Spare Pose Graph Decomposition and Optimization for SLAM

Tian Liu, *Yongfu* Chen , *Zhiyong* Jin, *Kai* Li, *Zhenting* Wang, *Jiongzhi* Zheng

National CAD Support Software Engineering Research Center, Huazhong University of Science &Technology, Wuhan 430074, China

Abstract. The graph optimization has become the mainstream technology to solve the problems of SLAM (simultaneous localization and mapping). The pose graph in the graph based SLAM is consisted with a series of nodes and edges that connect the adjacent or related poses. With the widespread use of mobile robots, the scale of pose graph has rapidly increased. Therefore, optimizing a large-scale pose graph is the bottleneck of application of graph based SLAM. In this paper, we propose an optimization method basing on the decomposition of pose graph, of which we have noticed the sparsity. With the extraction of the Single-chain and the Parallel-chain, the pose graph is decomposed into many small subgraphs. Compared with directly processing the original graph, the speed of calculation is accelerated by separately optimizing the subgraph, which is because the computational complexity is increasing exponentially with the increase of the graph's scale. This method we proposed is very suitable for the current multi-threaded framework adopted in the mainstream SLAM, which separately calculate the subgraph decomposed by our method, rather than the original optimization requiring a large block of time in once may cause CPU obstruction. At the end of the paper, our algorithm is validated with the open source dataset of the mobile robot, of which the result illustrates our algorithm can reduce the one-time resource consumption and the time consumption of the calculation with the same map-constructing accuracy.

1 Introduction

In order to make the mobile robot in unknown environment automatic movement, the robot need to construct environment model according to their own perception of environmental, so as to synchronous localization. Synchronous localization and mapping (SLAM) of mobile robot has received much attention in recent years. But the difficulty of the problem is the location and the mapping depend on each other. While the localization of robot depends on the map that has been constructed, but the mapping step relies on precise positioning coordinates to montage. So the simultaneous completion of mapping and location is the biggest challenge of SALM algorithm.

With the error of the robot sensor, the robot can't reach the predetermined target point according to the program order. Therefore, it is necessary to fuse the data of the controller and the observation sensor to eliminate the error of any single data source. The fundamental method of data fusion is based on a Bayes filter, which assume the current state is just influenced by the previous one. Each state is calculated to be a probability of error by Bayes filter. With the accumulation of errors, the accuracy of position estimation is getting worse and worse. When the scale of the environment is small, such error can be accepted. However, when the environment scale is increasing, the filtering method can hardly meet the needs of practical application.

In recent years, with the increasing of the environment scale, graph SLAM is widely concerned by academia and industry because it can solve global consistency solution of SLAM. The method we studied in this paper belongs to the branch of graph optimization. The pose series of robot is represented as a graph structure, in which the node represents the robot states and the edge represents the correlation of the different state. At the same time, according to the similarity match between the observation data and the historical observation data, an edge of loop closing will be inserted. The values of the loop edge are calculated according to the coincidence of scan-matching. After linearization, the graph can be calculated by transforming into a nonlinear least squares problem.

Graph SALM has great advantages in calculating global consistency solutions. However, due to the large amount of data in global optimization, graph optimization can only be performed at the back end of SLAM, which can't meet the need of practical application.

Therefore, the method based on the decomposition of pose graph's sparsity is proposed in this paper, which accelerates the calculation speed of graph optimization. The equivalent substitution of the Single-chain and the Parallel-chain is the core method to decompose the pose graph into subgraph, so as to reduced that the scale of the information matrix of the overall graph. Then the optimization of the graph is simplified into processing many subgraphs, rather than the overall graph. So the calculation complexity is reduced greatly.

2 Related work

SLAM is the combination of three problems: sensor data processing, data fusion and state estimation. The sensor data processing relies heavily on practical applications. The different sensor, such as LIDAR, monocular, binocular and so on, will use different data processing method. But the data fusion and the state estimation are often combined together to calculate. One example is the extended Kalman filter algorithm[1], which take the previous state to inference the present state while fusing the previous probability and the motion control. The filter based algorithm has maturely developed, such as EKF-SLAM[1], UKF-SLAM[2] and so on. When used for processing the nonlinear system, the linear approximation is obtained by the first order Taylor expansion of the nonlinear estimation.

Because the state estimation of the filtering method is based on the Markov inference, the error is accumulated in the series of states When the state sequence is further enlarged, the covariance of the state is gradually diverged. Therefore, the algorithm of filtering is just suitable in small scale environment and be inadequate in large-scale environment.

