Loop Closure Detection in Closed Environments

Nils Rottmann[1], Ralf Bruder[1], Achim Schweikard[1], Elmar Rueckert[1]

Abstract—Low cost robots, such as vacuum cleaners or lawn mowers employ simplistic and often random navigation policies. Although a large number of sophisticated mapping and planning approaches exist, they require additional sensors like LIDAR sensors, cameras or time of flight sensors. In this work, we propose a loop closure detection method based only on odometry data which can be generated using low-range or binary signal sensors together with simple wall following techniques. We show how to include the detected loop closing constraints into a pose graph formulation such that standard pose graph optimization techniques can be used for map estimation.

We evaluate our map estimate and loop closure approach using both, simulation and a real lawn mower in complex and realistic environments. Our results demonstrate that our approach generates accurate map estimates on the basis of odometry data only. We further show that our assumption about the discriminative nature of neighboring poses in the pose graph is solid, even under large odometry noise. These improved map estimates provide the basis for smart navigation policies in low cost robots and extends their abilities to goal-directed behavior like pick and place or complete coverage path planning in realistic environments.

I. INTRODUCTION

During the last decade, research and development in the area of autonomous mobile robots have made significant progress. Nowadays robots such as vacuum cleaners or lawn mowers can be found in many households fulfilling their intended tasks. However, most of the robots employ thereby simplistic navigation strategies, such as a random walk, due to the lack of suitable maps and accurate sensors required for successful path planning. While most existing work for mapping and localization utilizes long-range sensors, such as LIDAR sensors, RGB-D cameras or time of flight sensors, robots for private households lack such sensor-richness. The reason for that is that such sensors are either too expensive, aiming for low acquisition and maintenance costs, or not suitable for outdoor environments, e.g. to reflections and direct sunlight. Moreover, simultaneous localization and mapping (SLAM) algorithms require certain amount of computational power which is often not available. However, intelligent navigation from low cost hardware is essential for mobile robots to enter our daily life. For example, autonomous lawn mowers employed with random walk policies are limited to simple environments, e.g. they can not enter small corridors.

In this paper, we propose a sophisticated mapping algorithm that can generate a map estimate of the environment based only on odometry data. Such data can be generated by following the wall or border line for an area of interest. Such wall following

[1]Institute for Robotics and Cognitive Systems, University of Luebeck, Ratzeburger Allee 160, 23562 Luebeck, Germany {rottmann, bruder, schweikard, rueckert}@rob.uni-luebeck.de

strategies can be executed using simple wall sensors, bumpers or signal wire sensors. The generated odometry data can be used for the proposed map estimation method. We will focus in this paper on how to achieve the most suitable map using only the recorded odometry data.

In order to reduce computational complexity, we first prune our problem using path segmentation. Thereby, we cluster the individual path points generated from the odometry data into straight line segments. Based on the pruned data set we generate a pose graph in which we search for loop closing constraints using shape comparison. We include these loop closing constraints into our pose graph formulation and optimize the graph using standard pose graph optimization techniques, such as the Levenberg-Marquardt algorithm. Finally, we introduce a measure for the mapping error based on the deviation of areas between the estimated map and the true shape of the environment.

We first evaluate our method in a complex simulation environment using standard motion models. Afterwards, we show how our method performs in a real robot, a Viking MI 422P, set up in a real garden environment. In order to represent real conditions as best as possible we learned the odometry parameters for the motion model by means of Log Likelihood estimation. Moreover, we show in simulations that our approach is robust even under a large amount of odometric drift, which is essential for successfully mapping outdoor environments.

A. Contributions and Organization

The contributions of the paper are two-fold. First, an efficient and simple method for loop closure detection using odometry only data is presented and second, a map evaluation scheme based on comparing two map areas, the estimated map and a groundtruth, is proposed. Both contributions are necessary to generate accurate map estimates with only low-range or binary sensors, which then enables low-cost robots to implement intelligent navigation and path planning strategies. These contributions are discussed in Section II. In Section III we evaluate our approach in a realistic mowing scenario and in a challenging simulated apartment environment. We conclude in Section IV.

