Hindawi Publishing Corporation
Journal of Robotics
Volume 2011, Article ID 257852, 12 pages
doi:10.1155/2011/257852

Research Article

A Light-and-Fast SLAM Algorithm for Robots in Indoor Environments Using Line Segment Map

Bor-Woei Kuo,[1] Hsun-Hao Chang,[1] Yung-Chang Chen,[2] and Shi-Yu Huang[3]

[1] *Department of Electrical Engineering, National Tsing-Hua University 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan*
[2] *Department of Electrical Engineering, National Tsing-Hua University 720R, EECS Bldg, 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan*
[3] *Department of Electrical Engineering, National Tsing-Hua University 818, EECS Bldg, 101, Section 2, Kuang-Fu Road, Hsinchu 30013, Taiwan*

Correspondence should be addressed to Bor-Woei Kuo, bwkuo@larc.ee.nthu.edu.tw

Received 19 January 2011; Accepted 15 March 2011

Academic Editor: Heinz Wörn

Simultaneous Localization and Mapping (SLAM) is an important technique for robotic system navigation. Due to the high complexity of the algorithm, SLAM usually needs long computational time or large amount of memory to achieve accurate results. In this paper, we present a lightweight Rao-Blackwellized particle filter- (RBPF-) based SLAM algorithm for indoor environments, which uses line segments extracted from the laser range finder as the fundamental map structure so as to reduce the memory usage. Since most major structures of indoor environments are usually orthogonal to each other, we can also efficiently increase the accuracy and reduce the complexity of our algorithm by exploiting this orthogonal property of line segments, that is, we treat line segments that are parallel or perpendicular to each other in a special way when calculating the importance weight of each particle. Experimental results shows that our work is capable of drawing maps in complex indoor environments, needing only very low amount of memory and much less computational time as compared to other grid map-based RBPF SLAM algorithms.

1. Introduction

In recent years, Simultaneous Localization and Mapping (SLAM) has become one of the basic requirements for robotic navigation. This technique allows robots to have the ability to simultaneously build up a map and localize itself in an unknown environment. The main issue involved is that while localizing a robot we need an accurate map, and for updating the map the robot needs to estimate its location accurately. The relation between robot localization and map updating is usually described as a highly complex chicken-and-egg problem.

Clearly, lightweight SLAM algorithms are needed in intelligent robotic systems, because mobile robots are sometimes limited by its size and power budget, so it is usually equipped with a microprocessor which has lower capability than ordinary PCs. For instance, robotic vacuum cleaner has been widely used in many indoor housekeeping applications, but most of these robotic cleaners just randomly wander around the environment and cleans up the floor along its path, which is a very inefficient way to clean up a room. After integrating SLAM into the robotic system [1, 2], we can develop an intelligent robotic vacuum cleaner, which is capable of planning its path in a more efficient way by using the map and location information. However, as mentioned in [3], price, size, and accuracy of the sensor is not the major problem, due to the fast progress in sensing technology. The biggest challenge will be how to develop a robust lightweight algorithm and how to integrate it into these embedded robotic systems.

Rao-Blackwellized particle filter (RBPF), which was first introduced by Murphy and his colleagues [4, 5], has become one of the most successful ways of solving the SLAM problem [5–10]. Montemerlo et al. extended this ideal and proposed the FASTSLAM [7] algorithm which uses a Rao-Blackwellized particle filter to estimate the robot's poses and tracks the location of landmarks by using an extended Kalman filter (EKF). Another efficient approach to solve the SLAM problem using RBPF is the DP-SLAM [8, 11], which uses grid maps rather than landmark maps so that it does not need any assumption on landmarks. The algorithm assigns each so-called *particle* (i.e., a snapshot of the environment the robot recognizes in terms of a local map and the location and pose of the robot in the map) an individual map and maintains these maps by sharing the same parts of the map between particles using a technique called distributed particle mapping, which efficiently reduce the memory needed for grid map-based RPBF SLAM. Recently [9] proposed a hybrid approach which uses the grid map as the main map structure and enhance the grid map by a set of line segment maps; as a result, this hybrid approach is able to increase the accuracy of the algorithm.

However, the techniques mentioned above needs large amount of memory and high computational resource to achieve good results. This is due to the fact that most of the RBPF-based SLAM needs to maintain a large number of particles where each particle has its own map. The most commonly used map structure in RBPF SLAM is still the grid map, because it is capable of describing objects of random shapes and can easily calculate the importance weight of each particle by using scan matching methods [12, 13], but there is a drawback. Since each grid map is built in a fixed-sized 2D array where each cell represents a specific coordinate location, it requires extremely large amount of memory for storing these grid maps.

In this work, we aim to develop a lightweight laser range finder-based SLAM algorithm for indoor environments using Rao-Blackwellized particle filter (RBPF), while most of the major structures (walls, doors, etc.) and furniture of indoor environments can be easily represented as line segments, we decided to use line segments as our map structure instead of grid map. Also when storing line segments into the map, we only need to store the location and parameters of each line segment. As a result, the memory requirement is much more efficient as compared to grid map.

On the other hand, we also use the orthogonal property of indoor environments to increase accuracy of our algorithm. This property is based on the fact that most structures of indoor environments are parallel or perpendicular to each other [3]. By just considering these orthogonal lines, we can filter out most of the erroneous line segments caused by sensor noise or line extraction error. Although this concept has already been used in many other works [3, 14, 15], there is still a main differentiating factor in our work—we dynamically modify the reference direction, which is used to identify the orthogonal lines, rather than just aligning the lines with the x-axis and y-axis. This is extremely important since it can thereby allow the robot to have the ability to start its initial pose at any angle and need not to align its initial direction with the major structures of the environment.

The rest of this paper is organized as follows. Section 2 quickly describes how the Rao-Blackwellized particle filter solves the SLAM problem. In Section 3, we will discuss the details of our lightweight RBPF SLAM algorithm. Section 4 presents the experimental results, and Section 5 concludes.

2. RBPF for SLAM

As mentioned in [16], a full SLAM problem is to estimate the *joint posterior* by which a robot recognizes the environment after moving around for a while. This *joint posterior* is denoted as $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$, where $x_{1:t}$ denotes the entire robot trajectory estimated so far from time instant 1 to time instant t, m is the associated map in terms of some data structure (e.g., grid map or line segment based map in our system). The joint posterior is constantly derived and updated based on two given pieces of information—the observations $z_{1:t}$ from the environment-measuring sensor (which is a laser range finder in our system), and the odometry measurements $u_{1:t-1}$, which is a rough estimation of the trail a robot have traveled up to some point in time. Since odometry is based on how many rounds each of the robot's two wheels has turned, it often has significant errors referred to as odometry noise. A robust SLAM algorithm should be able to derive accurate joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ under significant observation inaccuracy and/or odometry noise.

The key idea of the Rao-Blackwellized particle filter based algorithm is to solve the SLAM problem by splitting the joint posterior $p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1})$ into two separate parts [4, 6]. The first part is about estimating the *robot trajectory*, that is, $x_{1:t}$, using a particle filter (to be explained in detail later). The second part is about the update of the map based on the derived trajectory. With this concept, we can rewrite the formula in two cascaded parts as

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) = p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} z_{1:t}, u_{1:t-1}). \tag{1}$$

While using a particle filter to estimate the robot's potential trajectories, a *motion model* is needed regarding the generation of the new particles $\{x_t^{(i)}\}$ from the corresponding previous particle $\{x_{t-1}^{(i)}\}$ by taking into account the odometry data. Due to the enormous error of the odometer, we need to determine how accurate each new particle is, that is, we assign each of them an *importance weight* according to how well the current measurement of the environment matches with the map. After assigning each particle an importance weight, the particles that have lower weight will be discarded, until only a small number of particles are left. Finally, the map is updated for each remaining particles using the current measurements from the sensor. By repeatedly executing the steps mentioned above, we can maintain a set of particles at any given time, indicating the mostly likely robot's poses and a map built by its previous trajectory. By doing so, the robot is capable of moving around in an unknown environment without getting lost.

