $T_i$ and robot pose $T_j$: $\Delta T_{ij} = T_j^{-1} \ast T_i$. $\Delta t_{ij}$ is the translation of $\Delta T_{ij}$ and $\Delta q_{ij}$ is the unit quaternion of $\Delta T_{ij}$. The estimated variables are unit quaternion $q_i$ and translation $t_i$.

$$\begin{bmatrix} q_i^*(p_j - p_i) - \Delta t_{ij} \\ \Delta d_{ij}(q_i^* \ast q_j^*) \end{bmatrix} = e_{ij}$$  \hspace{1cm} (3)

2. Residual errors of (4) are in se(3) space. This cost function is widely used in many SLAM algorithms such as ORB-SLAM.

$$\log\left((\Delta T_{ij}^{-1} \ast T_j^{-1} \ast T_i)^\vee\right) = e_{ij}$$  \hspace{1cm} (4)
3. In (5), the first three residual errors are in so(3) space and the last three residual errors are in Euclidean space. This cost function is used in Nasiri et al.\(^1\)\(^7\) and Bai et al.\(^1\)\(^8\)

\[
e_{ij} = \begin{bmatrix} \log(\Delta R^{-1}_{ij} R^{-1}_i R_j) \\ \Delta R^T_i (R^T_i (t_j - t_i) - \Delta t_{ij}) \end{bmatrix}_{6\times1} = \begin{bmatrix} e_{\omega, ij} \\ e_{t, ij} \end{bmatrix} \tag{5}
\]

In CERES library, (3) is used as PGO cost function, and the Jacobian matrix of this cost function is obtained by automatic derivatives using Jets operations.\(^1\)\(^4\) In (oriented FAST and rotated BRIEF) ORB-SLAM, the author uses (general graph optimization) G2O::EdgeSim3 structure to execute PGO optimization. In G2O::EdgeSim3, (4) is used as PGO cost function and the Jacobin matrix is obtained by numerical derivatives. Numerical derivative consumes much more time than analytical Jacobian matrix and iterative steps are larger than analytical ones. We show this phenomenon in the fifth section. In GTSAM library and “Robot State Estimation” book, analytical Jacobian matrix of (4) is derived. GTSAM library uses right multiplication to update pose, but Robot State Estimation book uses left multiplication to update pose. In Nasiri et al.\(^1\)\(^7\) and Bai et al.\(^1\)\(^8\) (5) is employed as cost function of PGO algorithm and supplementary materials of Bai et al.\(^1\)\(^8\) give analytical Jacobian matrix deduction.

We should note that the estimated variables of (5) are so(3) and translation vector. Thus nonlinear situation only occurs in rotation matrix R. It is different from (3) and (4) whose estimated variables are se(3) or unit quaternion and translation vector. The benefit of (5) is low nonlinear degree and more accurate result will be obtained. We compare batch PGO algorithms between CERES, GTSAM, ORB-SLAM, and (5). The experimental results are shown in the fifth section, which proves that (5) has smaller optimization steps and more accurate results.

**G-PGO**

It is easy for us to use (5) as cost function and corresponded Jacobian matrix to complete batch PGO. But batch PGO costs much more time, which cannot be finished online especially facing with large-scale areas. We want to estimate robot poses by recursively including recent loop closure measurement. This is incremental PGO algorithm. Here, we give an easy example to describe the differences between incremental and batch PGO.

There are eight poses in Figure 1 in which two-loop closures exist. The first-loop closure is made up of robot poses \{1,2,3,4\} and the second-loop closure is made up of \{2,3,4,5,6,7,8\}. For simplicity, we only bring pose-chain constraints, which are constructed by the robot pose \(n\) and robot pose \(n + 1\), into PGO. For off-line cases, we start batch PGO after obtaining robot pose 8. The Hessian matrix is shown in Figure 2. Off-line batch PGO optimizes all of robot poses at a time.

In ORB and many other SLAM algorithms, they do not use Schur complement to marginalize variables, which are outside of slide windowing. They just optimize robot poses in the current loop closure. For example, in Figure 1, the robot moves to m4 position and the first PGO starts when the loop closure is detected. Then the Hessian matrix transforms to Figure 3. After the first PGO, we have got the optimized robot poses \{1,2,3,4\}. The second PGO starts when m2 and m8 construct a loop edge. The Hessian matrix of the second PGO is shown in Figure 4. This incremental PGO does not take m1 robot pose into consideration during the second PGO process.

The incremental PGO should take FEJ problem into consideration. Compared with Figures 2 and 4, m1 is marginalized so the new Hessian matrix is \(A_c - A_b a_b^{-1} a_c\). The Hessian matrix of the second incremental PGO is shown in Figure 5. There is an important point we should notice that the Jacobian matrix which is related to m2 and m4 in Figure 5 should not use the iterative m2 and m4 states to calculate, on the contrary we should use the optimized m1 and m2 poses in the first loop to calculate the Jacobian matrix. If we do so, consistency problem is solved. The consistency problem has been published on paper\(^8\) and we do not describe in details here.
Figure 4. The Hessian matrix of PGO algorithm when the second-loop closure is detected. The blue squares are filled by the Jacobian matrix and the white squares are empty. PGO: Pose graph optimization.

