2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Kinodynamic Modular Approach Local Trajectory Planner for Straightforward Motions of Differential Wheeled Mobile Robots

Yong Jin Byeon[1*], Mingyung Jang[1*] and Yunjeong Kim[2*]

Abstract— **This paper introduces a Kinodynamic Modular Approach(KMA) local planning algorithm designed to generate an obstacle avoidance trajectory that has both feasible and straightforward motions for differential wheeled mobile robots(DWMRs). To enhance the smoothness and feasibility of the trajectory, the proposed planner incorporates a kinodynamic system that considers non-holonomic constraints with velocity and acceleration limits. Not only consistent but also straightforward motion is obtained by defining basic motions and selecting the appropriate one according to the given situation. The simulation and real-world experiment results demonstrate the intuitiveness and applicability of the proposed algorithm compared to the time elastic band(TEB) algorithm.**

I. INTRODUCTION

Over the past decade, the robotics industry market has been rapidly expanding. This trend is noticeable not only in traditional manufacturing robots but also in mobile robots designed for logistics, such as Amazon's Kiva. In mobile robots, there are various valuable assets to consider including motion planning, motion tracking, localization, traffic control, and more. Motion planning is a crucial and essential element to generate safe and efficient motions, especially in complex environments like logistics.

In motion planning, various research is conducted on path planning. Path planning aims to obtain the path that guides the robot from the initial pose to the desired pose. Graph search algorithms, including Dijkstra [1], A* [2] and D* [3], have been proposed for path planning. Probability-based approaches have also been proposed such as the probabilistic road map method (PRM) [4], the rapidly exploring random tree (RRT) [5], and genetic algorithm [6].

However, the paths created using the above-mentioned method have been problematic due to their jerky and erratic nature, leading to significant discrepancies with the actual movement of the robot. In confined spaces like warehouses, feasibility is crucial because a lack of it can result in colliding with obstacles or getting stuck. To solve this issue, research has been conducted on path smoothing to make it easier for the robot to follow the planned path. In [7] and [8], a path smoothing method for car-like mobile robots was proposed because smoothness is crucial especially in car-like mobile robots, due to the characteristic of being unable to perform in-place rotation. Additionally, RRT-based approaches that utilize splines has been explored in [9].

Furthermore, there are some studies that consider kinodynamics (kinematics and dynamics constraints) to generate a more feasible trajectory compared to path smoothing method. Kinodynamic RRT* algorithm [10] based on the RRT algorithm was proposed to generate a feasible trajectory by considering linear actuator dynamics. For differential wheeled mobile robots (DWMRs), studies [11], [12] generate the time-optimal feasible trajectory by considering the dynamics of the actuator.

In addition, a local path planner is necessary to avoid obstacles when navigating unknown environments. To react immediately to a sudden appearance of obstacles, it is important to generate path with low computation time. In [13] and [14], they utilize the concept of potential fields to ensure rapid reactivity. A local path planning algorithm improving the existing RRT to reduce computation time was proposed in [15]. Reinforcement learning is utilized to generate a local path in [16].

There are studies that consider kinodynamics when planning a local trajectory. The timed elastic band (TEB) algorithm [17], [18] proposed a local planner derived from elastic band (EB) algorithm [19] with consideration of time-related information such as velocity and acceleration. Additionally, other research [20], [21] utilizes the dynamic window approach (DWA) to generate a local trajectory, which identifies the available velocity space excluding regions where collisions occur and choose the best one. Adding time information, TEB and DWA achieved fast reactivity and smoothness for the sudden appearance of obstacles.

In many types of research, there is a tendency to prioritize mathematical completeness without consideration of users' experience [22]. In a shared space with both robots and humans, it is crucial for a robot to exhibit consistent movements in a similar environment, as it helps humans predict the robot's actions. If humans can recognize the behavior patterns of robots, it becomes easier for them to avoid collisions when they encounter robots. However, the previous work about planning a local trajectory considering kinodynamics only pursues the avoidance of obstacles and smoothing of planned path. Therefore, there is a problem that local detour is easy to occur due to the excessive steering when turning on a curve. In addition, they tends not to generate a similar movements even with slight environment changes especially in TEB.

*This work is supported by the Korea Agency for Infrastructure Technology Advancement (KAIA) grant funded by the Ministry of Land, Infrastructure and Transport (Grant RS-2022-00143717).

[1]YJ. Byeon and M. Jang are with TWINNY Co.Ltd., Deajeon, Republic Korea. yjbyeon@twinny.co.kr; jmg941120@twinny.co.kr

[2]Y. Kim (the corresponding author) is with TWINNY Co.Ltd., Deajeon, Republic Korea. yjkim@twinny.co.kr

*YJ. Byeon, M. Jang, and Y. Kim contributed equally to this work.

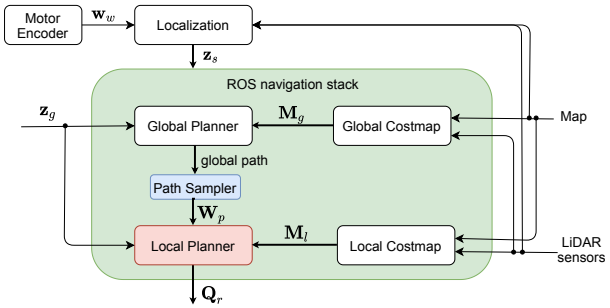


Fig. 1: Navigation system architecture: the blue box represents the difference from the conventional ROS navigation stack and the red box denotes the local planner we propose.