B. Related Work

Most of the existing work on SLAM [20] or graph-based SLAM [9] relies on long-range sensors, such as LIDARs or cameras [3], [7], [15], [8]. Most of these approaches are using sensor fusion and probabilistic reasoning, e.g. particle filter [11] or extended kalman filter [1]. However, there are

some approaches which try to handle the SLAM problem using only sparse sensor data, e.g. [2], to avoid expensive sensing. Existing work for low-range sensors [22], [6], such as sonars or infrared sensors, requires linear features which are not necessarily present in outdoor environments. Indoor mapping with limited sensing using a wall following approach has been presented in [24]. However, this approach makes the assumption of an approximately rectilinear structure, which may be true for most indoor environments but not for outdoor applications. In contrast, our proposed approach is not restricted to such structural assumptions and can be used for either indoor or outdoor applications with arbitrary shapes.

Generating the path from odometry data leads to a pose-graph representation, which is often used for the SLAM problem [21], [17] and has been first introduced by Lu and Milios 1997, [19]. Pose graph optimization has been addressed in several studies, e.g TORO [10], g2o [16], iSAM2 [14] and LAGO [5]. Thereby, TORO is based on gradient descent and is an extension of Olson's algorithm [21]. It applies a tree-based parameterization to distribute residual errors across the graph that improves the performance. The "general graph optimization" framework, g2o, has been designed to perform the optimization of different least squares problems, which can be represented as a graph. It thereby relies on the Gauss-Newton method. The iSAM2 method applies Bayes trees using incremental variable re-ordering and fluid relinearization to solve sparse nonlinear incremental optimization problems. LAGO addresses the pose graph optimization problem by decoupling the orientation and translation. We use the simpler Levenberg-Marquardt algorithm [9], which worked reasonably well for pose graph optimization. There, the function is approximated by its first order Taylor expansion around the current initial guess $\hat{p}$ in order to then find the solution to the optimization problem iteratively.

In order to reduce computational complexity, we prune the data using path segmentation. Such a pruning can lead to a huge reduction of computational time at the price of a small increase in error [18].

II. METHODS

We start by introducing the standard pose graph formulation: Let $p = \{p_0, \ldots, p_N\}$ be a set of $N+1$ poses representing the position and orientation of a mobile robot in a two dimensional space, hence $p_i = [x_i^\top, \varphi_i]^\top$. Here, $x_i \in \mathbb{R}^2$ is the cartesian position of the robot and $\varphi_i \in [-\pi, \pi]$ the corresponding orientation as an euler angle with the integer $i = 0{:}N$. The relative measurement between two poses i and j is then given as

$$\boldsymbol{\xi}_{ij} = \begin{bmatrix} \boldsymbol{R}_i^\top (\boldsymbol{x}_j - \boldsymbol{x}_i) \\ \varphi_j - \varphi_i \end{bmatrix} = \boldsymbol{p}_j \ominus \boldsymbol{p}_i, \qquad (1)$$

where $\boldsymbol{R}_i = \boldsymbol{R}_i(\varphi_i)$ is a planar rotation matrix and $\ominus$ the pose compounding operator which has been introduced in [19]. These relative measurements are, in general, affected by noise.

Fig. 1. Pose graph with five vertices connected with five edges. Four of the edges are odometric constraints and one is a loop closing constraint. On the right, the incidence matrix is shown divided into the parts containing the odometric constraints and the loop closing constraints.

Thus, including a zero mean Gaussian noise $\boldsymbol{\epsilon}_{ij} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{P}_{ij})$ leads to the noisy relative measurements

$$\hat{\boldsymbol{\xi}}_{ij} = \boldsymbol{\xi}_{ij} + \boldsymbol{\epsilon}_{ij}. \qquad (2)$$

In general, there are two different types of relative pose measurements: Odometric constraints and loop closing constraints. Here, the first constraints are generated by the wheel odometry of the differential drive robot. The second type of constraints are provided by the robot recognizing a match between actual measurements and past measurements by revisiting places. Subsection II-B shows how to efficiently identify and add loop closing constraints to the pose graph for odometry only data.

The pose graph is thereby represented as a directed graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $N+1$ vertices and $N+M$ edges, where N is the number of odometric constraints and M the number of loop closing constraints. The connection between the vertices by the edges can be compactly written using an incident matrix $\boldsymbol{A}$. There, every column represents an edge connecting two vertices with each other. The row number thereby represents the vertex from which the edge starts, denoted by -1, and the vertex where the edge leads to, denoted by 1. In Figure 1 a pose graph is exemplarily shown in combination with the according incidence matrix $\boldsymbol{A}$.

