https://doi.org/10.3311/PPee.21358
Creative Commons Attribution

Periodica Polytechnica Electrical Engineering and Computer Science, 67(4), pp. 403–412, 2023

A Collaborative Graph-based SLAM Framework Using a Computationally Effective Measurement Algebra

Gábor Péter[1]*, Bálint Kiss[1]

[1] Department of Control Engineering and Information Technology, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
* Corresponding author, e-mail: petergabor@iit.bme.hu

Received: 16 October 2022, Accepted: 05 May 2023, Published online: 05 June 2023

Abstract
Simultaneous localization and mapping (SLAM) is an essential task for autonomous rover navigation in an unknown environment, especially if no absolute location information is available. This paper presents a computationally lightweight framework to enable agents with limited processing power to carry out the SLAM cooperatively and without absolute onboard localization sensors in a 2D environment. The proposed solution is built on a graph-based map representation, where nodes (resp. edges) represent landmarks (resp. odometry-based relative measurements), a measurement algebra with embedded uncertainty, and a compact database format that could be stored on a server in a centralized manner. The operations required by the agents to insert a new landmark in the graph, update landmark positions and combine measurements as a loop is closed in the graph are detailed. The resulting framework was tested in a laboratory environment and on a public dataset with encouraging results; hence our method can be used for cost-effective indoor mobile agents with limited computational resources and onboard sensors to achieve a mapping while keeping track of the agent's position. The method can also be easily generalized for a 3D scenario.

Keywords
SLAM, lightweight, algebra, graph, uncertainty

1 Introduction

Simultaneous localization and mapping (SLAM) are crucial elements of autonomous navigation. It stands for the map-building process while continuously updating the rover's location. In-depth tutorials for understanding the basics of SLAM are provided by Durrant-Whyte and Bailey [1], and Bailey and Durrant-Whyte [2]. With perfect odometrical sensors and calculations or using absolute sensors such as GPS, the localization problem would become easily feasible, as the absolute position would be available to prepare the map of the environment. Building a map still requires some simplifications in this case as well, as some details must be discarded to have a representation of the environment in the on-board memory with limited capacity. One possible option is to use a landmark-based graph representation.

In many situations absolute sensors deliver measurements with great precision, but an easy-to-use alternative of GPS is often unavailable for many typical applications involving indoor navigation. The usually odometry-based relative displacement measurements between landmarks carried out by robots are noisy, but multiple measurements may be realized by different robotic agents in the same environment. Suitable, distributed algorithms may merge these measurements which take into consideration the limited computational features of the robotic agents, including processing power, available memory, available time, available communication bandwidth and so on.

A comprehensive survey of over 300 publications related to SLAM research, mostly from the last ten years, is presented by Huang et al. [3], demonstrating that this field is actively researched and of great importance.

Graph-based methods introduced new generations of SLAM algorithms. Early research results were reported by Thrun and Montemerlo [4], where a graph-based method for solving the online SLAM problem was presented. Ever since this method has existed as a resource-friendly alternative to filtering-based solutions thanks to its graph-based sparse nature.

As edges in the graph represent relative measurements, one may determine the relative location of two landmarks (i.e., graph nodes) by following a path. If a new measurement occurs thanks to a new path between two landmarks, this allows the fusion of the measurements along both paths. Such an operation is referred to as loop closure (LC). A robust LC method for graph-based SLAM is presented by Latif et al. [5]. Their novel algorithm enables the correction of previous misalignments in the graph based on new measurements. The authors also claim that the algorithm is almost real-time and works on large datasets.

Geometric primitives could be used to represent the map as reported by Aloise et al. [6]. However, these primitives are mainly found in indoor or urban environments, hence their usage might be limited to generic outdoor datasets. A more robust landmark relation-based method is presented by Himstedt et al. [7]. The proposed relations between detected landmarks could be used for outdoor environments as well, while storing the landmarks gives long-term stability to the algorithm.

A powerful landmark-based method with graph optimization is presented by Schuster et al. [8]. In their approach, a radar is used as environmental sensor, while inertial measurement unit (IMU)-boosted wheel-rotation based odometry is utilized for motion estimation. The landmarks are extracted from the point cloud aggregated by multiple radar scans using a measurement grid method. This method results in a map with high landmark density.

LIDAR-based SLAM methods are called LOAM (LIDAR based Odometry And Mapping). With the increasing availability of LIDAR sensors, promising research results are available in the field of sensor fusion such as GR-LOAM (Ground Robots-LOAM) presented by Su et al. [9]. In areas with few LIDAR detectable features, both wheel odometry and inertial data are heavily utilized as reported by Júnior et al [10]. Wheel-based odometry is a computationally lightweight method therefore it was selected for our purposes as well.

This article presents a graph-based method, treating the map as a collection of landmarks with the edges representing the connections between these landmarks. Most graph-based SLAM methods use pose-graphs, where a node represents the pose of the mapping agent at a given time. In our framework, they represent landmarks – a natural representation of the map. The major contribution of this paper is an edge-measurement algebra with embedded uncertainty which allows an efficient LC fusing the measurements. The method also enables collaborative mapping in a centralized way where the database of measurements is stored and refreshed on a server accessible by all robotic agents. The main benefit of this framework is that agents require only low computational power and basic communication capabilities, but thanks to the graph-based map representation and the novel measurement algebra, the resulting SLAM allows efficient fusion and higher precision. The algorithm consists of the following phases: position estimation, landmark matching, measurement fusion and relaxation. A novelty of the algebra is the use of a Delta-Star transformation during the relaxation phase, often referenced as Delta-Wye transformation in the literature since the work of Lehman [11]. A few early results of the authors were presented in [12].

The method studied here has been implemented on the embedded hardware of a reduced-size rover, equipped with a low-cost LIDAR device. The results obtained show the usability of the framework. Moreover, our method has been also tested off-line, on a public dataset provided by Bosse and Zlot [13] where the results are also promising.