Figure 5. The Hessian matrix of correct incremental PGO algorithm when the second-loop closure is detected. The blue squares are filled by the Jacobian matrix and the white squares are empty. The red squares are affected by Schur complement and we should use the first-pose graph optimized pose results to calculate Jacobian matrices in red squares, which are related to robot pose m1. PGO: Pose graph optimization.

It is very complex to manage incremental algorithm because of consistency problem. The consistency problem can be solved by using initial Jacobian matrix, but it also leads to local optimal solution not the global optimal solution. We explain this situation in Figure 6. The estimated variables are x1 and x2. We have obtained initial values of x1 and x2 at point P1. P2 is the second iterative point if we optimize x1 and x2 together. P3 is the final global optimal solution. If we marginalize x1 at point P1, we obtain marginalized optimized result at P2'. The marginalized final solution is at P3'. The marginalized result cannot converge to the global optimal solution. But if initial value P1 is close enough to P3, marginalized result P3' is equal to P3.

Here, we put forward a new incremental PGO algorithm and simplify incremental PGO Hessian matrix management. We also take Figure 1 as an example to describe G-PGO. When the first loop is detected between m1 and m4, we execute batch PGO optimization which is described in (5). We obtain optimized robot poses \{1, 2, 3, 4\}. We calculate poses between optimized robot poses, so relative poses \{\Delta T_{12}, \Delta T_{23}, \Delta T_{34}\} are obtained. \Delta T_{ij} is relative pose between pose i and pose j, k means that this relative pose is obtained by the kth loop closure optimization. \{\Delta T_{12}, \Delta T_{23}, \Delta T_{34}\} are measurements for estimated states \{\Delta T_{12}, \Delta T_{23}, \Delta T_{34}\}. When the second-loop closure occurs, we also use the batch PGO method to optimized robots poses \{2, 3, 4, 5, 6, 7, 8\}. We calculate the relative poses between the second-loop closure poses and we obtain \{\Delta T_{23}, \Delta T_{34}, \Delta T_{45}, \Delta T_{56}, \Delta T_{78}\}. Now, we have got two measurements for \Delta T_{23} and these two measurements are \{\Delta T_{12}, \Delta T_{34}\}. The same situation also happens in \Delta T_{24}. Before constructing cost function, we should deal with wrong loop closure information. We use constraint equation to produce metric to judge whether this measurement should be taken into optimization. If constraint metric is too large, we abandon measurement from this loop closure. Equation (6) shows how to calculate this constraint metric

$$
\log \left[ \Delta T_{i,m}^{\text{Loop}} \Delta T_{i+1,m}^{\text{VO}} \Delta T_{i+2,m}^{\text{VO}} \cdots \Delta T_{m-1,m}^{\text{VO}} \right]^{\top} = p
$$

In (6), \Delta T_{ij}^{\text{VO}} is relative pose between robot pose i and j obtained by odometry. \Delta T_{i,m}^{\text{Loop}} is relative pose between robot pose i and m obtained by loop closure. Robot pose i and m construct a loop closure. p is constraint metric.
example in Figure 1, estimated relative pose $\Delta T_{23}$ obtain two measurements $\{\Delta T_{23}^1, \Delta T_{23}^2\}$. The cost metrics of $\Delta T_{23}^1$ and $\Delta T_{23}^2$ are calculated as follows

$$\log \left[ \frac{1}{4} \left( \Delta T_{23}^{\text{app}} \right)^{-1} \Delta T_{23}^{\text{app}} \right] = p^1_{23}$$

$$\log \left[ \frac{1}{8} \left( \Delta T_{23}^{\text{app}} \right)^{-1} \Delta T_{23}^{\text{app}} \right] = p^2_{23}$$

$p^1_{23}$ is the cost metric of $\Delta T_{23}^1$, and $p^2_{23}$ is the cost metric of $\Delta T_{23}^2$. If $p^1_{23}$ is too large, we abandon $\Delta T_{23}^1$ measurement and only $\Delta T_{23}^2$ is taken into optimization. The gap between loop closure measurement and visual odometry (VO) accumulated result is large. The cause of large gap is that accumulated error exists in odometry. Although odometry produces accumulated errors, its relative measurement between poses $i$ and $i+1$ is accurate. If we distribute this total gap into every relative measurement, this average gap should be small no matter how long the trajectory is. We call this average gap as constraint metric. If a wrong loop closure occurs, its constraint metric must be very large because $\left( \Delta T_{i,m}^{\text{app}} \Delta T_{i+1,m}^{\text{app}} \Delta T_{i+2,m}^{\text{app}} \ldots \Delta T_{m-1,m}^{\text{app}} \right)$ must be far from identity matrix.

