Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

MASAT: A fast and robust algorithm for pose-graph initialization[☆,☆☆]

Károly Harsányi[a], Attila Kiss[a,b], Tamás Szirányi[a,c], András Majdik[a,*]

[a] Machine Perception Laboratory, Institute for Computer Science and Control (SZTAKI), Kende u. 13-17, Budapest 1111, Hungary
[b] Department of Computer Science, Eötvös Loránd University (ELTE), Egyetem tér 1-3, Budapest 1053, Hungary
[c] Budapest University of Technology and Economics (BME), Műegyetem rkp. 3., Budapest 1111, Hungary

ARTICLE INFO

Article history:
Received 12 March 2019
Revised 4 November 2019
Accepted 8 November 2019
Available online 9 November 2019

MSC:
41A05
41A10
65D05
65D17

ABSTRACT

In this paper, we propose a novel algorithm to compute the initial structure of pose-graph based Simultaneous Localization and Mapping (SLAM) systems. We perform a Breadth-First Search (BFS) on the graph in order to obtain multiple votes regarding the location of a certain robot position from all of its previously processed neighbors. Next, we define the initial location of a pose as the average of the multiple alternatives. By adopting the proposed initialization approach, the number of iterations needed for optimization is significantly reduced while the computational complexity remains lightweight. We perform quantitative evaluation on various 2D and 3D benchmark datasets to demonstrate the advantages of the proposed method.

1. Introduction

The robustness of Simultaneous Localization and Mapping (SLAM) algorithms highly depends on the tracked key-features in consecutive frames and the graph optimization methods for concatenating intermittent key-frames along the trajectory. For the features, point-based methods can be exceeded by structural elements [1] or object based [2] descriptors, while the position network (pose-graph) mostly needs iterative methods and an efficient initial network guess.

Numerous modern SLAM algorithms follow the pose-graph optimization formulation of the problem [3], where the nodes of the graph (the variables to be estimated) represent discrete robot positions sampled along the trajectory, and each edge (constraint) represents a measurement between a pair of poses. The measurements can originate either from ego-motion estimation *odometry* or from detecting *loop-closure* situations whenever the robot returns to a previously visited place [4].

The structure of the pose-graph is iteratively refined by nonlinear optimization (e.g., Gauss-Newton) starting from an initial guess [5]. Hence, a good initialization provides important benefits, since in case the initial estimate is near to the optimal solution, the optimization converges faster and the risk of convergence to a local minimum is reduced [6]. Conversely, a bad initial guess increases the computational time of the optimization and might lead the convergence of the algorithm to a local minimum, c.f., Figs. 1 and 2.

Commonly, the initial guess is computed by heuristic methods either by using the *odometry* measurements or by using a minimum *spanning tree* search. Both methods are computationally lightweight and have low-complexity. In order to avoid the caveats of a bad initial guess, several high-complexity initialization algorithms were proposed, more recently Cauchy algorithm [8]. However, all these algorithms are computationally complex and often include prior optimization steps, resulting in increased computational time. Conversely, the proposed approach is lightweight and has low-complexity. Furthermore, it can be applied as a preprocessing step before running more complex algorithms, in order to speed up and improve their results. Since the proposed algorithm uses all the previously computed nodes of the pose-graph to estimate a new location (c.f. Fig. 3), we name our method Multi-Ancestor Spatial Approximation Tree (MASAT).

To summarize, this paper advances the state-of-the-art of pose-graph optimization algorithms with the following contributions:

- A low-complexity and computationally efficient initialization algorithm (MASAT) is presented. MASAT is applicable for both 2D and 3D pose-graph optimizations, and it outperforms other lightweight baseline initialization algorithms. Furthermore, in case the data is affected by a large amount of noise, the pro-

[☆] Source code and dataset: Please note that this paper is accompanied by the source code of the proposed algorithm: https://github.com/karoly-hars/MASAT_IG_for_SLAM. The dataset used in this work is available at: http://mplab.sztaki.hu/masat_slam/masat_slam_data.zip.
[☆☆] Handled by Associate Editor Antonio Fernández-Caballero.
[*] Corresponding author.
E-mail address: majdik@sztaki.hu (A. Majdik).

