Comparison of Methods to Efficient Graph SLAM Under General Optimization Framework

Haoran Li[1,2],Qichao Zhang[1,2], Dongbin Zhao[1,2]

1. The state Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China

2. University of Chinese Academy of Sciences, Beijing, 100049, China
E-mail: lihaoran2015@ia.ac.cn,zhangqichao2013@163.com, dongbin.zhao@ia.ac.cn

Abstract: Simultaneous Localization and Mapping(SLAM) algorithms can infer the robot's trajectory as well as the map under unknown environment. Robust and time-efficient optimization methods are important requirements for SLAM. There are many algorithms designed for the graph optimization. However, it is hard to select an appropriate algorithm and corresponding software library, due to the difficulty of evaluating algorithms' adaptabilities under various situations. In this paper, we summarize these algorithms under general optimization framework, conduct several sets of experiments to compare these algorithms in three software libraries, and give some suggestions to choose algorithms.

Key Words: SLAM, pose graph, optimization

1 INTRODUCTION

SLAM has been an active topic in robotic communities and computer vision for decades. It simultaneously estimates the state of robot equipped with on-board sensors and constructs the model of the environment that the sensors have been perceiving. This technique has wide applications such as self-driving cars, autonomous navigation robots and virtual reality.

Many approaches have been proposed to solve the SLAM problem. These approaches can be classified either as filtering or smoothing [1]. Filtering methods estimate the current state of robots according to the collected sensor data. These approaches address the on-line SLAM problem. Smoothing methods, conversely, address the full S-LAM problem, as these approaches estimate the whole trajectory. Filtering is time-efficient, but it is inconsistent in the nonlinear SLAM problem [2]. Smoothing can obtain the extract solution consistently [3].

Optimization is the core composition of smoothing approaches. The graph structure converts the problem of state estimation to a nonlinear least squares optimization problem, and a large variety of optimization methods have been introduced, for example, gradient descent [4], Gaussian-Newtown [5], Levenberg-Marquardt [6] and so on. However, as these methods are sensitive to outliers, many robust approaches have been proposed [7] [8].

Although there are many approaches to solve the SLAM, it is still difficult to determine an efficient and robust method under different conditions. To address this problem, we summarize strengths and weaknesses of different approaches. In addition, we compare the performance of those approaches in term of convergence, sensitivity to noise and robustness.

In the following, we briefly review the relate work. Then we introduce the general formulation for pose graph SLAM problems, and the several methods under nonlinear optimization theory. And next we conduct sets of experiment to compare the performance of the mentioned methods, and finally we conclude our works and give some suggestions.

2 RELATED WORK

Lu and Milios [9] firstly reduced the error introduced by constraints by means of optimizing the system of equations to refine a map. Afterwards, many approaches for optimizing the graph constructed by smoothing and mapping have been proposed. Olson *et al.* [4] presented an efficient optimization approach which was based on stochastic gradient descent. Grisetti *et al.* proposed an extension of Olson's approach to increase the convergence speed [10], by applying a tree based parametrization. This approach was constrained by the assumption that the error in graphs was close to uniform.

The essential way of optimizing a graph that constructed by smoothing and mapping is nonlinear optimization. Levenberg-Marquardt method which is popular in Bundle Adjustment, plays an important role in solving graphs efficiently. One of the challenges in Levenberg-Marquardt is solving large a linear system obtained by linearizing the original likelihood function of the graph. Dellaert and Kaess [5] were the first to exploit sparse matrix factorizations to solve the linearized problem in full SLAM. The approach of Konolige *et al.* [6] were similar to Dellaert's. They used ordered data structures to construct the linear subproblem efficiently. Kummerle *et al.* [11] presented a framework for performing the optimization of nonlinear least squares problems that can be represented as a graph.

This work is supported by National Natural Science Foundation of China (NSFC) under Grants No. 61573353, No. 61533017, the National Key Research and Development Plan under Grants 2016YFB0101000.

