Artificial Life and Robotics (2024) 29:311–325
https://doi.org/10.1007/s10015-024-00937-8

ORIGINAL ARTICLE

Online simultaneous localization and mapping with parallelization for dynamic line segments based on moving horizon estimation

Haziq Muhammad[1] · Yasumasa Ishikawa[1] · Kazuma Sekiguchi[1] · Kenichiro Nonaka[1]

Received: 19 July 2023 / Accepted: 5 January 2024 / Published online: 1 March 2024
© International Society of Artificial Life and Robotics (ISAROB) 2024

Abstract
In this paper, to render SLAM robust in dynamic environments, we propose a novel LiDAR SLAM algorithm that estimates the velocity of all objects in the scene while suppressing speed of static objects by moving horizon estimation (MHE). We approximate environment features as dynamic line segments having velocity. To deal with static objects as well, MHE is employed, so that its objective function allows the addition of velocity suppression terms that treat stationary objects. By considering association probability, the SLAM algorithm can track the endpoints of line segments to estimate the velocity along the line segments. Even if it is temporarily occluded, the estimation is accurate, because MHE considers a finite length of past measurements. Parallelization of the robot's localization with the map's estimation and careful mathematical elimination of decision variables allows online implementations. Post-process modifications remove possible spurious estimates by considering the piercing of LiDAR lasers and integrating maps. Simulation and experiment results of the proposed method prove that the presented algorithm can robustly perform online SLAM even with moving objects present.

Keywords Simultaneous localization and mapping · LiDAR SLAM · Dynamic SLAM · Moving horizon estimation · Probabilistic data association filter

1 Introduction

Simultaneous localization and mapping (SLAM) is a powerful tool for modern-day robots [1, 2]. There is much research on SLAM, as this technique has many applications. The main problem with the conventional SLAM is that it cannot handle dynamic objects as it assumes everything is static [3–5]. In many situations, dynamic objects are present; thus, an SLAM that considers dynamic objects is necessary. Many pieces of research on dynamic SLAM have been conducted. The recurring theme to tackle this problem is either a SLAM that treats the dynamic objects as outliers [6–8] or executing SLAM while tracking the dynamic objects [9–11].

Past methods to solve this problem includes: Wang et al. [12] used an algorithm where they execute the SLAM for stationary objects while detecting and tracking moving objects (DATMO) separately to reduce the computational burden. Einhorn et al. [13] combined NDT SLAM and occupancy grid maps to represent the environment. They calculated the occupancy probability of each cell and used DATMO to track moving objects. Bahraini et al. [14] used a multi-level RANSAC (ML-RANSAC) strategy to robustly associate data by finding the highest consensus among multiple hypotheses while determining static or dynamic objects through SLAM and DATMO. Pfreundschuh et al. [15] leveraged a neural network capable of moving object detection and segmentation for accurate LiDAR SLAM performance. All past methods share a common theme of distinguishing static and moving objects. Incorrect distinguishments may lead to map inconsistencies and estimation errors. Moreover, an additional step of determining whether the object is moving or otherwise induces an extra computational burden.

In an ever-dynamic environment, an SLAM that distinguishes stationary and dynamic objects is not enough,

This work was presented in part at the joint symposium of the 28th International Symposium on Artificial Life and Robotics, the 8th International Symposium on BioComplexity, and the 6th International Symposium on Swarm Behavior and Bio-Inspired Robotics (Beppu, Oita and Online, January 25–27, 2023).

✉ Haziq Muhammad
g2291008@tcu.ac.jp

Kenichiro Nonaka
knonaka@tcu.ac.jp

[1] Tokyo City University, 1-28-1, Tamazutsumi, Setagaya, Tokyo 158-8557, Japan

because, in reality, a seemingly stationary object can still move. Thus, a uniform SLAM that estimates the velocity states of all objects is more proper to be applied in such cases. This study's objective is to achieve a true dynamic SLAM that can adapt to any dynamic change of any environment. Our proposed approach to this SLAM problem differs significantly from the traditional strategies, which often classify objects as static or dynamic based on their motion characteristics. We instead treat all objects as moving objects represented by dynamic line segments with velocity to approximate environmental features. Methods that distinguish between stationary or dynamic objects may have lag and, if an object's model is incorrectly assumed, may lead to estimation failure [16]. Treating all objects as dynamic, on the other hand, may cause problems for static objects as their velocity needs to remain zero. To address this issue, we offer a novel strategy that involves including velocity suppression terms derived from a moving horizon estimation (MHE) approach, which allows for numerical state optimization that considers past measurements while having constraints [17]. Leveraging the estimator's numerical nature, we implement artificial potential field terms that can suppress low-speed objects' velocity to near zero. Naturally, the computational load of the estimator will increase due to the estimation of both position and velocity states. To address this, we will explain how to reduce the optimization variable through variable elimination to achieve comparable speed to a method where it only estimates the position state. Additionally, we introduce a parallel optimization approach for the robot and each map part to enhance computational efficiency. Through this approach, we have established a robust SLAM that works in the presence of moving objects without the need to distinguish between static and dynamic entities.

To validate the algorithm's effectiveness, we ran numerical simulations and did on-site experiments. The simulation is to validate the algorithm's accuracy and repeatability. For the experiments, we have mainly two types of it where we use an electric wheelchair equipped with an LiDAR sensor. The first SLAM experiment is in a complex static environment. Due to our approach of estimating every object as moving object, this may affect the SLAM negatively in static environments. We verify whether our proposed novel velocity suppression can help improve the accuracy of SLAM. In the next experiment, we did a moving object SLAM experiment in the laboratory. We needed to verify its robustness to do SLAM within a dynamic environment. The results of all experiments show that the proposed algorithm can execute SLAM robustly for all by estimating all objects as dynamic line segments with low-speed suppression.

The sections of this paper are organized as follows; Sect. 2 will explain more on the estimation model, Sect. 3 will explain the SLAM algorithm, Sect. 4 will introduce the MHE strategy, Sect. 5 describes the type of experiments and its setup, Sect. 6 will show results on the SLAM algorithm enhancements, Sect. 7 will describe the simulation and online experiment results of the dynamic SLAM in full detail, and finally, Sect. 8 will give the conclusion.

2 Estimation model

2.1 Robot motion model

In this research, we use an electric wheelchair shown in Fig. 1 as the robot for our experiments. The model of the planar vehicle robot is depicted in Fig. 2. x_r and y_r are the Cartesian positions and θ_r is the angular position of the robot, respectively. The robot state pose is defined as, $x_p := [x_r, y_r, \theta_r]^T \in \mathbb{R}^3$. The state transition equation for the robot's pose is the following discrete-time dynamics:

$$x_p[k+1] = f(x_p[k], u_p[k]) + v_p[k], \tag{1}$$

where f is a nonlinear vector-valued function representing the motion dynamics, $u_p := [v_r, \omega_r]^T$ is an input vector where v_r and ω_r are the robot's linear and angular velocity, respectively. $v_p \sim N(0, Q_p)$ is the Gaussian distributed process noise where $Q_p \in \mathbb{R}^{3\times3}$ is the process noise covariance. The function f is described below

$$f(x_p, u_p) := x_p + \begin{bmatrix} v_r \cos\theta_r \\ v_r \sin\theta_r \\ \omega_r \end{bmatrix} \Delta, \tag{2}$$

where Δ is the sampling time.

