Implementation of Levenberg-Marquardt Point to Line Iterative Closest Point and Pose Graph Optimization for 2D Indoor Mapping on Differential Drive Mobile Robot

Rafi Darmawan[1], Ananta Adhi Wardana[1], Rodik Wahyu Indrawan[1,2], and Gama Indra Kristianto[2]

[1] Airlangga University, Indonesia
[2] Adhikara Wiyasa Gani, Indonesia
rafi.darmawan-2020@ftmm.unair.ac.id

Abstract. PT Adhikara Wiyasa Gani (AWG) faces a physical track especially a magnetic tape durability issue on its Automated Guided Vehicle (AGV) due to crossing by heavy-duty vehicles. Free navigation and path planning can be a solution that allows the AGV to dynamically adjust its path without relying on physical trajectories. For free navigation to be realized, the robot needs a map or knowledge of its working area. This research uses Iterative Closest Point (ICP) and Pose Graph Optimization (PGO) for mapping methods with LSLiDAR N10 and DDSM115 motors on three different artificial maps. The robot has a differential drive steering model with dimensions of 35 cm x 30 cm. The mapping results were compared to ground truth maps using Average Distance Nearest Neighbor (ADNN) and Structural Similarity Index Measure (SSIM) metrics. The results show that mapping method can be used for room localization and mapping quite well. The ground truth map is formed on a 10 x 6 squares grid map, with dimensions of 60 cm x 60 cm for each square. Mapping with the combination of ICP, PGO, and wheel odometry produced ADNN and SSIM values of 5,5 cm and 0,601; 8,8 cm and 0,669; and 8,5 cm and 0,629, respectively, for the three maps tested. The largest value of the ADNN metric is 8,8 cm, this value is used as padding in the robot dimensions so that there is a remaining 16,2 cm on the length side of the robot and 21,2 cm on the width side of the robot with respect to a square grid.

Keywords: Mobile Robot, Iterative Closest Point, Indoor, Mapping, Pose Graph Optimization

1 Introduction

Automated Guided Vehicle (AGV) is a mobile robot that navigates using the existing infrastructure in its operating environment [1]. The AGVs are commonly used for transporting goods automatically in factories, assembly lines, or warehouses. There are many methods to enable the AGV's automatic navigation. The most common method is by navigating through the physical tracks such as colored or magnetic lines.

T. Amrillah et al. (eds.), *Proceedings of the International Conference on Advanced Technology and Multidiscipline (ICATAM 2024)*, Advances in Engineering Research 245,
https://doi.org/10.2991/978-94-6463-566-9_11

is by navigating through the physical tracks such as colored or magnetic lines. However, the use of physical tracks is not suitable in the high traffic workplaces since it is not durable if other vehicles are continuously passing on it. Thus, it can disturb the operation of the AGV in the long term especially in factories. Another issue is that the robot must strictly follow the track in order to navigate. Therefore, the AGV is difficult to perform free navigation to find the shortest path flexibly without considering the tracks [2]. Furthermore, if the AGV faces some obstacles, it is difficult to avoid the obstacles dynamically by leaving the track.

The use of physical tracks can make AGVs struggle with navigation and obstacle avoidance dynamically. Additionally, the vulnerability of the physical track to damage is another issue for AGVs. PT Adhikara Wiyasa Gani uses Automated Guided Vehicles (AGVs) with magnetic tape to transport goods from multiple stations. It faces magnetic tape durability issues due to passing by heavy load vehicles while it is on the floor, necessitating daily replacements. Free navigation and path planning can solve the flexibility issues of AGVs [2]. Free navigation allows AGVs to dynamically adjust their movement path, avoid obstacles, and adapt to environmental changes without requiring additional physical track.

Nowadays, many researchers work extensively on AGV's navigation without using physical tracks. The challenge in AGV's navigation is to move from the starting point to the destination without causing damage to the surrounding environment [3]. This challenge arises because, to determine the path, the robot must know its position and orientation in real-time (localization) and the target position and orientation relative to its working environment and time. Localization using maps or landmarks is reliable and highly accurate, indicating that good navigation requires a map to obtain trustworthy position and orientation data. Accurate position and orientation provided from maps and landmarks enables an AGV to determine an efficient route matching the robot's destination [4]. Mapping can be done by combining two models, observation model and motion model [5]. The work environment mapping can be generated from the combination of a 2D LiDAR sensor as the observation sensor and a wheel encoder as the motion sensor. The use of LiDAR is due to its high accuracy and stability against lighting changes [6].

