A Comparison of Graph Optimization Approaches for Pose Estimation in SLAM

Anđela Jurić*,[1], Filip Kendeš*, Ivan Marković*, Ivan Petrović*

* University of Zagreb, Faculty of Electrical Engineering and Computing, Zagreb, Croatia
{andela.juric, filip.kendes, ivan.markovic, ivan.petrovic}@fer.hr

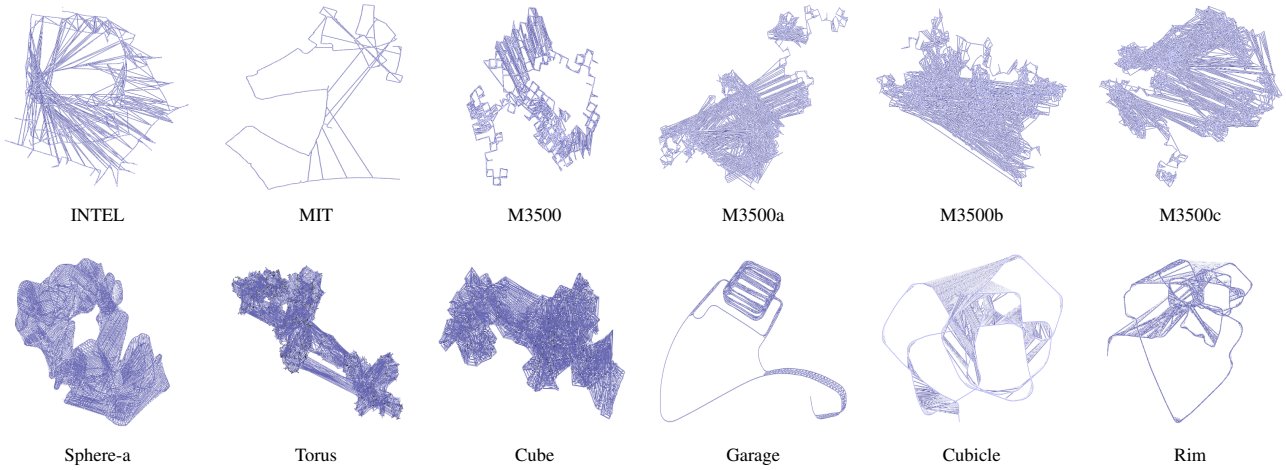


Fig. 1: Two-dimensional (first row) and three-dimensional (second row) pose graphs used in the benchmarking process.

***Abstract*—Simultaneous localization and mapping (SLAM) is an important tool that enables autonomous navigation of mobile robots through unknown environments. As the name SLAM suggests, it is important to obtain a correct representation of the environment and estimate a correct trajectory of the robot poses in the map. Dominant state-of-the-art approaches solve the pose estimation problem using graph optimization techniques based on the least squares minimization method. Among the most popular approaches are libraries such as g^{2}o, Ceres, GTSAM and SE-Sync. The aim of this paper is to describe these approaches in a unified manner and to evaluate them on an array of publicly available synthetic and real-world pose graph datasets. In the evaluation experiments, the computation time and the value of the objective function of the four optimization libraries are analyzed.**

***Index Terms*—pose-graph, optimization, trajectory estimation, SLAM, g^{2}o, GTSAM, Ceres, SE-Sync**

I. Introduction

After years of dominance in the SLAM scene, filter-based methods are more and more replaced by optimization-based approaches. Pose graph optimization (PGO) was first introduced in [1], but was not very popular due to computational inefficiency. Today, with the increase of the computational power, PGO methods have become the state-of-the-art and are able to solve SLAM optimization and estimation problems quickly and accurately.

Optimization-based SLAM methods generally consist of two parts. The first part identifies the constraints between new observations and the map using correspondences based on sensor data. The second part computes the robot poses and the map given the constraints. It can be divided into graph and smoothing methods. An example of a current state-of-the-art optimization-based method is g^{2}o [2]. It is a general optimization framework for nonlinear least squares problems. One of the first smoothing approaches, $\sqrt{\text{SAM}}$, was presented in [3]. An improvement to this method, incremental smoothing and mapping (iSAM), was introduced in [4]. iSAM extends $\sqrt{\text{SAM}}$ to provide an efficient solution to the full SLAM problem by updating the factorization of the sparse smoothing information matrix. The upgrade of iSAM, iSAM2, is presented in [5]. These smoothing methods are implemented in GTSAM [6], which is another state-of-the-art optimization library.