However, the environment's scale of mobile robot is gradually expanding, and the filtering algorithm becomes more and more difficult to fulfil the state estimation task of SLAM. Until Lu and Milios[3] put forward to use graph to represent the states of SLAM, the results of global consistency optimization were based on the graph. But this framework is not taken seriously at first, because the scale of the graph increases linearly with time, and the optimization of the graph takes a larger number of time. Subsequently, Davis and Timothy[4] found the sparsity of SLAM as a graph, and solved the graph with sparse linear algebra, which greatly shortened the calculation time.

Since then, a large number of algorithms derived from graph optimization have emerged. Howard[5] proposed the relaxation iteration based methods. Duckett[6] presented that Gauss Seidel relaxation to minimize the errors of the graph. Frese[7] put forward a multi-level relaxation iterative algorithm based on the Gauss Seidel algorithm. Olson[8] use the gradient descent method to solve the problem. Grisetti[9] presented a tree structure based on gradient descent method, by which the convergence speed of iteration was greatly accelerated. Konolige[10] adapt the conjugate gradient method, which is very efficient in solving large sparse pose constraints. So far, graph optimization has become the mainstream trend in SLAM.

Combining the loop detection algorithm to provide the constraints between the non-direct connected poses, the graph optimization can minimize the global error. Graph optimization is the most advanced tool to solve the current global optimization problem in SLAM. But the huge weakness of graph optimization is the speed of which can't meet the requirement of real-time computation. The mainstream SLAM framework, such as ORB-SLAM[11], [12] and PTAM[13], uses the front-end and back-end structure to deal with such kind of the problem, where the front-end process the sensor data fusion and the state estimation, and the back-end uses the G2O library [14] to calculate the global consistent solution.

Therefore, we focus on the decomposition of the graph's information matrix, so as to extract the Single-chain and the Parallel-chain according to its geometric features. In this way, the graph is decomposed into many small subgraphs which can be optimized quickly.

3 Problem description

The graph based SLAM use the node and edge to represent the robot trajectory and the constraint between pose and landmark, where the landmark can be extract from the camera's visual feature or the special object. Then the raw measurement provides the edge between pose and landmark. Each edge is consisted by the distance vector and the association probability.

In this paper, the back-end optimization is just focused, which assume the data association and the probability distribution is known from the front-end. So the task of the graph SLAM is to optimize the pose graph closing to the reality, as Eq. (1).

$$F_{(x)} = \sum_{\langle ij\rangle \in C} \mathrm{e}(x_i, x_3 \cdot 3^2 ij)^T \Omega_{ij} \mathrm{e}\left(x_i, x_j, z_{ij}\right) \quad (1)$$

$$x = arg\,\min_x F(x) \quad (2)$$

Let $x = (X_1, X_2 \ldots X_n)$ describe the set of the pose, which is calculated from the motion model. Let Z_{ij} and Ω_{ij} represent the means and the information matrix of the measurement between the node i and j. This measurement is calculated by the overlap of the observation from node i and j.

From the derivation in[15], the final problem is defined as a minimum problem as and its simplification is shown as

$$H\Delta x^* = -b, \quad (3)$$

where the information matrix H and the information vector b are derived from the linearized of the system.

The linearized solution is then obtained by adding the computed increment on the initial guess.

$$x^* = x + \Delta x^* \quad (4)$$

According to the previous description, the problem of Graph SLAM has been simplified to two steps. Firstly, the information matrix H and the information vector b are constructed from the control u and the measurement z, then the linear equations are solved to get Δx^*. With the increasing of trajectory, the increment of H and b will be linearly, and to optimize such a large matrix is too expensive. So we proposed the spare pose graph decomposition to accelerate this process.

4 Pose graph decomposition

4.1 Procedure outline

The spare pose graph decomposition proposed in this paper starts with analysing the geometric characteristics of the robot's pose graph, of which the sparsity can be

used to decompose the graph to accelerate the calculation by extracting and replacing the pose graph into small subgraph. Such kind of the decomposition has two advantages. On the one hand, the computational complexity of matrix inversion in Eq.(3) increases with the matrix dimension by $O(N^3)$ .But while it was decomposed into small matrices, the computational complexity becomes $O(n_1^3 + n_2^3 + n_3^3 \ldots)$. On the other hand, the most important purpose of the overall graph optimization is to make the robot get a more accurate environment map of the current position so that it can complete the task more accurately. The subgraphs can be optimized in order of priority. The subgraph where the robot current in will be firstly optimized, while the optimization tasks of other subgraphs are optimized in subsequent free time. This decomposition and optimization method is also suitable for parallel computing.