In this paper, we propose an implementation of Levenberg-Marquardt Point to Line Iterative Closest Point (LM-PLICP) and Pose Graph on a differential wheeled mobile robot to generate maps. The LM-PLICP is used as a scan matcher to generate maps from the 2D LIDAR. Pose Graph used to correct the drifting trajectory and wheel odometry used as initial guess transformation on LM-PLICP process.

We organize this paper as follows: Section 2 describes the related works of our proposed method. Section 3 discusses the implementation of LM-PLICP and Pose Graph Optimization on a differential wheeled mobile robot. Section 4 verifies the proposed method for generating maps. Section 5 provides the conclusion of this paper.

2 Related Work

2.1 Wheel Odometry

Odometry from moving mobile robots requires kinematics of the steering method of the robots. The robots is driven by two wheels on the side of the body frame named as differential drive. It moves based on the difference in speed of both wheels.

The robot coordinates q, q element 3x1 in generalized coordinate system is defined as follows:

$$q = [x \quad y \quad \theta] \tag{1}$$

Where x and y are the position of the robot in x-axis and y-axis in generalized coordinates, respectively, and θ denotes the orientation of the robot. The robot has one non-holonomic constraint which is expressed as follows :

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0 \tag{2}$$

Where $\dot{x}$ and $\dot{y}$ denote the rate of robot position in x-axis and y-axis, respectively. The non-holonomic constraint in Equation (2) can be expressed in as follows :

$$\mathrm{J}(q) = [\sin\theta \quad -\cos\theta \quad 0] \tag{3}$$

There exist a null matrix $\mathrm{S}(q)$, so that

$$\mathrm{S}^T(q)\mathrm{J}^T(q) = 0 \tag{4}$$

Which yield

$$\mathrm{S}^T(q) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

and

$$\dot{q} = \mathrm{S}^T(q)u = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6}$$

Where u is a input vector. Therefore, by considering the robot rates of q ($\dot{q}$ element 3x1) is obtained as follows :

$$\dot{q} = [\dot{x} \quad \dot{y} \quad \dot{\theta}] = [v \cdot \cos\theta \quad v \cdot \sin\theta \quad \omega] \tag{7}$$

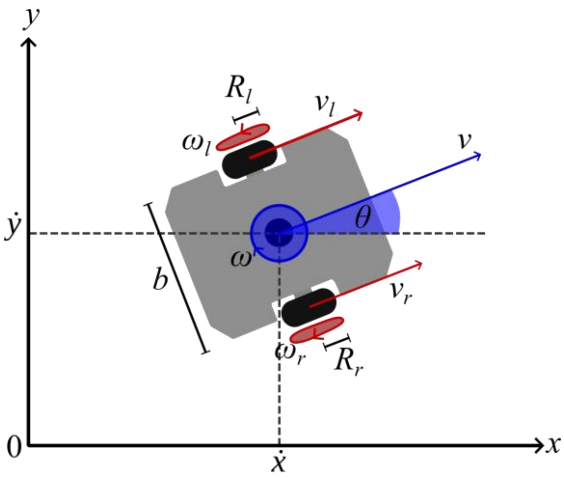


Fig. 1. Kinematics of Differential Drive Mobile Robot

The linear and angular velocity on Equation (7) can be achieved by using inputs control of both wheels. The v and ω described by a set of equations :

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\omega_r R_r + \omega_l R_l}{2} \\ \frac{\omega_r R_r - \omega_l R_l}{b} \end{bmatrix} = \begin{bmatrix} \frac{v_r + v_l}{2} \\ \frac{v_r - v_l}{b} \end{bmatrix} \tag{8}$$

Where ω_l and ω_r are the angular velocity of wheels on the left and right, respectively. Distances between two wheels b, and radius of the wheel r. The trajectory of robot can be achieved using 2nd-order Runge-Kutta integration [7]. It requires linear velocity v, angular velocity ω from Equation (8), and sampling period Δt.

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} + \Delta t\ v\ \cos\left(\theta_{t-1} + \Delta t\ \frac{\omega}{2}\right) \\ y_{t-1} + \Delta t\ v\ \sin\left(\theta_{t-1} + \Delta t\ \frac{\omega}{2}\right) \\ \theta_{t-1} + \Delta t\ \omega \end{bmatrix} \tag{9}$$

Pose at time t denoted by (x_t , y_t, θ_t). Being position on x and y axis at previous time step, x_{t-1} and y_{t-1}.