The remaining part of the paper is organized as follows. Section 2 presents our SLAM framework detailing the graph representation, the measurement algebra, and the update scheme, including the LC. The framework presentation is followed by the implementation results on a small mobile platform and on a public dataset, and Section 4 summarizes the results and gives some hints for future research.

2 SLAM framework

Section 2 details the graph-based map storage method, the edge-measurement algebra with embedded uncertainty and the operations on the map. The framework relies on two main unit types, the central server, and the agents. Agents gather information during the exploration phase in the form of odometrical data and landmark detections, while the server stores and updates the map, thus accumulating information provided by the agents.

2.1 Map storage

A key component of the whole framework is a compact, graph-based storage method representing a simplified version of the environment. Without loss of generality, we restrict ourselves to the 2D case. Graph nodes are landmarks (or waypoints) representing a point with special features. For instance, in case of a corridor, landmarks can represent junctions, corners and places where doors are located.

A landmark is represented by its [X, Y] coordinates in a Cartesian frame. The relative displacement of two neighboring landmarks consists of three measurements as depicted in Fig. 1 for a general case. The relative dis-

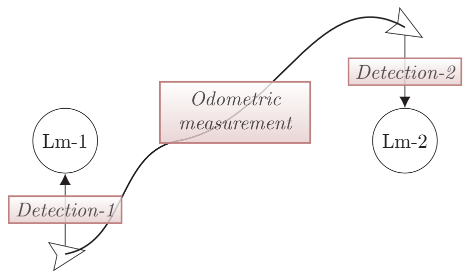


Fig. 1 Landmark detection

placement equals the path travelled by the agent between the two landmarks plus the detections themselves.

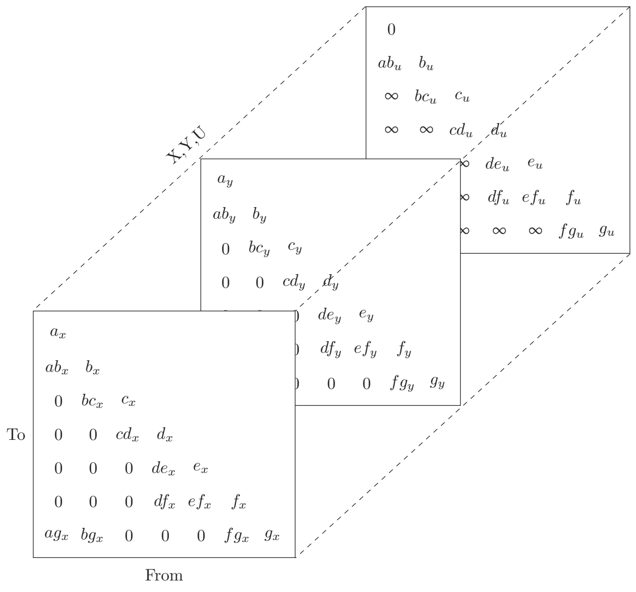
Paths or ways may connect the landmarks representing the edges in the graph. Their shape may vary according to the navigation algorithm used by the robot. These ways will be characterized by the overall displacement expressed by the ΔX and ΔY parameters. Handling multiple measurements between the same nodes requires a measurement fusion operation, while evaluating measurement-chains requires measurement addition. There is also a need for a graph-level operation - the delta-star conversion - to simplify the graph during the relaxation phase. All these operations define an edge-measurement algebra.

Robotic agents contribute to the map building by detecting landmarks during navigation and providing measurements on their relative positions. These measurements are noisy, therefore a third – uncertainty – parameter is added to both the landmark positions and to their pairwise relations. During calculations uncertainty denotes variance, while for plotting, standard deviation is used. The proposed algorithm relies on error-free long-term orientation estimation from the mapping agents as orientation is not stored in the framework. If the provided orientation estimation is error-free, the 1-dimensional uncertainty is a fair representation, although measurement vectors are 2-dimensional. The noise model is Gaussian – it is an additive, zero mean, white noise.

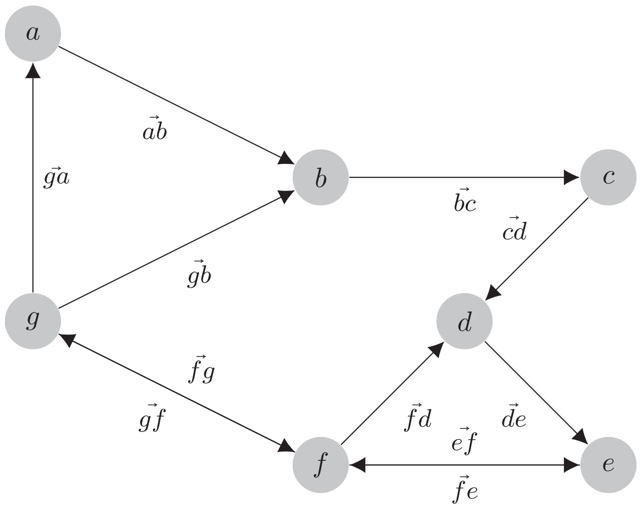
The graph-based map is stored on a server which also carries out the map-update step, thus agents have no need for high computational power. It is assumed that agents can communicate with the server through a channel with appropriate bandwidth.

Memory is a valuable resource; thus, a compact representation of the graph is necessary. The graph is stored as a matrix, where non-diagonal elements represent the edges and diagonal elements represent the nodes. Edges represent displacement between nodes combining all measurements. Due to the symmetry a lower triangular matrix representation can be chosen. The columns select the starting node of the edges, referred to as the *From* indices, while rows select the terminal node of the edges, referred to as the *To* indices. Having the lower triangular representation, the diagonal elements have the same *From* and *To* indices, thus giving the matrix depicted in Fig. 2.