The core process of the algorithm is displayed in Algorithm 1. The control parameter $u_{1:t}$ is inputted into the motion model to get the initialized trajectory pose v in Line 1. And the measurement $z_{1:t}$ of the co-visual poses is inputted in the measurement function to get the edge e in Line 2, of which the observation distance and the confidence probability are consisted. As described in Line 3, the adjacency matrix B is composed by the node number of the constraint relationships in e, where the correlation between the poses is reflected. And B will be used for the subsequent extraction process of the Single-chain and the Parallel-chain.

Algorithm 1 $DecomposeGraphSlam(u_{1:t}, z_{1:t})$:
$v = MotionModel(u)$
$e = MeasureModel(u, z)$
build the adjoin matrix B with e
$SingleSet = ExtractSingleChain(B, e)$;
$ParallelSET = ExtractParallelChain(B, e)$;
$e = e - SingleSet_{eids} - ParallelSet_{eids} + SingleSet_{eid:equal} - ParallelSet_{eid:equal}$
$vmenas = GlobalOptimization$
$vmeans = LocalLoopOptimize$
Return $vmeans$

The region passed once in the real world is represented by the Single-chain $\{s = [v, e, v_{equal}] | s \in Sc\}$, which is extracted from e in Algorithm 1. Line 4. And the region passed through many times in a single direction is represented by the Parallel-chain $\{p = [v, e, v_{equal}] | p \in Pc\}$,which is extracted in Algorithm 1. Line 5. The equivalent edge and the equivalent quadrangle are extracted in Algorithm 1. Line 6:7, after which all edges of Single-chain are replaced by an equivalent edge and all edges of the Parallel-chain are replaced by an equivalent quadrilateral. In the equivalent quadrilateral, the main opposite edge of the quadrilateral is used to represents the two Parallel-chains, and the second opposite edges represents the constraint between the beginning and ending nodes. The scale of the overall pose map after completing replacement is greatly reduced.

The correlation of the graph's nodes after the replacement of the Single-chain and the Parallel-chain is very dense, which can be optimized directly. The overall consistence solution of the remaining graph is solved in Algorithm 1. Line 8.

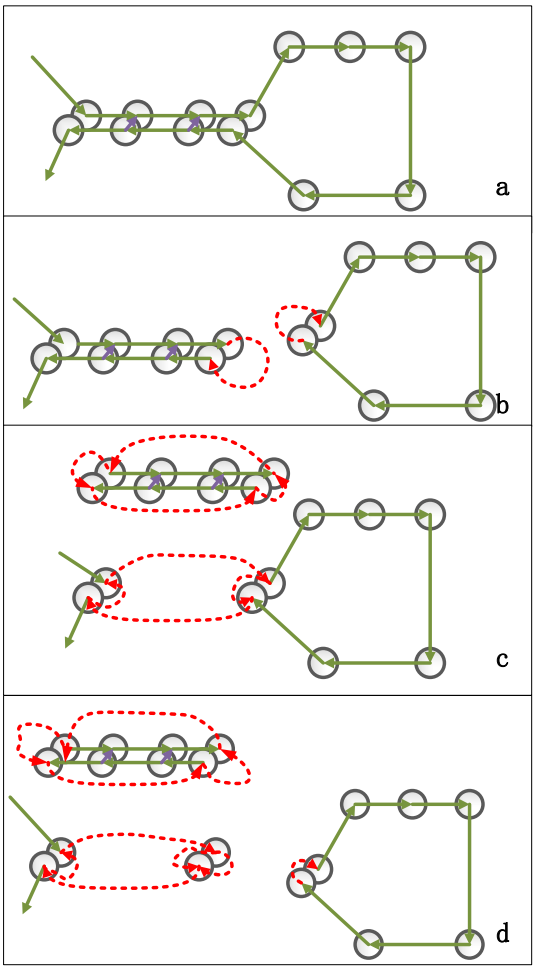

Fig. 1. Extraction process.
a. The original graph;
b. The remaining graph and the separated Single-chain
c. The remaining graph and the separated Parallel-chain
d. The remaining graph and the separated subgraph

After the overall graph's optimization is completed, the optimized equivalent edge will be used for the subgraph's optimization. In the local optimization of Single-chain and Parallel-chain, the original edges and the equivalent edges are combined into a subgraph, in which the covariance of the equivalent edge is set as **0**. The equivalent edges and equivalent quadrilaterals of the remaining graph is replaced by the corresponding Single-chain and Parallel-chain optimized. Finally, the overall trajectory with global consistency is obtained.