2.2 Line-based map

For this SLAM, dynamic line segment models represent the environment features [18] and the line model is shown in Fig. 3. The state that represents the model position state is

Fig. 1 The rear motored electric wheel chair equipped with LiDAR sensor used in this study

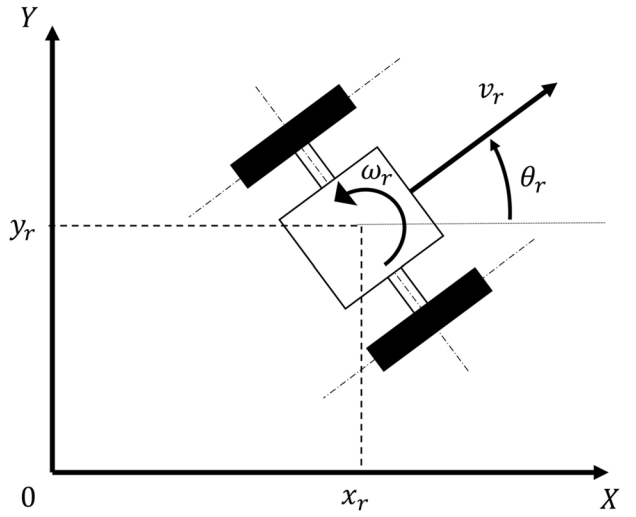


Fig. 2 Planar robot model of the electric wheelchair, x_r, y_r are the robot's x and y position, θ_r is the robot's orientation, and v_r and ω_r is the robot's linear and angular velocity, respectively

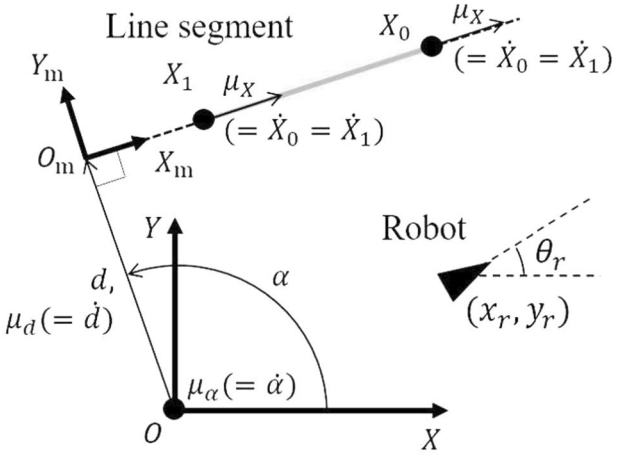


Fig. 3 The geographical representation of the robot and map state of the SLAM

$m^i := [d^i, \alpha^i, X_0^i, X_1^i]^T \in \mathbb{R}^4$, where i represents the ith line segment and d is the distance of the perpendicular line from the origin to the line segment, and while α is the angle from the x-axis, and X_0 and X_1 are, respectively, the endpoints located on the right and left sides of the line segment with respect to the local line coordinate O_m–$X_m Y_m$. We express the line in this manner as the estimations of the d and α states are helpful when measurements cannot observe the endpoints X_0 and X_1 due to occlusion. Furthermore, as this algorithm considers the velocity of the environment, we consider the velocity state as $\mu^i := [\mu_d^i, \mu_\alpha^i, \mu_X^i]^T \in \mathbb{R}^3$ where μ_d and μ_α represents the velocities of d and α, respectively. μ_X represents the velocities along the local axis X_m of both the endpoints X_0 and X_1 of the line segment. Naturally, both endpoints velocity should be estimated individually; however, in our study, we assume that the line's length is constant if both endpoints are estimated thereby, the velocity of X_0 is equal to X_1. To reduce computational effort, we simplify the state into just μ_X. With this, we define the line map state as $x_m^i := [m^{iT}, \mu^{iT}]^T \in \mathbb{R}^7$. The line map state's transition equation is the following discrete-time dynamics:

$$x_m^i[k+1] = Ax_m^i[k] + Gv_m^i[k], \tag{3}$$

where k represents the time step, and A and G are both matrices shown respectively in Eqs. (4) and (5)

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta \\ 0 & 0 & 0 & 1 & 0 & 0 & \Delta \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

$$G = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \Delta & 0 & 0 \\ 0 & \Delta & 0 \\ 0 & 0 & \Delta \end{bmatrix}, \tag{5}$$

and $v_m^i \sim N(0, Q_m^i)$ is a normally distributed process noise of the line segments with $Q_m^i \in \mathbb{R}^{3\times3}$. Combining both robot pose state x_p and all map state x_m, we get the whole system's state, $x := [x_p^T, x_m^T]^T$.

2.3 Observation vector and model function

For the observation vector, this research takes two types of observation: the LiDAR-measured distance and the line segments' endpoints angle. The LiDAR laser distance y_l^j for the jth point cloud (PC) data point is represented as follows:

$$y_l^j = h_l(x_p, m^i, \phi^j) + w_l, \tag{6}$$

where $w_l \sim N(0, \sigma_l^2)$ is the normal distributed observation range noise with σ_l as the observation range noise standard deviation and the scalar-valued function h_l computes the estimated distance of laser between LiDAR and the object in the environment which is written in Eq.(7)

$$h_l(x_p, m^i, \phi^j) = \frac{d^i - x_r \cos\alpha^i - y_r \sin\alpha^i}{\cos(\theta_r + \phi^j - \alpha^i)}, \tag{7}$$

where ϕ^j represents the angle of the jth LiDAR laser. Furthermore, in Eq. (8), y_e^i represents the state of the endpoints angle from the LiDAR for map state i

$$y_e^i = h_e(x_p, m^i) + w_e^i, \tag{8}$$

where $w_e^i \sim N(0, S)$ is the normally distributed endpoint's angle observation noise with $S \in \mathbb{R}^{2\times2}$ as the endpoint's angle observation noise covariance, h_e is a vector-valued function that computes both of the endpoints X_0 and X_1 angles as written below:

$$h_e(x_p, m^i) := \begin{bmatrix} h_0(x_p, d^i, \alpha^i, X_0^i) \\ h_1(x_p, d^i, \alpha^i, X_1^i) \end{bmatrix}, \tag{9}$$

The angle of the endpoint is calculated by the function $h_i\,(i = 0, 1)$:

$$h_i(x_p, d, \alpha, X_i) = \tan^{-1}\left(\frac{-X_i\cos\alpha + d\sin\alpha - y_r}{X_i\sin\alpha + d\cos\alpha - x_r}\right). \tag{10}$$

All observed PC distance and endpoint angles will be computed by Eqs. (6) and (8); thus, if we take vector y_l to define all observed PC distance and y_e to define all observed endpoint angles, we can get the observation vector Eq.(11)

$$y := [y_l^T, y_e^T]^T. \tag{11}$$

3 SLAM algorithm

3.1 Algorithm overall

For the algorithm of this method of SLAM, it will be done as the flowchart shown in Fig. 4.

The algorithm intakes the previous robot's state $x_p[k-1]$, odometry's data $u_p[k]$, LiDAR data and existing map state $x_m[k-1]$ if any as the input. Initially the robot state is set at the origin with $\theta = 0$. The algorithm will process these information to produce the updated state of the robot $x_p[k]$ and the map $x_m[k]$. We divide the algorithm into three main sections: preprocessing, estimation, and post-processing. In the preprocessing section, we predict the line models based on the measurements and associate them with existing line map states (Sect. 3.2). Next, the endpoint angle's measurements are modified by considering association probability (Sect. 3.3). Afterward, in the estimation section, the predicted states are used as inputs for MHE and Extended Kalman Filter (EKF) to compute the posteriors. Finally, in the post-processing section, we would eliminate spurious maps (Sect. 3.4) and integrate line maps we consider to originate from the same object (Sect. 3.5).

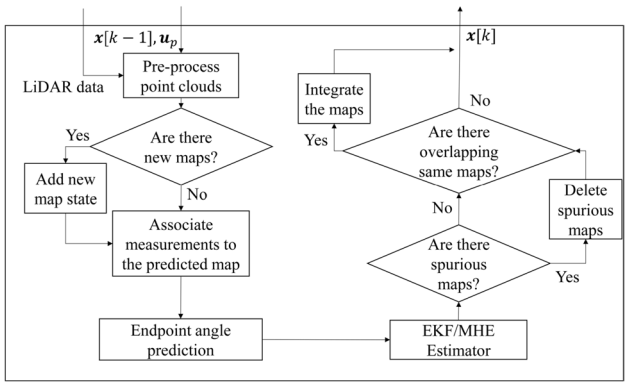


Fig. 4 SLAM algorithm's flowchart

3.2 Preprocessing of point clouds

In this study, we use 2-D LiDAR measurements for SLAM. To process the raw point cloud data (PC), we implement the random sample consensus (RANSAC) algorithm to transform them into 2-D line segment models to estimate the SLAM.