Using the matrix representation, landmark measurements are stored as a triplet: coordinates X and Y and the uncertainty U. The value U equals the variance, and it is initialized to ∞. If there is a single mapping agent with sufficient memory for storing the whole map and the processing power to evaluate the map-update step, the entire framework could be executed directly on the rover. In the case of multiple rovers mapping cooperatively so that only



(a)



(b)

Fig. 2 Map representation – matrix and graph form; (a) in matrix form; (b) in graph form

a limited computational resource is available on a single mapping agent, the database could be stored and updated on a central processing unit to reduce the computational needs for each rover. This requires however some basic communication between the agents and the central processing unit.

2.2 Measurement algebra

The core of the SLAM framework is an algebra defining operations on measurements considering disturbances. A measurement $\vec{M}$ is given as:

$$\vec{M} = \left[(X,Y),U\right] = \left[\vec{D},U\right], \tag{1}$$

where X and Y define a location (resp. displacement (D)) for a landmark (resp. along a path connecting two landmarks), and U denotes the uncertainty (variance of measurement errors). A home landmark position is needed for the framework. All other landmarks could be localized relative to this reference landmark. The uncertainty value of the home position is zero.

To define operations for measurements, let us introduce $\overrightarrow{M_1}$ and $\overrightarrow{M_2}$ measurements as:

$$\overrightarrow{M_1} = \left[\overrightarrow{D_1},U_1\right], \tag{2}$$

$$\overrightarrow{M_2} = \left[\overrightarrow{D_2},U_2\right]. \tag{3}$$

The basic operation of the algebra is negation.

$$-\overrightarrow{M_1} = \left[-\overrightarrow{D_1},U_1\right] \tag{4}$$

The uncertainty value is not affected by this operation. Negation is used for calculating the measurement vector after applying an endpoint swap. With path between landmarks A and B, the operation is straightforward since:

$$\overrightarrow{AB} = -\overrightarrow{BA}. \tag{5}$$

The next binary operation is addition.

$$\overrightarrow{M_1} + \overrightarrow{M_2} = \overrightarrow{M_2} + \overrightarrow{M_1} = \left[\overrightarrow{D_1} + \overrightarrow{D_2}, U_1 + U_2\right] \tag{6}$$

Displacement vectors and uncertainties are treated separately regarding this operation. The addition is both associative and commutative thus the order of measurements does not affect the result.

$$\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC} \tag{7}$$

Addition is used to calculate the resulting displacement of multiple vectors or to determine the position of a landmark, given a reference landmark and a displacement vector.

$$A + \overrightarrow{AB} = B \tag{8}$$

A measurement addition is depicted in Eq. (7) where A, B and C are landmarks, while $\overrightarrow{AB}$ and $\overrightarrow{BC}$ are the two measurements, resulting the $\overrightarrow{AC}$ virtual measurement vector. A landmark – measurement addition is defined in Eq. (8) where A is the starting landmark, $\overrightarrow{AB}$ is the measurement vector from A to B and B is the second landmark.

The last binary operation is fusion denoted as:

$$\overrightarrow{M_3} = \overrightarrow{M_1} \times \overrightarrow{M_2}. \tag{9}$$

Fusion is used to combine measurements along same edges during the mapping phase and the operation is defined by:

$$\overrightarrow{D_3} = \frac{\overrightarrow{D_1}U_2 + \overrightarrow{D_2}U_1}{U_1 + U_2} = \frac{\overrightarrow{D_1}\frac{U_2}{U_1} + \overrightarrow{D_2}}{1 + \frac{U_2}{U_1}}, \tag{10}$$

$$U_3 = \frac{U_1U_2}{U_1 + U_2} = \frac{U_2}{1 + \frac{U_2}{U_1}}. \tag{11}$$

Having a limitation for measurement uncertainty between (0, ∞) and having an initial value of ∞ for all measurement uncertainties, the second set of formulae are numerically stable. Thus, the fusion operation reads:

$$\overrightarrow{M_3} = \overrightarrow{M_1} \times \overrightarrow{M_2} = \left(\frac{\overrightarrow{D_1}\frac{U_2}{U_1} + \overrightarrow{D_2}}{1 + \frac{U_2}{U_1}}, \frac{U_2}{1 + \frac{U_2}{U_1}} \right). \tag{12}$$

Fusion is both associative and commutative if no measurements with infinity or zero uncertainty are involved. In the case of measurements between the same landmark-pairs with different detection orders, one of them should be negated before the fusion:

$$\overrightarrow{FG_3} = \overrightarrow{FG_1} \times \left(-\overrightarrow{GF_2}\right). \tag{13}$$

There is no unique identity element regarding fusion. However, the fusion with a measurement with unbounded uncertainty leaves the original measurement unchanged as the displacement could be of any value, only the unbounded uncertainty matters.

2.3 Delta-Star conversion

The Delta-Star conversion is an essential graph-level operation, as no simple operation could simplify a delta pattern thanks to its strongly connected nature. During the relaxation phase, the need for eliminating delta patterns and obtaining equivalent star patterns is a fundamental requirement. Fig. 3 illustrates the transformation.

The goal is to define the virtual measurement vectors $\vec{a}$, $\vec{b}$ and $\vec{c}$ and a virtual landmark E, knowing the estimated position of A, B and C and knowing the measurements $\vec{x}$, $\vec{y}$ and $\vec{z}$. The uncertainty values for $\vec{a}$, $\vec{b}$ and $\vec{c}$ are derived from the uncertainty values between points in the $[A, B, C]$ set.

