2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Practical Framework for Path Representation and Following Control in Mobile Industrial Robots

Youngil Koh[1] Woojeong Kim[2] and MidEum Choi[3]

Abstract— **This paper presents a practical framework for path representation and following control for mobile industrial robots. Mobile industrial robots perform various missions, such as navigating predefined routes and transporting cargo in industrial environments. Unlike traditional AGVs that rely on physical tapes installed densely along routes, our framework supports the generation of drivable paths by connecting waypoints specified by users. The proposed framework generates drivable work paths while considering the permissible deviation between waypoints and the path, as well as the maximum curvature, without nonlinear optimization techniques. Additionally, we introduce a "stopover" attribute for each waypoint to enhance usability in narrow workspaces. Furthermore, we have developed a practical path following control system that takes into account real-world challenges such as actuation delay and computational efficiency. The proposed speed planner incorporates predictive information without relying on optimization techniques, and supports robust following performance by braking intervention if path deviation increases. The effectiveness of the proposed framework has been validated through robot tests, demonstrating accurate path following performance and efficient computation.**

I. INTRODUCTION

Mobile robots have received significant attention for their various applications. Especially in logistics warehouses and production facilities, mobile robots that transport goods have increased productivity by navigate along the predefined paths, thus enhancing the level of automation in manufacturing processes.

Industrial mobile robots, AGVs (Automated Guided Vehicle), typically follow work paths designated by densely attached magnetic tapes on the floor of workspaces. While this system offers high reliability in following designated paths, maintaining the magnetic tapes is labor-intensive, when factory layouts are modified. However, autonomous mobile robots, equipped with localization and mapping techniques, mitigate these inefficiencies by following a reference path specified on a map.

While service robots autonomously plan the optimal route, once the destination is set, industrial mobile robots require specifically defined work paths. We assume a scenario where users specify a waypoint sequence on a map to designate the robot's work path. Accordingly, we have developed a practical method to represent drivable paths based on sparsely specified waypoint sequences.

A B-spline-based curvature relaxation method using given waypoint sequence was introduced in [1]. The method adds additional waypoints to reduce curvature when the generated path exceeds the allowable limit. However, this approach may lead to undesirable alternations in the user-specified waypoint sequence. The undesirable alternation makes the approach challenging to use in industrial workspaces where strict adherence to the user-specified waypoint sequence is required.

We also addressed the following control system from a practical perspective. Among the various control methods, pure pursuit is one of the simplest and powerful method. Once the target point to pursue is determined, the required curvature can be calculated geometrically. However, there is a significant performance trade-off issue regarding how far ahead to choose the target point [2].

In the case of autonomous vehicles, the linear quadratic regulator (LQR) method is widely applied for path following [3]. However, it requires linearization of the nonlinear system, making it challenging to identify modeling parameters such as cornering stiffness. Additionally, the tuning of weight parameters for each state and input variable is challenging, and gain scheduling based on speed may be necessary for optimal performance.

Model Predictive Control (MPC) is another widely used for path following [4], [5], which offers the advantage of achieving stable performance based on prediction of the dynamics. The method is based on optimization techniques, thereby, it is challenging to straightforwardly incorporate various constraints, and it also requires a considerable amount of computation time for complex problems [2].

Many studies on path following have primarily addressed steering control issues. However, the sophistication of the velocity planner significantly impacts following performance. [6] proposed a robust steering controller for varying speeds but did not address velocity planning to support stable following performance. [7] proposed a real-time velocity planning method for stable path following by incorporating the curvature of the forward path. However, it segmented the path and suggested a method to maintain a constant speed for local maximum curvature.

In our proposed control framework, a simple lateral controller is employed to converge path deviation to a critically damped response. In cases where path deviation occurs due to real-world issues such as actuation delay, the proposed speed planner is intended to apply braking intervention to

[1]Youngil Koh is an engineer at the Robot Center, Samsung Research, Samsung Electronics, 56, Seongchon-gil, Seocho-gu, Seoul, 06765, Korea youngil.koh@samsung.com

[2]Woojeong Kim is an engineer at the Robot Center, Samsung Research, Samsung Electronics, 56, Seongchon-gil, Seocho-gu, Seoul, 06765, Korea wj124.kim@samsung.com

[3]MidEum Choi is an engineer at the Robot Center, Samsung Research, Samsung Electronics, 56, Seongchon-gil, Seocho-gu, Seoul, 06765, Korea me1.choi@samsung.com

stabilize the robot. Additionally, the predictive information, such as preview corner-entry speed and clearance to obstacles, is incorporated without optimization-based predictive control methods.

We validated our proposed path representation and following control framework on a mobile robot using S-shaped and R-shaped paths. The proposed framework, ensuring real-time capability, generated a smooth and drivable path, and accurate following performance.

The main contributions of our study are as follows:

- We propose a practical and straightforward path following framework suitable for industrial robots, encompassing both path representation and following control. Through rigorous testing with robots, we have validated the accurate following performance and super light computation capabilities.
- We propose a path representation method that incorporates constraints on allowable deviation and maximum curvature without optimization techniques. Additionally, we introduce a "stopover" attribute for each waypoint, which users can configure to facilitate efficient work path generation in narrow passages.
- We propose a speed planner that incorporates predictive information, including preview curvature and clearance to obstacle, etc. without relying on optimization techniques. Moreover, the proposed speed planner enhances the robustness of path following performance by implementing braking intervention in order to overcome the path deviation caused by actuation delay.