Several online algorithms are designed for real-time S-LAM. Kaess *et al.* [12] presented iSAM which needed periodic batch steps to reorder variables and relinearize. These batch steps were expensive and damaged the intended online nature of the algorithm, and Kaess *et al.* [3] proposed iSAM2 which used the Bayes tree to address the incremental reordering and just-in-time relinearization. Rosen *et al.* [13] adopted Powell's dogleg method instead of Gaussian-Newton in iSAM2, to increase the algorithm's robustness for highly nonlinear systems and numerically ill-conditioned Jacobians.

Since least squares optimization methods are in general not robust to outliers [7], the approaches mentioned above are likely to fail where the graph is ill-defined. Sünderhauf and Protze [7] proposed a switching behavior to identify outliers, by adding a variable for each data association constraints to switch off potential outlier constraints. This method increased the scale of optimization, as it introduced new constraints for switching variables. Agarwal *et al.* [8] presented a convenient way to estimate the switching variables without additional constraints.

3 METHODS TO GRAPH OPTIMIZATION

The objective of SLAM is to estimate the robot's poses and the environment's map. In the unknown environment m, a robot executes a sequence of control commands $u_{1:T}$ from a initial position x_0, and takes a sequence of poses $x_{1:T}$. At the same time it acquires a sequence of measurements $z_{1:T}$ of the environment. Due to the noise of sensor measurements, most literatures describe SLAM by means of posterior probability. The posterior can be described as $P(x_{1:T}, m|u_{1:T}, z_{1:T}, x_0)$. By the probabilistic inference in the context of maximum likelihood (ML) or maximum a posteriori (MAP), this state estimation problem can convert to the nonlinear least squares problem [1]:

$$X^* = \text{arg min} \sum_{i \in \varepsilon} e_i^T \Omega_i e_i \tag{1}$$

where X is the vector of the robot's poses and e_i is the i-th error function in the pose graph. Ω_i is the information matrix of e_i. ε is the set of constraints in the graph. We can define $r = \Sigma^{1/2} e$, where the diagonal block of Σ is Ω_i and e is the vector of e_i. We can simplify (1) to obtain

$$X^* = \text{arg min } r^T r. \tag{2}$$

We define the cost function $f = r^T r$. There are many iterative methods to solve this problem, and these methods can be classified either as linear search method or trust region method.

3.1 Line Search Methods

Iterative search methods require the search step length and the search direction. Line search methods find the step length and search direction individually. According to the types of search directions, the methods of graph optimization can be divided into two classes: gradient descent and Gauss-Newton.

3.1.1 Stochastic Gradient Descent

A basic method to solve minimization is gradient descent. The gradient of (2) is

$$\nabla f = J^T r \tag{3}$$

where J is the Jacobian of the r. The cost function decreases most sharply in the negative gradient direction . However, gradient descent has to scan through the whole elements of r before taking a single step, and if the number of the elements is large, this progress will be rather slow. Olson *et al.* proposed[4] a fast iterative version which was an extension of stochastic gradient descent

$$d \approx -2\alpha M^{-1} J_i^T r_i \tag{4}$$

where d is the search direction and α is the search step length. M is the diagonal matrix with same diagonal elements as $J^T J$. J_i is the i-th column of J and r_i is the i-th element of r. Without waiting for the whole elements of r, this method can update right now when looks at each elements of r. And this method speeds up the convergence of traditional stochastic gradient descent. Although stochastic gradient descent is easy to implement and can converge to minimum, as the gradient is not the true gradient and it can be viewed as a sample of true gradient, this method's convergence speed is slow.

3.1.2 Gauss-Newton

A well-known direction to speed up convergence is Newton direction. To generate the search direction d, we need solve the standard Newton equations $\nabla^2 f d = -\nabla f$ [14], and the Hessian matrix is

$$\nabla^2 f = J^T J + \sum_{j=1}^{m} r_j \nabla^2 r_j. \tag{5}$$

Gauss-Newton method is approximate to the Newton method. It excludes the second-order term from Hessian matrix, and uses $J^T J$ instead of $\nabla^2 f$

$$J^T J d = -J^T r. \tag{6}$$

In essence, Gauss-Newton uses a quadratic function to approximate the cost function at current step.

$$\text{min } q(d) = f + g^T d + \frac{1}{2} d^T B d \tag{7}$$

where $g = 2J^T r$ and $B = 2J^T J$. This method has many advantages over the naive Newton's method, as it eliminates the trouble of computing the Hessian $\nabla^2 r_j$. Nevertheless, this approximation also brings some problems. When the term $J^T J$ can not dominate the second-order term in the Hessian, the speed of convergence will be rather slow. Many algorithms use Gauss-Newton to optimize the pose graph in SLAM, such as square root SAM [5].