2.2 Levenberg-Marquardt Point to Line Iterative Closest Point Algorithm

Iterative Closest Point aims to align two point clouds P and Q by finding a rigid-body transformation ($\boldsymbol{R}$, t) applied to point cloud P and iteratively minimizes the error metric. Point to Point distance of vanilla ICP [8] uses Euclidean distance between correspondences points. Point to Line Iterative Closest Point (PLICP) is a variant of ICP with a difference in error metric. PLICP computes error metric by projecting distances of correspondence points to the normal vector of the target point cloud [9]. PLICP can be solved by an optimization process such as Gauss-Newton and Levenberg-Marquardt methods [11, 10]. The error metric of PLICP is expressed by:

$$E(\mathbf{x}) = \sum_i \left[\left(\boldsymbol{R}(\mathbf{x}_\theta) p_i + t\left(\mathbf{x}_x, \mathbf{x}_y\right) - q_i \right) \cdot n_{q,i} \right]^2 \quad \text{, for } \begin{matrix} \boldsymbol{R} \in SO(2) \\ t \in \mathbb{R}^2 \end{matrix} \tag{10}$$

Refer to Equation (10), the error metric of PLICP E computed by given parameters $\mathbf{x}$ consisting of (x, y, θ), rotation matrix and translation vector in 2 dimensions $\boldsymbol{R}$ and t obtained from $\mathbf{x}$, correspondences of source and target point cloud p and q, normals vector of target point cloud n_q. Left terms in the bracket of Equation (10) describe the distance of correspondence point in the transformed source point to target point. The right terms describes projecting the distance to normals of target point.

Levenberg-Marquardt (LM) is an optimization procedure that attempting reduce a the value of non-linear function ($E(\mathbf{x})$ in this case) [10, 12]. Minimizing E needs to compute Gradient and Hessian matrix of E at any parameters [10]. Given current initial guess $\breve{\mathbf{x}}$ and perturbation of $\mathbf{x}$ donated by $\Delta\mathbf{x}$.

Let $\mathbf{e}_i(\mathbf{x}) = \left(\boldsymbol{R}(\mathbf{x}_\theta) p_i + t\left(\mathbf{x}_x, \mathbf{x}_y\right) - q_i\right) \cdot n_{q,i}$,

$$E(\mathbf{x}) = \sum_i \underbrace{\mathbf{e}_i(\mathbf{x})^T \mathbf{e}_i(\mathbf{x})}_{E_i} \tag{11}$$

Thus, optimal parameters $\mathbf{x}^*$ expressed by

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\ E(\mathbf{x}) \tag{12}$$

Equation (11) can be approximated by using first-order Taylor expansion around current initial guess $\breve{\mathbf{x}}$ [12, 13].

$$\begin{aligned} \mathbf{e}_i(\breve{\mathbf{x}} + \Delta\mathbf{x}) &= \mathbf{e}_i(\breve{\mathbf{x}}) + \nabla \mathbf{e}_i(\breve{\mathbf{x}}) \cdot \Delta\mathbf{x} \\ &\simeq \mathbf{e}_i(\breve{\mathbf{x}}) + \mathbf{J}_i \Delta\mathbf{x} \end{aligned} \tag{13}$$

Substitute Equation (13) to E_i in Equation (11),

$$E_i(\breve{\mathbf{x}} + \Delta\mathbf{x}) \simeq (\mathbf{e}_i(\breve{\mathbf{x}}) + \mathbf{J}_i\,\Delta\mathbf{x})^T(\mathbf{e}_i(\breve{\mathbf{x}}) + \mathbf{J}_i\,\Delta\mathbf{x}) = \underbrace{\mathbf{e}_i^T\mathbf{e}_i}_{\mathbf{c}_i} + 2\,\underbrace{\mathbf{e}_i^T\mathbf{J}_i}_{\mathbf{b}_i}\,\Delta\mathbf{x} + \Delta\mathbf{x}^T\,\underbrace{\mathbf{J}_i^T\mathbf{J}_i}_{\mathbf{H}_i}\,\Delta\mathbf{x} \quad (14)$$

Rewrite error metric in Equation (11) with Equation (14) by using the same method in [13].

$$E(\breve{\mathbf{x}} + \Delta\mathbf{x}) = \mathbf{c} + 2\mathbf{b}\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H} \quad (15)$$

Thus, we can obtain $\Delta\mathbf{x}^*$ by solving linear system and Levenberg-Marquardt

$$\Delta\mathbf{x}^* = -(\mathbf{H} + \lambda\mathbf{I})^{-1}\,\mathbf{b} \quad (16)$$

$$\Delta\mathbf{x}^* = -(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e} \quad (17)$$

A damping factor λ becomes smaller when $E(\mathbf{x})$ reduced, and the behaviors of are optimization more like gradient descent. When it is big, the behaviors more like Gauss-Newton. Optimal rigid-body transformation $\mathbf{x}^*$ can be obtained by adding an initial guess $\breve{\mathbf{x}}$ to computed perturbation $\Delta\mathbf{x}^*$.

