

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,400

Open access books available

174,000

International authors and editors

190M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Autonomous Mobile Mapping Robots: Key Software Components

Janusz Będkowski and Jacek Szklarski

Abstract

This chapter discusses key software components of autonomous mobile mapping robots equipped with an inertial measurement unit (IMU) and light detection and ranging (LiDAR). In recent years, new LiDARs with nonrepetitive scanning pattern have appeared in the market. They are also equipped with an IMU; thus, the front end of simultaneous localization and mapping (SLAM)—a robust LiDAR-inertial odometry framework—significantly improves unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAV) in 3D mapping scenarios. Our study incorporates FAST-LIO as the front end of SLAM. The main focus is a lightweight back-end implementation of pose graph simultaneous localization and mapping (SLAM). It is an alternative solution to state-of-the-art g2o or GTSAM implementations. We also elaborate on iterative closest point, normal distributions transform, and their extension for multiview 3D data registration/refinement. It is based on C++ using Eigen library. This chapter also discusses path planning in already mapped environment. All software components are available as open-source projects.

Keywords: multiview normal distributions transform, SLAM, path planning, coverage

1. Introduction

This chapter presents key software components for autonomous mobile mapping robots shown in **Figure 1** (software components are available in [1, 2]). This set of consecutive functionalities is composed of robust light detection and ranging (LiDAR)-inertial odometry FAST-LIO [3], pairwise matching algorithm (iterative closest point [ICP] or normal distributions transform [NDT]) [1] for minimizing an error for loop closures, pose graph simultaneous localization and mapping (SLAM) [2], final refinement (multiview NDT) [1], and path planning. These functionalities are the core components for autonomous mobile mapping robots equipped with an inertial measurement unit (IMU) and LiDAR.

Autonomous mobile mapping robots have already been widely investigated within the context of commercial applications, for example, power line inspection [4], smart factory production [5], offshore oil plant [6], and nuclear power plant (NPP) inspection [7]. Robots improve rescue missions in hazardous environments [8]. The research related to COVID-19 and public support for autonomous technologies shows great interest in artificial intelligence (AI) direction [9]. There are plenty of areas for

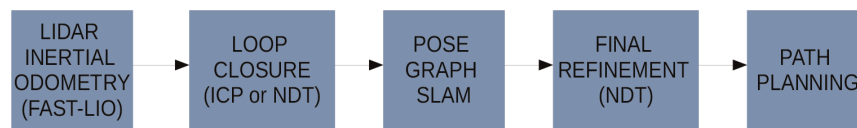


Figure 1.
Scheme of key software components elaborated in this chapter.

autonomous mobile mapping robots such as floor scrubbing, delivery, warehouse, and service robots. The rapid improvements in autonomous mobile mapping are evident for LiDAR with a nonrepetitive scanning pattern [10]. This LiDAR is capable of acquiring massive 3D data in short time with a limited field of view. The advantage is the long range, even up to 500 meters and high density of points covering entire measurement cone in short time. Narrow field of view can be extended by multiple LiDAR systems [11]. Owing to synchronized IMU data and robust feature classification, a robust LiDAR-inertial odometry framework significantly improves unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs) in 3D mapping scenarios [3, 12]. Moreover, the overall cost of such LiDAR is rather small compared with its competition. It is advised to study [13] for further precision and accuracy comparative evaluation.

SLAM is a core component of the autonomous mobile mapping robot that builds a map based on the estimated trajectory and estimates this set of consecutive poses based on this map [14]. SLAM is composed of front-end capable reconstructing smooth and continuous trajectory from onboard sensors. This trajectory is affected by constantly growing error. To reduce this error, the additional measurements should be incorporated into the back end as the so-called loop closure. The back end is typically solved using the pose graph SLAM method implemented in g2o [15] and GTSAM [16] frameworks. An alternative implementation is available in [2] that extends the possibility of rotation matrix parameterizations. Pose graph SLAM optimizes a graph composed of vertices (poses) and edges (consecutive odometry readings, loop closures, and other constraints); thus, it is supposed to preserve the shape of an initial trajectory (motion model) and minimize an error between observed (current relative pose) and measured (desired relative pose) loop closure edges. For the calculation of desired relative pose between two scans, an iterative closest point [17] or normal distributions transform [18] can be incorporated. The final step of the 3D mapping can be the final refinement of all 3D measurements performed with, for example, multiview normal distributions transform [19]. A similar approach is evident in general mobile mapping applications [20, 21].

The last functionality elaborated in this chapter is path planning being a fundamental software component of the autonomous mobile mapping system dedicated for missions where full coverage is desirable [22]. Examples are nuclear power plant inspection [23] and cleaning robotics [24]. In order to perform a mapping task, a mobile robot has to act according to some kind of a motion planning algorithm. This is also true for other related tasks such as inspecting, searching, cleaning, image mosaicking, etc. There are many factors that determine which algorithm should be used for the path plan generation:

- Is the map of the environment known in advance?
- What are the available sensors (LiDARs, RGB(D) cameras, proximity sensors, etc)?

- How many robots participate in the task? If more than one, what are the means of communication between units?
- How the environment map is represented? Is the map 2D or 3D? Does robot move on a planar space or is it a UAV (six degrees of freedom)?
- What are the available computational resources for path calculations and what is the required working regime: real-time online or offline planning?
- What is the ratio of covering radius to the size of the robot?

All the aspects mentioned above profoundly impact the algorithms necessary to guide the mobile robot position. For example, if the map of the environment is not known in advance, one should focus on a version of SLAM with exploration algorithm. If, on the other hand, the map is known and offline planning is allowed, one may use a solver that generates plan giving some near-optimal path plan. The optimal criteria can also vary depending on a specific application: minimization of coverage time, minimization of energy, prioritization of certain regions, etc. If there is a group of robots involved, the question of equal workload distribution must also be addressed.

A path planning algorithm should also take into account kinematic properties of robots involved in the process. A different algorithm will be applied for path generation for aircraft taking aerial images for mosaicking and a different one for a mobile robot cleaning floor in a warehouse.

For the autonomous mapping of unknown environments, in order to obtain a map of the environment, a robot should be able to simultaneously localize and map and, at the same time, explore the environment. Path planning, in this context, is related to the exploration process. The robot should gradually explore the environment, and a map is incrementally built for new poses, while the robot localizes itself in this map. New, temporary goal poses are often chosen by means of frontier extraction [25]. Frontier areas represent boundary regions between known (mapped) and unknown regions of the environment. Robot motion between its current position and such temporary goal is realized by a typical path planning, which navigates between waypoints while avoiding obstacles. Details regarding the frontier exploration vary depending on application and may also include exploration in 3D, for example, [26].