3. Proposed Lightweight RBPF SLAM

3.1. System Overview. Figure 1 shows the overall flowchart of our lightweight RBPF SLAM algorithm. When a robot starts executing SLAM in an unknown environment, we will first set the robot's initial pose at the origin of its coordinate system as $(x, y, \theta) = (0, 0, 0)$, and build an initial map using the information from the laser range finder according to

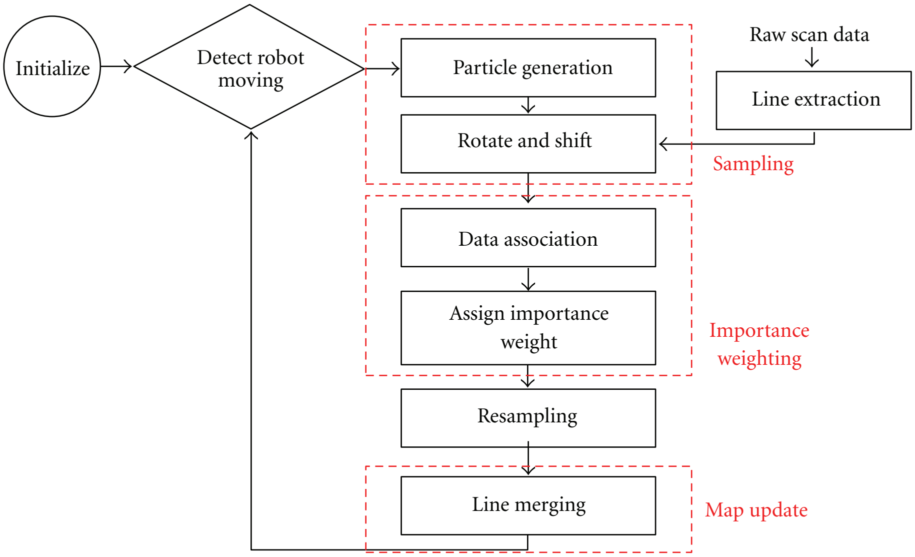


Figure 1: Overall flowchart of a particle filter-based algorithm.

the initial pose. After initializing the system, it will repeatedly estimates the correct robot pose using a particle filter and update the map of each particle by a line merging technique each time when the robot moves farther than a distance.

Usually an RBPF-based SLAM algorithm can be implemented by using the *Sampling Importance Resampling* (SIR) filter, which is one of the most commonly used particle filtering algorithms [6], and a map update technique. The entire procedure can be summarized in the following steps,

(1) *Sampling*: New particles are generated from the previous particle using a motion model.

(2) *Importance Weighting*: Each new particle is assigned an importance weight to determine the accuracy of the particle according to how well the current observation matches the map it has already built.

(3) *Resampling*: Particles with low weights are likely to be replaced by the ones with high weights.

(4) *Map Update*: The most current map observed by the laser range finder is updated to each remaining particle after the resampling step according to its individual pose, so that each particle has a most updated map of the environment.

Because of using line segments as our map structure, we need some extra procedure to maintain these lines. First of all, we need to extract the line segments from the raw scan data provided by a laser range finder; this will allow us to have the information about the environment the robot currently sees. But extracting the line segments for every new particle generated in the particle generation step will cost an enormous effort, so our algorithm only extracts a set of reference line segments, then by rotating and shifting this set of line segments to the corresponding new particles pose. In the importance weighting step, we need to find out the relation between the new extracted line segments and the map we have already built before calculating the importance weight of each particle; this procedure is called *data association*. Also in the data association step, we mainly consider the orthogonal lines [3] extracted from the laser range finder, to filter out erroneous lines caused by sensor noise or line extraction error. Finally, in the map-update step, line segments which are too close to each other are merged together to maintain consistency of our map and lower the number of line segments. After updating the map for each particle, we will wait until the robot has moved farther than a distance before starting the new iteration over again.

In the following subsections, we will discuss the details of each step mentioned in Figure 1 to implement a lightweight RBPF SLAM for indoor environments.

3.2. Particle Generation. This is the first step of the particle filtering process, where a motion model is used to generate new potential robot poses according to its previous pose based on the odometry data. These potential robot poses are usually called particles. The motion model we used here can be found in [17], where the authors proposed a motion model that is capable of capturing the relationship between odometry data and the change of robot configuration, then by using this information to modify its parameters and increase the accuracy of the model. The motion model can be written as

$$\begin{aligned} x_t &= x_{t-1} + D\cos\left(\theta_{t-1} + \frac{T}{2}\right) + C\cos\left(\theta_{t-1} + \frac{T+\pi}{2}\right), \\ y_t &= y_{t-1} + D\sin\left(\theta_{t-1} + \frac{T}{2}\right) + C\sin\left(\theta_{t-1} + \frac{T+\pi}{2}\right), \\ \theta_t &= \theta_{t-1} + T \bmod 2\pi, \end{aligned} \tag{2}$$

where D, T, and C are three noise models which represent the robots *travel distance*, *amount of turns performed*, and the *shift* in the orthogonal direction to the major axis, respectively.

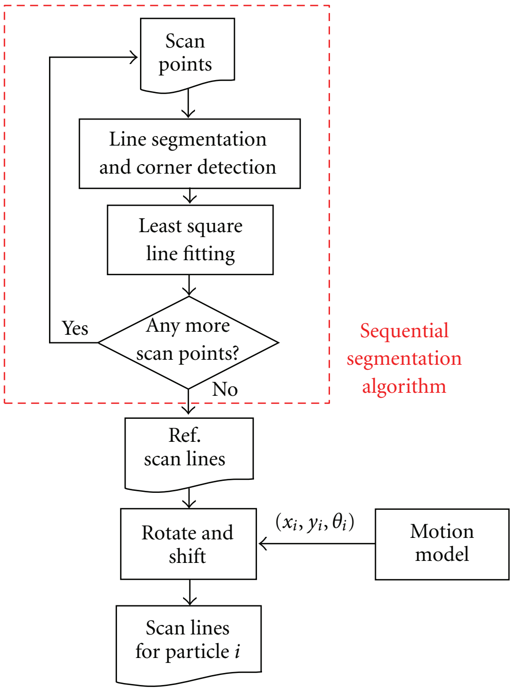

Scan points
Line segmentation and corner detection
Least square line fitting
Yes
Any more scan points?
No
Sequential segmentation algorithm
Ref. scan lines
Rotate and shift
(x_i, y_i, θ_i)
Motion model
Scan lines for particle i

Figure 2: Line extraction procedure.

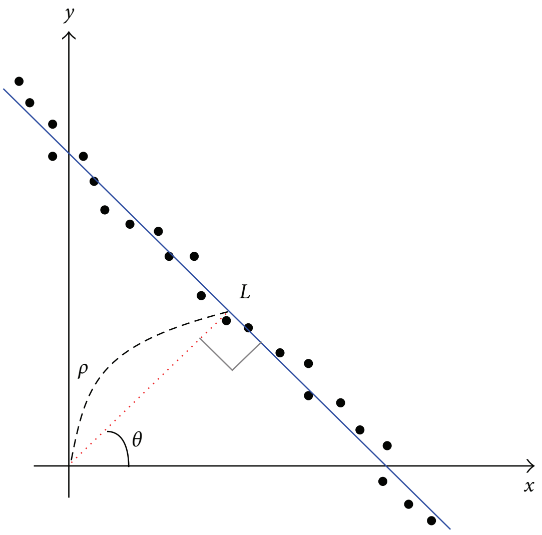


Figure 3: The relationship of line L in two different spaces, where the black dots are the points that contains line L.