After removing wrong loop closure measurement, we construct the cost function for estimated variable $\Delta T_{23}$

$$e_{\Delta T_{23}} = e(\Delta T_{23}, \Delta T_{23}^m)^T \ast W_1 \ast e(\Delta T_{23}, \Delta T_{23}^m) + e(\Delta T_{23}, \Delta T_{23}^m)^T \ast W_2 \ast e(\Delta T_{23}, \Delta T_{23}^m)^T$$

(9)

e is the cost function and $W_k$ is the weight matrix which is the residual error produced by the $k$th loop closure optimization.

We design two different methods to form cost function $e_{\Delta T_{23}}$. Cost function $e$ is obviously a unary vertex optimization problem, so we only need to calculate one Jacobian matrix.

**Method 1**

$$e(\Delta T_{ij}, \Delta T_{ij}^m) = \log \left( \left( \Delta T_{ij}^k \right)^{-1} \Delta T_{ij} \right)^\vee$$

(10)

$\Delta T_{ij}$ is to-be-estimated relative pose between pose $i$ and $j$. $\Delta T_{ij}^k$ is measurement relative pose which is obtained by the $k$th loop closure.

**Method 2**

$$e(\Delta T_{ij}, \Delta T_{ij}^m) = \log \left( \left( \Delta R_{ij}^k \right)^T \Delta R_{ij} \right)^\vee = \begin{bmatrix} e_{\phi, ij} \\ e_{t, ij} \end{bmatrix}$$

(11)

$\Delta R_{ij}^k$ is rotation matrix of $\Delta T_{ij}^k$ and $\Delta R_{ij}$ is rotation matrix of-to-be-estimated relative pose $\Delta T_{ij}$. $t_{ij}^k$ is the translation of $\Delta T_{ij}^k$ and $t_{ij}$ is the translation of $\Delta T_{ij}$.

According to our experiments in lab, we find that these two methods produce the same result. The main reason is that every measurement is close to each other and these measurements are very close to global solution after deleting wrong edge information. So we just show one of results in the fifth section. Here, we describe the flow diagram of our G-PGO algorithm in Table 1.

**Experimental results**

In this section, we present experimental results of G-PGO algorithm. Section “Evaluation metrics” describes the performance metrics to evaluate PGO results. Section “Data sets” describes data sets we use in our experiments. Section “Results on data sets” evaluates the performance of PGO cost function in batch PGO and the comparison between G-PGO and ISAM.

**Evaluation metrics**

We use three data sets to verify our algorithm: KITTI, TUM, and New College data sets. KITTI and TUM data sets all provide six degree of freedom (6-dof) true poses, but true poses are not aligned in TUM data set, so we only can use python script from TUM data set to evaluate performances. This script outputs two indexes to describe accuracy: “AME” and “RME.” “AME” means absolute pose error, and “RME” means relative pose error. Specific definitions for “AME” and “RME” can be found in Sturm et al.12

In KITTI data set, we use four metrics from Geiger et al.11 for our evaluation. The first two metrics are absolute orientation and translation errors. These two errors are defined below

$$\text{Error}^a = \sum_{i=1}^{n} \text{acos}(\text{max}(0.5 * (\Delta R_i(0,0) + \Delta R_i(1,1) + \Delta R_i(2,2)), 1), -1)) \frac{n}{n}$$

$$\text{Error}^a = \sum_{i=1}^{n} \text{norm}(|\Delta t_i|) \frac{n}{n}$$

(12)
Error$_{\text{R}}^i$ is the absolute rotational error. Error$_{\text{T}}^i$ is the absolute translational error. $\Delta R_i = R_i^\text{true} R_i$ is rotation matrix of the $i$th true robot pose, and $R_i$ is rotation matrix of the $i$th optimized robot pose. $n$ is the total number of poses in the world frame. $\Delta t_i = t_i^\text{true} - t_i$ is the translation of the $i$th optimized robot pose and $t_i$ is the translation of the $i$th true robot pose. Absolute errors are intuitive metrics for evaluating PGO performance because the bird-eye view trajectory map is drawn according to the absolute robot pose, so the absolute errors can reflect the trajectory map directly. But at the same time, the absolute errors cannot evaluate PGO results very well if the pose errors happen in the initial segment of trajectory which accompanied with subsequent poses. We also provide relative errors for evaluating errors. The relative errors are defined below

$$\Delta T_i, \text{true} = T_i^{-1} T_i^\text{true}$$

$$\Delta T_i = T_i^{-1} T_i^\text{true}(100)$$

$$\Delta T_{i, \text{error}} = \Delta T_{i}^{-1} \Delta T_{i, \text{true}} = \begin{bmatrix} \Delta R_{i, \text{error}} & \Delta t_{i, \text{error}} \\ 0 & 1 \end{bmatrix}$$

$T_i^\text{true}$ is the $i$th true robot pose which is obtained by global positioning system/inertial measurement unit (GPS/IMU). $T_i(100), \text{true}$ is the robot pose which is 100 m far from $T_i^\text{true}$. $T_i$ is the $i$th optimized robot pose, and $T_i(100)$ is the optimized robot pose which is 100 m from $T_i(100)$. After obtaining $\Delta R_{i, \text{error}}$ and $\Delta t_{i, \text{error}}$, (12) and (13) are used to calculate relative errors. The relative errors only take 100 m’ accumulative errors into consideration, so it can relatively evaluate the PGO approach much better than absolute ones.

