An Iterative Graph Optimization Approach for 2D SLAM

He Zhang, Guoliang Liu, *Member, IEEE,* and Zifeng Hou

***Abstract*— The-state-of-the-art graph optimization method can robustly converge into a solution with least square errors for the graph structure. Nevertheless, when a biased edge (erroneous transformation with over-confident information matrix) exists, the optimal solution can produce the large deviation because of error propagation produced by the biased edge.**

In order to solve this problem in graph-based 2D SLAM system, this paper proposed an iterative graph optimization approach. To reduce the errors propagated from the biased edges, we iteratively reconstruct the graph structure by referring to the result of the graph optimization process. Meanwhile, to maintain the information of the other well estimated edges, we strictly update the graph structure by considering the scan-correlation score and the marginal covariance. In addition, we apply a novel key-node mechanism to robustly detect the loop-closure by a linear interpolation algorithm. The experiments show that the proposed method is more robust and accurate than the previous methods when the biased edges exist.

I. INTRODUCTION

The simultaneously localization and mapping (SLAM) problem is a pivotal problem in the robotics community, since it handles the most significant information for the autonomous mobile robot. In 2D SLAM, laser scanner and odometry usually be applied in the-state-of-the-art methods such as Grid Mapping (GMapping) [11] and Graph Optimization (GO) [13]. These methods work well and noisy-resistent, but they will degenerate when motions are estimated erroneously but with high confidence. This often occurs when the vehicle traverses through a corridor-like place and the vehicle slips or applying laser-odometry algorithms [3][7][17]. The previous methods can fail when such edges exist.

The Gmapping implements a rao-blackwellized particle filters(RBPF) to predict its pose distribution under gaussian assumption and update the weights of the particles under the Baysian rule with a likelihood evaluation for each prior pose. Then resampling is carried out to eliminate particles with small weights. However, if motions are estimated with extreme noise such as vehicle slipper, resampling may eliminate particles with true trajectory and could never be recovered.

The graph-based SLAM methods strive to reduce the errors of the odometry constraint and the loop constraint. To reduce the errors of the loop constraint, previous works such as JCBB[16], SCGP[19], RRR[14] involve front-end validation of loop closure to eliminate the false positive loop edges. For the errors of the odometry constraint, they are assumed to be corrected during the graph optimization when true loop edges are added into the graph structure. However, if some odometry constraint contains large errors and over-confident information matrixes, the graph optimization will also result in erroneous solution even when no false loop edges exist.

The key insight of the proposed approach is that we can correct the biased motion estimation in the front-end by referring to the result of the graph optimization in the back-end. When the poses of the nodes are updated, we can use the same scan-matching algorithm to recalculate the edge information with various prior motion guesses. We perceive that in our 2D graph-based SLAM method, biased edges result from poor prior motion guess that can be improved when loops are accurately closed and the graph structure is optimized. Therefore, we propose the iterative graph optimization algorithm to update the graph structure in the front-end by the aid of the optimization process in the back-end.

The minor insight is that, in contrast to detecting loops among nodes in the graph, we construct recoverable key-node to verify loop closures. We perceive that the local map information contained in key-node can be inconsistent and thus we can rebuild the local map for each key-node after the graph optimization. If the local map of a key-node is maintained as a whole in a higher level for the map representation [4][5][23], we can never eliminate the errors in the local map. Therefore, we rebuild the local map of each key-node when the nodes are updated by the graph optimization process. In addition, because of the accumulated motion error, the initial motion guess to align a scan frame with a local map may fall out of the right convergent basin. Thus we provide multiple interpolated initial motion guesses between current pose and the pose of the key-node to robustly detect loops.

In short, the central contributions of this paper are following:

- We apply a new iterative graph optimization approach that reduces the errors propagated by the biased edges in the graph.
- We adopt key-nodes mechanism to robustly detect loop. The corruptly constructed key-nodes could be recovered when good loop edges are inserted into the graph.
- We employ an adaptive linear interpolation algorithm to detect loop and recalculate edge information after graph optimization.