$$\mathbf{x}^* = \breve{\mathbf{x}} + \Delta\mathbf{x}^* \quad (18)$$

In Equation (17), $\mathbf{J}$ is a Jacobian of $\mathbf{e}_i(\mathbf{x})$ and $\mathbf{I}$ is an identity matrix**.** Jacobian of $\mathbf{e}_i(\mathbf{x})$ expressed by taking partial derivatives of $\mathbf{e}_i(\mathbf{x})$ w.r.t parameters $\mathbf{x}$:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{e}}{\partial x} & \frac{\partial \mathbf{e}}{\partial y} & \frac{\partial \mathbf{e}}{\partial \theta} \end{bmatrix} \quad (19)$$

$$\mathbf{J} = \begin{bmatrix} n_{q,x} & n_{q,y} & n_{q,x}\left(-q_x \sin\theta - q_y \cos\theta + n_{q,y}\left(q_x \cos\theta - q_y \sin\theta\right)\right) \end{bmatrix} \quad (20)$$

2.3 Pose Graph Optimization

Pose Graph Optimization (PGO) is an algorithm to eliminate the small accumulated errors on the robot trajectory [14]. ICP based localization is impossible to repair the accumulated trajectory errors [14]. Lu and Milios [15] proposed a method to construct the graph by using relative motion from scan-matching and optimizing the existing graph by iterative. Pose Graph Optimization is a nonlinear least squares problem [16] that can be linearized by using Taylor expansion and solved using an optimization procedure. Grisetti et al. [13] explain the problem of PGO and derive the linearized loss function. On PGO, nodes are describe as poses and edges as virtual measurements. The loss function of nonlinear pose graph optimization expressed by :

$$\mathbf{F}(\mathbf{p}) = \sum_{\langle i,j\rangle \in C} \boldsymbol{e}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{z}_{ij})^T \, \boldsymbol{\Omega}_{ij} \boldsymbol{e}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{z}_{ij}) \tag{21}$$

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \, \mathbf{F}(\mathbf{p}) \tag{22}$$

Let $\mathbf{p} = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)^T$ as parameters to optimize, where $\mathbf{p}_i$ to represent a pose of node *i*, $\mathbf{z}_{ij}$ and $\boldsymbol{\Omega}_{ij}$ describe mean and the weighting matrix of $\mathbf{p}_j$ to $\mathbf{p}_i$, respectively. Error function $\boldsymbol{e}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{z}_{ij})$ defines how good the virtual measurement compares to real measurement.

$$\boldsymbol{e}_{ij}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{z}_{ij}) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{p}_i, \mathbf{p}_j) \tag{23}$$

In Equation (23), $\hat{\mathbf{z}}_{ij}(\mathbf{p}_i, \mathbf{p}_j)$ is an inverse pose composition $\boldsymbol{T}_i^{-1} \cdot \boldsymbol{T}_j$, for $\boldsymbol{T} \in SE(2)$. We can be obtained $\hat{\mathbf{z}}_{ij}$ by :

$$\hat{\mathbf{z}}_{ij}(\mathbf{p}_i, \mathbf{p}_j) = \begin{bmatrix} \boldsymbol{R}_i^T \boldsymbol{R}_j & \boldsymbol{R}_i^T (t_j - t_i) \\ 0 & 1 \end{bmatrix} \triangleq \begin{bmatrix} \boldsymbol{R}_i^T (t_j - t_i) \\ \theta_j - \theta_i \end{bmatrix} = \begin{bmatrix} \Delta t \\ \Delta\theta \end{bmatrix}, \text{for } \begin{matrix} \boldsymbol{R} \in SO(2) \\ t \in \mathbb{R}^2 \end{matrix} \tag{24}$$

Where $\triangleq$ describes $\hat{\mathbf{z}}_{ij}$ in vector space, *t* represents translation vector in *x* and y, and compute $\boldsymbol{e}_{ij}(\mathbf{p}_i, \mathbf{p}_j, \mathbf{z}_{ij})$.

$$\boldsymbol{e}_{ij} \triangleq \begin{bmatrix} \boldsymbol{R}_{ij}^T \left(\boldsymbol{R}_i^T (t_j - t_i) - t_{ij} \right) \\ \theta_j - \theta_i - \theta_{ij} \end{bmatrix}, \text{for } \begin{matrix} \boldsymbol{R} \in SO(2) \\ t \in \mathbb{R}^2 \end{matrix} \tag{25}$$