The overall optimization problem is then to minimize the sum of weighted residual errors $\boldsymbol{r}_{ij}(\boldsymbol{p})$ in regard to the pose estimates $\boldsymbol{p}$

$$\min_{\boldsymbol{p}} \sum_{(i,j) \in \mathcal{E}} ||\boldsymbol{r}_{ij}(\boldsymbol{p})||^2_{\boldsymbol{P}_{ij}} \qquad (3)$$

with

$$||\boldsymbol{r}_{ij}(\boldsymbol{p})||^2_{\boldsymbol{P}_{ij}} = [(\boldsymbol{p}_j \ominus \boldsymbol{p}_i) - \hat{\boldsymbol{\xi}}_{ij}]^\top \boldsymbol{P}_{ij}^{-1}[(\boldsymbol{p}_j \ominus \boldsymbol{p}_i) - \hat{\boldsymbol{\xi}}_{ij}]. \quad (4)$$

The covariance matrix corresponding to the relative measurements $\hat{\boldsymbol{\xi}}_{ij}$ is thereby given as $\boldsymbol{P}_{ij}$.

A. Path Segmentation / Data Pruning

Path segmentation is used in order to cluster the individual path points retrieved from the odometry data into straight line segments. It is mostly used to reduce the complexity of the mapping problem, e.g. [18] or [24]. The reduction

Algorithm 1 DP Generation

- **Parameters:** $L_{\min}$, $e_{\max}$
- **Inputs:** x
- **Outputs:** DP

1: $DP = [\boldsymbol{x}_0]$
2: $S = [\boldsymbol{x}_0]$
3: **if** new $\boldsymbol{x}$ available **then**
4:     $d = ||DP_{\text{end}} - \boldsymbol{x}||$
5:     **if** $d < L_{\min}$ **then**
6:         $S \leftarrow [S, \boldsymbol{x}]$
7:     **else**
8:         $S_{\text{tmp}} = [S, \boldsymbol{x}]$
9:         $e = \text{errorLineFit}(S_{\text{tmp}})$
10:         **if** $e < e_{\max}$ **then**
11:             $S \leftarrow S_{\text{tmp}}$
12:         **else**
13:             $DP \leftarrow [DP, S_{\text{end}}]$
14:             $S \leftarrow [S_{\text{end}}, \boldsymbol{x}]$
15:         **end if**
16:     **end if**
17: **end if**

Fig. 2. To generate a new dominant point, the distance e_i with $j = 2, \ldots, |S| - 1$ is investigated. This distance represents the shortest distance between point S_i and vector $\boldsymbol{v}$. More details are given in Subsection II-A.

of complexity will allow later improvements to the path by standard pose graph optimization techniques. Therefore, as shown in Figure 2, the errors between the individual odometry data points and a straight line segment are calculated and summed up to compare it with a certain threshold $e_{\max}$. If the sum of the errors exceeds the threshold a new line segment with different orientation is generated. The path segmentation scheme is presented in Algorithm (1) and is inspired by [24]. We now shortly state the idea of the path segmentation approach.

Assume the position estimates based on the odometry are given as $X = \{\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n\}$. Here $n+1$ is the number of data points received from the odometry. First, a set of dominant points $DP = \{\boldsymbol{x}_0\}$ and a temporarily subset $S = \{\boldsymbol{x}_0\}$ are initialized using the first position estimate $\boldsymbol{x}_0$. Hereby, the set of dominant points DP represents the pruned data set and the temporarily subset S contains the data points which are approximated by the current straight line segment. We now successively include all odometry data points into the algorithm. Thereby, a new data point $\boldsymbol{x}$ is integrated into the temporary subset $S \leftarrow \{S, \boldsymbol{x}\}$. Afterwards, two conditions are checked:

$$L_{\min} > ||DP_{\text{end}} - S_{\text{end}}|| \qquad (5)$$

and

$$e_{\max} > \frac{1}{|S| - 2} \sum_{i=2}^{|S|-1} e_i, \qquad (6)$$

where DP_{end} and S_{end} are the last added items in the respective set. If both, Equation (5) and Equation (6), are true, the temporary subset S is not any longer a valid representation for the current straight line segment. Thus, the position $S_{\text{end}-1}$ is added to the set DP and the temporary subset S is set

back to $S = \{S_{\text{end}}, \boldsymbol{x}\}$. Here, e_i defines the shortest distance between the point S_i and the vector $v = S_{\text{end}} - S_1$ and $|S|$ is the cardinality of the set S. In Figure 2 the error calculation is exemplarily shown and in Figure 3 a resulting pruned data set is depicted compared to the original data set. The parameters $L_{\min}$ and $e_{\max}$ are problem specific and have to be tuned accordingly.