The structure of this paper is as follows. In the fol-

H. Zhang and Z. Hou are with the Institute of Computing Technology, University of Chinese Academy of Sciences and with the Lenovo (Beijing) Ltd. 6th Academy Road, Haidian District Beijing Email: fuyinzh@gmail.com; houzf@lenovo.com

G. Liu is with the Lenovo (Beijing) Ltd. Email: liugl6@lenovo.com

lowing section, the previous related works are discussed. In section III, the process of graph construction will be demonstrated. After that, the iterative graph optimization algorithm is illustrated in detail. Then we carry out two experiments using the real data to prove the superiority of our method in the section V. In the end, the conclusion and future work are presented.

II. Related Work

Kümmerle *et al*. [13][9] illustrate the advantages of the graph-based SLAM methods. But when the graph structure contains false loop edges, these methods will fail catastrophically. To solve this problem, many front-end validation approaches [16][19][14] have been proposed. In addition, other methods strive to alter the graph structure during the back-end optimization process. Sünderhauf and Protzel [20] propose switchable variables to alter the graph structure during the graph optimization process, that can turn off the false loop edges. Olson and Agarwal explicitely [18] model the errors of loop edges and track multiple hypotheses for each edge during the graph optimization process. However, as far as we know, none of these methods make effort to reduce the errors of the biased edges. In fact, the graph optimization could converges into an undesirable solution when the graph contains a biased edge even no false loop edge. Our 2D SLAM method dedicates to reduce the errors of the biased edges by iteratively reconstructing the graph structure with reference to the result of the graph optimization process.

In addition, our 2D SLAM method adopt the submap mechanism to improve the robustness of data association. This idea is not entirely new and has been intensely explored [5][6][4]. The improvements of the proposed method lies in two parts: interpolated initial motion guesses for scan-matching [17] and recoverable mapping. The first part can improve the accuracy of loop edges by aligning a single scan with a previous submap with multiple initial motion guesses. The second part can reduce the errors of the local map by reconstructing the local map after the graph optimization.

III. Graph Construction

The graph-based SLAM algorithm contains two components: the front-end and the back-end. The front-end includes graph construction and the back-end focuses on graph optimization. In this section, we mainly talk about the front-end process, and leave the back-end in the next section. The front-end mainly contains two parts: motion estimation and loop detection. For the motion estimation, we use the particle-type representation to model the uncertainties like the work in [15][10]. We randomly choose a set of hypothesis based on the prior odometry model. And then we employ scan-matcher algorithm [17] to compute the scan correlation score between the current laser frame with the global grid map [21] as the weight of each hypothesis. We choose the hypothesis with maximum weight and add its transformation and information matrix into graph. For the loop detection, we adopt the key-node idea, and construct local grid map in each key-node. We apply scan alignment with interpolated prior guesses to calculate loop constraints.

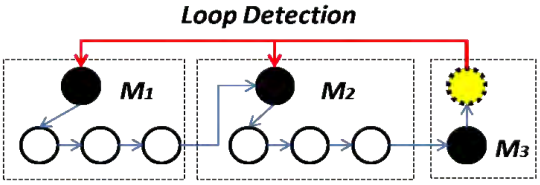

Fig. 1: Graph Structure: black solid circle stands for key-node, dotted rectangle includes the information in each key-node, and the dotted yellow circle means the current node; the blue edges stands for the transformation between the nodes, and the red edges are loop-edges

A. Motion Estimation

In our method, we estimate a set of hypotheses on the relative motion. We provide prior guesses P_{pri} for robot position P_i according to its odometry model with gaussian noise Σ_{odo}.

$$\Sigma_{odo} = \begin{bmatrix} \Sigma_{tt} & \Sigma_{tr} \\ \Sigma_{rt} & \Sigma_{rr} \end{bmatrix} \tag{1}$$

After that, we measure the scan correlation score between current observation and global gridmap M_g with these prior poses P_{pri}. We set P_i as the pose with the maximum scan correlation score. Then we calculate the relative transformation and covariance Σ_{obs} [2] between successive robot pose P_{i-1} and P_i. In the experiment, we found that the Σ_{obs} is often over-confidently estimated. Therefore we take the correlation score into consideration when the covariance is estimated between P_{i-1} and P_i as follows:

$$\Sigma_{i-1,i} = \begin{cases} \Sigma_{obs} & \text{if } Score \geq minScore \\ \Sigma_{obs} + \frac{minScore - Score}{minScore} * \Sigma_{odo} & \text{otherwise} \end{cases} \tag{2}$$

minScore represents the least percentage laser beams for a good estimation in the scan-matcher algorithm. In our experiments we found it works well when it is set as 85% of the total laser beams. Then, we add node n_i and edge $e_{i-1,i}$ into graph. The information matrix in $e_{i-1,i}$ is $\Sigma^{-1}_{i-1,i}$. After that, we construct key-node and detect loops which will be explained as follows.