II. Path Representation

A. Preliminary

Industrial robots repeatedly traverse designated work paths to execute assigned tasks. To specify the robot's work path, users can designate waypoints at key locations within the workspace, taking into account the placement of obstacles. We propose a practical path representation method to generate the navigation path from the user-defined waypoint sequence. A waypoint sequence can be represented as follows.

$$\mathcal{W} = \{\mathbf{p}_{w.i} | i = 0, \cdots, n_w\} \tag{1}$$

where, $\mathbf{p}_{w.i} = [x_{w.i}, y_{w.i}]^T \in \mathbb{R}^{2\times 1}$ is the ith waypoint

We considered several basic requirements for the navigation path in industrial workplaces:

First, the navigation path must be bounded within a convex hull defined by the given waypoint sequence. Otherwise, the navigation path may violate obstacles and also be contrary to the user's intuitive intention. Therefore, "flyover" transition is not allowed at the waypoints except the start and end waypoint.

Second, users may need to adjust the positions of some waypoints in response to changes in obstacle placement in the workspace. We considered that such partial waypoint modifications should not significantly affect the overall navigation path.

Third, the navigation path must start at the first waypoint and end at the last waypoint. This is because the first and the last waypoints typically correspond to the initial and final locations of the mission process, respectively.

Fourth, the curvature of the navigation path must change continuously. Sudden changes in curvature can make the robot's motion unnatural, so the navigation path must satisfy the C^2 continuity condition in all segments.

Fifth, the navigation path is not allowed to deviate beyond a specific range from each given waypoint.

To satisfy the above requirements, we represented the navigation path using the clamped cubic B-spline method. B-splines can be represented by a control point sequence and a knot sequence, and a point on the B-spline can be obtained by a weighted sum of control points as follows:

$$\mathbf{p}_b(u) = \sum_{i=0}^{n_c} \mathbf{p}_{c.i} \cdot N_i^k(u) \tag{2}$$

where, $\mathbf{p}_b(u)$ is a point on the B-spline path at the path parameter $u \in [0, 1]$. $\mathbf{p}_{c.i} \in \mathcal{C}$ is the ith control point. k is a degree of B-spline. For the cubic B-spline, $k = 3$ as the minimum degree of B-spline that ensures C^2 continuity across all segments.

The weight, $N_i^k(u)$, corresponding to the control point, $\mathbf{p}_{c.i}$, is referred to as a basis function. The value of the basis function at a specific path parameter, u, is calculated using a recursive formula using the knot sequence, $\mathbf{u}$.

$$N_i^k(u) = \frac{u - u_i}{u_{i+k} - u_i} \cdot N_i^{k-1}(u) + \frac{u_{i+k+1} - u}{u_{i+k+1} - u_{i+1}} \cdot N_{i+1}^{k-1}(u) \tag{3}$$

$$N_i^0(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{else} \end{cases} \tag{4}$$

The exception has to be made as $N_{n_c}^0(u_{n_c+1}) = 1$.

The knot sequence, $\mathbf{u}$, can be defined as follows:

$$\mathbf{u} = \{u_0, u_1, \cdots, u_{n_c+k+1}\} \tag{5}$$

For the clamped B-spline, the multiplicity of the boundary knots is as follows:

$$u_0 = u_1 = \cdots = u_k = 0 \tag{6}$$

$$u_{n_c+1} = u_{n_c+2} = \cdots = u_{n_c+k+1} = 1 \tag{7}$$

The internal knots, $\{u_{k+1}, u_{k+2}, \cdots, u_{n_c}\}$, are set to be equidistant non-decreasing sequence.

One of the most powerful advantages of B-splines is to easily calculate exact derivative values at specific path parameter. Particularly, the first derivative at a specific location represents the path tangent, serving as the reference heading for the robot. The first derivative, $\mathbf{p}_b'(u) \in \mathbb{R}^{2\times 1}$, can be calculated as follows:

$$\mathbf{p}_b'(u) = \sum_{i=0}^{n_c-1} \underbrace{\frac{k \cdot (\mathbf{p}_{c.i+1} - \mathbf{p}_{c.i})}{u_{i+k+1} - u_{i+1}}}_{\mathbf{q}_i} \cdot N_{i+1}^{k-1}(u) \tag{8}$$

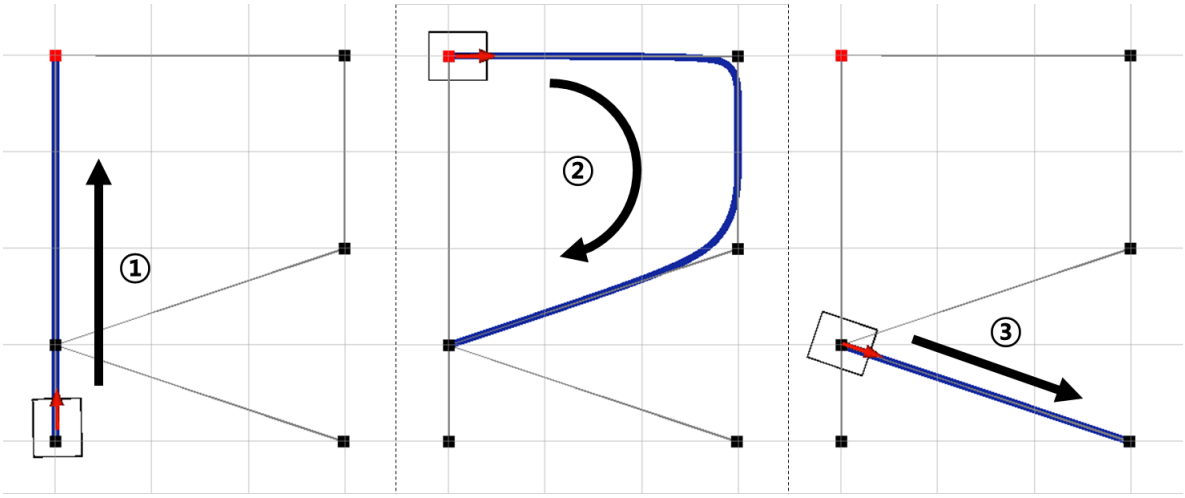

Fig. 1. Path generation with the R-shaped waypoint sequence. A red square represents a stopover waypoint, and black squares stands for "pass-by" waypoints. Generated paths are illustrated as a blue curve. The entire paths are segmented at both the stopover waypoint and a waypoint where the curvature exceeds the maximum limit.