This algorithm is conceptualized in Fig. 5. There are three steps to transform the raw PC into line models. For the first step, we establish regions of interest (ROI) and extract point clouds nearby them for RANSAC model fitting [19]. The ROIs are the areas within the threshold distance set to the predicted line maps. The algorithm will then fit these extracted PC into line models by RANSAC. For the second step, we will cluster the remaining PCs based on Euclidean distance and predict the line model using RANSAC for each PC cluster. This process will continue until no more line models can be predicted. However, at the initial time $k = 0$, since there are no predicted map states yet to establish ROI, we skip step 1 and start from step 2. In the final step, we will

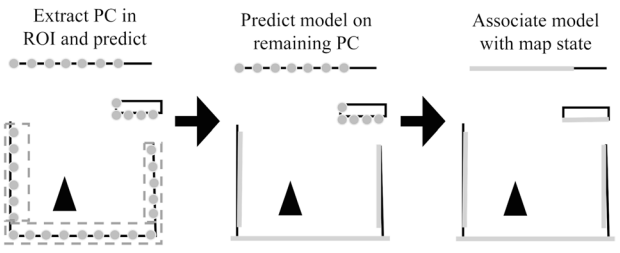


Fig. 5 The process of transforming point clouds into 2-D line models. These schematics are top-view examples of the LiDAR point clouds (PC) in the robot's environment where the black triangle represents the robot, the dots are the PC, and the black lines are the objects. The first step is extracting PC within the region of interest (ROI) of existing predicted line maps and predicting the gray line model using RANSAC. The ROIs are areas within the dashed line. Afterward, we will fit the remaining point cloud (PC) through clustering and RANSAC to predict line models. Finally, those line models will be associated with the estimated maps

associate the fitted models to update the line map state. If there is a model that does not have any association, a new map state is generated for it.

3.3 Line segment's endpoint angle prediction

Due to the discrete angle of the LiDAR sensor, the endpoints' angle could not be measured directly. Thus, to treat the observed laser's angle y_e, the following formula was used:

$$y_e = \eta^{x\to y_0} h_e(x_p^-, m^-) + \sum_{n\in\Gamma^x(\epsilon)} \eta^{x\to y_n} \nu_n. \tag{12}$$

In Eq. (12), the first term is the weighting of the endpoint angle prediction $h_e(x_p^-, m^-)$, where x_p^- and m^- are the priori robot and map position states, respectively, and the second term is the weighting of the candidate endpoints angle ν_n, that exist in the verification region Γ^x which is the Mahalanobis distance region around the prediction value below ϵ computed by the 1-dimensional Chi-square distribution based on P_G. η is the association probability in the Probabilistic Data Association Filter that considers probabilistic data association [20] where $\eta^{x\to y_0}$ represents the weight based on the likelihood of the target not associating with any measurements and $\eta^{x\to y_n}$ represents the weight based on the likelihood of the target associating to the measurement y_n. These weights are expressed in Eqs. (13) and (14)

$$\eta^{x\to y_0} = \frac{1 - P_D P_G}{1 - P_D P_G + \sum_{n\in\Gamma^x(\epsilon)} L^{x\to y_n}}, \tag{13}$$

$$\eta^{x\to y_n} = \frac{L^{x\to y_n}}{1 - P_D P_G + \sum_{n\in\Gamma^x(\epsilon)} L^{x\to y_n}}, \tag{14}$$

where P_D is the target detection probability, P_G is the gate probability, and $L^{x\to y_n}$ is the likelihood of the target associating with measurement y_n as written in Eq. (15)

$$L^{x\to y_n} = \frac{e^{-0.5 w_e^T (HP^- H^T + S)^{-1} w_e}}{\sqrt{2\pi \det(HP^- H^T + S)}}, \tag{15}$$

where H is the Jacobian for the function h_e and P^- is the EKF prediction error covariance matrix. Figure 6 can help to understand the idea of this algorithm. Using this prediction method, it is possible to obtain an appropriate observation of the endpoint angle.

3.4 LiDAR-based false estimation removal

The existence of falsely estimated objects may affect the robot negatively, and thus, to deal with this problem, a LiDAR-based removal program was implemented. The estimated object in the map where high amounts of LiDAR laser passed through for a certain amount of time should be deleted. Consider the estimated laser distance $\hat{r}_i^j$:

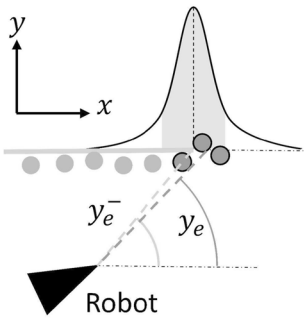


Fig. 6 PDAF modification of endpoint angle observation. The line is the predicted line map, dots represent the PC, the bell curve represents the Gaussian distributed association probability of point clouds with endpoint, outlined dots represent candidate endpoints, the triangle below represents the robot, the area under the bell curve is the verification region $\Gamma^x(\epsilon)$, y_e^- is the predicted endpoint angle computed by $h_e(x_p, m)$, and y_e is the modified endpoint angle by PDAF

$$\hat{r}_i^j = h_l(\hat{x}_p, \hat{m}^i, \phi^j), j \in \mathbb{P}^i, \tag{16}$$

where $\hat{r}_i^j$ expresses the jth estimated distance of the robot to the line map i, $\hat{x}_p$ is the estimated robot state, $\hat{m}^i$ is the estimated map's position state, and $\mathbb{P}^i$ is the PC associated with line map i denoted as

$$\mathbb{P}^i := \{n | n \in \mathbb{P}, \theta_a^i \le \delta n \le \theta_b^i\}, \tag{17}$$

where $\mathbb{P}$ represents the index set containing all PC index from the LiDAR sensor currently, n represents an individual PC index, δ denotes the LiDAR's constant increment angles from a laser to its neighboring laser, and θ_a^i and θ_b^i represent the minimum and maximum of the angle computed by the vector-valued function $h_e(x_p, m^i)$, respectively. With the estimated distance $\hat{r}_i^j$ defined, the removal consideration condition is written as below

$$y_l^j - \hat{r}_i^j \ge \tau(\hat{r}_i^j), \tag{18}$$

where y_l^j expresses the LiDAR's measurement distance at angle ϕ^j and $\tau(\hat{r}_i^j)$ is the threshold distance error as expressed by the square arctan function in Eq. (19),

$$\tau(\hat{r}_i^j) = \Omega_0 + \Omega_1(\tan^{-1}(\Omega_2 \hat{r}_i^j))^2, \tag{19}$$

where Ω_0, Ω_1 and Ω_2 are tunable constants. Figure 7 represents the function relationship with the respective constants. The minimum of τ is Ω_0 for $\hat{r}_i^j = 0$, whereas, as the estimated distance $\hat{r}_i^j$ increases, the threshold also increases until it converges to $\Omega_1\left(\frac{\pi}{2}\right)^2$. This threshold function is used to adapt

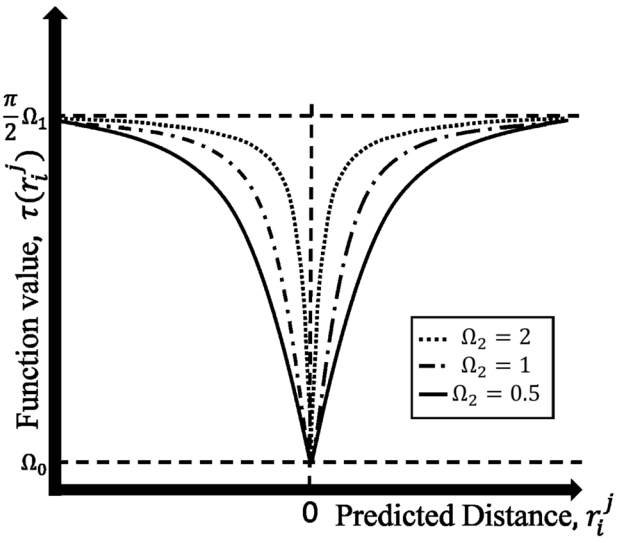


Fig. 7 The square arctan function. $\tau(r_i^j)$ value will converge at a certain point of r_i^j. This convergence prevents the threshold from increasing infinitely as $\hat{r}_i^j$ increases. Ω_0 and Ω_1 determine the lower and higher boundary of the function. The different lines represent the possible shapes of the function by tuning the weight Ω_2

the acceptable error based on the distance of the wall. Longer distance should have higher tolerance because of the increasing noise error. To prevent the threshold value from increasing indefinitely, this function is helpful. If the condition of Eq. (18) is satisfied for a certain amount of times, the following map state is removed.