https://doi.org/10.1016/j.patrec.2019.11.010

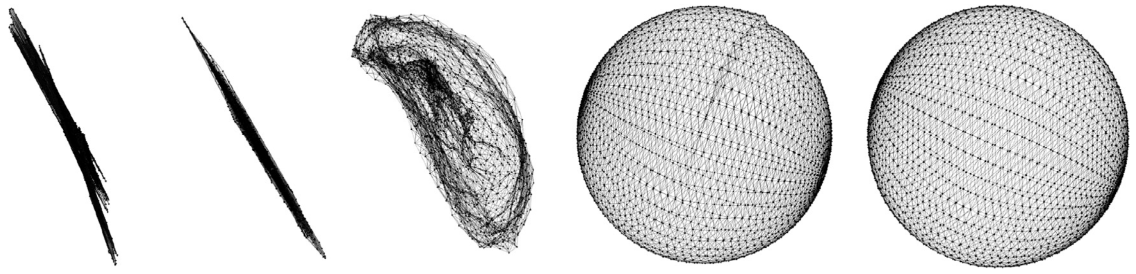

Fig. 1. The results of Gauss-Newton optimization from different initial guesses on the noisy version of the sphere dataset (independent, Gaussian noise with 0 mean, and 0.03 and 0.06 standard deviation on the coordinates and angles). Left to right: odometry, spanning tree, TORO [7], Cauchy algorithm [8], MASAT (proposed).

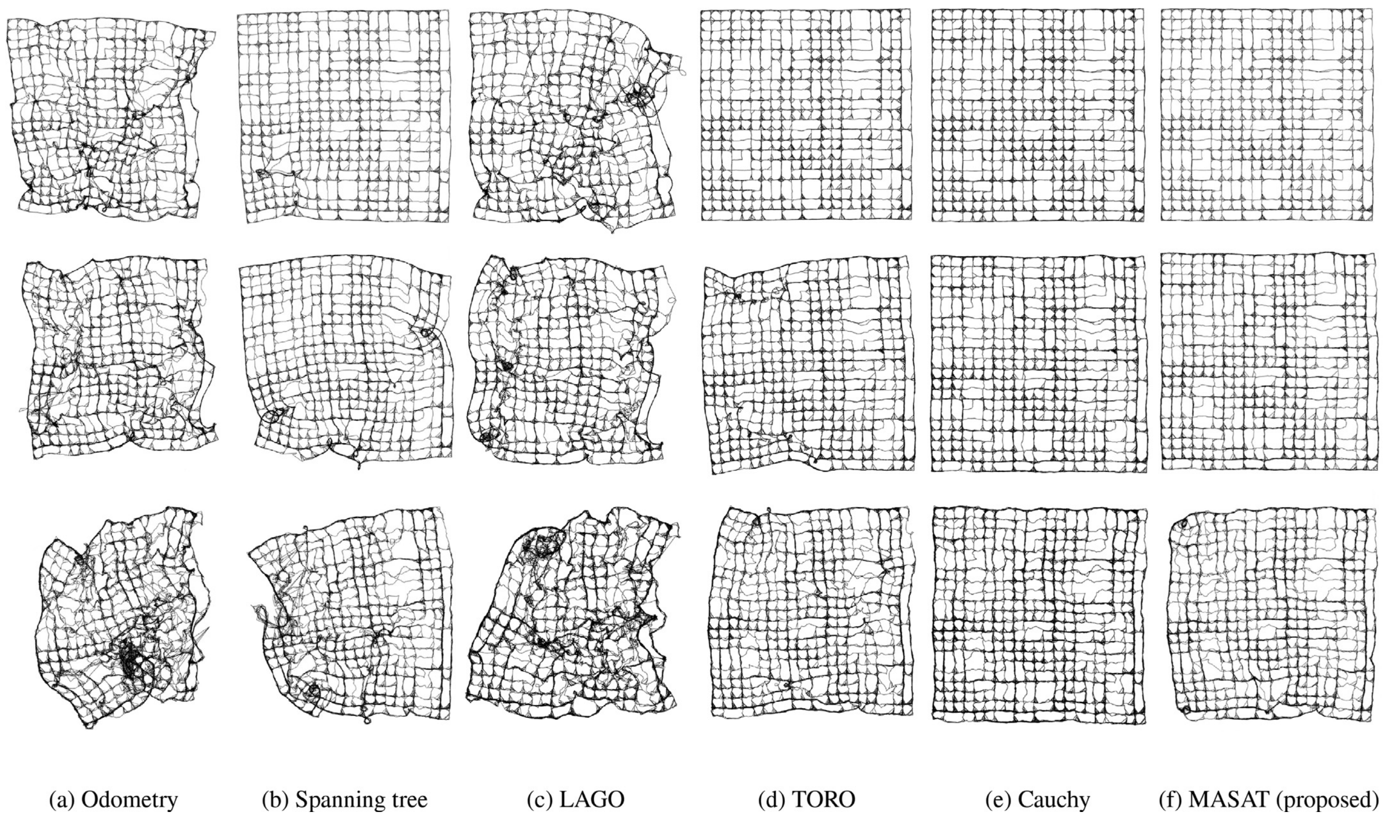

Fig. 2. The results of running 50 Gauss-Newton iterations from different initial guesses on the noisy version of the City10k dataset, with different noise levels; the deviation of the Gaussian noise is (0.1, 0.1) in the **first row**, (0.2, 0.2) in the **second row**, and (0.35, 0.35) in the **last row** for the coordinates and angles respectively.

posed algorithm even outperforms high-complexity initialization algorithms. This is notably true in case of 3D pose-graph structures.