One of the fundamental applications of mobile robots is to perform a coverage task. Such tasks that the robot will visit points in the environment, eventually visiting (or observing) the entire region of interest. This is necessary for tasks like cleaning, mowing, harvesting, planting, spraying, mapping, searching, painting, mosaicking, etc. Normally, the first step for such application is to obtain the map, for example, by means of exploration with SLAM. If the map is known, and the robot is able to localize itself using the map, a coverage path planning (CPP) algorithm should be employed in order to find consecutive waypoints. The area of interest will be covered entirely after the robot will visit all the points. The problem of CPP is well known and well studied in the field of robotics [27, 28]. Nevertheless, it remains challenging and even the simplest variants, like the lawnmower problem, are NP-hard [29]. There exist a large number of exact, approximate, and heuristic algorithms to solve many variations of both types of CPP [30–32].

The rest of this chapter is organized as follows. Section 2 discusses key software components for SLAM. Section 3 addresses path planning in known environments.

Section 4 shows an example of mobile mapping applications. Finally, Section 5 concludes this chapter.

2. Key software components for SLAM

In this section, the key software components are elaborated. The fundamental element of SLAM is an observation equation. The set of optimization equations builds an optimization system. Finding an optimal solution results in the final map and trajectory. The proposed lightweight implementation uses symbolic computing in Python (SymPy) [33] to generate C++ code for each observation equations. An open-source project is available in [2].

2.1 Observation equations

Observation Eq. (1) is composed of a target value y_i , a model function $\Psi_{[\beta]}(\mathbf{x}_i)$, and its *residual* r_i defined as the difference between the target value and the value of the model function for \mathbf{x}_i and state vector β

$$\underbrace{r_i}_{\text{residual}} = \underbrace{y_i}_{\text{target value}} - \underbrace{\Psi_{[\beta]}(\mathbf{x}_i)}_{\text{model function}} \quad (1)$$

where β is the vector of n optimized parameters. The weighted nonlinear least squares optimization method finds the optimal n parameter values (β) by minimizing the objective function being a sum of C squared residuals

$$\text{Sum} = \sum_{i=1}^C r_i^2 = \underbrace{\sum_{i=1}^C (y_i - \Psi_{[\beta]}(\mathbf{x}_i))^2}_{\text{objective function}} \quad (2)$$

Therefore, the optimization problem is defined as

$$\beta^* = \min_{\beta} \sum_{i=1}^C (y_i - \Psi_{[\beta]}(\mathbf{x}_i))^2 \quad (3)$$

where there are C observation equations. It is efficiently solved using the iterative Levenberg-Marquardt algorithm [34]. A single k th iteration provides an update for β given as

$$\beta^{k+1} = \beta^k + \left(\mathbf{J}^T \mathbf{W} \mathbf{J} + \lambda \mathbf{I} \right)^{-1} \mathbf{J}^T \mathbf{W} \mathbf{r}(\beta^k) \quad (4)$$

where \mathbf{I} is the identity matrix, \mathbf{J} is the Jacobian of the model function, and \mathbf{W} is the weight matrix modeling the impact of the observation equation into the optimization process. λ starts from an initial small value. During the optimization process, λ increases once $\text{Sum} = \sum_{i=1}^C r_i^2$ decreases; otherwise, λ decreases and the optimization

process starts from the previous step. The observation equation for pose graph SLAM is given as

$$\underbrace{\begin{bmatrix} t_x^\delta \\ t_y^\delta \\ t_z^\delta \\ \omega^\delta \\ \varphi^\delta \\ \kappa^\delta \end{bmatrix}}_{\text{residuals}} = \underbrace{\begin{bmatrix} t_x \\ t_y \\ t_z \\ \omega \\ \varphi \\ \kappa \end{bmatrix}}_{\text{target values}} - \underbrace{m2v_{[t_x, t_y, t_z, \omega, \varphi, \kappa]}([\mathbf{R}, \mathbf{t}]_{12})}_{\text{model function}} \quad (5)$$

where $[t_x^\delta \ t_y^\delta \ t_z^\delta \ \omega^\delta \ \varphi^\delta \ \kappa^\delta]^\top$ are *residuals*, $[t_x \ t_y \ t_z \ \omega \ \varphi \ \kappa]^\top$ are *target values*, and $m2v_{[\beta]}([\mathbf{R}, \mathbf{t}]_{12})$ is the *model function*. Target values describe the desired edge (relative pose expressed as translation (t_x, t_y, t_z) and orientation $(\omega, \varphi, \kappa)$ between two optimized vertices (poses) of the graph. Relative pose $[\mathbf{R}, \mathbf{t}]_{12}$ from pose $[\mathbf{R}, \mathbf{t}]_1$ to pose $[\mathbf{R}, \mathbf{t}]_2$ is given as

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{1 \times 3} & 1 \end{bmatrix}_{12} = \left(\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{1 \times 3} & 1 \end{bmatrix}_1 \right)^{-1} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^{1 \times 3} & 1 \end{bmatrix}_2. \quad (6)$$

Function $m2v_{[\beta]}([\mathbf{R}, \mathbf{t}]_{12})$ retrieves $\beta = (t_x, t_y, t_z, \omega, \varphi, \kappa)^\top$ for the Tait-Bryan parametrization of the rotation matrix. This parameterization is essential to preserve the orthonormality of the rotation matrix during the optimization process. It is important to notice that other parameterizations exist, such as quaternion, Rodrigues, etc., but this is not the main topic of this chapter.

The iterative closest point algorithm finds the relative pose between two point clouds by incorporating the following source-point-to-target-point observation equation:

$$\underbrace{\begin{bmatrix} x^\delta \\ y^\delta \\ z^\delta \end{bmatrix}}_{\text{residuals}} = \underbrace{\begin{bmatrix} x^{tg} \\ y^{tg} \\ z^{tg} \end{bmatrix}}_{\text{target values}} - \underbrace{\Psi_{[\mathbf{R}, \mathbf{t}]}(\mathbf{R}, \mathbf{t}, x^l, y^l, z^l)}_{\text{model function}} \quad (7)$$

where, $[x^\delta \ y^\delta \ z^\delta]^\top$ are *residuals*, $[x^{tg} \ y^{tg} \ z^{tg}]^\top$ are *target values*, and $\Psi_{[\mathbf{R}, \mathbf{t}]}(\mathbf{R}, \mathbf{t}, x^l, y^l, z^l)$ is the *model function* that transforms 3D points (x^l, y^l, z^l) expressed in the local coordinate system into the global one.