The second derivative, along with the first derivative, can be used to accurately determine the reference curvature. The exact curvature information of the desired path is valuable for path following controllers. The burden of deviation compensation can be reduced and the rapid response can also be achieved from the exact curvature information. The second derivatives, $\mathbf{p}_b''(u) \in \mathbb{R}^{2\times 1}$, can be calculated as follows:

$$\mathbf{p}_b''(u) = \sum_{i=0}^{n_c-2} \frac{(k-1)\cdot(\mathbf{q}_{i+1}-\mathbf{q}_i)}{u_{i+k+1}-u_{i+2}} \cdot N_{i+2}^{k-2}(u) \tag{9}$$

From the first and second derivative, the curvature at the path parameter u can be calculated as follows:

$$\kappa(u) = \frac{\mathbf{p}_b'(u) \times \mathbf{p}_b''(u)}{\|\mathbf{p}_b'(u)\|^3} \tag{10}$$

B. Stopover waypoint

To facilitate efficient work path generation in narrow passages, we introduce the additional attribute of "stopover" for waypoints that users can configure. When the robot reaches a stopover waypoint, it plans the path to the next stopover waypoint and resumes movement. At this point, the robot comes to a complete stop at the stopover waypoint, rotates to the next waypoint, and then begins to move again. This approach enables efficient navigation in narrow working environments with dead ends and similar challenges.

Therefore, the path represented using B-splines targets a subset of the entire waypoint sequence. The sequence of path generation utilizing a stopover waypoint is illustrated in Fig. 1. The user designated the waypoint sequence in an R-shaped pattern, and the robot started to move from the bottom-left waypoint. In total, three times of path generations were made. The initially generated path is represented by a straight line up to the stopover waypoint indicated in red, as shown in the leftmost figure.

The second path wasn't planned all the way to the last waypoint despite the absence of any more stopover waypoints. This because the curvature at the second-to-last waypoint exceeded the allowable range. Our proposed path representation method designates a specific waypoint as a "stopover" if the curvature at that waypoint exceeds the allowable range, ensuring stability in path following.

C. Path refinement

In order to apply the fifth requirement, we propose a practical iterative algorithm to generate navigation path without nonlinear optimization method. If the path deviates from a specific waypoint beyond the allowable range, we insert a midpoint pair in the segments before and after that waypoint to satisfy the constraint. Accordingly, the control point sequence, $\mathcal{C}$ for B-spline is an union set of the user-defined waypoint sequence $\mathcal{W}$ and the inserted midpoints $\mathcal{M}$.

This proposed method differs from the method in [1] because it inserts midpoints only at the center of adjacent waypoints and modifies the user-defined waypoints.

The proposed method adjusts positions of the midpoints iteratively until the deviation distance and the curvature converges to below a specified value. The allowable maximum curvature can be defined as follows:

$$\kappa_{max} = \frac{\omega_{max}}{|v|_{min}} \tag{11}$$

where, ω_{max} and $|v|_{min}$ represent the user-defined maximum angular velocity and minimum absolute linear velocity, respectively.

The allowable deviation distance is referred to as the passing tolerance. The deviation distance between each waypoint and the path is defined as the distance to the projected point from the waypoint. A path parameter $u_{w.i}$ for the projected point $\mathbf{p}_b(u_{w.i})$ corresponding to the waypoint can be computed as follows:

$$u_{w.i} = \underset{u\in[0,1]}{\operatorname{argmax}} N_i^k(u) \tag{12}$$

Algorithm 1: Computation of midpoint pair positions

Input: target waypoint ($\mathbf{p}_{w.i}$), control point sequence for B-spline ($\mathcal{C}$), passing tolerance (d_{th}), maximum curvature (κ_{max})
Output: midpoint pair $\{\mathbf{p}_{m.l}, \mathbf{p}_{m.r}\}$ for $\mathbf{p}_{w.i}$

$\mathbf{p}_{m.l} \leftarrow \mathbf{p}_{w.i}$, $\mathbf{p}_{m.r} \leftarrow \mathbf{p}_{w.i}$
$\mathbf{u}_{prev} \leftarrow \overrightarrow{\mathbf{p}_{w.i}\mathbf{p}_{w.i-1}} / \|\overrightarrow{\mathbf{p}_{w.i}\mathbf{p}_{w.i-1}}\|$
$\mathbf{u}_{nest} \leftarrow \overrightarrow{\mathbf{p}_{w.i}\mathbf{p}_{w.i+1}} / \|\overrightarrow{\mathbf{p}_{w.i}\mathbf{p}_{w.i+1}}\|$
$l \leftarrow \min\left\{\|\overrightarrow{\mathbf{p}_{w.i}\mathbf{p}_{w.i-1}}\|, \|\overrightarrow{\mathbf{p}_{w.i}\mathbf{p}_{w.i+1}}\|\right\}$
for $1 \leq i \leq$ *max_iter* **do**
  $path$ = build a path from $\mathcal{C} \cup \{\mathbf{p}_{m.l}, \mathbf{p}_{m.r}\}$
  $\mathbf{p}$ = compute a projected point on $path$ from $\mathbf{p}_{w.i}$
  $d = \|\mathbf{p} - \mathbf{p}_{w.i}\|$
  if $d <= d_{th}$ **then**
    if $i \leq 2$ *or* $|\kappa(\mathbf{p})| <= \kappa_{max}$ **then**
      return $\{\mathbf{p}_{m.l}, \mathbf{p}_{m.r}\}$
  $l \leftarrow \frac{1}{2} \cdot l$
  $\mathbf{p}_{m.l} \leftarrow \mathbf{p}_{m.l} + sign(d_{th} - d) \cdot l \cdot \mathbf{u}_{prev}$
  $\mathbf{p}_{m.r} \leftarrow \mathbf{p}_{m.r} + sign(d_{th} - d) \cdot l \cdot \mathbf{u}_{next}$
return $\{\mathbf{p}_{m.l}, \mathbf{p}_{m.r}\}$