- An extensive evaluation and comparison is reported using three 2D and two 3D datasets (with multiple noise levels) between the proposed method and five other baseline algorithms. The best results in terms average normalized error, rate of successful convergence (robustness), and average number of iterations is achieved when the proposed algorithm is applied as a preprocessing step of the Cauchy algorithm [8].
- We release the source code of the proposed algorithm, and provide all the data used to perform the comprehensive evaluation and comparison of the different approaches.

2. Related work

Numerous modern image based visual odometry (SVO algorithm [9]) and visual SLAM (LSD-SLAM [10], ORB-SLAM2 [11] algorithms) systems use the pose-graph representation to solve the

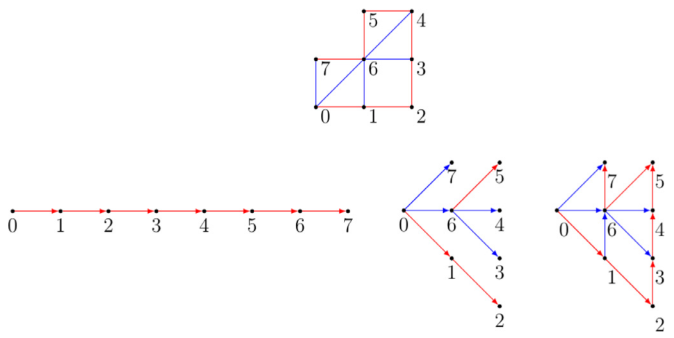

Fig. 3. Comparison of the pose-graph traversal methods for initial guess estimation; **top:** example pose-graph, consecutive incremental measurements are marked by the red edges, loop-closure measurements are marked by the blue edges; **bottom left:** *odometry*; **bottom center:** *spanning tree*; **bottom right:** the proposed method *MASAT*. Note that the proposed algorithm uses all the previously computed nodes of the pose-graph to estimate a new location. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

underlying SLAM problem. Therefore, there is a great need for robust and efficient pose-graph optimization back-ends. Generally, the main task of a pose-graph based SLAM back-end is to minimize the accumulated error of the measurement flow with the restrictions gained by different loop closures in the movement.

SLAM was first formulated as a pose-graph optimization problem in the seminal work of [12]. Since then, great effort was spent on studying this topic, a comprehensive overview is presented in [3]. Also, several open-source, computationally efficient pose-graph optimization back-ends were proposed in the literature: Georgia Tech Smoothing and Mapping library (GTSAM) [13], General Graph Optimization framework (g2o) [5], and Efficient Compact Pose SLAM library (SLAM++) [14].

Generally, non-linear optimization has a tendency to diverge, or converge into a local minimum if its initial state—the starting-point of the optimization—is too far from the ground truth. In order to avoid a bad initial guess, complex initialization techniques (also known as bootstrappers) were proposed: LAGO [15], TORO [7], and more recently Cauchy algorithm [8]. LAGO is a linear approximation technique that provides an accurate initial estimate for 2D pose-graphs, however, it can produce inadequate results if the measurements are noisy. TORO is an optimization framework that is robust against bad initial guesses, thus, it can be used as bootstrapper for other non-linear optimization techniques. Finally, Cauchy algorithm is an iterative approach based on M-estimation which produces accurate and reliable initial guesses even in scenarios with noisy measurements.

It was shown, that estimating rotations first in case of pose-graph initialization has significant advantages in terms of computational costs and robustness [16]. As rotation estimation in pose-graph optimization is a major issue, a survey is given about its advantages for 3D SLAM in [6]. Following the Lagrangian relaxation of [17] in the 3D case for tight process, in [18] remarkably good initialialization can be achieved even for non-tight relaxations. Applying these relaxation based methods to large scale, real-world problems is difficult, because of their high computational complexity. In special cases, some additional features can make the iteration process faster and more accurate:

- Sparsification is one possible solution to make the convergence faster, see [19] for finding particularities of pose-graph SLAM to exploit a novel factor descent iterative optimization method, achieving 80% of node reduction.
- Pose of absolute orientation can be better estimated if we can detect well-defined local structural cues, as architectural elements [20].