3.5 Post-line map integration

In the final step, the algorithm modifies the estimated maps by matching and merging similar maps. At times, multiple estimated lines originate from the same object. To represent a coherent and consistent map, we will find among multiple line maps which are highly likely to associate with the same object by checking on these three conditions:

1. Distance between lines
2. Angle between lines
3. Overlapping between lines.

These criteria are conceptualized in Fig. 8. For the computation of distance d_i^j between line maps i and j, it is computed as below

$$d_i^j = |a_j x_{X,i} + b_j y_{X,i} + c_j|, \tag{20}$$

where $x_{X,i}$ and $y_{X,i}$ are the x and y position of the ith line map endpoint's state, respectively; a_j, b_j, and c_j are jth map state's line parameters that are being compared to map state

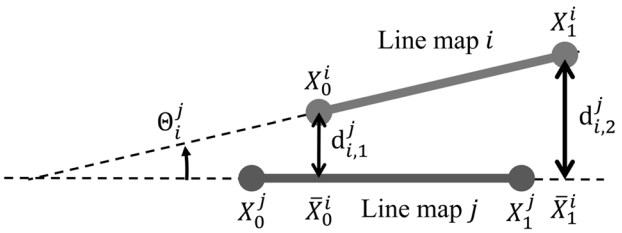


Fig. 8 Two line maps are being compared for integration. $d_{i,1}^j$ and $d_{i,2}^j$ are distance of endpoints X_0^i and X_1^i from line map j, Θ_i^j is the angle between these two lines, $\bar{X}_0^i$ and $\bar{X}_1^i$ are mapped positions to line map j from endpoint X_0^i and X_1^i, respectively

i. Since there are two endpoints, we will choose the minimum distance between them. Next, (a_i, b_i) and (a_j, b_j) are the normal for line i and j, respectively. Equation (21) indicates the angle between two lines φ_i^j

$$\varphi_i^j = \cos^{-1}(a_i a_j + b_i b_j). \tag{21}$$

Note that both Eqs. (20) and (21) are true if and only if a_i, a_j, b_i and b_j are the respective lines' unit normal vector. Finally, to determine overlapping between lines, the following overlapping conditions are checked:

- if $\bar{X}_0^i \le X_0^j \le \bar{X}_1^i$
- or $\bar{X}_1^i \le X_0^j \le \bar{X}_0^i$
- or $\bar{X}_0^i \le X_1^j \le \bar{X}_1^i$
- or $\bar{X}_1^i \le X_1^j \le \bar{X}_0^i$
- or $(X_0^j \le \bar{X}_0^i \le X_1^j \;\&\; X_0^j \le \bar{X}_1^i \le X_1^j)$
- or $(X_1^j \le \bar{X}_0^i \le X_0^j \;\&\; X_1^j \le \bar{X}_1^i \le X_0^j)$
- or $(\bar{X}_0^i \le X_0^j \le \bar{X}_1^i \;\&\; \bar{X}_0^i \le X_1^j \le \bar{X}_1^i)$
- or $(\bar{X}_1^i \le X_0^j \le \bar{X}_0^i \;\&\; \bar{X}_1^i \le X_1^j \le \bar{X}_0^i)$,

where X_0^j and X_1^j are jth line map endpoints state, respectively, and $\bar{X}_0^i$ and $\bar{X}_1^i$ are the ith line map endpoints state X_0^i and X_1^i that are being projected to the jth map state local coordinates, respectively, as shown below

$$\begin{bmatrix} \bar{X}_0^i \\ \bar{Y}_0^i \\ 1 \end{bmatrix} = \mathbf{T}^{-1} \begin{bmatrix} x_0^i \\ y_0^i \\ 1 \end{bmatrix}, \tag{22}$$

where T is the transformation matrix derived from the rotational matrix about the z-axis, and x_0^i and y_0^i are the global Cartesian coordinates of X_0^i. We rotate the global axis by $(\alpha^j - \frac{\pi}{2})$ to match the angle of line j and displace it for $\{d^j \cos\alpha^j, d^j \sin\alpha^j, 0\}$ as expressed below

$$\mathbf{T} = \begin{bmatrix} \sin(\alpha^j) & \cos(\alpha^j) & d^j \cos\alpha^j \\ -\cos(\alpha^j) & \sin(\alpha^j) & d^j \sin\alpha^j \\ 0 & 0 & 1 \end{bmatrix}. \tag{23}$$

If all conditions are met, we merge them by taking the map states average, and for the endpoint states X_0 and X_1, we will take the furthest to be the new endpoints.

4 MHE based SLAM considering the motion of dynamic objects

4.1 Moving horizon estimation

MHE is a model-based filter that estimates the optimal state by considering the measurements in the past time within a finite time length called horizon, H [21]. The MHE considers all optimization variables from the current time T until the end of the horizon $T-H+1$. This filter can consider constraints as it utilizes a numerical optimization approach. In this study, for the state's probability distribution, we assume them to be Gaussian. MHE estimates the maximum likelihood of it by minimizing the evaluation function below

$$\begin{aligned} J = & \sum_{k=T-H+1}^{T-1} \left\| \hat{x}_p[k+1] - f(\hat{x}_p[k], u_p[k]) \right\|^2_{Q_p^{-1}} \\ & + \sum_{k=T-H+1}^{T-1} \sum_{i=1}^{\mathbb{M}[k]} \left\| \hat{x}^i_m[k+1] - A\hat{x}^i_m[k] \right\|^2_{G^\dagger Q_m^{-1} (G^\dagger)^T} \\ & + \sum_{k=T-H+1}^{T} \frac{1}{\sigma_l^2} \sum_{j \in \Upsilon[k]} \left\{ y^j_l[k] - h_l(\hat{x}_p[k], \hat{m}[k], \phi^j) \right\}^2 \\ & + \sum_{k=T-H+1}^{T} \sum_{i \in \Psi[k]} \left\| y^i_e[k] - h_e(\hat{x}_p[k], \hat{m}^i[k]) \right\|^2_{S^{-1}} \\ & + \sum_{k=T-H+1}^{T-1} \sum_{i=1}^{\mathbb{M}[k]} \left\| \hat{\mu}^i_{X,1}[k] - \hat{\mu}^i_{X,0}[k] \right\|^2_{q_e^{-1}} \\ & + \| \hat{x}[T-H+1] - \hat{x}^-[T-H+1] \|^2_{P^-[T-H+1]^{-1}} \\ & + \sum_{k=T-H+1}^{T} \sum_{i=1}^{\mathbb{M}[k]} w_1 (\tan^{-1}(w_2 \hat{\mu}^i[k]))^2. \end{aligned} \tag{24}$$

The evaluation function has 6 Mahalanobis distance terms represented as the weighted 2-norm where weights are the respective term's noise covariance and a suppression term. In this section, we denote the decision variables as hatted symbols in the math equation, for example, $\hat{x}_p$. The first term evaluates the error of the robot's state where $\hat{x}_p$ is the optimized robot's pose. The second term evaluates the error of the map's state where $\mathbb{M}[k]$ is the total number of line maps, $\hat{x}_{m,i}$ is the optimized map's state, and $G^\dagger$ is the pseudo-inverse of matrix G. The third term evaluates the innovation of laser distance where $\Upsilon[k]$ is the index set for observed point clouds, σ_l is the laser observation noise standard deviation, y^j_l is the j-th point cloud distance, and $\hat{m}$ is the optimized map's position state. The fourth term evaluates the innovation of the line map's endpoints angle where $\Psi[k]$ is the index set for observed endpoints on a line map. The fifth term is the endpoint soft constraint term where $\hat{\mu}^i_{X,0}$ and $\hat{\mu}^i_{X,1}$ are the approximated velocities corresponding to the i-th endpoints X_0 and X_1, respectively, and q_e is the endpoint soft constraint noise variance. The sixth term is the arrival cost function where $\hat{x}$ is the optimized system state, $\hat{x}^-$ is the priori EKF state, and P^- is the EKF prediction error covariance. The last term is a velocity suppression term that suppresses objects with low speed where w_1 and w_2 are suppression weights and $\hat{\mu}^i$ is the i-th optimized map velocity states. We will explain each term in order.

Prediction error of robot's state

The first term evaluates the robot's state prediction error. Minimizing this will get the optimal estimate of x_p.

Prediction error of map's state

The second term evaluates the line segment's state prediction error. To decrease the computational cost of this term, the velocity state $\hat{\mu}$ except at current time T were not used as the optimization variable, because it can be determined by the position states d, α, X_0 and X_1. This is further clarified in Sect. 4.2.

Observation error of laser distance

The third term evaluates the LiDAR laser sensor distance innovation. With this innovation term, we can innovate the robot pose $\hat{x}_r$ and the map position state $\hat{x}_m$. However, this term alone is not enough to estimate the overall motion of the map state as it only accounts for the d and α states as written in Eq. (7); thus, an additional term that estimates movement of the map along the line segment is needed which is the next term.

Observation error of endpoints angle

The fourth term evaluates the endpoints' angle innovation from the robot and is essential to estimate the size of the line segment and its motion along the line segment. Since it innovates both line segments' endpoints, even if one of the endpoints is occluded, the observation of the other endpoint can correct the line map's estimation.