4.2 Extraction and replacement

4.2.1 Extraction of single-chain

In the pose graph of the robot, many parts trajectory are composed of node that is only connected to the previous one and the next one of the current node. Such a trajectory is named as "Single-chain". For the rest of the graph, the beginning and ending nodes of the

Single-chain are only related, as shown in Figure 1.a. Therefore, an edge named as "Equivalent edge" is used to describe the effect of the Single-chain on the remaining graph in this paper.

Algorithm 2 $ExtractSingleChain(B, e)$:

$I = \{i | Row_i - index(sum(B\ by\ row) == 3)\}$;

for each element in I:

if $I(a:b)$ is continue && $b - a > 5$:

pushback(SC_series,$I(a:b)$)

for each element in SC_series :

$$e' = \begin{Bmatrix} e_{id}^{a:b} \\ e_{mean}^{a:b} \\ e_{inf}^{a:b} \end{Bmatrix}$$

$$e_{equal;id} = \begin{Bmatrix} e_{id}^{b} \\ e_{id}^{a} \end{Bmatrix}$$

$$e_{equal;means} = t2v\left\{ v2t\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} * \prod_a^b v2t(e_{mean}^i) \right\}$$

$$e_{equal;inf} = \frac{1}{\sum_a^b \frac{1}{e_{inf}^i}};$$

add $\begin{Bmatrix} e' \\ e_{equal} \end{Bmatrix}$ to $\{SingleSet\}$

Return $\{SingleSet\}$

Single-chain is a region where the robot has only travelled once in the physical world and there are no loop-closing edges. Although the pose structure in the Single-chain is very simple, it accounts for a large proportion of the overall pose graph, whose equivalent replacement can significantly reduce the calculation complexity of the overall graph optimization. The extraction process of the Single-chain is described in Algorithm 2.

In the Single-chain's extraction algorithm, the adjacency matrix previously constructed will be used for filtering. The connection of one node to another is represented as a row in the adjacency matrix. The current point is a single point only when the column number of the current pose cell and its previous and next positions cells are non-zero in one row of the adjacency matrix. Among all the single points selected, one series with more than five adjacent single points are constructed as a Single-chain after filtering again. Add The information $\left[e_{id}^{a:b} \quad e_{mean}^{a:b} \quad e_{inf}^{a:b}\right]$ of the edge contained in the Single-chain is added to e'.

After the information of the Single-chain is extracted, the Single-chain needs to be equivalent its influence in the overall graph, which is provided by calculating the equivalent edge. The initial poses of the equivalent edge's two endpoints, the observation of the equivalent edge and the information matrix of the equivalent edge are included in the equivalent information.

The initial poses of the equivalent edge's two endpoints can be obtained directly from the beginning and ending nodes of the Single-chain, and the corresponding index $v_{e_{equal;id}}$ is searched from the set v.

The value of the equivalent edge is derived from all poses contained in the Single-chain. To facilitate vector operation, two reciprocal operators, $v2t(x)$ and $t2v(x)$, are defined as Eq. (5) and Eq. (6).

$$v2t\begin{pmatrix} x \\ y \\ \theta \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

$$t2v\begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{6}$$

The starting node of the Single-chain is set to a rigid constraint $[0 \quad 0 \quad 0]^T$, which guarantees that the overall graph does not diverge during the optimization process. The equivalent edge's value is solved by convert the edge vector into a matrix, and then multiply it by $\prod_a^b v2t(e_{mean}^i)$.

The information matrix of the equivalent edge is derived from all edges' in the Single-chain. Since the information matrix is the inverse of the covariance matrix and the covariance of the equivalent edges is the sum $\sum_i \varepsilon_i$ of the covariance of all edges, the final equivalent information matrix is

$$e_{equal;inf} = \frac{1}{\sum_a^b \frac{1}{e_{inf}^i}}. \tag{7}$$

Finally, the edge's information contained in each Single-chain and its equivalent edge information $\left[e' \quad e_{equal}\right]$ are stored into the {SingleSet} of the collection.

4.2.2 Extraction of parallel-chain

Parallel-chains are equivalent substitution to areas such as narrow corridors and aisles in the real world. Because these areas are long and narrow, the co-visual relationship of nodes is very regular. Attention, the parallelism mentioned in this paper is not the parallelism in the physical world, but the two trajectory are aligned and co-visible with each other. In this paper, the Parallel-chain will be replaced by an equivalent quadrilateral to reduce the scale of the information matrix. The core feature of the Parallel-chain is that it contains two pose series that is coherent or opposite, which is extracted by identifying such kind of the feature in pose graph.