The proposed algorithm is unlike to the aforementioned relaxation based approaches: instead of performing a preliminary optimization step, our method traverses the pose-graph only once, similarly to the straightforward *odometry* and *spanning tree* techniques. During the traversal, it approximates each node's location based on its already positioned neighbors. The details of the proposed approach are described in the next section.

3. Proposed algorithm

The proposed algorithm approximates the exact position of a node by the votes on its location made by its previously processed neighbors. We illustrate this major difference between the proposed method and the competitor ones in Fig. 3. The figure illustrates that in contrast to other low-complexity algorithms, the proposed algorithm uses all the previously computed nodes of the pose-graph to estimate a new location. Formally, we run a Breadth-First Search (BFS) on our graph $G=(V,E)$ starting from node x_1. Than for each node $x_i \in V$ we investigate

$$N_i = \left\{\forall x_j \in V : x_i x_j \in E\right\}$$

the set of the neighbors of node x_i. In each step for every node $x_j \in N_i$ that was previously processed during the BFS, we get a vote regarding the position of x_i by the formula:

$$p_{x_j}(x_i) = (p_1, \ldots, p_d)$$

based on the measured distance between the two nodes and the angle of view from node x_j. In the aforementioned formula d stands for the dimensions of our problem.

Practically, the value of d is usually 2 or 3, but theoretically, this algorithm works on different values efficiently. After we got all the votes, we define the new location of x_i as the average of the votes on its position. The algorithm ends when all the nodes were processed by the BFS, c.f. (see Algorithm 1).

Algorithm 1: MASAT algorithm.

```
1  MASAT ($V, E, Z$);
   Input  : Raw measurements, vertices $v_i \in V$, edges $e_{ij} \in E$ and
            measurements $z_{ij}$ for each edge $e_{ij}$
   Output: An initial guess for the real pose-graph
2  Initially every vertex is labeled "not fix"
3  set vertex $v_0$ as the origo and its label is "fix"
4  Choose all the neighbors of vertex $v_0$ into the set $Q$
5  while $Q \neq \emptyset$ do
6      $w =$ the index of a vertex from the set $Q$.
7      for edges $e_{wi}$ do
8          if vertex $v_i$ is "not fix" then
9              put vertex $v_i$ into the set $Q$
10         else if $v_i$ is "fix" then
11             add measurement $z_{iw}$ to our existing estimation
12             Modify our existing estimation for the position of
               vertex $v_w$ with our new estimation
13     end
14     set the position of vertex $v_w$
15     vertex $v_w$ is "fix"
16     take vertex $v_w$ out from the set $Q$
17 end
```

4. Results

We compared our algorithm to low-complexity heuristic methods mentioned before (odometry/ODO, and spanning tree/SPT), and to high-complexity approaches: the LAGO, TORO, and Cauchy bootstrapping methods, on several well-known 2D and 3D benchmark datasets.

In details, we used the following process to generate scenarios with different noise levels: for each dataset we used in our experiments, we added independent, Gaussian noise with 0 mean and (σ_c, σ_a) standard deviation (for the coordinates and angles) to every measurement, and we examined how the Gauss-Newton algorithm performs from the different initial guesses. To perform the experiments, we used the g2o framework [5], since it is one of the most popular and widely adopted pose-graph optimization back-ends.

Let an attempt on finding the optimal placement of the nodes be defined successful if the growth of the error function χ_2 is less than a threshold $\left|10^{-6}\right|$ after an iteration step. The maximum number of iteration steps is fifty, and after reaching this amount of iteration steps an attempt will be defined unsuccessful. In the cases of successful experiments, we measured the average χ^2 error (the χ^2 error divided by the number of measurements) after the algorithm is finished, and the average number of Gauss-Newton iterations from start to end.

In order to quantitatively evaluate the proposed algorithm for every dataset and every noise level, we repeated this process 50

Table 1
Quantitative evaluation on the Manhattan3500 dataset.