An elegant example of a SLAM problem is the so-called pose SLAM, which avoids building an explicit map of the environment. The objective of pose SLAM is to estimate the trajectory of the robot given the loop closing and odometric constraints. These relative pose measurements are usually obtained from IMU, laser sensor, cameras or wheel odometry using ego-motion estimation, scan matching, iterative closest point (ICP) or some form of minimizing visual reprojection error. It is worth noting that methods in [7] and [8] present filter-based pose SLAM algorithms, but our focus in the current paper is on the approaches that implement optimization-based pose SLAM.

[1]This work has been supported by the Croatian National Science Foundation under the grant no. DOK-2018-01-9392.

Among the most popular approaches are g^{2}o (*"general graph optimization"*) [2]), Ceres Solver [9], GTSAM ("Georgia Tech Smoothing and Mapping") [6], and SE-Sync ("Synchronization over Special Euclidean group SE(n)") [10]. There are few works in the literature that compare some of these approaches. For example, in [11] the authors provide an overview of visual SLAM and compare g^{2}o, GTSAM, and HOG-Man [12] as back-ends. The authors in [13] discuss the importance of rotation estimation in pose graph estimation and compare g^{2}o and GTSAM on benchmarking datasets with different rotation estimation techniques. In [14], the authors compare different optimization algorithms under g^{2}o framework. A unified SLAM framework, GLSLAM, proposed in [15], provides various SLAM algorithm implementations, but can also perform benchmarks for different SLAM approaches. However, to our knowledge, there are none that compare g^{2}o, Ceres, GTSAM and SE-Sync in a unified manner. The aim of this paper is to describe these approaches in a unified manner and to evaluate them on an array of publicly available synthetic and real-world pose graph datasets (visualized in Figure 1). In the future, we would like to use this comparison to facilitate the choice of the PGO method.

The paper is organized as follows. Section II describes nonlinear graph optimization in general and each of these four approaches. The experiment is the main part of this paper and is described in Section III. In the first part of the section, the hardware, the experimental setup, and the benchmarking datasets are described. The results of the experiment are discussed at the end of the section. In the end, Section IV concludes the paper.

II. Nonlinear Pose-Graph Optimization Approaches

Every pose graph consists of nodes and edges. The nodes in a pose graph correspond to the poses of the robot in the environment, and the edges represent spatial constraints between them. The edges between contiguous nodes are odometric constraints, and the remaining edges represent loop closing constraints. This is visualized in Figure 2a. The goal of pose graph optimization is to find a configuration of nodes that minimizes the least squares error over all constraints in the pose graph. In general, a nonlinear least squares optimization problem can be defined as follows:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, \mathbf{F}(\mathbf{x}), \tag{1}$$

where $\mathbf{F}(\mathbf{x})$ is the sum of errors over all constraints in the graph:

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle i,j\rangle \in \mathcal{C}} \mathbf{e}_{ij}^\top \mathbf{\Omega}_{ij} \mathbf{e}_{ij}. \tag{2}$$

Here, $\mathcal{C}$ represents the set of index pairs between connected nodes, $\mathbf{\Omega}_{ij}$ represents the information matrix between nodes i and j, and $\mathbf{e}_{ij}$ is the nonlinear error function that models how well the poses $\mathbf{x}_i$ and $\mathbf{x}_j$ satisfy the constraint imposed by the measurement $\mathbf{z}_{ij}$. Finally, each constraint is modeled with the information matrix $\mathbf{\Omega}_{ij}$ and the error function $\mathbf{e}_{ij}$. This is illustrated in Figure 2b.