B. Key-Node Construction

The key-node is the same as other nodes except that it integrates observations into a local map. As shown in the Figure 1, the black solid circle means a key-node. The dotted rectangle shows the nodes and the local gridmap M_i of key-node i. As explained more explicitly in [12][8], the motivation behind key-node is to increase loop detection accuracy by matching with a local submap instead of a single scan. Therefore, when searching loop connections, we only select the key-nodes as potential loop matches. For example in Figure 1, the yellow circle is the current pose of the robot, and it detect potential loops by matching with the key-nodes k_n_1 and k_n_2. In addition, the transition between key-nodes can be triggered under different constrains such

as observation area, trajectory length or number of nodes. In our method, we find that using trajectory length can guarantee substantial grid map size but also works as a clue to detect loops. In order to reduce error in the key-node, the maintained grid map M_i of key-node k_n_i will be rebuilt in the iterative optimization process. This will be illustrated in the section IV.

As mentioned previously, we switch to a new key-node when the length of the trajectory contained in current key-node exceeds a predefined distance T_l. T_l is set based on the scale of the environment in which the robot traverses. On the one hand, it must contain enough observed information to make accurate scan alignment when loop occurs. On the other hand, key-nodes should be distributed broadly for the current node to efficiently check whether it reenter the area in one of the previous key-nodes. Once this happen, we verify whether there is loop-closure detected. This process is discussed in the next part.

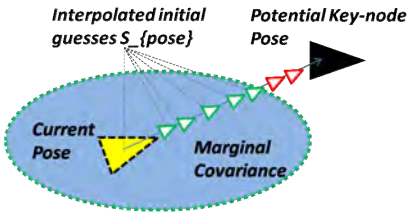

Fig. 2: Interpolated Poses for Initial Guess in Scan-Matcher

C. Loop Closure

When the current pose of robot gets close to a previous key-node, that means the distance between the pose P_i of current node i and the pose P_{k_j} a key-node j is less than T_l, a potential loop may exist. Then we further check if there exist a node n_l in the key-node k_n_j, and the Mahalanobis distance between them satisfy:

$$e_{dis} = P_i \ominus P_l \tag{3}$$

$$\Sigma_{il} = R'_{li} * \Sigma_l * R_{li} + \Sigma_i \tag{4}$$

$$\mid e_{dis}^{-1} * \Sigma_{il} * e_{dis} \mid < \varepsilon \tag{5}$$

Σ_i and Σ_l is the marginal covariance for the node i and l respectively. P_i and P_l are the pose for the node i and l. R_{li} is the rotation part of the transformation from pose P_l to P_i. ε is a predefined threshold. When these conditions are satisfied, we will align the current scan frame and the gridmap M_j in k_n_j to verify whether a consistent transformation can be estimated.

When the robot traverses along a long path, the accumulated motion error becomes large. In this case, the initial guess maybe far from the right convergent basin, therefore using scan-matching algorithms [3][7][17] may fail to precisely estimate the transformation for the loop constraint. Even if the overlap between the current laser frame and the gridmap in a key-node is large, without proper initial guess, the frame alignment algorithm may fall into poor local optima. To solve this problem, we make multiple initial guesses by adaptively interpolating initial motion guesses between P_i and P_{k_j}. As depicted in the Figure 2, we provide multiple initial guesses (small triangles) by linearly interpolating poses between current robot position and key-node position. We select the result from scan-matcher with the biggest scan correlation score. The linear interpolation is a dynamical process, it takes into P_{dis} and marginal covariance Σ_i into consideration. In our experiments, we assume that the proper initial motion values distribute along the direction from P_i and P_{k_j}. Therefore along this direction, we interpolate an initial pose every $5cm$ within P_{dis}.

