Lightweight SLAM with automatic orientation correction using 2D LiDAR scans

Gábor Péter
Dept. of Control Eng. and Information Technology
Budapest University of Technology and Economics
Budapest, Hungary
petergabor@iit.bme.hu

Bálint Kiss
Dept. of Control Eng. and Information Technology
Budapest University of Technology and Economics
Budapest, Hungary
bkiss@iit.bme.hu

***Abstract*—Simultaneous localization and mapping (SLAM) is about consistent maps in the long run. Loop closing is the most popular way for ensure long-term consistency in presence of multiple measurements by the same or multiple robots. Loop closure can be executed using raw odometrical data, but a more sophisticated, yet still light-weight method is presented in this paper: a landmark descriptor-based relative displacement calculation method for diminishing unwanted orientation errors that otherwise often lead to map inconsistency. Landmark descriptors are created using light detection and ranging (LiDAR) scans and the relation is calculated using scan-matching. The novelty of this research is a method providing long-term orientation and position correction without additional overhead between landmark detections, thus enabling simple agents to do the SLAM in a cooperative way.**

***Index Terms*—SLAM, LiDAR, mapping, orientation, correction, uncertainty**

I. Introduction

Simultaneous localization and mapping (SLAM) is one of the fundamental problems in mobile robotics [1]. As the palette of available sensors increases, so does their usage in SLAM algorithms. In contrast to the early days where no advanced ambient sensors were available, today's systems often rely on point clouds generated in two or even in three dimensions. Cameras are the most popular sensors for SLAM research as they are widely available in various size and resolution at a low price-point and by using global registration techniques, global consistency can be maintained as reported by Han et al. [2]. However, these methods are computationally still too heavy in order to be executed on low-cost mapping agents with moderate or low computational power.

A more specialized sensor unit is a light detection and ranging sensor (LiDAR) which generates a distance map of the robot's environment. For autonomous vehicles, these sensors often provide a 3D representation of the environment for advanced object detection and classification, but for simple mobile robots, the price range of these advanced devices is often off budget hence simpler, 2D sensors are employed. By using 3D LiDAR, primitive geometries could be detected and they can improve long term stability [3].

LiDAR measurements represent a new era of environmental sensing, especially the usage of the reflectance parameter provided by high-end devices. Aldibaja et al. [4] presented a method based on the 2D reflectance images with excellent results. The quality of today's high-end, 3D LiDAR devices allow scan matching that can generate the trajectory of a mapping agent without using any other sensing techniques [5]. In [6], Vlaminck et al. present a method for scan-matching based loop closure detection, relying on GPU processing, thus greatly increasing the performance of the overall system. Ren et al. [7] provides an improved correlative scan matcher algorithm signaling the importance of robust and reliable scan matching.

An alternative sensor to LiDAR is Radar that generates less precise, but more weather-proof measurements, suitable for automotive applications. In [8] the authors use iterative closest point (ICP) algorithm with Radar as a promising alternative to LiDAR.

Pose graph based SLAM methods are popular thanks to their efficient storage method compared to raster-based solutions. A typical application of SLAM is GPS disabled environments such as mines or sewage networks [9]. Relative orientation between different poses is a key to overall improvement of the resulting map as detailed by Yen et al. [10] - their orientation correction is based on WiFi signal strength gradients. A relative feature measurement is utilized in [11] to improve the orientation estimate of the mapping agent.

LiDAR information is incorporated here in a descriptor as explained in the sequel. Section II describes the homogeneous transformation based algebra in addition to the LiDAR-scan based landmark descriptors, resulting a compact framework. Section III details the experimental results with the system followed by a short conclusion.

An earlier work of the authors presents the basic ideas of their research [12]. The novelty of this article is the orientation correction, the LiDAR-based landmark descriptor system and an advanced graph-simplification method, enabling systems with limited processing power to do the SLAM.