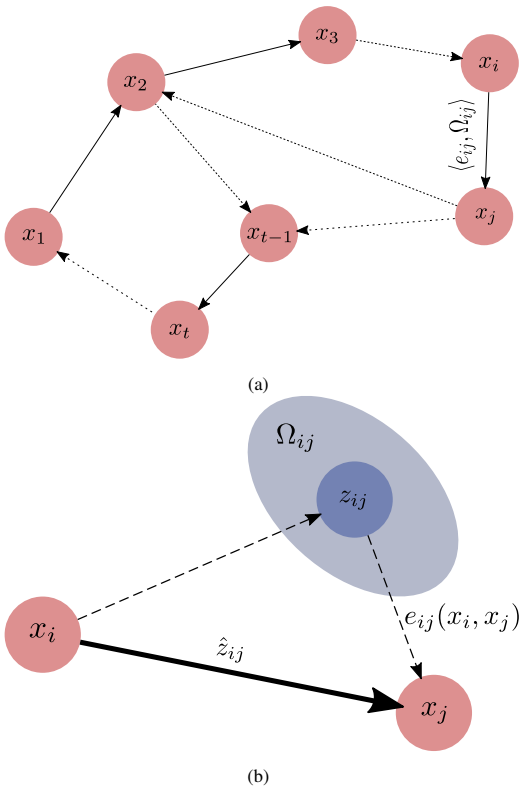


Fig. 2: A pose-graph representation of the SLAM problem. (a) Every node corresponds to a robot pose. Neighboring poses are connected with edges. The edges model spatial constraints between two poses. The edges between contiguous poses represent odometry and remaining edges represent repeated observation of the same part of the environment (loop closures). (b) Each edge is defined with its error function $\mathbf{e}_{ij}$ and information matrix $\mathbf{\Omega}_{ij}$. Error function is the difference between the real measurements z_{ij} and the approximated constraints $\hat{z}_{ij}(x_i, x_j)$.

Traditionally, the solution to (1) is obtained by iterative optimization techniques (e.g., Gauss-Newton or Levenberg-Marquardt). Their idea is to approximate the error function with its first-order Taylor expansion around the current initial guess. In general, they consist of four main steps:

1) Fix an initial guess.
2) Approximate the problem as a convex problem.
3) Solve 2) and set it as a new initial guess.
4) Repeat 2) until convergence.

Pose SLAM is easier to solve because it does not build a map of the environment. Graph formulated problems have sparse structure, so computation is faster. Another advantage is that it is robust to bad initial guesses. The disadvantage of pose SLAM is that it is generally not robust to outliers and does not converge when there are many false loop closures. Moreover, rotation estimation makes it a hard noncovex optimization problem, so convex relaxation leads to problems with local minima and there is no guarantee of a global optimum. In this section, we briefly describe the optimization frameworks based on the nonlinear least squares method that provide solutions in the form of pose graphs.

A. g^2o

g^{2}o [2] is an open-source general framework for optimization of nonlinear functions that can be defined as graphs. Its advantages are that it is easily extensible, efficient, and applicable to a wide range of problems. The authors state in [2] that their system is comparable to other state-of-the-art algorithms while being highly general and extensible. They achieve efficiency by exploiting sparse connectivity and the special structure of the graph, using advanced methods to solve sparse linear systems, and utilizing the features of modern processors. The framework contains three different methods that solve PGO, Gauss-Newton, Levenberg-Marquardt and Powell's Dogleg. It is mainly used to solve the SLAM problem in robotics and the bundle adjustment problems in computer vision. ORB-SLAM ([16], [17]) uses g^{2}o as a back-end for camera pose optimization, and SVO [18] uses it for visual odometry.

B. Ceres

Ceres Solver [9] is an open-source C++ library for modeling and solving large, complicated optimization problems. It is mainly dedicated to solving nonlinear least squares problems (bundle adjustment and SLAM), but can also solve general unconstrained optimization problems. The framework is easy to use, portable, and extensively optimized to provide solution quality with low computation time. Ceres is designed to allow the user to define and modify the objective function and optimization solvers. The solvers that are implemented include trust region solvers (Levenberg-Marquardt, Powell's Dogleg) and line search solvers. Since it has many advantages, Ceres is used in many different applications and domains. OKVIS ([19], [20]) and VINS [21] use Ceres to optimize nonlinear problems defined as graphs.