Endpoints soft constraint term

In this study as mentioned in Sect. 2.2, for a line map with both endpoints estimated, $\hat{\mu}_{X,0}$ and $\hat{\mu}_{X,1}$ should be equal as the line's length is constant. Both of them are determined as below

$$\hat{\mu}_{X,i}[k] = \frac{\hat{X}_i[k+1] - \hat{X}_i[k]}{\Delta}, \ (i = 0, 1). \tag{25}$$

We can assume both endpoints' velocity noise n_e to be Gaussian where $n_e \sim \mathcal{N}(0, q_e)$. Using the fifth term, we can keep the line map from expanding or contracting erratically due to outliers.

Arrival cost

This sixth term represents the error of the past states $\hat{x}^{-}$ with the optimized states $\hat{x}$ at the end of the Horizon $[T-H+1]$.

Conditional velocity suppression term

The algorithm considers the velocity state of all objects. Consequently, stationary objects will have low speeds due to noise. The last term suppresses those objects' speed to zero while not affecting entities with reasonably high speed. For that purpose, the square arc tangent function depicted in Fig. 9 is used. The function value does not significantly change for entities with relatively high speeds. Thus, high-speed objects' velocity remains unaffected by the suppression term, unlike objects with relatively low speed.

4.2 Eliminating the past velocity states

As explained in Sect. 4.1, we increased the computational speed of the MHE by removing the past velocity states as the optimization variable. This elimination is possible due to the assumption of our system where only velocity states contain noises in Eq. (3). Let us take the dynamics of d for example

$$\hat{d}[k+1] = \hat{d}[k] + \Delta\hat{\mu}_d[k]. \tag{26}$$

Since the position states do not contain noise, $\hat{\mu}_d[k]$ can be determined based on $\hat{d}[k]$ and $\hat{d}[k+1]$ as shown

$$\hat{\mu}_d[k] = \frac{\hat{d}[k+1] - \hat{d}[k]}{\Delta}. \tag{27}$$

This is also true for the other velocity states

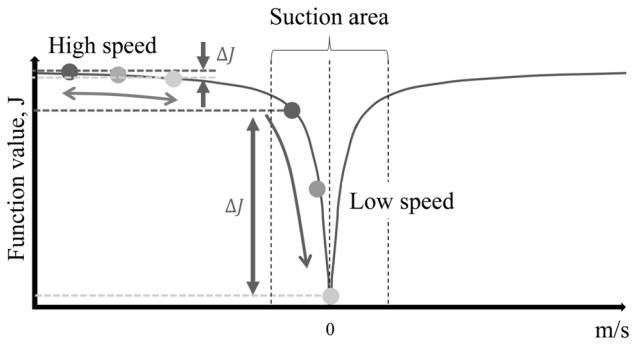


Fig. 9 Conditional velocity suppression term function. Top left dots represent relatively high-speed objects (HSO), dots within suction area represent low-speed objects (LSO), and ΔJ is the difference of function value, J. The low gradient part of the arctan function does not significantly affect the evaluation function, because ΔJ is negligible. However, the suppression function greatly minimizes the entities' speed within the suction area due to the significant ΔJ value. HSO will be affected more by other optimization terms, whereas LSO will be heavily influenced by the suppression term. Thus, this term can suppress the estimated velocity of LSO without affecting HSO

$$\hat{\mu}_\alpha[k] = \frac{\hat{\alpha}[k+1] - \hat{\alpha}[k]}{\Delta}, \tag{28}$$

$$\hat{\mu}_X[k] = \frac{\hat{\mu}_{X,0}[k] + \hat{\mu}_{X,1}[k]}{2}. \tag{29}$$

Equation (29) is the average of both endpoints' velocity. However, in cases where one of the endpoints is not observable, $\hat{\mu}_X[k] = \hat{\mu}_{X,0}[k]$ or $\hat{\mu}_X[k] = \hat{\mu}_{X,1}[k]$.

4.3 Parallelization of the split optimization function

To reduce optimization complexity, the previous approach involved splitting the optimization of the robot and line map states into separate parts. However, it was not enough to achieve real-time speed. To enhance the computational speed, we parallelize the optimization. It was impossible to parallelize at first due to the posterior map state depending on the posterior robot state. Therefore, we use the priori states to estimate the posterior states to make them independent. The parallelization can be depicted by Fig. 10.

5 Experiment types and setups

5.1 The experiment's environment

In this paper, we extensively evaluated the proposed algorithm's viability and performance through comprehensive on-site and numerical experiments. We thoroughly test the algorithm's repeatability and precision through numerical simulations and assess the algorithm's SLAM capacity in static and dynamic environments. In the static environment

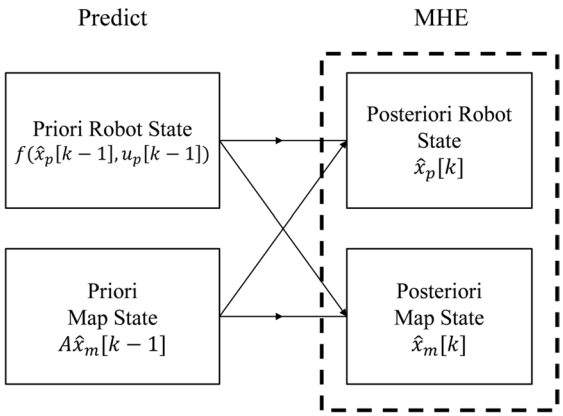
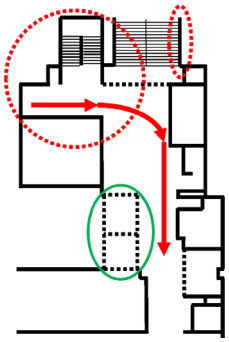
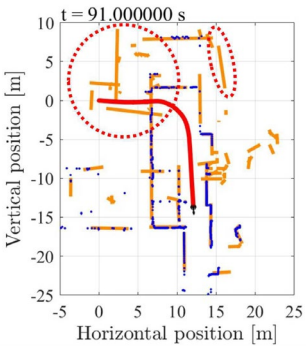


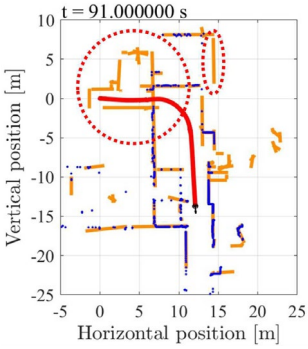
Fig. 10 The parallel method of MHE optimization. The priori robot state is constant when optimizing the posterior map state, while the priori map state is constant when optimizing the posterior robot state. This technique ensures robot and map states are independent during parallelization



(a) The traced onsite map layout of the university campus hallway, the dotted black lines represent glass walls, and the red curved arrow line is the robot wheelchair's transverse path. Due to LiDAR piercing the glass walls at the green circled area, the algorithm may not estimate the wall correctly. Within the red-circled area are areas of the map that are unobservable to the robot.



(b) SLAM simulation without velocity suppression term. The robot wheelchair is indicated by the black circle marker, the red curved line is the robot's path, the blue dots are the PCs and the orange lines represent the estimated walls of the building. Within the red-dotted circle area, unobservable walls slip away from their original positions due to low estimated speed.



(c) SLAM simulation with velocity suppression term. Within the red-dotted circle area, the velocity suppression term maintains the position of the unobservable walls.

Fig. 11 Comparison of map layout with SLAM estimation and the effects of velocity suppression term (color figure online)

experiment, we evaluate the performance of this algorithm in a campus hallway with many challenging features like uneven wall structure, the presence of glass walls, and others. We conduct the static SLAM experiment offline, allowing us to meticulously analyze and validate the algorithm's performance as a conventional SLAM. For the next experiment, we execute online SLAM in an environment featuring multiple moving objects, evaluating the algorithm's robustness and accuracy in estimating the dynamic objects position in real-life scenarios. During the experiment, we traverse the room in a wheelchair, encountering moving objects with different paths, thereby assessing the proposed algorithm's capability in highly dynamic environments.