The proposed midpoint insertion method is outlined in Algorithm 1. A B-spline path is initially generated from a waypoint sequence as the initial control point sequence $\mathcal{C}$. The algorithm sequentially checks if the path deviates from each waypoint beyond the passing tolerance. If a waypoint is found where the path violates this condition, the algorithm inserts a midpoint pair and adjusts positions iteratively. If a path satisfying the passing tolerance condition at a specific waypoint exceeds the maximum curvature condition, the algorithm generates a path that makes the robot stopover at that waypoint.

The results of the Algorithm 1 is illustrated in Fig. . From the inserted midpoints, we can adjust the distance between each waypoint and the desired path without compromising the user's intended path.

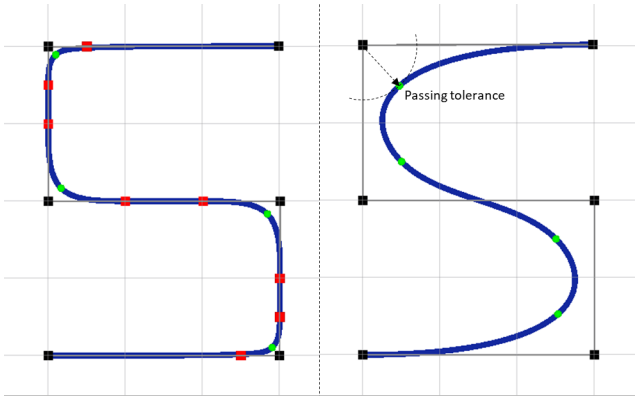

Fig. 2. Path refinement by insertion of midpoint pairs. The right figure depicts the initially generated path without constraints. The refined path is presented in the left figure. Black squares represent the user-defined waypoint sequence. Green dots are the projected points corresponding to each waypoint. Red squares are the inserted midpoint pairs.

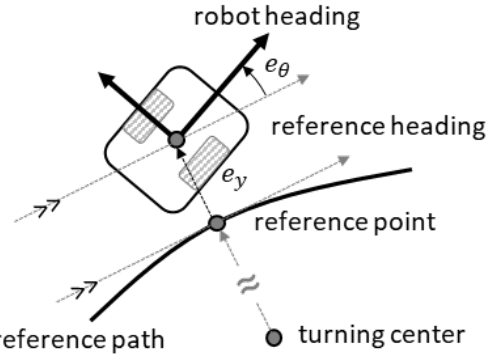

Fig. 3. Definition of path deviation

III. Path Following Control

We have designed a loosely coupled architecture for the path following control, separating the computation of command linear velocity and command angular velocity. We can determine the reference point, $\mathbf{p}_b(u_c)$ on the desired path corresponding to the current position of the robot as follows:

$$u_c = \underset{u \in [0,1]}{\operatorname{argmin}} \|\mathbf{p}_b(u) - \mathbf{p}_{robot}\|^2 \tag{13}$$

where, $\mathbf{p}_{robot}$ is the current position of the robot.

Subsequently, the path tangent at the reference point can serve as the reference heading. Path deviation can then be defined as the difference between the robot's pose and the reference pose. This relationship is illustrated in Fig. 3.

A. Deviation compensator

The proposed framework is designed for industrial robots to adhere to designated work paths. To prevent accidents and mitigate risks, the robots are intended to stop and wait when encountering obstacles on the path instead of attempting to avoid them.

The angular velocity serves as a motion component that directly compensates for path deviation. The desired angular velocity can be computed as follows:

$$\omega_d = \kappa_r \cdot v \cdot cos(e_\theta) - 2k_e \cdot tan(e_\theta) - \frac{k_e^2}{v \cdot cos(e_\theta)} \cdot e_y \tag{14}$$

where, κ_r is a curvature at the reference point. v is a longitudinal velocity. e_y is a lateral deviation. e_θ is a heading deviation from a reference heading, θ_r. k_e is a feedback gain.

Compared to [8], small angle assumption is not applied and single control gain is used.

Assuming no side slip condition and constant linear velocity, the time rate of change of lateral deviation and its derivative can be defined as follows:

$$\dot{e}_y = v \cdot \sin(e_\theta) \tag{15}$$

$$\ddot{e}_y = v \cdot \cos(e_\theta) \cdot \dot{e}_\theta \tag{16}$$

The time derivative of the heading deviation can be defined as follows:

$$\dot{e}_\theta = \omega - \kappa_r \cdot v \cdot cos(e_\theta) \tag{17}$$

where, ω is a angular velocity.

Substituting the desired angular velocity from (14) into (17) and simplifying (16) with (15) yields the following:

$$\ddot{e}_y = -2 \cdot k_e \cdot \dot{e}_y - k_e^2 \cdot e_y \tag{18}$$

Equation (18) represents the second-order error dynamics of lateral deviation, indicating a critically damped system (i.e., a damping ratio of 1.0). The feedback gain, k_e acts as the natural frequency of the second-order system. Accordingly, the transient response of the error dynamics can be designed directly by the feedback gain.