The normal distributions transform algorithm is an alternative solution for pairwise matching with ICP, and it can be easily extended for multiview point cloud data registration (final refinement of the 3D map). It decomposes the 3D scene into a regular grid where for each cell, the centroid $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$ are calculated with formulas (8) and (9). These formulas incorporate all points \mathbf{P}_k^g in a single cell expressed in the global coordinate system

$$\boldsymbol{\mu} = \frac{1}{m} \sum_{k=1}^m \mathbf{P}_k^g \quad (8)$$

$$\Sigma = \frac{1}{m-1} \sum_{k=1}^m (\mathbf{P}_k^g - \boldsymbol{\mu})(\mathbf{P}_k^g - \boldsymbol{\mu})^\top. \quad (9)$$

The NDT observation equation is given as

$$\underbrace{\begin{bmatrix} x^\delta \\ y^\delta \\ z^\delta \end{bmatrix}}_{\text{residuals}} = \underbrace{\begin{bmatrix} \mu_x \\ \mu_y \\ \mu_z \end{bmatrix}}_{\text{target values}} - \underbrace{Y_{[\mathbf{R}, \mathbf{t}]}(\mathbf{R}, \mathbf{t}, x^l, y^l, z^l)}_{\text{model function}} \quad (10)$$

where the target value is $\boldsymbol{\mu}$ and $\Sigma^{-1} = \mathbf{W}$ for each NDT observation equation incorporated in the Levenberg–Marquardt algorithm from eq. (4).

3. Path planning in known environments

In this section, we will focus on a practical example of a cleaning robot whose task is to clean a large area. Therefore, one needs to apply a path planning algorithm for a single device that moves in a known environment, and the map of static obstacles is known in advance (c.f. [35]).

3.1 Map decomposition

A cleaning robot has the coverage area equal to the area of its cleaning/sweeping device. For industrial cleaners, it is a dedicated brush equipped with a water and soap reservoir. Consequently, the size of the coverage area, being a circle with radius r_{cov} , is comparable with the size of the robot. The area to be cleaned is a large warehouse with the total area A , so it should be assumed that $r_{\text{cov}} \ll A$. From this assumption, it follows that any grid-based approach for planning should be avoided. This is because in most grid-based methods, the time for finding a solution grows significantly with the grid size (for many methods even exponentially [28]). For the discussed problem, the number of grid cells would be too large to come up with a feasible solution.

It should be noted that this is not always the case for coverage problems. For example, photo mosaicking has much larger r_{cov} than the size of the robots, for example, UAVs equipped with cameras. Area being photographed is much larger than the area of the device. Such problems may use a different planning approach than the one for cleaning robots.

Consequently, a solution based on the geometric decomposition of the grid map into a set of polygons should be considered. Afterward, a path on this set of polygon is found in a way that minimizes a given optimization criterion, in this case the total coverage time. The pipeline for the system is depicted in **Figure 2**.

After mapping the environment with SLAM and the proper optimization, a grid map of static obstacles is obtained. There are a number of ways to convert such a grid map into a set of polygons. This process is known as map decomposition. The most well-known method of decomposition is the *trapezoidal decomposition* where the grid map is “scanned” line by line along one direction and trapezoids are found, which cover the free space completely. Afterward, trapezoids are covered by simple back-and-forth motion patterns.

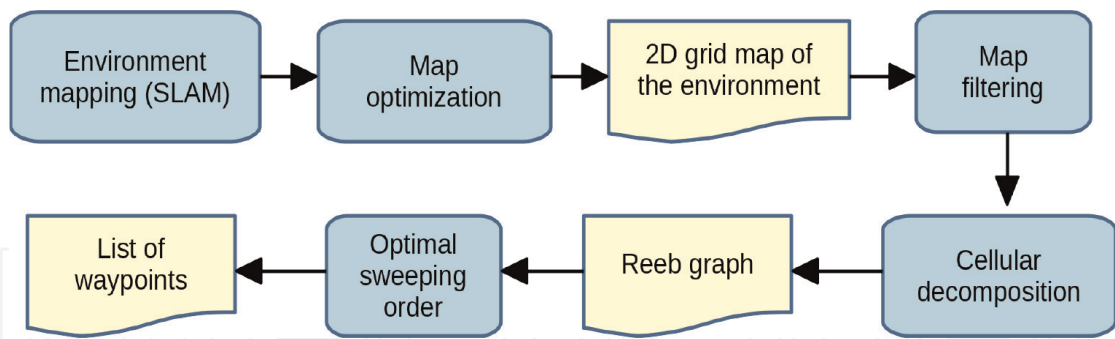


Figure 2. A processing pipeline for the generation of a coverage path plan for a single robot operating in an environment that is first mapped with the SLAM method.

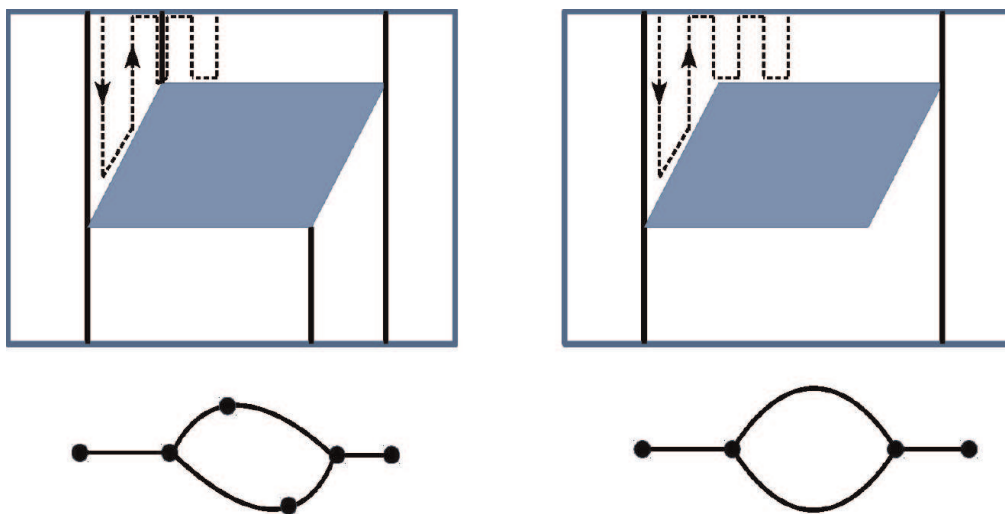


Figure 3. An example of trapezoidal and boustrophedon decompositions into cells together with their Reeb graphs (bottom). The latter decomposition allows for better sweeping pattern fit (the dashed line).