C. GTSAM

GTSAM [6] is another open-source C++ library that implements sensor fusion for robotics and computer vision applications. It can be used to solve optimization problems in SLAM, visual odometry, and structure from motion (SfM). GTSAM uses factor graphs [22] to model complex estimation problems and exploits their sparsity to be computationally efficient. It implements Levenberg-Marquardt and Gauss-Newton style optimizers, the conjugate gradient optimizer, Dogleg, and iSAM: inceremental smoothing and mapping. GTSAM is used alongside various sensor front-ends in academia and industry. For instance, there is a variant of SVO [23] that uses GTSAM as a back-end for visual odometry.

D. SE-Sync

SE-Sync [10] is a certifiably correct algorithm for performing synchronization over the special Euclidean group. It's objective is to estimate the values of a set of unknown poses (positions and orientations in Euclidean space) given noisy measurements of relative transformations between nodes. Their main applications are in the context of 2D and 3D geometric estimation. For example, pose-graph SLAM (in robotics), camera motion estimation (in computer vision), and sensor network localization (in distributed sensing). Authors state in [10] that SE-Sync improves on previous methods by exploiting a novel (convex) semidefinite relaxation of the special Euclidean synchronization problem to directly search for globally optimal solutions, and is able to generate a computational certificate of correctness for the found solution. They apply truncated-Newton Riemannian Trust-Region method [24] to find efficient estimates of poses.

III. Experiments

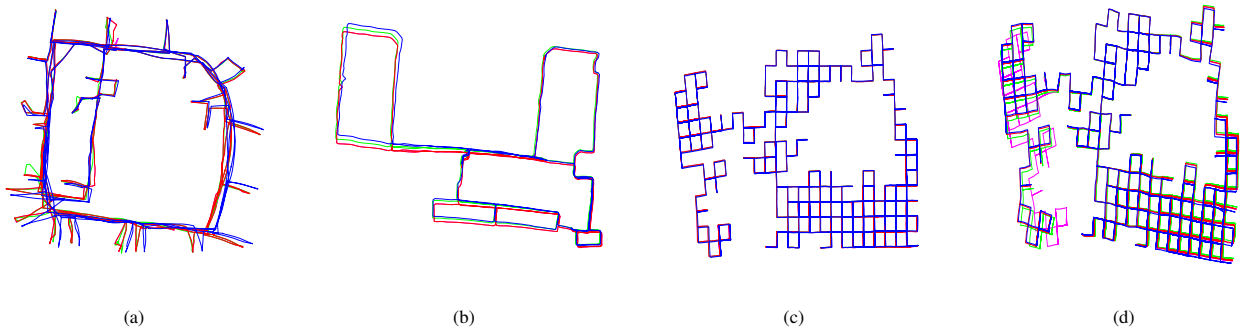
Our goal is to experimentally evaluate the optimization frameworks described in Section II and compare them. For this purpose, we are considering their performance in terms of total computation time and final value of the objective function described by (2). We use publicly available synthetic and real-world benchmarking datasets.

A. Experimental setup

The experiment was conducted on a Lenovo ThinkPad P50 equipped with an octa-core Intel Core i7-6700HQ CPU operating at 2.60 GHz and 16 GB RAM. The computer is running Ubuntu 20.04. The same solver, Levenberg-Marquardt, is chosen for the g^{2}o, Ceres and GTSAM frameworks, while SE -Sync uses the Riemannian trust-region (RTR) method ([24]). Each algorithm is limited to a maximum of 100 iterations. The stopping criteria are based on reaching the maximum number of iterations and the relative error decrease. SE-Sync uses a slightly different method, so an additional criterion based on the Riemann gradient norm must be specified. The tolerance for the relative error decrease is set to 10^{-5} and the norm of the gradient is set to 10^{-2}. The parameters are chosen according to the recommendations in [10]. The authors in [13] study the influence of orientation initialization on finding the global optimum. Inspired by this work, we also do the pose graph initialization before the optimization process. To achieve better results, we obtain initial pose graphs using the spanning tree method [2].