The extraction of the Parallel-chain is shown in Algorithm 3, which is consisted with the extraction part and equivalence part.

In the extraction process, if there are other associated points in one row of the adjacency matrix except for the previous one and the next one, the current point is a Parallel-point, which will be extracted out totally by traversing each row.

Whether the Parallel-points can be consisted in a trajectory series and the number of the series is bigger than a fixed tolerance is the judgment of this series can be selected as a candidate. Then the relationship with the relative edge is judged by $RelativeJudge(B_{i,:})$. If the non-adjacency edges of all nodes except the first and last nodes in the two series can be mapped into others, the requirements that the two series can be consisted as a Parallel-chain are met and this Parallel-chain will be added to P.

In the equivalent process of the Parallel-chain, the information of the edge containing the Parallel-chain is

extracted according to the extracted node number of the Parallel-chain, and those info will be stored into $\begin{Bmatrix} \begin{pmatrix} e_{id}^{Bc_{a:b}} & e_{id}^{Rc_{a:b}} \end{pmatrix} \\ \begin{pmatrix} e_{mean}^{Bc_{a:b}} & e_{mean}^{Rc_{a:b}} \end{pmatrix} \\ \begin{pmatrix} e_{inf}^{Bc_{a:b}} & e_{inf}^{Rc_{a:b}} \end{pmatrix} \end{Bmatrix}$.

Algorithm 3 ExtractParallelChain(B, e):

$Bc = \varphi, Rc = \varphi$
$P = \varphi$
for each row in B:
  if $\sum B_{i,:} > 4$ && $RelativeJudge(B_{i,:})$:
    if $Bc = \varphi, Rc = \varphi$:
      initial Bc with $B_{i,:}$
      initial Rc with $B_{i,:}$ relative node
    else:
      update Bc with row
      update Rc with row's relative node
    if $Bc_b - Bc_a > minPts$:
      push back $\begin{pmatrix} \mathrm{Bc} \\ Rc \end{pmatrix}$ to P
    $Bc = \varphi,\ Rc = \varphi$
for each element in P:
  $e' = \begin{Bmatrix} \begin{pmatrix} e_{id}^{Bc_{a:b}} & e_{id}^{Rc_{a:b}} \end{pmatrix} \\ \begin{pmatrix} e_{mean}^{Bc_{a:b}} & e_{mean}^{Rc_{a:b}} \end{pmatrix} \\ \begin{pmatrix} e_{inf}^{Bc_{a:b}} & e_{inf}^{Rc_{a:b}} \end{pmatrix} \end{Bmatrix}$
  $e_{equal;id} = \begin{Bmatrix} e_{id}^{Bc_b} & e_{id}^{Rc_b} & e_{id}^{Bc_b} & e_{id}^{Rc_b} \\ e_{id}^{Bc_a} & e_{id}^{Rc_a} & e_{id}^{Rc_a} & e_{id}^{Bc_a} \end{Bmatrix}$
  $V' = OptimizeAndSolve(e', v)$
  $\mathrm{e}_{equal;means} = V_{e_{equal;id}}$
  $\mathrm{e}_{equal;inf} = \begin{Bmatrix} \frac{1}{\sum_a^b \frac{1}{e_{inf}^i}} & e_{inf}^i & \frac{1}{\sum_a^b \frac{1}{e_{inf}^i}} & e_{inf}^i \end{Bmatrix}$
  add $\begin{Bmatrix} e' \\ e_{equal} \end{Bmatrix}$ to {ParallelSET}
Return $\{ParallelSET\}$

The four nodes values of the equivalent quadrilateral are derived from the state and endpoints of the Parallel-chain's Single-chains. The sum of the covariance of all the edges in each Single-chain is taken as the value of the main opposite edge of the equivalent quadrilateral. The Parallel-chain's first endpoint constitutes the value of one second pair edges of the equivalent quadrilateral, and the other is consisted of last endpoint constraints, as $\begin{Bmatrix} e_{id}^{Bc_b} & e_{id}^{Rc_b} & e_{id}^{Bc_b} & e_{id}^{Rc_b} \\ e_{id}^{Bc_a} & e_{id}^{Rc_a} & e_{id}^{Rc_a} & e_{id}^{Bc_a} \end{Bmatrix}$.