If the robot's angular velocity converges to the desired velocity given by (14) within a finite time, we can infer that the lateral deviation will also converge to zero.

From the second and third terms of the desired angular velocity, we can restrict the allowable range of heading deviation to be between -90 degrees and +90 degrees. However, exceeding a magnitude of 90 degrees during typical following motion implies control failure, indicating that the robot is moving in the opposite direction of its intended path. Thus, this range can be considered reasonable.

However, in practical operation, the robot can start its following motion by facing orthogonally towards the path for faster convergence. As the heading deviation approaches 90 degrees, compensation for lateral deviation may lead to unintended consequences. To address this, we set the feedback gain with an adaptive policy as follows:

$$k_e = k_e^* \cdot cos(e_\theta) \tag{19}$$

where, k_e^* is a tuning parameter as a base gain.

Substituting equation (19) into (14), the desired angular velocity can be rewritten as follows.

$$\omega_d^m = \kappa_r \cdot v \cdot cos(e_\theta) - 2k_e^* \cdot sin(e_\theta) - \frac{{k_e^*}^2 \cdot cos(e_\theta)}{v} \cdot e_y \tag{20}$$

From the adaptive gain in (19), it is intended to focus more on orientation alignment when the heading deviation is very large, and to prioritize compensation for lateral deviation when the heading deviation is relatively small. The transition is smooth and continuous.

It should be noted that the critical damped response of lateral deviation, as described in (18), can be achieved when the desired angular velocity matches the current angular velocity. Therefore, it is important that the angular velocity closely tracks the desired angular velocity without significant error or delay.

When the robot is following a path with severe curvature, even minor actuation delays can impact following performance significantly. Thus, we employ a preview curvature at a path point a certain distance ahead in (20) to compensate for actuation delay as follows:

$$\kappa_p = \kappa(u_c + \Delta u_p) \tag{21}$$

where, $\kappa_r = \kappa(u_c)$ and Δu_p is a path parameter difference corresponding to $t_{lag} \cdot v$. t_{lag} is an averaged time lag representing actuation delay.

It should be noted that the desired angular velocity, as derived from (20) and (21) is based on the current linear velocity, v. As a result, the intended motion trajectory is directly determined by both the desired angular velocity and the current linear velocity. Therefore, the final desired angular velocity should be adjusted from the desired linear velocity, determined in the next subsection, to maintain the intended motion trajectory. The final desired angular velocity can be computed as follows:

$$\omega_d^* = \left(\frac{\omega_d^m}{v} \right) \cdot v_d^* \tag{22}$$

where, v_d^* is the final desired linear velocity.

B. Speed planner

A desired linear velocity at the current time has to be determined by considering various independent factors such as the clearance to obstacles, the remaining distance, and the curvature of the desired path in the preview horizon. The desired linear velocity is determined as the minimum value among velocities considering each factor.

The linear velocity to prevent collisions with obstacles is computed based on a user-defined deceleration parameter.

$$v_c = \sqrt{2 \cdot a_{x.d} \cdot \max\{c - d_{safe}, 0\}} \tag{23}$$

where, $a_{x.d}$ is the user-defined deceleration parameter, c is the clearance and d_{safe} is a safe stopping distance.

For non-holonomic robots, the future trajectory can be approximated as an arc determined by the linear and angular velocities. Consequently, the clearance to obstacles can be defined as illustrated in Fig. 4

Additionally, we employ a different obstacle reflection strategy for path following compared to general navigation. As illustrated in Fig. 4, the robot determines the instantaneous turning radius based on the desired velocity to compute clearance against obstacles within the swept arc. In scenarios like wall-side docking, unnecessary deceleration may occur due to the swept arc generated by temporary desired velocity pointing towards the wall.

To mitigate this issue, when the robot adheres to the path, we establish a swept path envelope along the forward path and only reflect obstacles within it. If the robot violates the swept path envelope, all obstacles are immediately reflected to ensure safety. The swept path envelope is represented as a convex hull of vertices of footprints at each preview path point along the forward path as depicted in Fig. 5.

We sample the preview path points along the forward path iteratively. Initial sampled path points are extracted at equal intervals of the path parameter, u, until the cumulative sum of the lengths between successive points exceeds the

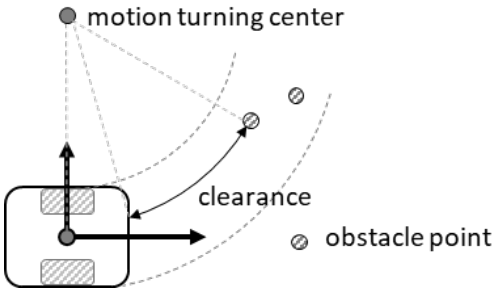

Fig. 4. Clearance to obstacles during turning motion for a non-holonomic mobile robot.

preview distance. The preview distance is defined as follow, considering the user-defined deceleration parameter.

$$s_p = \max\left\{ l_f + d_{safe}, \frac{v^2}{2 \cdot a_{x.d}} \right\} \tag{24}$$

where, v is a robot's linear velocity and l_f represents the distance from the center point of the robot to the front of footprint.

Since the analytic length of a path is always greater than the rectified length, the robot consistently obtains preview information over the preview distance. Until the difference of the rectified length becomes less than a threshold, path points are sampled between successive points and added to the initial sample set. This preview path point sampling method is outlined in Algorithm 2.

