Real-Time Indoor Mapping for Mobile Robots with Limited Sensing

Ying Zhang, Juan Liu
Palo Alto Research Center
Palo Alto, California 94304
Email: yzhang@parc.com

Mark Quilling, Kenneth Payne
Lockheed Martin Space Systems Center
Sunnyvale, California 94089
Email: mark.l.quilling@lmco.com

Abstract—Mapping and localization for indoor robotic navigation is a well-studied field. However, existing work largely relies on long range perceptive sensors in addition to the robot’s odometry, and little has been done with short range sensors. In this paper, we propose a method for real-time indoor mapping using only short range sensors such as bumpers and/or wall sensors that enable wall-following. The method uses odometry data from the robot’s wall-following trajectory, together with readings from bumpers and wall sensors. The method first performs trace segmentation by fitting line segments to the noisy trajectory. Given the assumption of approximately rectilinear structure in the floor plans, typical for most indoor environments, a probabilistic rectification process is then applied to the segmented traces to obtain the orthogonal wall outlines. Both segmentation and rectification are performed on-line onboard the robot during its navigation through the environment. The resulting map is a set of line segments that represents the wall outline. The method has been tested in office buildings. Experimental results have shown that the method is robust to noisy odometry and non-rectilinear obstacles along the walls.

I. INTRODUCTION

A. Motivation

During the last decade, research on robotic mapping and localization for exploration and navigation in indoor environments has made significant progress. Most existing work has used robots with a comprehensive sensor suite, such as ultrasonic rangers, LIDAR, and cameras, and computers with substantial capabilities. This perceptive sensing capability, together with the robot’s odometry, can be used for simultaneous localization and mapping (SLAM), using probabilistic or constrained optimization approaches [8]. While such approaches have been popular and successful, they typically require sensor-rich robots and powerful computation, which may be infeasible for many low-power robotic systems. In this paper, we address a quite different mapping problem, using only odometry data and short range sensors, and with sufficient algorithmic efficiency to run on low-power embedded processors in real time.

The robot platform for this work is the iRobot Create, a commercially available, cheap testbed. The built-in sensing capability of these robots is rather primitive, consisting of:

- odometry, measuring distance traveled and angle turned;
- a wall sensor on the right side of the robot, sensing proximity to the wall; and
- bumper sensors on the left and right of the front side, sensing contact with obstacles.

Note that wall and bumper sensors are both short-range sensors; the wall sensor has a typical range of under 10 cm. There is low correlation between measurements over even short distances of travel, hence it is challenging to build a map with these “blindfolded” robots.

For this robot platform, we use a wall-following behavior for exploration and navigation to enable the robot to travel around the boundary of a region (see an example floor plan in Fig. 1 (b)). Our algorithms are designed to obtain wall segments for representing an outline of a map. For many applications, such an outline map is useful for identifying corners and intersections which are typically the most critical places in robot placement for surveillance or networking. For a robot exploring by wall-following, it is also important to know if it is loopsing around an internal island, or it has already traversed the whole external boundary. What we will focus in this paper are efficient algorithms to extract an outline map on-board and in real-time, using noisy odometry and limited sensor readings. Loop detection and closure, due to space limitations, will be presented in another paper.

B. Overview

Due to the lack of long-range perceptive sensing, we use odometry to estimate wall locations while executing a wall-following behavior. Due to noisy odometry measurements from quantization and wheel slip, the trajectory estimation error accumulates (see Fig. 2 (a)). The goal is to obtain a map from the raw trajectory data.

To correct noise in odometry data, we take advantage of prior knowledge regarding the navigation environment. In this paper, we assume an indoor environment, mostly consisting of straight walls and rectilinear corners. The straight wall and rectilinear turn assumptions are strong enough to eliminate odometry noise. This can be considered as a rectification process, where raw odometry angles are classified into discrete directions (multiples of \( \pi/2 \)). On the other hand, even in indoor environments we often observe non-rectilinear angles due to obstacles such as small furniture or chairs. Straight-forward rectification would fail in this case. To enable mapping in the presence of obstacles, we have designed a probabilistic
mapping algorithm where direction classification depends not only on the current odometry angle, but also on its past history. This state-based approach adds stability to the mapping algorithm, and can work successfully around obstacles. The probabilistic approach simultaneously maintains up to $N$ hypotheses ($N = 100$ in our experiment), each of which is a candidate map. Hypotheses consistent with the input data are given high probabilities. The algorithm of segmentation and rectification are performed on-line during the wall-following exploration, and the hypothesis with the highest probability is the most probable map, hence chosen as the mapping output.