Since the two chains in the Parallel-chain are connected frequently, the measured mean between the two endpoints is not only affected by the current Single-chain edge, but also by the edges in the other Single-chain. Therefore, it is necessary to consider the effects of all the edges in the Parallel-chain. To achieve this, we use the local optimization method to calculate the observation value of the edge in the equivalent quadrilateral. This local optimizing process is shown in Algorithm 4, whose specific process is explained in Section 4.3. The equivalent quadrilateral of the Parallel-chain can be obtained by optimizing the graph of the Parallel-chain, of which the edge corresponds to the optimized value $V_{e_{equal;id}}$ of the four edges added by the Parallel-chain.

The information matrix's calculation method for each edge of the equivalent quadrilateral is the same as the method in the Single-chain. The equivalent quadrilateral's information is stored into the $\{ParallelSET\}$ of the collection.

4.2.3 Replace

After the extraction and equivalence of Single-chains and Parallel-chains, the edges contained in the remaining overall graph need to be replaced with the equivalent one. The edges belonging to a Single-chain are extracted and replaced with the corresponding equivalent edges in $\{SingleSet\}$, and the edges belonging to the Parallel-chain are replaced by the edges of the equivalent quadrilateral in $\{ParallelSet\}$. After being replaced, the remaining graph is

$$e = e - SingleSet_{eids} - ParallelSet_{eids} + SingleSet_{eid:equal} - ParallelSet_{eid:equal} \quad (8)$$

4.3 Global optimization and solve

After the replacement of the Single-chain and the Parallel-chain, the scale of the remaining overall graph is already very small, and the remaining information matrix is very dense. At this point, the graph can be directly optimized to obtain a globally consistent solution, using the Algorithm 4.

This algorithm has been explained very exhaustive in [15]. For the sake of the integrity of the paper, here we simply describe the processing steps of the algorithm.

For each edge in the graph, the Jacobian matrix

$$A_{ij} = \frac{\partial e_{ij}(x)}{\partial x_i}, \quad (9)$$

$$B_{ij} = \frac{\partial e_{ij}(x)}{\partial x_j} \quad (10)$$

of the two endpoints is first calculated. Then, the information matrix H_{ij} is obtained from the rank numbers of A and B

$$H_{ij} = A_{ij}^T \Omega_{ij} B_{ij}. \quad (11)$$

Each edge's information matrix is used to construct the overall information matrix H by adding to the specific position of the edge's index, and the overall information vector b is also obtained by the same way.

Algorithm 4 $OptimizeAndSolve(E, v)$:

While! Coverage:
  $b = 0, H = 0$
  for all $\{e_{ij}, \Omega_{ij}\} \in E$:
    $A_{ij} = \frac{\partial e_{ij}(x)}{\partial x_i}$, $B_{ij} = \frac{\partial e_{ij}(x)}{\partial x_j}$
    $H_{ii} \mathrel{+}= A_{ij}^T \Omega_{ij} A_{ij}, H_{ij} \mathrel{+}= A_{ij}^T \Omega_{ij} B_{ij},$
    $H_{ji} \mathrel{+}= B_{ij}^T \Omega_{ij} A_{ij}, H_{ij} \mathrel{+}= B_{ij}^T \Omega_{ij} B_{ij},$
    $b_{[i]} \mathrel{+}= A_{ij}^T \Omega e_{ij}$, $b_{[j]} \mathrel{+}= A_{ij}^T \Omega e_{ij}$
  $\Delta v = -H^{-1} b$
  $v = v + \Delta e$
return v

Finally, the changing variable Δv of the edge is obtained by

$$\Delta v = -H^{-1}b, \quad (12)$$

which is added to the original variable

$$v = v + \Delta e. \quad (13)$$

This process will be executed many times till the changing variable Δv is small enough, which is assumed having been converged.

5 Experiments

The algorithm we proposed in this paper has been coded with Matlab, which is separated to 4 parts, as talked above, to optimize the robot pose graph. Now we have an experiment with the dataset used in[16] to test our algorithm.

As shown in the Figure 2.a, there is big error in the original pose graph, where the same place of the environment has mapped to different trajectory after mapping algorithm. As shown in Figure 2.b and 2.c, the Single-chains and the Parallel-chains are extracted and replaced from the overall trajectory by our algorithm.