After the particle generation step, we will have a set of new particles $\{x_i\}_{i=1,\ldots,m}$ generated by the motion model, each representing a potential pose of the robot (x, y, θ) and an updated map considering its previous trajectory. Due to the enormous error of the odometry data, we cannot rely only on the motion model to estimate the true robot pose. So, we need the help of a laser range finder, which is significantly more precise, to provide the information of the current environment and matches it with the map we have already built to filter out particles that are too much inconsistent with the true pose, which will be introduced in the following sections.

3.3. Line Segment Extraction. The accuracy of a feature-based SLAM algorithm can be increased by using a robust *feature extraction method*; this is because feature-based SLAM heavily depends on the features extracted from the sensor to estimate the robot pose. Figure 2 shows the procedure of our line extraction algorithm, where an *enhanced sequential segmentation* algorithm is used to extract a set of reference line segments from raw scan data provided by a laser range finder, and, then, we rotate and shift the reference scan lines to the corresponding particle pose (x_i, y_i, θ_i) generated by the motion model, so that each particle has its own set of scan lines.

In this section, we will describe how to enhance the sequential segmentation algorithm proposed by [18], and creating scan lines for each new particle by rotating and shifting the reference scan lines.

3.3.1. Least Square Fitting. Least square fitting is a technique that finds the best fitting line $L: x \cdot \cos\theta + y \cdot \sin\theta = \rho$ to a given set of sample points $\{(x_i, y_i)\}_{i=1,\ldots,n}$, which minimizes the following error:

$$E_{\text{fit}} = \sum_{i=1}^{n} (x_i \cos\theta + y_i \sin\theta) - \rho, \qquad (3)$$

θ and ρ can be computed by using a least square method [19] as follows:

$$\theta = \frac{1}{2} \operatorname{atan2} \frac{-2\left(S_{xy} - N X_N Y_N\right)}{\left(S_{yy} - S_{xx}\right) - \left(Y_N^2 - X_N^2\right)}, \qquad \rho = X_N \cos\theta + Y_N \sin\theta, \qquad (4)$$

where

$$S_{xx} = \sum_{i=1}^{N} x_i^2, \qquad S_{yy} = \sum_{i=1}^{N} y_i^2, \qquad S_{xy} = \sum_{i=1}^{N} x_i y_i, \qquad X_N = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad Y_N = \frac{1}{N}\sum_{i=1}^{N} y_i. \qquad (5)$$

The relation of line $L: x \cdot \cos\theta + y \cdot \sin\theta = \rho$ between Hough space and x- y plane are shown in Figure 3.

3.3.2. Enhanced Sequential Segmentation Algorithm. The sequential segmentation line extraction algorithm was first introduced in [18], which is known as a very efficient method for line extraction [18, 20]. The main difference between our work and [18] is that we enhanced the algorithm by adding a *corner detection* technique and an *adaptive breakpoint detector* for line segmentation. This will lead to better quality in complex indoor environments.

Sequential segmentation algorithm works by first giving a set of raw scan data, denoted as $\{d_i\}_{i=1,\ldots,N}$. If the distance between three consecutive points $\{d_k, \ldots, d_{k+2}\}_{1 \leqq k \leqq N-2}$ are within a threshold δ_d, which is $\|d_{k+1} - d_k\| \leqq \delta_d$ and $\|d_{k+2} - d_{k+1}\| \leqq \delta_d$, then a line L is extracted from the points $\{d_k, \ldots, d_{k+2}\}$ using least square fitting [19]. After finding

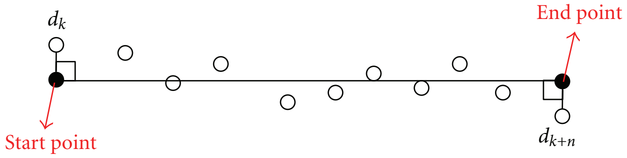


Figure 4: Projecting the start/end points d_k and d_{k+n} onto the extracted line.

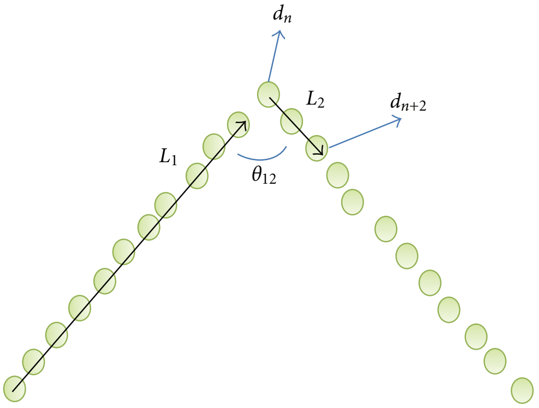


Figure 5: An example of corner detection.

the first three consecutive points that can be extracted into a line segment L, we will determine whether the next consecutive point d_{k+3} can be merged into line L by the following steps:

(a) $\|d_{k+3} - d_{k+2}\| \leqq \delta_d$,

(b) the perpendicular distance from scan point d_{k+3} to line L is within a threshold.

If the two conditions all achieves, we will merge the scan point d_{k+3} into line L by using a sequential least square method proposed in [18], and the next point d_{k+4} is tested. Otherwise, a new scan line S_i is determined from L, and the same process is repeated for extracting new scan lines starting from d_{k+3} and ends until all the scan points have been tested. When a new scan line S_i is extracted from a set of scan points $\{d_k, d_{k+1}, \ldots, d_{k+n}\}$, the two terminal points of the new scan line are determined by projecting the first point d_k (start point) and the last point d_{k+n} (end point) onto the scan line as shown in Figure 4.

The enhanced sequential segmentation algorithm takes place in two parts in the original algorithm. First, we determine the threshold value δ_d, which decides if two consecutive points are close enough to be in the same line segment, by using an adaptive breakpoint detector introduced in [21] rather than a constant value. This is due to the fact that the distance between two consecutive points will increase while the distance between the laser range finder and the scan point increases. So, δ_d can be defined as

$$\delta_d = r_{k+1} \cdot \frac{\sin \Delta\varphi}{\sin(\lambda - \Delta\varphi)} + 3\sigma_r, \qquad (6)$$

where $\Delta\varphi$ is the angular resolution of a laser range finder, r_{k+1} is the distance between laser range finder and scan point d_{k+1}, λ is a constant parameter, and σ_r is the residual variance. As a result, by dynamically changing the threshold δ_d can let our line extraction algorithm achieve a better quality in detecting break points when two consecutive scan points are very close to or very far away from the laser range finder.

Second, a corner detection technique is performed while adding new consecutive scan point d_n into the line L, so now we not only detect if the distance $\|d_n - d_{n-1}\|$ and the perpendicular distance from d_n to L are under a threshold, but also detect if d_n is on the corner of two line segments. The corner detection procedure can be summarized by the following steps, which is also shown in Figure 5.

(a) Detect if the scan point d_n and its next two consecutive points $\{d_{n+1}, d_{n+2}\}$ can be merged into a line L_2, which is positive when the distance from d_n to d_{n+1} and d_{n+1} to d_{n+2} are both under a threshold δ_d.

(b) Detect if the perpendicular distance from d_{n+2} to L_1 is larger than a threshold.

(c) If one of the two conditions mentioned above does not achieve, the scan point will not be at the corner of two line segments. Otherwise, we will calculate the angle θ_{12} between the lines we are currently extracting, that is, L_1, and L_2, which are extracted from the three new scan points $\{d_n, d_{n+1}, d_{n+2}\}$. If θ_{12} is smaller than a threshold, a corner is detected.