Linearize the error function $\boldsymbol{e}_{ij}$ around $\mathbf{p}$ by first order Tylor expansion for solving nonlinear least squares function $\mathbf{F}(\mathbf{p})$. The step by step to linearize the equation exists in [13, 16]. The final linearized equation through Taylor expansion is similar to Equation (15).

$$\mathbf{F}(\breve{\mathbf{p}} + \Delta\mathbf{p}) = \mathbf{c} + 2\mathbf{b}\Delta\mathbf{x} + \Delta\mathbf{x}^T\mathbf{H} \tag{26}$$

Where $\mathbf{c} = \sum \boldsymbol{e}_{ij}^T \Omega_{ij} \boldsymbol{e}_{ij}$, $\mathbf{b} = \sum \boldsymbol{e}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}$, and $\mathbf{H} = \sum \mathbf{J}_{ij}^T \Omega_{ij} \mathbf{J}_{ij}$. Solve $\Delta\mathbf{p}^*$ by using optimization procedure same as Equation (16) and Equation (17). The Jacobian matrix contains only on row where the node *i* and *j* exist. The partial derivatives of error function are expressed by:

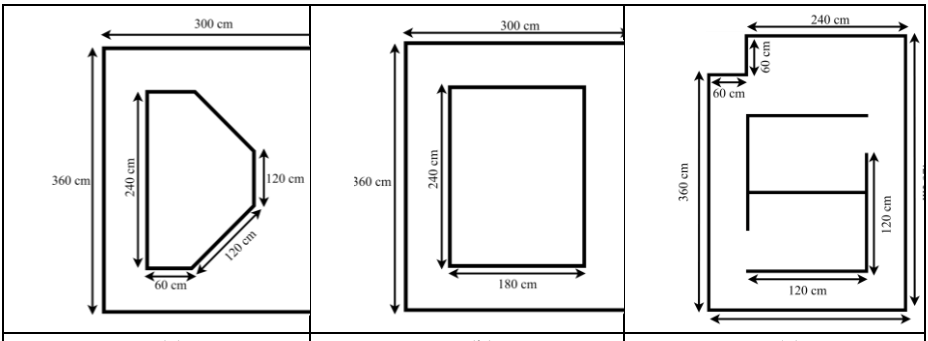
$$\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{p}_i} = \begin{bmatrix} -\boldsymbol{R}_{ij}^T \boldsymbol{R}_i^T & \boldsymbol{R}_{ij}^T \frac{\partial \boldsymbol{R}_i^T}{\theta_i}(t_j - t_I) \\ 0 & -1 \end{bmatrix} \quad (27)$$

$$\frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{p}_j} = \begin{bmatrix} \boldsymbol{R}_{ij}^T \boldsymbol{R}_i^T & 0 \\ 0 & 1 \end{bmatrix} \quad (28)$$

$$\mathbf{J}_{ij} = \begin{bmatrix} 0 & 0 & 0 & \dots & \frac{\partial \boldsymbol{e}_{ij}}{\partial \mathbf{p}_i} & \dots & \frac{\partial \boldsymbol{e}_{ij}}{\partial \mathbf{p}_j} & \dots & 0 & 0 & 0 \end{bmatrix} \quad (29)$$

3 Experimental Results

The mapping method tested three different artificial rooms. They are built on top of 10 x 6 grid map with 60 x 60 cm for each grid. Data was collected by using a ros2 bag. The data captured are pose of the robot from wheel odometry and LiDAR data correspondence.



(a) (b) (c)

Fig. 2. Artificial Maps

We use wheel odometry as an initial guess for LM-PLICP. Key frames for Pose Graph construction are selected based on the robot's travel from wheel odometry, with distance threshold for both rotation and translation. LM-PLICP is also used for detect loop closure, the criteria for two nodes classify as loop nodes when the transformed point cloud at current frame has mean distance less than a threshold to loop closure candidate at previous nodes. A loop closure candidate is selected by calculating the distance between the current node and each previous node. If any of these distances is less than the threshold, the candidate is considered for loop closure.

While optimizing the existing graph we need the uncertainty of each nodes. Since each node produce by using LM-PLICP, we implemented the method proposed by Yuan et al. [17], adapting their approach by changing the 3-dimensional rotation matrix to a 2-dimensional representation. The entire mapping implementation is outlined in the pseudocode **Algorithm 1**.