$$U_{AB} = \frac{(U_y + U_z)U_x}{U_x + U_y + U_z} = U_a + U_b \tag{14}$$

$$U_{AC} = \frac{(U_x + U_z)U_y}{U_x + U_y + U_z} = U_a + U_c \tag{15}$$

$$U_{BC} = \frac{(U_x + U_y)U_z}{U_x + U_y + U_z} = U_b + U_c \tag{16}$$

By summing both sides, one gets:

$$\frac{U_xU_y + U_xU_z + U_yU_z}{U_x + U_y + U_z} = U_a + U_b + U_c. \tag{17}$$

The uncertainties (variances) read:

$$U_a = \frac{U_xU_y}{U_x + U_y + U_z}, \tag{18}$$

$$U_b = \frac{U_xU_z}{U_x + U_y + U_z}, \tag{19}$$

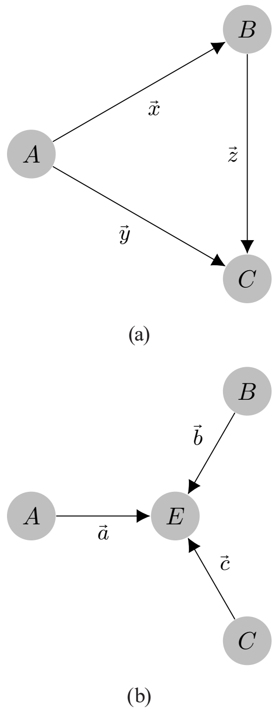


Fig. 3 Delta-Wye transformation; (a) Delta form; (b) Wye form

$$U_c = \frac{U_yU_y}{U_x + U_y + U_z}. \tag{20}$$

The position of the new virtual point E could be anywhere, but for practical reasons, it is placed in the centroid of the ABC triangle. The displacement values of the virtual measurements could be expressed from basic geometrical correlations.

$$\overrightarrow{D_a} = \frac{\overrightarrow{D_x} + \overrightarrow{D_y}}{3},\ \overrightarrow{D_b} = \frac{-\overrightarrow{D_x} + \overrightarrow{D_z}}{3},\ \overrightarrow{D_c} = \frac{-\overrightarrow{D_y} + \overrightarrow{D_z}}{3} \tag{21}$$

2.4 Map update scheme

The map update scheme relies on the operations defined as part of the measurement algebra. The steps are presented in Sections 2.4.1 to 2.4.5.

2.4.1 Initialization

The SLAM framework pre-allocates the required memory to store the map in a matrix of a given number of landmarks and measurements. All but one triplet is initialized to $[(0,0)\ \infty]$. As the algorithm needs a reference landmark, the first diagonal element of the matrix is set to the position of the reference landmark, with zero uncertainty: $[(X_R, Y_R), 0]$.

2.4.2 Landmark detection and position estimation

If an agent detects a landmark, an odometry-based measurement is obtained, as depicted in Fig. 1. The measurement may contain errors from both the odometry calculations and due to limited detection accuracy. The position of the latest landmark passed by the agent combined with the displacement equals the estimated position of the landmark detected.

2.4.3 Landmark matching

Once the current landmark position is available it is checked against the diagonal elements of the map matrix, considering the uncertainties. If the detected landmark is already stored in the map, a measurement fusion and LC are triggered, otherwise it is simply inserted in the map.

2.4.4 New landmark insertion

In the case of the detection of a new landmark, two elements are modified in the map matrix. An off-diagonal element will store the new odometry measurement, and a diagonal element will store the landmark position.

2.4.5 Fusion of measurements and loop closure

Reaching a landmark already detected closes a loop of measurements. The measurement that was already in the database must be fused with the new one, using the binary fusion operator of the measurement algebra. Once a new measurement is added that is part of a loop, the position of all landmarks must be updated. The estimated position of each landmark is recalculated in the so-called relaxation phase.

2.5 Relaxation

Once the measurements are up to date, each landmark position is recalculated by relaxing the graph using an iterative combination of the following steps: dead-end removal, simple node removal, and delta-star conversion.

Dead-end removal is the process of eliminating nodes with single connections. Simple node removal is the process of eliminating a node with two connections, optionally creating a new virtual measurement, or fusing it with an already existing one.

Fig. 4 illustrates the algorithm for a simple example where a is the reference landmark, and the position of landmark d is updated. Thus, the relaxation results in a simple edge connecting a to d.

3 Experimental results

The proposed framework was analyzed with real data in both indoor and outdoor scenarios. The public Kenmore dataset was used for the outdoor case [13].

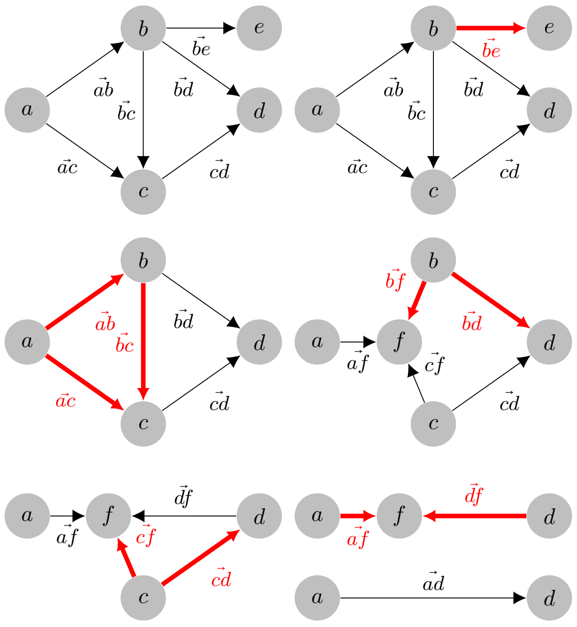


Fig. 4 Illustration of the relaxation procedure

3.1 Indoor tests using a simple mobile robot

The indoor test environment depicted in Fig. 5 has a simple geometry consisting of a square corridor with an outer two meters wide square and an inner one-meter wide square. Landmarks are the corners of the outer square.

The robot, depicted in Fig. 6, navigates along the walls, keeping a predefined distance while running a simple odometric algorithm based on the angular displacement measurements of its wheels. It is also possible to incorporate into the odometric calculations that the neighboring wall sections are perpendicular to each other. This information is considered with a Kalman-filter and it is based on a line fitting algorithm for the point cloud recorded by the LIDAR on board.