In our approach, a map is represented as a planar polyline, i.e., a collection of line segments living in a 2-D plane (floor plane). Each line segment is represented as $(l_i, \theta_i)$, where $l_i$ is the length, and $\theta_i$ is its direction. This polyline representation is especially suitable for indoor environments since walls are straight. In other mapping work such as SLAM, occupancy maps are often represented as grids, but the polyline representation can offer many advantages, such as in [10]. For wall-following robot with short range sensors, the polyline representation is more efficient for computation and storage, as well as capable of representing non-discretized distances, hence we use polyline maps here.

A polyline map is a connected sequence of lines. There is a 1-D mapping from the distance $d$ traveled from the starting point of the polyline to the direction of travel $\theta$. Figure 3 shows a polyline map (a) and its traveled distance vs. travel direction graph (b). The horizontal axis is $d$, and the vertical axis is $\theta$. In our indoor mapping setting, $\theta$ is a multiple of $\pi/2$ due to the rectilinear assumption. We introduce the term Accumulated Turn Counts (ATC): let a left turn be 1 and a right turn be -1, ATC is the sum of all turns up to the current distance; $\theta = ATC \cdot \frac{\pi}{2}$. High positive or low negative ATCs indicate loops. ATC is used to (1) find similarity between maps such that exploration maps from two different robots can be merged, and (2) find repetition in the map from a robot, so that looping behavior can be detected on-line. If a loop is identified, loop closure is performed. The resulting map shows a complete enclosed outline of the environment (such as Fig. 2 (b)). The system has been tested extensively in hallway environments in two office buildings.

C. Related Work

Most existing work on SLAM relies on a long ranging device, such as a laser scanner or a camera [5], [7]. Most of these algorithms are based on probabilistic reasoning with data fusion from multiple sources. There has been related work on mapping and localization for low-cost robots in indoor office-
like environments [3], where robot movements are described by rectilinear displacements and an occupancy map is created and updated in a host given trajectories from robots. In our case, the trajectory is recorded while executing the wall-following behavior, from which, the rectilinear segments are computed by the embedded microprocessor using line fitting and probabilistic reasoning in real-time.

Multi-robot landmark-based map building has been studied for many years [1], [9], [11], [2]. In particular, each robot detects local features and a global map is obtained by merging matched features when two robots meet at some point. Whereas LIDAR and cameras have spatially correlated data over a large region, SLAM can effectively detect loop closure [6] using those additional observations or constraints from the environment. In our case, the features of the environments are represented by ATC, obtained from rectilinear wall segments. Common features of the environments can be obtained by computing auto-correlation of a single ATC sequence or correlation between the two ATC sequences. Computation of common features using ATC sequences is simpler and therefore more efficient than that of grid or occupancy maps. While map merging using ATC has been studied as well, due to space limitation, this paper focuses on mapping from data of a single robot wall-following.

D. Contribution and Organization

The contributions of the paper are two-fold: (1) an efficient representation of rectilinear maps, and (2) a method of constructing a rectilinear map using only short-range sensors and odometry data from wall-following.

The rest of the paper is organized as follows. Section II describes wall-follow trajectories as map inputs. Section III presents the trace segmentation and Section IV explains the map rectification algorithm. Section V shows some experimental results. Section VI concludes the paper and discusses future work.

II. WALL-FOLLOWING TRAJECTORIES

We choose to use right-wall-following for exploration of unknown environments. Wall-following is advantageous to use as a motion strategy for indoor environment exploration when odometry is noisy and the range sensor’s capabilities are limited. First, wall-following guarantees the complete exploration of the boundary of the environment without becoming trapped. Second, it enables previously passed locations to be reached without accurate odometry by reverse wall-following, i.e., moving along the wall in the opposite direction. In addition, robots can extract an outline of the environment by using trajectories of wall-following.