New College data set only provides true translation and the true pose is also not aligned. So the classic iterative closest point (ICP) algorithm is used to align optimized poses using half of data. The metric we use in New College data set is absolute translation error. Specific definitions for this error can be found in Smith et al. We use New College and KITTI data sets to evaluate PGO initialization algorithm, which is mentioned in “Initialization techniques” section. We use TUM and KITTI data sets to evaluate batch PGO algorithm, which is mentioned in “PGO cost function” section. We use New College and TUM data sets to evaluate incremental PGO algorithm, which is which is shown in Figure 7. The true and VO poses of these data sets are plotted in Figure 7. The first figure in Figure 7 is the robot pose. Red color trajectory is the true trajectory and blue color is the trajectory of odometry. The second figure in Figure 7 is loop closure information in every data set. The x-axis is the index of robot pose and the y-axis is the start index and end index in every loop.

Results on data sets

First, we give the performance of initial algorithm for PGO. Second, we compare our batch PGO algorithm. Finally, we prove that G-PGO is accurate and robust in solving incremental PGO problem.

Initialization results. Initialization step is to guarantee that initial state of nonlinear optimization is close to global optimal solution. We compare initialization results between COP-SLAM and two-step initialization method. Here, we use KITTI and TUM data sets to evaluate initialization algorithm. New College data set cannot provide 6-dof true poses, so we do not use it here. Pittsburgh data sets provide 6-dof true pose, but VO poses and true poses are not in the same coordinate. So we just use absolute pose error to evaluate performances. We can see from Table 2 that absolute translation, absolute rotation, and relative rotation errors decrease after initialization algorithm. “Na” in Table 2 is the meaning of none value which represents data sets do not provide true value or optimization failure. Data set Pittsburgh_A, Pittsburgh_B, and Pittsburgh_C only provide true absolute translation values, so the values in 3 columns are “Na.” It can be seen from Table 2 that two-step initialization method leads to smaller absolute translation, absolute rotation, and relative translation errors compared with COP-SLAM algorithm. It proves that two-step initialization algorithm is more accurate than COP-
SLAM algorithm. But one thing to note is that relative translation error becomes larger after initialization algorithm, which is marked red in Table 2. The main reason is that two-step algorithm divides the optimization process into two steps. The first step is to optimize rotation matrix and fix the translation. The second step is to optimize translation but fix rotation matrix. The optimized variables of initialization technique are absolute poses in the world frame, so nonlinear optimization adjusts the absolute variables to close loops but the direction of adjustment is to decrease corresponded cost function not the whole cost function like batch PGO. We can see from (2) that rotation matrix affects translation error, and two-step optimized algorithm does not optimize rotation matrix and translation together. It is the drawback of initialization step and we use batch PGO algorithm to polish initialization results in the next step.

**Batch PGO results.** After initialization process, we use batch PGO to optimize all of robot poses in world frame. The batch PGO algorithm is suitable for off-line situation. Here, we use KITTI00, KITTI02, and TUM data sets to compare function.
different cost functions. Table 3 displays results of KITTI data sets and Table 4 displays results of TUM data sets. Values in steps column mean that how many steps are needed to finish optimization. “Na” in steps column represents optimization failure.

CERES in Tables 3 and 4 is corresponded with cost function (3). ORB is corresponded with cost function (4) and the Jacobian matrix is obtained by numerical method. STATE is corresponded with cost function (4) and the Jacobian matrix is obtained by left multiplication analytical deduction. GTSAM is corresponded with cost function (4) and the Jacobian matrix is obtained by right multiplication analytical deduction. Separated process (SP) is corresponded with cost function (5) and the Jacobian matrix is obtained by analytical deduction.

After batch optimization, relative translation error is smaller than odometry, and robot pose is closer to true solution. That is the reason why we need batch optimization after initialization process. We can see from Tables 3 and 4 that the convergence speed of CERES and ORB is very slow. The worst problem of CERES method is that it does not direct the optimization process to the global optimal solution which can be seen from KITTI02 and TUM-f1_room. The main reason of ORB’s bad performance is that ORB uses numerical Jacobian matrix to update poses in iterative process. We have talked about numerical Jacobian matrix before, and it is much less accurate than analytical Jacobian matrix. Another problem is that CERES uses automatic derivation; it costs more iterative steps than analytical Jacobian matrix. We can see from KITTI00, KITTI02, and TUM-f1 that CERES needs more than 50 steps to finish optimization which means that its time cost is very large. But we have to say that ORB SLAM code is not totally wrong. In ORB SLAM source code, the PGO algorithm deals with only one loop but our batch experiment includes more than loop closures. The PGO algorithm of ORB-SLAM source code performs not bad facing only one loop closure. Robot State Estimation (STATE), GTSAM, and SP PGO algorithms perform much better than CERES and ORB. The main reason is that these three algorithms use analytical Jacobian matrix and Lie algebra as cost function.