3.2 Trust Region Methods

As same as line search methods, trust region methods generate steps with the help of an quadratic model (7) of the cost function, but they use this model in different ways. They adjust a region to acquire an adequate representation of the cost function, and then choose the step to arrive at the approximate minimum of the model in this region.

$$\min\ q(d) = f + g^T d + \frac{1}{2} d^T B d \quad \text{s.t.} \quad \|d\| \leq \Delta \tag{8}$$

where Δ is the region where quadratic model is approximate adequately to the cost function at current step. Levenberg-Marquardt and dogleg method are popular way to address this constrained optimization for SLAM.

3.2.1 Levenberg-Marquardt

The Lagrange multiplier method is a simple and efficient way to solve the constrained optimization (8), and this is the key idea of Levenberg-Marquadt method. And thus the search step [11]

$$(J^T J + \lambda I)d = -J^T r \tag{9}$$

where λ controls the size of trust region. This method combines the advantages of gradient descent and Gauss-Newton. If λI dominates the left term, it is approximate to gradient descent. Otherwise, it is similar to Gauss-Newton method. This approximation can prevent the divergence caused by ill-conditioned $J^T J$.

3.2.2 Dogleg

Powell's dogleg method is one of the most popular trust region methods. It also combines the steepest descent steps and the Gauss-Newton steps in order to obtain the minimum in trust region [13].

$$d_{dl}(t) = \begin{cases} t d_{gd} & 0 \leq t \leq 1 \\ d_{gd} + (t-1)(d_{GN} - d_{gd}) & 1 \leq t \leq 2 \end{cases} \tag{10}$$

where d_{gd} is the gradient descent step, and d_{GN} is the Gauss-Newton step. This method can guarantee $q(d)$ to be a good approximation for f at current step. The different between dogleg and Levenberg-Marquardt is the intensity of the region's constraints. Trust region is a hard constraint for dogleg while it is a soft constraint for Levenberg-Marquardt. Thus dogleg method is robust to highly nonlinear systems as well as numerically ill-conditioned Jacobians.

3.3 Approaches to Linear System

As what we have described above, wether Gauss-Newton, Levenberg-Maquardt or dogleg needs to solve a linear system. When the number of poses is large, Sparse matrix factorization methods are popular approaches for this problem, such as Cholesky factorization and QR factorization. Cholesky calculates upper triangular matrix R that satisfies

$$R^T R = J^T J. \tag{11}$$

It has a significant disadvantage since the condition number of $J^T J$ is larger than the condition number of J, and this leads to relative error increasing. The second QR aims to obtain a orthogonal transformation matrix Q and a upper triangular matrix R that satisfies

$$J = QR. \tag{12}$$

QR is more stable than Cholesky, as it is based on the condition number of J.

3.4 Approaches to Robust Optimization

The robust optimization has two implications. One means that the algorithm is free from the outliers of loop closures [15], another is for noise of sensors. When the robot comes back to a place where it had arrived before, the pose graph will construct a constraint between the current pose and the previous pose. And this constraint is named loop closure [16]. The approaches mentioned above are based on perfect loop closures. In practice, with unexpected noise and error, the pose graph is always accompanied with false positive constraints.

A direct method addressing this problem is weighting each constraint

$$X^* = \arg\min \sum_{i=1}^{T} \|f_i(x_{i-1}, u_i) - x_i\|^2_{\Omega_{u_i}} + \sum_{k=1}^{K} \omega_k \|h_k(x_k) - z_k\|^2_{\Omega_{z_k}} \tag{13}$$

where ω_k is a weight for k-th association. The value of the weight is either 0 or 1, and 0 means the constraint is inconsistent, vice versa. As this discrete weight is non differentiable, Sünderhauf and Protzel used sigmoid function to approximate this discrete value, and proposed switchable constraints(SC) method. According to [7], the optimization problem had to introduce additional constraints for these switch variables in order to derive the solutions through optimization. To avoid increasing the scale of optimization by introducing additional constraints, Agarwal *et al.* used dynamic covariance scaling(DCS) to adjust this switch variable dynamically by estimating the current constraint error [8].