II. SLAM framework with orientation correction

The proposed framework consists of a graph-based compact storage method, an algebra with embedded uncertainty using homogeneous coordinate transformations, and a LiDAR-scan based landmark descriptor system for orientation correction. Mapping agents do not rely on a priori information about

the environment or the landmarks. The system also supports cooperative mapping out of the box. Let us present the components next.

A. Compact storage method using homogeneous parameters

The framework is based on a graph where each node represents a landmark and every edge represents a relative displacement including orientation change. As the landmarks themselves have no orientation parameter, it denotes the orientation of the LiDAR-scan for every landmark. As a LiDAR scan is created in the frame of the mapping agent, the orientation parameter describes the relative orientation of the mapping agent relative to the world frame at the time instance the LiDAR-scan was generated. As time goes on, all the parameters - other than the parameters for the initial landmark - might change, but for long-term stability the initial LiDAR-scan of each node is unchanged.

The system uses homogeneous coordinate transformations for calculations, therefore it stores the transformation parameters as a compact parameter vector. For further details on homogeneous transformations, see [13]. As the system is designed for planar mapping, only the X, Y and ϕ parameters and the corresponding uncertainties U_{XY} and U_ϕ are used, but an extension to 3D is feasible and easily done.

The parameter vector associated to a node has the form of $[X, Y, U_{XY}, \phi, U_\phi]$. As the X and Y parameters are often treated together, a simplified form of the vector equals $[D, U_D, \phi, U_\phi]$. Translational transformation matrices are formed from X and Y, while rotational matrices require just ϕ. A parameter vector either represents a landmark, or a measurement between two landmarks. In the former case, the parameters define the position of the landmarks and its uncertainty and the orientation of the landmark descriptor and its uncertainty. If the vector represents a measurement, it stores the parameters of the homogeneous transformation between two landmarks with the uncertainties.

The vectors are stored in a three dimensional matrix so that each layer has a lower triangular form. The first layer, or page stores the X, the second the Y, the third the U_{XY}, the fourth the ϕ and the last the U_ϕ parameters. The columns represent the $From$ indices while the rows represent the To indices. The diagonal of every page stores the parameters of the landmarks, while the lower elements denote measurements as depicted in Fig. 1.

B. Homogeneous transformation based algebra

An algebra with embedded uncertainty using homogeneous coordinates was defined as the engine under the hood allowing measurement fusion and loop closure operations for the map. Let $M_1 = [D_1, U_{D1}, \phi_1, U_{\phi 1}]$ and $M_2 = [D_2, U_{D2}, \phi_2, U_{\phi 2}]$ denote measurements or landmark positions. The homogeneous transformation matrix is generated by multiplying the rotational transformation matrix with the translational transformation matrix as given by Eq. 1. Let us denote $cos(\phi)$ as C_ϕ and $sin(\phi)$ as S_ϕ.