We can see from KITTI00, KITTI02, TUM-f2, and TUM-f3 data sets that SP algorithm is more accurate than CERES, ORB, and STATE algorithms. Another important superiority is that SP has the smallest optimized steps because of its cost function’s lower nonlinearity. Although steps of GTSAM in TUM-f3 and TUM-f2 are smaller than SP, SP algorithm is more accurate. According to our experiments, we choose SP model to optimize batch PGO in our G-GPO algorithm. The accuracy of final result is also dependent on the way you solve linear equation. We have to admit that different open source library such as G2O, GTSAM, and CERES uses different strategies to solve linear equation. But in our experiment, we just compare different cost function and open source library and choose the
Table 1. The flow diagram of G-PGO algorithm.

Notation

$L_k \in L = \{L_1, L_2, L_n\}$, $n$ is the total number of loop closure and $L$ is the set of loop closures. $L_k$ is the kth loop closure and is made up of robot poses $\{T_{i_{start}}; T_{i_{start}}; \cdots; T_{i_{end}}\}$. $i_{start}$ is the index of robot pose which is the first pose in the kth loop closure and $i_{end}$ is the index of robot pose which is the last pose in the kth loop closure.

$O^k = (\Delta T_{i_{start}}^{1}, \Delta T_{i_{start}}^{2}, \cdots, \Delta T_{i_{end}}^{n})$. $\Delta T_{i_{start}}$ is the relative pose between kth loop closure optimized robot pose $i_{start}$ and $i_{start} + 1$. $O^k$ is relative pose measurement from the kth loop closure. $O = (O_1, O_2, \cdots, O_n)$. $O$ is the set which consists of all of previous loop closures relative pose measurements.

Algorithm

$k = 0$; $O = \{\}$

While (Obtain new robot pose $T_i$ in the world frame)

if the new robot pose meets the loop closure

$O + 1$;

Optimize $L_k$ and obtain $O^k$;

Update $O = \{O^k\}$;

Evaluate loop closure metric and select which relative poses to be excluded from set $O$. After excluding outliers in set $O$, we optimize relative poses in set $O$ and obtain $\{\Delta T_{i_1}, \Delta T_{i_2}, \ldots, \Delta T_{i_{n-1}}\}$;

Use $\{\Delta T_{i_1}, \Delta T_{i_2}, \ldots, \Delta T_{i_{n-1}}\}$ to update absolute robot poses in the world frame ($T_1, T_2, \ldots, T_n$);

else

Do nothing;

end

Output optimized robot poses $\{T_1, T_2, \ldots, T_n\}$.

Table 2. The initialization technique result compared between COP-SLAM and two-step method. KITTI00, KITTI02, Pittsburgh_A, Pittsburgh_B, and Pittsburgh_C data sets are used to evaluate performances. Four metrics in columns are corresponded with (12)-(16).

| Data sets       | Method | Error$^a_t$ | Error$^a_r$ | Error$^r_t$ | Error$^r_r$ |
|-----------------|--------|-------------|-------------|-------------|-------------|
| KITTI00         | VO     | 52.754      | 0.234       | 1.708       | 0.0352      |
|                 | COP    | 11.590      | 0.0913      | 4.094       | 0.0347      |
|                 | Two-step | 10.693      | 0.0776      | 3.526       | 0.0316      |
|                 | VO     | 78.367      | 0.247       | 1.540       | 0.0233      |
| Pittsburgh_A    | VO     | 29.05       | Na          | Na          | Na          |
|                 | COP    | 19.606      | Na          | Na          | Na          |
|                 | Two-step | 19.592      | Na          | Na          | Na          |
| Pittsburgh_B    | VO     | 237.551     | Na          | Na          | Na          |
|                 | COP    | 97.922      | Na          | Na          | Na          |
|                 | Two-step | 83.890      | Na          | Na          | Na          |
| Pittsburgh_C    | VO     | 418.323     | Na          | Na          | Na          |
|                 | COP    | 173.986     | Na          | Na          | Na          |
|                 | Two-step | 77.642      | Na          | Na          | Na          |

COP: closed-form PGO; SLAM: simultaneous localization and mapping; VO: visual odometry; PGO: pose graph optimization

Table 3. The batch optimization performances between CERES, ORB, Robot State Estimation, and our method using KITTI data set. We set the max iteration step 50 and if the iteration step is larger than 50, the iteration will stop and output results.