Manhattan3500 (2D)		noise$(\sigma_c;\sigma_a) = (0.1; 0.1)$			noise$(\sigma_c;\sigma_a) = (0.2; 0.2)$			noise$(\sigma_c;\sigma_a) = (0.15; 0.3)$		
	runtime[1] (sec)	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2
ODO	**< 0.1**	0.64	30.19	7.656	0.26	37.54	5.025	0.04	46.0	2.54
SPT	**< 0.1**	**1.0**	7.76	1.092	0.82	21.44	1.492	0.28	36.07	1.532
LAGO	**< 0.1**	0.66	21.52	8.458	0.48	31.33	5.891	0.04	41.0	3.11
TORO[2]	1.05	**1.0**	4.9	1.074	0.96	7.1	**1.071**	0.72	15.56	1.107
CAU[3]	0.36	**1.0**	**4.26**	**1.073**	**1.0**	**5.32**	**1.071**	**0.96**	**7.77**	**1.073**
MASAT	**< 0.1**	**1.0**	6.3	**1.073**	0.96	12.92	1.148	0.76	19.95	1.146

Table 2
Quantitative evaluation on the Manhattan10000 dataset.

Manhattan10000 (2D)		noise$(\sigma_c;\sigma_a) = (0.1; 0.1)$			noise$(\sigma_c;\sigma_a) = (0.2; 0.2)$			noise$(\sigma_c;\sigma_a) = (0.15; 0.3)$		
	runtime[1] (sec)	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2
ODO	**< 0.1**	0.22	31.09	3.777	0.08	41.0	3.082	0.0	–	–
SPT	**< 0.1**	0.96	11.79	2.553	0.46	28.35	2.626	0.02	48.0	2.616
LAGO	0.24	0.18	24.11	3.693	0.06	34.67	3.102	0.0	–	–
TORO[2]	12	**1.0**	4.84	2.536	0.94	7.96	2.536	0.72	20.83	2.545
CAU[3]	3.5	**1.0**	**3.74**	**2.535**	0.94	**5.43**	**2.535**	**0.84**	**9.5**	2.537
MASAT	**< 0.1**	**1.0**	5.36	**2.535**	**0.96**	9.52	**2.535**	**0.84**	19.5	**2.536**

Table 3
Quantitative evaluation on the City10K dataset.

City10k (2D)		noise$(\sigma_c;\sigma_a) = (0.1; 0.1)$			noise$(\sigma_c;\sigma_a) = (0.2; 0.2)$			noise$(\sigma_c;\sigma_a) = (0.15; 0.3)$		
	runtime[1] (sec)	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2
ODO	**< 0.1**	0.0	–	–	0.02	30.0	11.874	0.0	–	–
SPT	**< 0.1**	0.96	8.75	1.584	0.2	32.5	1.922	0.0	–	–
LAGO	**< 0.1**	0.0	–	–	0.0	–	–	0.0	–	–
TORO[2]	5	**1.0**	4.42	**1.55**	0.48	18.38	1.671	0.02	36.0	1.763
CAU[3]	2.4	**1.0**	**4.0**	**1.55**	**1.0**	**4.9**	**1.546**	**0.72**	**6.75**	**1.544**
MASAT	**< 0.1**	**1.0**	6.16	**1.55**	0.96	11.83	1.568	0.22	23.82	1.569

Table 4
Quantitative evaluation on the Torus dataset.

Torus (3D)		noise$(\sigma_c;\sigma_a) = (0.02; 0.02)$			noise$(\sigma_c;\sigma_a) = (0.04; 0.04)$			noise$(\sigma_c;\sigma_a) = (0.03; 0.06)$		
	runtime[1] (sec)	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2
ODO	**< 0.1**	0.52	39.77	41.941	0.52	40.27	12.358	0.18	42.0	12.824
SPT	**< 0.1**	**1.0**	7.78	**32.188**	**1.0**	8.2	**9.18**	**1.0**	11.9	**11.688**
TORO[2]	8.7	0.0	–	–	0.0	–	–	0.0	–	–
CAU[3]	21	0.84	6.36	32.192	0.86	6.37	9.186	0.78	8.49	11.702
MASAT	**< 0.1**	**1.0**	6.24	**32.188**	**1.0**	6.82	**9.18**	**1.0**	9.06	**11.688**
MASAT+CAU	21	**1.0**	**5.4**	**32.188**	**1.0**	**5.48**	**9.18**	**1.0**	**8.0**	**11.688**

times. The outcome of our experiment is summarized in Tables 1–6, where ’conv.’ refers to the ratio of successful experiments to the 50 repetitions,[1,2,3] while ’avg. iter.’ and ’avg. χ^2 shows the average number of iterations before convergence and the average χ^2 error over the successful cases. A low χ^2 error indicates that the final pose-graph is close to the ground truth, and the number of Gauss-Newton iterations is directly proportional to the runtime of the optimization.

[1] On an Intel Core i7-8700 CPU

[2] 100 TORO iterations.

[3] 50 Cauchy iterations.