After the enhanced sequential segmentation has completed, we will have a set of scan lines $\{s_i\}_{i=1,\ldots,N}$ extracted from the raw scan data, and each scan line is defined as.

$$S_i = \left\{\left(\mathrm{sp}_i,\ \mathrm{ep}_i\right), N, S_{xx}, S_{yy}, S_{xy}, X_N, Y_N\right\}, \qquad (7)$$

where $(\mathrm{sp}_i,\ \mathrm{ep}_i)$ are the start point and end point of the scan line and N, S_{xx}, S_{yy}, S_{xy}, X_N, Y_N are parameters generated in the least square fitting step, which are used to speed up the algorithm while updating the scan lines and merging two line segments.

3.3.3. Rotate and Shift. Since each new particle generated by the motion model has its own pose (x_i, y_i, θ_i), each particle needs a set of line segments representing the information about the current environment according to its individual pose. However, due to the fact that all particles extract line segments by projecting from the same set of raw scan data, performing line extraction for each particle is a very inefficient way since it has a lot of repeated and redundant computation. In our work, we speed up the process tremendously by exploiting a property—there exists shifting and rotating relations between scan lines derived for the particles generated by the motion model, that is, once we have derived the scan lines of one particle, we can use them as the reference scan lines, and map these reference scan lines into those for some other particle via shifting and rotation transforms. In detail, we pick up a reference particle first and generate a set of reference scan lines from the raw scan data for this particle. Then for every other particle, we rotate and shift the reference scan lines according to the particle's relative location to the reference particle.

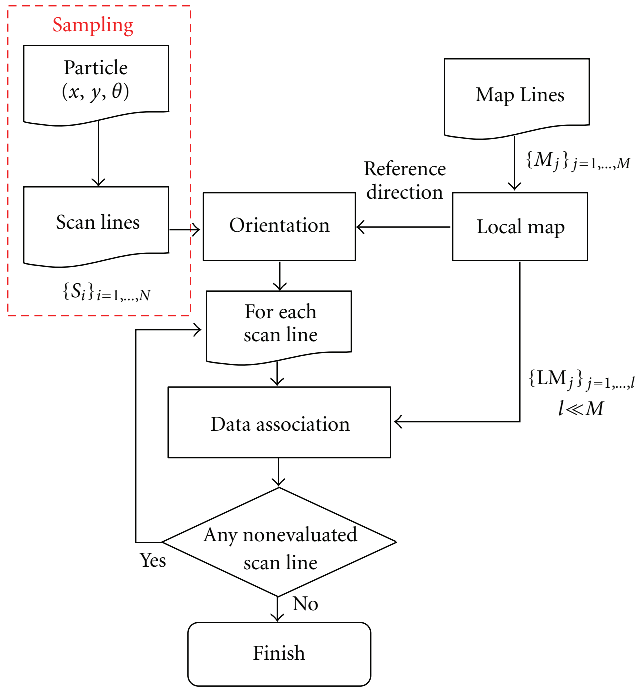


Figure 6: Overall process of data association.

The way of rotating and shifting the reference scan lines is inspired by the sequence least square fitting algorithm [18], where we do not need to recalculate all the parameters using least square fitting for each particle, but only updates the six parameters of a line segment in a sequential manner as follows.

(a) *Rotation*

$$\begin{aligned} S^{xx}_{\mathrm{rot}} &= S^{xx}_{\mathrm{ref}}\cos^2\theta_i - 2S^{xy}_{\mathrm{ref}}\sin\theta_i\cos\theta_i + S^{yy}_{\mathrm{ref}}\sin^2\theta_i, \\ S^{yy}_{\mathrm{rot}} &= S^{xx}_{\mathrm{ref}}\sin^2\theta_i + 2S^{xy}_{\mathrm{ref}}\sin\theta_i\cos\theta_i + S^{yy}_{\mathrm{ref}}\cos^2\theta_i, \\ S^{xy}_{\mathrm{rot}} &= \left(S^{xx}_{\mathrm{ref}} - S^{yy}_{\mathrm{ref}}\right)\sin\theta_i\cos\theta_i + S^{xy}_{\mathrm{ref}}\left(\cos^2\theta_i - \sin^2\theta_i\right), \\ X^n_{\mathrm{rot}} &= \cos\theta_i \cdot X^n_{\mathrm{ref}} - \sin\theta_i \cdot Y^n_{\mathrm{ref}}, \\ Y^n_{\mathrm{rot}} &= \sin\theta_i \cdot X^n_{\mathrm{ref}} + \cos\theta_i \cdot Y^n_{\mathrm{ref}}. \end{aligned} \tag{8}$$

(b) *Shift*

$$\begin{aligned} S^{xx}_{\mathrm{shift}} &= S^{xx}_{\mathrm{rot}} + 2N \cdot x_i \cdot X^n_{\mathrm{rot}} + N \cdot x_i^2, \\ S^{yy}_{\mathrm{shift}} &= S^{yy}_{\mathrm{rot}} + 2N \cdot y_i \cdot Y^n_{\mathrm{rot}} + N \cdot y_i^2, \\ S^{xy}_{\mathrm{shift}} &= S^{xy}_{\mathrm{rot}} + N \cdot y_i \cdot X^n_{\mathrm{rot}} + N \cdot x_i \cdot Y^n_{\mathrm{rot}} + N \cdot x_i \cdot y_i, \\ X^n_{\mathrm{shift}} &= X^n_{\mathrm{rot}} + x_i, \\ Y^n_{\mathrm{shift}} &= Y^n_{\mathrm{rot}} + y_i, \end{aligned} \tag{9}$$

where the start/end points of each line can also be computed by rotating and shifting from the reference scan lines by

$$\begin{aligned} x &= \left(x_{\mathrm{ref}} \cdot \cos\theta_i - y_{\mathrm{ref}} \cdot \sin\theta_i\right) + x_i, \\ y &= \left(y_{\mathrm{ref}} \cdot \cos\theta_i + x_{\mathrm{ref}} \cdot \sin\theta_i\right) + y_i. \end{aligned} \tag{10}$$

3.4. *Data Association.*

Data association is to find the relationship between the new extracted scan lines and the map lines which we have already built in the map for each particle, and by using this relationship to calculate an importance weight of each particle to indicate the correctness of its pose. As we can see, the accuracy of data association is a crucial issue to achieve a robust SLAM algorithm.

As shown in Figure 6, before entering the data association step, a local map is created to reduce search space while matching scan lines with the map; this is due to the fact that the motion model will generate a large amount of new particles (i.e., 500 particles per iteration), where each particle has its own set of scan lines and each scan line needs to be matched with the entire map. Also, the orthogonal assumption for indoor environments [3] is performed in the orientation step, where we only consider scan lines which are parallel or perpendicular to each other. This is because most major structures of indoor environments usually have the orthogonal relationship, and by using this assumption we can reduce the number of scan lines and filter out erroneous scan lines caused by sensor noise or line extraction error to increase the accuracy of the algorithm. The details of building a local map, the orientation step, and data association are described below:

3.4.1. *Local Map.*

The size of a local map is determined by a few meters larger than the largest area that can cover all the scan lines of the new particles generated by the same previous particle, which we called parent particle. All the new particles generated from the same parent particle shares a local map, because the maps they contain are built from the same previous robot trajectory. This can let us reduce the number of times to build local maps, and efficiently reduce the search space for finding the best match of each scan line in the data association step.