| Data sets       | Method | AME  | RME  | Steps |
|-----------------|--------|------|------|-------|
| KITTI00         | VO     | 0.07049 | 0.09282 | NA    |
|                 | CERES  | 0.0867 | 0.114 | 50    |
|                 | ORB    | 1.438  | 2.210 | 35    |
|                 | STATE  | 0.0508 | 0.0788 | 18    |
|                 | GTSAM  | 0.05082 | 0.07884 | 4     |
|                 | SP     | 0.05082 | 0.07884 | 6     |
| TUM-f1_room     | VO     | 0.01067 | 0.0304 | NA    |
|                 | CERES  | 0.01011 | 0.0304 | 11    |
|                 | ORB    | 0.4724 | 0.6015 | 24    |
|                 | STATE  | 0.01011 | 0.03038 | 30    |
|                 | GTSAM  | 0.01024 | 0.03085 | 2     |
|                 | SP     | 0.01011 | 0.03038 | 5     |
| TUM-f2_desk     | VO     | 0.01014 | 0.02304 | NA    |
|                 | CERES  | 0.009424 | 0.02278 | 11    |
|                 | ORB    | 1.6131 | 2.2935 | 28    |
|                 | STATE  | 0.009424 | 0.02278 | 6     |
|                 | GTSAM  | 0.009378 | 0.02317 | 2     |
|                 | SP     | 0.009424 | 0.02279 | 5     |

AME: absolute pose error; RME: relative pose error; VO: visual odometry; GTSAM: Georgia tech smoothing and mapping.

Table 4. The batch optimization performances between CERES, ORB, Robot State Estimation, and Our method using TUM sequence data set. We set the max iteration step 50 and if the iteration step is larger than 50, the iteration will stop and output results.

| Data sets       | Method | AME  | RME  | Steps |
|-----------------|--------|------|------|-------|
| TUM-f1_room     | VO     | 0.07049 | 0.09282 | NA    |
|                 | CERES  | 0.0867 | 0.114 | 50    |
|                 | ORB    | 1.438  | 2.210 | 35    |
|                 | STATE  | 0.0508 | 0.0788 | 18    |
|                 | GTSAM  | 0.05082 | 0.07884 | 4     |
|                 | SP     | 0.05082 | 0.07884 | 6     |
| TUM-f2_desk     | VO     | 0.01067 | 0.0304 | NA    |
|                 | CERES  | 0.01011 | 0.0304 | 11    |
|                 | ORB    | 0.4724 | 0.6015 | 24    |
|                 | STATE  | 0.01011 | 0.03038 | 30    |
|                 | GTSAM  | 0.01024 | 0.03085 | 2     |
|                 | SP     | 0.01011 | 0.03038 | 5     |
| TUM-f3_longoffice | VO    | 0.01014 | 0.02304 | NA    |
|                 | CERES  | 0.009424 | 0.02278 | 11    |
|                 | ORB    | 1.6131 | 2.2935 | 28    |
|                 | STATE  | 0.009424 | 0.02278 | 6     |
|                 | GTSAM  | 0.009378 | 0.02317 | 2     |
|                 | SP     | 0.009424 | 0.02279 | 5     |

AME: absolute pose error; RME: relative pose error; VO: visual odometry; GTSAM: Georgia tech smoothing and mapping.

best performance one which is used in our G-PGO algorithm

Incremental PGO results. The last experiment is incremental experiment. We use KITTI00, KITTI02, and New College data sets to verify G-PGO algorithm. TUM data set only contains one-loop closure, so we do not use it here. True poses are aligned in KITTI data set, but true poses are not aligned in New College data set. When dealing with New College data set, we use half of true poses and optimized poses to align two coordinates. Classic ICP algorithm is
used here to align two coordinates. After aligning two coordinates, we use absolute translation error, which is described in the fifth section, to evaluate incremental PGO algorithms. In Table 5, we divide experimental results into two classes. The first class is noise free data which means that wrong loop closure information is not added into input data. Results of noise free data is listed in the second and third row of Table 5. Another class is noisy input data. We add wrong loop closure information in data, and wrong loop closure information in every data set is shown in red color in Figure 9. The last two rows in Table 5 are results of noisy data.

Table 5. The experimental results of incremental PGO algorithm. Sequential PGO method is the Schur way of ORB-SLAM source code executing PGO. The unit of every element in table is meter.

| Data Set | ISAM | KITTI00 | G-PGO | Pitts_A | Pitts_B | Pitts_C |
|----------|------|---------|-------|---------|---------|---------|
| ISAM     | 5.30 | 78.37  | 20.86 | 77.99   | NA      |         |
| G-PGO    | 4.72 | 64.15  | 17.66 | 74.41   | 85.56   |         |
| Outlier-ISAM | 31.89 | 55.07  | 38.29 | 237.55  | NA      |         |
| Outlier-PGO | 4.72 | 64.15  | 17.66 | 74.41   | 85.563  |         |

PGO: pose graph optimization; SLAM: simultaneous localization and mapping; ISAM: incremental smoothing and mapping.

Figure 8. The red trajectory is true robot pose and the blue trajectory is G-PGO result. KITTI00 sequence is shown in (a) and KITTI02 sequence is shown in (b). Pittsburgh_A sequence from New College data set is shown in (c). Pittsburgh_B sequence from New College data set is shown in (d). Pittsburgh_C sequence from New College data set is shown in (e). PGO: pose graph optimization.
promise to converge to global best solution. Our algorithm makes full use of every loop closure’s information and produces more accurate results. The optimized robot trajectories produced by G-PGO are shown in Figure 8.