Based on the pruned data set DP, the poses for the pose graph are generated as

$$\boldsymbol{p} = \{[DP_1^\top, \varphi_1]^\top, \ldots, [DP_{|DP|-1}^\top, \varphi_{|DP|-1}]^\top\}, \qquad (7)$$

where $|DP|$ is the cardinality of the set of dominant points and

$$\varphi_i = \text{atan2}(\boldsymbol{v}_{i,y}, \boldsymbol{v}_{i,x}) \quad \text{with} \quad \boldsymbol{v}_i = DP_{i+1} - DP_i. \qquad (8)$$

For the pose graph $N = |DP| - 2$ and the relative measurements $\hat{\boldsymbol{\xi}}$ can be calculated using Equation (1). In Figure 4 an example for the generation of the pose graph based on the set of dominant points is shown. The part of the corresponding incidence matrix containing the odometric constraints can be then written as

$$A_{\text{Odometry}} = \begin{bmatrix} -1 & 0 & 0 & \ldots \\ 1 & -1 & & \\ 0 & 1 & \ddots & \\ 0 & & \ddots & \\ \vdots & & & \end{bmatrix} \in \mathbb{R}^{N+1,N}. \qquad (9)$$

B. Loop Closure Detection

Only the pose data generated in Subsection II-A can be used to find loop closing constraints. Therefore, it is required for the robot to cycle several times along the boundary line of the area of interest. We then compare the shape of the path, searching for poses with similar shaped neighborhoods in order to find vertices which can be matched onto each other. These matched vertices can then be added as loop closing

Fig. 3. Path Segmentation: In the left panel the original odometry data with 37687 data points is shown. In the right panel the path segmentation with 137 dominant points, marked as red dots, is presented.

Fig. 5. Example for the piecewise linear orientation function $\theta(x)$. The green circled regions show similar path segments. The vertices of the pose graph are pictured as red dots.

constraints to the pose graph.

First, generate a piecewise linear function of the orientation in regard to the length of the path $\theta = \theta(x)$. The function is defined as

$$\theta(x) = \phi_i \quad \text{for} \quad l_{i-1} \leq x < l_i, \quad i = 0, 1, \dots, N. \quad (10)$$

The cumulated orientations ϕ_i and path lengths l_i can be calculated as

$$\begin{aligned} \phi_i &= \phi_{i-1} + \Delta\phi_i \\ l_i &= l_{i-1} + \|\boldsymbol{v}_i\| \end{aligned} \quad (11)$$

starting with $\phi_0 = \varphi_0$ and $l_0 = 0$ and going through all poses of the pruned graph. Here $\boldsymbol{v}_i = \boldsymbol{x}_i - \boldsymbol{x}_{i-1}$ and $\Delta\phi_i = \varphi_i - \varphi_{i-1}$ with $\boldsymbol{p}_i = [\boldsymbol{x}_i^\top, \varphi_i]^\top$. In Figure 5 an example function based on the segmented data from Figure 3 is shown.

Second, a comparison between the shapes of the neighborhood of the poses using the introduced piecewise orientation function $\theta(x)$ is done. Therefore, let L_{NH} be the length to both sides of a pose which defines its neighborhood. A correlation error matrix

Fig. 4. The figure shows how the dominant points are transformed to a set of poses. Thereby, we start by with the first dominant point as the initial pose given by $\boldsymbol{p}_0 = [\boldsymbol{x}_0^\top, \varphi_0]^\top$ with $\boldsymbol{x}_0 = DP_1$ and $\varphi_0 = \text{atan2}(\boldsymbol{v}_{1,y}, \boldsymbol{v}_{1,x})$, $\boldsymbol{v}_1 = DP_2 - DP_1$.