4.1. Evaluation on 2D datasets

We summarize the cumulated results on 2D datasets in Table 1 for Manhattan3500 [21], in Table 2 for Manhattan10000 [7], and in Table 3 for City10k [22] dataset respectively. Note on the tables, that in case the Gaussian noise is moderately large (noise$(\sigma_c;\sigma_a) = (0.1; 0.1)$), the proposed low-complexity algorithm (MASAT) performs similarly to other high-complexity methods (LAGO, TORO, and Cauchy) in terms of average number of iterations until convergence . However, the results are obtained an order of magnitude faster (in less than 0.1 seconds) with the proposed approach.

Table 5
Quantitative evaluation on the Sphere dataset.

Sphere (3D)									
		noise$(\sigma_c; \sigma_a) = (0.02; 0.02)$			noise$(\sigma_c; \sigma_a) = (0.04; 0.04)$			noise$(\sigma_c; \sigma_a) = (0.03; 0.06)$		
	runtime[1] (sec)	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2
ODO	**< 0.1**	0.0	–	–	0.0	–	–	0.0	–	–
SPT	**< 0.1**	0.88	8.64	4.733	0.4	13.1	5.023	0.0	–	–
TORO[2]	22	0.0	–	–	0.0	–	–	0.0	–	–
CAU[3]	16	**1.0**	**3.0**	**4.732**	0.74	4.32	5.26	0.26	12.08	11.827
MASAT	**< 0.1**	**1.0**	5.16	**4.732**	**1.0**	6.86	**5.026**	0.88	10.95	**5.323**
MASAT+CAU	16	**1.0**	**3.0**	**4.732**	**1.0**	**3.38**	**5.026**	**1.0**	**3.98**	**5.323**

Table 6
Quantitative evaluation on the Parking Garage dataset.

Garage (3D)										
		noise$(\sigma_c; \sigma_a) = (0.02; 0.02)$			noise$(\sigma_c; \sigma_a) = (0.04; 0.04)$			noise$(\sigma_c; \sigma_a) = (0.03; 0.06)$		
	runtime[1] (sec)	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2	conv.	avg. iter.	avg. χ^2
ODO	**< 0.1**	0.32	12.31	3.78	0.0	–	–	0.0	–	–
SPT	**< 0.1**	**1.0**	6.22	**2.85**	**1.0**	8.46	**2.48**	0.3	19.87	2.69
TORO[2]	2.5	**1.0**	4.66	**2.85**	**1.0**	5.56	**2.48**	0.66	8.06	**2.53**
CAU[3]	1.8	0.9	3.22	2.92	0.16	**3.5**	**2.48**	0.0	–	–
MASAT	**< 0.1**	**1.0**	5.06	**2.85**	**1.0**	6.5	**2.48**	**0.98**	8.84	**2.53**
MASAT+CAU	1.8	**1.0**	**3.1**	**2.85**	0.88	3.77	**2.48**	0.38	**4.37**	**2.53**

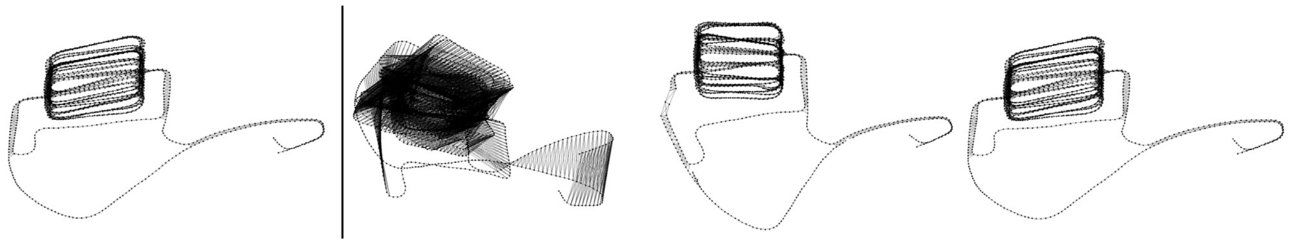
Fig. 4. The complete SLAM graph optimization process on the Parking Garage dataset. Left to right: ground truth, noisy data, MASAT initial guess, results after Gauss-Newton optimization from the initial guess.

If the Gaussian noise is larger (noise$(\sigma_c; \sigma_a) = (0.2; 0.2)$), the high-complexity Cauchy algorithm slightly outperforms MASAT at the cost of a significantly longer running time. Interestingly, the proposed approach performs best on the Manhattan10000 dataset. The other, high-complexity methods (LAGO, TORO) are outperformed by MASAT in terms of convergence rate. On the 2D datasets, the localization accuracy - in terms of average χ^2 error - after 50 iterations of Gauss-Newton optimisation is roughly the same whether we initiate from an estimation made by MASAT, TORO or Cauchy.