The linear velocity required to stop at the goal is determined as follows:

$$v_a = \sqrt{2 \cdot a_{x.d} \cdot d_{goal}} \tag{25}$$

where, d_{goal} represents the remaining distance which can be computed as the rectified length of the preview path point sequence obtained from Algorithm 2.

$$d_{goal} = \sum_{i=0}^{n-1} \|p_i - p_{i+1}\| \tag{26}$$

It should be noted that path points contain not only position information but also curvature information at their corresponding positions. Therefore, we can utilize the sampled point set to plan the required velocity, v_p, considering

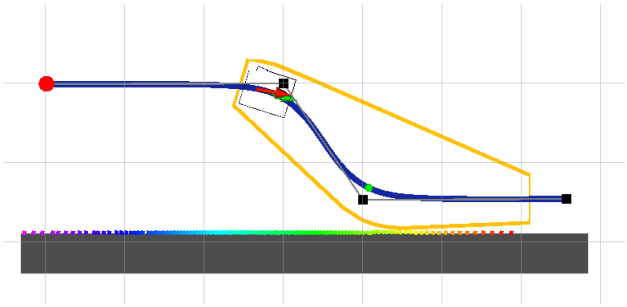

Fig. 5. Representation of swept path envelope to prevent unnecessary braking during wall-side docking scenario

Algorithm 2: Sampling of preview path points

Input: path, current path parameter (u_c), preview distance (s_p), path parameter interval (Δu), rectification tolerance (k_{tol})
Output: sampled preview point list ($\mathcal{P}_p$)

Procedure: Initial sampling
$u \leftarrow u_c$, $\mathbf{p} \leftarrow \mathbf{p}_b(u_c)$, $s \leftarrow 0$
$\mathcal{U}_p \leftarrow [u_c]$, $\mathcal{P}_p \leftarrow [\mathbf{p}]$
for $1 \le i \le$ *max_iter* **do**
  $u \leftarrow u + \Delta u$
  $s \leftarrow s + \|\mathbf{p} - \mathbf{p}_b(u)\|$
  $\mathbf{p} \leftarrow \mathbf{p}_b(u)$
  AppendList($\mathcal{U}_p$, u)
  AppendList($\mathcal{P}_p$, $\mathbf{p}$)
  if $s \ge s_p$ **then break**
end Procedure

Procedure: Populate point densely
for $1 < i \le \mathbf{card}(\mathcal{U}_p)$ **do**
  $u_m = (\mathcal{U}_p[i-1] + \mathcal{U}_p[i])/2$
  $\mathbf{p}_m = \mathbf{p}_b(u_m)$
  $l_{ref} = \|\overrightarrow{\mathcal{P}_p[i-1]\mathcal{P}_p[i]}\|$
  $l = \|\overrightarrow{\mathcal{P}_p[i-1]\mathbf{p}_m}\| + \|\overrightarrow{\mathbf{p}_m\mathcal{P}_p[i]}\|$
  AppendList($\mathcal{P}_p$, $\mathbf{p}_m$)
  if $(l - l_{ref} \ge k_{tol} \cdot l_{ref})$ **then**
    $\mathcal{U}_p \leftarrow$ Insert u_m between $\mathcal{U}_p[i-1]$ and $\mathcal{U}_p[i]$
    $\mathcal{P}_p \leftarrow$ Insert $\mathbf{p}_m$ between $\mathcal{P}_p[i-1]$ and $\mathcal{P}_p[i]$
end Procedure
return $\mathcal{P}_p$

the preview curvature information. The algorithm for calculating the required velocity considering preview curvature information is presented in Algorithm 3.

Numerous approaches employ the model predictive control to compute predictive velocity profile for incorporating preview curvature information. However, from our proposed method we can obtain the safe corner-entry velocity without relying on optimization techniques.

It should be noted that the planned angular and linear velocities are based on the ideal geometric model. However, achieving stable following performance for all parameter sets is not always possible due to real-world problems such as limitation in actuation performance. Therefore, we propose a braking intervention strategy to decelerate the robot when deviating from the path.

We defined the intervention-required condition as the product of lateral deviation and the rate of change of lateral deviation being greater than a certain positive value. The braking intervention strategy helps the robot regain controllability by reducing the rate of change of lateral deviation.

$$v_b = \begin{cases} v_{d.prev} - a_{x.d} \cdot \Delta t_s & \text{if } e_y \cdot \dot{e}_y \ge \epsilon_{th} \\ v_{set} & \text{else} \end{cases} \tag{27}$$

Algorithm 3: Linear velocity for preview curvature

Input: preview curvature list ($\mathcal{K}_p$) and curve length list ($\mathcal{S}_p$) corresponding to $\mathcal{P}_p$
Output: predictive velocity (v_p)

$i^* \leftarrow \underset{i}{\text{argmax}}|\mathcal{K}_p[i]|$, $i \leftarrow i^*$
$s_m \leftarrow \mathcal{S}_p[i^*]$, $v_m \leftarrow \min\{v_{max}, \omega_{max}/\mathcal{K}_p[i^*]\}$
while $i \neq 0$ **do**
  $v = \min\{v_{max}, \omega_{max}/\mathcal{K}_p[i]\}$
  $v_p \leftarrow \sqrt{v_m^2 + 2 \cdot a_{x.d} \cdot (s_m - \mathcal{S}_p[i])}$
  if $v_p \geq v$ **then**
    $s_m \leftarrow \mathcal{S}_p[i]$
    $v_m \leftarrow \min\{v_{max}, \omega_{max}/\mathcal{K}_p[i]\}$
  $i \leftarrow i - 1$
return v_p

where, $v_{d.prev}$ is the desired linear velocity at previous time step, Δt_s is the computation period, v_{set} is a user-defined reference velocity.