$\boldsymbol{C}$ is then generated as follows:
Comparing the neighborhood of the pose $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$, the orientation function $\theta(x)$ is grounded according to the orientations ϕ_i, ϕ_j and lengths l_i, l_j such that two neighborhood functions $\theta_i(x)$ and $\theta_j(x)$ are generated, e.g. $\theta_i(x) = \theta(x + l_i) - \phi_i$. Both functions are then evaluated at m linearly distributed points from $-L_{\text{NH}}$ to $+L_{\text{NH}}$ which result into two vectors $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$. The correlation error then add up to

$$\boldsymbol{C}_{ij} = \frac{1}{m} \sum_{k=1}^{m} \|\boldsymbol{\theta}_{i,k} - \boldsymbol{\theta}_{j,k}\|^2. \quad (12)$$

In Figure 6 the correlation errors between the vertices are shown.

Third, we search for local minima which are below a threshold c_{min} in the correlation error data. This leads to convenient pairs for loop closure $SP_k = \{\boldsymbol{p}_i, \boldsymbol{p}_j\}$ for $i \neq j$. In general, this procedure leads to a set of loop closing pairs $SP = [SP_1, SP_2, \dots, SP_M]$, where M is the number of loop closing constraints found. The selection of an appropriate value for L_{NH} is hereby crucial to find sufficiently accurate pairs for loop closure. In general, picking a larger value for L_{NH} will lead to a more cautious selection and vice versa. Also, the threshold c_{min} affects the loop closing detection significantly. Thereby, a huge neighborhood parameter L_{NH} and large odometry errors should be compensated by choosing a large value for c_{min} and vise versa.

The loop closing constraints can now be included into the pose graph representation by adding the relative measurements $\hat{\boldsymbol{\xi}}_{ij} = [0, 0, 0]^\top$ between the poses i and j for the loop closing pair $SP_k = \{\boldsymbol{p}_i, \boldsymbol{p}_j\}$. The incident matrix for the loop closing measurements can be built as

$$\boldsymbol{A}_{\text{Loop Closing}} = \begin{bmatrix} \boldsymbol{a}_1 & \dots & \boldsymbol{a}_M \end{bmatrix} \in \mathbb{R}^{N+1,M} \quad (13)$$

with the vector $\boldsymbol{a}_k$ representing the loop closing pair $SP_k = \{\boldsymbol{p}_i, \boldsymbol{p}_j\}$ by denoting $\boldsymbol{a}_{k,i} = -1$ and $\boldsymbol{a}_{k,j} = 1$ and the

Fig. 6. Correlation error of the shapes of the neighborhood between the vertices of the pose graph. For better reading we plotted $\log(1-\boldsymbol{C}_{ij})$ and only a section of the matrix. The variables x_i and x_j are representing the position l of the vertices i and j in meter along the path.

rest of the entries with zeros. The complete incidence matrix for the pose graph is then defined as

$$\boldsymbol{A} = [\boldsymbol{A}_{\text{Odometry}}, \boldsymbol{A}_{\text{Loop Closing}}] \in \mathbb{R}^{N+1,N+M}. \qquad (14)$$

C. Pose Graph Optimization

Let $\boldsymbol{e}_{ij}(\boldsymbol{p}_i, \boldsymbol{p}_j) = (\boldsymbol{p}_j \ominus \boldsymbol{p}_i) - \hat{\boldsymbol{\xi}}_{ij}$ such that Equation (3) can be rewritten as

$$\min_{\boldsymbol{p}} \sum_{(i,j)\in\mathcal{E}} \boldsymbol{e}_{ij}^{\top}\boldsymbol{\Omega}_{ij}\boldsymbol{e}_{ij}, \qquad (15)$$

where $\boldsymbol{\Omega}_{ij}$ is the information matrix for the relative measurement $\hat{\boldsymbol{\xi}}_{ij}$. To solve Equation (15) the information matrices $\boldsymbol{\Omega}_{ij}$ have to be determined with $\boldsymbol{\Omega}_{ij} = \boldsymbol{P}_{ij}^{-1}$. For the odometric constraints we generate the covariance matrices as

$$\boldsymbol{P}_{\text{odometric},ij} = \text{diag}\left(\begin{bmatrix}\cos(\varphi_i)(\alpha_3\delta_T + \alpha_4\delta_R)\\ \sin(\varphi_i)(\alpha_3\delta_T + \alpha_4\delta_R)\\ \alpha_1\delta_R + \alpha_2\delta_T\end{bmatrix}\right) \qquad (16)$$