The main drawback of this simple decomposition is the fact that it generates only convex polygons, and therefore, it results in a large number of polygons and suboptimal sweeping patterns (c.f. **Figure 3**). Another possibility is to apply the so-called *boustrophedon cellular decomposition* (BCD), which also generates nonconvex cells [36]. However, these polygons can also be covered only by zig-zag motions, usually in a more efficient way than for the trapezoidal decomposition. It should be noted that some more sophisticated decompositions have been proposed in the literature, for example, [37]. In such an approach, optimization is focused in the decomposition process itself. Here, however, we optimize the path by finding proper sweeping patterns at a later stage.

After the decomposition, a set of polygons is obtained. A geometrical relation between these polygons may be represented by the so-called Reeb graph. The Reeb graph is a special type of a graph for environment representation, in which each link corresponds to a polygon and each node represents an adjacency between the polygons. Consequently, the problem may be treated with the help of existing solutions known from graph theory. This can be done by formulating the optimization problem into a variant of the traveling salesman problem and employing some known efficient solvers.

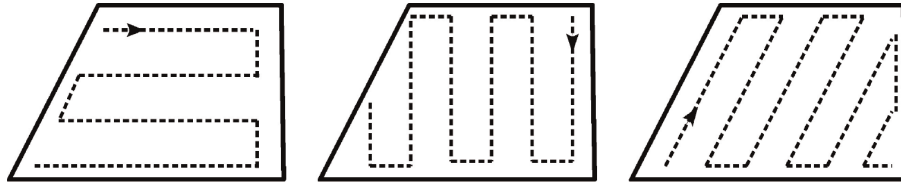


Figure 4.

For each cell, the coverage time is determined by sweeping direction and entry point into the cell (i.e., start and end vertices). The figure depicts some possible entry/exit points and directions for a sample trapezoid.

3.2 Finding near-optimal sweeping patterns

Let us consider the covering process for a single polygon. Usually, in order to minimize the coverage time, one needs to minimize the number of turns since, for each turn, a robot must slow down, stop, turn, and accelerate again to its maximum velocity. For a single polygon, various points of entrance for the zig-zag pattern should be considered (see **Figure 4**).

After the decomposition process, the entire environment is represented as a Reeb graph. The order of visiting all the links, that is, polygons, determines entry points to each of them and, therefore, the time cost associated with covering it. It can be shown that finding the minimum of the total time required for covering all the polygons is equivalent to solving the equality generalized traveling salesman problem (EG-TSP) [38]. This is an NP-hard problem for which approximate heuristic solvers may be applied. The results presented in this section are obtained with the use of a memetic solver, as proposed in [38].

4. Example applications

4.1 Robust LiDAR-inertial odometry and multiview NDT

Figure 5 demonstrates the result of FAST-LIO [3] as the 3D point cloud of underground garage recorded using Livox AVIA LiDAR. This robust LiDAR-inertial odometry provides an input for multiview NDT shown in **Figure 6**.

4.2 Pose graph SLAM

This section demonstrates the pose graph SLAM functionality available with data in [2]. **Figure 7** demonstrates the 2D case and **Figure 8** is related to the 3D case. Pose graph SLAM implementation efficiently solves the optimization problem represented as a consecutive set of poses (trajectory) connected via odometry readings (edges) and loop closure edges. This is a core component of the autonomous mobile mapping robot.

4.3 Final refinement with NDT

Multiview normal distributions transform 3D data registration is capable to increase the accuracy of the 3D map, as shown in **Figure 6**. The implementation is rather offline since it requires plenty of calculations. These calculations are related

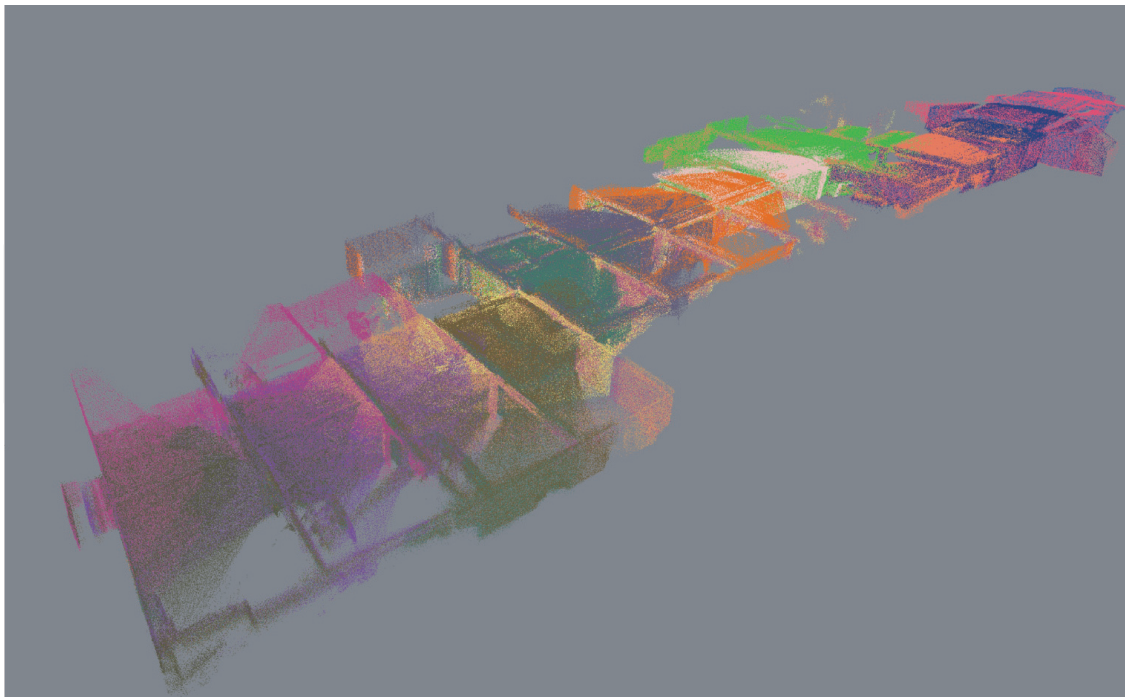


Figure 5.
Result of robust LiDAR-inertial odometry FAST-LIO [3] as the 3D point cloud of underground garage.

mostly to 3D data decomposition, where for each 3D bucket, the mean value and covariance are calculated. This method is efficient mostly for urban environments with many planar shapes.