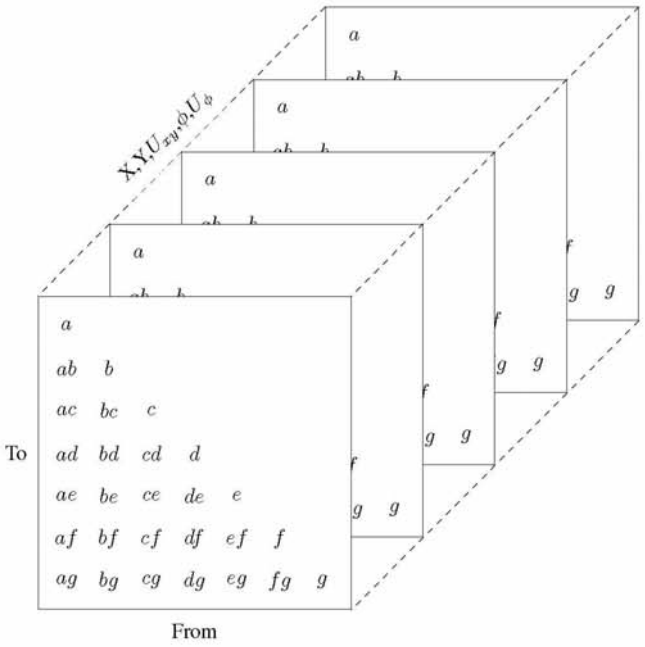


Fig. 1. Three dimensional matrix representation of the map

$$
\begin{aligned}
Tr_1 &= Rot(\phi_1) * Trans(D_1) \\
&= \begin{bmatrix} C_{\phi_1} & -S_{\phi_1} & 0 & 0 \\ S_{\phi_1} & C_{\phi_1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & x_1 \\ 0 & 1 & 0 & y_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} C_{\phi_1} & -S_{\phi_1} & 0 & x_1 \\ S_{\phi_1} & C_{\phi_1} & 0 & y_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned} \tag{1}
$$

Similarly,

$$
Tr_2 = \begin{bmatrix} C_{\phi_2} & -S_{\phi_2} & 0 & x_2 \\ S_{\phi_2} & C_{\phi_2} & 0 & y_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2}
$$

The basic operation of the algebra is negation. $M_3 = -M_1$ Uncertainties are unaffected by this operation and the homogeneous parameters have to be inverted. The inverted parameters arise after inverting the Tr_1 matrix. Negation has a neutral element: [0 0 - 0 -].

The next operation is addition or serial combination. $M_3 = M_1 + M_2$ Homogeneous parameters are obtained by simply multiplying the two transformation matrices:

$$
Tr_3 = Tr_1 * Tr_2 \tag{3}
$$

Uncertainties (both translational and orientational) add in square:

$$
U_{D3} = \sqrt{U_{D1}^2 + U_{D2}^2}, U_{\phi 3} = \sqrt{U_{\phi 1}^2 + U_{\phi 2}^2} \tag{4}
$$

Addition is used for combining two consecutive measurements during the map simplification phase, and it is also used for calculating a new landmark position estimate having a stored landmark and a new measurement originating from the known landmark.

Addition is non-commutative, but it is associative, regarding transformation parameter calculations. The identity element is the identity matrix, resulting a parameter vector of [0 0 0 0 0].

Fusion or parallel combination is the operation where two measurements along the same edge are being fused. Fusion uses a weighted sum of all the transformation parameters weighted by uncertainty values.

$$M_3 = M_1 \times M_2 = [D_3, U_{D3}, \phi_3, U_{\phi 3}] \tag{5}$$

$$D_3 = \frac{D_1 * U_{D2}^2 + D_2 * U_{D1}^2}{U_{D1}^2 + U_{D2}^2} = \frac{D_1 * \frac{U_{D2}^2}{U_{D1}^2} + D_2}{1 + \frac{U_{D2}^2}{U_{D1}^2}} \tag{6}$$

$$U_{D3} = \frac{U_{D1}^2 * U_{D2}^2}{\sqrt{U_{D1}^2 + U_{D2}^2}} = \frac{U_{D2}}{\sqrt{1 + \frac{U_{D2}^2}{U_{D1}^2}}} \tag{7}$$

$$\phi_3 = \frac{\phi_1 * U_{\phi 2}^2 + \phi_2 * U_{\phi 1}^2}{U_{\phi 1}^2 + U_{\phi 2}^2} = \frac{\phi_1 * \frac{U_{\phi 2}^2}{U_{\phi 1}^2} + \phi_2}{1 + \frac{U_{\phi 2}^2}{U_{\phi 1}^2}} \tag{8}$$