However, trajectory noise accumulates. For Creates, distance errors are typically small and angle errors large due to roundoff and accumulator reset-when-read. The resolution of distance measurements is 0.001 m and angle measurements is 1°, or 0.0175 radians. Direct compensation for accumulator reset is possible by reducing the read rate of the sensor, however this introduces other noise sources. Consequently, we aim to correct the error by using assumptions on external environment constraints. In addition to odometry, we use readings from short-range sensors: the left (bl) bumper and the right-side wall sensor (wr). A bumper reading is 1 if the bumper touches an obstacle, and 0 otherwise. If the left bumper (bl) fires (changing from 0 to 1), the robot is likely to make a left turn. If the bumper sensor does not fire, there is no indication regarding the robot’s direction. The measurement wr is 1 if the reading of the wall sensor is larger than a threshold, i.e., the robot is close to an obstacle on its right side.

If the right wall sensor loses contact, the robot is likely to make a right turn. These sensors provide intermittent measurements and are indicative but not conclusive of the robot’s motion direction. The right bumper sensor does not provide too much information for the right-hand wall-following, therefore we choose to ignore it.

III. TRACE SEGMENTATION

Trace segmentation is a process of piecewise line fitting of input trajectories. Rather than using individual points as mapping input, we first cluster points into short linear segments, then use a probabilistic approach to rectify the segments into discrete rectilinear directions. The motivation for trace segmentation is as follows: (1) clustering into short segments drastically reduces computation complexity, since the inference algorithm for rectification works on a small number of segments as opposed to a large number of individual points; (2) for wall-following robots, trajectory points should be following roughly the same direction, and trace segmentation helps to even out measurement noise.

There are various optimization algorithms to do line fitting. We have developed a simple on-line algorithm that produces line segments given input point data over time, i.e.,
The algorithm grows the current line segment as long as:

• the left bumper sensor fired: \( bl_k = 1 \) and \( bl_{k-1} = 0 \): it is likely to be the left turning point, i.e., \( t \) for the new segment will be LEFT.

• the right wall sensor reset: \( wr_k = 0 \) and \( wr_{k-1} = 1 \): it is likely to be the right turning point, i.e., \( t \) for the new segment will be RIGHT.

• detection of turn in odometry: two measurements are compared: (1) the distance from the point to the current segment, and (2) the distance between the current and previous points. In particular, for any new point \((x_k, y_k)\), we compute its distance to the current line segment by \( d_k = |\cos(\theta)(y_k - y_b) - \sin(\theta)(x_k - x_b)| \), where \((x^b, y^b)\) is the beginning point of the current segment, and \(\theta\) is its orientation. The distance from the current point \((x_k, y_k)\) to the previous point is \(d = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2} \).

A turn is detected if a point deviates too much from the current segment, i.e., \( d_k \geq \kappa d \), where \( 0 < \kappa < 1 \) is a constant which is a threshold setting the minimum turning angle between segments.

The pseudo code of trace segmentation is in Table I. In addition to \( \kappa \), parameter \( L \) sets the minimum segment length. We used \( \kappa = 0.1 \) and \( L = 0.7 \) meters in our experiments.

IV. Map Rectification

Trace segmentation produces line segments that fit to the trajectory of wall-following. Due to odometry errors, trajectories are deformed due to measurement noise (Fig. 2 (a)). However, most indoor floor plans have perpendicular wall segments; indoor environments are often rectilinear. Using that assumption, we apply the rectification process to the line segments to make line segments either parallel or perpendicular to each other. The input of the rectification is a sequence of line segments \((l, \theta, t)\), and the output is a new sequence of line segments \((l, s)\) where \( s \) is the rectified angle, with the constraint that two neighboring segments have to be aligned or rectilinear, i.e., \( s_{i+1} - s_i \) can only be multiples of \( \pi/2 \).