We can see from Table 5 that G-PGO performs better than classic incremental PGO algorithm especially in the outlier free case. ISAM algorithm directs iterative process to wrong direction and ATE becomes even larger after optimization compared with odometry result. G-PGO uses constraint metric to exclude wrong loop closure information and produces the same result with noise free data. In Figure 9, we display wrong loop closure information. The Y-axis of Figure 9 is constraint metric which is corresponded with (6). The X-axis of Figure 9 is corresponded with loop closure in every data set. The label of X-axis is robot pose index, which leads to a loop closure. Red bar is wrong loop closure information we add. We can see from Figure 9 that constraint metric of wrong loop closure is much larger than correct ones. So wrong loop closure can be easily deleted from input data.

Conclusions
In this article, we review the previous PGO algorithms that include initialization algorithm and cost function. We compare two-step initialization algorithm and COP-SLAM in large real scenes. We find that two-step initialization algorithm performs better than COP-SLAM. Two-step algorithm estimates the rotation matrix without taking translation into consideration and thenes fix the optimized rotation matrix to solve linear equation to obtain translation of robot pose. This step guarantees that the initial robot poses are not far from the global optimal solution. After initialization process, we compare different cost functions used in PGO. We find that SP model is more accurate than others. In this model, the estimated variables are so(3) and translation. The benefit of this cost function is to decrease the nonlinearity of cost function compared with others. At last, to bypass the Schur complement and consistency problem of incremental PGO, we put forward a new incremental PGO algorithm named G-PGO. G-PGO does not optimize the absolute robot pose in the world but the estimated variables are relative robot poses. Every loop closure performance is introduced into the G-PGO cost function as weight matrix to balance measured relative poses. Another superiority is that G-GPO algorithm can delete wrong loop closure information using constraint metric automatically. We compare G-PGO with ISAM algorithm. G-PGO performs much better than ISAM result and G-PGO is also more robust to outliers than ISAM. In our future work, we will focus on observability of PGO algorithm and to testify what kind of loop closure can obtain accurate results according to their observability.

Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by the national natural science foundation of china (Grant ID-61601123), national basic research program of
china (973 program) (Grant ID-2016YFB0502103) and natural science foundation of jiangsu province (Grant ID-BK20160696).

**ORCID iD**

Feng Youyang [https://orcid.org/0000-0002-5980-4164](https://orcid.org/0000-0002-5980-4164)