B. Benchmarking datasets

There are six two-dimensional pose graphs obtained from [25], two real-word graphs, and four graphs created in simulation. **INTEL** and **MIT** pose graphs are real-world datasets created by processing raw wheel odometry and laser range sensor measurements collected at the Intel Research Lab in Seattle and MIT Killian Court. **M3500** pose graph is a simulated Manhattan world [26]. The **M3500a**, **M3500b**, and **M3500c** datasets are variants of the **M3500** dataset with Gaussian noise added to the relative orientation measurements. The standard deviation of the noise is 0.1, 0.2, and 0.3 rad, respectively. There are also six three-dimensional datasets obtained from [13]. The pose graphs **Sphere-a**, **Torus**, and **Cube** are created in simulation. **Sphere-a** dataset is a challenging problem released in [27]. The other three pose graphs are real-world datasets. The **Garage** dataset was introduced



(a) (b) (c) (d)

Fig. 3: Optimized pose graphs from (a) INTEL, (b) MIT, (c) M3500 and (d) M3500a datasets. Results are color coded for each algorithm: pink - g^2o, green - Ceres, red - GTSAM, blue - SE-Sync.

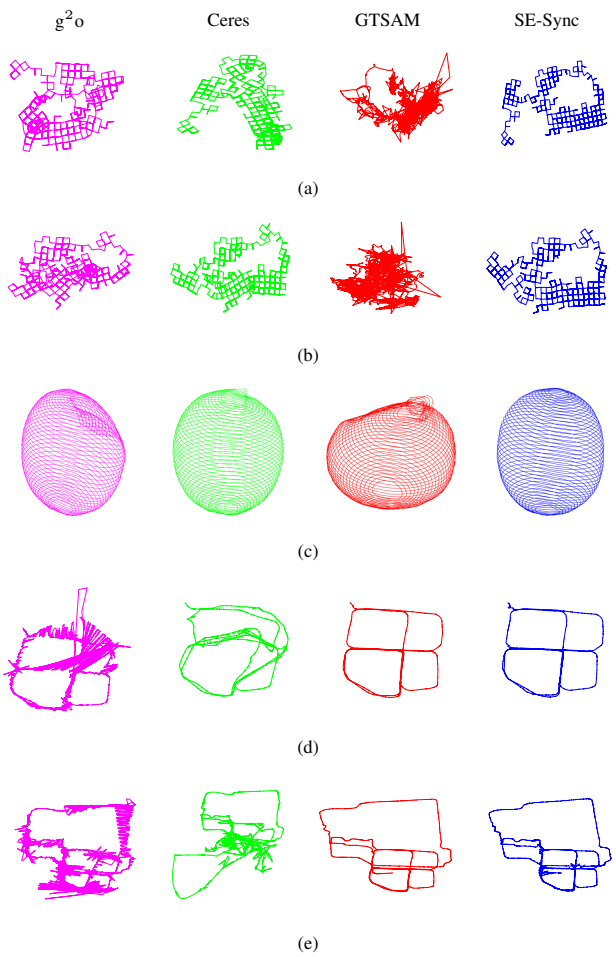


Fig. 4: Results on five challenging datasets: (a) M3500b, (b) M3500c, (c) Sphere-a, (d) Cubicle, (e) Rim for each algorithm. From left to right: g^2o (pink), Ceres (green), GTSAM (red), SE-Sync (blue). SE-Sync shows superior results, but GTSAM performs just as well on Cubicle and Rim dataset.

in [28], and **Cubicle** and **Rim** are acquired using ICP on point clouds from the 3D laser sensor at the RIM center at Georgia Tech. All these pose graphs are visualized in Fig. 1 with their odometric and loop-closing constraints. Table I contains the number of nodes and edges for each dataset. These numbers determine the number of parameters in the optimization process and the complexity of the problem.