When noise of sensors is conspicuous and the initial guess for optimization is bad, the performances of algorithms will change dramatically. As mentioned above, Olson *et al.* used stochastic gradient descent to solve this optimization problem, and this method is robust to wrong initial guess. However, many methods mentioned above speed up the convergence speed of the stochastic gradient descent do not have this advantage. In order to enhance their robustness, it is common to use the robust cost function such as Huber kernel[17]. Huber kernel is defined as

$$\rho_H(x) = \begin{cases} x^2 & \text{if } |x| < b \\ 2b|x| - b^2 & \text{else} \end{cases}$$

where b is a parameter. In SLAM problem, x is defined as $\sqrt{e_j^T \Omega_j e_j}$. When the error is large, this kernel is linear

Table 1: The datasets used in experiments

Dataset	Poses	Constraints	Initial RMSE
City10000	10000	20687	37.2026
M10000	10000	64311	*
manhattan3500	3500	5453	22.4383

Note: * means that we don't have the ground truth of the manhattan3500 dataset.

function, thus it can relieve the nonlinear increase of the error function when the initialization is bad.

4 EXPERIMENTS

In order to compare the above methods that implemented in different libraries, we conduct sets of experiments on the synthetic datasets. We consider three popular software libraries, including g2o, GTSAM and TORO. All experiments are conducted on an Inter Core i5 CPU.
We use publicly available datasets to support our comparisons. Since the datasets are synthetic, we can change the noise of data and the number of loop closure's outliers to compare the performance of different algorithms. Table 1 lists the detail of datasets.

4.1 Accuracy Metrics

During the experiments, we use RMSE(root mean squared error) between the estimated trajectories and the ground truthes [18][15] and χ^2 [19][20] as the performance metrics for accuracy evaluation. Those metrics are defined as

$$RMSE = \sqrt{\frac{1}{T}\sum_{i=1}^{T} \|\hat{x}_i \ominus x_i\|^2} \tag{14}$$

$$\chi^2 = \sqrt{\frac{1}{K}\sum_{k=1}^{K} \|\hat{z}_k \ominus z_k\|^2} \tag{15}$$

where $x_i \ominus x_j$ means the motion estimation from x_j to x_i. $\hat{x}$ is the estimation of the pose and $\hat{z}$ is the estimation of the measurement. Another simple metric of the convergence is the cost of the cost function (2). We also use this cost for analysing the convergence.

4.2 Batch Experiments

In batch experiments, all poses of the robot and measurements are fed to the algorithms at once. In this section we combine the iterative methods such as Levenberg-Marquardt(LM) and dogleg(dl) with matrix factorization algorithms such as Cholesky and QR. In g2o library, there are two high-efficient implementation of Cholesky, namely CHOLMOD and CSparse. But g2o library does not have implementation of QR. GTSAM library has itself implementations of Cholesky and QR. Stochastic gradient descent(SGD) only is implemented in TORO library. Table 2 summarizes these combinations. Then we add the Huber kernel to the best combination. The Huber kernel is piecewise function of the error. When the error is large, this kernel is linear function, thus it can relieve the nonlinear increase of the error function when the initialization is bad. In addition, we also consider robust kernels such as SC and DCS in order to find out the most robust methods for pose graph optimization.

Table 2: The combination algorithms and software libraries used in the convergence experiments

	g2o		GTSAM		TORO
	CHOLMOD	CSparse	Cholesky	QR	*
LM	LM+cholmod	LM+CSparse	LM+Cholesky	LM+QR	*
dogleg	dl+cholmod	dl+CSparse	dl+Cholesky	dl+QR	*
SGD	*	*	*	*	SGD

Note: * means that the corresponding software libraries doesn't include this combination.