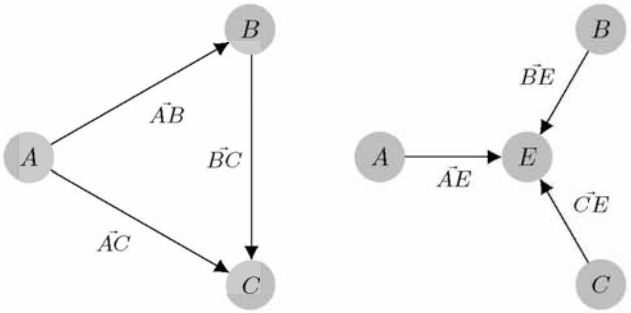
$$U_{\phi 3} = \frac{U_{\phi 1}^2 * U_{\phi 2}^2}{\sqrt{U_{\phi 1}^2 + U_{\phi 2}^2}} = \frac{U_{\phi 2}}{\sqrt{1 + \frac{U_{\phi 2}^2}{U_{\phi 1}^2}}} \tag{9}$$

Fusion is both associative and commutative as long as no operand has infinity or zero uncertainty. Fusion has a left identity element, since

$$[D_2, U_{D2}, \phi_2, U_{\phi 2}] = [D_1, \infty, \phi_1, \infty] \times [D_2, U_{D2}, \phi_2, U_{\phi 2}]. \tag{10}$$

A graph-level operation is the Delta-Star transformation. The closed loop update step requires the recalculation of all landmark positions, while fusing all the available measurements. This step is realized by aggregating all measurements into one super-measurement between the initial position and the landmark pose for each landmark. The delta-formation is the only one, that could not be eliminated by removing all dead-ends from the graph and by combining all serial and parallel edges into super-edges while removing excess nodes from the graph. A delta-formation needs its own transformation as the resulting star-formation could be further simplified thanks to its weakly connected nature.

Having nodes A, B and C in the delta formation, the transformation parameters for edges leading to the virtual star-node E could be defined with the equations



a: Delta-form b: Star-form

Fig. 2. Complex formations

$$AE_{tr} = \left[\frac{D_{AB}}{3}, -\phi_{AB}\right] + \left[\frac{D_{AC}}{3}, -\phi_{AC}\right] \tag{11}$$

$$BE_{tr} = \left[\frac{D_{BC}}{3}, -\phi_{BC}\right] + \left[\frac{D_{BA}}{3}, -\phi_{BA}\right] \tag{12}$$

$$CE_{tr} = \left[\frac{D_{CA}}{3}, -\phi_{CA}\right] + \left[\frac{D_{CB}}{3}, -\phi_{CB}\right] \tag{13}$$

The uncertainty parameters could be expressed as

$$AE_u = \sqrt{\frac{AB_u * AC_u}{AB_u^2 + AC_u^2 + BC_u^2}} \tag{14}$$

$$BE_u = \sqrt{\frac{AB_u * BC_u}{AB_u^2 + AC_u^2 + BC_u^2}} \tag{15}$$

$$CE_u = \sqrt{\frac{AC_u * BC_u}{AB_u^2 + AC_u^2 + BC_u^2}} \tag{16}$$

both for displacement uncertainty and for orientation uncertainty. The graphical representation of the two different forms are depicted in Fig. 2.

C. Landmark handling

A main feature of the framework is the landmark-descriptor system. It stores a LiDAR-scan for all the detected landmarks and stores an orientation value recorded on first encounter for every landmark. The LiDAR-scan is a non-alterable entity, ensuring long term stability, while the orientation value is re-evaluated as more measurements appear during the mapping. The orientation value is therefore stored in the matrix structure as the orientation of the landmark, and it is just the orientation of the LiDAR-scan stored for a given landmark relative to the world frame.