**References**

1. Durrant-Whyte H and Bailey T. Simultaneous localization and mapping: part I. *IEEE Robot Autom Mag* 2006; 13(2): 99–110.
2. Bailey T and Durrant-Whyte H. Simultaneous localization and mapping (SLAM): part II. *IEEE Robot Autom Mag* 2006; 13(3): 108–117.
3. Demim F, Nemra A, and Louadj K. Robust SVSF-SLAM for unmanned vehicle in unknown environment. *IFAC-Papers OnLine* 2016; 49(21): 386–394.
4. Fethi D, Nemra A, Louadj K, et al. Simultaneous localization, mapping, and path planning for unmanned vehicle using optimal control. *Adv Mech Eng* 2018; 10(1): 1687814017736653.
5. Demim F, Nemra A, Louadj K, et al. Visual SVSF-SLAM algorithm based on adaptive boundary layer width. In: *International conference on electrical engineering and control applications*, 2017, pp. 97–112. Cham: Springer.
6. Demim F, Boucheilouk A, Nemra A, et al. A new adaptive smooth variable structure filter SLAM algorithm for unmanned vehicle. In: 2017 6th international conference on systems and control (ICSC), Batna, Algeria, 7–9 May 2017, pp. 6–13. IEEE.
7. Dong-Si TC and Mourikis AI. Motion tracking with fixed-lag smoothing: algorithm and consistency analysis. In: *2011 IEEE international conference on robotics and automation*, Shanghai, China, 9–13 May 2011, pp. 5655–5662. IEEE.
8. Ekenhoff K, Paull L, and Huang G. Decoupled, consistent node removal and edge sparsification for graph-based SLAM. In: *2016 IEEE/RSJ international conference on intelligent robots and systems* (IROS), Daejeon, South Korea, 9–14 October, 2016, pp. 3275–3282. IEEE.
9. Kaess M, Johannsson H, Roberts R, et al. iSAM2: incremental smoothing and mapping using the Bayes tree. *Int J Rob Res* 2012; 31(2): 216–235.
10. Ilia V, Polok L, Solomon M, et al. SLAM+—a highly efficient and temporally scalable incremental SLAM framework. *Int J Rob Res* 2017; 36(2): 210–230.
11. Geiger A, Lenz P, Stiller C, et al. Vision meets robotics: the KITTI dataset. *Int J Rob Res* 2013; 32(11): 1231–1237.
12. Sturm J, Engelhard N, Endres F, et al. A benchmark for the evaluation of RGB-D SLAM systems. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura, Portugal, 7 October, 2012–12 October, 2012, pp. 573–580. IEEE.
13. Smith M, Baldwin I, Churchill W, et al. The new college vision and laser data set. *Int J Rob Res* 2009; 28(5): 595–599.
14. Ceres Solver, http://ceres-solver.org (2015, accessed 15 December 2019).
15. GTSAM, https://bitbucket.org/gtborg/gtsam (2018, accessed 15 December 2019).
16. Rosen DM, Carlone L, Bandeira AS, et al. SE-Sync: a certifiably correct algorithm for synchronization over the special Euclidean group. *Int J Rob Res* 2016; 38(2-3): 95–125.
17. Nasiri SM, Hosseini R, and Moradi H. A recursive least square method for 3D pose graph optimization problem. arXiv preprint arXiv:1806.00281, 2018.
18. Bai F, Vidal-Calleja T, and Huang S. Robust incremental SLAM under constrained optimization formulation. *IEEE Robot Autom Lett* 2018; 3(2): 1207–1214.
19. Khosoussi K, Huang S, and Dissanyake G. A sparse separable SLAM back-end. *IEEE Trans Robot* 2016; 32(6): 1536–1549.
20. Bai F, Vidal-Calleja T, Huang S, et al. Predicting objective function change in pose-graph optimization. In: *2018 IEEE/RSJ international conference on intelligent robots and systems* (IROS), Madrid, Spain, 1–5 October, 2018, pp. 145–152. IEEE.
21. Khosoussi K, Giamou M, Sukhatme GS, et al. Reliable graphs for slam. *Int J Rob Res* 2019; 38: 260–298.
22. Lee GH, Fraundorfer F, and Pollefeys M. Robust pose-graph loop-closures with expectation-maximization. In: *2013 IEEE/RSJ international conference on intelligent robots and systems*, Tokyo, Japan, 3–7 November 2013, pp. 556–563. IEEE.
23. Latif Y, Cadena C, and Neira J. Robust loop closing over time for pose graph SLAM. *Int J Rob Res* 2013; 32(14): 1611–1626.
24. Dubbelman G and Browning B. COP-SLAM: closed-form online pose-chain optimization for visual SLAM. *IEEE Trans Robot* 2015; 31(5): 1194–1213.
25. Carlone L, Tron R, Daniilidis K, et al. Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization. In: *2015 IEEE international conference on robotics and automation* (ICRA), Seattle, WA, USA, 25 May 2015, pp. 4597–4604. IEEE.
26. Carlone L, Aragues R, Castellanos JA, et al. A fast and accurate approximation for planar pose graph optimization. *Int J Rob Res* 2014; 33(7): 965–987.
27. Carlone L, Calafiore GC, Tommoliello C, et al. Planar Pose Graph Optimization: Duality, Optimal Solutions, and Verification. *IEEE Transactions on Robotics* 2016; 32(3): 545–565.
28. Carlone L, Rosen DM, Calafiore G, et al. Lagrangian duality in 3D SLAM: verification techniques and optimal solutions. In: *2015 IEEE/RSJ international conference on intelligent robots and systems* (IROS), Hamburg, Germany, 28 September–3 October 2015, pp. 125–132. IEEE.
29. Carlone L, Aragues R, Castellanos JA, et al. A linear approximation for graph-based simultaneous localization and mapping. *Int J Rob Res* 2004; 77(7): 41–48.
30. Kümmerle R, Grisetti G, Strasdat H, et al. G2o: A general framework for graph optimization. In: *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, 9–13 May 2011, pp. 3607–3613. IEEE.
31. Barfoot TD. *State estimation for robotics*. Cambridge: Cambridge University Press, 2017.
32. Mur-Artal R, Montiel JMM, and Tardos JD. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans Robot* 2015; 31(5): 1147–1163.
33. Kümmerle R, Steder B, Dornhege C, et al. On measuring the accuracy of SLAM algorithms. *Auton Robot* 2009; 27(4): 387.
34. Vertigo, https://github.com/OpenSLAM-org/openslam_vertigo (2013, accessed 15 December 2019).
35. Sünderhauf N and Protzel P. Switchable constraints for robust pose graph SLAM. In: *2012 IEEE/RSJ international conference on intelligent robots and systems*, Vilamoura, Portugal, 7–12 October 2012, pp. 1879–1884. IEEE.
36. Geiger A, Ziegler J, and Stiller C. Stereoscan: dense 3d reconstruction in real-time. In: *2011 IEEE intelligent vehicles symposium (IV)*, Baden-Baden, Germany, 5–9 June 2011, pp. 963–968. IEEE.
37. Labbé M and Michaud F. Memory management for real-time appearance-based loop closure detection. In: *2011 IEEE/RSJ international conference on intelligent robots and systems*, San Francisco, CA, USA, 25–30 September 2011, pp. 1271–1276. IEEE.
38. Horn BK. Closed-form solution of absolute orientation using unit quaternions. *J Opt Soc Amer A* 1987; 4(4): 629–642.