TABLE I: Pose-graph datasets with number of nodes and edges

Dataset		Nodes	Edges
Sphere-a	3D	2200	8647
Torus	3D	5000	9048
Cube	3D	8000	22236
Garage	3D	1661	6275
Cubicle	3D	5750	16869
Rim	3D	10195	29743
Intel	2D	1228	1483
MIT	2D	808	827
M3500	2D	3500	5453
M3500a	2D	3500	5453
M3500b	2D	3500	5453
M3500c	2D	3500	5453

C. Results

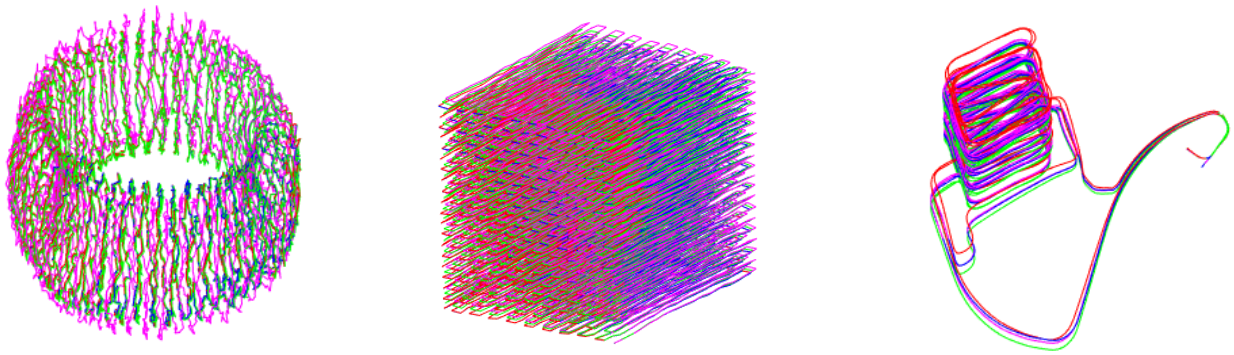
We summarized all the performance results in Table II in terms of total computation time and the value of the objective function (2). For each dataset, we state the termination reason of the algorithm. The algorithm converges if it completes the optimization within the maximum iteration limit. We also give the validation time for SE-Sync to certify the global optimum. Figures 3, 4 and 5 show optimized pose graphs for the benchmarking problems.

1) INTEL: INTEL is one of the easiest problems to solve. All approaches have solved it successfully. The trajectories are shown together in Figure 3a and it can be seen that all of them achieved a similar result. GTSAM took the longest time to finish the optimization, but achieved the lowest objective function value. SE-Sync is the fastest in this case and has the value slightly larger than GTSAM. Considering this, SE -Sync seems to be the best solution.

2) MIT: MIT is the smallest problem, but has only a few loop closure constraints. Therefore, it is important to start the optimization with a good initial guess. Otherwise, GTSAM, Ceres, and g^2o are not able to converge to a meaningful solution with the Levenberg-Marquardt algorithm. All algorithms converged in less than the maximum number of iterations and they achieved almost the same objective function values. The optimization problem is solved in less than half a second, but Ceres and g^2o are the fastest. Ceres also achieved the lowest objective function value and seems to be the best solver for the dataset MIT. The final trajectories can be seen in Figure 3b.

TABLE II: Optimization times and objective values for each algorithm and dataset are organized in the table. Total time corresponds to elapsed optimization time, and F(x) to the value of the objective function. Termination reason for each dataset is given: maximum number of iterations reached (iter), relative function decrease (conv), divergence (no conv) and maximum level of Riemannian staircase (r max). Validation time for SE-Sync is the time it takes for the algorithm to check the optimality of the solution. The minimum optimization time is marked in red, and the lowest objective value is written in green. The blue color corresponds to the other minimum objective values with the poor visual representations of the trajectory, or the same as the value marked in green.