The Σ_i is the marginal covariance of the node i. When no loop is closed, it accumulates according to:

$$\Sigma_i = R_{i-1,i}^{-1} * \Sigma_{i-1} * R_{i-1,i} + \Sigma_{i-1,i} \tag{6}$$

$R_{i-1,i}$ is the rotation part of the transformation from pose P_{i-1} to P_i. $\Sigma_{i-1,i}$ is the estimated covariance between observations in the node $i-1$ and i by equation 2. If loop closure happens, it is calculated following the rule in [22].
The blue eclipse in the Figure 2 represents the current marginal covariance, and the red small triangles that exceed the eclipse will be discarded. When the loop is verified, a loop edge will be inserted into the graph, and we will perform graph optimization process explained in the next section.

Algorithm 1 Iterative Graph-based Optimization Algorithm

```
1: function REBUILDGRAPHANDMAP
2:     S_l                          ▷ Loop edges and nodes detected in Frontend
3:     M_g ← empty                  ▷ Global gridmap
4:     for n_i in graph do
5:         if n_i ∈ S_l then
6:             continue             ▷ skip loop nodes
7:         end if
8:         S_pose = interpolate(P_i, P_inew)
9:         e_{i-1,i} = maxScanMatcher(n_{i-1}, n_i, S_pose, M_g)
10:        P_i = P_{i-1}.oplus(e_{i-1,i})
11:        graph.replace(e_{i-1,i})   ▷ replace with the new edge
12:        graph.update(P_i)          ▷ reset pose of node
13:        M_g.insert(n_i)            ▷ reconstruct global gridmap
14:    end for
15:    for e_{j,k} in S_l do
16:        S_pose = interpolate(P_k, P_knew)
17:        e_{j,k} = maxScanMatcher(key_n_j, n_k, S_pose, M_j)
18:        P_k = P_j.oplus(e_{j,k})
19:        graph.replace(e_{j,k})
20:        graph.update(P_k)
21:    end for
22:    reconstructKeyMap()          ▷ rebuild gridmap in each key_node
23: end function
24: function ITERATIVEGRAPHOPTIMIZATION(iter)
25:    last_chi2 = optimizeGraph()
26:    while i++ < iter do
27:        rebuildGraphandMap()
28:        curr_chi2 = optimizeGraph()
29:        if |curr_chi2 − last_chi2| < ε then
30:            break
31:        end if
32:        last_chi2 = curr_chi2
33:    end while
34: end function
```

IV. ITERATIVE GRAPH OPTIMIZATION

The graph optimization dedicates to find a solution with least square errors given the topological graph structure.

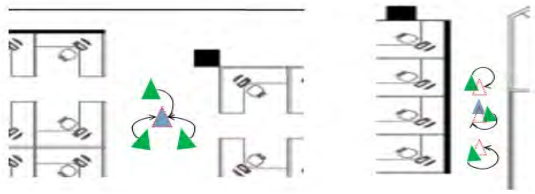

Fig. 3: Scan-Matching using the same laser frame in the Office-like scenario (left) and the Corridor-like scenario (right): blue triangle is the target vehicle pose, green triangles stand for different prior reference vehicle pose, and the red dashed triangle means the convergent vehicle pose calculated by scan-matching

However, even when no false loop edge exists in the graph, some biased edges (erroneous transformation with over-confident information matrix) can result in solution with significant error. To reduce the errors propagated from the biased edges rather than false loop edges, we propose the iterative graph optimization algorithm that will be explained in the following contents.

A. Biased Edges

In our 2D SLAM system, we use scan-matching to estimate the relative motion between sequential vehicle poses. However, the accuracy of scan matching depends highly on the scenarios where the vehicle stays. As shown in Figure 3, in the left office like environment, the scan-matching can result in the same accurate pose (red dashed triangle) even with different priors (green triangles). While in the right corridor like place, its results differ and ranks along the corridor, and thus it's hard to decide where the vehicle stands. With poor prior odometry or laser-odometry, the transformation of the edges in this scenario is often erroneously estimated and the information matrix is over-confidently calculated, that we call biased edges. To improve the accuracy of the biased edges, a better prior motion guess must be provided. We find that when the true loop edges are added, the graph optimization can update the poses of the nodes, which might provide a better prior motion to adjust the biased edges. Therefore, the essence of iteratively optimizing graph is to alter the graph structure in the front-end using the result from the back-end.