One of the primary causes of unstable following performance in real robots is sudden acceleration. When the robot exits a corner, abrupt acceleration can occur if the user-defined reference acceleration is relatively high. To mitigate this issue, we established an upper limit velocity with the low-pass filtered acceleration to mitigate this issue as follows:

$$v_u = v_{d.prev} + a^*_{x.a} \cdot \Delta t_s \tag{28}$$

where, $a^*_{x.a}$ stands for the low-pass filtered acceleration as follows:

$$a^*_{x.a} = \begin{cases} \rho \cdot a^*_{x.a.prev} + (1-\rho) \cdot a_{x.a} & \text{if } \widetilde{v}_d \geq v_{d.prev} \\ 0 & \text{else} \end{cases} \tag{29}$$

where, $\rho \in [0, 1]$ represents the delay intensity. $a^*_{x.a.prev}$ is the filtered acceleration at previous time step. $a_{x.a}$ stands for the user-defined reference acceleration, and $\widetilde{v}_d = \min\{v_{set}, v_c, v_a, v_p, v_b\}$.

Finally, the desired linear velocity is determined the minimum value of velocities derived from above.

$$v^*_d = \min\{v_{set}, v_c, v_a, v_p, v_b, v_u\} \tag{30}$$

IV. VALIDATION

The proposed path following framework has been validated through robot tests, demonstrating accurate real-time following capabilities with fast computation times. Tests were conducted using S-shaped and R-shaped paths. The desired path was generated from the given waypoint sequence, which included "stopover" waypoints, with a passing tolerance of 0.25m.

The generated path and robot trajectory represented in Fig. 6(a). The black squares represent the given waypoint sequence, and the red squares are "stopover" waypoints. The robot trajectory is depicted as a black solid line, and

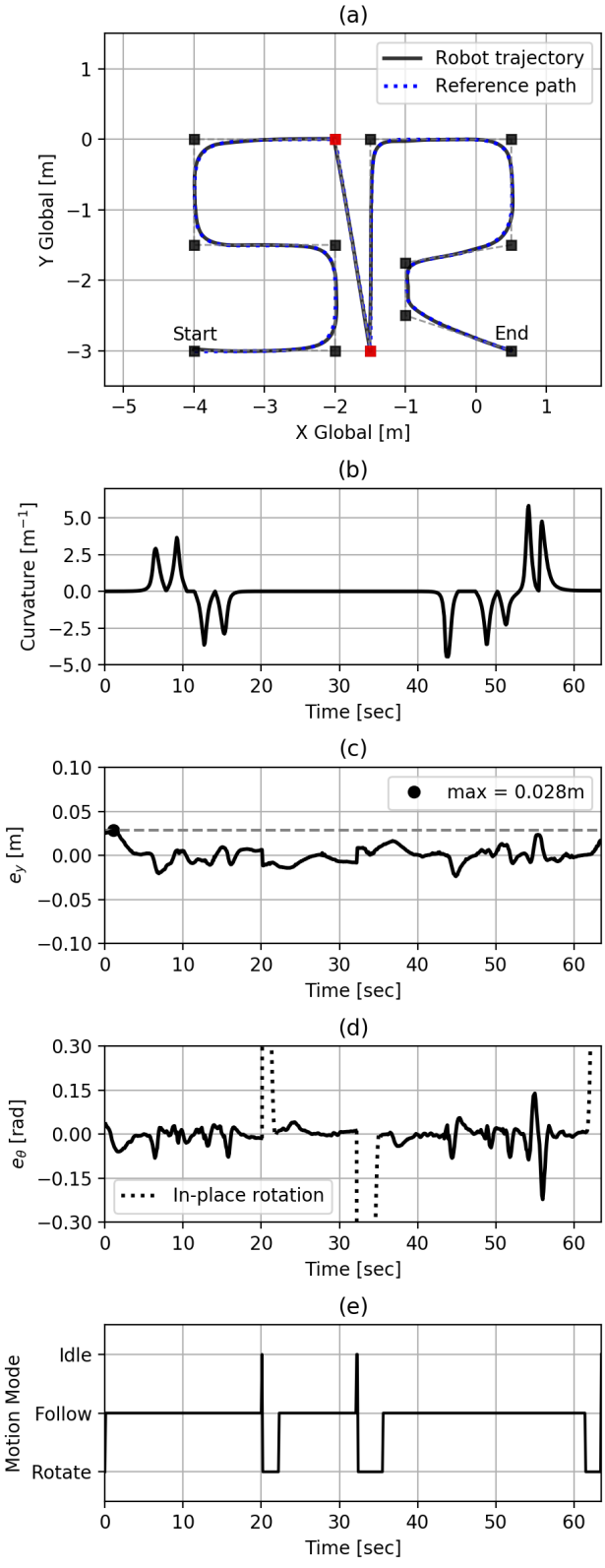

Fig. 6. (a) represents desired path and robot trajectory. (b) represents reference curvature profile. (c) and (d) depict lateral and heading deviation, respectively. (e) illustrate motion mode transition history. **The motion mode transition occurs at a stopover waypoint. The rotating mode stands for in-place rotation toward the next waypoint.**