		INTEL	MIT	M3500	M3500a	M3500b	M3500c	Sphere-a	Torus	Cube	Garage	Cubicle	Rim
g^2o	total time (s)	0.28	0.07	0.66	0.86	1.90	2.08	8.07	10.66	599.05	1.53	338.06	243.12
	F(x)	6.17E+04	4.12E+01	1.38E+02	9.12E+02	9.27E+03	6.61E+03	8.58E+05	1.46E+04	4.92E+04	1.24E+00	9.38E+07	1.45E+09
	termination	iter	conv	conv	conv	iter	iter	conv	conv	conv	conv	iter	iter
Ceres	total time (s)	0.04	0.06	0.23	0.31	1.56	0.76	18.87	2.04	62.59	0.73	25.66	49.82
	F(x)	6.40E+05	1.89E+01	6.90E+01	4.56E+02	2.88E+03	1.86E+03	1.56E+06	1.21E+04	4.22E+04	6.34E-01	7.26E+06	1.01E+08
	termination	conv	conv	conv	conv	iter	conv	iter	conv	conv	conv	iter	iter
GTSAM	total time (s)	21.55	0.16	0.41	1.23	3.88	1.30	12.16	2.34	113.89	0.66	2.54	19.41
	F(x)	1.14E+02	2.06E+01	6.90E+01	4.56E+02	3.60E+07	5.33E+06	3.30E+06	1.21E+04	4.22E+04	6.34E-01	1.36E+03	4.11E+04
	termination	iter	conv	conv	conv	no conv	no conv	conv	conv	conv	conv	conv	conv
SE-Sync	total time (s)	0.01	0.15	1.57	0.24	0.37	0.56	0.15	0.44	8.53	1.07	5.58	4.21
	F(x)	1.97E+02	3.06E+01	9.69E+01	7.99E+02	1.84E+03	2.29E+03	1.48E+06	1.21E+04	4.22E+04	6.31E-01	3.59E+02	2.73E+03
	validation (s)	0.165	1.749	15.716	1.426	52.891	10.202	0.188	0.51	14.452	5.521	12.052	35.851
	termination	conv	conv	conv	conv	r max	conv	conv	conv	conv	conv	conv	conv



(a) (b) (c)

Fig. 5: Final trajectories for (a) Torus, (b) Cube and (c) Garage dataset. Color codes are the same as in Figure 3.

3) M3500: All four variants of the M3500 dataset are presented here together. GTSAM and Ceres are the best at solving the basic M3500 problem, as both achieve the lowest objective function value, but Ceres is slightly faster. All approaches were successful in solving the problem, and their optimized pose graphs are visualized in Figure 3c. M3500a is a more difficult problem due to the noise added to the relative orientations. Nevertheless, all approaches solved it, but as it can be seen in Figure 3d, the g^2o solution deviates from other solutions. Ceres and GTSAM converged to the lowest value, but again Ceres is faster. M3500b and M3500c are very challenging problems because the amount of noise is high. GTSAM does not converge in these two cases, and Ceres and g^2o get stuck in a local minimum. SE-Sync is the most successful approach in solving these two variants of the problem. Considering the lack of loop closure constraints connecting the left and right parts of the pose graph, and the amount of noise in the measurements, SE-Sync has achieved a reasonably good solution. These pose graphs can be seen in Figures 4a and 4b.

4) Sphere-a: The Sphere-a problem is also challenging because high amount of noise is added to the relative orientation measurements. Even with an initial guess, only SE-Sync was able to solve it. It converges to the global minimum in only 0.15 s which is 50 times faster than g^2o and 100 times faster than Ceres and GTSAM. The optimal solution by SE-Sync is shown in Figure 4c with local optima obtained by the other three approaches.

5) Torus: Torus is one of the easiest 3D problems. All approaches except g^2o converge to the global optimum, but again SE-Sync wins in speed. g^2o solves the problem, but looking at Figure 5a it can be seen that its optimized trajectory drifts slightly in comparison to the other three.

6) Cube: The cube dataset is more complex and time consuming due to the large number of nodes and edges, but all approaches have found a solution. SE-Sync is by far the fastest in solving this problem. It takes 8 seconds, while others take more than a minute. Final solutions are shown in Figure 5b.

7) Garage: The Garage dataset is the smallest and easiest 3D dataset to solve, for which each approach achieved a solution. This is illustrated in Fig. 5c. Ceres, GTSAM and SE-Sync converge to the same objective value, and GTSAM is the fastest in this case.

8) Cubicle and Rim: The Cubicle dataset is a subset of the Rim dataset, so we discuss them together. These two datasets are challenging because they both contain large number of nodes and edges. Moreover, large number of edges are false loop closures. Only GTSAM and SE-Sync converged to a solution in both cases. The pose

graphs after optimization are visualized in Figures 4d and 4e. Ceres and g^{2}o are unable to find a solution. GTSAM solves the Cubicle two times faster than SE-Sync, but SE-Sync converges to the global optimum. SE-Sync optimizes the Rim five times faster than GTSAM and converges to the global optimum.

