International Journal of Advanced Robotic Systems

OPEN ACCESS
ARTICLE

On a Hopping-points SVD and Hough Transform-based Line Detection Algorithm for Robot Localization and Mapping

Regular Paper

Abhijeet Ravankar[1]*, Ankit A. Ravankar[1], Yohei Hoshino[2], Takanori Emaru[1] and Yukinori Kobayashi[1]

1 Laboratory of Robotics and Dynamics, Division of Human Mechanical Systems and Design, Hokkaido University, Sapporo, Japan
2 Laboratory of Bio-Mechatronics, Department of Mechanical Engineering, Kitami Institute of Technology, Kitami, Japan
*Corresponding author(s) E-mail: abhijeetravankar@gmail.com

Received 24 January 2016; Accepted 08 April 2016

DOI: 10.5772/63540

Abstract

Line detection is an important problem in computer vision, graphics and autonomous robot navigation. Lines detected using a laser range sensor (LRS) mounted on a robot can be used as features to build a map of the environment, and later to localize the robot in the map, in a process known as Simultaneous Localization and Mapping (SLAM). We propose an efficient algorithm for line detection from LRS data using a novel hopping-points Singular Value Decomposition (SVD) and Hough transform-based algorithm, in which SVD is applied to intermittent LRS points to accelerate the algorithm. A reverse-hop mechanism ensures that the end points of the line segments are accurately extracted. Line segments extracted from the proposed algorithm are used to form a map and, subsequently, LRS data points are matched with the line segments to localize the robot. The proposed algorithm eliminates the drawbacks of point-based matching algorithms like the Iterative Closest Points (ICP) algorithm, the performance of which degrades with an increasing number of points. We tested the proposed algorithm for mapping and localization in both simulated and real environments, and found it to detect lines accurately and build maps with good self-localization.

Keywords Line-segment Detection, Singular Value Decomposition, Hough Transform, Data Association, Simultaneous Localization and Mapping (SLAM)

1. Introduction

Simultaneous Localization and Mapping (SLAM) is an indispensable component of any autonomous robot which deals with making a map of its surroundings and localizing itself in the map at the same time. The robot perceives the exterior environment from sensors such as a laser range sensor (LRS), RGB-camera, stereo camera, inertial sensors, GPS, etc. The robot has wheel encoders which give estimates about how much the robot has moved in the environment. The sensor data and encoder data are both used to build a map and estimate the position of the robot in the map. It is a challenging problem, mainly because the sensors and encoders are prone to errors, which is further complicated by the dynamics of the environment. These errors accumulate over time as the robot moves in the environment. Hence, a robot must incrementally build the

map and keep on correcting both the map and its position at each step. Various approaches to solve this problem have been proposed and the earliest work can be found in the works of Smith et al. [1]. Probabilistic approaches including Extended Kalman Filter (EKF) SLAM have been proposed in [2], which uses particle filters. Similar approaches using different sensors like cameras [3] and depth sensors [4] have also been proposed.

In most of the approaches used to solve SLAM, a robot needs to match its current data with the previously saved map data to estimate its position and then apply correction. Matching can either be point-based or feature-based. In the former, feature points obtained from sensors such as LRS are directly matched, such as in [5, 6, 7]. One of the most common point-matching algorithms is ICP (Iterative Closest Point) [8]. ICP is a classical rigid-point set registration algorithm, and many variants of ICP have been proposed in [9, 10, 11, 12]. When given two point clouds, ICP algorithm tries to iteratively minimize the least square difference between them to determine rotation and translation. A major drawback of ICP is that it is computationally complex and the performance deteriorates for large number of points. ICP has a computational complexity of $\mathcal{O}(m^d n^d)$ [13], where m and n are the number of points in the two point sets in a d dimensional space. Faster variants of ICP which employ KD-tree [14], such as [15], have also been proposed and accelerating ICP is an active area of research.

On the other hand, feature-based matching utilizes map features like line segments, corners, colour histogram information, scale invariant features (SIFT) [16] and others to match the sensor data. Compared to point-based matching, feature-based mapping systems generally require less memory, are relatively computationally efficient, and have been extensively utilized in SLAM. For distance sensors such as LRS, line segment-based SLAM has been studied extensively, employing line extraction algorithms such as Split and Merge [17] and incremental [18] algorithms. Among these, some of the most prominent algorithms for line extraction are Hough transform [19, 20, 21] and RANSAC [22]. Other techniques include the Expectation-Maximization algorithm [23, 24], line regression [25], and clustering-based line map generation [26, 27, 28]. Mapping based on line segments have also been reported in [29, 30, 31]. Line-segment extraction and its integration in particle filter SLAM has been proposed in [32, 33]. A good summary of various line-extraction techniques can be found in [34].

This paper proposes a novel line detection algorithm using SVD and Hough transform with good computational efficiency. In order to speed up the line extraction process, we propose a hopping mechanism in which SVD is applied to intermittent points. A reverse hop mechanism ensures that the end points of each line segment are accurately extracted. Moreover, the line extraction algorithm is integrated in a SLAM framework and the map comprising the extracted lines are used to match the current LRS data for efficient localization.

This paper is structured as follows: Section 2 explains the problem of straight-line detection from LRS data using Hough transform. Section 3 explains incorporating SVD in the equations of Hough transform to accurately detect lines. The forward and reverse hopping scheme to accelerate the algorithm are explained in Section 3.3. The proposed line extraction algorithm is integrated into SLAM and explained in Section 4. Mapping is explained in Section 4.1 and localization in Section 4.2. Results are discussed in Section 5. Section 5.1 discusses the simulation results, and Section 5.2 shows the results of the experiments in a real environment. Finally, Section 6 concludes the paper.

2. Straight-Line Detection from LRS data with Hough Transform

A typical indoor environment of a building is shown in Fig. 1. It consists of straight passages which are diverting and opening in both left and right directions. A robot with an LRS sensor mounted on it moves through the passage and the sensor records the points on the edges of the walls of the environment, which have been indicated by heavy black lines. How far an LRS can detect these points depends on the specifications of the sensor. In general, every LRS sensor is characterized by a maximum detection distance $d_{\max}$, the minimum detection distance $d_{\min}$, the maximum scan angle $\theta_{\max}$ and the minimum scan angle $\theta_{\min}$. The centre of the scan angle is, therefore, $\theta_{\text{center}}=(\theta_{\min}+\theta_{\max})/2$. We set this centre on the coordinate system O-XY with the absolute rotating angle ϕ_k. This is shown in Fig.2 which represents the kth scan step. In Fig.2, o-xy is a local coordinate system fixed on the LRS and the y-axis agrees with θ_{center}.

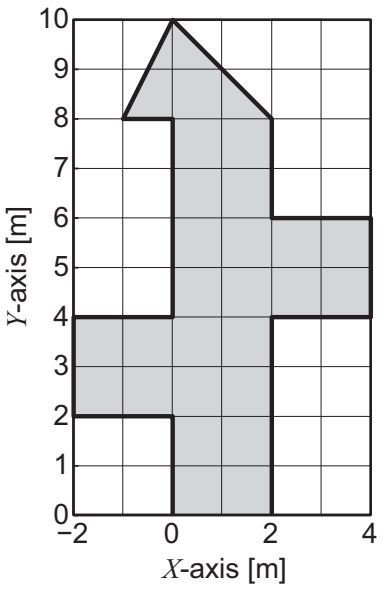


Figure 1. Environment for simulation

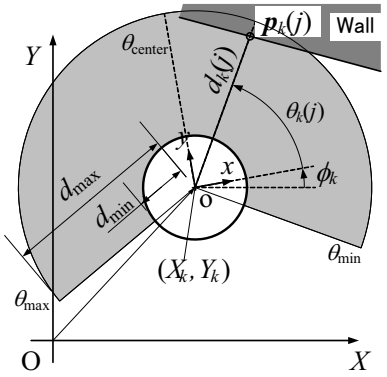


Figure 2. Local coordinate for laser-range sensor

As shown in Fig.2, LRS outputs a point $\boldsymbol{p}_k(j)$ at the jth scan at an angle $\theta_k(j)(1 \le j \le N)$ measured from the x-axis. Here, the number N is the resolution with respect to the angle of scanning. Figure 3 shows the LRS output at position X_k =1 m, Y_k =1 m, and angle ϕ_k =20 deg in the map shown in Fig. 1.

The position of the detected wall is indicated by the heavy line in Fig.2 and is expressed as,

$$\boldsymbol{p}_k(j) = \begin{bmatrix} p_{xk}(j) \\ p_{yk}(j) \end{bmatrix} = \begin{bmatrix} d_k(j)\cos(\theta_k(j)) \\ d_k(j)\sin(\theta_k(j)) \end{bmatrix} \tag{1}$$

on the coordinate system o-xy.

The vector $\boldsymbol{p}_k(j)$ $(1 \le j \le N)$ represents the N number of points on the coordinate system o-xy. We can apply Hough transform [35] to detect a straight line from the measured N points.