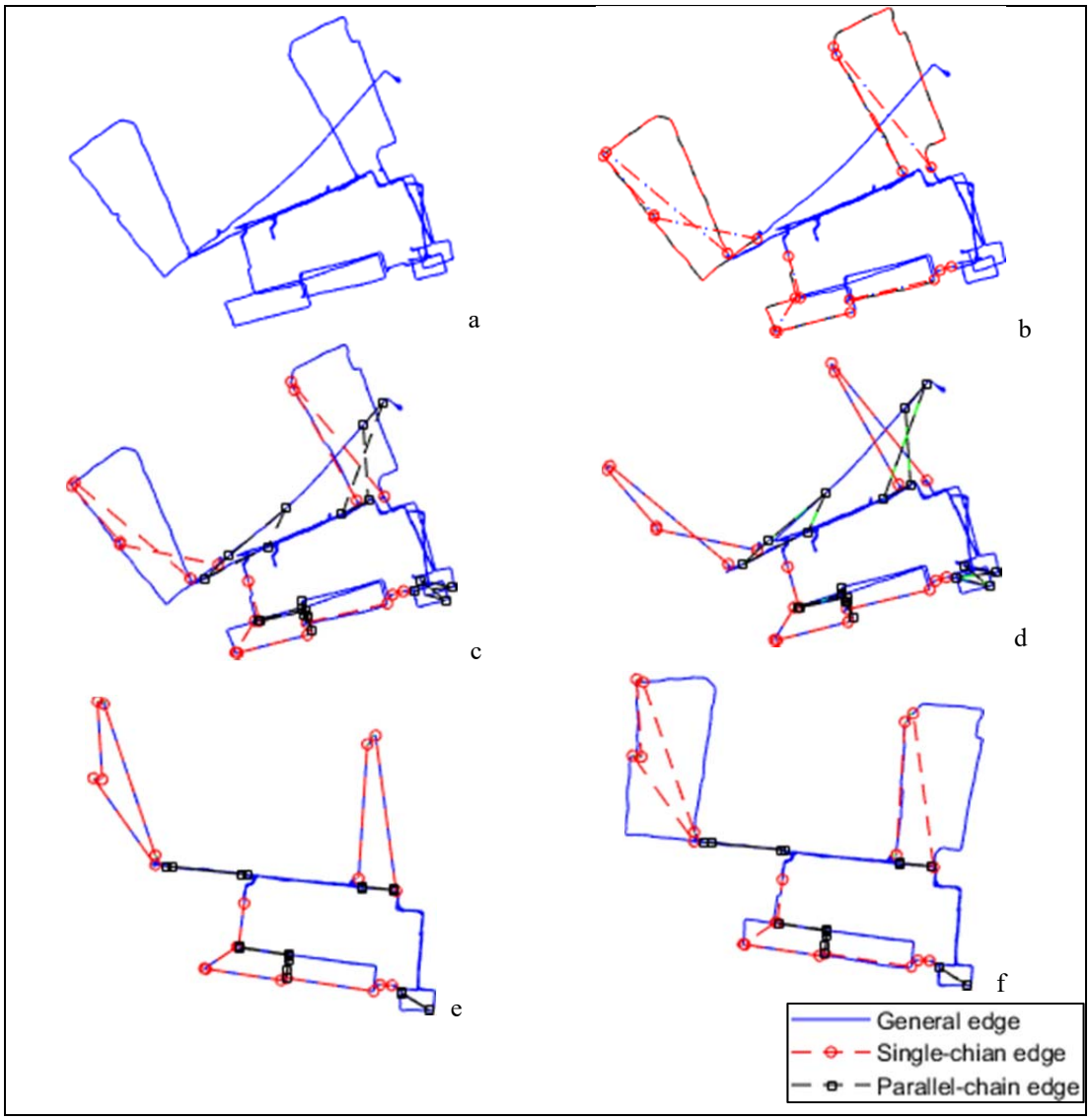

Fig. 2. Results of each step on real date.
a. The original pose graph
b. Extraction of Single-chain
c. Extraction of Parallel-chain
d. The remaining graph after replacement
e. The optimized remaining graph
f. The overall graph after substitution

Then the rest of the pose graph is optimized by Algorithm 4, as shown in Figure 2.e. Each equivalent edge is then removed and replaced in each subgraph. Finally, the subgraph is pieced together with the remaining graph to get the final result, as shown in Figure 2.f.

As shown in the Figure 3, the running time of our method and the original method are compared by iterating five times. After removing the Single-chain and the Parallel-chain, the remaining part of the graph takes a total of 4.188 seconds to iterate 5 times. The original algorithm iterated 5 times with 6.635, which is 36.88% less than our algorithm time.

Although the advantage in the overall optimization time is not very obvious, in the actual robot operation, only the environment around the current location is optimized, so the actual operation time is

$$CoreGraph + \frac{other\ part}{N} = 4.6484. \quad (14)$$

So the overall process has increased by 29.94%. In addition, the algorithm is conveniently applied to parallel computing platforms, which is because the overall graph is decomposed into many subgraphs to separate calculation. As shown in the Figure 4, the algorithm has great advantages in optimizing speed and parallel computing deployment.