Table 3: The RMSE of the algorithms on city10000

	(0.1, 0.05)	(0.2, 0.1)	(0.4, 0.2)	(0.6, 0.3)
LM	1.4123	2.8648	5.8458	8.9165
LM+Huber	1.6208	**2.7242**	3.9185	5.0286
dl	1.4932	2.9469	5.9109	9.0185
dl+Huber	1.6208	**2.7242**	**3.9184**	**5.0285**
SGD	**1.3973**	2.9284	8.3566	44.9621

Note: (x,y) means that x is the noise variance of position and y is the noise variance of orientation. It's same as Table 4.

4.2.1 Convergence Speed

Figure 1 depicts the comparison results of the convergence. Dogleg in g2o is satisfactory, but in GTSAM, dogleg is instable. For M10000 dataset and manhattan3500 dataset, dogleg in GTSAM is faster than others methods, while this method do not converge on city10000 dataset. For factorization algorithms, CHOLMOD has a slight advantage than CSparse on big datasets, but it is in reverse on small dataset. In GTSAM, QR is more stable than Cholesky, since dogleg with Cholesky fails on M10000 dataset. SGD has a good performance on city10000 dataset, which benefits from constant time per iteration with increasing scale of problem.

4.2.2 Sensitivity to Noise

In this set of experiments, according to the literature [4], we add four sets of Gaussian noise to initial poses and constraint measurements on city10000 dataset and manhattan3500 dataset. We add the Huber kernel to the combinatorial algorithms which have good performance in last set of experiments. These algorithms include LM with CHOLMOD in g2o corresponding to LM in Table 3 and Table 4, as well as dogleg with CHOLMOD in g2o corresponding to dl in Table 3 and Table 4.
The noise obviously affects the convergence speed of stochastic gradient descent when the dataset is large. This influence does not happen on LM and dogleg in g2o. Dogleg method is still faster than LM. The results is illustrated in Figure 2. The Huber kernel can not improve the convergence speed obviously, but it can increase the accuracy of the algorithms according to Table 3 and Table 4. LM has similar accuracy to dogleg during different noise, and it is little better than dogleg in the case of small datasets such as manhattan3500.

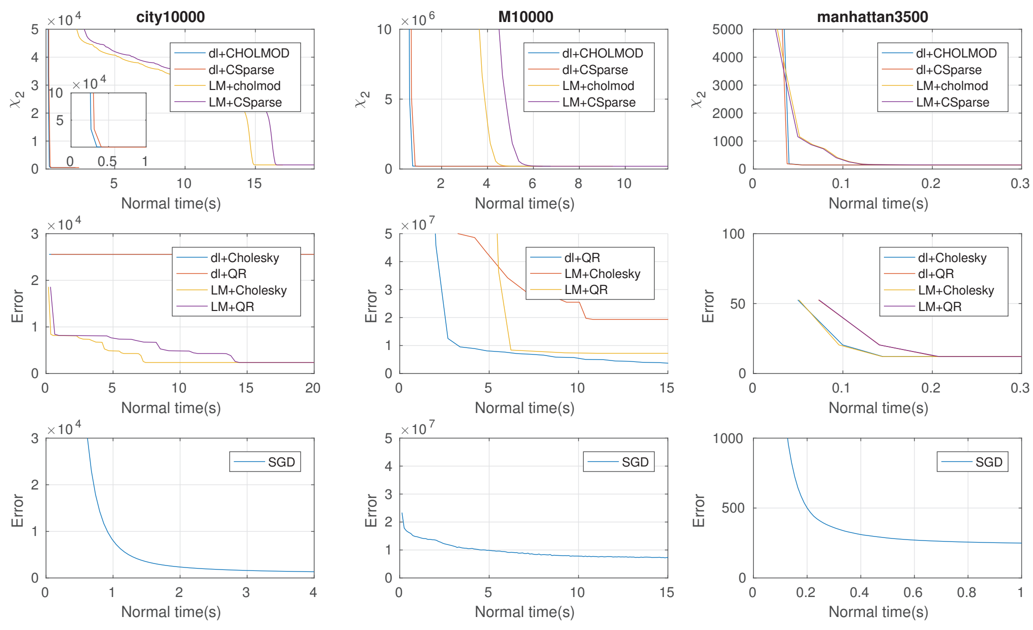

Figure 1: The convergence speed of algorithms listed in Table 2 on three datasets. Top row: χ^2 of four algorithms in g2o. Median row: the function error of four algorithms in GTSAM. The results on M10000 dataset do not have dl+Cholesky, since this algorithm fails in factorization. Bottom row: the cost of the function (2) for stochastic gradient descent in TORO.