B. Iterative Graph Reconstruction

The motivation behind iterative graph reconstruction is to recover the well estimated edges and improve the biased edges. As shown in the Figure 4, the red edge $e(3,4)$ between node 3 and 4 is a biased edge, and the green edge $e(1,5)$ between node 1 and 5 is a validated loop edge. The initial graph structure is depicted in the Figure 4.(1), a loop is closed and a biased edge has been added into the graph. After graph optimization, because of the propagated error from the biased edge, the result of the whole trajectory degenerates, shown in the Figure 4.(2). Then we reconstruct the graph structure as the same process in the front-end, but using the result of the back-end as the prior motion guess. For the good

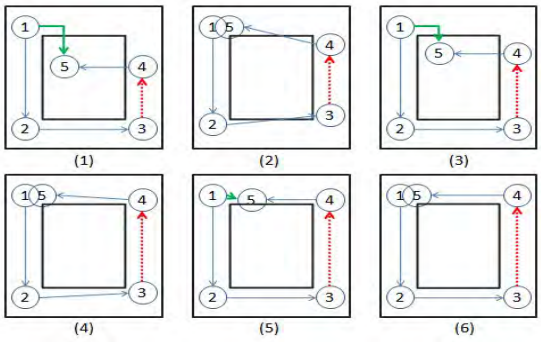

Fig. 4: Iterative Graph Reconstruction: (1) initial graph structure, (2) 1st graph optimization, (3) 1st graph reconstruction, (4) 2nd graph optimization, (5) 2nd graph reconstruction, (6) final graph optimization. Green arrow stands for loop edge, blue for good edge and red dashed for biased edge

matches, different prior motion guesses can still falls into the same convergent basin as shown in the left part in the Figure 3. While for the biased matches, the relative motion can be updated with a better initial motion guess. This is illustrated in the Figure 4.(3), we see that the relative motion of edges $e(1,2), e(2,3), e(4,5)$ and $e(1,5)$ be recovered, while that $e(3,4)$ be updated. Then we again repeat the same process as shown in the Figure 4.(4)(5), but notice that the red edge $e(3,4)$ in the Figure 4.(3) and the Figure 4.(5) differs, the latter has been improved by iteratively using the result of the back-end optimization process. When the graph structure no longer varies, we again optimize it and obtain the final result as shown in the Figure4.(6).

C. Algorithm Explantation

The major part of the Algorithm 1 is to rebuild the graph structure and the global gridmap. In line 5 and 6 we skip nodes that has been updated by loop edge for which will be recalculated in the loop set S_l. In line 8 and 16 we interpolate poses between the original position P_i and new position $P_{i_{new}}$ that is updated by graph optimization. We use the same interpolation mechanism as explained in Figure 2 in III-C, and the only difference is that the target position is $P_{i_{new}}$ and current position is P_i. We return the transformation with maximum score, that means most consistent with the rebuilt map M_g. In line 4-14, we rebuild the edges between successive nodes along the robot trajectory. In line 15-21, we recompute the loop edges by aligning laser scan in node n_k with gridmap M_j in the key-node k_n_j. To accurately recover the original estimations once perturbed by graph-based optimization, we strictly reset the pose of each node and the edges between them by considering the marginal covariance and the scan correlation score. After that, in line 22 we reconstruct the local gridmap in each key-node with the newly updated nodes.

The iterative graph optimization part is quite straightforward: we iteratively rebuild graph and optimize it until it converges or the iteration time is more than the threshold $iter$. The $optimizeGraph()$ will optimize the graph and return

the total square error over the graph.

D. Computational Time Analysis

Suppose the average computational time for graph optimization and scan-matching are $T(o)$ and $T(m)$ respectively. For a graph with E edges, iGO (iterative graph optimization) costs $k(T(o)+E*T(m))+T(o)$ while GO only $T(o)$. k is the average iteration number. However, we can compare the transformation of the edges before and after the graph reconstruction. Therefore, for the edges with no alterations, that means well estimated edges, we will not update in the next loop. For example in Figure, we only recalculate the edges $e(1,2), e(2,3), e(4,5)$ and $e(1,5)$ in the first iteration, and yet update edge $e(3,4)$ in every iteration. Then, the computational time for iGO is $k(T(o)+b*T(m))+E*T(m)+T(o)$. b is the number of biased edges. If no biased edges exist, iGO costs $2*T(o)+E*T(m)$, and the extra time consumed can be seemed as to detect biased edges.