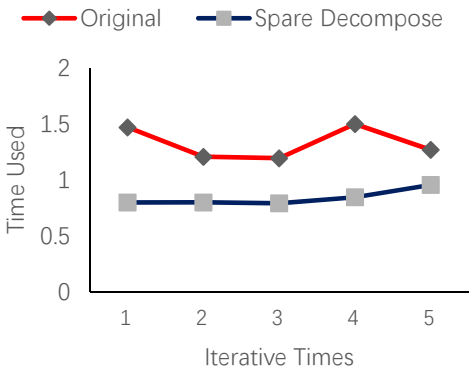

Fig. 3. The time consumption of each iteration

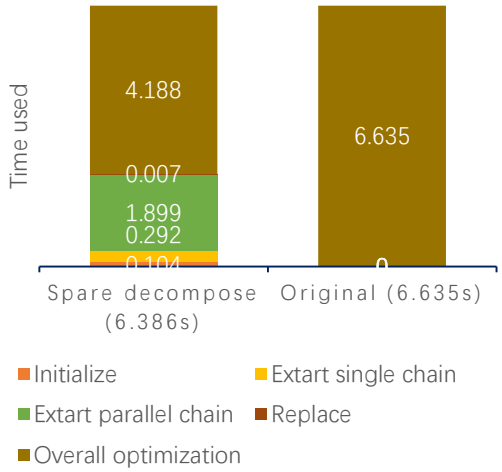

Fig. 4. The time consumption of each step

6 Conclusion

In this paper, a novel method of spare pose graph decomposition for slam is presented, which can accelerate the speed of algorithm in a graph SLAM. The equivalent substitution of the Single-chain and the Parallel-chain is used to decompose the pose graph into subgraph, so that the scale of the information matrix of the overall graph is reduced greatly, and the optimization task is simplified into optimize many subgraphs instead of the overall one.

And compared with the original method, the method we presented can greatly reduce the cost of the optimizing time on 29.94%. At the same time, the optimization algorithm is more suitable for the parallel optimization platform. In the follow-up work, we will constantly improve the local characteristics and make the performance of the algorithm more significant.

REFERENCES

[1] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial relationships in robotics," in *IEEE International Conference on Robotics and Automation. Proceedings*, 850 (2003)

[2] S. A. Holmes, G. Klein, and D. W. Murray, "An O(N(2)) square root unscented Kalman Filter for visual simultaneous localization and mapping," *IEEE Trans Pattern Anal Mach Intell*, **31**, 7, 1251–1263 (2009)

[3] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Auton. Robots*, **4**, 4, 333–349 (1997)

[4] T. A. Davis, *Direct Methods for Sparse Linear Systems (Fundamentals of Algorithms 2)*. Society for Industrial and Applied Mathematics, (2006)

[5] A. Howard, M. J. Mataric, and G. Sukhatme, "Relaxation on a mesh: a formalism for generalized localization," in *IEEE International Conference on Intelligent Robots and Systems*, 1055–1060 (2001)

[6] T. Duckett, S. Marsland, and J. Shapiro, "Fast, On-Line Learning of Globally Consistent Maps," *Auton. Robots*, **12**, 3, 287–300, (2002)

[7] U. Frese, "Treemap: An O (log n) algorithm for indoor simultaneous localization and mapping," *Auton. Robots*, **21**, 2, 103–122, (2006)

[8] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2006, no. May, 2262–2269, (2006)

[9] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear constraint network optimization for efficient map learning," *IEEE Trans. Intell. Transp. Syst.*, **10**, 3, 428–439, (2009)

[10] K. Konolige, "Sparse Sparse Bundle Adjustment," in *British Machine Vision Conference, BMVC 2010, Aberystwyth, UK, August 31 - September 3, 2010. Proceedings*, 1–11 (2010)

[11] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Trans. Robot.*, **33**, 5, 1255–1262 (2017)

[12] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Trans. Robot.*, **31**, 5, 1147–1163 (2015)

[13] G. Klein and D. Murray, "Parallel Tracking and Mapping on a camera phone," in *IEEE International Symposium on Mixed and Augmented Reality*, 83–86 (2009)

[14] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for

graph optimization," in *IEEE International Conference on Robotics and Automation*, 3607–3613 (2011)

[15] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intell. Transp. Syst. Mag.*, **2**, 4, 31–43, (2011)

[16] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intell. Transp. Syst. Mag.*, **2**, 4, 31–43, (2010)