4.4 Path planning

As an example of a real-world application of a coverage task, let us consider the cleaning process of an underground garage. First, the garage is scanned with 3D laser scanners; it is optimized and flattened to a 2D obstacle map (see **Figure 9**, top). Afterward, the map is used for robot motion planning and navigation.

The result of boustrophedon decomposition for this map is shown in **Figure 9** (middle). In this particular case, it consists mostly of rectangles. The next stage is to find the covering path that connects all these polygons. In order to formally state the optimization goal, one needs to define kinematic characteristics of the robot. Here, we assume realistic parameters: $a_{\max} = 0.3 \text{ ms}^{-2}$ and $v_{\max} = 1 \text{ ms}^{-1}$. This corresponds to real devices that are being used for the cleaning tasks [35]. After using the memetic solver for the associated EG-TSP, a trajectory for the robot is obtained. It is shown at the bottom of the figure. For the depicted scale and the assumed kinematic model, the total time for coverage is 2302 s in this case.

In order to validate the approach for path planning and to estimate its usefulness, one should use a large number of maps, perform planning, and measure efficiency. One possibility is to use a synthetic albeit realistic set of layouts provided by Li et al. [39]. Based on real experiments with LiDARs, the authors have developed a method to generate about 60,000 various maps, which can be used by researchers to test various algorithms. Some example layouts together with the planned coverage paths are depicted in **Figure 10**.

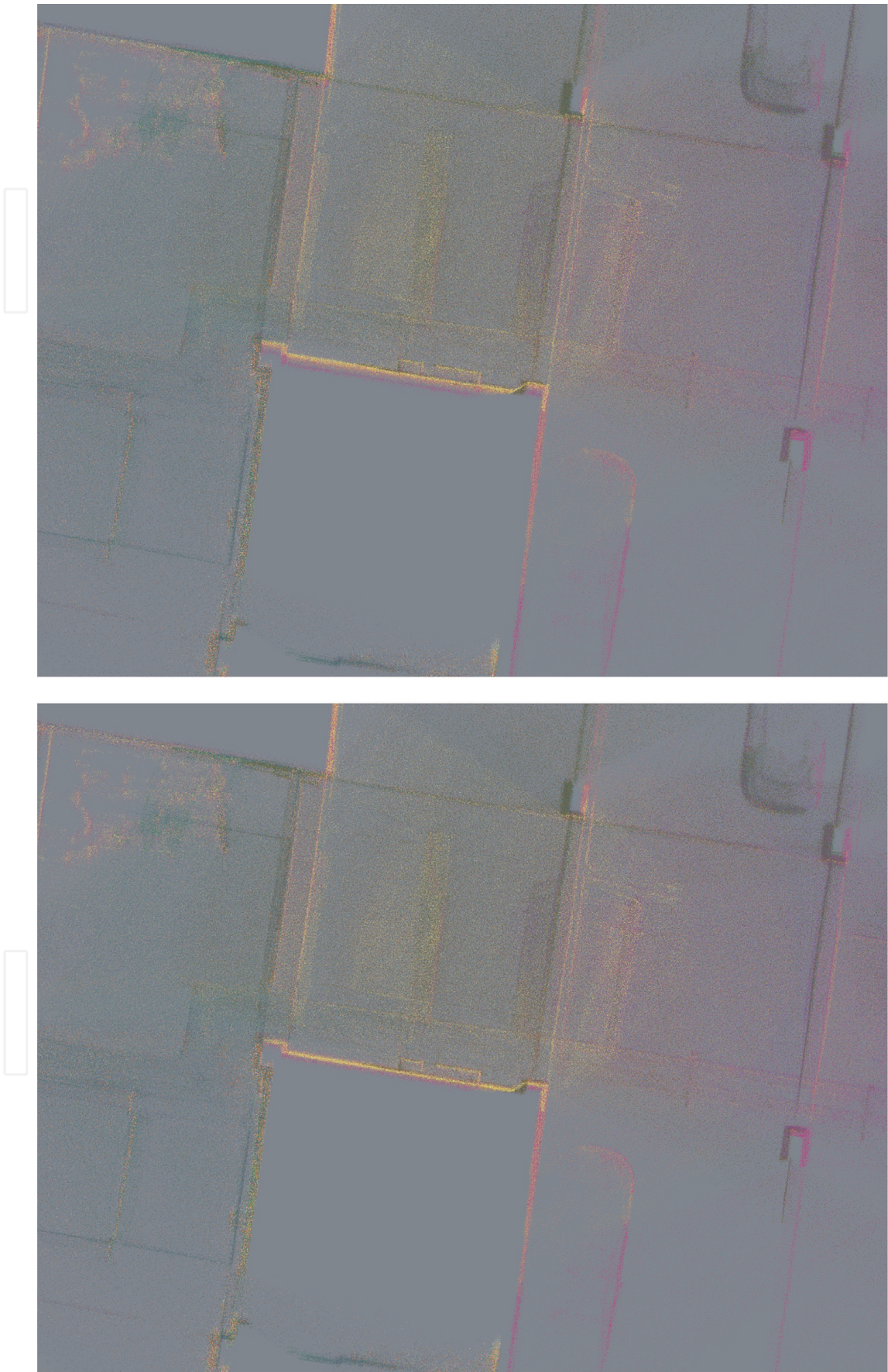


Figure 6.
Top: input data produced by robust LiDAR-inertial odometry FAST-LIO [3] from Figure 5. Bottom: result of final refinement with multiview NDT.



Figure 7.
Top: input data for pose graph SLAM (purple dots: graph vertices, blue lines: graph edges). Bottom: result of pose graph SLAM, 2D case.