In contrast, MASAT surpasses other fast (low-complexity) methods by a large margin in terms of convergence and average number of iterations, in all of our experiments. Note from the data presented in the tables, that in case of large noise (noise$(\sigma_c; \sigma_a) = (0.2; 0.2)$) other simple methods (ODO, SPT) and LAGO have a very low convergence rate in comparison to the proposed approach. This is notably true in case of large pose-graphs like the Manhattan1000 and City10k datasets, which contain ten thousand nodes. Furthermore, if there is a large rotation error (noise$(\sigma_c; \sigma_a) = (0.15; 0.3)$), these methods fail to compute a proper initialization, and the pose-graph optimization will not converge. Therefore, the cumulated results on 2D datasets indicate that the proposed algorithm provides a good balance between speed and accuracy.

We illustrate some representative results in Fig. 2. The comparison shows that by applying the proposed initialization algorithm the grid-like structure of the Manhattan dataset is successfully computed even in case of large rotation errors (bottom right).

4.2. Evaluation on 3D datasets

Next, we used the simulated Torus dataset and a Sphere dataset (generated with g2o [5]), and the real world multi-level Parking Garage dataset [17] to evaluate the bootstrapping methods' performance for 3D measurements. The detailed evaluation shows that MASAT is superior to all other low or high-complexity approaches in terms of both convergence and accuracy (c.f., Tables 4, 5, and 6) at all three noise levels, while the runtime of the proposed algorithm is two order of magnitude faster than in the case of high-complexity approaches.

Although, Toro fails on the simulated Torus and Sphere datasets, it achieves good performances on the real-world Parking Garage dataset. Conversely, the Cauchy algorithm produces good results on the Torus and Sphere datasets and performs worse on the Parking Garage dataset, notably in the case of large noise levels (noise$(\sigma_c; \sigma_a) = (0.04; 0.04)$) and noise$(\sigma_c; \sigma_a) = (0.03; 0.06)$)). This might be due to the fact that in the simulated datasets the nodes are well connected between each other. To the contrary, the Parking Garage dataset contains fewer links between the consecutive and loop-closure positions recorded onboard a vehicle driving within a multi-storey parking facility. Better performance might be achievable by fine tuning the parameters of the Toro and Cauchy algorithms. In our experiments, we used the standard set of parameters originally proposed by the authors. Another major benefit of the proposed MASAT algorithm is that it does not rely on any predefined parameter or threshold.

A sample result obtained with the proposed algorithm using the real-world Parking Garage dataset is shown in Fig. 4. The ground truth trajectory of the vehicle (left) was successfully retrieved after optimisation (right) from very noisy measurements (second left) using the initial guess computed with the proposed MASAT algorithm (second right).

Finally, since the runtime of MASAT is rather low, we examined the possibility of running MASAT before the Cauchy bootstrapping, thus providing an initial guess using the combination of the two methods. Our findings show that the initialization has a large effect on the accuracy of Cauchy algorithm, and that Cauchy initialized with MASAT exceeds the regular Cauchy algorithm (initialized with odometry as suggested in [8]) in every aspect.

The generated noisy datasets and the C++ code of MASAT is publicly available at http://mplab.sztaki.hu/masat_slam/masat_slam_data.zip and https://github.com/karoly-hars/MASAT_IG_for_SLAM.

5. Conclusions

In this paper, we examined the problem of initial guess computation for off-line SLAM algorithms. We introduced MASAT, a heuristic, fast, and low-complexity method for bootstrapping. The utility of the proposed algorithm has been demonstrated through extensive experiments in both 2D and 3D settings. The method has been tested thoroughly on artificial datasets and on a real-life experimental data recording. Compared to high-complexity state-of-the-art solutions, the proposed method makes the pose-graph calculus for SLAM more efficient, and its excellent performance and precision in our experiments shows that it can be effective when applied in real-time evalutation.

Declaration of Competing Interest

None.

Acknowledgments

The research reported in this paper was supported by Hungarian Scientific Research Fund (No. NKFIH OTKA K-120499 and KH-126513) and by the project: Exploring the Mathematical Foundations of Artificial Intelligence 2018-1.2.1-NKP-00008.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.patrec.2019.11.010.