5.2 MHE and sensor's setting parameters

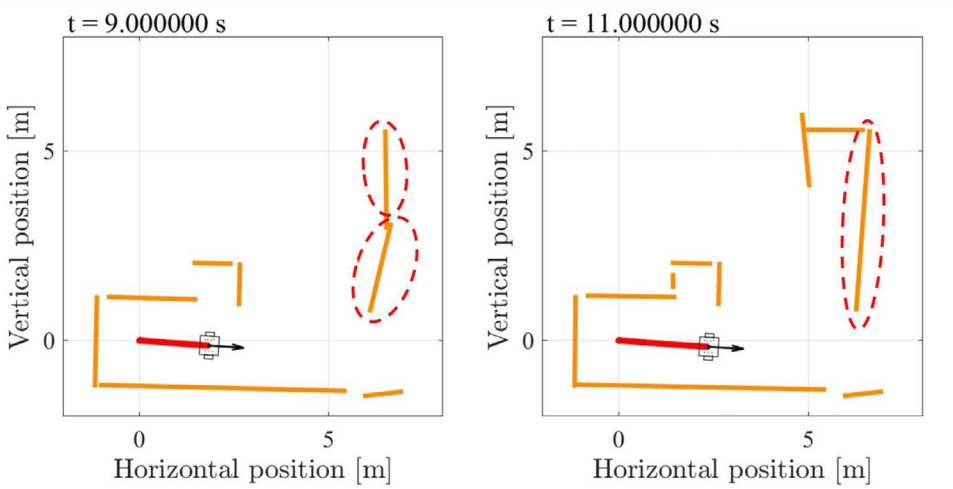
In this SLAM, the parameters are set as in Table 1. Depending on the environment and system, the MHE setting should be tuned accordingly. H is set to 8 as it was the optimal value balancing accuracy and computational time for online implementations. Q_p is set with low values, because the odometry is reliable in indoor flat environments. Q_m is set with relatively larger values as we do not have information on the map's internal velocity state. σ_l is set based on the LiDAR sensor's accuracy which has around ±5 mm error. S is set based on the horizontal observation angle resolution around ±0.3°. q_e is set with very low value to maintain the length of line maps with both endpoints estimated. w_1 and w_2 are set where velocity above 0.1 m/s will not be suppressed. P_D is set where we assume 80% of the time we can detect

Table 1 MHE parameters

Parameter	Value
H	8
Q_p	Diag(10^{-5},10^{-5},10^{-8}))
Q_m	Diag(0.082,0.001,0.082))
σ_l	0.005
S	Diag(2.704×10^{-5},2.704×10^{-5})
q_e	10^{-6}
w_1	0.0811
w_2	10^3
P_D	0.8
P_G	0.7
ϵ	1.0742
Ω_0	0.2
Ω_1	$2/\pi$
Ω_2	0.1

measurement around the line's endpoint. P_G is set where we assume 70% of the correct measurement lies in the gating region bounded by epsilon ϵ. In the removal algorithm, we set Ω_0, Ω_1 and Ω_2 where minimum distance threshold error is 0.2 m and maximum is 1 m depending on the laser's distance.

We used the VLP-16 of Velodyne as the laser range sensor. However, we only took the laser measurements at the elevation angle of +1° to decrease the dimensionality of the cloud point measurement into 2-dimension.



(a) In $t = 9\,\mathrm{s}$, two line maps circled red were estimated even though the measurement came from the same wall.

(b) In $t = 11\,\mathrm{s}$ through the merging post process, the SLAM successfully merges the line maps into the one circled red.

Fig. 12 Merging of estimated line maps during the university campus SLAM (color figure online)

6 SLAM algorithm enhancement results

6.1 Improved map using velocity suppression term

As introduced in Sect. 4, we implement the novel velocity suppression term to address the problem of low-speed estimation in static objects. We conducted experiments in the university hallway to validate the algorithm's performance in complicated static environments. Figure 11 depicts the results of the SLAM experiment. Figure 11a illustrates the traced layout of the on-site map. Subsequently, we compare two variations of SLAM execution: one without velocity suppression terms in Fig. 11b and the other with velocity suppression terms in Fig. 11c. In Fig. 11b, the unobserved walls have shifted from their positions due to their low estimated speed, highlighting the problem of estimating every object as a moving object. However, in Fig. 11c, we can observe the suppression term effectively reducing the walls' speed, thus, maintaining the walls' position and improving the estimation. Proper adjustments of weights can allow suppression for only low-speed objects such as walls and not affect moving objects. The presented results emphasized the importance of applying velocity suppression terms in improving the estimation's accuracy of static objects.

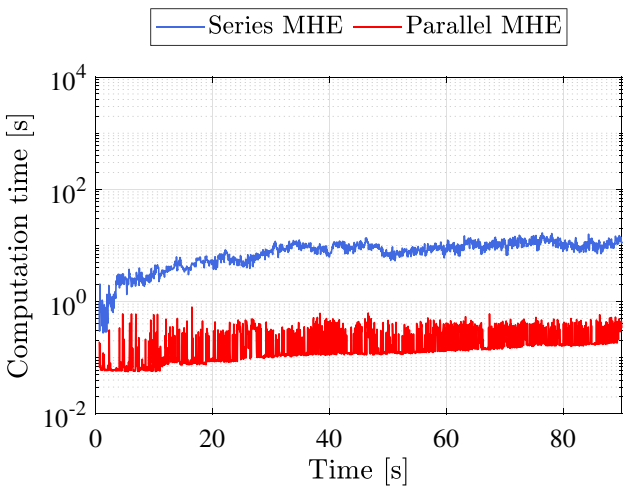
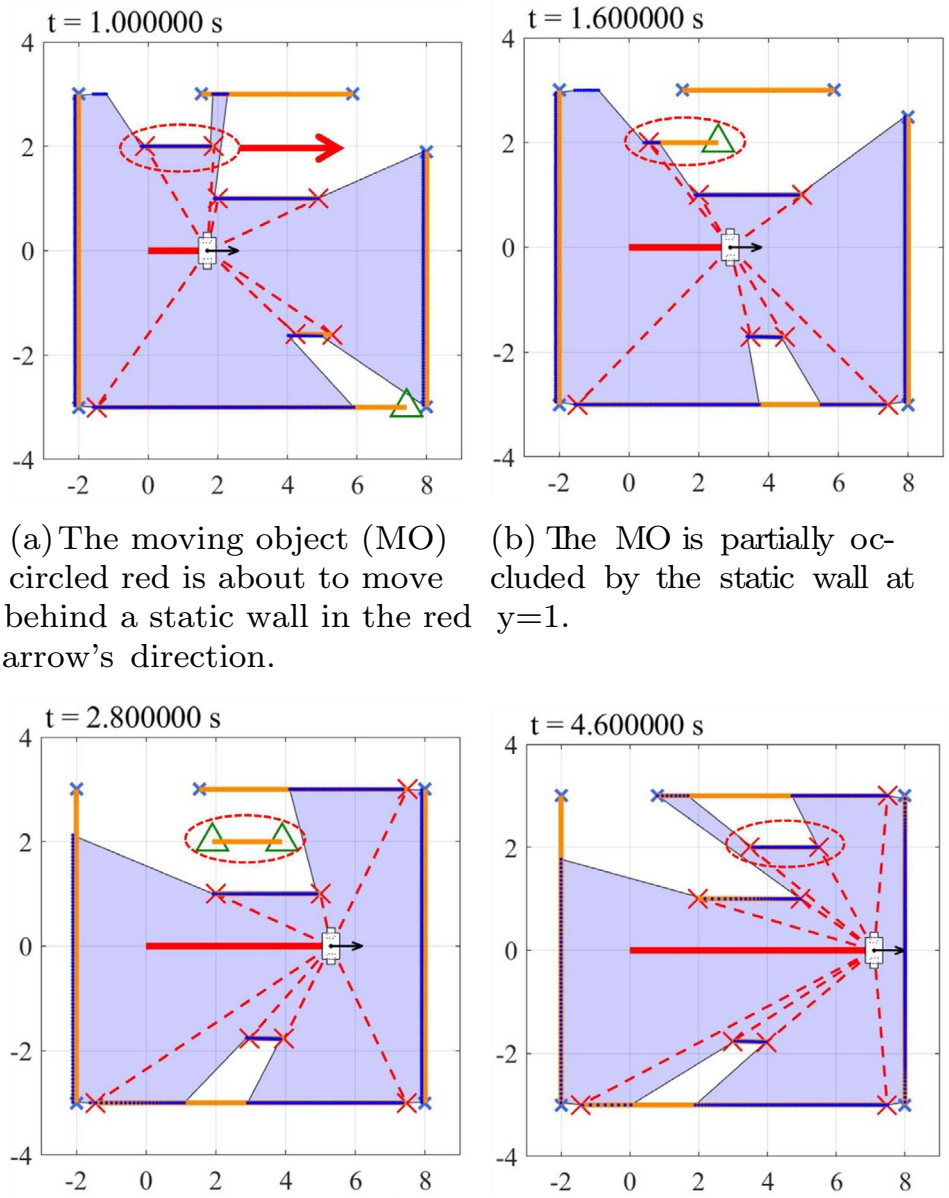


Fig. 13 Time comparison of the series (blue) and parallel (red) methods (color figure online)

6.2 Merging algorithm for line maps

SLAM sometimes estimates multiple line maps from the same wall due to occlusion or measurement noises. The algorithm will merge these estimated maps. Figure 12 demonstrates a successful merging during the university campus SLAM experiment. As the robot moves further, more point clouds are observed, and a new line map is estimated. Since the merging conditions explained in Sect. 3.5 are met, the two line maps merged. This method improves the SLAM map and prevents map corruption.