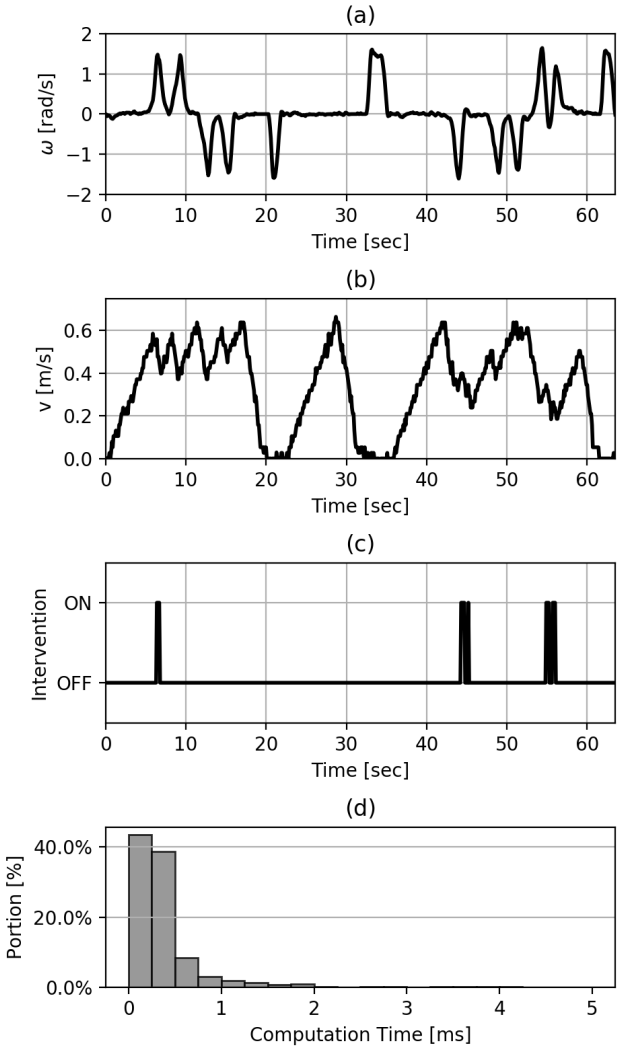

Fig. 7. (a) and (b) represent angular and linear velocities, respectively. (c) illustrates braking intervention history, and (d) depicts a histogram of computation time

the generated path is illustrated as a blue dotted line. The curvature profile is presented in Fig. 6(b), demonstrating the continuous curvature of the generated path. Additionally, it is noted that the maximum curvature constraint, set at $7.5m^{-1}$, was satisfied.

Lateral deviation is presented in Fig. 6(c), with a maximum lateral deviation of 0.028m. Considering the localization error, the proposed following control system demonstrated accurate following performance on the robot.

It should be noted that the sudden changes in heading deviation around 20 and 35 seconds result from the reference heading being adjusted for in-place rotation towards the next waypoint at stopover waypoints.

In Fig. 7(b), a predictive deceleration before entering corners is observed, and the braking intervention is activated when path deviation increases as in Fig. 7(c). Fig. 7(d) presents the real-time capability of the proposed framework with a Qualcomm QRB5165 processor. The average computation time was 0.43ms and roughly 95% of the total measurement recorded 1ms or less.

V. CONCLUSIONS

We proposed a practical framework for path representation and following control in mobile industrial robots. For industrial robots that require specific work paths, we suggest a path representation method that generates drivable paths from the given waypoint sequence provided by the user. The generated paths are bounded within the convex hull formed by the specified waypoint sequence and support the intuitive specification of work paths. The paths ensure that the deviation from each waypoint and the curvature do not exceed the allowable range. All constraints on the path are satisfied by iteratively adjusting the positions of midpoint pairs between waypoints without using nonlinear optimization techniques. Furthermore, the introduction of "stopover" waypoints enhances practicality in narrow workspaces. We proposed a simple and practical control system for following the generated paths. A stable compensator that converges lateral deviation to a critical damped response and a speed planner supporting reliable following performance have been developed. Notably, the proposed speed planner plans desired speed without using optimization techniques by incorporating predictive information such as forward path curvature. Moreover, to prevent unnecessary deceleration caused by irrelevant obstacles along the path, we utilize an obstacle reflection method based on the swept path envelope. The proposed speed planner also employs a braking intervention strategy if the robot deviates from the path due to actuation delay. The proposed framework has been validated through robot tests, demonstrating accurate following performance. The real-time performance of the proposed framework was achieved with an average computation time of 0.43ms.

REFERENCES

[1] Elbanhawi, M., Simic, M., & Jazar, R. N. (2015). Continuous path smoothing for car-like robots using B-spline curves. Journal of Intelligent & Robotic Systems, 80, 23-56.

[2] Amer, N. H., Zamzuri, H., Hudha, K., & Kadir, Z. A. (2017). Modelling and control strategies in path tracking control for autonomous ground vehicles: a review of state of the art and challenges. Journal of intelligent & robotic systems, 86, 225-254.

[3] Xu, S., & Peng, H. (2019). Design, analysis, and experiments of preview path tracking control for autonomous vehicles. IEEE Transactions on Intelligent Transportation Systems, 21(1), 48-58.

[4] Wang, H., Liu, B., Ping, X., & An, Q. (2019). Path tracking control for autonomous vehicles based on an improved MPC. IEEE access, 7, 161064-161073.

[5] Künhe, F., Gomes, J., & Fetter, W. (2005, September). Mobile robot trajectory tracking using model predictive control. In II IEEE latin-american robotics symposium (Vol. 51, p. 5).

[6] Cheng, S., Li, L., Chen, X., & Wu, J. (2020). Model-predictive-control-based path tracking controller of autonomous vehicle considering parametric uncertainties and velocity-varying. IEEE Transactions on Industrial Electronics, 68(9), 8698-8707.

[7] Li, X., Zhu, S., Aksun-Guvenc, B., & Guvenc, L. (2021). Development and evaluation of path and speed profile planning and tracking control for an autonomous shuttle using a realistic, virtual simulation environment. Journal of Intelligent & Robotic Systems, 101, 1-23.

[8] Goh, J. Y., Goel, T., & Christian Gerdes, J. (2020). Toward automated vehicle control beyond the stability limits: drifting along a general path. Journal of Dynamic Systems, Measurement, and Control, 142(2), 021004.