Algorithm 1 : LM-PLICP-Pose Graph SLAM

Input : $\{O\}$, $\{C\}$
Params : $\beta_{translation}$, $\beta_{rotation}$, $\beta_{closure}$, $\beta_{candidate}$, $\beta_{correspondences}$
Output : $\{V\}$, $\{C_{buffer}\}$
// Pose Graph //
$G = \{V, E\}$
$\mathbf{T_0} = \mathbf{I_3}$
$\mathbf{T_0} \rightarrow V_1$
$C_{buffer} = \{\}$
$O_{prev} = O_0$
$C_0 \rightarrow C_{buffer}$
for i, $O_i \leftarrow 1$ **to** N:
    $\Delta O = O_i - O_{prev}$
    if $||\Delta O_{translation}|| < \beta_{translation}$ **or** $||\Delta O_{rotation}|| < \beta_{rotation}$:
        $P_{target} = C_{i-1}$
        $P_{source} = C_i$
        $\mathbf{T_{ICP}}, \Phi, D_\Phi = \text{LM-PLICP}(P_{source}, P_{target}, \Delta O, \beta_{correspondences})$
        $\mathbf{Cov_{ICP}} = \text{KalmanICPCov}(P_{source}, P_{target}, T_{ICP}, \Phi)$
        $\mathbf{T_i} = \mathbf{T_{i-1}} \cdot \mathbf{T_{ICP}}$
        $\mathbf{T_i} \rightarrow V_i$
        $\mathbf{Cov_{ICP}} \rightarrow E_{i-1 \rightarrow i}$
        $O_{prev} = O_i$
        $P_{source} \rightarrow C_{buffer}$

```
        if i > 6:
            {N} = KNN(T_i, T_0:i-1, radius = β_candidate)
            for k in N :
                P_target = C_buffer(k)
                T_closure, Φ, D_Φ = LM-PLICP(P_source, P_target, β_correspondences)
                Cov_closure = I_3
                if mean(D_Φ) < β_closure :
                    Cov_closure → E_i→k
            Optimize(G)
    i++
end for
```

This section presents an result of Levenberg-Marquardt Point to Line Iterative Closest Point and Pose Graph Optimization on three maps. We used the following five parameter values in **Algorithm 1** :

Table 1. Parameters used for Mapping Method

Parameters	Values (meters)
$\beta_{translation}$	0.05
$\beta_{rotation}$	0.05
$\beta_{closure}$	0.005
$\beta_{candidate}$	0.5
$\beta_{correspondences}$	0.1

We found misalignment in the LM-PLICP when robot moved to the corners of the maps, as illustrated in Fig. 3(a). This issue arises when robot cannot "see" the adjacent walls due to the perpendicular nature of the corners. It makes the correspondence points have large distances, as illustrated in Fig. 3(b). To address this problem, we applied a distance threshold to exclude correspondence points with excessive distances. The result shown in Fig. 4.

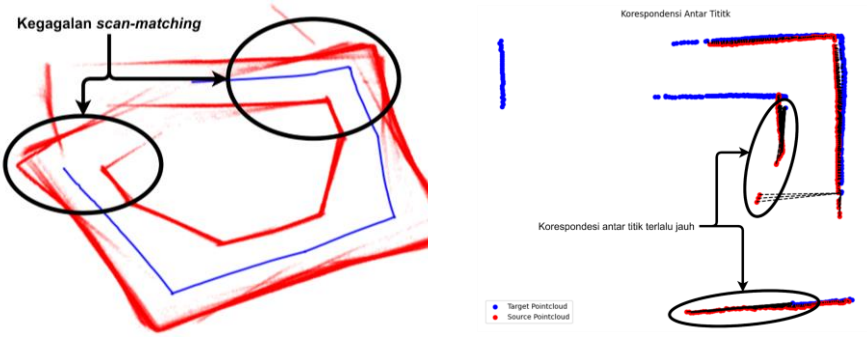


Fig 3. LM-PLICP Misalignment

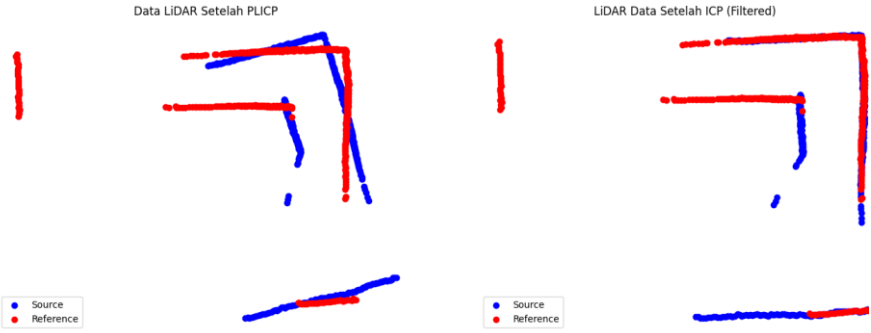


Fig 4. Before and After Applied Correspondences Distance Threshold

For generating the occupancy grid maps, we use the Bresenham Line algorithm as the inverse sensor model in the occupancy grid mapping pseudocode presented in [20]. Evaluation metrics for calculate the performance of our maps, we use Structural Similarity Index Measure (SSIM) and Average Distance Nearest Neighbor (ADNN) same as study from [18] and [19].