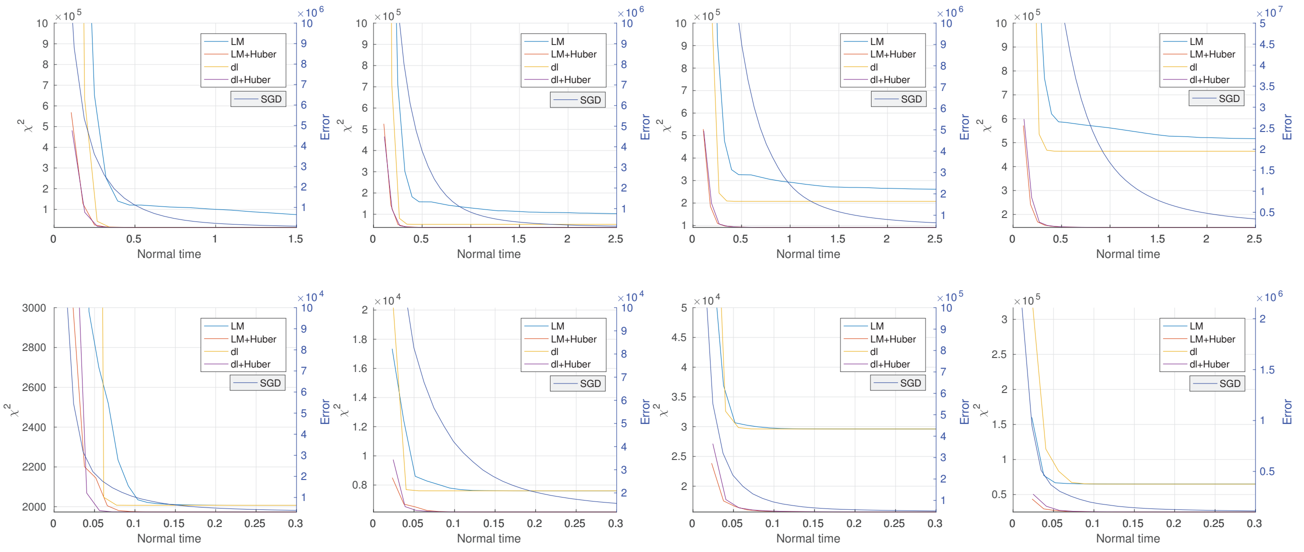

Figure 2: Comparison of convergence speed of algorithms on two datasets with four different variances of noise, and the variances increase from left to right for each row in the figure. Noise variances are listed in Table 3. and Table 4. Top row: comparison on city100000 dataset. Bottom row: comparison on manhattan3500 dataset.

Table 4: The RMSE of the algorithms on manhattan3500

	(0.1, 0.05)	(0.2, 0.1)	(0.4, 0.2)	(0.6, 0.3)
LM	2.0136	5.0197	11.3636	17.4279
LM+Huber	**1.8508**	**3.9227**	**8.4931**	**10.576**
dl	2.0136	5.0197	11.3636	17.4279
dl+Huber	**1.8508**	3.9229	8.5503	10.68
SGD	2.4561	6.4192	19.387	15.6396

4.2.3 Robustness to Outliers

In order to compare the robustness to outliers, we introduce 7 sets of false positive loop closures to datasets. The number of these sets is different from 100 to 2000. Since Agarwal *et al.* compared the time-consuming between SC and DCS, and pointed out DCS is time-efficient. Thus we only compare the accuracy of the two algorithms with the increase of outliers.

As expected, the combinatorial methods without any robust kernel has large RMSE. Robust kernels can reduce the error caused by outliers. And the accuracy of DCS is better than SC as shown in Figure 3. The accuracy of both algorithms will not be reduced when the outliers increase.