The Hough transform of the jth point $(p_{xk}(j), p_{yk}(j))$ is expressed as,

$$\rho(\theta) = p_{xk}(j)\cos(\theta) + p_{yk}(j)\sin(\theta). \tag{2}$$

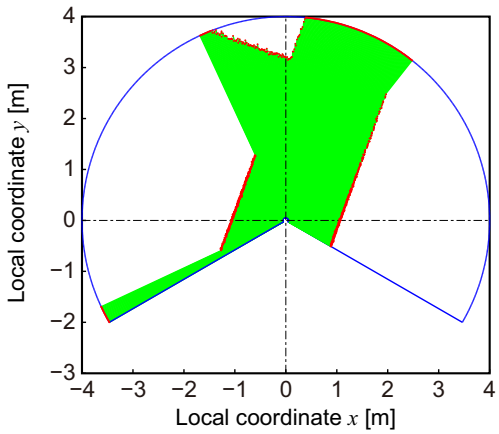


Figure 3. Example of laser-range sensor data

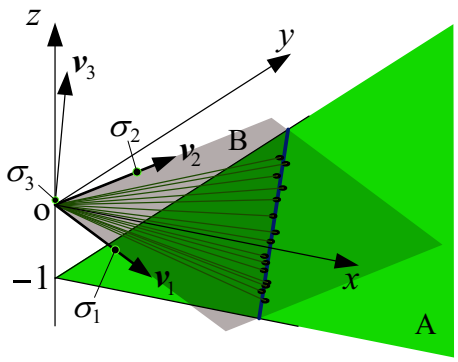


Figure 4. Detection of a line from points with SVD

Hough transform changes the points from Cartesian (x,y) space to (ρ,θ) space. It can be seen that the value of ρ is a function of θ and, hence, Hough transform on a set of points which lie on the same line intersect at a common point in the (ρ,θ) accumulator space. Hough transform employs a voting scheme and the point (ρ,θ) with the maximum number of votes is the point which corresponds to the straight line passing thorough the points.

If the sensor is accurate without any errors, and there is no noise, then all the points on the line will intersect at exactly the same point (with maximum votes) in the (ρ,θ) accumulator space. However, non-linear points, or linear points in the presence of sensor and environmental noise, do not intersect at a single (ρ,θ) point, and multiple points with the same maximum votes may be generated. In order to extract the lines, we have to estimate (ρ,θ) points with maximum votes. Hence, when N measured points make multiple lines, a two-dimensional search is required in the θ and ρ plane, which is computationally expensive.

3. SVD-Based Line Detection Algorithm

This section describes the proposed algorithm for detecting straight line segments using SVD and Hough transform.

3.1 Straight line detection with SVD

We can re-write the equation of Hough transform (2) as,

$$p_{xk}(j)\cos(\theta) + p_{yk}(j)\sin(\theta) - \rho(\theta) = 0. \tag{3}$$

This equation can be further expressed as,

$$\begin{bmatrix} p_{xk}(j) & p_{yk}(j) & -1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & \rho(\theta) \end{bmatrix}^{\mathrm{T}} = 0. \tag{4}$$

A regression line which starts from the ith point and comprises a total of l points is determined by the combination of ρ and θ, and satisfies the equation,

$$\boldsymbol{A}_i(l)\boldsymbol{X}(\rho,\theta) = \boldsymbol{0}\,(1 \le i \le N-2,\ 3 \le l \le N-i+1), \tag{5}$$

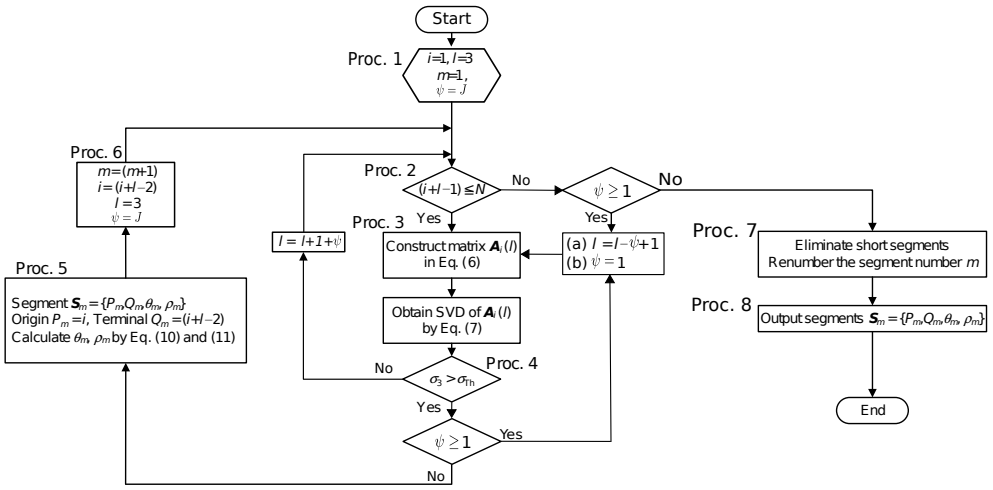


Figure 5. Flow chart of line-segment detection

where

$$\boldsymbol{A}_i(l) = \begin{bmatrix} p_{xk}(i) & p_{yk}(i) & -1 \\ \vdots & \vdots & \vdots \\ p_{xk}(i+l-1) & p_{yk}(i+l-1) & -1 \end{bmatrix}, \qquad \boldsymbol{X}(\rho,\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) & \rho(\theta) \end{bmatrix}^{\mathrm{T}} \neq \boldsymbol{0} \tag{6}$$

We employ Singular Value Decomposition (SVD) to solve this equation. SVD of the matrix $\boldsymbol{A}_i(l)$ is expressed as

$$\boldsymbol{A}_i(l) = \boldsymbol{U}\,\boldsymbol{S}\,\boldsymbol{V}^{\mathrm{T}}, \tag{7}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}^{\mathrm{T}}$ are the orthonormal matrices consisting of left and right singular vectors. $\boldsymbol{S}$ is the diagonal matrix with singular values arranged in decreasing order along the diagonal as follows:

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 \cdots \boldsymbol{u}_l \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} \mathrm{diag}(\sigma_1\ \sigma_2\ \sigma_3) \\ \boldsymbol{0} \end{bmatrix} (\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0), \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1\ \boldsymbol{v}_2\ \boldsymbol{v}_3 \end{bmatrix}. \tag{8}$$

Here, the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ satisfy $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{V}^{\mathrm{T}}\boldsymbol{V} = \boldsymbol{I}$. The matrix $\boldsymbol{I}$ represents the identity matrix. When the minimum singular value σ_3 is sufficiently small and has near-zero value, the least square solution of Eq.(5) is given by the right singular vector $\boldsymbol{v}_3$ corresponding to σ_3.

When a sufficiently small singular value cannot be obtained, it means that it is difficult to determine a regression line of l points, starting from the ith point. It also implies that the coefficient of correlation of these points gives a near-zero value.

Figure 4 gives a graphical intuition of the proposed method. Points indicated by • are the points measured by LRS. The detected straight line is shown by a bold line. The z -axis is normal to the x - y plane. We can interpret Eq.(3) as expressing a plane of $x = p_{xk}(j)$, $y = p_{yk}(j)$, $z = -1$ on the coordinate system o- xyz. In the proposed method, placing the measured points $(p_{xk}(j),\ p_{yk}(j))$ on the plane $z = -1$ by Eq.(4), singular vectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$ are derived by SVD.

A straight line is detected as a line of intersection of the plane which consists of the vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, and the plane $z = -1$. The minimum singular value σ_3 indicates the variation in measured points along the direction of the singular vector $\boldsymbol{v}_3$.

A threshold σ_{Th} is defined for σ_3. When σ_3 is smaller than σ_{Th}, the solution $\boldsymbol{X}(\rho,\theta)$ of Eq.(5) is expressed as,

$$\boldsymbol{X}(\rho,\theta) = \alpha \boldsymbol{v}_3. \tag{9}$$

where α is an undetermined factor. Expressing the singular vector $\boldsymbol{v}_3$ as,

$$\boldsymbol{v}_3 = \begin{bmatrix} v_{31}\ \ v_{32}\ \ v_{33} \end{bmatrix}^{\mathrm{T}}, \tag{10}$$

we get,

$$\cos(\theta) = \alpha v_{31},\ \ \sin(\theta) = \alpha v_{32},\ \ \rho = \alpha v_{33} \tag{11}$$

from Eq.(6) and α is determined as,

$$\alpha = \frac{1}{\sqrt{v_{31}^2 + v_{32}^2}} \tag{12}$$

from the relationship $\cos^2(\theta) + \sin^2(\theta) = 1$.

3.2 Line-segment detection algorithm

In this subsection, we describe the entire process of line segment detection. A flow-chart of the algorithm is shown in Fig.5. The parameter ψ in Fig.5 deals with the 'hopping' feature of the algorithm and is explained in detail in Section 3.3. First, the algorithm is explained without the hopping feature, in which the value of ψ is set to 0 (i.e., $J=0$).

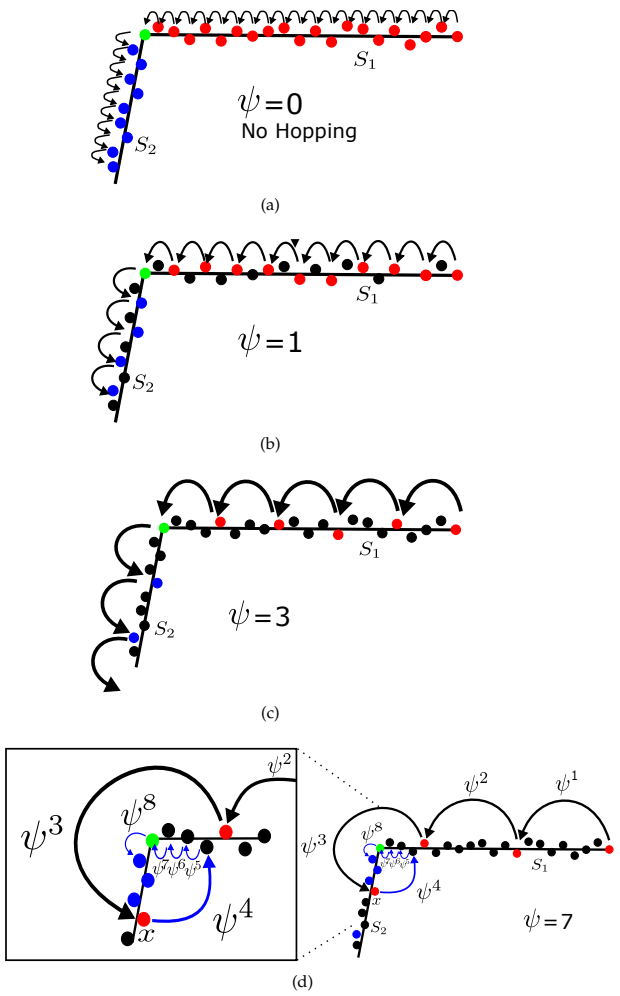


Figure 6. Explanation of hopping point SVD-based line segment detection. (a) No hopping with $\psi=0$. SVD is applied to each and every point. (b) Line detection with $\psi=1$. SVD is applied to every intermittent point. (c) Line detection with $\psi=3$. SVD is applied after skipping three points. (d) Hopping with $\psi=7$ and reverse hopping. Reverse hopping occurs at point x and the end point shown in green is accurately detected.