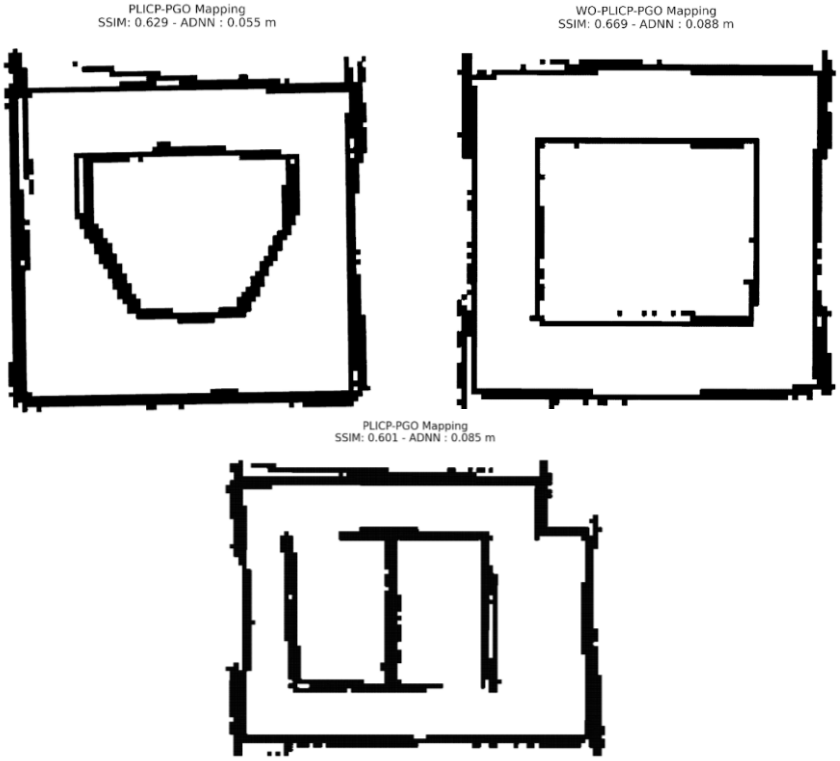


Fig 5. The Results of LM-PLICP and PGO

The Average Distance Nearest Neighbor (ADNN) values for LM-PLICP and PGO are 5.5 cm for map I, 8.8 cm for map II, and 8.5 cm for map III. The robot dimensions are 35 cm x 30 cm, and the grid map measures 60 cm x 60 cm. With the largest ADNN value from the mapping experiment being 8.8 cm, we add this value as padding to the robot dimensions, resulting in adjusted dimensions of 38.8 cm x 43.8 cm. Given the 60 cm x 60 cm grid map, there remains an empty space of 21.2 cm x 16.2 cm. Therefore, the mapping method can be used to navigate mobile robots.

4 Conclusion

In this paper, we presented the implementation of Levenberg-Marquardt Point to Line Iterative Closest Point (LM-PLICP) and Pose Graph Optimization (PGO) for 2D room mapping using a differential drive mobile robot. The initial pose from wheel odometry serves as an initial guess for LM-PLICP to minimized the iterations. To handle misalignment, a correspondences distances threshold is applied with values 0.1 meters. The pose from LM-PLICP are then fed into pose graph, which is executed when loop closure is detected. Loop closure is identified using LM-PLICP, where if the average distance between corresponding points is less than 0.05 meters to loop closure candidate, the node is considered a loop closure. Finally, an occupancy grid map is constructed using the Bresenham Line algorithm.

The performance of purposed method is evaluated using Average Distance Nearest Neighbor (ADNN) and Structural Similarity Index Measure (SSIM) metrics. The ADNN values indicate that map I, II, and III have respective values of 5.5 cm, 8.8 cm, and 8.5 cm, while the SSIM values are 0.601, 0.669, and 0.629, respectively. To account for the maximum ADNN error of 8.8 cm, the robot's dimensions are adjusted with it as padding, leaving 16.2 cm on the length side and 21.2 cm on the width side of the robot.

The code for collecting the data, mapping, and evaluating our purposed method is available at https://github.com/drmwnrafi/ROS2-PLICP-POSE-GRAPH

Acknowledgments. This study was funded by PT Adhikara Wiyasa Gani.

Disclosure of Interests. The authors declare that they have no competing interests.