Also a reference direction θ_{ref} is calculated in order to let the orientation step determine which scan line extracted from the laser range finder are orthogonal lines. This is done by first determining the most observed line in the local map as a reference line, which is the line that has been observed most of the time. If there is more than one most observed line, the longest one is picked. Then, by using all the lines that are orthogonal to the reference line in the local map to compute a weighted reference direction similar to the method proposed in [3] by the following formula:

$$\theta_{\mathrm{ref}} = \frac{\sum_{i=1}^{N_{||}} w_i\theta_i + \sum_{j=1}^{N_{\perp}} w_j\left(\theta_j - \pi/2\right)}{\sum_{i=1}^{N_{||}} w_i + \sum_{j=1}^{N_{\perp}} w_j}, \tag{11}$$

where $N_{||}$ and $N_{\perp}$ are the number of lines that are parallel and perpendicular to the reference line, respectively, w_i is the length of line i, and θ_i is the angular parameter of line i. Our

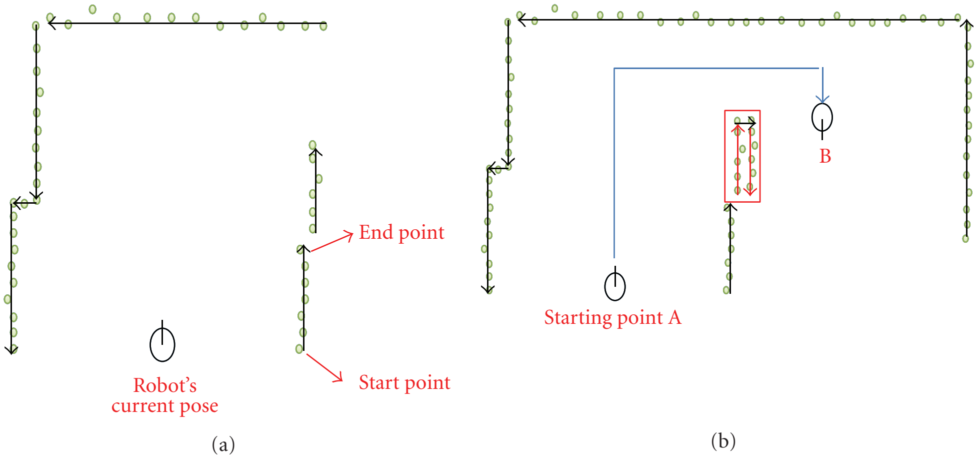


Figure 7: (a) Vector lines determined by its start/end points. (b) Example of vector lines representing opposite sides of a wall.

work dynamically modifies the reference direction each time the local map is built rather than just aligning the orthogonal lines with the x axis, so that the robot is capable of starting its initial pose without aligning with the major structures of the environments and can still correctly detect the orthogonal lines.

3.4.2. Orientation. The orientation step uses the reference direction θ_{ref} to identify whether the scan line is an orthogonal line by the following equation:

$$\begin{gathered} \left|\theta_i - \theta_{\text{ref}}\right| \leq \varepsilon_\theta, \quad \text{parallel to } \theta_{\text{ref}}, \\ \left|\theta_i - \theta_{\text{ref}} - \frac{\pi}{2}\right| \leq \varepsilon_\theta, \quad \text{perpendicular to } \theta_{\text{ref}}, \end{gathered} \tag{12}$$

where ε_θ is a threshold about 5° and θ_i is the angle of a scan line i in polar coordinates. If a line segment does not match one of the above two equations, it is considered as a nonorthogonal line and will not be taken into account for calculating the importance weight of the particle.

3.4.3. Vector-Based Line Representation. Before discussing the details of the data association step, we will explain the *vector-based line representation technique* [18, 20, 21] used in our line matching method, in which each line segment in the map is assigned a direction starting from its first point to its last as shown in Figure 7(a), so that each line segment can be presented as a vector. The main advantage of using a vector-ased line representation is we can avoid mismatch in enclosed areas (e.g., opposite sides of a thin wall) by determining the direction of the vectors. Figure 7(b) shows an example of using vector lines to represent an indoor environment, where the robot starts at one side of the wall (at starting point A) and moves along to the other point B. We can see that the vectors representing two sides of the wall (enclosed in a box in Figure 7) have different directions, and thus they can be easily distinguished from each other to avoid mismatch in the data association step.

3.4.4. Data Association. In the data association step, only orthogonal scan lines $\{S_i\}_{i=1,\ldots,N}$ of each particle are taken into account to find a best match in the local map $\{\text{LM}_j\}_{j=1,\ldots,l}$. If a scan line does not matches any lines in the map, it will be defined as a new observed line and be added into the map during the map-update step. Otherwise, the scan line is able to find the best match, which means it had already been observed by the robot previously and can be taken into account to calculate the particles importance weighting.

Our line matching algorithm for finding the best match in the local map $\{\text{LM}_j\}_{j=1,\ldots,l}$ for each scan line S_i is done by the following procedures.

(a) Detect if the angle θ_{ij} between a scan line S_i and a local map line LM_j is under a threshold. Here, we determine the line segments as vectors, so we can compute the angle quickly by

$$\theta_{ij} = \cos^{-1}\left(\frac{\vec{S_i} \cdot \vec{\text{LM}_j}}{\left|\vec{S_i}\right|\left|\vec{\text{LM}_j}\right|}\right). \tag{13}$$

(b) Detect if there is an overlap between S_i and LM_j by calculating L_{ov} as follows:

$$\begin{gathered} l_t = \sqrt{\left(x_e - x_s\right)^2 + \left(y_e - y_s\right)^2}, \\ L_{\text{ov}} = l_m + l_s - l_t = \begin{cases} \text{positive}, & \text{if there exists overlap}, \\ \text{negative}, & \text{otherwise}, \end{cases} \end{gathered} \tag{14}$$

where l_m is the length of a local map line LM_j, l_s is the length of a scan line S_i, and l_t is the length of a potential line L_t by merging S_i and LM_j, as shown in Figure 8.

(c) If one of the conditions mentioned above does not achieve, this means that S_i and LM_j are too far away

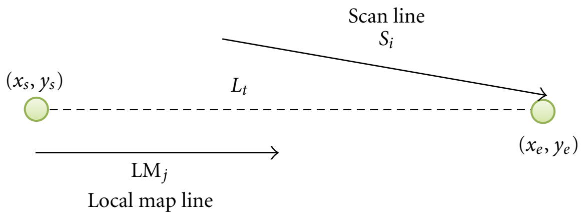


FIGURE 8: Overlap detection.

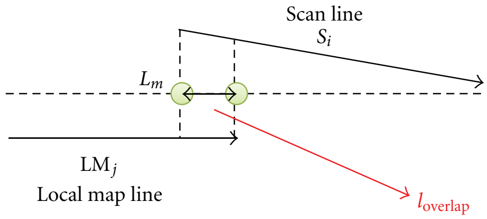


FIGURE 9: Illustration of the overlap distance l_{overlap} between the matching pair S_i and LM$_j$.

and could not be the best match. Otherwise the weighted Euclidean distance D_{Hj} [14] are computed to find out how well S_i and LM$_j$ matches by using the parameters θ and ρ of the two lines in Hough space as follows:

$$\begin{aligned} D_{Hj} &= \sqrt{w_\rho (\rho_s - \rho_m)^2 + w_\theta (\theta_s - \theta_m)^2}, \\ w_\rho + w_\theta &= 1. \end{aligned} \tag{15}$$

After all the lines in the local map $\{\mathrm{LM}_j\}_{j=1,\dots,l}$ are matched with the scan line S_i, we can determine the best match by finding a matched pair that has the smallest weighted Euclidean distance D_{Hj} by

$$D_H = \mathrm{MIN}(D_{H1}, D_{H2}, \dots, D_{Hl}). \tag{16}$$

If no matching pairs are found for a scan line S_i, we will determine the scan line as a new discovered line, and therefore insert it into the map in the map-update stage.

3.5. Importance Weighting. Each particle generated from the motion model is assigned an importance weight according to how well its current measurement of the environment matches the map we have already built, so that we can determine the accuracy of each particle by its importance weighting.