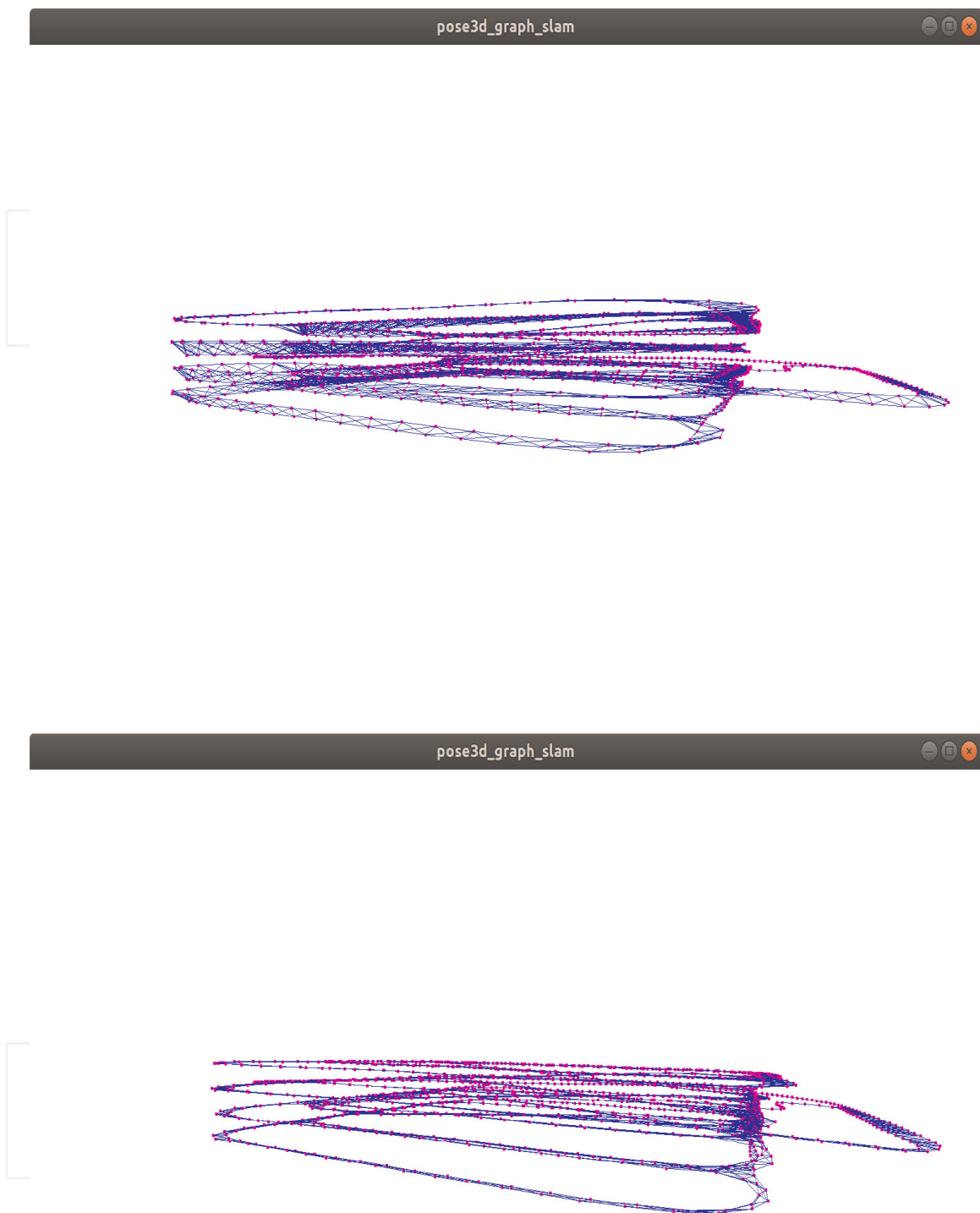


Figure 8.
Top: input data for pose graph SLAM (purple dots: Graph vertices, blue lines: Graph edges). Bottom: result of pose graph SLAM, 3D case.

5. Conclusion

This chapter elaborates key software components of autonomous mobile mapping robots equipped with Livox AVIA LiDAR. It is new LiDAR with a nonrepetitive scanning pattern equipped also with the IMU. LiDAR and IMU are synchronized; thus, this advantage is addressed by the robust LiDAR-inertial odometry framework FAST-LIO. It improves unmanned ground vehicles (UGVs) and unmanned aerial

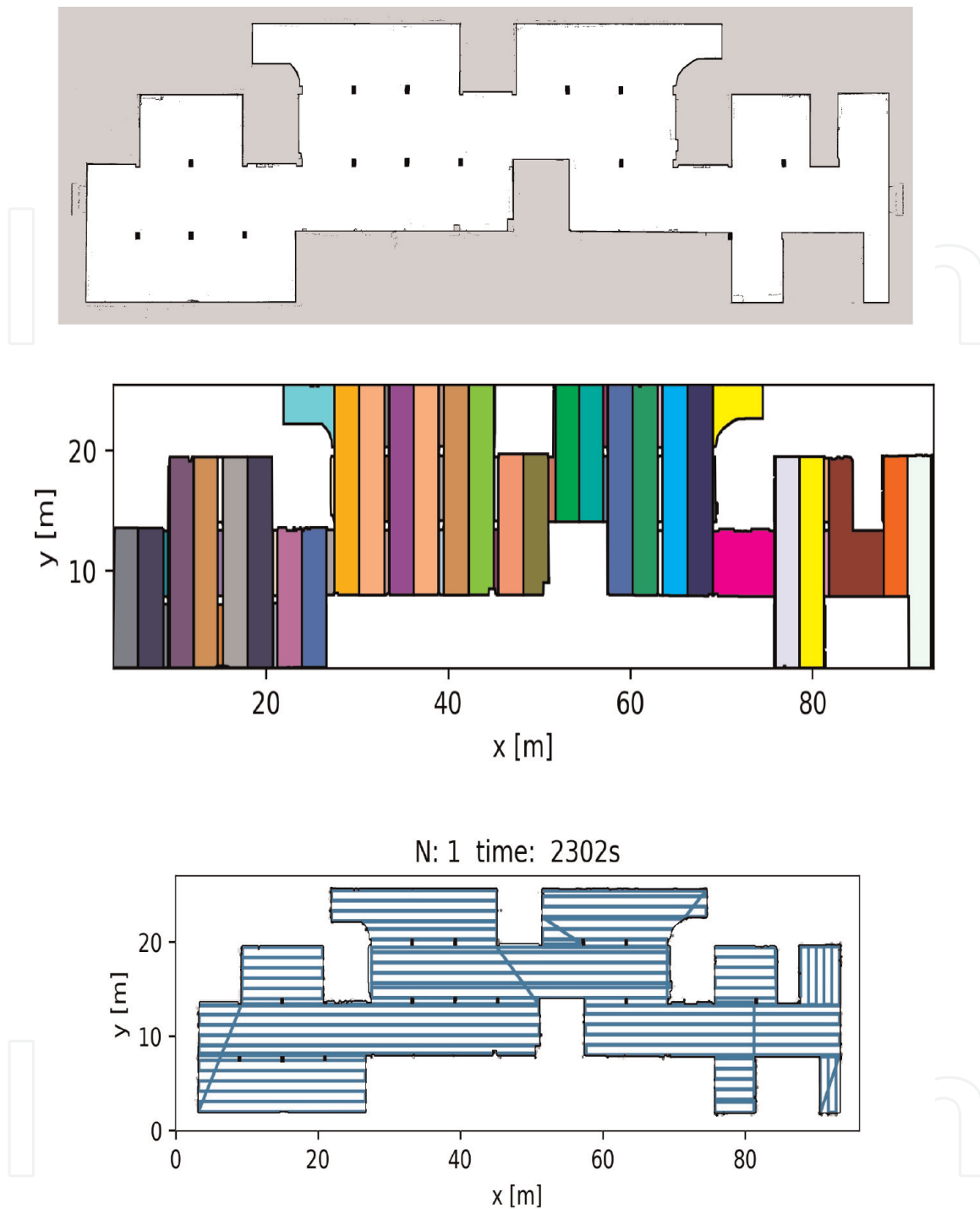


Figure 9. Top: a grid representing obstacles in an underground garage. Pixel size is $3\text{ cm} \times 3\text{ cm}$. Middle: the result of a boustrophedon decomposition. Bottom: A trajectory for a single robot.