(a) The moving object (MO) circled red is about to move behind a static wall in the red arrow's direction.

(b) The MO is partially occluded by the static wall at y=1.

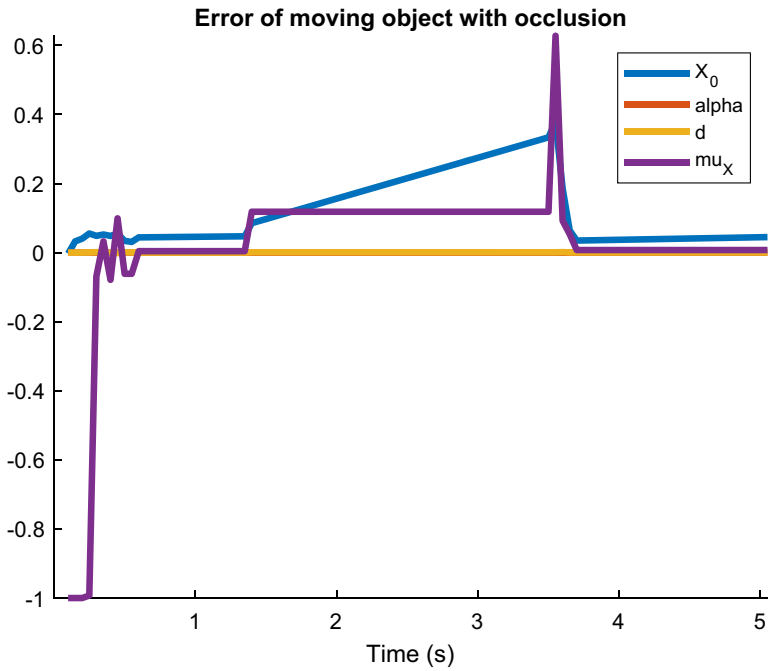
(c) Even when MO is fully occluded, the state is estimated based on previous motion.

(d) Blue PC clusters that belong to the red-circled MO emerged from behind the y=1 wall. Since it is accurately estimated, the PC is correctly associated with the MO.

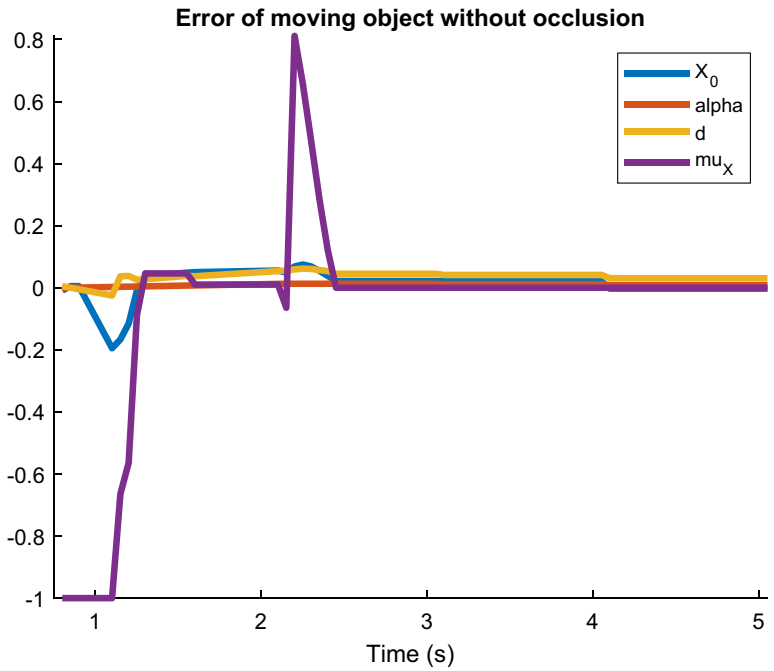
Fig. 14 Simulation results of SLAM executed with multiple moving obstacles. One moving object will be occluded as it traverses, while one is not. The robot is represented by the small box-shaped object with an arrow, orange lines are estimated maps, blue dots hovering above the orange lines are PC, the blue area is the observable area, the red X icon at the end of lines indicates endpoints are observable, the green triangle indicates unobservability, and dotted purple line are the sights from robot to the observed endpoints (color figure online)

6.3 Parallelization of MHE

For online implementations, the optimization problem is computationally demanding. To resolve this, we divide them into manageable individual subproblems and optimize them concurrently as written in Sect. 4.3. Figure 13 compares the proposed parallel and series optimization methods. Both methods increases computational time linearly as more objects are estimated. However, by parallelizing the optimization process of the robot and map, the average time for the optimization problem process decreases drastically from 7.92 to 0.21 s enhancing the computational speed by around 37 times. This improvement in computational cost allows online implementations in a limited framework.



(a) The error of d, α, X and μ_X states for moving object with occlusion.



(b) The error of d, α, X and μ_X states for moving object without occlusion.

Fig. 15 Comparison of moving objects' state error with and without occlusion (blue: X, red: α, yellow: d, and purple: μ_X) (color figure online)

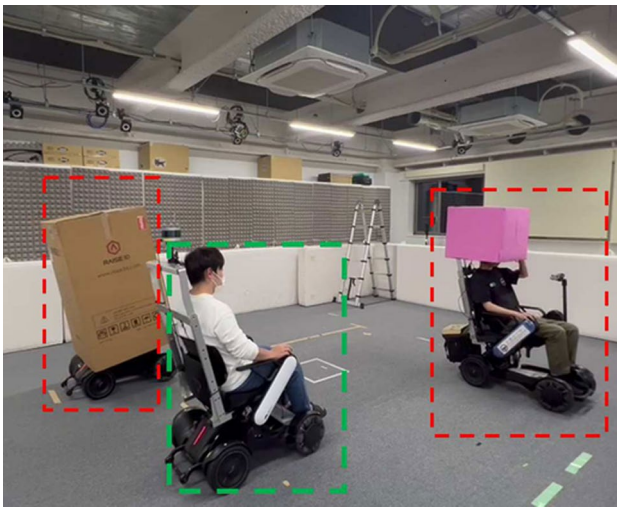
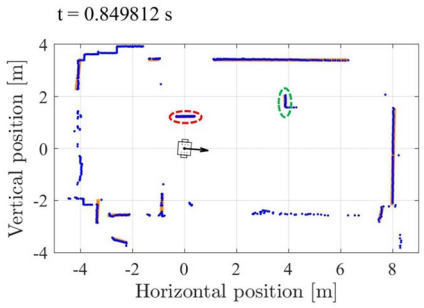


Fig. 16 The real-time experiment with the robot electric wheelchair (green dotted box) in the center and 2 moving obstacles moving on its left and front. Boxes were attached to both moving objects to prevent irregularly shaped PC clusters, because the algorithm can only estimate straight-line models (color figure online)

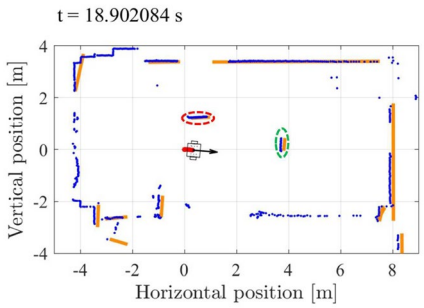
7 Moving object experiments

7.1 Numerical simulation experiment

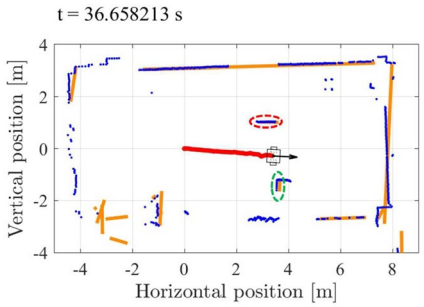
To ensure repeatability and accuracy, we validated our algorithm in a numerical simulation experiment where multiple moving and stationary objects are present with occlusion scenarios. Figure 14 displays the simulation results and demonstrates the successful estimation of the proposed SLAM method, even under partial or complete occlusion. In Fig. 15, we compare the state error of the occluded and non-occluded moving objects to evaluate the accuracy. Figure 15a denotes that state error gradually increases when the moving object is occluded. However, because the prediction is close, the MHE can correct the deviation after the measurements' re-observance. The μ_X state error is high in Fig. 15a, at $t = 3.55$ s, due to the innovation after the re-observance of the point clouds. On the other hand, in Fig. 15b, for the non-occluded object, the error of all states remains minor except at $t = 2.2$ s, where the error of $\mu_X = 0.8$ m/s. The reason for such a spike is due to the sudden stop of the moving object, shown in Fig. 14c, d at the position between points (3.0, 1.9) and (4.0, 1.9). Since the map state prediction uses a linear velocity model, the estimation is sensitive to sudden direction changes. The simulation proves the proposed SLAM's benefits as it can accurately estimate both moving and static objects even under occlusion and validates its potential for real-world applications.