The LIDAR sensor collects distance measurements at a rate of 4 kHz, with a rotation speed of around 10 Hz, resulting in an angular resolution of about one degree. The resolution of the wheel rotation sensors is 6144 increments per revolution, while the tire perimeter equals 187 mm, resulting in a displacement resolution of 0.03 mm. Motion estimation

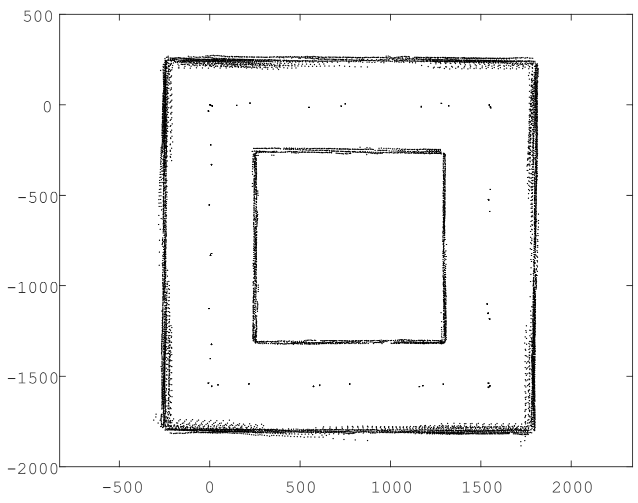


Fig. 5 View of the corridor using LIDAR-based measurements. Units are in mm

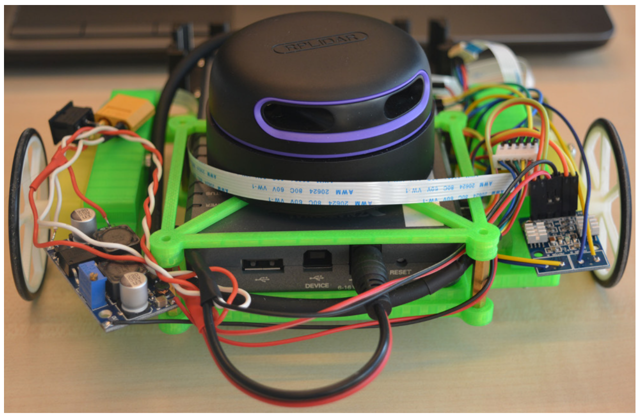
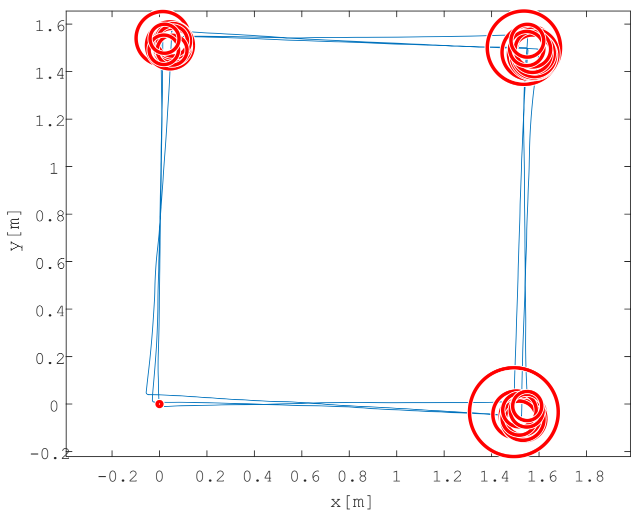


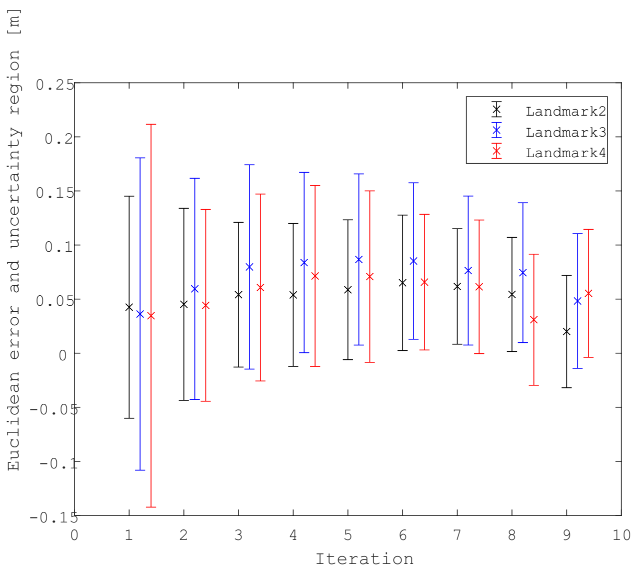
Fig. 6 Mobile agent

is based on odometry with a switchable Kalman-filter using LIDAR measurements. The σ_{Odo} value equals 0.01 mm as 0.03 mm equals three sigmas. In the case of LIDAR measurements, the point cloud is processed, and lines are fitted, representing the walls. The precision of these fits is unknown, a worst-case estimation of 10 degrees is utilized. Having the three sigmas equaling 10 degrees, σ_{LIDAR} equals around 0.05.

The rover follows the centerline of the actual corridor in a clockwise manner. The framework was tested with and without orientation correction. Fig. 7 depicts the trajectory after three consecutive rounds together with the landmark estimates without orientation correction. The radius of a landmark estimate corresponds to the 3-sigma threshold, while the center-point depicts the belief of where the landmark is estimated to be.



(a)



(b)

Fig. 7 Mapping without orientation correction; (a) Measurement vectors and landmark estimates; (b) Euclidean errors and variances

The standard deviation of the measurements was set in a way that 3-sigmas equals 10 cm for every section.

The first LC in the graph occurs when the robot revisits the home position at (0,0) after the first round along the corridors. Having noisy orientation during the measurements, the estimates are significantly changing over time, while the overall uncertainty decreases from measurement to measurement after LC.

The uncertainty value of each landmark together with the Euclidean error of the estimated position is plotted against the iteration count starting from the detection of the fourth landmark. The greatest improvement is shown at the first LC. The 3-sigma region of the last landmark along the chain dropped from 5.8 cm to 2.9 cm.