IV. Conclusion

In this paper, we compared graph optimization methods used for pose estimation in SLAM. We considered g^{2}o and GTSAM, which are current state-of-the-art approaches, Ceres, a user-friendly open-source framework, and SE-Sync, a novel and robust method for pose synchronization. The evaluation process takes into account the elapsed optimization time and the value of the objective function, and the results are given in the form of a table for twelve benchmarking datasets.

SE-Sync achieved the smallest total time on most datasets when compared to other three methods. It is also able to validate the certificate of global optimality if required, but at the cost of additional computation time. g^{2}o had the highest total time but performed well on simple 2D datasets. Ceres is easy to use, offers a lot of flexibility, and is relatively fast. GTSAM performs almost as well as SE-Sync, except on very noisy 2D datasets. Paired with a proper front-end, these methods can be very powerful in solving SLAM problems. For poor data association, high noise, and a poor performing front-end, it seems best to use SE-Sync as the back-end. With a good initialization method, GTSAM seems to do an equally good job. If the front-end is excellent, dataset relatively simple, or the noise is low, it is a matter of personal preference to decide between these back-ends. We hope this comparison can help other researchers in choosing a back-end method for their SLAM applications.

References

[1] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[2] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613.

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

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

[5] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "ISAM2: Incremental smoothing and mapping using the Bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[6] F. Dellaert, V. Agrawal, A. Jain, M. Sklar, and M. Xie, "GTSAM," https://gtsam.org.

[7] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly Sparse Delayed-State Filters for View-Based SLAM," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

[8] K. Lenac, J. Ćesić, I. Marković, and I. Petrović, "Exactly sparse delayed state filter on Lie groups for long-term pose graph SLAM," *International Journal of Robotics Research*, vol. 37, no. 6, pp. 585–610, 2018.

[9] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[10] D. Rosen, L. Carlone, A. Bandeira, and J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special Euclidean group," *Intl. J. of Robotics Research*, vol. 38, no. 2–3, pp. 95–125, Mar. 2019.

[11] D. M. a. Latif, M. a. Megeed Salem, H. Ramadan, and M. I. Roushdy, "Comparison of Optimization Techniques for 3D Graph-based," *Proceedings of the 4th European Conference of Computer Science (ECCS '13) Recent Advances in Information Science*, p. 288, 2013.

[12] G. Grisetti, R. Kümmerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2d and 3d mapping," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 273–278.

[13] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, no. June. Institute of Electrical and Electronics Engineers Inc., jun 2015, pp. 4597–4604.

[14] H. Li, Q. Zhang, and D. Zhao, "Comparison of methods to efficient graph SLAM under general optimization framework," in *Proceedings - 2017 32nd Youth Academic Annual Conference of Chinese Association of Automation, YAC 2017*. Institute of Electrical and Electronics Engineers Inc., jun 2017, pp. 321–326.

[15] Y. Zhao, S. Xu, S. Bu, H. Jiang, and P. Han, "Gslam: A general slam framework and benchmark," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1110–1120.

[16] R. Mur-Artal, J. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, oct 2015.

[17] R. Mur-Artal and J. D. Tardos, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, oct 2017.

[18] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., sep 2014, pp. 15–22.

[19] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual-inertial odometry using nonlinear optimization," *International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[20] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization," Tech. Rep., 2016.

[21] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," Tech. Rep. 4, 2018.

[22] F. Dellaert and M. Kaess, "Factor Graphs for Robot Perception," *Foundations and Trends in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[23] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, feb 2017.

[24] P.-A. Absil, C. Baker, and K. Gallivan, "Trust-region methods on Riemannian manifolds," *Found. Comput. Math.*, vol. 7, no. 3, pp. 303–330, Jul. 2007.

[25] L. Carlone and A. Censi, "From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 475–492, 2014.

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

[27] C. Stachniss, U. Frese, and G. Grisetti, "Openslam," http://www.openslam.org/.

[28] R. Kümmerle, D. Hähnel, D. Dolgov, S. Thrun, and W. Burgard, "Autonomous driving in a multi-level parking structure," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2009, pp. 3395–3400.