vehicles (UAVs) in 3D mapping scenarios. Our study incorporates this robust LiDAR-inertial odometry framework FAST-LIO as the front end of SLAM. The main focus is a lightweight back-end implementation of pose graph simultaneous localization and mapping (SLAM). This lightweight implementation is an alternative solution to state-of-the-art g2o or GTSAM implementations. We also elaborate iterative closest point, normal distributions transform, and their extension for multiview 3D data registration/refinement. It is based on C++ using Eigen library. This chapter also discusses

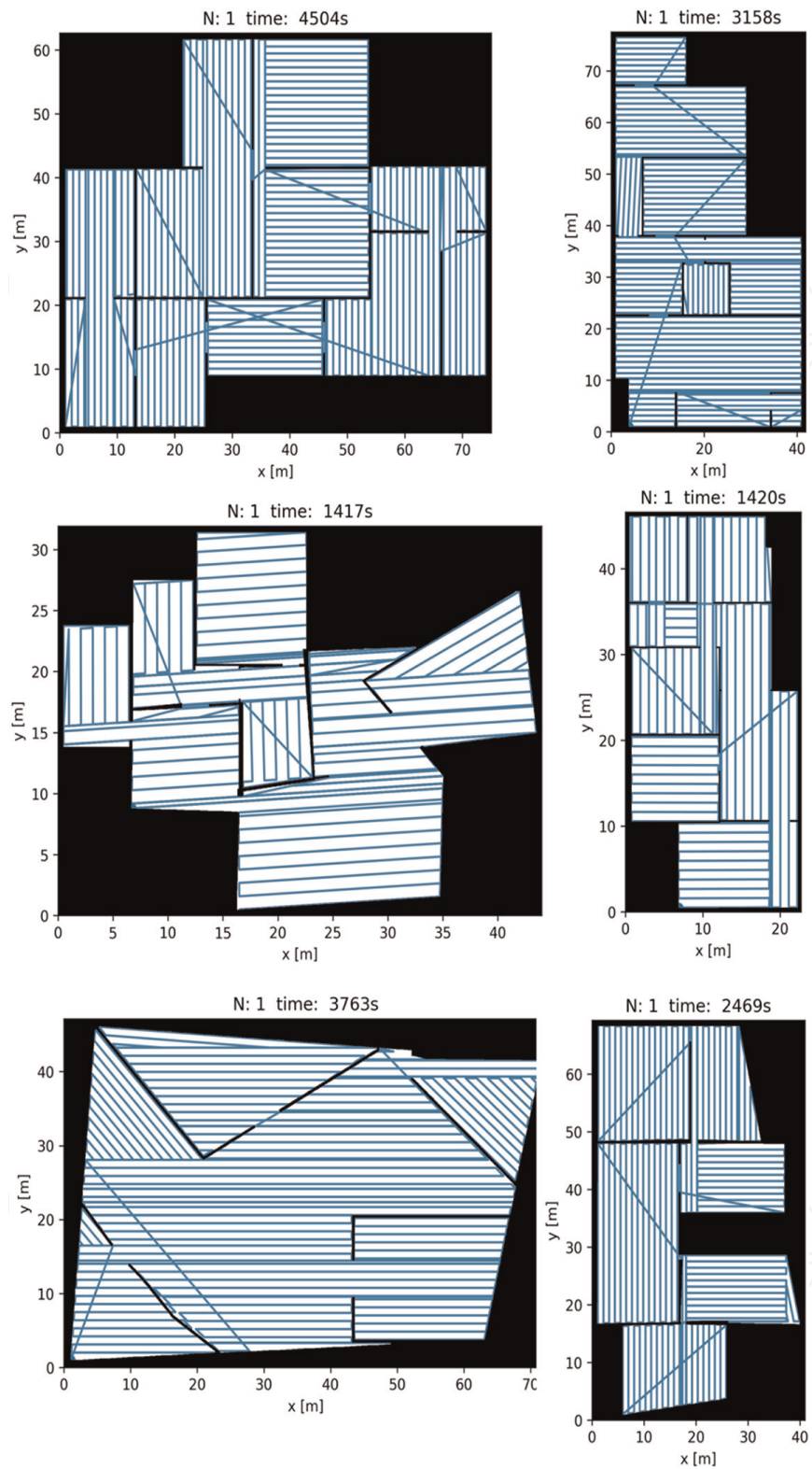


Figure 10. Covering path plans for a small subset of the realistic, synthetic dataset of building layouts [39]. The covering area is a circle with diameter equal to 0.5 m. Notice various scales for the x and y axes and—consequently—various times required for the complete coverage (indicated above the plots).

path planning in already mapped environment. All software components are available as an open-source project. This chapter provides insights for useful software components for building autonomous mobile mapping robots.

Acknowledgements

The authors acknowledge the financial support of National Centre for Research and Development, project POIR.01.01.01-00-0206/17”Designing an autonomous platform which operates in an industrial production environment.”