The LiDAR-scans are stored in a separate structure, using angles and distances. The system is designed for low-cost systems, therefore an angular resolution of around 1 degree is a good compromise between precision and memory-footprint. For some robustness margin a vector of length 400 is allocated for angular and distance values to store the LiDAR-scan of a landmark. The scan-matching algorithm works with different

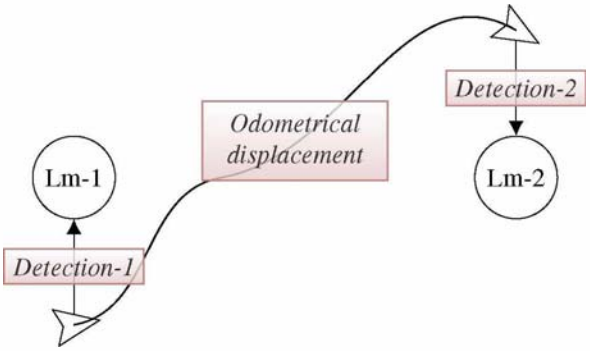


Fig. 3. Relative displacement of neighbouring landmarks

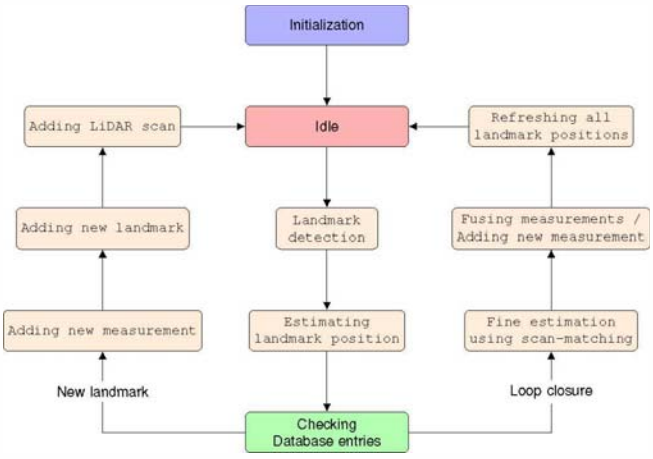


Fig. 4. Flowchart of the measurement update scheme of the SLAM framework

length of scans, therefore one measurement might contain more points than others.

Landmark detection is implemented by localizing intersections and turning points along the path. Other definitions of landmarks might be used as well, this method is used for its simplicity. For every landmark, the first detection is realized with zero offset (both for displacement and orientation), but the upcoming detections of a landmark, already having a descriptor results in a detection with non-zero offset. The transformation matrix, representing the transformation between the actual pose of the robot and the pose of the landmark is calculated using scan-matching. The scan-matching algorithm by Biber et al. [14] is used for obtaining the numerical values of the transformation.

The relative configuration of two landmark is determined by the combination of three measurements as

$$M_{Lm1-Lm2} = (-M_{Det1}) + M_{Odo} + M_{Det2} \tag{17}$$

where M_{Det1} and M_{Det2} are obtained using LiDAR-scans of the two landmarks and M_{Odo} is the result of odometric calculations along the path. By applying the rule of negation and addition of the algebra, the measurement could be expressed by transformation matrices as:

$$Tr_{Lm1-Lm2} = Tr^{+}_{Det1} * Tr_{Odo} * Tr_{Det2} \tag{18}$$

A travel move between two already stored landmarks generates a trajectory that could be simplified as a transformation matrix, and two landmark detections, that could also be represented by their own transformation matrices. By multiplying the three matrices, the resultant transformation matrix arises, defining the relationship between the two landmarks. Having a new measurement, the map should be updated accordingly. The flowchart of the mapping is depicted in Fig. 4.

By using scan matching, each new detection of a previously stored landmark gives the opportunity to realign the agents orientation. This is a crucial task as odometry often gets corrupted by an incorrect heading, while elapsed distance measurements tend to be more or less correct. By utilizing the LiDAR-based descriptor of the landmark and the actual LiDAR-scan at the new detection, the rovers relative displacement to its previous pose could be obtained by scan-matching. The scan matching also results in a new landmark-landmark measurement that needs to be fused to the map, decreasing the uncertainty of potentially multiple nodes. After refreshing all landmark poses, the agent can use the inverse of the transformation matrix obtained by scan-matching in order to estimate its actual pose. By revisiting the initial landmark during the mapping, the robot should be able to cancel out all its accumulated odometrical errors.

$$Pose_{Agent} = Pose_{Landmark} * Tr^{+}_{Det} \tag{19}$$

A mapping strategy could be defined where an agent returns to landmarks with little to no uncertainty in order to keep its pose uncertainty value below a predefined threshold value. Such a strategy could exclude the possibility of a false loop closure as the agent would be capable of controlling its own pose uncertainty.