References

[1] J. Zhang, G. Zeng, H. Zha, Structure-aware slam with planes and lines in man-made environment, Pattern Recognit. Lett. (2018), doi:10.1016/j.patrec.2018.10.037.
[2] Y. Fan, H. Han, Y. Tang, T. Zhi, Dynamic objects elimination in slam based on image fusion, Pattern Recognit. Lett. (2018), doi:10.1016/j.patrec.2018.10.024.
[3] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J. Leonard, Past, present, and future of simultaneous localization and mapping: towards the robust-perception age, IEEE Trans. Rob. 32 (6) (2016) 1309–1332, doi:10.1109/TRO.2016.2624754.
[4] S. Lowry, N. Snderhauf, P. Newman, J.J. Leonard, D. Cox, P. Corke, M.J. Milford, Visual place recognition: a survey, IEEE Trans. Rob. 32 (1) (2016) 1–19, doi:10.1109/TRO.2015.2496823.
[5] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, W. Burgard, g2o: a general framework for graph optimization, in: International Conference on Robotics and Automation (ICRA), 2011, pp. 3607–3613, doi:10.1109/ICRA.2011.5979949.
[6] L. Carlone, R. Tron, K. Daniilidis, F. Dellaert, Initialization techniques for 3D SLAM: a survey on rotation estimation and its use in pose graph optimization, in: Intl. Conf. on Robotics and Automation, 2015, pp. 4597–4604, doi:10.1109/ICRA.2015.7139836.
[7] G. Grisetti, C. Stachniss, S. Grzonka, W. Burgard, A tree parameterization for efficiently computing maximum likelihood maps using gradient descent., in: Robotics: Science and Systems, 2007, pp. 27–30.
[8] G. Hu, K. Khosoussi, S. Huang, Towards a reliable SLAM back-end, in: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2013, pp. 37–43.
[9] C. Forster, M. Pizzoli, D. Scaramuzza, Svo: Fast semi-direct monocular visual odometry, in: International Conference on Robotics and Automation (ICRA), 2014, pp. 15–22.
[10] J. Engel, T. Schps, D. Cremers, LSD-SLAM: Large-scale direct monocular slam, in: European Conference on Computer Vision (ECCV), 2014, pp. 834–849.
[11] R. Mur-Artal, J.D. Tardós, ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras, IEEE Trans. Rob. 33 (5) (2017) 1255–1262, doi:10.1109/TRO.2017.2705103.
[12] F. Lu, E. Milios, Globally consistent range scan alignment for environment mapping, Auton. Robots 4 (4) (1997) 333–349.
[13] F. Dellaert, Factor graphs and GTSAM: A hands-on introduction, Technical Report, Georgia Institute of Technology, 2012.
[14] V. Ila, L. Polok, M. Solony, P. Svoboda, Slam++-a highly efficient and temporally scalable incremental slam framework, Int. J. Rob. Res. 36 (2) (2017) 210–230, doi:10.1177/0278364917691110.
[15] L. Carlone, R. Aragues, J. Castellanos, B. Bona, A linear approximation for graph-based simultaneous localization and mapping, in: Proc. of Robotics: Science and Systems, 2011, pp. 384–392, doi:10.15607/RSS.2011.VII.006.
[16] L. Carlone, A. Censi, From angular manifolds to the integer lattice: guaranteed orientation estimation with application to pose graph optimization, IEEE Trans. Rob. 30 (2) (2014) 475–492, doi:10.1109/TRO.2013.2291626.
[17] L. Carlone, D. Rosen, G. Calafiore, J. Leonard, F. Dellaert, Lagrangian duality in 3D SLAM: Verification techniques and optimal solutions, in: IROS, 2015, pp. 125–132.
[18] J. Briales, J. Gonzalez-Jimenez, Initialization of 3D pose graph optimization using Lagrangian duality, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 5134–5139, doi:10.1109/ICRA.2017.7989600.
[19] J. Vallv, J. Sol, J. Andrade-Cetto, Pose-graph slam sparsification using factor descent, Rob. Auton. Syst. 119 (2019) 108–118, doi:10.1016/j.robot.2019.06.004.
[20] S. Agarwal, K.S. Parunandi, S. Chakravorty, Robust pose-graph slam using absolute orientation sensing, IEEE Rob. Autom. Lett. 4 (2) (2019) 981–988, doi:10.1109/LRA.2019.2893436.
[21] E. Olson, J. Leonard, S. Teller, Fast iterative alignment of pose graphs with poor initial estimates, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., 2006, pp. 2262–2269, doi:10.1109/ROBOT.2006.1642040.
[22] M. Kaess, A. Ranganathan, F. Dellaert, iSAM: Fast incremental smoothing and mapping with efficient data association, in: Proc. IEEE Intl. Conf. on Robotics and Automation, ICRA, Rome, Italy, 2007, pp. 1670–1677.