In our case, the importance weight W^i_{tot} for particle i is computed using the matching pairs we found in the data association step, by first assigning each matching pair a weight as follows:

$$w = l_{\text{overlap}} e^{-D_H}, \tag{17}$$

where D_H is the weighted Euclidean distance of the matched pair and l_{overlap} is the overlap distance between the matched scan line and the map line, which is shown in Figure 9.

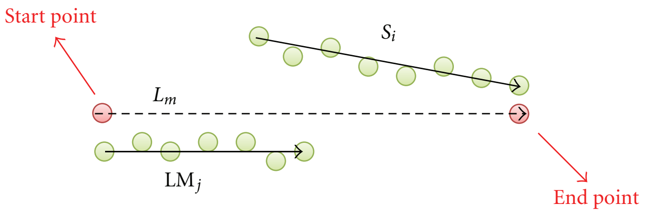


FIGURE 10: Merging two lines S_i and LM$_j$ by considering all the scan points.

After all the matched pairs are assigned a weight, we can compute the importance weight of the ith particle by

$$W^i_{\text{tot}} = \frac{N_m}{N_{\text{tot\_scan}}} \sum_{k=1}^{N_m} w_k, \tag{18}$$

where N_m is the number of matched pairs, $N_{\text{tot\_scan}}$ is the total number of orthogonal scan lines extracted in this iteration, and w_k is the weight of the kth matching pair. The main purpose of multiplying the ratio $N_m/N_{\text{tot\_scan}}$ is to avoid some particles having extremely high importance weight caused by matching pairs which have long overlap distance between them.

After assigning each particle an importance weight, the resampling step will take place by deleting the ones that have lower weights. Also particles that are assigned a higher weight (meaning that it has a higher probability to be the correct robot pose) will have the opportunity to generate more new offspring particles from the motion model in the next iteration.

3.6. Map Update. The map of each particle is maintained and updated in this step, where new observed lines are added to the map and lines that are close to each other, will be merged into a single line segment.

Figure 10 shows an example of merging a scan line S_i and a map line LM$_j$ into a single line L_m, where L_m is generated using all the scan points constituting S_i and LM$_j$. By considering all the points, line segments can be merged more accurately, but with higher requirement of memory since it requires the storing of all the points constituting the individual lines. To overcome this issue, the recursive least-squares (RLSs) method [20] for line merging is used in our work, which does not need to store all the points, but only requires six additional parameters $(N, S_{xx}, S_{yy}, S_{xy}, X_N, Y_N)$ of each line being merged. The RLS method operates by first updating the six parameters for the merged line and then determines the start/end points of the line by projecting the terminal points onto the new line. Here, we briefly review the RLS algorithm. More information can be found in [20].

In order to reduce the computational complexity of maintaining the map for each particle, the map-update procedure is split into two modes: (1) local map update and (2) global map update, where the local map update executes at every iteration and the global map update only executes once in a period of time.

Table 1: Summarizes the total memory, average time, and maximum time needed for the two algorithms.

	Total memory (MB)	Avg. time (ms)	Max. time (ms)
General approach	193.2 (100%)	453.4 (100%)	1062 (100%)
Proposed algorithm	10.7 (5.5%)	29.5 (6.5%)	78 (7.4%)

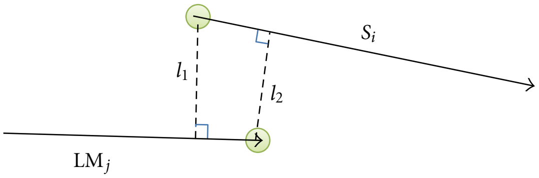


Figure 11: Illustrates the perpendicular distance l_1 and l_2 from the terminal points in the overlap section to the opposite line.

3.6.1. Local Map Update. In this step, new scan lines are added into the map and the matched pairs determined in the data association step will be merged together. Also line segments in the local map are merged if they are close to each other and have an overlap between them. The way of determining if two line segments in the local map can be merged together is similar to finding the matching pairs in the data association step, but instead of calculating the weighted Euclidean distance, we only calculates the average distance between two overlapped lines to determine if the lines can be matched as shown in Figure 11.

Where l_1 and l_2 are the perpendicular distance from the terminal points covering the overlap section to the opposite lines, the average distance can be computed as

$$l_{\text{dist}} = \frac{l_1 + l_2}{2}. \quad (19)$$

If the average distance l_{dist} is smaller than a threshold, the two lines are close to each other and will be merged together.

3.6.2. Global Map Update. In order to avoid having several broken line segments that are meant to be the same line in the map, we need to check all the map lines and determine whether the lines can be merged together by using the same way as in local map update, but this time the whole map will be tested. Due to the fact that checking all the line segments will cost an extremely large effort and the local map update can merge most of the broken lines together, we only execute the global map update once in a predetermined period of time. As a result, we are able to maintain the map more efficiently and reduce the number of line segments in our map.

4. Experimental Results

Our algorithm has been implemented for a P3-DX robot equipped with SICK LMS-100 laser rangefinder, where we also test its performance using different datasets from Radish [22]. All the experiments are performed on a laptop computer with 2G CPU and 3G RAM, which our algorithm is capable of running in real-time with low memory usage.

The first dataset is recorded by our P3-DX robot at the Center of Innovation Incubator, room 212, in our NTHU campus, where the size of the room is about 7 m × 4 m. Our main purpose is to test whether the algorithm can successfully detect the orthogonal lines in the environment even when the robot does not starts its initial pose aligned with the walls. Figure 12 shows an experiment of starting the robot at different directions, all of which are not aligned with the major structures of the environment. We use our proposed algorithm to determine the orthogonal lines (blue lines) and nonorthogonal lines (red lines) in the initial map. Also, error line segments caused by sensor noise or line extraction error can be filtered out by simply considering the orthogonal lines in the map. Figure 13(a) shows the floorplan of the room, and Figure 13(b) is the map of the entire room built by our algorithm, which only considers orthogonal lines in the environment. As we can see, by dynamically modifying the reference direction to identify the orthogonal lines, the algorithm is capable of determining the correct orthogonal direction even when the robot's initial pose is not aligned with the major structures of the environment.

The second experiment uses the dataset called *intel_oregon* by M. Batalin, which was recorded at Intel Lab in Hillsboro, Ore, USA, using a P2-DX robot with LSM-200 laser rangefinder. As mentioned in [20], the covering area of this dataset is about 750 m^2 and the total moving distance of the robot is 110 m, which contains several loops and complex indoor structures. In this experiment, we also implemented a general approach RPBF SLAM using grid map as its map structure to compare the total memory usage and computation time against our proposed lightweight SLAM algorithm, which uses line segments as our map structure. Figure 14 depicts the mapping result of *intel_oregon* using general approach (Figure 14(a)) and our proposed algorithm (Figure 14(b)), and Figure 15 shows the computation time of each iteration needed for executing the two algorithms, respectively. As it can be seen, our lightweight SLAM not only reduces the memory usage, but also needs less computation time as compared to the general RBPF SLAM. This is mainly due to the fact that line segment map only needs to store the geometric position and some parameters of each line, but grid map needs to store the whole environments information in a large 2D array. Also when a new particle is generated, a map built by the robot's previous trajectory is copied to the new particle so that each particle maintains its own map. Since copying memories is often time consuming, we can thereby increase the speed of the algorithm by lowering the amount of data needed to be copied. Table 1 summarizes the total required memory, average computation time, and maximum computation time needed in an iteration of the general approach RBPF and in our lightweight algorithm. As compared to the general approach RBPF SLAM, the total