(a) Moving objects were observed to the left of the robot at t = 0 .85 s around (0 .0, 1.2) and (3 .9, 2.0) circled red and green, respectively.



(b) The robot continues moving in a straight line while estimating both moving objects. The red-circled object moves right while the green-circled object moves downwards.



(c) The moving objects are successfully estimated until the robot stops.

Fig. 17 The online SLAM experiment with the presence of moving obstacles. The robot is represented by the square with the arrow, the orange line represents the estimated map, the blue dots represent LiDAR PC, and two moving objects are circled red and green (color figure online)

7.2 Online SLAM experiments with moving objects

We executed online SLAM within the presence of moving objects to validate the performance and effectiveness of our

proposed method in real-life scenarios. The average sampling time Δ is 0.7 s. The experiment and result of SLAM are shown in Figs. 16 and 17, respectively. As shown in Fig. 16, we attached boxes to the moving objects, because our algorithm could only detect straight-line PC clusters. Remarkably, even though the two moving objects traverse in different directions, the algorithm can successfully estimate their position. The map tracking of the moving objects exhibits high precision as the edge angle observation is appropriately modified using association probabilities to correct innovations. Notably, the room walls were also successfully estimated while simultaneously estimating the moving objects, proving that the velocity suppression term does not adversely affect relatively high-speed objects. The results show that with our proposed algorithm, online dynamic SLAM is possible by estimating the velocity state of the moving objects.

7.3 Long-term performance

Considering long-term performance, depending on the environment, the algorithm is expected to maintain a consistent map by estimating both moving and static objects with the proposed method. In rare cases of spurious maps appearing, the post-processing of the SLAM could deal with them by deleting or merging maps to prevent SLAM corruption. While the algorithm continuously updates new map states without limit as it traverses the environment, this could increase computational demands in the long term. This can be seen in Fig. 11 where the past unobservable maps are still being estimated by MHE. The system may experience performance issues, potentially leading to a collapse. Regarding estimation accuracy, SLAM in an environment with many uneven structures might also cause it to fail, because it does not have the proper model to address them. Fast-moving objects may also not be accurately estimated, since the current average computational time is around 0.21s. We recommend to apply the algorithm in a relatively simple environment to maintain a computationally feasible number of estimated objects for this method to work online. However, note that this study only applies line models to achieve our research objectives and is not limited to using only line models. Alternatives such as deleting past distant objects when it becomes too heavy to compute should be considered.

8 Conclusion

In this paper, we proposed a novel SLAM that explicitly estimates the velocity of all objects and effectively suppressed the speed of static objects. The proposed SLAM could estimate robustly in dynamic environments without distinguishing between stationary and moving objects. The uniform nature of assuming all objects as moving objects prevents map corruption due to the misassumption of an object's nature compared to traditional methods that distinguish stationary and moving objects. By leveraging MHE's numerical optimizer, the velocity suppression term successfully suppressed the speed of static walls while treating all objects as dynamic objects, improving estimation performance. Additionally, estimating moving objects is possible even under occlusion by considering past measurements through MHE. Eliminating past velocity optimization variables through mathematical derivations and parallelization of the MHE objective function accelerated the computational time, allowing online implementations of SLAM in a limited framework.

It is noted that, in this study, we only applied a line model for this algorithm to verify our proposed method. Moving objects such as pedestrians should use other models or algorithms for proper estimation. The continuous estimation of new objects can increase computational time, potentially making it too heavy for online use. The reason is because the SLAM algorithm deals and estimates all objects regardless if it is observable or not. An alternative to prevent that is by deleting past or distant objects to save computational cost. In future work, we will apply the SLAM with new models to detect objects of any shape, especially pedestrians.

Acknowledgements This work was supported by JSPS KAKENHI under Grant No. JP 19H02098.

Data availability The data is available and can be requested from the corresponding authors.

References

1. Cadena C, Carlone L, Carrillo H, Latif Y, Scaramuzza D, Neira J, Reid I (2016) Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans Rob 32(6):1309–1332
2. Dissanayake G, Durrant-Whyte H, Bailey T (2000) A computationally efficient solution to the simultaneous localisation and map building (SLAM) problem. In: Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia Proceedings (Cat. No.00CH37065), vol 2, pp 1009–1014
3. Mur-Artal R, Tardós JD (2017) ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Trans Rob 33(5):1255–1262
4. Forster C, Pizzoli M, Scaramuzza D (2014) SVO: fast semi-direct monocular visual odometry. IEEE Int Conf Robot Autom (ICRA) 2014:15–22
5. Walcott-Bryant A, Kaess M, Johannsson H, Leonard JJ (2012) Dynamic pose graph SLAM: long-term mapping in low dynamic environments. IEEE/RSJ Int Conf Intell Robot Syst 2012:1871–1878
6. Hahnel D, Triebel R, Burgard W, Thrun S (2003) Map building with mobile robots in dynamic environments. In: 2003 IEEE

international conference on robotics and automation, vol 2, pp 1557–1563
7. Demim F, Nemra A, Boucheloukh A, Louadj K, Hamerlain M, Bazoula A (2018) Robust SVSF-SLAM algorithm for unmanned vehicle in dynamic environment. In: 2018 international conference on signal, image, vision and their applications (SIVA). IEEE, pp 1–5
8. Li A, Wang J, Xu M, Chen Z (2021) DP-SLAM: a visual SLAM with moving probability towards dynamic environments. Inf Sci 556:128–142
9. Vu TD, Burlet J, Aycard O (2011) Grid-based localization and local mapping with moving object detection and tracking. Inf Fusion 12(1):58–69
10. Bescos B, Campos C, Tardós JD, Neira J (2021) DynaSLAM II: tightly-coupled multi-object tracking and SLAM. IEEE Robot Autom Lett 6(3):5191–5198
11. Lin KH, Wang CC (2010) Stereo-based simultaneous localization, mapping and moving object tracking. In: 2010 IEEE/RSJ international conference on intelligent robots and systems, pp 3975–3980
12. Wang CC, Thorpe C, Thrun S, Hebert M, Durrant-Whyte H (2007) Simultaneous localization, mapping and moving object tracking. Int J Robot Res 26(9):889–916
13. Einhorn EE, Gross HM (2015) Generic NDT mapping in dynamic environments and its application for lifelong SLAM. Robot Auton Syst 69:28–39
14. Bahraini MS, Bozorg M, Rad AB (2018) SLAM in dynamic environments via ML-RANSAC. Mechatronics 49:105–118
15. Pfreundschuh P, Hendrikx HF, Reijgwart V, Dubé R, Siegwart R, Cramariuc A (2021) Dynamic object aware lidar slam based on automatic generation of training data. In: 2021 IEEE international conference on robotics and automation (ICRA). IEEE, pp 11641–11647
16. Burgard W, Brock O, Stachniss C (2008) Simultaneous localisation and mapping in dynamic environments (SLAMIDE) with reversible data association. Robotics: science and systems III. MIT Press, Cambridge, pp 105–112
17. Rao CV, Rawlings JB, Mayne DQ (2003) Constrained state estimation for nonlinear discrete-time systems: stability and moving horizon approximations. IEEE Trans Autom Control 48(2):246–258
18. Garulli A, Giannitrapani A, Rossi A, Vicino A (2005) Mobile robot SLAM for line-based environment representation. In: Proceedings of the 44th IEEE conference on decision and control. IEEE, pp 2041–2046
19. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395
20. Bar-Shalom Y, Daum F, Huang J (2009) The probabilistic data association filter. IEEE Control Syst Mag 29(6):82–100
21. Kikuchi T, Nonaka K, Sekiguchi K (2020) Moving horizon estimation with probabilistic data association for object tracking considering system noise constraint. J Robot Mechatron 32(3):537–547

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.