The line detection process comprises the following steps:

Proc.1 In order to detect the mth line segment, which consists of the total l points and starts from the ith point, we set the initial parameters as, i=1, l=3 and m=1. This means that the process will detect the first (m=1) line segment, which starts from i=1 and has just three points initially (the number of points will increase incrementally).

Proc.2 Since the maximum number of points in an LRS scan is N, we check if $(i+l-1)$ is greater than N. If it is greater than N, it means that the algorithm has already been applied to all of the measured points, and line segments have been extracted. Therefore, we finish the line-segment detection and jump to **Proc.7**.

Proc.3 We construct a matrix $\boldsymbol{A}_i(l)$ from Eq.(6), and apply SVD given by Eq.(7) to obtain the minimum singular value σ_3.

Proc.4 If the minimum singular value σ_3 is less than σ_{Th}, the points from the ith point to the $(i+l-1)$ th point are estimated to form a straight line. The next $(i+l+\psi)$ th point ($\psi=0$, for the non-hopping case) might also exist on the same straight line, and hence we set $l=(l+1)$ and jump back to **Proc.2**. If we reach a point where σ_3 is greater than σ_{Th}, we execute the next **Proc.5**.

Proc.5 A value of σ_3 greater than σ_{Th} means that the $(i+l-1)$ th point cannot be estimated on the straight line constructed by points from the ith point to the $(i+l-2)$ th point. A new line segment starts from this point. Hence, we first extract the m th line segment which contains the points from the ith point to the $(i+l-2)$ th point. We set the origin point $P_m=i$ and the terminal point $Q_m=(i+l-2)$. The slope θ_m and the distance ρ_m from the origin point of the line segment are determined by Eqs.(11) and (12). Hence, the mth line segment S_m is detected and stored as four parameters $\{P_m, Q_m, \theta_m, \rho_m\}$.

Proc.6 To start detecting the next line segment, segment number m is incremented by one to m+1. The end point of the previous line segment (S_m) marks the beginning of the (m+1)th line segment and, hence, it is set to $(i+l-2)$. The total number of points l of the new (m+1)th line segment is initialized to three, and the line-extraction process is repeated from **Proc.2** to extract the next line segment.

Proc.7 Short line segments are eliminated and line segments $\boldsymbol{S}_m$ are renumbered. Concretely, the total number of points of the detected line segment $\boldsymbol{S}_m$ is calculated as (P_m-Q_m+1) points. Only longer line segments $\boldsymbol{S}_m$ which contain more than L_{Th} points are kept. Here, L_{Th} is a predetermined threshold which controls the length of the lines to be detected. Since some shorter line segments get deleted, the line segment number m is renumbered.

Proc.8 All the extracted line segments $\boldsymbol{S}_m$ are outputted and the algorithm stops.

The accuracy of the line segments are determined by the threshold σ_{Th} and L_{Th} for the minimum singular value σ_3 and the length of the line segment. Note that if σ_{Th} is set to an extremely small value, many short line segments will be extracted, as LRS point data contain measurement error and noise, which is not desired.

3.3 Accelerating the line detection algorithm using hopping-points SVD

In the proposed line-detection method, SVD is applied to each and every point, and the lower singular value (σ_3) is

checked against a threshold value (σ_{Th}) for each point. SVD is computationally expensive and the computational complexity of an $m \times n$ matrix is $\mathcal{O}(mn^2)$. However, we need not apply SVD to each and every point. Instead, SVD is applied to intermittent points governed by a hopping factor ψ. The basic idea is that if a set of points lie on the same regression line, we can assume that the consecutive points will also lie on the same line and are skipped. However, the intermittent points are still included and SVD is applied to check if the hypothesis still holds and whether the points can still be skipped. Even after skipping ψ points, $\sigma_3 < \sigma_{Th}$ indicates that the hypothesis holds and the skipped points also lie on the same regression line. Concretely, skipping the ith point means that SVD is not applied to matrix $\boldsymbol{A}_i(l)$ in Eq.6 for the $i>1$ points. However, in the next hop at the jth point ($j>i>1$), the matrix $\boldsymbol{A}_i(l)$ in Eq.6 includes all the points from one to j. Parameter ψ determines the number of points which are skipped before applying SVD.

This process is explained graphically in Fig.6. The normal SVD on every point is shown in Fig.6(a). Fig.6(b) shows SVD applied to intermittent points with $\psi=1$, i.e, one point (shown in black) is skipped. Fig.6(c) shows the hopping mechanism with $\psi=3$ and three points are skipped. In both Fig.6(b) and Fig.6(c), note that the constructed matrix $\boldsymbol{A}_i(l)$ from Eq.(6) contains the information of skipped points too, and SVD is applied by Eq.(7) to obtain the minimum singular value σ_3. Skipping points accelerates the process.

A 'reverse-hop' mechanism ensures that the end points of line segments are accurately extracted. In both Fig.6(b) and Fig.6(c), the hop accurately terminates at the end-points of the line segments S_1 and S_2 (indicated by the colour green). However, this is not true in the case of Fig.6(d) with a larger value of $\psi=7$. In Fig.6(d), ψ^i represents the i th hop. It can be observed that in the third hop, i.e., ψ^3, the actual end point of the segment S_1 is missed, as a larger value of σ_3 will be detected at point x, as shown in Fig.6(d). To correct this, whenever σ_3 is detected to be greater than σ_{Th}, a check is performed if the value of the hopping parameter (ψ) is greater than or equal to one. In that case, a 'reverse-hop' is performed and the algorithm goes back to the next point of the previous hop's initial point and ψ is set to zero. A reverse-hop is shown in Fig.6(d) by the ψ^4, which takes it to the next point of the previous hop position. The value of ψ is set to zero and, hence, no hopping is performed in ψ^5,ψ^6 and ψ^7. The condition $\sigma_3 > \sigma_{Th}$ is detected in ψ^8, and the end point of S_1 is correctly detected. After successfully extracting line segment S_m, ψ is set back to $\psi=J$ in **Proc.6**, i.e., hopping is reset for the next segment.

If the time taken to compute SVD over n points of a line segment is given by the function $t(n)$, the algorithm gets speeded up by a factor of,

$$\frac{\sum_{j=1}^{n-1} t(j)}{\sum_{j=1}^{h_{\text{major}}} t\big(1+(j-1)(\psi+1)\big) + \sum_{j=h_{\text{major}}+1}^{h_{\text{total}}} t(j)}, \tag{13}$$

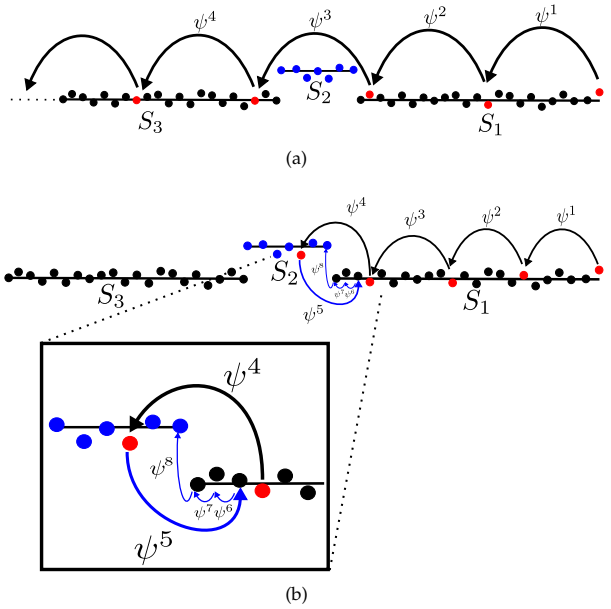


Figure 7. (a) Overhopping with a large value of Ψ. (b) An appropriate value of Ψ causes a reverse hop and the end point is detected accurately.

where h_{major} is the total number of long hops and h_{total} is the total number of hops (long hops when $\psi \geq 1$ and short hops when $\psi=0$), and is given by,

$$\begin{aligned} h_{\text{major}} &= \left(\frac{n-1}{\psi+1}\right), \\ h_{\text{total}} &= \left(\frac{n-1}{\psi+1}\right) + \underbrace{(n-1) \bmod (\psi+1)}_{\text{minor-singular-hops-without-skipping}}. \end{aligned} \tag{14}$$

Note that when $\psi=0$, the algorithm is the same when no hopping is applied.

Setting ψ to a large value might seem advantageous but it may lead to 'overhopping' and some shorter line-segments' detection might be skipped. This may lead to a drop in the accuracy of the line detector, as the end points of line segments might not be detected correctly. As shown in Fig. 7(a), segment S_2 might be omitted in the hop ψ^3. Setting ψ to a value less than $L_{Th}/2$ prevents the skipping of shorter segments, and the reverse-hop mechanism ensures accurate end-point detection as shown in Fig.7(b).