Graph simplification is executed using the following algorithm:

procedure LMPOSUPDATE(Graph,LmIdx)
  Remove dead-ends
  Remove simple nodes
  Convert Delta-formation to Star-formation
  if There was a D-S conversion **then**
    LmPosUpdate(Graph,LmIdx)
  end if
  return $LmPosEstimate$
end procedure

The algorithm requires the graph and a node-index to perform the simplification by removing all dead-ends, removing all simple nodes (by merging edges into super-edges). After all simple conversions were executed, the graph is checked for a remaining Delta-formation. In case a Delta-formation is present, it is converted into a Star-formation and the function is called recursively until only the initial node, the target node, and the transformation between them remains. The new

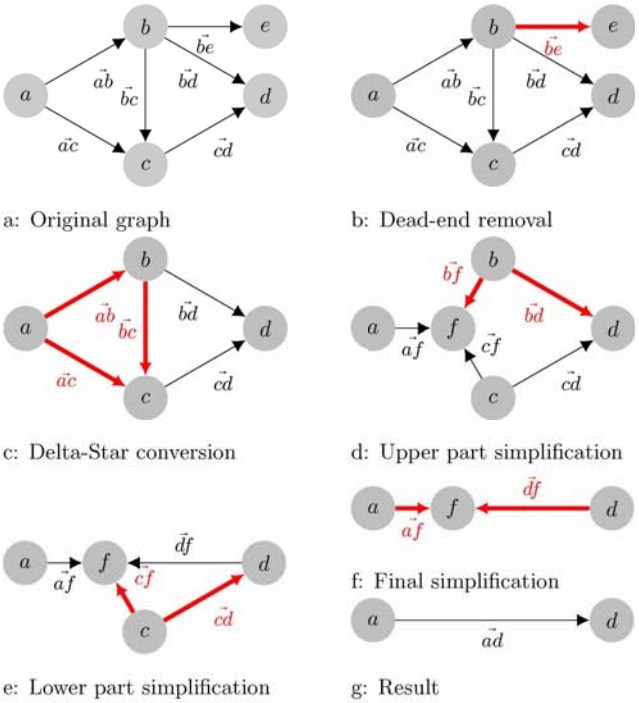


Fig. 5. Graph simplification steps for Landmark d (Origin: a)

updated pose of the given node equals transformation applied to the initial node.

D. Cooperative behavior

As the framework uses homogeneous transformations for all relative displacements, and all landmarks have their own LiDAR-based descriptors, multiple agents can cooperate to decrease the overall mapping time or to decrease the uncertainty of the landmarks by concurrent mapping. During cooperative mapping, all map-related calculations are preferably executed on a central entity to ensure consistency. A decentralized mapping is also possible, but the benefits of cooperative mapping are unavailable until the merge of decentralized map-segments, requiring a transformation between all landmark-pairs as the descriptors can be potentially different for all agents. This transformation presents an unwanted and unnecessary overhead, that is eliminated by simply evaluating all landmark measurements on a central entity - which might be one of the agents as well.

Cooperative mapping does not require any agent-agent interaction, it just increases the number of landmark detections thus generating more measurements and decreasing the uncertainty of the map, while maintaining consistency. Each agents reports a landmark detection to the central unit, thus increasing the common knowledge with a new measurement. The central unit has to be able to handle all agents simultaneously.

The initial localization of the first agent is trivial as it is the position of the initial landmark. In order to initialize additional agents, their starting position has to be an already detected landmark, and their pose had to be initialized with a given

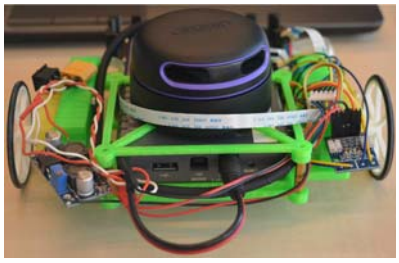


Fig. 6. Rover with LiDAR sensor on top

uncertainty. As a second agent is added to the game, its estimated pose and its initial LiDAR scan is compared to the map already generated by other agents. Thanks to the estimated position and the LiDAR-scan, the agents starting pose can be determined precisely relative to the given landmark. After this pose initialization, the agent is now part of the mapping team and can generate its own measurements.

III. Experimental results

The proposed framework was tested in a laboratory environment with a LiDAR equipped mobile agent. The test environment is a simple corridor-square, with a corridor width of 50 centimeters and an outer dimension of 2 meters by 2 meters, giving a corridor-center square with a length of 1.5 meters each side. The mapping agent uses the LiDAR for ambient sensing, while wheel encoders provide precise odometrical data. Fig. 6 depicts the mapping agent.

As the framework relies on scan-matching for its orientation correction step, both the odometrical data and the LiDAR scans at landmark detections were recorded for off-line processing. Data collection was implemented as a corridor-center following behavior using LiDAR measurements. The experimental run consists of three consecutive laps in the corridor, with an initial 90 degrees right turn at the origin and an extra run along the first edge at the end. This results in 14 LiDAR scans at the landmark locations (corners), triggering 10 scan-matching events, thus the orientation correction capabilities of the system could be measured by comparing the resulting map to the actual square-shaped map.

As depicted by Fig. 7, a small amount of systematic error is present in the odometrical system as the depicted lines along same edges tend to diverge from being parallel.

As the LiDAR sensor being used is a low-cost sensor and it was used in legacy mode, the scans are somewhat imperfect as at low angles of incidence, the corridor wall could not be detected at all. Such an incomplete scan is depicted in Fig. 8. However this leads to no issues as the scans still contain enough information for the scan-matching algorithm to detect landmarks.

The results of the test run are depicted in Fig. 9. The first iteration depicts the Euclidean position error and the translational uncertainty value before the loop closure. As the rover reaches the initial corner, it creates a new LiDAR scan and compares it to the one already stored at initialization

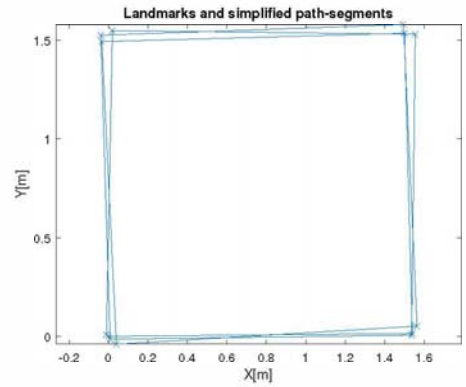


Fig. 7. Landmarks and simplified path-segments

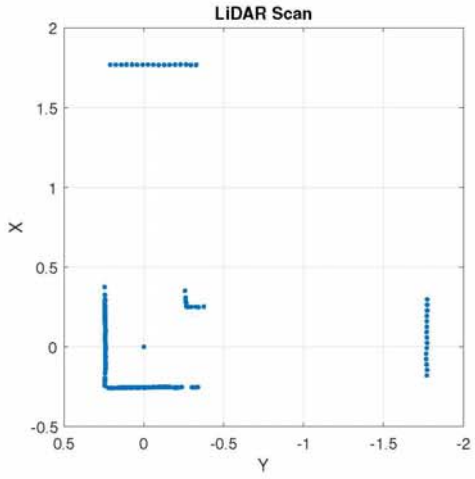


Fig. 8. LiDAR-based view of the corridor from the initial pose - in [m]

time. As the robot adds the new measurement, the position uncertainty value for all landmarks decreases - the fourth landmark having the largest decrease as it became directly connected via a measurement to the initial node from the state where it has two intermittent nodes.