One straight-forward idea is to examine the turning angle between two neighboring segments and classify it into discrete angles: 0°, 90°, and so on. A turning angle smaller than a threshold, e.g., 45°, is classified as 0° (going straight); otherwise classified as a rectilinear turn. But this classification algorithm could easily fail. For instance, consider the case of four line segments, with direction \( \theta_i = 0°, 30°, 60°, 90° \), respectively. The turning angle between any two segments is only 30°, hence the straight-forward classification algorithm will classify as 0°. However, it is apparent that the robot has finished a turn. This can be fixed by extending the classification from stateless classification of instantaneous turning angles to a state-based classification with turning angle history. We use a history horizon: for the classification of any segment \( i \), we not only consider the turning angle of segment \( i \), but also the history \( \{i - 1, i - 2, \cdots, i - H\} \) where \( H \) is the horizon length. The horizon length could be defined as physical distance, e.g., any linear segment within a 5-meter distance are included in the horizon, or it could be defined as the number of segments. We use the former in this paper. Segments within the horizon are collectively considered to rectify the current linear segment and hence avoid the pathological case above. Furthermore, it is able to map in the presence of obstacles as long as the obstacles are small in size compared to the history horizon. This is because the segments before and after the obstacle are both used for rectification.

We use an approximate sequential Bayesian filtering method, recursively updating an estimate of the posterior of underlying state \( s_{i+1} \) with observation sequences \( z_{i+1} = \{z_0, z_1, \cdots, z_{i+1}\} \) as:

\[
p(s_{i+1}|z_{i+1}) \propto p_0(z_{i+1}|s_{i+1}) \cdot \int_S p_d(s_{i+1}|s_i) \cdot p(s_i|z_i) \, ds_i.
\]
Equation (1) is sequential: the belief $p(s_{i+1} | z_{i+1})$ is computed from the belief $p(s_i | z_i)$ from the previous time $i$ at every step. The integral makes a single step of prediction bringing the previous state up the the current time and then applying a new external measurement to modify the state. The prediction is then multiplied by a likelihood, reflecting the contribution of new external measurement to modify the state. The prediction horizon, i.e., walls intersect each other at perpendicular angles. Therefore we make the assumption that walls are rectilinear, i.e., walls intersect each other at perpendicular angles. Therefore we have the following model:

\[
\begin{align*}
\begin{cases}
  s_i & \text{if } j = 1, \frac{\pi}{2}, \pi, \\
  s_i + \pi & \text{if } j = \frac{\pi}{2}, \\
  s_i - \pi & \text{if } j = 1, \\
  s_i + \frac{\pi}{2} & \text{if } j = \pi,
\end{cases}
\end{align*}
\]

Although we have used one particular probability distribution in our model, we have noticed that the result shows low sensitivity to different distributions. This model can be extended, e.g. to eight directions instead of four if the environment has a lot of $45^\circ$ angles.

- **Observation model** $p_o(z_i | s_i)$ describes the relationship between the underlying states and measurement observations. Here the measurement from odometry is $z = \{\theta_i, \theta_{i-1} \cdots \theta_{i-H}\}$, i.e., all measurements within the history horizon. We assume the measured turning angle is the true turning angle contaminated with noise, i.e., $\theta_{i+1} - \theta_{i-j} = s_i - s_{i-j} + n_j$, where $j$ indicates the $j$-th previous segment and $n_j$ is noise. We assume a generalized Gaussian distribution (GGD) [4] for noise. GGD is parametrized by a shape parameter $r$ and a standard deviation $\sigma$. GGD includes Gaussian distributions as a special case, when $r = 2$, is more flexible because the shape parameter $r$ can be varied to describe noise with heavy tails (when $r < 2$). This is necessary for modeling angle noise because the odometry data are quite noisy, especially when the line segment length is short. Hence we choose GGD for its increased flexibility. In our test, we use a shape parameter $r = 0.5$ and a standard deviation of $10^\circ$ for $n_1$ and $(H - 1) \cdot 20^\circ$ for $n_j$ where $1 < j \leq H$. The parameters are chosen by experimental data from real trajectories that result in best performance. Another observation available is the turn information $t = \{\text{STRIGHT,LEFT,RIGHT}\}$, obtained from trace segmentation. For this observation, we use a simple likelihood function: $p(t | s_i) = 1$ if the turn state in the hypothesis agrees with the turn observation $t$, and $p(t | s_i) = 0.5$ if the two do not agree. This discounts hypotheses which conflict with sensor observations. The observation likelihood is multiplied to that of the odometry data, assuming that the different sensing modalities are independent.