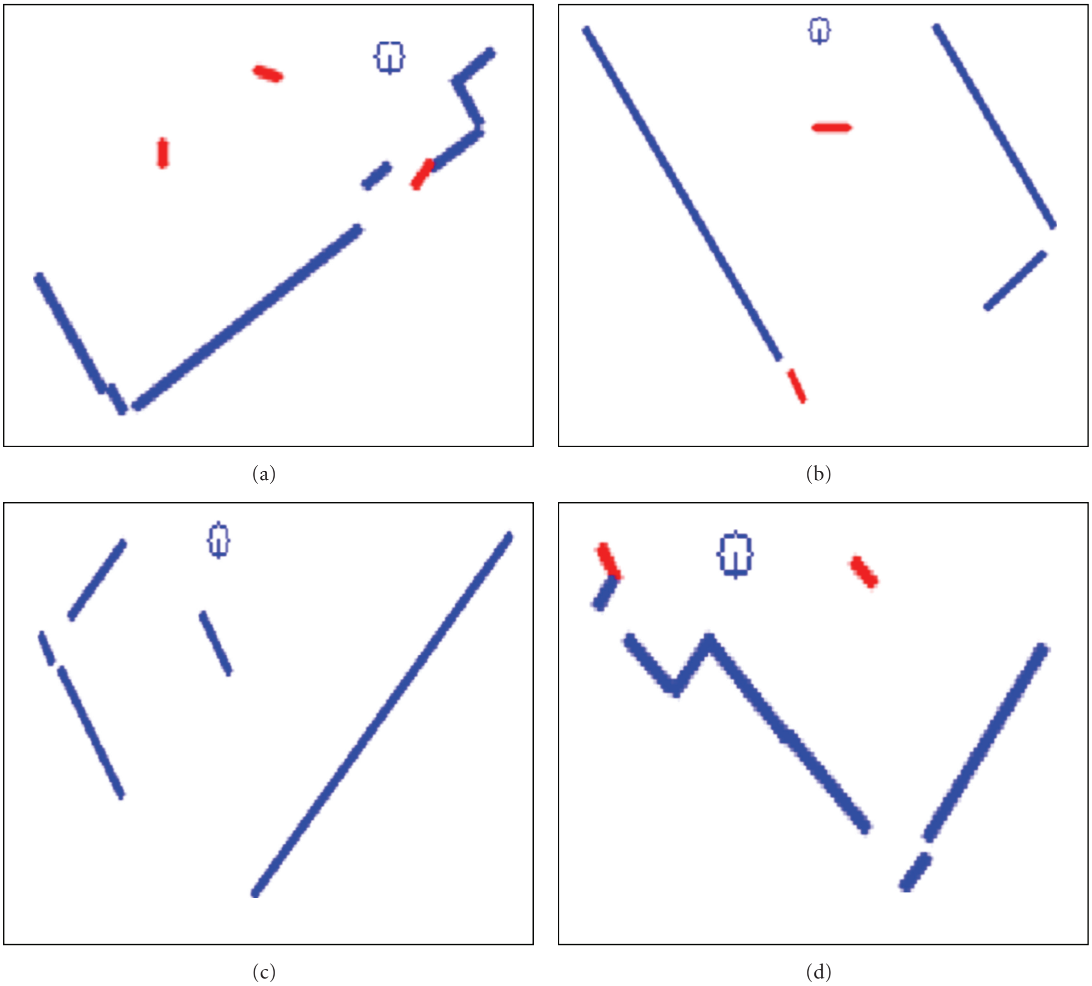


Figure 12: Initial map built by robots starting at different directions and indicates orthogonal (blue lines) and nonorthogonal (red lines) lines.

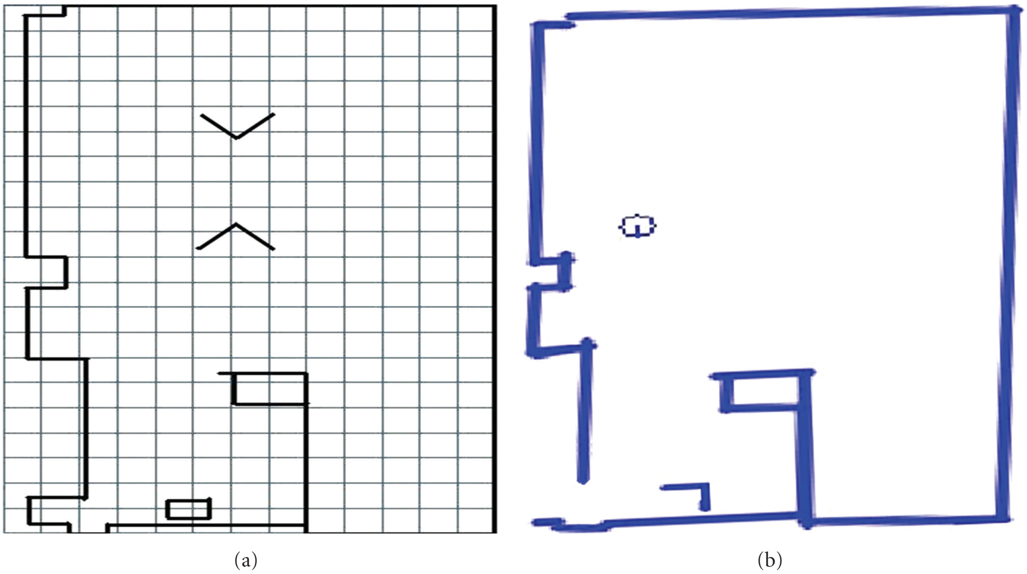


Figure 13: (a) Floorplan of the room. (b) Resulting map drawn by our algorithm.

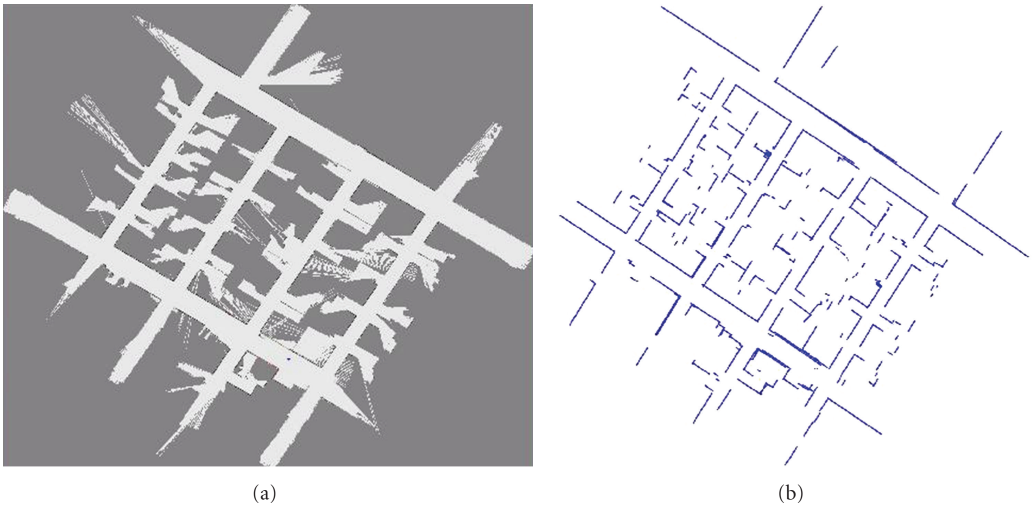


Figure 14: Mapping results of *intel_oregon* using (a) general RBPF SLAM (grid map) and (b) proposed lightweight SLAM (line segment map).

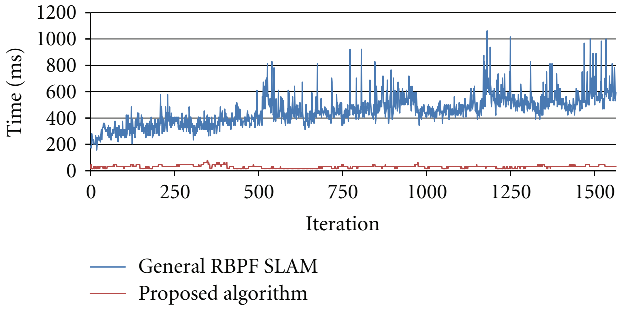


Figure 15: Computation time of each iteration needed for a grid map-based RBPF SLAM (blue line) and our proposed lightweight approach (red line).

memory usage and average computation time can be reduced to only 5.5% and 6.5%, respectively, by our algorithm. Since our algorithm is also much faster, it can support the SLAM operation for a faster-moving robot. In some sense, our algorithm is 453.4/29.5 = 15.36 times faster.