By using the same measurement set combined with orientation correction applied off-line, the trajectory and the estimates have a smoother look as depicted in Fig. 8. As the orientation is estimated way better over the entire three consecutive runs, the estimates are more stable regarding their positions.

The robustness of the algorithm can be seen by comparing Fig. 7 to Fig. 8. Although the estimated trajectories are way closer to the ground truth in the second case, the landmark position estimates are almost identical for both cases proving the SLAM framework uses the advantage of LC.

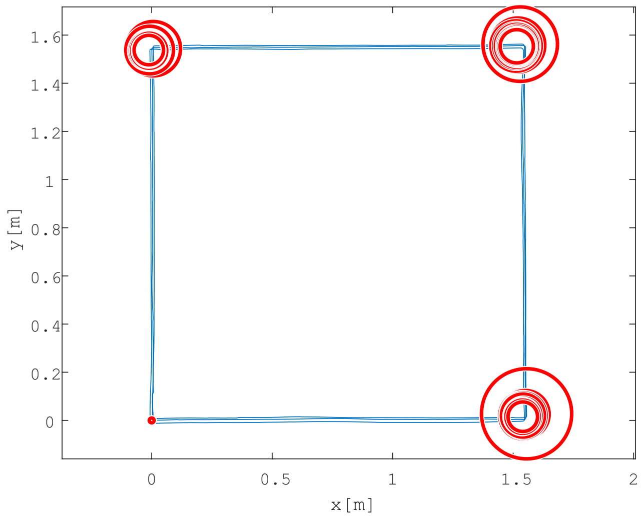
3.2 Kenmore dataset

The Kenmore dataset is an outdoor dataset consisting of odometry-based trajectory estimation and LIDAR measurements [8].

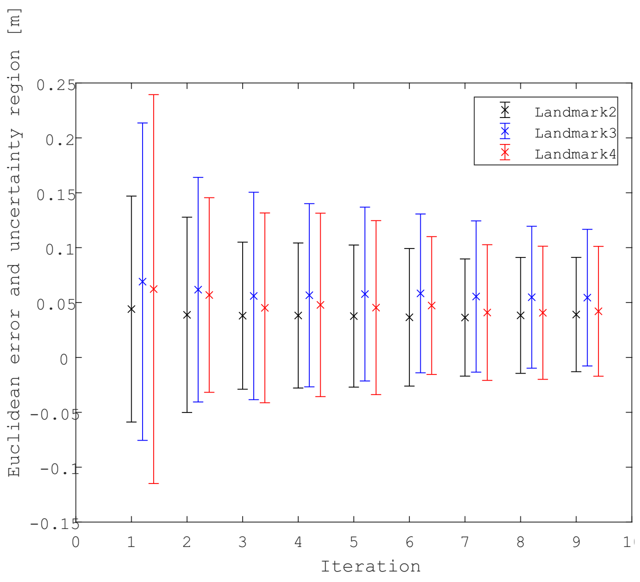
With no prior knowledge of the geometry of the environment, the previously utilized line fitting method and landmark selection cannot be used. Therefore, a more versatile method, the NDT-algorithm was used for scan-matching [14]. In order to generate measurements from the dataset, first the odometry-based pose estimation and the LIDAR-based pose estimation (Fig. 9) had to be fused.

The landmarks were generated at points where the orientation changed considerably using an offline smoothing filter. As most of the Kenmore dataset would generate no loops in the graph representing the map of the environment, only the measurement samples from 6000 to 7800 were used during testing. The starting position was given with coordinates (0,0) and with a heading of zero to the mapping agent.

Having the fused trajectory, the local maxima of the curvature were identified after a simple differentiation and smoothing of the orientation, and these points were selected as landmarks. By using such a simple and coarse landmark definition, all measurements were given the



(a)



(b)

Fig. 8 Mapping with orientation correction; (a) Measurement vectors and landmark estimates; (b) Euclidean errors and variances

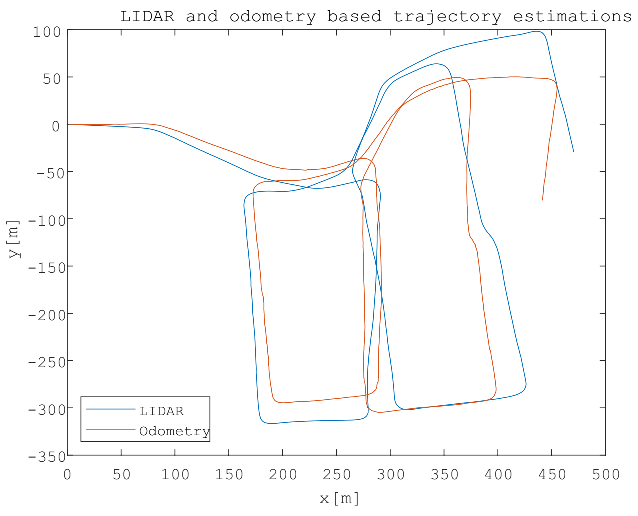


Fig. 9 Kenmore dataset - trajectory estimations

same uncertainty – corresponding to a standard deviation of 3 meters. The fused trajectory estimation and the generated landmarks are depicted in Fig. 10.

The Kenmore dataset was evaluated by feeding the estimated measurements between upcoming landmarks to the framework. As there were 20 landmarks identified other than the reference landmark, the framework was called 20 times.

The output after the last call of the framework is depicted in Fig. 11. Comparing this to Fig. 10, the differences are obvious: the framework correctly closed multiple loops, thus decreasing the uncertainty value for all the nodes above the third node, while providing better position estimates for these nodes by fusing multiple measurements of uncorrelated errors. The uncertainty values for the landmarks are depicted in Fig. 12 as a function of iterations.

The ground-truth is given with the dataset in form of an image, depicted in Fig. 11. The final graph-based representation shows great similarity to the ground-truth, however no numerical values were found, and therefore no exact comparison could be made.