The rectification process takes the line segment inputs over time and maintains up to $N$ (100 in experiment) most probable state sequences and their respective probabilities $p_n$. Whenever a new segment $(l, \theta, t)$ comes, each existing state sequence $(s_0, s_1, \cdots s_i)$ forks into four possibilities, with the new traveling direction $s_{i+1}$ being straight, a left or right, or U-turn. Probabilities of the new state sequences are updated using the dynamics model and observation model as illustrated above. State sequences with low probabilities ($p_n < 1/N$) are discarded, and up to $N$ most probable state sequences are kept, their probabilities re-normalized to sum up to 1. That brings us to the next update. The final mapping result is the most probable state sequence.

This method does not only build the map but also localize the robot within the map, where the location of the robot is represented by the distance traveled along the outline map from its start position. This method is one-dimensional SLAM since both the map and the location is represented by one degree of freedom.

V. EXPERIMENTS AND RESULTS

Our map construction algorithms have been implemented both in Matlab and in C++ on the Gumstix, a PXA270 600 MHz computer. We have tested the system using a Create doing wall-following in hallways in two office buildings. The system works reliably even with noisy odometry and with obstacles and/or furniture along the walls.

Fig. 1 and Fig. 2 illustrate the capability of our mapping system to create an outline map of an enclosed environment by following the hallway wall along its right side. Fig. 4 demonstrates an outline mapping of an island with non-rectilinear edges. Fig. 4 (a) is the floor plan of the area the robot travels around (dashed line indicating the path of the robot), with an open lounge at a corner. Fig. 4 (b) shows the scene in the lounge with three garbage cans (one of which is a cylinder) and the kitchen counter with $45^\circ$ angles. Fig. 4 (c) displays raw trajectory data from odometry, which is clearly deformed due to noise and can be hardly recognized as a closed loop of the trajectory. Fig. 5 shows the result of mapping. We see that although the raw trajectory from odometry is very noisy, our algorithm successfully creates an outline of the area. From the ATC plot, we can clearly identify the matching points, that indicating a loop.

So far our experiments are mostly done in hallways of office buildings where most segments follow the rectilinear assumptions. The system will not work well if most segments do not satisfy this assumption. Occasional obstacles along the walls can be handled, as we have demonstrated. Too many obstacles may introduce error in rectification that may cause the failure of loop detection by proximity measurements.

VI. CONCLUSIONS AND FUTURE WORK

We have presented a real-time floor map construction method for cheap and small robots without long-range perceptive sensing, using the wall-following motion strategy. We have shown by experiments using iRobot Creates that the
method is robust to odometry noise and occasional obstacles along the walls. Such a method can be very useful for indoor robotic applications with limited sensing. The advantages of such a method are that it is extremely efficient, and the information extracted from the raw trajectories and short-range sensors is useful for many applications in surveillance and networking. Note that our probabilistic mapping does not use particle filters as most other mapping algorithms do. Rather, we choose to update probability for each possible case at every cycle. Particle filters are much less efficient since it needs a lot of particles (> 100) to keep low probability cases in consideration.

We have developed algorithms for map merging, loop detection and loop closure. Due to space limitations, we will present them in other papers. We will also investigate the integration of other information from the environment, such as radio signal strengths from stationary known or unknown locations, for map merging and loop detection. Furthermore, we will extend our methods to non-rectilinear environments, e.g. with circular curves or diagonal edges. We will develop map-building capability with navigation and exploration strategies in unknown environments, and multi-robot coordination for efficient sensing coverage and networking.

ACKNOWLEDGMENT

This work was sponsored by the Defense Advanced Project Research Agency (DARPA) contract #FA8650-08-C-7814, LANdroids.

REFERENCES