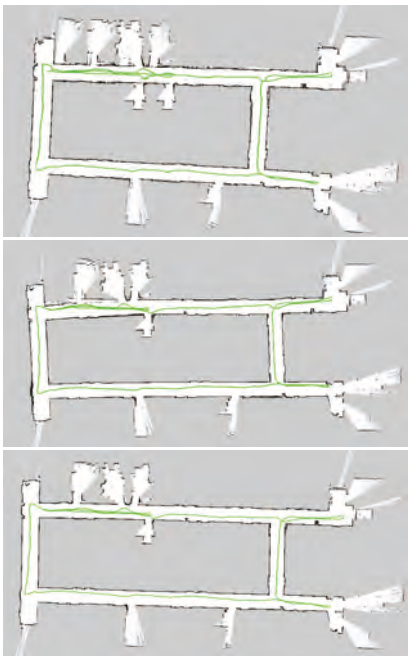

Fig. 5: 2D Gridmap Comparisons Top to Bottom: GMapping, GO and iGO

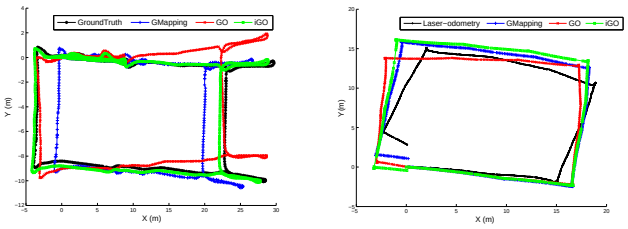

Fig. 6: Trajectory Comparison. Left: GroundTruth, GMapping, GO and iGO; Right: Laser-odometry, GMapping, GO and iGO

V. EXPERIMENT

To demonstrate the proposed method can recover from biased motion estimation, we conducted two experiments using the data from the real environment in two cases:

- When slippage occurs, motion estimation model contains high uncertainty.
- When applying laser-odometry, accuracy of motion estimation varies heavily.

We compare the results via three SLAM methods: GMapping [11], GO [13], and iGO, the proposed method. The only difference between GO and iGO lies in the optimization process in the back-end, while the process in the front-end is the same.

A. experiment I

In the first experiment, we use the uscsal data from the Radish [1]. To simulate vehicle slippage, we increase the motion model covariance Σ_{odo} with $\Sigma_{tt}=1.6$ and $\Sigma_{rr}=0.8$. In this case, for GMapping, the weight of particles jumps and resampling occurs which can eliminate some particles with true motion. As shown in the first row of the Figure 5 and the blue cross line in the left part of the Figure 6, the total trajectory shrinks and results in inconsistent 2D gridmap. For GO, some edges along the corridors may contain biased information. Therefore, depicted in the second row of the Figure 5, after closing loops and optimization, the whole graph can converges into a worse solution. Compared to the groundtruth, the result of GO bears large angular error shown in the left part of the Figure 6. In comparison, our approach can maintain information in the well estimated edges and improve the biased edges. As depicted in the third row in the Figure 5 and the green square line in the left part of the Figure 6, the result of our method is more consistent and accurate.

B. experiment II

In the second experiment, we control the vehicle traverse along a circle in the Lenovo B2 office which is about 17m width and 22m length, and the total length of the trajectory is about 80 meters. We use laser-odometry to predict the realtive motion. Therefore the covariance of the motion estimation depends highly on the scenes it traverse. For example, laser-odometry provide less accurate position information in the corridor-like place than other places with rich observation. The trajectory and map comparisons under different methods are shown in the Figure 7 and the right part of the Figure 6.