3.4 Comparison with Hough transform

Figure 8(a) shows a typical example of LRS data in which points are not necessarily linear due to sensor and environment noise. Figure 8(b) shows the corresponding Hough (ρ,θ) space which is obtained after applying Hough transform [19, 20, 21] to the LRS data. Since the points are non-collinear, they generate multiple peaks in the (ρ,θ) space with a maximum number of votes. A computationally expensive two-dimensional search is required to detect the maximum votes in Hough space. Multiple lines are

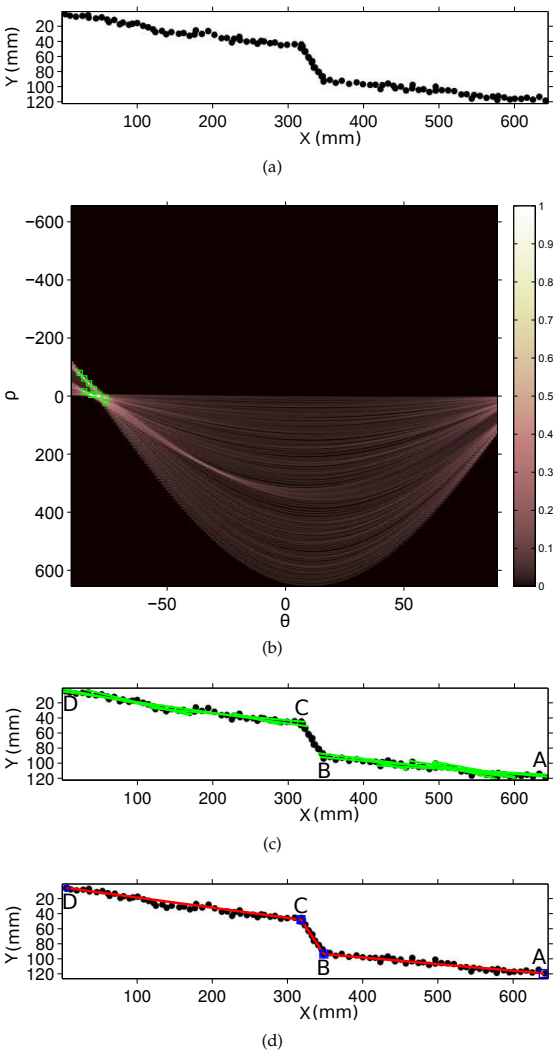


Figure 8. Comparison of the proposed method with Hough transform-based line detection. (a) A typical example of LRS scan data with three line segments. (b) Hough space of the data with multiple points, with maximum votes shown in green boxes. (c) Multiple lines extracted by Hough transform. (d) Line segment detected by the proposed method. The start point and end point of each segment are shown by blue boxes and circles, respectively.

extracted corresponding to these peaks in Hough space, as shown in Figure 8(c). Applying inverse Hough transform to these (ρ,θ) pairs results in the generation of multiple lines within segments AB and CD, most of which are overlapping. The shorter segment BC is not extracted as it has fewer votes in (ρ,θ) space. In order to detect segment BC, the maximum vote value needs to be lowered. However, that results in an even larger number of shorter segments to be extracted within segments AB and CD. This has adverse effects on the robot's localization as there are multiple lines corresponding to the same feature. In contrast, the proposed method detects the segments accurately, as shown in Figure 8(d). In the proposed method, parameter σ_{Th} governs the permissible deviation of points along a segment and L_{Th} controls the length of the shortest segment to be detected. The shorter segment BC is also accurately detected in Figure 8(d) using the proposed method. Regarding the execution time, Hough transform took 157.42 ms to detect the segments in Figure 8(c). On the other hand, by using the proposed method with hopping (ψ=5), accurate line segment extraction could be done in 80.6 ms. Both computations were performed using MATLAB 7.12 (R2011a) running on a Linux machine with an Intel Core i7-4500U CPU, 1.80 GHz and 16GB RAM.

Like Hough transform, the proposed method is sensitive to outliers. RANSAC [22] gives better performance than the proposed method in the case of data with many outliers. However, RANSAC requires a large number of iterations, and requires the setting of many parameters like the minimum number of points, the threshold distance between points and fitting line and the inlier ratio, which are often problem-specific thresholds [36, 22]. In the presence of too many outliers, the proposed method can easily be combined with noise-removal techniques [37, 26] to detect line segments from LRS data.

4. Integration of the Line-extraction Algorithm into the SLAM Framework

This section explains the integration of the proposed line-extraction algorithm into the framework of simultaneous localization and mapping. As shown in Fig.2, at the k th scan step, LRS is at the absolute attitude angle of ϕ_k and position (X_k,Y_k). Initially, when k=1, the position is given by X_1, Y_1 and ϕ_1. At the k th scan step, the true values of the absolute attitude angle ϕ_k and position (X_k,Y_k) are assumed to be unavailable. Instead, the estimated values of absolute attitude angle $\hat{\phi}_k$ and position $\hat{X}_k$, $\hat{Y}_k$, are calculated. When the robot starts its operation, the position at the first scan step is assumed as the anchor point and both the true and estimated values are set as being equal, i.e., $\hat{X}_1=X_1$, $\hat{Y}_1=Y_1$, and $\hat{\phi}_1=\phi_1$.

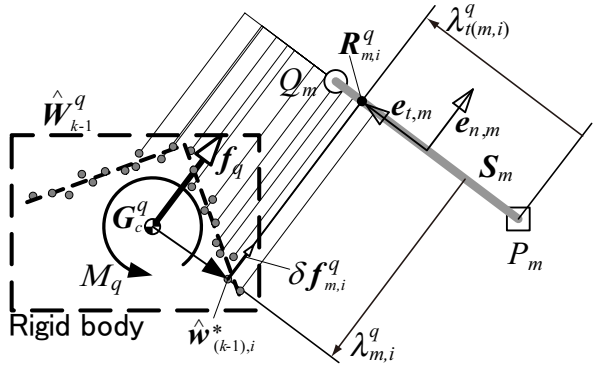


Figure 9. Matching algorithm

4.1 Mapping from LRS information

The output data of the LRS at the kth scan step is given by $\boldsymbol{p}_k(j)\,(1\le j\le N)$ of Eq.(1).

Let $\hat{\boldsymbol{V}}_k$ represent the map of the surrounding environment. It consists of LRS data given by,

$$\hat{\boldsymbol{V}}_k = \left[\hat{\boldsymbol{p}}_k(1) \quad \cdots \quad \hat{\boldsymbol{p}}_k(j) \quad \cdots \quad \hat{\boldsymbol{p}}_k(N)\right], \tag{15}$$

where $\hat{\boldsymbol{p}}_k(j)$ is the surrounding environment data obtained from the estimated position by applying coordinate transform from coordinate system o $-xy$ fixed on the LRS to the coordinate system O $-XY$ for the environment data $\boldsymbol{p}_k(j)$. In order to carry out this transformation, the rotational transform matrix $\boldsymbol{T}(\hat{\phi})$ and the parallel translation vector $\hat{\boldsymbol{X}}_k$ are given as,

$$\boldsymbol{T}(\hat{\phi}) = \begin{bmatrix} \cos(\hat{\phi}) & -\sin(\hat{\phi}) \\ \sin(\hat{\phi}) & \cos(\hat{\phi}) \end{bmatrix}, \hat{\boldsymbol{X}}_k = \begin{bmatrix} \hat{X}_k \\ \hat{Y}_k \end{bmatrix}. \tag{16}$$

Hence, we obtain $\hat{\boldsymbol{p}}_k(j)$ as,

$$\begin{aligned} \hat{\boldsymbol{p}}_k(j) \quad \left(= \begin{bmatrix} \hat{p}_{Xk}(j) \\ \hat{p}_{Yk}(j) \end{bmatrix} \right) &= \boldsymbol{T}(\hat{\phi})\boldsymbol{p}_k(j) + \hat{\boldsymbol{X}}_k \\ &= \begin{bmatrix} d_k(j)\cos(\hat{\phi} + \theta_k(j)) + \hat{X}_k \\ d_k(j)\sin(\hat{\phi} + \theta_k(j)) + \hat{Y}_k \end{bmatrix}. \end{aligned} \tag{17}$$

We can represent the estimated global map ($\hat{\boldsymbol{W}}_k$) made by accumulating the LRS data at k scan steps as,

$$\hat{\boldsymbol{W}}_k = \left[\hat{\boldsymbol{W}}_{k-1} \;\; \hat{\boldsymbol{V}}_k\right] = \left[\hat{w}_{k,1}\, \hat{w}_{k,2} \cdots \hat{w}_{k,N_{\text{map}}}\right], \tag{18}$$

where $\hat{w}_{k,i}$ is the ith column vector of matrix $\hat{\boldsymbol{W}}_k$, and N_{map} represents the total count of points stored in $\hat{\boldsymbol{W}}_k$.

At the initial condition, the estimated global map is given by $\hat{\boldsymbol{W}}_1 = \hat{\boldsymbol{V}}_1$. Also, at the very first scan, the true and estimated map is the same, i.e., $\hat{\boldsymbol{V}}_1 = \boldsymbol{V}_1$.

The robot updates the estimated global map at each scan. In order to do so, estimated values of absolute position $\hat{X}_k$, $\hat{Y}_k$ and absolute attitude angle $\hat{\phi}_k$ are necessary. The next subsection describes how to calculate $\hat{X}_k$, $\hat{Y}_k$ and $\hat{\phi}_k$.

4.2 Self-localization by matching LRS data with the estimated global map

For very short distances, the estimated value of the absolute position and absolute attitude angle of the LRS can be obtained from wheel encoders. However, for long distances, the odometry errors continue to integrate and we cannot rely on encoder data alone. Therefore, correction of the estimated values by using landmarks is necessary.

An outline of the correction of the self-localization method's estimation value is shown in Fig.9.

After executing one LRS scan at the k th step, we apply the algorithm mentioned in Subsection 3.2 to the N points of data $\boldsymbol{p}_k(j)(1 \le j \le N)$ of Eq.(1). A total of N_{Seg} line segments are extracted, from which the shorter segments are removed and $\boldsymbol{S}_m$ line segments are stored, where $1 \le m \le N_{\text{Seg}}$. $\hat{X}_k$, $\hat{Y}_k$ and $\hat{\phi}_k$ of the LRS at the k th scan step are estimated by matching the line segments $\boldsymbol{S}_m(1 \le m \le N_{\text{Seg}})$ with the global map $\hat{\boldsymbol{W}}_{k-1}$ at the $(k-1)$ steps given by Eq.(18) as,

$$\begin{cases} \hat{\boldsymbol{X}}_k = \hat{\boldsymbol{X}}_{k-1} + \boldsymbol{T}(\hat{\phi}_{k-1})\Delta\boldsymbol{X}\left(k \mid k-1\right) \\ \hat{\phi}_k = \hat{\phi}_{k-1} + \Delta\phi\left(k \mid k-1\right) \end{cases}, \tag{19}$$

by using the transform matrix $\boldsymbol{T}(\hat{\phi}_{k-1})$ and estimated position vector $\hat{\boldsymbol{X}}_{k-1}$ of Eq.(16).

Here, $\Delta\boldsymbol{X}(k \mid k-1)$ and $\Delta\phi(k \mid k-1)$ represent the displacements of the position and angle of the LRS from the $(k-1)$ th to the k th scan step.

In order to estimate $\Delta\boldsymbol{X}(k \mid k-1)$ and $\Delta\phi(k \mid k-1)$, first, the map $\hat{w}^*_{(k-1),i}$ on the coordinate system o $-xy$ fixed on the LRS is obtained by applying the translational and rotational transformation as,

$$\hat{w}^*_{(k-1),i} = \boldsymbol{T}\left(-\hat{\phi}_{k-1}\right)\left(\hat{w}_{(k-1),i} - \hat{\boldsymbol{X}}_{k-1}\right), \tag{20}$$

to all points $\hat{w}_{(k-1),i}$ in the estimated global map $\hat{\boldsymbol{W}}_{k-1}$ of Eq. (18).

Points $\hat{w}^*_{(k-1),j}$ within the LRS range are filtered from the map points $\hat{w}^*_{(k-1),i}$, and transformation of translational displacement $\Delta\boldsymbol{X}_q(k \mid k-1)$ and rotational displacement $\Delta\phi_q(k \mid k-1)$ are applied iteratively to all the points. At the qth iteration count, the points are defined as,

$$\hat{w}^q_{(k-1),j} = \boldsymbol{T}\left(-\Delta\phi_q\left(k \mid k-1\right)\right)\hat{w}^*_{(k-1),j} - \Delta\boldsymbol{X}_q\left(k \mid k-1\right). \tag{21}$$

In addition, the matrix, which is constructed by arranging the vector $\hat{w}^q_{(k-1),j}$ in a row, is defined as,

$$\hat{\boldsymbol{W}}^q_{k-1} = \left[\hat{w}^q_{(k-1),1} \quad \hat{w}^q_{(k-1),2} \quad \cdots \quad \hat{w}^q_{(k-1),N^*_{\text{map}}}\right]. \tag{22}$$

Here, the centre of gravity $\boldsymbol{G}^q_c$ of all point in the matrix $\hat{\boldsymbol{W}}^q_{k-1}$ can be obtained as,

$$\boldsymbol{G}^q_c = \frac{1}{N^*_{\text{map}}} \sum_{i=1}^{N^*_{\text{map}}} \hat{w}^q_{(k-1),i}. \tag{23}$$

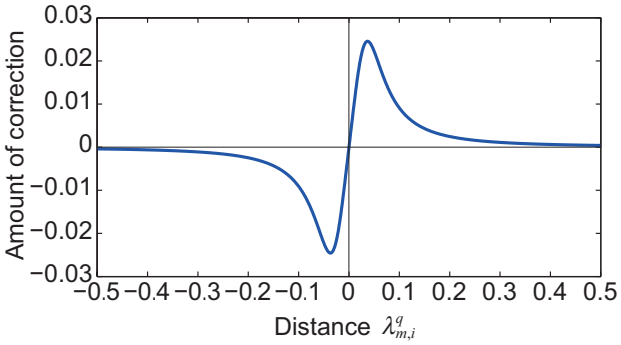


Figure 10. Example of $\delta \boldsymbol{f}^{\,q}_{m,i}$ ($\gamma_f = 10^{-4}$, $\varepsilon = 10^{-4}$, $\kappa = 3$)

In this study, the amount of correction of translational displacement generated by a line segment $\boldsymbol{S}_m$ for a point $\hat{w}^q_{(k-1),i}$ in $\hat{\boldsymbol{W}}^q_{k-1}$ is given by,

$$\delta \boldsymbol{f}^q_{m,i} = \begin{cases} \gamma_f \dfrac{\lambda^q_{m,i}}{\left|\lambda^q_{m,i}\right|^{\kappa} + \varepsilon} \boldsymbol{e}_{n,m} & \left(\lambda_{t(m,P_m)} \le \lambda^q_{t(m,i)} \le \lambda_{t(m,Q_m)}\right) \\ \boldsymbol{0} & \left(\lambda^q_{t(m,i)} < \lambda_{t(m,P_m)},\ \lambda_{t(m,Q_m)} < \lambda^q_{t(m,i)}\right) \end{cases}, \tag{24}$$

as shown in Fig.9, where the vector $\boldsymbol{e}_{n,m}$ represents a unit normal vector to line segment $\boldsymbol{S}_m$ expressed as,

$$\boldsymbol{e}_{n,m} = \begin{bmatrix} \cos\theta_m & \sin\theta_m \end{bmatrix}^{\mathrm{T}}, \tag{25}$$

and $\lambda^q_{m,i}$ represents the distance between the line segment $\boldsymbol{S}_m$ and the point $\hat{w}^q_{(k-1),i}$ expressed as,

$$\lambda^q_{m,i} = -\left(\boldsymbol{e}^{\mathrm{T}}_{n,m} \hat{w}^q_{(k-1),i} - \rho_m\right). \tag{26}$$

As shown in Fig.9, $\boldsymbol{R}^q_{m,i}$ is a point on segment $\boldsymbol{S}_m$ which is perpendicular from the point $\hat{w}^q_{(k-1),i}$. Moreover, $\lambda^q_{t(m,i)}$ in Fig. 9 represents the distance between start point $\boldsymbol{p}_k(P_m)$ of the segment $\boldsymbol{S}_m$ and point $\boldsymbol{R}^q_{m,i}$, which is given as,

$$\boldsymbol{R}^q_{m,i} = \hat{w}^q_{(k-1),i} + \lambda^q_{m,i} \boldsymbol{e}_{n,m}, \tag{27}$$

and the distance $\lambda^q_{t(m,i)}$ is given by,

$$\lambda^q_{t(m,i)} = \boldsymbol{e}^{\mathrm{T}}_{t,m}\left(\boldsymbol{R}^q_{m,i} - \boldsymbol{p}_k\left(P_m\right)\right). \tag{28}$$

Here, the vector $\boldsymbol{e}_{t,m}$ is the unit vector along the same direction as the segment $\boldsymbol{S}_m$ and expressed as,

$$\boldsymbol{e}_{t,m} = \begin{bmatrix} -\sin\theta_m & \cos\theta_m \end{bmatrix}^{\mathrm{T}}. \tag{29}$$

Similarly, it follows from Eq.(28) that the distance $\lambda_{t(m,Q_m)}$ from the start point P_m to the end point Q_m is given by,

$$\lambda_{t(m,Q_m)} = \boldsymbol{e}^{\mathrm{T}}_{t,m}\left(\boldsymbol{p}_k\left(Q_m\right) - \boldsymbol{p}_k\left(P_m\right)\right). \tag{30}$$

Obviously, the distance $\lambda_{t(m,P_m)}$ is zero, as it is the distance from the start point P_m to itself on the line segment $\boldsymbol{S}_m$ as,

$$\lambda_{t(m,P_m)} = \boldsymbol{e}^{\mathrm{T}}_{t,m}\left(\boldsymbol{p}_k\left(P_m\right) - \boldsymbol{p}_k\left(P_m\right)\right) = 0. \tag{31}$$

Therefore, the amount of correction of translational displacement $\delta \boldsymbol{f}^{\,q}_{m,i}$ in Eq.(24) becomes the normal vector to the line segment $\boldsymbol{S}_m$ when the point $\boldsymbol{R}^q_{m,i}$ exists on the segment $\boldsymbol{S}_m$, and becomes the zero vector when the point $\mathbf{R}^q_{m,i}$ does not exist on the segment. Note that in Eq.(24) the condition $(\lambda_{t(m,P_m)} \le \lambda^q_{t(m,i)} \le \lambda_{t(m,Q_m)})$ ensures that the point lies on the segment, and the condition $(\lambda^q_{t(m,i)} < \lambda_{t(m,P_m)},\ \lambda_{t(m,Q_m)} < \lambda^q_{t(m,i)})$ ensures that the point does not lie on the segment.

Equation (24) is introduced to simulate a force such as the gravitational or magnetic force which decreases rapidly with increasing distance between two objects, where γ_f, ε and κ are the parameters. Figure 10 is an example of $(\boldsymbol{e}^{\mathrm{T}}_{n,m} \delta \boldsymbol{f}^{\,q}_{m,i})$ with respect to $\lambda^q_{m,i}$ given by Eq.(24). It can be observed that the amount of correction closes to zero in order to decrease the influence on the amount of correction when the distance between a line segment and a point is increased, i.e., for points which are too far from the line. On the other hand, the amount of correction is almost linear within a region of a small distance.

The amount of correction of rotational displacement is defined as,

$$\delta M^q_{m,i} = \gamma_M \left(\hat{w}^q_{(k-1),i} - \boldsymbol{G}^q_c\right) \otimes \delta \boldsymbol{f}^{\,q}_{m,i}, \tag{32}$$

which is the moment around the centre of gravity $\boldsymbol{G}^q_c$ due to the amount of correction of translational displacement $\delta \boldsymbol{f}^{\,q}_{m,i}$. Here, operator $\otimes$ gives the norm of the vector given by the cross product of two vectors.

The amount of correction for the map $\hat{\boldsymbol{W}}^q_{k-1}$ is given by the average of all corrections from N_{Seg} segments $\boldsymbol{S}_m (1 \le m \le N_{\mathrm{Seg}})$ to each point of map $\hat{\boldsymbol{W}}^q_{k-1}$ as follows:

$$\boldsymbol{f}_q = \frac{1}{N_{\mathrm{Seg}} N^*_{\mathrm{map}}} \sum_{m=1}^{N_{\mathrm{Seg}}} \sum_{i=1}^{N^*_{\mathrm{map}}} \delta \boldsymbol{f}^q_{m,i},$$

$$M_q = \frac{1}{N_{\mathrm{Seg}} N^*_{\mathrm{map}}} \sum_{m=1}^{N_{\mathrm{Seg}}} \sum_{i=1}^{N^*_{\mathrm{map}}} \delta M^q_{m,i}. \tag{33}$$

Initially, at the start of the robot operation, we set $\Delta \boldsymbol{X}_1(k \mid k-1)=\mathbf{0}$ and $\Delta\phi_1(k \mid k-1)=0$. Later, $\Delta \boldsymbol{X}_q(k \mid k-1)$ and $\Delta\phi_q(k \mid k-1)$ are updated by the following equations,

$$\Delta \boldsymbol{X}_{q+1}(k \mid k-1) = \xi\, \Delta \boldsymbol{X}_q(k \mid k-1) - \eta\, \boldsymbol{f}_q - \zeta \sum_{i=1}^{q-1} \boldsymbol{f}_i \tag{34}$$

$$\Delta\phi_{q+1}(k \mid k-1) = \xi\, \Delta\phi_q(k \mid k-1) - \eta\, M_q - \zeta \sum_{i=1}^{q-1} M_i \tag{35}$$

where $\boldsymbol{f}_q$ and M_q are given by Eq.(33). Here, the speed of the convergence can be tuned by parameters ξ, η and ζ. The update is finished at the time when $|\boldsymbol{f}_q| < \rho_f$ and $|M_q| < \rho_M$ are satisfied, where ρ_f and ρ_M represent the thresholds of convergence.

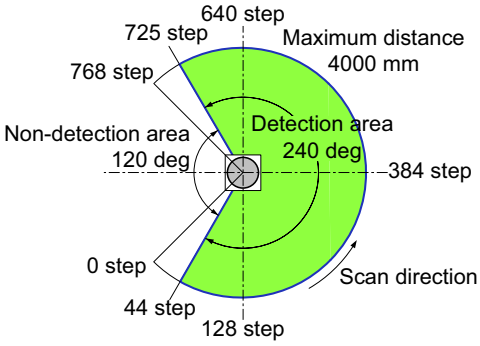
Finally, the estimated values of $\Delta \boldsymbol{X}(k \mid k-1)$ and $\Delta\phi(k \mid k-1)$ can be obtained by,

$$\Delta \boldsymbol{X}(k \mid k-1) = \Delta \boldsymbol{X}_{q+1}(k \mid k-1), \tag{36}$$

$$\Delta\phi(k \mid k-1) = \Delta\phi_{q+1}(k \mid k-1), \tag{37}$$

The calculations are executed recursively returning to Eq. (21). Moreover, self-localization can be achieved by Eq.(19). Simultaneous Localization And Mapping (SLAM) is also achieved by executing alternately the map updating in Subsection 4.1 and self-localization in Subsection 4.2.

(a) URG-04LX



(b) Measurement range

Figure 11. Laser range sensor URG-04LX (HOKUYO AUTOMATIC CO., LTD.) and measurement range

5. Results and Discussion

This section discusses the results in both simulation and real environments. For the real environment, we used the Pioneer-P3DX [38] robot shown in Fig.12. It is a two wheeled, differential drive robot. We first describe the motion model of the robot. The distance between the left and the right wheel is W_r, and the robot state at position P is given as $[x,y,\theta]$. From Fig.13, the turning angle β is calculated as,

$$\begin{aligned} r &= \beta \cdot (R + W_r), \\ l &= \beta \cdot R \\ \therefore \beta &= \frac{r-l}{W_r} \end{aligned} \tag{38}$$

and the radius of turn R as,

$$R = \frac{l}{\beta},\ \beta \neq 0. \tag{39}$$

The coordinates of the centre of rotation (C, in Fig.13), are calculated as,

$$\begin{bmatrix} Cx \\ Cy \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \left(R + \frac{W_r}{2} \right) \cdot \begin{bmatrix} \sin\theta \\ -\cos\theta \end{bmatrix} \tag{40}$$

The new heading θ' is,

$$\theta' = (\theta + \beta) \bmod 2\pi, \tag{41}$$

from which the coordinates of the new position P' are calculated as,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} Cx \\ Cy \end{bmatrix} - \left(R + \frac{W_r}{2} \right) \cdot \begin{bmatrix} \sin\theta' \\ -\cos\theta' \end{bmatrix},\ \beta \neq 0,\ r \neq l. \tag{42}$$

If $r=l$, i.e., if the robot motion is straight, the state parameters are given as,

$$\theta' = \theta, \tag{43}$$

and,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + l \cdot \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, (l = r). \tag{44}$$

The flowchart of mapping and localizing process integrated with the proposed line-extraction method is shown in Fig.14. For each LRS scan at the k th step, the line segments

Laser-range sensor	Simulated model of URG-04LX
Distance range	60 mm ~ 4000 mm
Accuracy	20 ~ 1000 mm : ± 25 mm
	1000 ~ 4000 mm : ± 2.5 %
Distance resolution	3.84 mm ((4000-60) mm / 1024)
Scan angle	240 deg
Angular resolution	1.875 deg (240 deg / 128)

Table 1. Laser-range sensor for simulation

are extracted and shorter segments are removed using the algorithm in Subsection 3.2. For the very first scan, the true and estimated map is the same. For subsequent scans, the global map $\hat{\boldsymbol{W}}_{k-1}$ at the $(k-1)$ th step given by Eq.(18) is extracted and the detected line segments are matched against the map to estimate the position $\Delta \boldsymbol{X}(k \mid k-1)$ (by Eq. 34) and angle $\Delta\phi(k \mid k-1)$ (by Eq.35). Based on this estimation, the map is updated and the process is repeated for the LRS scan at the $(k+1)$ th step.

5.1 Simulation results

We performed a simulation based on a URG-04LX (HOKUYO AUTOMATIC) laser-range sensor (Fig.11), the specifications of which have been summarized in Table 1. Figure 11(b) shows the measurement range of the LRS. Gaussian noise values of ±25 mm and ±2.5% were added to the distance ranges of 1000 mm and 4000 mm, respectively. The effects of errors caused by the shape of objects due to reflection etc. were not taken into consideration. Compared to the real LRS, the angular resolution was set to 1/8 for faster computation. The simulation was carried out using MATLAB 7.12 (R2011a) running on a Linux machine with an Intel Core i7-4500U CPU, 1.80 GHz and 16GB RAM.

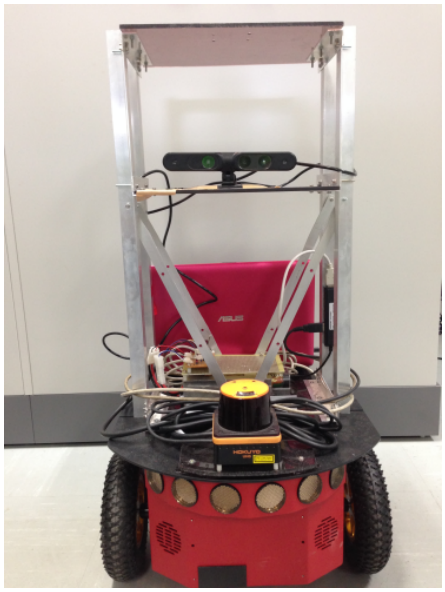


Figure 12. Pioneer P3-DX

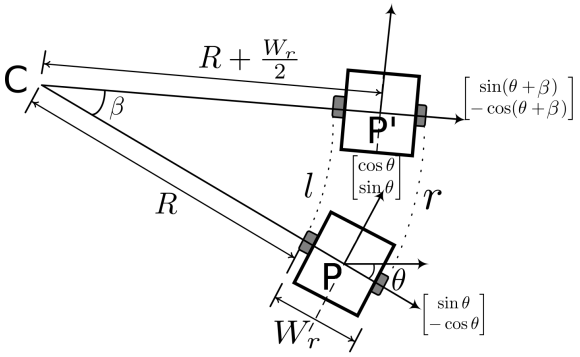


Figure 13. Motion model of two wheel, differential drive robots

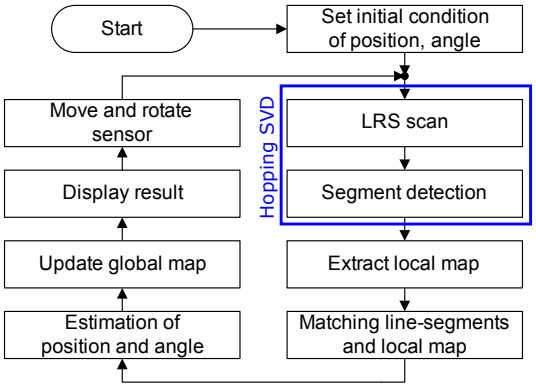


Figure 14. Flow chart of the SLAM process integrated with the proposed line-segment detection. The marked portion shows the proposed line extraction accelerated by hopping.

Taking the distance resolution of the sensor into consideration (Table 1), the value of threshold σ_{Th} in the line-detection algorithm (Fig.5) was set to 25/1024. The map of the sensor environment used for self positioning was constructed by randomly extracting 75% of points from the world map that are within the sensor's range. The values of parameters in the correction Eq.(24) were set to $\kappa=3$, $\varepsilon=0.0001$, $\gamma_f=0.001$, and the parameter γ_M in Eq.(32) was set to $\gamma_M=10$. The convergence parameters of Eqs.(34) and (35) were set to $\xi=0.99$, $\eta=1$, $\zeta=0.01$, and the parameters of the convergence condition were set to $\rho_f=10^{-5}$, $\rho_M=10^{-5}$. The values of these parameters were chosen as a trade-off between computation time and accuracy of localization, and determined by experimental trials. The hopping factor ψ was set to a small value of one.

An example of a map-matching result in the sensor's o $-xy$ coordinate is shown in Fig.15. The results of simultaneously applying mapping and localization by the proposed method while moving the sensor in the upward direction for around six metres is shown in Fig.16. Estimation is performed for each point and the map is updated.

The starting point of the extracted line is shown by □, and the end point by ○ in Fig.15 and Fig.16. The final constructed map is shown by the small '+' mark in pink. The sensor location is shown by a larger '+' sign in pink and, similarly, the sensor's orientation is shown by an arrow.

Both accurate mapping and accurate robot localization were achieved with the proposed method. The total execution time was 1015 seconds, which included the calculations and displaying of graphs. It can be seen that setting the threshold σ_{Th} to a very small value caused many smaller line segments to be extracted.

Figure 17, Fig.18 and Fig.19 represent the estimation errors in the X-axis, Y-axis and sensor angle ϕ, respectively. Figure 20 represents the number of iterations q in Eqs.(34) and (35) at step k. For a complex environment, the upper limit for the total number of iterations was set to 1000. It can still be seen that map construction and localization was done accurately. From Fig.20 it can be seen that although there are steps where the total iterations were limited to 1000, on an average, the iteration count was limited to 200 iterations and fewer than 500 iterations. An angular estimation error of 0.5 deg is shown in Fig.19. Similarly, from Fig.17 and Fig.18, the error was limited to 5 cm for a 6 m run. The accuracy can be further improved by lowering the values of threshold ρ_f and ρ_M, at the cost of the efficiency of the SLAM process.

We tested the proposed hopping point SVD-based line extraction algorithm for a large set of almost linear 2280 points generated by URG-04LX in three subsequent scans (760 × 3=2280, 760 points per scan) against different values of hopping factor ψ. As shown in Fig.5, the proposed algorithm applies SVD to data generated from the starting point of each detected segment to its end point, after which the process is repeated from the beginning for the new line. The worst-case scenario for the algorithm is when all of the 2280 points are almost linear, and this has been considered for calculating the speed-up. The execution time for different values of ψ is summarized in Table 2. It can be seen that the speed-up is nearly linear.

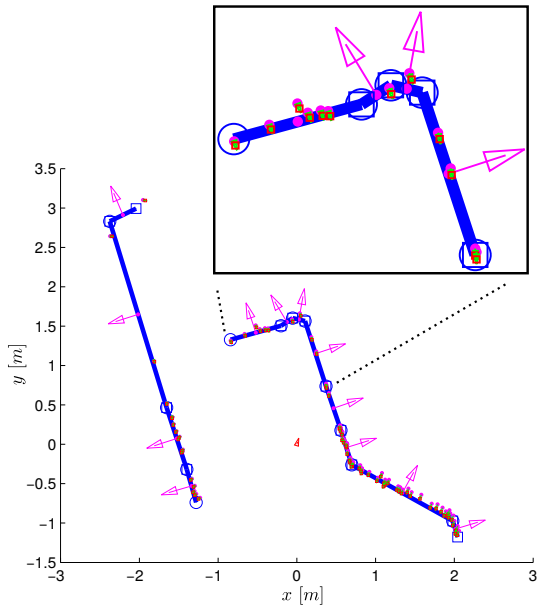


Figure 15. Simulation results of the estimation of sensor position with map matching

Parameter	ψ=0	ψ=1	ψ=2	ψ=3	ψ=4	ψ=5	ψ=6
Time (s)	28.36	14.90	10.01	7.68	6.08	5.09	5.25
Speed up	1.00	1.90	2.83	3.68	4.66	5.56	5.40

Table 2. Execution time for line detection using hopping point SVD on 2280 (760 × 3) points and the speed-up for various values of Ψ

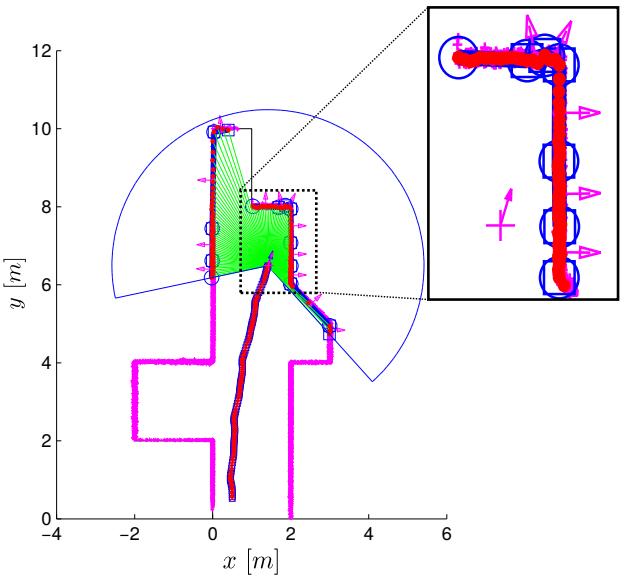


Figure 16. Results of the simulation of simultaneous localization and mapping

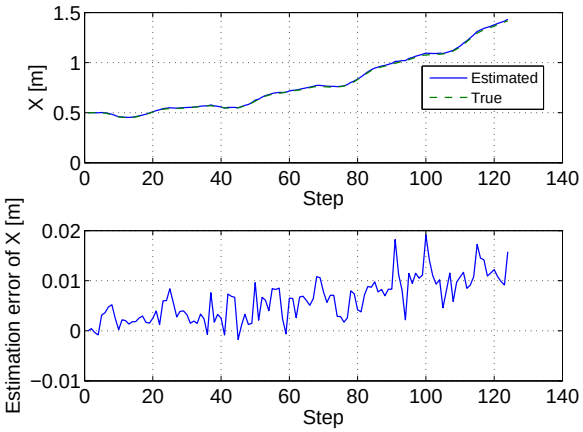


Figure 17. Estimation error in the X-axis

5.2 Experiments in a real environment

The robot P3-DX (Fig.12) was used in the experiment and the motion model was explained earlier. The robot was equipped with the same sensor used in the simulation, moved in the environment shown in Fig.21(a) and data were collected. The environment is one of the corridors of our university, cluttered with real objects such as boxes and tables. The corridor environment shown in Fig.21(c) is approximately 4 m in breadth and 35 m in length.

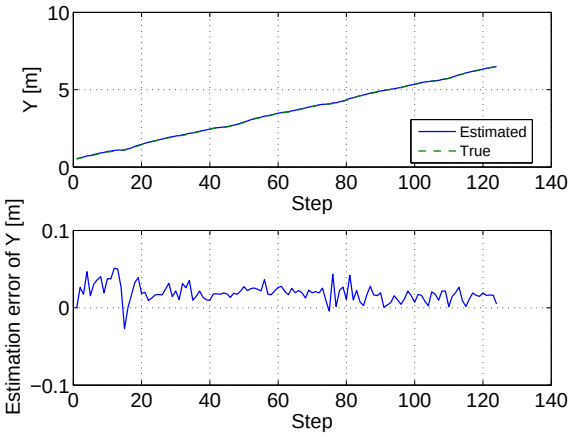


Figure 18. Estimation error in the Y-axis

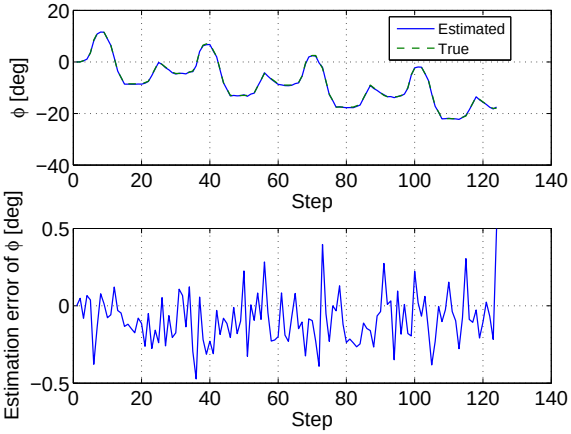


Figure 19. Estimation error of the sensor angle ϕ

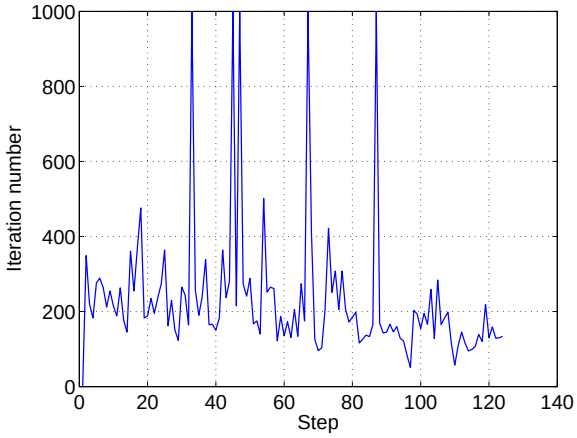


Figure 20. Results of the simulation of the iteration number for estimation of the sensor's position

Figure 21(b) shows the dead reckoning of the collected sensor data. Figure 21(c) shows the final map constructed by the proposed method. The line segments are indicated by a bold black line on the map. Many line segments are detected, and the starting point of each line segment is indicated by a blue □, and the end point by red ○ in Fig. 21(c). Note that when the end point of a line segment is the starting point of the new line segment, the blue □ and the ○ overlap each other. Some minor segments were also detected by the robot in the map as the environment was cluttered. When the objects are too far from the LRS and not enough data are available, a bunch of very small line segments which are not joined to each other can also be seen. However, this can be corrected by moving the robot sufficiently close to the far away objects and by collecting enough data. A line map was obtained by the proposed method with a runtime of 1453.2 seconds. The bulk of computation (1302 s ≈ 89%) was consumed by map updating and localization with a large number of iterations for the convergence of Eq.(34) and Eq.(35). As explained in Section 4.2, this convergence can be tuned by parameters ξ, η and ζ. Although not realized in the current work, the proposed line-extraction algorithm has a scope of parallelization with different CPU threads working on different sections of line segments that can later be joined.

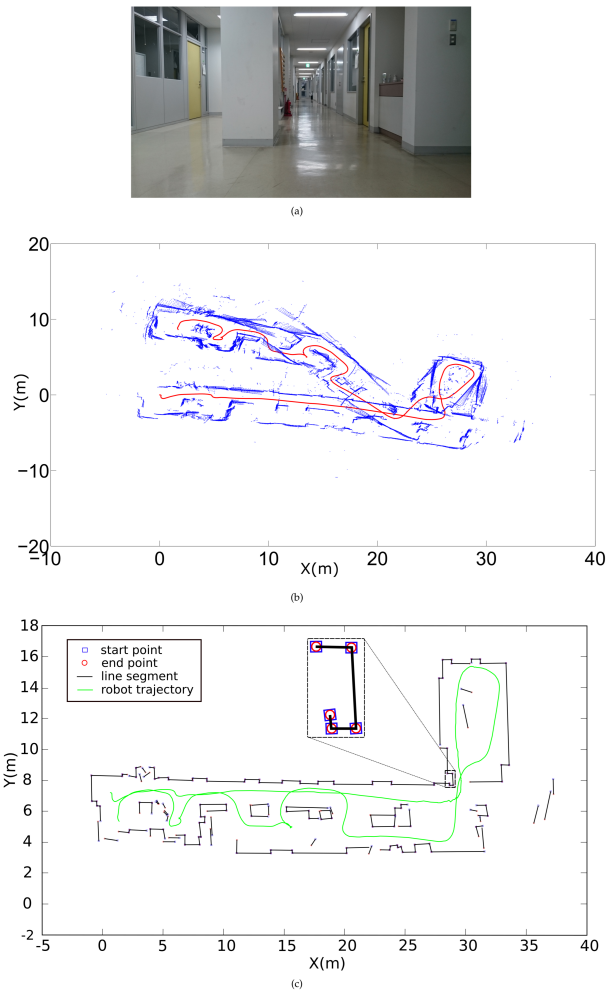


Figure 21. (a) Corridor environment for experiment. (b) Dead reckoning map with trajectory in red. (c) Corrected line map with robot trajectory in green.

6. Conclusion

While there exists a large plethora of literature on building maps using traditional techniques like the Extended Kalman Filter, and other probabilistic methods which have been successfully demonstrated in various scenarios, this paper took a fresh approach in using SVD and Hough transform for line detection from LRS data to build maps. We accelerated the algorithm using a novel hopping-points method in which SVD is applied to intermittent points governed by a hopping parameter Ψ. A reverse hop mechanism ensures that the end points of the line segment are detected accurately. We integrated the proposed line-detection method within a SLAM framework and showed how lines detected using the proposed method can be used as landmarks for the robot to localize itself in the map and build a map of the environment at the same time. By appropriately setting various parameter values, the proposed algorithm gives users the flexibility to obtain maps with desired accuracy and speed. We tested the proposed method in both simulation and real environments, and found that the proposed method can generate accurate maps with fewer errors. Although the parameters of the proposed method need to be determined experimentally, once they have been determined for a given sensor's characteristics, the parameters actually help in tuning the convergence speed and accuracy of mapping and localization. The proposed algorithm has a scope of parallelism and different line segments can be detected separately and later joined in the single map. Our future work deals in exploiting this parallelism and testing the algorithm in more complex and noisy environments. A robust comparison with other line-segment detection techniques is also currently under work.

7. Acknowledgements

This work is supported by MEXT (Ministry of Education, Culture, Sports, Science and Technology), Japan. We are thankful to the anonymous reviewers for giving us important insights to improve the manuscript.

8. References

[1] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainly. *Int. J. Rob. Res.*, 5(4):56–68, December 1986.

[2] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[3] A.J. Davison, I.D. Reid, N.D. Molton, and O. Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, June 2007.

[4] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard. Real-time 3d visual slam with a hand-held camera. In *Proc. of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, Vasteras, Sweden, April 2011.

[5] Ingemar J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *Robotics and Automation, IEEE Transactions on*, 7(2): 193–204, Apr 1991.

[6] Ingemar J. Cox. Blanche: an autonomous robot vehicle for structured environments. In *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*, pages 978–982 vol.2, Apr 1988.

[7] Ingemar J. Cox. Blanche: Position estimation for an autonomous robot vehicle. In *Intelligent Robots and Systems '89. The Autonomous Mobile Robots and Its Applications. IROS '89. Proceedings., IEEE/RSJ International Workshop on*, pages 432–439, Sep 1989.

[8] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[9] Feng Lu and Evangelos Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18(3):249–275.

[10] J. Minguez, F. Lamiraux, and L. Montesano. Metric-based scan matching algorithms for mobile robot displacement estimation. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3557–3563, April 2005.

[11] Andrea Censi. An icp variant using a point-to-line metric. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 19–25, May 2008.

[12] S.T. Pfister, K.L. Kriechbaum, S.I. Roumeliotis, and J.W. Burdick. Weighted range sensor matching algorithms for mobile robot displacement estimation. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1667–1674 vol.2, 2002.

[13] Esther Ezra, Micha Sharir, and Alon Efrat. On the performance of the icp algorithm. *Comput. Geom. Theory Appl.*, 41(1-2):77–93, October 2008.

[14] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[15] M. Greenspan and M. Yurick. Approximate k-d tree search for efficient icp. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, pages 442–448, Oct 2003.

[16] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[17] Geovany Araujo Borges and Marie-José Aldon. Line extraction in 2d range images for mobile robotics. *J. Intell. Robotics Syst.*, 40:267–297, July 2004.

[18] M.D. Adams and A. Kerstens. Tracking naturally occurring indoor features in 2-d and 3-d with lidar range/ amplitude data. *The International Journal of Robotics Research*, 17(9):907–923, 1998.

[19] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, January 1972.

[20] E. R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[21] J. Forsberg, U. Larsson, and A. Wernersson. Mobile robot navigation using the range-weighted hough transform. *Robotics Automation Magazine, IEEE*, 2(1): 18–26, Mar 1995.

[22] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[23] Daniel Sack and Wolfram Burgard. A comparison of methods for line extraction from range data. In *In Proc. of the 5th IFAC Symposium on Intelligent Autonomous Vehicles (IAV)*, 2004.

[24] Samuel T. Pfister, Stergios I. Roumeliotis, and Joel W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *In International Conference Robotics and Automation, ICRA*, pages 14–19, 2003.

[25] Kai Oliver Arras and Roland Y. Siegwart. Feature extraction and scene interpretation for map-based navigation and map building. In *Proc. of SPIE, Mobile Robotics XII*, pages 42–53, 1997.

[26] Ankit A. Ravankar, Yohei Hoshino, Abhijeet Ravankar, Lv Jixin, Takanori Emaru, and Yukinori Kobayashi. Algorithms and a framework for indoor robot mapping in a noisy environment using clustering in spatial and hough domains. *International Journal of Advanced Robotic Systems*, 12, 2015.

[27] A. Ravankar, Y. Kobayashi, A. Ravankar, and T. Emaru. A connected component labeling algorithm for sparse lidar data segmentation. In *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*, pages 437–442, Feb 2015.

[28] A. A. Ravankar, Y. Hoshino, T. Emaru, and Y. Kobayashi. Map building from laser range sensor information using mixed data clustering and singular value decomposition in noisy environment. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 1232–1238, Dec 2011.

[29] J. Vandorpe, H. Van Brussel, and H. Xu. Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2d range finder. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 1, pages 901–908 vol.1, Apr 1996.

[30] S.T. Pfister, S.I. Roumeliotis, and J.W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 1304–1311 vol.1, Sept 2003.

[31] A. Garulli, A. Giannitrapani, A. Rossi, and A. Vicino. Mobile robot slam for line-based environment representation. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 2041–2046, Dec 2005.

[32] Su-Yong An, Jeong-Gwan Kang, Lae-Kyoung Lee, and Se young Oh. Slam with salient line feature extraction in indoor environments. In *Control Automation Robotics Vision (ICARCV), 2010 11th International Conference on*, pages 410–416, Dec 2010.

[33] Su-Yong An, Jeong-Gwan Kang, Lae-Kyoung Lee, and Se-Young Oh. Line segment-based indoor mapping with salient line feature extraction. *Advanced Robotics*, 26(5-6):437–460, 2012.

[34] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart. A comparison of line extraction algorithms using 2d laser rangefinder for indoor mobile robotics. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1929–1934, Aug 2005.

[35] P.V.C Hough. Method and means for recognizing complex patterns, us patents 3069654, 1962.

[36] Wikipedia. Ransac. 2016. Available at https://en.wikipedia.org/wiki/RANSAC [Accessed on 25-03-2016].

[37] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231. AAAI Press, 1996.

[38] Pioneer P3-DX. Pioneer p3-dx robot. 2016. Available at http://www.mobilerobots.com/Mobile_Robots.aspx [Accessed on 23-01-2016].