on the basis of the odometry model presented in [23] and under the assumption that only one translation δ_T and one rotation δ_R occur. The parameters $\alpha_1, \ldots, \alpha_4$ are robot specific and must be determined. For the loop closing constraints, the covariance matrices can be calculated using the correlation error from Equation (6) and the parameters γ_1 and γ_2

$$\boldsymbol{P}_{\text{lc},ij} = \text{diag}\left(\begin{bmatrix}\gamma_1 & \gamma_1 & \gamma_2\end{bmatrix}\right)\boldsymbol{C}_{ij}. \qquad (17)$$

The parameters γ_1 and γ_2 can be used to improve the map estimate in complex environments. In our experiments all parameters were set to one. We then use the popular Levenberg-Marquardt algorithm as presented in [9] in order to solve the pose graph optimization problem from Equation (15).

D. Map Generation and Evaluation

With Subsection II-C we optimized our pose graph, for which an example can be seen in the mid panel of Figure 7. Now, we shortly explain how to generate a closed trajectory from the optimized pose graph which can then be stored as a map of the environment. First, we cut off the prefix and the suffix of the pose graph, because these parts have not been optimized due to missing loop closing constraints. Thereby, the prefix is the part of the pose graph before the first loop closing constraint and the suffix the part of the pose graph after the last loop closing constraint. In order to generate a closed path, we investigate further the generated loop closing pairs. Here, we search for the first loop closing pair which represents a complete turn around the borderline. Setting the points of the this loop closing pair onto each other leads to a closed trajectory as presented in the right panel of Figure 7.

To compare different map estimates with each other we calculate the deviation of area between the estimated map generated using the above algorithms and the true shape of the environment. Therefore, we assume that the true shape of the environment is given as a polygon defined by the points X_{true}. The idea is to determine a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{T}$ which transforms the set of points X, representing the closed map estimate, onto the set of points X_{true}

$$\widehat{X} = \boldsymbol{R} \cdot X + \boldsymbol{T}, \qquad (18)$$

such that

$$\Delta A = 1 - \frac{A_{\text{true}} \cap A_{\text{estimate}}}{A_{\text{true}} \cup A_{\text{estimate}}} \qquad (19)$$

is minimized. Here, ΔA represents the difference between the areas of the map estimate and the true shape of the environment. We use simple gradient descent in order to find a convenient rotation matrix and translation vector for Equation (18). Since gradient descent requires a good initial guess in order to not get stuck into a local minimum, we use Horns method [13] to first find such initial guess. Therefore, we use the in Subsection II-B described method for loop closure detection in order to find pairs of points between the estimated map and the true shape of the environment which can be mapped onto each other. In Figure 8 the deviation between the true shape of the environment and the estimated map is depicted in light blue.

Fig. 7. The figures shows the steps we use for generating a polygon map of the boundary from the odometry data. The left panel shows the original odometry data, the mid panel the optimized pose graph and the right panel the closed polygon.

Fig. 8. Deviation of areas between an original map and the map estimate. The deviation is presented in light blue with $\Delta A = 9\%$.

III. RESULTS

We tested the above proposed mapping method for odometry only data in simulations with challenging environments and on real data. For the real system, a Viking MI 422P, a purchasable autonomous lawn mower, has been used. For the simulation environment and for the estimation of the covariance matrices Equation (16), we used the odometry motion model presented in [23]. We calibrated the odometry model by tracking lawn mower movements using a visual tracking system (OptiTrack) and computed the parameters using maximum likelihood estimation. The calibrated parameters for the Viking MI 422P are presented in Table I. For generating real data, we drove the lawn mower manually along the boundary line of the courtyard depicted in Figure 9(a). Additionally, we simulated a differential drive robot with the specifications of the lawn mower in a challenging apartment environment. Moreover, we successively increased the odometry error in order to show that the proposed approach works even with large odometry errors and leads to sufficiently accurate results. As error measurement we used the deviation of areas between the true shape of the environment and the estimated map, as presented in Subsection II-D. Parameter specifications regarding the robot and the proposed method can be found in the Appendix, Section V.

A. Apartment Environment - Simulation

We used simulations in order to test our algorithm in a challenging environment, namely an apartment floor as depicted in Figure 10. Since the apartment environment is complex we set the threshold $c_{\min}$ to 1.0. Simulating the differential drive robot using the in Table I presented odometry parameters and a simple wall following algorithm results into the estimated path depicted in the left panel of Figure 11. The resulting map estimate can be seen in the

TABLE I
MEASURED PARAMETERS FOR THE ODOMETRY MOTION MODEL ([23]).

α_1	α_2	α_3	α_4
0.0849	0.0412	0.0316	0.0173

TABLE II
MEAN VALUES AND STANDARD DEVIATION FOR DIFFERENT ODOMETRY PARAMETERS.

α	0.1	0.2	0.3	0.4	0.5
$\mu_{\Delta A}$	7.91%	11.71%	14.01%	17.35%	27.30%
$\sigma_{\Delta A}$	1.21%	2.09%	2.95%	3.79%	6.68%

right panel together with the true shape of the apartment. The error between the true shape and estimated map is $\Delta A = 4.5\%$.

To simulate a drastic odometry drift, we set the odometry parameters to $\alpha_i = 0.4$ for $i = 1, \ldots, 4$. This leads to an estimated path of the simulated robot as depicted in the left panel of Figure 12. The results of our map estimate is presented in the right panel of Figure 12. The mapping error is $\Delta A = 17.8\%$.

In order to show the robustness of our approach, we evaluated our method on the apartment environment for different values of α with $\alpha_i = \alpha$ for $i = 1, \ldots, 4$. Therefore, we simulated the robot for every parameter setting $\alpha = \{0.1, 0.2, \ldots, 0.5\}$ ten times and calculated the mean error $\mu_{\Delta A}$ and standard deviation $\sigma_{\Delta A}$, which can be seen in Table II. As expected, the mean $\mu_{\Delta A}$ rises monotonously with increasing α as well as the standard deviation $\sigma_{\Delta A}$.

B. Courtyard Environment - Real Data

For the real data set the parameter $c_{\min}$ is set to 0.3 to take into account the shape of the estimated path generated from the odometry data. This path can be seen in the left panel at Figure 9(b). In the right panel, the map estimate generated using the proposed method is depicted together with the true shape of the courtyard environment. The error between both, according to the method presented in Subsection II-D, is $\Delta A = 11.87\%$. The data for the courtyard has been retrieved from available CAD data.

IV. CONCLUSION

We have presented a method for map estimation based on odometry only data. This method is essential for cheap or small robots, such as vacuum cleaners or lawn mowers. Our method does not require any additional assumptions like for example a rectilinear structured environment [24] or linear features [22], [6]. The required odometry data can be collected using a wall following scheme for which low range sensors or binary sensors are sufficient. Such sensors are widely used in actual purchasable household robots. Furthermore, we showed that our approach performs well in challenging environments, such as an apartment. Even when simulating large odometry noise, our approach can generate an accurate map estimate.

The map estimate of the environment allows the robot to plan its intended task, such as cleaning the floor or mowing the lawn, instead of executing a random walk behavior. In a related work [12], probability distributions were used to update a dirt coverage map to keep the dirt level below a certain threshold. In another work [4], complete coverage is proven under the

(a) The courtyard of our Institute. We used the inner lawn area for testing the proposed mapping method.

(b) The left panel shows the estimated path of the robot generated from its wheel odometry and the right panel the estimated map and the true shape of the test environment.

Fig. 9. The real courtyard depicted in (a) and the collected odometry data together with the map estimate shown in (b).

Fig. 10. The apartment envrionment used for the simulations.

Fig. 11. The left panel shows the estimated path of the simulated robot generated using the introduced odometry model. The right panel shows the estimated map and the true shape of the apartment environment.

Fig. 12. The left panel shows the estimated path of the simulated robot generated using the introduced odometry model and harsher odometry parameter, $\alpha_i = 0.4$ for $i = 1, \ldots, 4$. The right panel shows the estimated map and the true shape of the apartment environment.

TABLE III
DEFAULT PARAMETERS FOR THE PROPOSED MAP ESTIMATION METHOD.