As shown in the right part of the Figure 6, laser-odometry [17] offers erroneous transformation in the corridor-like places. However, since the laser-scans are matched well along the corridor, it estimates high confidence information matrix. Because of this biased information, GMapping fails to correctly estimate the poses of the particles and result in inconsistent gridmap, Figure 7 (a). For the same reason, even GO can accurately close the loop, the optimization degenerates the whole vehicle trajectory and constructs a worse 2D map Figure 7 (b). On the contrary, our method can

(a) GMapping (b) GO (c) iGO

Fig. 7: Mapping Comparison: the red dots stand for the final 2D grid map, green line is the trajectory and the background is the sketch 2D map for the Lenovo B2 office

not only detect the loop closure precisely, but also reduce the error propagation introduced by the biased edges and improves the transformation in the biased edges. Compared with the sketch map of the work station, the gridmap of the proposed method is more consistent and accurate than those of the other methods, depicted in the Figure 7 (c).

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we introduce a new iterative graph optimization method. Our major contributions are two folds: a) iteratively rebuilding and optimizing the graph structure can maintain the well estimated edges, and improve the biased edges; b) a novel key-node mechanism and an interpolation algorithm that can help the loop-closure. The effectiveness of the proposed new ideas have been verified in our experiments.

Although the proposed method has many advantages compared with the previous methods, it can not recover from errors propagated by the false loop edges. A single large misleading loop closure could make the algorithm fail. In the future, we can combine the loop validation algorithms with the proposed method to make the graph-based SLAM method more robust.

REFERENCES

[1] http://cres.usc.edu/radishrepository/view-all.php.

[2] O. Bengtsson and A. Baerveldt. Robot localization based on scan-matchingłestimating the covariance matrix for the idc algorithm. *Robotics and Autonomous Systems*, 44(1):29–40, 2003.

[3] P. Besl and N. McKay. A method for registration of 3-d shapes. *IEEE Transactions on pattern analysis and machine intelligence*, 14(2):239–256, 1992.

[4] J. Blanco, J. Fernández-Madrigal, and J. Gonzalez. A new approach for large-scale localization and mapping: Hybrid metric-topological slam. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2061–2067. IEEE, 2007.

[5] M. Bosse. *Atlas: a framework for large scale automated mapping and localization*. PhD thesis, Massachusetts Institute of Technology, 2004.

[6] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based slam. *The International Journal of Robotics Research*, 27(6):667–691, 2008.

[7] A. Censi. An icp variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25. IEEE, 2008.

[8] C. Estrada, J. Neira, and J. D. Tardós. Hierarchical slam: Real-time accurate mapping of large environments. *Robotics, IEEE Transactions on*, 21(4):588–596, 2005.

[9] G. Grisetti, R. Ku?mmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based slam. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[10] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *Robotics, IEEE Transactions on*, 23(1):34–46, 2007.

[11] G. Grisettiyz, C. Stachniss, and W. Burgard. Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437. IEEE, 2005.

[12] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Computational Intelligence in Robotics and Automation, 1999. CIRA'99. Proceedings. 1999 IEEE International Symposium on*, pages 318–325. IEEE, 1999.

[13] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[14] Y. Latif, C. C. Lerma, and J. Neira. Robust loop closing over time. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2012.

[15] J. Min, J. Kim, S. Shin, and I.-S. Kweon. Data-driven mcmc sampling for vision-based 6d slam. *Electronics letters*, 48(12):687–689, 2012.

[16] J. Neira and J. D. Tards. Data association in stochastic mapping using the joint compatibility test. *IEEE T. Robotics and Automation*, 17(6):890–897, 2001.

[17] E. Olson. Real-time correlative scan matching. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4387–4393. IEEE, 2009.

[18] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *The International Journal of Robotics Research*, 32(7):826–840, 2013.

[19] E. Olson, M. Walter, S. J. Teller, and J. J. Leonard. Single-cluster spectral graph partitioning for robotics applications. In S. Thrun, G. S. Sukhatme, and S. Schaal, editors, *Robotics: Science and Systems*, pages 265–272. The MIT Press, 2005.

[20] N. Snderhauf and P. Protzel. Switchable constraints for robust pose graph slam. In *IROS*, pages 1879–1884. IEEE, 2012.

[21] S. Thrun, W. Burgard, and D. Fox. Probability robotics, 2005.

[22] G. D. Tipaldi, G. Grisetti, and W. Burgard. Approximate covariance estimation in graphical approaches to slam. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3460–3465. IEEE, 2007.

[23] H. Zhang, Z. Hou, N. Li, and S. Song. A graph-based hierarchical slam framework for large-scale mapping. *Intelligent Robotics and Applications*, pages 439–448, 2012.