References

1. Oyekanlu, E.A., Smith, A.C., Thomas, W.P., Mulroy, G., Hitesh, D., Ramsey, M., Kuhn, D.J., Mcghinnis, J.D., Buonavita, S.C., Looper, N.A., Ng, M., Ng'oma, A., Liu, W., Mcbride, P.G., Shultz, M.G., Cerasi, C., Sun, D.: A Review of Recent Advances in Automated Guided Vehicle Technologies: Integration Challenges and Research Areas for 5G-Based Smart Manufacturing Applications. IEEE Access. 8, 202312–202353 (2020). https://doi.org/10.1109/access.2020.3035729
2. Teso-Fz-Betoño, D., Zulueta, E., Fernandez-Gamiz, U., Aramendia, I., Uriarte, I.: A Free Navigation of an AGV to a Non-Static Target with Obstacle Avoidance. Electronics. 8, 159 (2019). https://doi.org/10.3390/electronics8020159
3. Fikri, A.A., Anifah, L.: Mapping And Positioning System On Omnidirectional Robot Using Simultaneous Localization And Mapping (SLAM) Method Based On LIDAR. Jurnal Teknologi. 83, 41–52 (2021). https://doi.org/10.11113/jurnalteknologi.v83.16918
4. Cai, Y., Lu, Z., Wang, H., Chen, L., Li, Y.: A Lightweight Feature Map Creation Method for Intelligent Vehicle Localization in Urban Road Environments. IEEE Transactions on

Instrumentation and Measurement. 71, 1–15 (2022). https://doi.org/10.1109/tim.2022.3181903
5. Indrawan, R.W., Sulistijono, I.A., Basuki, A.: Pemetaan 3 Dimensi Untuk Menentukan Jalur Evakuasi Alternatif Pada Smart Robot Rescue. INOVTEK POLBENG. 9, 128 (2019). https://doi.org/10.35314/ip.v9i2.1013
6. Li, F., Liu, S., Zhao, X., Zhang, L.: Real-Time 2-D Lidar Odometry Based on ICP. Sensors. 21, 7162 (2021). https://doi.org/10.3390/s21217162
7. De Giorgi, C., De Palma, D., Parlangeli, G.: Online Odometry Calibration for Differential Drive Mobile Robots in Low Traction Conditions with Slippage. Robotics. 13, 7 (2023). https://doi.org/10.3390/robotics13010007.
8. Besl, P.J., McKay, N.D.: Method for registration of 3-D shapes. In: SPIE Proceedings. SPIE (1992).
9. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: Proceedings. IEEE International Conference on Robotics and Automation. IEEE Comput. Soc. Press (1992).
10. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: The Art of Scientific Computing. (1992).
11. niosus: ICP, https://github.com/niosus/notebooks/blob/master/icp.ipynb, last accessed 2024/08/20.
12. Fitzgibbon, A.W.: Robust registration of 2D and 3D point sets. In: Procedings of the British Machine Vision Conference 2001. p. 43.1-43.10. British Machine Vision Association (2001).
13. Grisetti, G., Kummerle, R., Stachniss, C., Burgard, W.: A Tutorial on Graph-Based SLAM. IEEE Intelligent Transportation Systems Magazine. 2, 31–43 (2010). https://doi.org/10.1109/mits.2010.939925.
14. Suzuki, R., Kataoka, R., Ji, Y., Umeda, K., Fujii, H., Kono, H.: SLAM using ICP and graph optimization considering physical properties of environment. In: 2020 21st International Conference on Research and Education in Mechatronics (REM). IEEE (2020).
15. Lu, F., and Milios, E. 1997. Globally consistent range scan alignment for environment mapping. Autonomous Robots 4:333–349.
16. Kummerle, R., Grisetti, G., Strasdat, H., Konolige, K., Burgard, W.: G2o: A general framework for graph optimization. In: 2011 IEEE International Conference on Robotics and Automation. IEEE (2011).
17. Yuan, R.H., Taylor, C.N., Nykl, S.L.: Accurate Covariance Estimation for Pose Data From Iterative Closest Point Algorithm. NAVIGATION: Journal of the Institute of Navigation. 70, navi.562 (2023). https://doi.org/10.33012/navi.562.
18. Dichtl, J., Le, X.S., Lozenguez, G., Fabresse, L., Bouraqadi, N.: PolySLAM: A 2D Polygon-based SLAM Algorithm. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). IEEE (2019).
19. Zi, B., Wang, H., Zheng, H.: A new metric for assessing the performance of 2D Lidar SLAMs. Collaborative European Research Conference. 2815, (2021).
20. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press (2005).