$L_{\min}$	$e_{\max}$	L_{NH}	M	γ_1	γ_2
0.1 m	0.001	30 m	100	1.0	1.0

with a frequency of approximately $20\,\text{Hz}$. The algorithmic parameters used for the map estimation can be found in Table III. Here the chosen parameters $L_{\min} = 0.1\,\text{m}$, $M = 100$ and $e_{\max} = 0.001$ for the path segmentation part reduce the complexity of the problem sufficiently while maintaining the path given by the odometry data. Moreover, the parameter for loop closure detection $L_{\text{NH}} = 30\,\text{m}$, since the used test environments have circumferences of $U_{\text{apartment}} = 100\,\text{m}$ and $U_{\text{courtyard}} = 106.79\,\text{m}$ respectively. Thus, slightly over 50% of the length of the circumference is used for shape comparison.

assumption of a bounded error. According to the first example a probability distribution encoded by a particle filter can be used to update a coverage map. This map allows then the execution of path planning strategies to avoid random walk behavior. This approach seems promising and will be further investigated.

V. APPENDIX

The velocity of the lawn mower driving along the boundary has been set to $0.3\,\text{m\,s}^{-1}$. The odometry data has been sampled

REFERENCES

[1] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the ekf-slam algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562–3568. IEEE, 2006.
[2] Kristopher R Beevers and Wesley H Huang. Slam with sparse sensing. In *ICRA*, pages 2285–2290, 2006.

[3] Christian Brenneke, Oliver Wulf, and Bernardo Wagner. Using 3d laser range data for slam in outdoor environments. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, volume 1, pages 188–193. IEEE, 2003.

[4] Timothy Bretl and Seth Hutchinson. Robust coverage by a mobile robot of a planar workspace. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4582–4587. IEEE, 2013.

[5] Luca Carlone, Rosario Aragues, José A Castellanos, and Basilio Bona. A fast and accurate approximation for planar pose graph optimization. *The International Journal of Robotics Research*, 33(7):965–987, 2014.

[6] Young-Ho Choi, Tae-Kyeong Lee, and Se-Young Oh. A line feature based slam with low grade range sensors using geometric constraints and active exploration for mobile robot. *Autonomous Robots*, 24(1):13–27, 2008.

[7] Javier Civera, Dorian Gálvez-López, Luis Riazuelo, Juan D Tardós, and JMM Montiel. Towards semantic slam using a monocular camera. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1277–1284. IEEE, 2011.

[8] Nikolas Engelhard, Felix Endres, Jürgen Hess, Jürgen Sturm, and Wolfram Burgard. Real-time 3d visual slam with a hand-held rgb-d camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum, Vasteras, Sweden*, volume 180, pages 1–15, 2011.

[9] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[10] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. Nonlinear constraint network optimization for efficient map learning. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):428–439, 2009.

[11] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. Fast and accurate slam with rao–blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38, 2007.

[12] Jürgen Hess, Maximilian Beinhofer, and Wolfram Burgard. A probabilistic approach to high-confidence cleaning guarantees for low-cost cleaning robots. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 5600–5605. IEEE, 2014.

[13] Berthold KP Horn, Hugh M Hilden, and Shahriar Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988.

[14] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[15] Kurt Konolige, Motilal Agrawal, Robert C Bolles, Cregg Cowan, Martin Fischler, and Brian Gerkey. Outdoor mapping and navigation using stereo vision. In *Experimental Robotics*, pages 179–190. Springer, 2008.

[16] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g 2 o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3607–3613. IEEE, 2011.

[17] Yasir Latif, César Cadena, and José Neira. Robust loop closing over time for pose graph slam. *The International Journal of Robotics Research*, 32(14):1611–1626, 2013.

[18] Yasir Latif and José Neira. Go straight, turn right: Pose graph reduction through trajectory segmentation using line segments. In *Mobile Robots (ECMR), 2013 European Conference on*, pages 144–149. IEEE, 2013.

[19] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous robots*, 4(4):333–349, 1997.

[20] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.

[21] Edwin Olson, John Leonard, and Seth Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2262–2269. IEEE, 2006.

[22] Ozan Ozisik and Sirma Yavuz. Simultaneous loclization and mapping with limited sensing using extended kalman filter and hough transfrom. *Tehnicki vjesnik/Technical Gazette*, 23(6), 2016.

[23] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[24] Ying Zhang, Juan Liu, Gabriel Hoffmann, Mark Quilling, Kenneth Payne, Prasanta Bose, and Andrew Zimdars. Real-time indoor mapping for mobile robots with limited sensing. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 636–641. IEEE, 2010.