IntechOpen


IntechOpen

Author details

Janusz Będkowski* and Jacek Szklarski
Institute of Fundamental Technological Research, Polish Academy of Sciences, Poland

*Address all correspondence to: januszbedkowski@gmail.com

IntechOpen

© 2023 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Janusz Bedkowski. Hdmapping. 2022. Available from: <https://github.com/MapsHD/HDMapping>
- [2] Janusz Bedkowski. Observation equations. 2022. Available from: github.com/JanuszBedkowski
- [3] Wei Xu and Fu Zhang. Fast-Lio: A Fast, Robust Lidar-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter, 2020
- [4] Yang L, Fan J, Liu Y, Li E, Peng J, Liang Z. A review on state-of-the-art power line inspection techniques. *IEEE Transactions on Instrumentation and Measurement*. 2020;**69**(12):9350-9365
- [5] Hercik R, Byrtus R, Jaros R, Koziorek J. Implementation of autonomous mobile robot in smartfactory. *Applied Sciences*. 2022;**12**(17):8912
- [6] Nagatani K, Endo D, Watanabe A, Koyanagi E. Design and development of explosion-proof tracked vehicle for inspection of offshore oil plant. In: Hutter M, Siegwart R, editors. *Field and Service Robotics, Results of the 11th International Conference, FSR 2017, Zurich, Switzerland, 12–15 September 2017*. Vol. volume 5 of Springer Proceedings in Advanced Robotics. Springer; 2017. pp. 531-544
- [7] Zhang Zhonglin F, Bin LL, Encheng Y. Design and function realization of nuclear power inspection robot system. *Robotica*. 2021;**39**(1):165-180
- [8] Nagatani K, Kiribayashi S, Okada Y, Otake K, Yoshida K, Tadokoro S, et al. Emergency response to the nuclear accident at the Fukushima daiichi nuclear power plants using mobile rescue robots. *Journal of Field Robotics*. 2013;**30**(1):44-63
- [9] Horowitz MC, Kahn L, Macdonald J, Schneider J. Covid-19 and public support for autonomous technologies—Did the pandemic catalyze a world of robots? *PLoS One*. 2022;**17**(9):1-18
- [10] Lin J, Zhang F. R³ live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package. In: 2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23–27, 2022. IEEE; 2022. pp. 10672-10678
- [11] Wang Y, Lou Y, Zhang Y, Song W, Huang F, Zhiyong T. A robust framework for simultaneous localization and mapping with multiple non-repetitive scanning lidars. *Remote Sensing*. 2021;**13**(10):2015
- [12] Li K, Li M, Hanebeck UD. Towards high-performance solid-state-lidar-inertial odometry and mapping. *IEEE Robotics and Automation Letters*. 2021;**6**(3):5167-5174
- [13] Kelly C, Wilkinson B, Abd-Elrahman A, Cordero O, Andrew Lassiter H. Accuracy assessment of low-cost lidar scanners: An analysis of the velodyne hdl32e and livox mid40 temporal stability. *Remote Sensing*. 2022;**14**(17):4220
- [14] Thrun S. *Simultaneous Localization and Mapping*. Berlin Heidelberg, Berlin, Heidelberg: Springer; 2008. pp. 13-41
- [15] Kümmerle R, Grisetti G, Strasdat H, Konolige K, Burgard W. G2o: A general framework for graph optimization. In: ICRA. IEEE; 2011. pp. 3607-3613
- [16] Michael Kaess. *Gtsam library*, 2015
- [17] Besl PJ, McKay ND. A method for registration of 3-d shapes. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence. 1992;**14**(2): 239-256

[18] 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). Vol. 3. 2003. pp. 2743-2748

[19] Jihua Zhu, Di Wang, Jiayi Mu, Huimin Lu, Zhiqiang Tian, and Zhongyu Li. 3d mndt:3d Multi-View Registration Method Based on the Normal Distributions Transform, 2021

[20] Bosse M, Zlot R. Continuous 3d scan-matching with a spinning 2d laser. In: ICRA. IEEE; 2009. pp. 4312-4319

[21] Kaul L, Zlot R, Bosse M. Continuous-time three-dimensional mapping for micro aerial vehicles with a passively actuated rotating laser scanner. Journal of Field Robotics. 2016;**33**(1): 103-132

[22] Lin H-Y, Huang Y-C. Collaborative complete coverage and path planning for multi-robot exploration. Sensors. 2021; **21**(11):3709

[23] Iqbal J, Tahir AM, Islam R u, Nabi R u. Robotics for nuclear power plants — Challenges and future perspectives. In: 2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI). 2012. pp. 151-156

[24] Woohyeon Moon, Bumgeun Park, Sarvar Hussain Nengroo, Taeyoung Kim, and Dongsoo Har. Path planning of cleaning robot with reinforcement learning, 2022 IEEE International Symposium on Robotic and Sensors Environments (ROSE), Abu Dhabi, United Arab Emirates, IEEE, 2022

[25] Yamauchi B. A frontier-based approach for autonomous exploration. In: Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'. IEEE; 1997. pp. 146-151

[26] Belavadi SS, Beri R, Malik V. Frontier exploration technique for 3d autonomous slam using k-means based divisive clustering. In: 2017 Asia Modelling Symposium (AMS). IEEE; 2017. pp. 95-100

[27] Almadhoun R, Taha T, Seneviratne L, Zweiri Y. A survey on multi-robot coverage path planning for model reconstruction and mapping. SN Applied Sciences. 2019;**1**(8):1-24

[28] Yan Z, Jouandeau N, Cherif AA. A survey and analysis of multi-robot coordination. International Journal of Advanced Robotic Systems. 2013;**10**

[29] Arkin EM, Fekete SP, Mitchell JSB. Approximation algorithms for lawn mowing and milling. Computational Geometry. 2000;**17**(1-2):25-50

[30] Choset H. Coverage for robotics—a survey of recent results. Annals of Mathematics and Artificial Intelligence. 2001;**31**(1):113-126

[31] Galceran E, Carreras M. A survey on coverage path planning for robotics. Robotics and Autonomous Systems. 2013;**61**(12):1258-1276

[32] Saeedi S, Trentini M, Seto M, Li H. Multiple-robot simultaneous localization and mapping: A review. Journal of Field Robotics. 2016;**33**(1):3-46

[33] Meurer A, Smith CP, Paprocki M, Čertík O, Kirpichev SB, Rocklin M, et al.

Sympy: Symbolic computing in python.
PeerJ Computer Science. 2017;3:e103

[34] Marquardt DW. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*. 1963;11(2):431-441

[35] Szklarski J. Multi-robot coverage with reeb graph clustering and optimized sweeping patterns. *Computer Assisted Methods In Engineering And Science*. 2022;29(4):379-395

[36] Choset H. Coverage of known spaces: The boustrophedon cellular decomposition. *Autonomous Robots*. 2000;9(3):247-253

[37] Nielsen LD, Sung I, Nielsen P. Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges. *Sensors*. 2019;19(19):4165

[38] Bähnemann R, Lawrance N, Chung JJ, Pantic M, Siegwart R, Nieto J. Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem. In: *Field and Service Robotics*. Springer; 2021. pp. 277-290

[39] Li T, Ho D, Li C, Zhu D, Wang C, Meng MQ-H. Houseexpo: A large-scale 2d indoor layout dataset for learning-based algorithms on mobile robots. In: *In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE; 2020. pp. 5839-5846