5. Conclusion

In this paper, we have successfully developed a robust light-and-fast RBPF-based SLAM algorithm for indoor environments. We use an enhanced sequential segmentation algorithm to increase the reliability of line segments extraction from the raw scan data. We also integrated the vector based line representation with the orthogonal assumption of indoor environments to avoid mismatches in the data association step. Experimental results show that our work needs much low amount of memory (e.g., only 5.5%) and much less computation time (e.g., only 6.5%) as compared to previous grid map-based RBPF SLAM. In other words, ours is capable of performing accurate SLAM for 15.36x faster-moving robots in complex indoor environments.

Acknowledgment

This work was supported in part by Project SOC Joint Research Lab sponsored by NOVATEK.

References

[1] W. Jeong and K. Lee, "CV-SLAM: a new ceiling vision-based SLAM technique," in *Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems*, Albert, Canada, August 2005.

[2] H. Myung, H. M. Jeon, and W. Y. Jeong, "Virtual door algorithm for coverage path planning of mobile robot," in *Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE '09)*, Seoul, Korea, July 2009.

[3] V. Nguyen, A. Harati, A. Martinelli, and R. Siegwart, "Orthogonal SLAM: a step toward lightweight indoor autonomous navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '06)*, Beijing, China, October 2006.

[4] K. Murphy, "Bayesian map learning in dynamic environments," in *Proceedings of the Conference on Neural Information Processing Systems (NIPS '99)*, pp. 1015–1021, Denver, Colo, USA, November-December 1999.

[5] A. Doucet, J. F. G. de Freitas, K. Murphy, and S. Russel, "Rao-Blackwellized particle filtering for dynamic Bayesian networks," in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI '00)*, pp. 176–183, Stanford, Calif, USA, June-July 2000.

[6] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "Fast-SLAM: a factored solution to the simultaneous localization and mapping problem," in *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, July-August 2002.

[8] A. Eliazr and R. Parr, "DP-SLAM: fast, robust simultaneous localization and mapping without predetermined landmarks,"

in *Proceedings of the the International Conference on Artificial Intelligence (IJCAI '03)*, Acapulco, Mexico, August 2003.

[9] W. J. Kuo, S. H. Tseng, J. Y. Yu, and L. C. Fu, "A hybrid approach to RBPF based SLAM with grid mapping enhanced by line matching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '09)*, St. Louis, Mo, USA, October 2009.

[10] C. Schroter, H.-J. Bohme, and H.-M. Gross, "Memory-efficient gridmaps in Rao-Blackwellized particle filters for SLAM using sonar range sensors," in *Proceedings of the European Conference on Mobile Robots (ECMR '07)*, pp. 138–143, Freiburg, Germany, September 2007.

[11] A. Eliazr and R. Parr, "DP-SLAM 2.0," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '04)*, New Orleans, La, USA, April-May 2004.

[12] S. Thrun, W. Burgard, and D. Fox, "A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '00)*, San Francisco, Calif, USA, April 2000.

[13] D. Hahnel, D. Schulz, and W. Burgard, "Map building with mobile robots in populated environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS' 02)*, EPFL, Switzerland, September-October 2002.

[14] Y. H. Choi, T. K. Lee, and S. Y. Oh, "A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot," *Autonomous Robots*, vol. 24, no. 1, pp. 13–27, 2008.

[15] P. Newman, J. Leonard, J. D. Tardos, and J. Neira, "Explore and return: experimental validation of real-time concurrent mapping and localization," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '02)*, Washington, DC, USA, May 2002.

[16] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, The MIT Press, 2005.

[17] A. I. Eliazar and R. Parr, "Learning probabilistic motion models for mobile robots," in *Proceedings of the International Conference on Machine Learning*, Alberta, Canada, July 2004.

[18] H. J. Sohn and B. K. Kim, "An efficient localization algorithm based on vector matching for mobile robots using laser range finders," *Journal of Intelligent and Robotic Systems*, vol. 51, no. 4, pp. 461–488, 2008.

[19] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249–275, 1997.

[20] H. J. Sohn and B. K. Kim, "VecSLAM: an efficient vector-based SLAM algorithm for indoor environments," *Journal of Intelligent and Robotic Systems*, vol. 56, no. 3, pp. 301–318, 2009.

[21] G. A. Borges and M. Aldon, "Line extraction in 2D range images for mobile robotics," *Journal of Intelligent and Robotic Systems*, vol. 40, no. 3, pp. 267–297, 2004.

[22] http://cres.usc.edu/radishrepository/view-all.php.

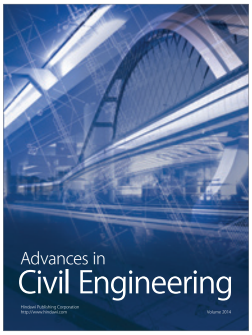
Journal of
Engineering

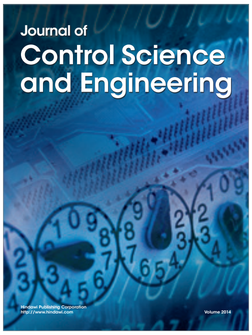
The Scientific
World Journal

International Journal of
Rotating
Machinery

Journal of
Sensors

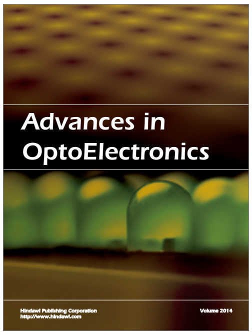
International Journal of
Distributed
Sensor Networks


Advances in
Civil Engineering


Journal of
Control Science
and Engineering

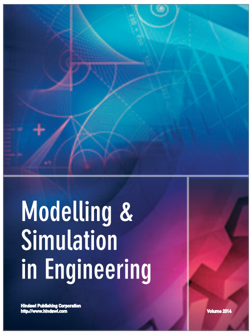
Journal of
Robotics

Journal of
Electrical and Computer
Engineering

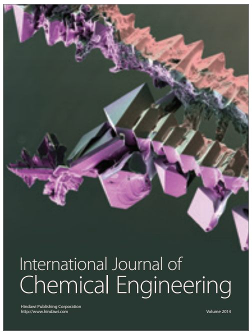

Advances in
OptoElectronics

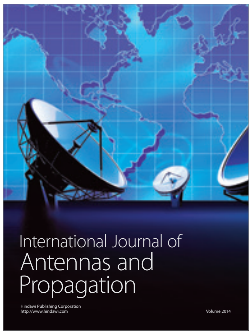
VLSI Design

International Journal of
Navigation and
Observation

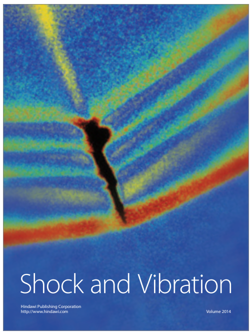

Modelling &
Simulation
in Engineering

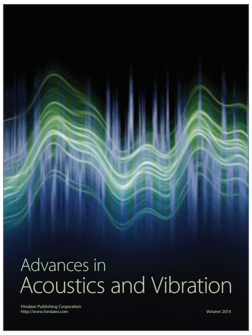
International Journal of
Aerospace
Engineering


International Journal of
Chemical Engineering


International Journal of
Antennas and
Propagation

Active and Passive
Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration