Development of Adaptive Line Tracking Breakpoint Detection Algorithm for Room Sensing using LiDAR Sensor

Deddy El Amin, Karlisa Priandana, Medria Kusuma Dewi Hardhienata
Computer Science Department, Bogor Agricultural University (IPB University), Bogor, Indonesia

***Abstract*—This research focuses on the use of Light Detection and Ranging (LiDAR) sensors for robot localization. One of the most essential algorithms in LiDAR localization is the breakpoint detector algorithm which is used to determine the corner of the room. The previously developed breakpoint detection methods have weaknesses, such as the Adaptive Breakpoint Detector (ABD), could generate dynamic threshold values. The ABD results, on the other hand, still require Line Extraction to obtain the corner breakpoint. Line Extraction method, e.g. Iterative End Point Fit (IEPF), is used to categorize data, resulting in the generation of a line pattern as an interpretation of a wall. The computational method for obtaining the corner breakpoint becomes longer as the line is extracted. To address this issue, our algorithm proposes a new threshold area in the form of an ellipse with the threshold value parameter obtained from previously identified room size and sensor characteristics. As a result the corner breakpoint detection becomes more adaptive. The goal of this research is to create an Adaptive Line Tracking Breakpoint Detector (ALTBD) approach that will reduce the computing time required to detect corner breakpoints. Furthermore, the Line Extraction method required for corner breakpoint detection is modified in the ALTBD. To distinguish between the edge of the wall and the corner of the room, the boundary value is increased. The ALTBD method was tested in a simulation arena comprised of multiple rooms and halls. According to the results, the ALTBD computation time is faster in detecting corner breakpoints than the ABD IEPF method, also the accuracy for determining the position of the robot was improved.**

Keywords*—LiDAR; breakpoint detector; robot localization; corner detection; line segmentation

I. Introduction

Robot localization which is a method to acquire information about a robot's direction and position in its working environment is required by every autonomous robot in order to move and accomplish its task. Three types of robot localization are global localization, predictive localization, and local localization [1]. When the robot is outside, global localization works relatively well [2]. Whenever the robot is indoors, it uses probabilistic localization, local localization, or a combination of the two [3]. Our research leads to probabilistic localization, which determines the robot's position and orientation based on sensor measurement data combined with prior knowledge in the form of a map of the arena in the room.

Several papers have been written about the sensors used in localization and probabilistic localization methods, including encoder sensors [4][5], magnetic compass [6], magnetic anomalies map [7], low-cost gyro [8], ultrasonic sensors [9] [10], RFID [11] to the Light Distance and Ranging (LiDAR) sensor. Sensors such as encoder disc sensors, magnetic compasses, low-cost gyros, and ultrasonic sensors require significant movement before the robot can identify its position. If time is of the essence in accomplishing the objective of the robot, the sensor is not the first option to be installed in the robot. As a result, this research utilizes a LiDAR sensor capable of identifying the position and surroundings without requiring the robot to move, improving the robot's movement more effective.

Previous research has widely proposed the LiDAR sensor since it has excellent spatial accuracy, fast data renewal, and does not rely on object illumination or reflectance [12]. This research focuses on determining the form of the room using the LiDAR sensor measurement results. The issue is determining how to identify the geometry of the space based on its corner position. Breakpoint detection is one way for determining the corner of a room. This method detects the room's corner points and the edge of the wall by verifying the discontinuity between the two points scanned consecutively by the LiDAR sensor [13].

Breakpoint detection method that combines Successive Edge Following (SEF), Line Tracking (LT), and Iterative End Point Fit (IEPF) was developed by Siadat et al. [14]. However, all three methods have a weakness: finding a consistent threshold value to detect a breakpoint between two consecutive sensor scan points is challenging. The observed breakpoints can take the form of room corners or the edge of a wall. The breakpoint detection threshold value should be dynamic, based on the distance between the sensor and the object being scanned [13].

Several researchers have indicated determining dynamic threshold values. The threshold value in Lee et al. [15] research is based on the current and prior measurement distance values. The DIET method, which takes the measurement angle into account when establishing the threshold value, was developed by Dietmayer et al. [16]. The method by including the virtual wall angle parameter as an estimate of the detected wall slope angle was improved by Santos et al. [17]. The threshold value in their Adaptive Breakpoint Detector (ABD) method as well developed by Borges and Aldon [13]. They use auxiliary angles to determine the point of intersection and use it as a reference for the threshold value, and Certad et al. [18] also

adjusted the threshold value in the ABD. They modify the measurement distance value in the ABD equation with the shortest distance value from either the current or previous distance. In his research on the ABD method, Su Young et al. [19] identified the lack of a line cluster perpendicular to the sensor as a problem. The challenge was then solved by combining the ABD and LT methods to create the Dual Breakpoint Detector (DBD), and Weerakoon et al. [20] also used ABD in their research. In order to increase the feature extraction rate value, Weerakoon et al. [20] research suggests taking the measurement error value and the longest distance value into consideration. The corner breakpoint was determined in three phases in this investigation. The first stage is ABD, which is used to segment the data depending on a threshold value. The following stage is Line Extraction with IEPF. Line Extraction is a method for organizing data based on line patterns that can be formed as an interpretation of a wall. The intersection of the two lines obtained by the IEPF two-segment linear regression is then used to locate the corner breakpoint in the third phase. Dingyao et al. [21] research's to determine the corner breakpoint included one more phase into four phases, i.e. point feature matching to compare angles and distances. The second, third and fourth stages of Corner Breakpoint Detection render the algorithm's computational process worthless.

To overcome these issues, The Adaptive Line Tracking Breakpoint Detector (ALTBD) method is developed in this research. The ALTBD method is a variation of the Line Tracking (LT) and Adaptive Breakpoint Detector (ABD) algorithms that introduces a new threshold area to improve corner breakpoint detection. By using established room sizes, this method also solves the challenge of finding the threshold. The breakpoint detection threshold value is made dynamic in this method by taking into consideration the difference between the sensor measurement distance and the projected virtual wall distance. Furthermore, the Line Extraction method required for corner breakpoint identification is improved in the ALTBD to reduce computational time. To distinguish between the edge of the wall and the corner of the room, the boundary value is increased.

This paper outline is as follows: The next section reviews some relevant works on LT and ABD method. Section III defines the proposed adaptive line tracking breakpoint detector method. Section IV provides the methodology that used to in this research, and Section V shows the results and analysis, whilst the conclusion is presented in the final section.

II. Related Work

In this section, several methods are discussed as the basis for the proposed method, namely Line Tracking (LT) and Adaptive Breakpoint Detector (ABD) method.

A. Line Tracking Method

Line tracking (LT) is a method of detecting breakpoints or segmenting line data that use linear regression and works in Cartesian coordinates. The following pseudo code [22] describes the working principle of this method:

1) Begin with two points of measurement, then draw a line between them.

2) Insert the next point to make a new line model.

3) Reprocess the line's parameters.

4) If the result of the generated line is satisfactory, continue (repeat to step 2).

5) If the result is not satisfactory, make the last point the line segment's end point, re-process the line's parameters, and make it as one line segment.

6) Go back to step 2 after completing the next two points.

The difficulty in defining the threshold value (D_{max}) to determine the breakpoint between the two measures is this LT's disadvantage [13]. Fig. 1 depicts two segments of data generated by LT when $d_5 > D_{max.}$

B. Adaptive Breakpoint Detector Method

The Adaptive Breakpoint Detector (ABD) method to address the issue of a fixed threshold value at LT was proposed by Borges and Aldon [13]. This method performs on the same principles as the LT method, which utilizes a threshold to detect breakpoints. The distinction is that there is no linear regression in this method, and the threshold value is made dynamic by utilizing the angle difference between the two measurement results. The ABD method is illustrated in Fig. 2 along with the parameters utilized. In ABD, breakpoints are detected by creating a border circle with a radius of D_{max} and a center at p_{n-1}. A breakpoint is detected if the current scan point (p_n) is found to be outside the circle. As a result, the previous scan point (p_{n-1}) is considered the end point of the current segment, whereas p_n is considered the beginning point of the following segment.

The adaptive breakpoint detector (ADB) method has the advantage of having a threshold value (D_{max}) that can change depending on the previously created distance (r_{n-1}) and a difference in the angle of measurement ($\Delta\phi$). This value is used as a comparison (border value) to distinguish between two segments of line data.

Equation (1) [13] describes the data terms stated as breakpoints:

$$\text{if } \|p_n - p_{n-1}\| > D_{max} \text{ then } k^b_n := \text{TRUE and } k^b_{n-1} := \text{TRUE} \quad (1)$$

Where, $\|p_n - p_{n-1}\|$ represents the distance between the p_n point and the p_{n-1} point. D_{max} is the breakpoint threshold whose value is determined by the sensor's distance to a measurable object (r_n). The current measured position is p_n, while the previous measured point is p_{n-1}. Fig. 2 depicts this ADB method. Detection is accomplished by drawing a dividing circle with the center point at p_{n-1} and a radius of D_{max}.

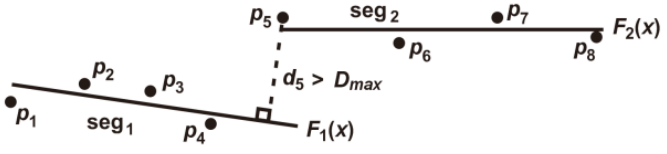

Fig. 1. Line Tracking Method [14].

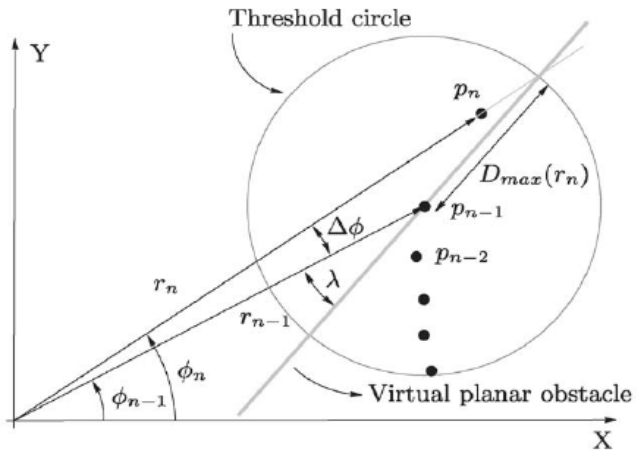

Fig. 2. Adaptive Breakpoint Detector Method [4].

If the current scan point (p_n) is found to be outside the circle, p_{n-1} is considered the endpoint of the current segment and p_n is determined the start point of the following segment. Equation (2) [13] describes its mathematical function:

$$\|p^h{}_n - p_{n-1}\| = r_{n-1}.\,(\sin(\Delta\phi)/\sin(\lambda\text{-}\Delta\phi)) \quad (2)$$

$\|p^h{}_n - p_{n-1}\|$ as the breakpoint testing threshold value Equations (3) and (4) are provided a real implementation of the threshold formula with [13]:

$$D_{max} = \|p^h{}_n - p_{n-1}\| + 3\sigma_r \quad (3)$$

$$D_{max} = r_{n-1}.\,(\sin(\Delta\phi)/\sin(\lambda\text{-}\Delta\phi)) + 3\sigma_r \quad (4)$$

Where, σ_r is the laser scanner manufacturer's parameter and λ the angle between the virtual line and the line connecting the object point to the laser scanner (r_{n-1}).

1) Line *extraction:* The ABD method generates breakpoints only at the ends of segments, which we refer to as terminal breakpoints in our research. As a result, another algorithm, Line Extraction, is required to reprocess existing data into new segments that can represent a line. As a result, the ABD computation time exceeds the LT in determining the angular breakpoint. Linear Regression [23], Split and Merge algorithm [24], Iteration End Point Fit [25], RANSAC algorithm [26], Hough Transform algorithm and Expectation-Maximization algorithm are some line extraction algorithms that can be used [27]. The Iteration End Point Fit (IEPF) method is discussed in this chapter as a comparison to the new method developed.

The IEPF algorithm [25] is a recursive Line Extraction method. Data checks are performed repeatedly for each segment base due to its recursive nature. The IEPF method is depicted in Fig. 3 by truncating one segment of data resulting from ABD into three data segments that form three lines. First, the initial data (p_1) and final data (p_n) are connected to form a straight line. The distance between the second data and the line is then calculated orthogonally. This is repeated for the third data point, and so on until all members of the segment have been checked. The maximum orthogonal distance (d_i^m) of the result will be compared to the limit value (d_{thd}). The limit value is the comparison value that was declared at the beginning. If the maximum value (d_i^m) is greater than the limit value, the segment will be halved at the maximum distance point. Fig. 9 illustrates how the first cut of a segment ($Z_1 = \{p_1,..., p_n\}$) into two segments ($Z_2 = \{p_1,..., p_i\}$ and $Z_3 = \{p_i,..., p_n\}$) occurs at the p_i point. The initial data (p_1) and final data (p_i) of the Z_2 segment are then connected to form a straight line. The distance between all of the data points in the segment Z_2 is then calculated orthogonally to the line. The result's maximum orthogonal distance (d_h^m) will be recompromised by the limit value (d_{thd}). One line segment has been checked because the maximum orthogonal distance (d_h^m) is less than the limit value (d_{thd}). The segment is a Z_2 made up of p_1 to p_i. The Z_3 segment is treated similarly. However, the maximum orthogonal distance (d_j^m) in the Z_3 segment $\{(p_i,..., p_n)\}$ is greater than the limit value (d_{thd}), so the segment must be cut in half with the intersection point at the p_j. The maximum orthogonal distance between the segments $\{p_i,..., p_j\}$ and $\{p_j,..., p_n\}$ is then calculated and compared to the limit value (d_{thd}). The data grouping on the Z_1 segment is complete because the maximum orthogonal distance of each segment is less than the limit value (d_{thd}). Through IEPF on the Z_1 segment, the final result is three line segments, namely $S_1\{p_1,..., p_i\}$, $S_2\{p_i,..., p_j\}$, and $S_3\{p_j,..., p_n\}$.

Weerakoon et al. [20] research's describes examples of ABD and Line Extraction implementation. The ABD threshold value equation was modified to Equation (5) in the research [20].

$$D_{max} = min\{r_n, r_{n+1}\}(1 - 3\sigma_r)(\sin(\Delta\phi)/\sin(\lambda\text{-}\Delta\phi)) + 3\sigma_r.max\{r_n, r_{n+1}\} \quad (5)$$

The ABD method has a disadvantage in that the angle value generating the threshold value is obtained based on the results of the experiment and requires Line Extraction to detect the corner breakpoint.

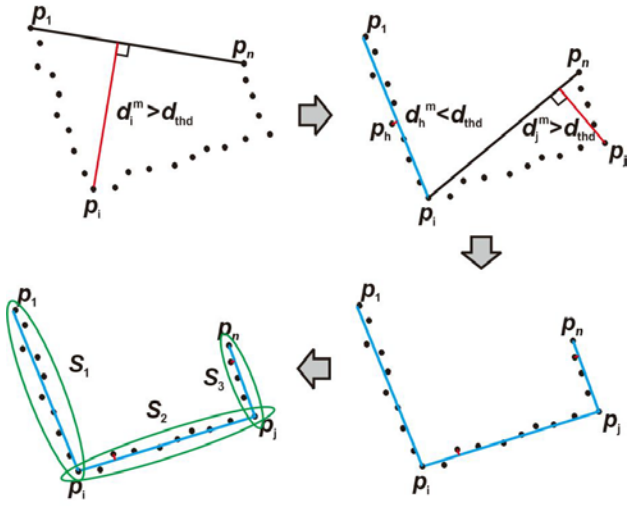

Fig. 3. IEPF Algorithm Working Principle.

III. The Proposed Adaptive Line Tracking Breakpoint Detector Method

A. Propose a New Method (Adaptive Line Tracking Breakpoint Detector)

The Adaptive Line Tracking Breakpoint Detector (ALTBD) method we propose is a combination of the Line tracking (LT) and Adaptive Breakpoint Detector (ABD) methods. The existing breakpoint detector (ABD) obtains coverage in the form of a circle with a threshold value diameter, whereas ALTBD obtains coverage in the form of an elliptical area with an elliptical center point from the prediction point. The LT method calculation yielded the prediction point. Although elliptical shapes require more complicated equations, they depict a point spread rather than a circle. The threshold value on ABD is determined by the virtual angle derived from the experiment, whereas in ALTBD, the parameter producing an elliptical area, which is a minor value, is made adaptable by taking the measurement error of the LiDAR sensor into consideration.

The linear equation $F'_n(x)$ generated by LT is used to predict the next point of measurement, as shown in Fig. 4. The point's polar angle value (θ_{n+1}) is then used as an input to calculate the equation of the line $F_{n+1}(x)$. The predictive point p'_{n+1} is then obtained by intersecting the two equations of the line. The prediction point p'_{n+1} is defined as a point (0,0) or the center point of the elliptical area, with the x-axis representing as the major axis and the y-axis representing as the minor axis. The length of the major axis ($qmax_{n+1}$) is determined by the distance between the points p_n and the prediction line $F'_n(x)$ at the point of intersection. Minor axis length ($hmax_{n+1}$) is determined by calculating using orthogonal regression against the prediction line and the error values at the previous points. Fig. 4 and 5 depict the parameters used in the ALTBD method. The ALTBD algorithm's flow is described in detail:

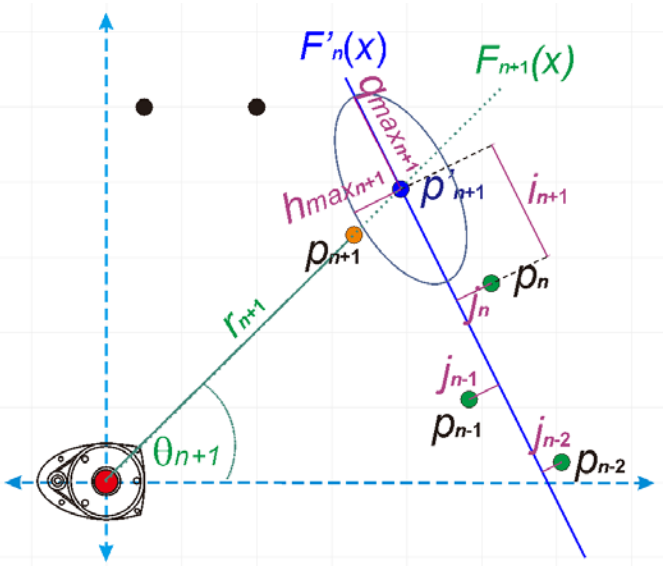

Fig. 4. ALTBD Method.

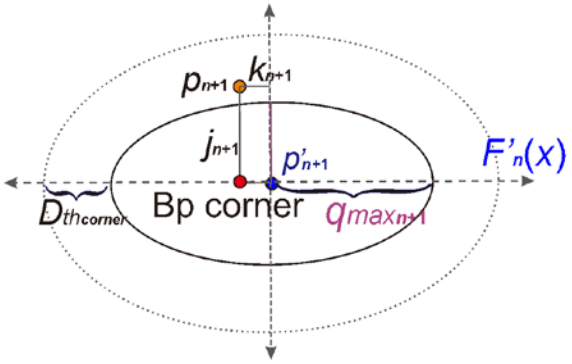

Fig. 5. Corner Detection ALTBD.

Algorithm for Detecting Breakpoints in Adaptive Line Tracking:

1) Determine the linear regression equation from three starting points (p_{n-2}, p_{n-1}, p_n) and transform it into a predictive line ($F'_n(x)$).
2) Determine the difference between the starting point (j_{n-2}, j_{n-1}, j_n) and the predicted line ($F'_n(x)$).
3) Using the equation $\boldsymbol{\varepsilon}_n = |j_n|/r_n$, calculate the percentage error ($\boldsymbol{\varepsilon}_{n-2}$, $\boldsymbol{\varepsilon}_{n-1}$, $\boldsymbol{\varepsilon}_n$) relative to the distance to the sensor (r_n).
4) Determine the average error percentage, $\sigma_n = (\Sigma^n_{i=n-2}\boldsymbol{\varepsilon}_i)/3$
5) Determine the minor axis limit, $hmax_{n+1} = \sigma_n + \sigma_{offset}$.
6) Verify out the next angle data (θ_{n+1}).
7) Create an equation using the angle gradient (θ_{n+1}) and the line ($F_{n+1}(x)$) that passes through the point (0,0).
8) Make the intersection point of the prediction line ($F'_n(x)$) and the scan line ($F_{n+1}(x)$) the prediction point (p'_{n+1}).
9) Calculate the distance between the prediction point (p'_{n+1}) and the last scan point (p_n) parallel to the prediction line ($F'_n(x)$), and set it as the major axis limit ($qmax_{n+1}$).
10) Read the next data set (p_{n+1}).
11) Transform the predicted point (p_{n+1}) to new coordinates, with the predicted point (p'_{n+1}) as the origin (0,0) and the predicted line ($F'_n(x)$) as the x-axis.
12) Calculate the new position of the point (p_{n+1}), and update x by k_{n+1} and y by j_{n+1},
$k_{n+1} = \cos(\theta).(xp_{n+1} - xp'_{n+1}) + \sin(\theta).(yp_{n+1} - yp'_{n+1})$
$j_{n+1} = -\sin(\theta).(xp_{n+1} - xp'_{n+1}) + \cos(\theta).(yp_{n+1} - yp'_{n+1})$
13) Enter the minor axis boundary ($hmax_{n+1}$), the major axis boundary ($qmax_{n+1}$), and the new measurement point position (k_{n+1}, j_{n+1}) into the ellipse equation,
$e\_result = ((k^2_{n+1})/(qmax_{n+1})) + ((j^2_{n+1})/(hmax_{n+1}))$
14) If $e\_result > 1$, a breakpoint has been detected. Then, determine whether the position of the point (p_{n+1}) is within the tolerance range of the corner (Dth_{corner}).
15) If it is within the corner's tolerance limit, set the point (k_{n+1},0) as the corner breakpoint and transform it back to the original coordinates.
16) If it is not within the corner's tolerance range, use the following three measurement points (p_{n+2}, p_{n+1}, p_n) as the starting point and return to step 1.
17) If $e\_result \leq 1$, then calculate the difference between the measurement points and the prediction line (j_{n+1}).
18) Recalculate the percentage of error, $\boldsymbol{\varepsilon}_{n+1} = |j_{n+1}|/r_{n+1}$
19) Recalculate the minor axis limit, $hmax_{n+1} = \sigma_{n+1} + \sigma_{offset}$
20) Repeat step 6.

IV. METHODOLOGY

A. Materials and Tools

This research was performed out with the following hardware and software: The RpLidar A1 module, Beaglebone Black Wireless, Intel i5 2.5 GHz RAM 4GB Win32bit Laptop, and Jupyter Notebook.

B. Stages of Research

The seven research stages in the development of the Adaptive Line Tracking Breakpoint Detector (ALTBD) method are as follows: data acquisition, LiDAR data noise model design, LiDAR data generator module design with simulation, data processing using the ALTBD method, data processing using comparison methods (ABD Line Extraction and LT), testing the ALTBD algorithm, and analysis and evaluation. Fig. 6 depicts the flow of the research method.

Because the data utilized in the test will use simulation data from the LiDAR data generator module, the first stage of this research is LiDAR data acquisition, which is done in real time to obtain a noise sensor model. The purpose of this data acquisition is to determine the noise characteristics of the RpLidar data and to validate the noise characteristics of the sensor manual document. Observations were taken on a straight wall with no obstacles that was greater than 5 meters long.

1) LiDAR data acquisition: The next stage is LiDAR data acquisition, which is done to create a LiDAR sensor noise model. The distance between the LiDAR sensor and the wall was varied from 15 to 225 cm with 5 cm intervals to obtain appropriate data. For each distance variation, data were collected five times.

2) Design of noisi model for the RpLiDAR A1: The simulation module use the noise model to verify that the generated data is as close to the real LiDAR data as possible. Residual noise will be added at random and generated using a normal distribution. The model is built using a polynomial regression method on the outcomes of LiDAR data collecting. Linear or polynomial interpolation is used to approximate the values of the element variables.

3) Design of a LiDAR data generator simulation module: The arena simulation module is used to test the ALTBD method. A square-shaped arena with four rooms and multiple tunnels is used for simulation testing. That's the 2.4 m by 2.4 m arena for the Indonesian Fire Extinguisher Robot Contest (KRPAI 2019). In the KRPAI guide document [28], the position of the doors in rooms 3 and 4 distinguishes four distinct room layout combinations. This simulation module is made up of five blocks, as depicted in Fig. 7. This module takes as input the robot's position (x,y) and direction (azimuth), as well as the choice of space combinations 3 and 4. This module generates level 3 scan data with angle and distance attributes.

4) Testing: The tests were analyzed by comparing three Breakpoint Detector methods: the one we developed (ALTBD), the LT method, and the ABD IEPF method from the Weerakoon et al. [20] research. In this research, the computing time, percentage of distance inaccuracy, and position difference between the Breakpoint Pattern Recognition results will be sought from each method. Computational time is measured beginning with level 3 scan data, which is processed in each method to produce an output in the form of a corner breakpoint. The percentage of distance measurement error is calculated by subtracting the results of the Breakpoint Detector from the position of the reference breakpoint.

The difference is then compared to the sensor's distance from the reference breakpoint. But first, the Breakpoint Detector data must pass through the Breakpoint Extraction module to enable determining the reference breakpoint point easier. This research will investigate the position difference between the Breakpoint Pattern Recognition results and the actual global position. On the LiDAR data generator simulation module, the global position is the input sensor location in the arena. Fig. 8 depicts the test diagram.

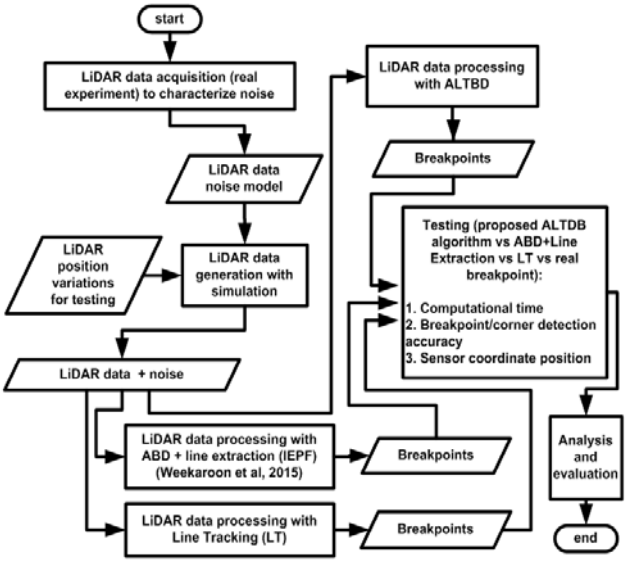

Fig. 6. Research Stages.

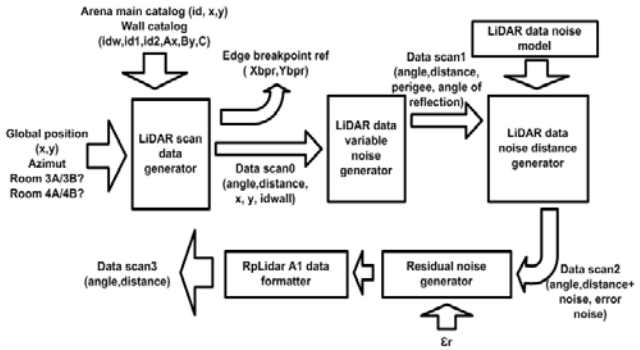

Fig. 7. Arena Simulation Module Block Diagram.

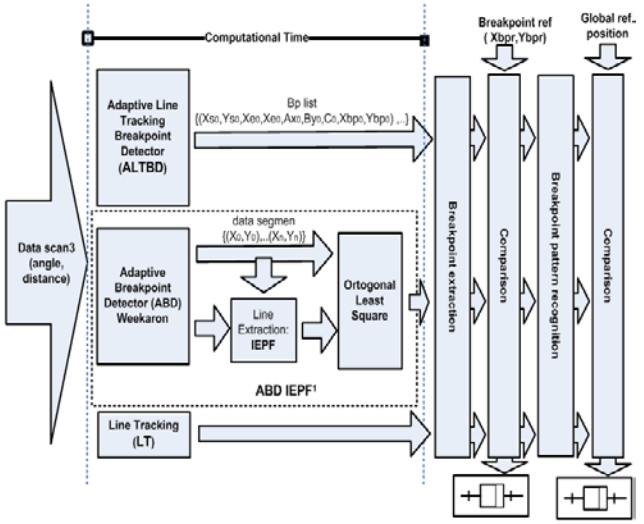

Fig. 8. ALTBD Test Method.

a) Breakpoint Extraction: This module sorts breakpoints, namely corner breakpoints, corner breakpoint terminal points, and other points that form only one side of the wall. This module also removes breakpoints that are overlapping. The breakpoint extraction module is required to support and facilitate the pattern matching process, allowing it to be developed as needed to achieve the best results.

b) Breakpoint Pattern Recognition: The closest or adjacent star methodology is used in this research's breakpoint pattern identification method, which is based on the concept of a star pattern recognition algorithm using a satellite star sensor [29]. The method's star pattern is replaced with a breakpoint pattern. The main catalog and the sub-catalogue are the two catalogs. The primary catalog contains IDs for all breakpoints, as well as two attributes: position in Cartesian coordinates and type (x,y). The sub catalog contains all of the neighboring breakpoint point IDs, as well as the distance between the breakpoints. The main idea behind this method is to compare each distance between the three observed breakpoint points to a catalog of previously defined space mappings.

V. Result and Discussion

A. Noise Data Model Design of RpLidar A1

The Rplidar A1 noise data model is made up of three conditions: 0-60, 60-100, and 100-225 cm at the point nearest to the sensor and the wall. The first condition is approximated by 6-order polynomial regression, while the values of the variables in the equation are approximated by 3rd order polynomial interpolation. The regression used is appropriate to utilize 2nd order polynomials in the range of 60 to 225, but the range of 60 to 100 values of the element variables is produced by 3rd order interpolation. The range of 100 to 225 is then estimated using linear interpolation.

B. Adaptive Line Tracking Breakpoint Detector (ALTBD)

The measurement inaccuracy (σ_{offset}) percentage value utilized is 0.04. This value is the average of the percentage of errors obtained from data collection results in the first stage of the research. The tolerance value as the corner breakpoint limit (Dth_{corner}) is 23 cm, which is half the door distance of 46 cm. Fig. 9 depicts one of the ALTBD module's results and Fig. 10 depicts one of the ABD IEPF module's results.

The ALTBD results show the detection of several corner breakpoints generated by the 18 mm thick side wall. Because the detected side is close to the sensor, the number of scanning points created is sufficient to make a segment with more than four points. Breakpoints that should be designated corner breakpoints by the IEPF method are regarded terminal breakpoints, in contrast to ABD IEPF. This is due to the fact that the required threshold distance is not met on that side.

C. Adaptive Breakpoint Detector (ABD) and IEPF

The threshold value equation utilized is consistent with the results of the Weerakoon et al. [20] research. The residual error is 0.0038, whereas the angle (λ) is 8º. The extraction line used is an IEPF with a limit value of 46 cm. The intersection of the two lines generated by the IEPF two-segment linear regression is used to determine the corner breakpoint.

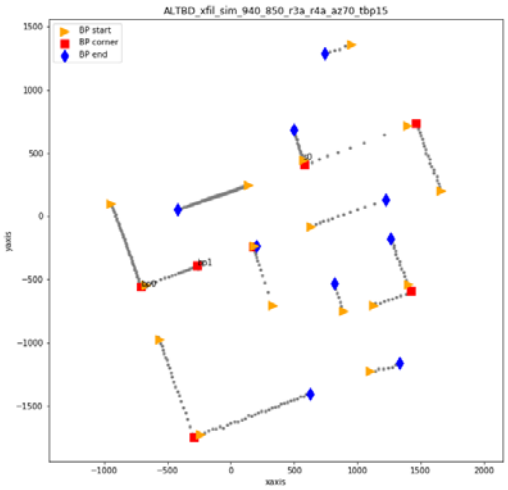

Fig. 9. Breakpoint Detector Results of ALTBD.

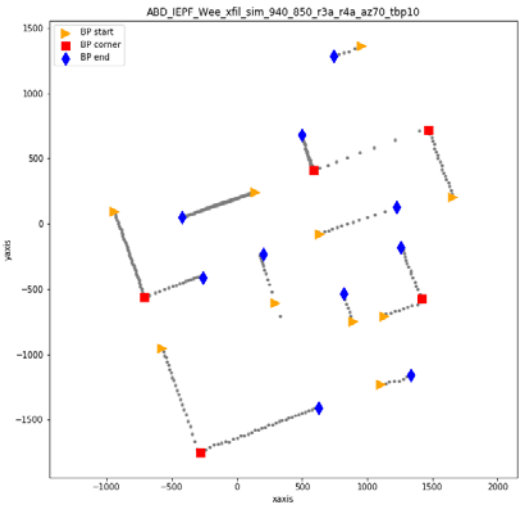

Fig. 10. Breakpoint Detector Results of ABD IEPF.

D. Testing

Fig. 11 depicts a comparison of the computing times of the three Breakpoint Detector methods. The comparison of the ALTBD, ABD IEPF, and LT methods reveals that the LT method computes faster than the ALTBD and ABD IEPF methods. The average computation time for the LT technique is 59,87 ± 1,94 ms, whereas the ALTBD and ABD IEPF methods require 101.61 ± 2.63 ms and 175.4 ± 10.13 ms, respectively. Although LT has the shortest calculation time, the resulting corner breakpoints are not as precise as ALTBD and ABD IEPF. If the accuracy of the LiDAR sensor's position in the room is a major priority in robot localization, this will be a disadvantage of the LT method. The ALTBD method has the second fastest computation time. Despite the fact that the ALTBD calculation is more sophisticated, each data point is only processed once. Unlike the IEPF ABD, the data is processed at least twice in this method. This method processes data twice, once using the ABD method and once using the IEPF method. The theory behind the significant difference in computational time is that the LT, ABD, and ALTBD methods are linear search algorithms with a time complexity of O(n), whereas the IEPF method is a linear recursive algorithm with a time complexity of $T(n) = 2T(n/2) + O(n)$, with the worst-case complexity being $O(n^2)$. Table I presents the detailed results of the computational time comparison.

According to the test results, the IEPF's disadvantage is the recognition of an inaccurate corner breakpoint. This takes the form of a four-sided space, as illustrated in Fig. 10. This is due to the fact that the IEPF seeks the greatest deviation from the line connecting the starting and finishing points. Furthermore, there is a lot of noise at that point. This type of room is said to be unfavorable for detecting corner breakpoints using the IEPF method.

E. Breakpoint Extraction

The results of the tests show that the IEPF method has weaknesses, including the inaccuracy of recognizing corner breakpoints in the form of a four-sided room, as illustrated in Fig. 12. Table II indicates that the ALTBD method detects corner breakpoints with only two errors out of 43 position samples tested, but the ABD IEPF detects them with ten. When compared to ALTBD and ABD IEPF, the LT method has the most detection errors. If there are only two corner breakpoints on one side of the wall, the accuracy of the corner breakpoint is declared correct. It is still considered a corner breakpoint detection error if there are more than two corner breakpoints, even if they are close to one other.

The IEPF method is inaccurate in recognizing corner breakpoints because it looks for the greatest difference between the start and finish lines. Furthermore, there is a lot of noise at that point. When using IEPF, this type of room will result in an inaccurate corner breakpoint. In addition, ALTBD generates an inaccurate a corner breakpoint. Because the erroneous corner breakpoint is typically located distant from the LiDAR sensor, it has little effect on the location results of the Breakpoint Pattern Recognition, as illustrated in Fig. 13. However, with the spatial pattern depicted in Fig. 12, the resulting corner breakpoint position in the IEPF is not quite close to the LiDAR sensor. The Breakpoint Pattern Recognition method will not produce the right position if these two spots are not actual corner breakpoints. Some corner breakpoint detection problems in the LT method occur when the incorrect position is close to the sensor and the distance to the other reference corner breakpoint is within the Breakpoint Pattern Recognition tolerance limit, as illustrated in Fig. 14. As a result, the Breakpoint Pattern Recognition procedure is unable to locate a pattern match in the database.

F. Breakpoint Pattern Recognition

There are three stages to the breakpoint pattern recognition process. The first step is to look for a pattern match for the three initial reference points, which are the three breakpoints nearest to the sensor. The results of the search throughout the sub-catalogues are saved. Step two must be completed if there is more than one possibility. The second stage is to find a match against the original three reference points on the remaining corner breakpoints and terminal breakpoints in all sub-catalogues. The match of the nearby breakpoint points with the most points that determines the three initial patterns is eligible for selection. If stage two yields two or more results, stage three will be carried out. Stage three follows the same principles as stage two, with the exception of the matching point. This is the last of the breakpoints on the list (points that form a line). If step three is completed, all breakpoints have been verified.

TABLE I. COMPUTATIONAL TIME RESULTS

Location	**Num. sample**	**Computational time in ms**		
		ALTBD	***ABDiepf***	***LT***
Room 1	9	103,3 ± 8,0	198,9 ± 11,0	62,4 ± 0,0
Room 2	8	103,8 ± 8,0	208,6 ± 8,0	58,5 ± 7,2
Room 3	10	99,5 ± 8,0	157,9 ± 15,5	56,5 ± 8,0
Room 4	8	101,4 ± 8,0	169,0 ± 11,7	62,4 ± 0,0
Street	8	100,3 ± 8,0	135,9 ± 21,5	60,2 ± 5,9

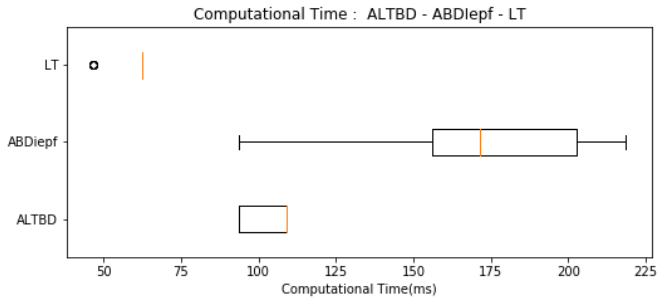

Fig. 11. Comparison of ALTBD, ABD IEPF and LT Computational Times in a Boxplot.

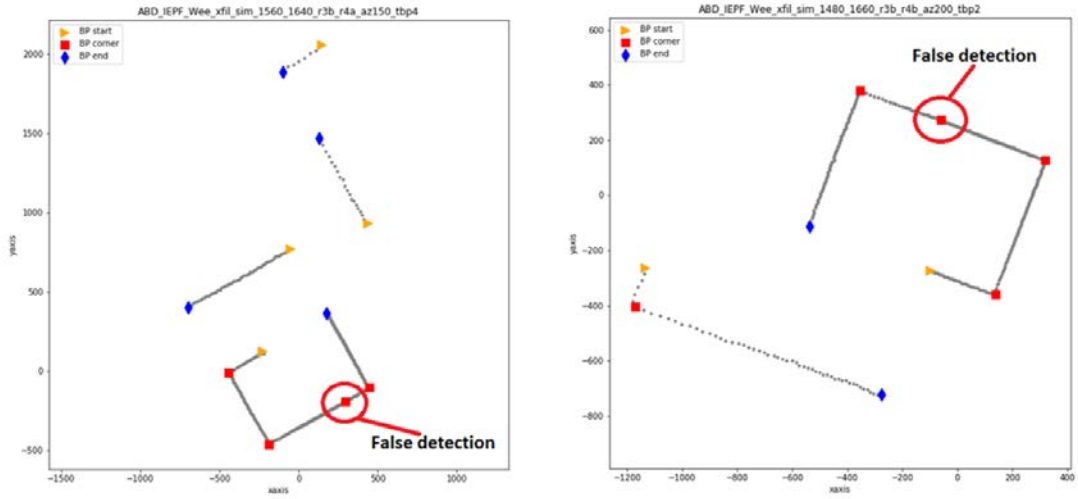

Fig. 12. IEPF Corner Breakpoint Detection.

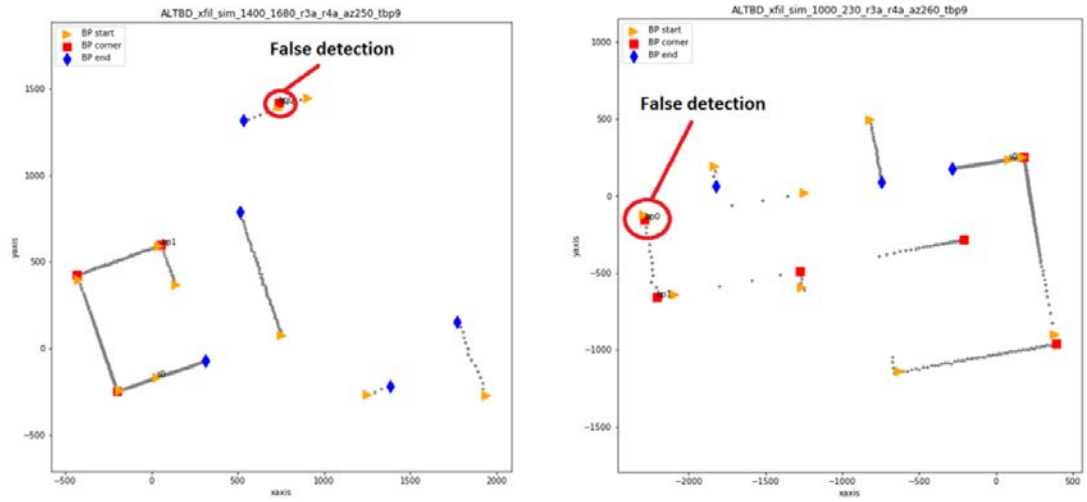

Fig. 13. ALTBD Corner Breakpoint Detection.

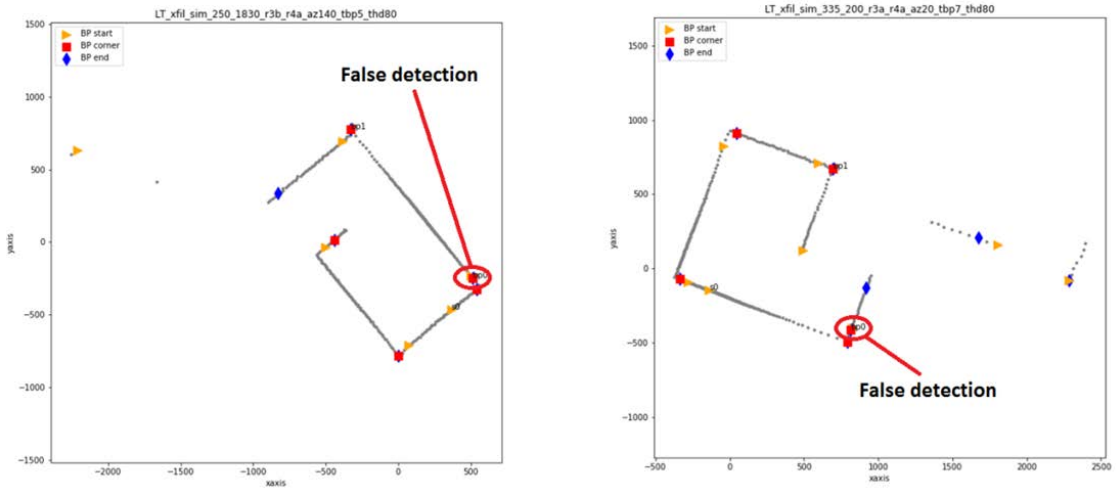

Fig. 14. LT Corner Breakpoint Detection.

TABLE II. CORNER BREAKPOINT DETECTION RESULTS

Location	Num. sample	Corner breakpoint					
		ALTBD		*ABDiepf*		*LT*	
		True	*False*	*True*	*False*	*True*	*False*
Room 1	9	33	0	34	3	36	4
Room 2	8	32	0	32	0	32	6
Room 3	10	30	0	32	3	36	3
Room 4	8	28	1	28	2	26	3
Street	8	34	1	32	2	45	7

Furthermore, the major catalog should be summarized in order to limit the number of candidate positions generated. In fact, one wall has four sides: two long sides and two wide sides, or what is generally referred to as the thick side. The wall has four breakpoints at first, which are then combined into two breakpoints with the condition that the wall thickness is included in the tolerance limit value. The main catalog, which originally had 40 points, was reduced to 19 points. Similarly, the sub-catalog averaged 19 nearby breakpoints at the end. The reduced number of neighbors in the reference point accelerates the matching process.

Fig. 15, 16, and 17 shows that the position generated by ALTBD is more accurate than ABD IEPF through the pattern recognition process using the closest breakpoint method. This is due to the fact that the reference points for position computations are only the two closest corner breakpoints. ALTBD produces the closest corner breakpoint more precisely than ABD IEPF.

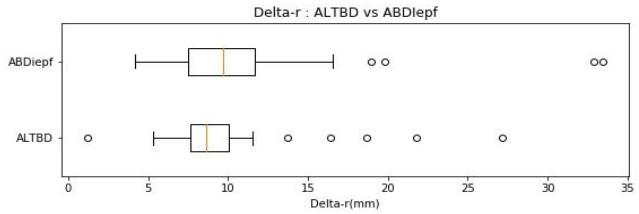

Fig. 15. Boxplot of the difference between ALTBD and ABD IEPF Position Errors.

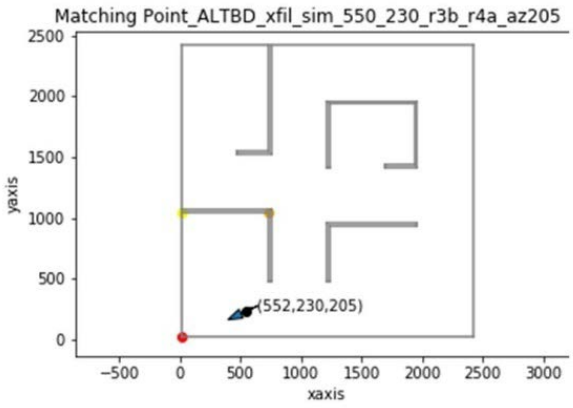

Fig. 16. Breakpoint Pattern Recognition Results of ALTBD.

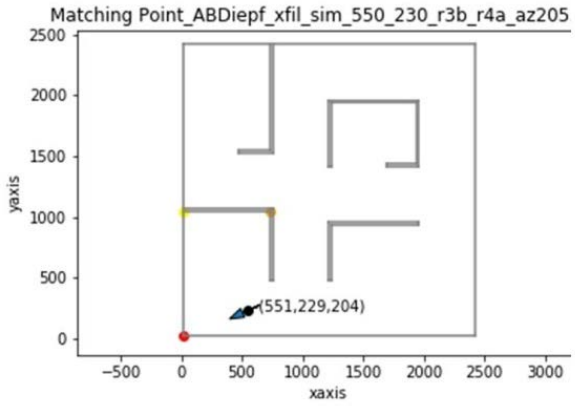

Fig. 17. Breakpoint Pattern Recognition Results of ABD IEPF.

VI. CONCLUSION

This research was successful in developing a new algorithm for breakpoint detection called the Adaptive Line Tracking Breakpoint Detector (ALTBD). This method modifies the Line Tracking (LT) and Adaptive Breakpoint Detector (ABD) algorithms by introducing a new threshold area in the shape of an ellipse, resolving corner breakpoint detection more adaptive and fasting. Algorithm testing was done by apply the ALTBD and ABD algorithms with Iterative End Point Fit (ABD IEPF) to detect the position of the robot in the room.

The results of the tests prove that the ALTBD computation time is faster in detecting corner breakpoints than the ABD IEPF method. The average computation time for the ALTBD method is 101.61 ± 2.63 ms, while the ABD IEPF is 175.4 ± 10.13 ms. The corner breakpoint detection error in the ALTBD method is only two errors out of 43 position samples, whereas the ABD IEPF method has ten detection errors. Furthermore, the ALTBD method is more accurate in determining the position of the robot than the ABD IEPF method, with a distance difference of 9.72 ± 1.55 mm, instead of 11.2 ± 2.14 mm in the ABD IEPF.

ACKNOWLEDGMENT

The authors would like to thank the Head of the Satellite Technology Research Center (BRIN) and Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM) IPB University for their support in completing this research. This research is partially sponsored by the Ministry of Research and Technology of the Republic of Indonesia (RISTEK-BRIN) through Penelitian Dasar Unggulan Perguruan Tinggi (PDUPT) research grant number 3626/IT3.L1/PT.01.03/ P/B/2022.

REFERENCES

[1] R. Gonzalez, F. Rodiguez, J. L. Guzma, and M. Berengguel, “Comparative study of localization techniques for mobile robots based on indirect kalman filter”, International Symposium on Robotics (ISR), pp:253–258. Switzerland. http://www.ual.es/~rgs927/papers/r amon-gonzalez-isr09.pdf, 2009.

[2] J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, (1997) “Mobile robot positioning: sensors and techniques”, Journal of Robotic Systems, 14(4), pp. 231–249. doi: 10.1002/(SICI)1097-4563(199704)14:4<231 ::AID-ROB2>3.0.CO;2-R.1997.

[3] J. Park, M. Choi, Y. Zu, and J. Lee, “Indoor localization system in a multi-block workspace”, *Robotica*, 28(3), pp. 397–403. doi: 10.1017/S0263574709005712, 2010.

[4] N. L. Doh, H. Choset, and W. K. Chung, “Relative localization using path odometry information”, Autonomous Robots, 21(2), pp. 143-154. doi: 10.1007/s10514-006-6474-8, 2006.

[5] M. Faisal and H. ElGibreen, "Adaptive Self-Localization System for Low-Cost Autonomous Robot", 7th International Conference on Control, Automation and Robotics (ICCAR),Singapore, DOI:10.1109/ICCAR52225.2021.9463494, 2021.

[6] J. H. Kim, and P. H. Seong, “Experiments on orientation recovery and steering of autonomous mobile robot using encoded magnetic compass disc”, IEEE Transactions on Instrumentation and Measurement vol. 45, no. 1, pp. 271-273. https://doi.org/10.1109/19.481346, 1996.

[7] P. Artiemjew dan K. Ropiak, "Robot Localization in the Magnetic Unstable Environment", 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, doi: 10.1109/IRC.2019.00105, 2019.

[8] K. T. Song, and Y. H. Suen, “Design and implementation of a path tracking controller with the capacity of obstacle avoidance”, Journal of Control Systems and Technology , Vol. 4, No. 3, pp.151-160. Taipei, Taiwan, 1996.

[9] T. D. Kwon, and J. S. Lee, “A stochastic map building method for mobile robot using 2-d laser range finder”, 7th Autonomous Robots, 1-18. Kluwer Academic Publishers. Boston. https://doi.org/10.1023/A:1008966218715, 1999.

[10] C. C. Hsu, H. C. Chen,C. C. Wong, and C. Y. Lai, “Omnidirectional Ultrasonic Localization for Mobile Robots”, Sensors and Materials, Vol. 34, No. 2, https://doi.org/10.18494/SAM3419, Tokyo, 2022.

[11] B. Tao, H. Wu, Z. Gong, Z. Yin, and H. Ding, “An RFID-Based Mobile Robot Localization Method Combining Phase Difference and Readability”, IEEE Transactions on Automation Science and Engineering, Vol. 18, doi: 10.1109/TASE.2020.3006724, 2021.

[12] P. Jensfelt, and H. Christensen, “Laser based position acquisition and tracking in an indoor environment”, Proc. of the Intl. Symposium on Robotics and Automation, 1(May), pp. 331–338, Available at: https://www.researchgate.net/publication/238648464, 1998.

[13] G. A. Borges, and M. J. Aldon, “Line extraction in 2D range images for mobile robotics”, Journal of Intelligent and Robotic Systems: Theory and Applications, 40(3), pp. 267–297. doi: 10.1023/B:JINT.0000038945.55712.65, 2004.

[14] A. Siadat, A. Kaske, S. Klausmann, M. Dufaut, and R. Husson, “An optimized segmentation method for a 2D laser-scanner applied to mobile robot navigation”, IFAC Proceedings Volumes, 30(7), pp. 149–154. doi: 10.1016/s1474-6670(17)43255-1, 1997.

[15] K. J. Lee, “Reactive navigation for an outdoor autonomous vehicle”, Master Thesis. University of Sydney, Department of Mechanical and Mechatronic Engineering. Australia, 2001.

[16] K. C. J. Dietmayer, J. Sparbert, and D. Streller, ”Model based object classification and object tracking in traffic scenes from range images”, In: Proceedings of IV IEEE Intelligent Vehicles Symposium, Tokyo, 2001.

[17] S. Santos, J. E. Faria, F. Soares, R. Araujo, and U. Nunes, “Tracking of multi-obstacles with laser range data for autonomous vehicles”, In: Proc. 3rd National Festival of Robotics Scientific Meeting (ROBOTICA), pp. 59-65, 2003.

[18] N. Certad, R. Acuna, A. Terrones, D. Ralev, J. Cappelletto, and J. C. Grieco, ”Study and improvements in landmarks extraction in 2D range images based on an Adaptive Curvature Estimation”, Andean Region International Conference (ANDESCON) VI. Cuenca, Ecuador. https://doi.org/10.1109/Andescon.2012.31, 2012.

[19] S. Y.An, J. G.Kang, L. K. Lee, and S. Y. Oh, “Line segment-based indoor mapping with Salient Line Feature Extraction”, *Advanced Robotics*, 26(5–6), pp. 437–460. doi: 10.1163/156855311X617452, 2012.

[20] T. Weerakoon, K. Ishii, and A. A. F. Nassiraei, “Geometric feature extraction from 2D laser range data for mobile robot navigation”, *2015* IEEE 10th International Conference on Industrial and Information Systems, ICIIS 2015 - Conference Proceedings, pp. 326–331. doi: 10.1109/ICIINFS.2015.7399032, 2016.

[21] J. Dingyao, C. Jin, and X. Yuan, “An Extracting Method of Corner Points from Laser Sensor Readings”, Proceedings of the 37th Chinese Control Conference, Wuhan, China, doi:10.23919/ChiCC.2018.8482534, 2018.

[22] V. Nguyen, A. Martinelli, N. Tomatis, and R. Siegwart, “A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics”, in 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 1929–1934. doi: 10.1109/IROS.2005.1545234, 2005.

[23] J. Vandorpe, H. V. Brussel, and H. Xu, “Exact dynamic map building for a mobile robot using geometrical primitives produced by a 2D range finder”, Proceedings of IEEE International Conference on Robotics and Automation. pp.901-908. Minnesota, USA. https://doi.org/10.1109/ROBOT.1996.503887, 1996.

[24] G. A. Borges, and M. J. Aldon, “A Split-and-Merge Segmentation Algorithm for Line Extraction in 2-D Range Images”, Proceedings 15th International Conference on Pattern Recognition, ICPR-2000. Barcelona, Spain. https://doi.org/10.1023/B:JINT.0000038945.55712.65, 2000.

[25] R. O. Duda RO and P. E. Hart, “Pattern Classification and Scene Analysis”, John Wiley and Sons, Hoboken, 1973.

[26] M. Fischler and R. Bolles, “Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography”, Communications of the ACM, 24(6):381–395. https://doi.org/10.1145/358669.358692, 1981.

[27] D. A. Forsyth and J. Ponce, “Computer Vision: A Modern Approach”, Prentice Hall, 2003.

[28] KRPAI 2019 Guide, [online] Available: https://kontesrobotindonesia.id/data/2019/Panduan_KRPAI2019.pdf.

[29] M. A. Saifudin, and R. H. Triharjanto, ”*Algoritma pengenalan pola bintang untuk deteksi posisi bintang pada star sensor satelit LAPAN*”, *Jurnal Teknologi Dirgantara*, 8(1), pp. 36–42, Indonesia, 2010.