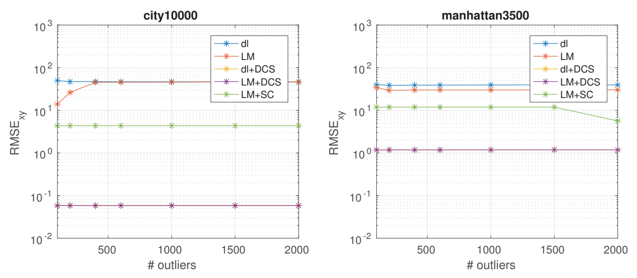

Figure 3: The RMSE error of pose position for increasing number of outliers. Left: the RMSE of the algorithms on city10000. Right: The RMSE of the algorithms on manhattan3500.

5 CONCLUSION AND DISCUSSION

In this paper, we summarize graph based optimization algorithms in general optimization framework and conduct sets of experiments. It is hard to report the best method among the evaluated algorithms under all conditions. It is decided on the properties of problem we faced.

In batch problem, dogleg method in g2o is a good choice. If datasets are more than 10000 poses, CHOLMOD for linear system is highly efficient. And when datasets are small, CSparse is more appropriate. The Huber kernel for constraints can reduce the influence caused by the noise on estimated trajectories. And DCS can be added into the methods to guarantee the reliable results despite of false positive loop closures.

In batch experiments, DCS will fail to get reliable results when the initial noise is large. And the Huber kernel cannot estimate better results under false positive loop closures. How to combine DCS and the Huber kernel to break the embarrassing situation is a worthwhile direction of future research.

ACKNOWLEDGMENT

We thank E.Olson for M10000 dataset and TORO software libraries, M.Kaess for city10000 dataset, R.Kummerle for the manhattan3500 dataset and g2o software libraries. We also thank F.Dellaert and his team for GTSAM software libraries.

REFERENCES

[1] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based slam,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[2] S. J. Julier and J. K. Uhlmann, “A counter example to the theory of simultaneous localization and map building,” in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 4. IEEE, 2001, pp. 4238–4243.

[3] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, “isam2: Incremental smoothing and mapping using the bayes tree,” *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[4] E. Olson, J. Leonard, and S. Teller, “Fast iterative alignment of pose graphs with poor initial estimates,” in *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*. IEEE, 2006, pp. 2262–2269.

[5] F. Dellaert and M. Kaess, “Square root sam: Simultaneous localization and mapping via square root information smoothing,” *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[6] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, “Efficient sparse pose adjustment for 2d mapping,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 22–29.

[7] N. Sünderhauf and P. Protzel, “Towards a robust back-end for pose graph slam,” in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1254–1261.

[8] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust map optimization using dynamic covariance scaling,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 62–69.

[9] F. Lu and E. Milios, “Globally consistent range scan alignment for environment mapping,” *Autonomous robots*, vol. 4, no. 4, pp. 333–349, 1997.

[10] G. Grisetti, C. Stachniss, and W. Burgard, “Nonlinear constraint network optimization for efficient map learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 428–439, 2009.

[11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g 2 o: A general framework for graph optimization,” in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 3607–3613.

[12] M. Kaess, A. Ranganathan, and F. Dellaert, “isam: Incremental smoothing and mapping,” *IEEE Transactions on Robotics*, vol. 24, no. 6, pp. 1365–1378, 2008.

[13] D. M. Rosen, M. Kaess, and J. J. Leonard, “Rise: An incremental trust-region method for robust online sparse least-squares estimation,” *IEEE Transactions on Robotics*, vol. 30, no. 5, pp. 1091–1108, 2014.

[14] G. N. Vanderplaats, *Numerical optimization techniques for engineering design: with applications*. McGraw-Hill College, 1984.

[15] Y. Latif, C. Cadena, and J. Neira, “Robust graph slam backends: A comparative analysis,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 2683–2690.

[16] J. Aulinas, Y. R. Petillot, J. Salvi, and X. Lladó, “The slam problem: a survey.” in *CCIA*. Citeseer, 2008, pp. 363–371.

[17] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on information theory*, vol. 47, no. 2, pp. 498–519, 2001.

[18] L. Carlone, A. Censi, and F. Dellaert, “Selecting good measurements via ? 1 relaxation: A convex approach for robust estimation over graphs,” in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 2667–2674.

[19] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, “On measuring the accuracy of slam algorithms,” *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[20] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 573–580.