IV. Conclusion and Future Work

The proposed orientation correction method is based on a long term memory for each visited landmark via LiDAR-scans. It enables long-term stability and map consistency by orientation-correction at every landmark re-visitation while minimizing the computational overhead. The system was tested in a laboratory environment with a single agent with excellent results. Further plans include testing on publicly available datasets and testing with multiple agents so that the environment includes more complex landmarks.

Acknowledgment

The research was supported by the EFOP-3.6.1-16-2016-00014 project - financed by the Ministry of Human Capacities of Hungary.

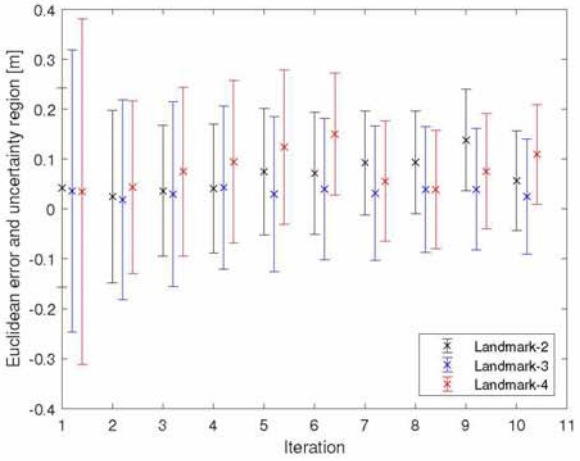


Fig. 9. Landmark position estimation errors with uncertainty regions

References

[1] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[2] L. Han, L. Xu, D. Bobkov, E. Steinbach, and L. Fang, "Real-time global registration for globally consistent rgb-d slam," *IEEE Transactions on Robotics*, vol. 35, no. 2, pp. 498–508, 2019.

[3] C. Pang, Y. Tan, S. Li, Y. Li, B. Ji, and R. Song, "Low-cost and high-accuracy LiDAR SLAM for large outdoor scenarios," in *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2019, pp. 868–873.

[4] M. Aldibaja, N. Suganuma, R. Yanase, and K. Yoneda, "Reliable graph-slam framework to generate 2d lidar intensity maps for autonomous vehicles," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*. IEEE, 2020, pp. 1–6.

[5] C. Chen, L. Pei, C. Xu, D. Zou, Y. Qi, Y. Zhu, and T. Li, "Trajectory optimization of lidar slam based on local pose graph," in *China Satellite Navigation Conference*. Springer, 2019, pp. 360–370.

[6] M. Vlaminck, H. Luong, and W. Philips, "Have I seen this place before? a fast and robust loop detection and correction method for 3D LiDAR SLAM," *Sensors*, vol. 19, no. 1, p. 23, 2019.

[7] R. Ren, H. Fu, and M. Wu, "Large-scale outdoor SLAM based on 2D LiDAR," *Electronics*, vol. 8, no. 6, p. 613, 2019.

[8] M. Holder, S. Hellwig, and H. Winner, "Real-time pose graph SLAM based on radar," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 1145–1151.

[9] Z. Ren, L. Wang, and L. Bi, "Robust GICP-based 3D LiDAR SLAM for underground mining environment," *Sensors*, vol. 19, no. 13, p. 2915, 2019.

[10] H.-C. Yen, C.-C. Wang, and C.-F. Chou, "Orientation constraints for wi-fi slam using signal strength gradients," *Autonomous Robots*, pp. 1–12, 2020.

[11] S. Agarwal, V. Shree, and S. Chakravorty, "RFM-SLAM: Exploiting relative feature measurements to separate orientation and position estimation in SLAM," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6307–6314.

[12] G. Péter and B. Kiss, "Compact pose-graph slam framework based on algebra with embedded uncertainty," in *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2018, pp. 57–62.

[13] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[14] P. Biber and W. Straßer, "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)*, vol. 3. IEEE, 2003, pp. 2743–2748.