The evaluation of all 20 measurements takes roughly 23 ms on a 4th generation mobile i7 CPU, meaning that the framework is well suited for embedded systems as evaluation is required only upon landmark detection. As depicted in Fig. 13 the time requirement of the open loop landmark addition takes less than 0.1 milliseconds, while closed loop update time depends on graph complexity and node count. As a comparison, it takes 59 ms on average for GTSAM to solve the factor-graph created with the same measurements, however GTSAM requires both the landmark indices as inputs along the measurements and an initial estimate for all landmark positions [15]. Our approach does not require these as inputs, they are rather part of the output of our algorithm.

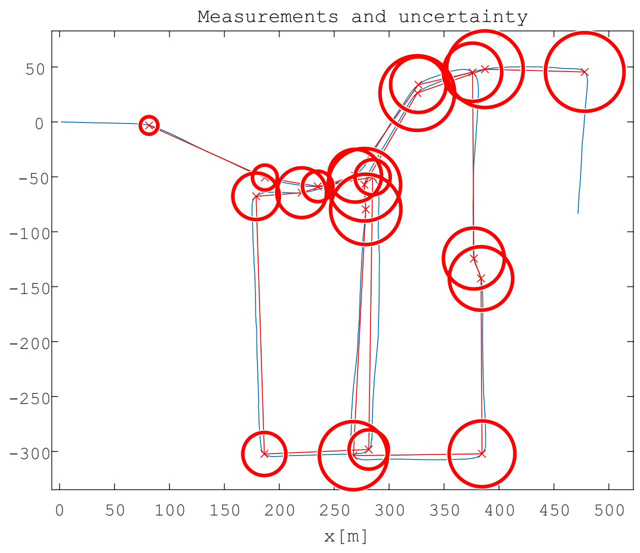
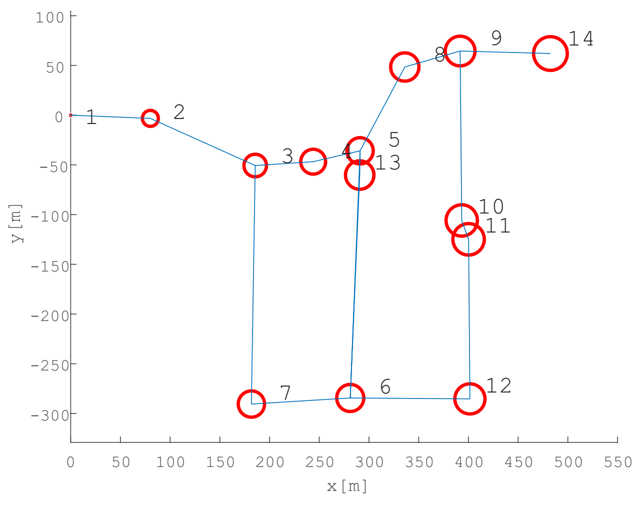
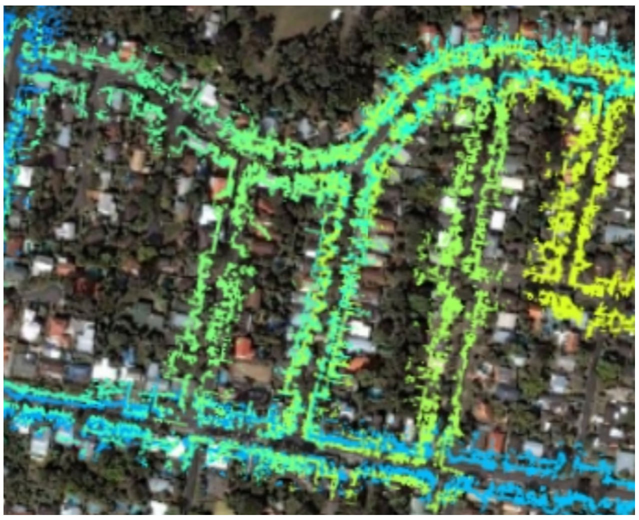


Fig. 10 Kenmore dataset – Uncertainty without loop closures



(a)



(b)

Fig. 11 Kenmore dataset – Estimated map vs ground truth; (a) Estimated map; (b) Ground truth

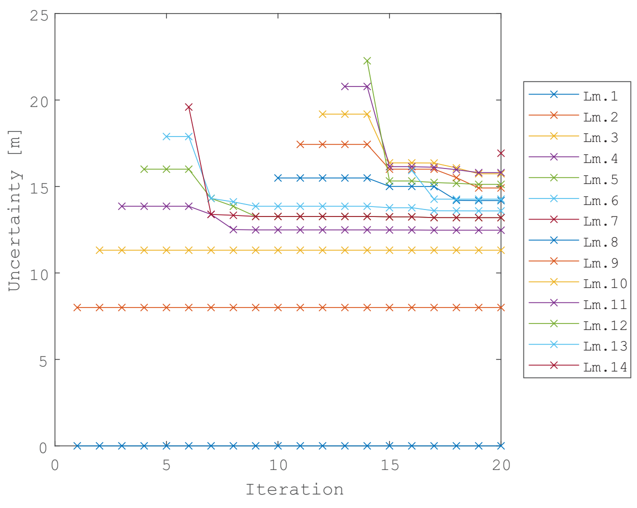


Fig. 12 Loop closures

Our method relies on odometry calculations between landmark detections, which are usually executed even if no higher-level SLAM algorithms are present, thus giving only limited extra overhead between detections.

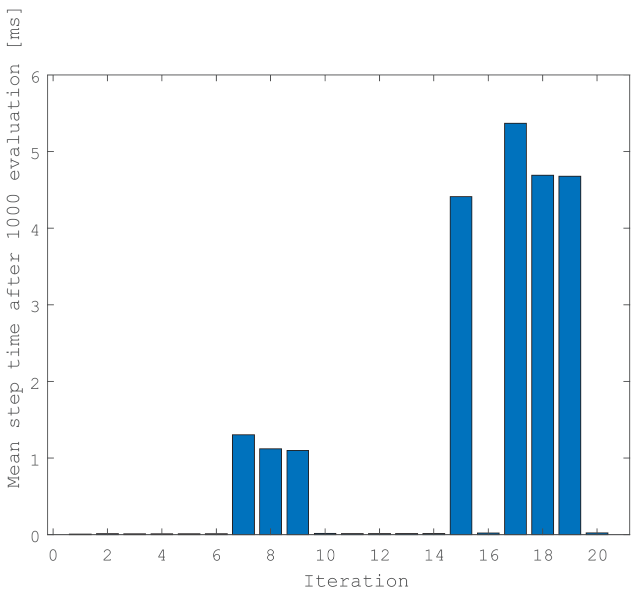


Fig. 13 Timing

4 Conclusions

The proposed SLAM framework delivers a lightweight solution, thus enabling agents with low processing power to do the mapping during navigation. (The choice of the landmark detection algorithm may increase the required computational power.) The simple algebra with its embedded uncertainty delivers a robust framework that ensures proper handling of real-life measurements if long-term error-free orientation estimation is available. Further research goals are the implementation of landmark descriptors (e.g., LIDAR measurements) to handle orientation correction and to robustly reject false LC; and the addition of cooperative mapping experiments (the framework supports it by design). Most of the publicly available datasets do not provide an orientation estimation with long-term stability. Therefore, the algorithm could not be tested on those datasets. A next version of the framework is planned to have orientation correction and 2-dimensional uncertainty as well and shall be tested on datasets such as Intel research lab and Infinity corridor.

Acknowledgement

We thank Michael Bosse and Robert Zlot for providing the Kenmore dataset via MRPT.org.

The research reported in this paper is part of project no. BME-NVA-02, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021 funding scheme.

References

[1] Durrant-Whyte, H., Bailey, T. "Simultaneous localization and mapping: part I", IEEE Robotics & Automation Magazine, 13(2), pp. 99–110, 2006.
https://doi.org/10.1109/MRA.2006.1638022

[2] Bailey, T., Durrant-Whyte, H. "Simultaneous localization and mapping (SLAM): Part II", IEEE Robotics & Automation Magazine, 13(3), pp. 108–117, 2006.
https://doi.org/10.1109/MRA.2006.1678144

[3] Huang, B., Zhao, J., Liu, J. "A Survey of Simultaneous Localization and Mapping with an Envision in 6G Wireless Networks", [preprint] arXiv, arXiv:1909.05214, 24 August 2019.
https://doi.org/10.48550/arXiv.1909.05214

[4] Thrun, S., Montemerlo, M. "The graph SLAM algorithm with applications to large-scale mapping of urban structures", The International Journal of Robotics Research, 25(5–6), pp. 403–429, 2006.
https://doi.org/10.1177%2F0278364906065387

[5] Latif, Y., Cadena, C., Neira, J. "Robust loop closing over time for pose graph SLAM", The International Journal of Robotics Research, 32(14), pp. 1611–1626, 2013.
https://doi.org/10.1177%2F0278364913498910

[6] Aloise, I., Corte, B. D., Nardi, F., Grisetti, G. "Systematic handling of heterogeneous geometric primitives in graph-SLAM optimization", IEEE Robotics and Automation Letters, 4(3), pp. 2738–2745, 2019.
https://doi.org/10.1109/LRA.2019.2918054

[7] Himstedt, M., Frost, J., Hellbach, S., Bohme, H.-J., Maehle, E. "Large scale place recognition in 2D LIDAR scans using geometrical landmark relations", In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 2014, pp. 5030–5035. ISBN 978-1-4799-6934-0
https://doi.org/10.1109/IROS.2014.6943277

[8] Schuster, F., Keller, C. G., Rapp, M., Haueis, M., Curio, C. "Landmark based radar SLAM using graph optimization", In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 2016, pp. 2559–2564. ISBN 978-1-5090-1890-1
https://doi.org/10.1109/ITSC.2016.7795967

[9] Su, Y., Wang, T., Shao, S., Yao, C., Wang, Z. "GR-LOAM: LiDAR-based sensor fusion SLAM for ground robots on complex terrain", Robotics and Autonomous Systems, 140, 103759, 2021.
https://doi.org/10.1016/j.robot.2021.103759

[10] Júnior, G. P. C., Rezende, A. M. C., Miranda, V. R. F., Fernandes, R., Azpúrua, H., Neto, A. A., Pessin, G., Freitas, G. M. "EKF-LOAM: An Adaptive Fusion of LiDAR SLAM With Wheel Odometry and Inertial Data for Confined Spaces With Few Geometric Features", IEEE Transactions on Automation Science and Engineering, 19(3), pp. 1458–1471, 2022.
https://doi.org/10.1109/TASE.2022.3169442

[11] Lehman, A. "Wye-delta transformation in probabilistic networks", Journal of the Society for Industrial and Applied Mathematics, 11(3), pp. 773–805, 1963.

[12] Péter, G., Kiss, B. "Compact pose-graph SLAM framework based on algebra with embedded uncertainty", In: 2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 2018, pp. 000057–000062. ISBN 978-1-5386-4641-0
https://doi.org/10.1109/SACI.2018.8440967

[13] Bosse, M., Zlot, R. "Keypoint design and evaluation for place recognition in 2D lidar maps", Robotics and Autonomous Systems, 57(12), pp. 1211–1224, 2009.
https://doi.org/10.1016/j.robot.2009.07.009

[14] Biber, P., Strasser, W. "The normal distributions transform: A new approach to laser scan matching", In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453), Las Vegas, NV, USA, 2003, pp. 2743–2748. ISBN 0-7803-7860-1
https://doi.org/10.1109/IROS.2003.1249285

[15] Dellaert, F. "Factor graphs and GTSAM: A hands-on introduction", Georgia Institute of Technology, Atlanta, GA, USA, Rep. GT-RIM-CP&R-2012-002, 2012.