In this paper, we propose a trajectory planning algorithm based on kinodynamic modular approach (KMA) for DWMRs to obtain a feasible and straightforward local trajectory which makes it easy for humans to predict the robot's motion. By considering kinodynamic with acceleration and velocity constraints, the proposed algorithm generate a accurate and smooth trajectory. Furthermore, the resulting motion exhibits simple and consistent movements without inefficient or oscillated movements by defining the movements of the robot with basic motions and avoiding the use of randomness. To verify the proposed algorithm, simulations are conducted in Gazebo [23] environments, and experiments in the real-world are also performed using the NarGo robot from TWINNY Co.Ltd., comparing the results with the TEB algorithm.

II. Navigation System

In this section, we introduce the navigation system and its sub-modules, excluding the *Local Planner* module, which are designed for our proposed local planner. Our system is built upon the ROS navigation stack [24] with some modifications as depicted in Fig. 1.

A. Localization

The localization module provides the current robot state $\mathbf{z}_s$ to the modules in the ROS navigation stack. While various methods can be employed for the localization module, we have chosen to utilize the AMCL package [25] provided by ROS, configured to align with DWMRs. Unlike the original AMCL, we incorporate the robot's current pose provided by AMCL with wheel odometry $\mathbf{w}_w$ obtained from motor encoders. This integration ensures that the system is equipped with the most up-to-date information regarding the robot's current pose and velocity.

B. Costmap

To check for collisions and calculate the distance to obstacles, our system employs two types of costmaps: a global costmap $\mathbf{M}_g$ and a local costmap $\mathbf{M}_l$. The global costmap represents the obstacle cost of the entire map and is used in the *Global Planner*. On the other hand, the local costmap only reflects obstacles around the robot obtained by map and LiDAR sensors and is used in the *Local Planner*. By utilizing these costmaps and representing the robot as a polygon, these modules provide information about the collision occurrence and the distance to the nearest obstacle at a specific robot pose.

C. Global Planner

The *Global Planner* generates a global path from the starting pose to the destination pose on the global costmap $\mathbf{M_g}$. We employed the Dijkstra planner [1] as our *Global Planner*. The output from this planner, representing the shortest path from a global perspective, serves as a reference for the *Local Planner*. However, the global path cannot be directly used because it lacks smoothness and is planned with the assumption that the robot has a circular shape. Therefore, we sampled specific points along the global path by the *Path Sampler* in Fig.1 where a significant deviation in distance or angle occurs—just enough to indicate the overall direction of the path. These strategically selected points are presented as waypoints $\mathbf{W}_p$ for the *Local Planner*, reducing dependency on the global path and enhancing navigation efficiency.

III. Problem Statement

A. Differential Wheeled Mobile Robot's Kinodynamic

Under the pure rolling assumption, DWMRs have the non-holonomic constraint and its kinematics is defined as

$$\dot{\mathbf{P}} = \begin{bmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v} \tag{1}$$

where $\mathbf{P} = [\mathbf{p}^T \ \theta]^T = [x \ y \ \theta]^T$ is the robot pose, $\mathbf{v} = [v \ w]^T$ is the velocity vector, $\mathbf{p}$ is the robot's position, x, y, and θ represent x-coordinate, y-coordinate, and heading angle and v and w are the linear and angular velocity, respectively.

In this paper, we assume that the acceleration, $\mathbf{a} = [a_v \ a_w]^T$, is constant such as

$$\dot{\mathbf{v}} = \mathbf{a} \tag{2}$$

where a_v is the linear acceleration and a_w is the angular acceleration. The velocities and accelerations are constrained by the robot's hardware limits, such as

$$\begin{aligned} |\mathbf{v}| &\leq \mathbf{v}^{max} \\ |\mathbf{a}| &\leq \mathbf{a}^{max} \end{aligned} \tag{3}$$

where $\mathbf{v}^{max} = [v^{max} \ w^{max}]^T$ is the velocity limit and $\mathbf{a}^{max} = [a_v^{max} \ a_w^{max}]^T$ is the acceleration limit.

B. Problem Formulation

The objective of the proposed planner is to generate a trajectory $\mathbf{Q}_r$ that satisfies (1)-(3) at every planning period ΔT_p. This trajectory aims to guide the robot from its initial state, $\mathbf{z}_s = [x_s, y_s, \theta_s, v_s, w_s]^T$, to the destination state, $\mathbf{z}_g = [x_g, y_g, \theta_g, 0, 0]^T$, while avoiding obstacles.

The robot's current state $\mathbf{z}_s$ on the map is obtained through the localization module described in Section II-A. To prevent local minimums, the local trajectory planner collaborates

with the global planner by using the sampled points $\mathbf{W}_p$ as waypoints, which are periodically updated. This implies that our local planner plans a trajectory passing through $\mathbf{W}_p$ and reaching $\mathbf{z}_g$, satisfying a final angle θ_f.

The trajectory comprises the robot's states at discrete timestamps ΔT_c, equivalent to the control time period, represented as $\mathbf{Q} = \{\mathbf{z}_i | i = 0, \ldots, L_{max} - 1\}$ where $\mathbf{z}_i = [x_i, y_i, \theta_i, v_i, w_i]^T$ denotes the state at time $i \cdot \Delta T_c$, and $L_{max} \cdot \Delta T_c$ is the maximum planning time.

IV. Local Planner

In this section, we explain our proposed KMA local planner. To generate a feasible and straightforward trajectory, we define three types of basic trajectories outlined in Section IV-A that satisfy (1)-(3). In Section IV-B, the overall algorithm is explained how to select a suitable trajectory among the basic trajectories. Especially when the trajectory towards the current target point (waypoints or destination) is expected to collide with obstacles, a finite number of local waypoints are generated. Among these waypoints, the local waypoint that minimizes some costs is selected to avoid obstacles through Section IV-C.

The local planner is executed at every planning period ΔT_p, and it outputs the velocity associated with these trajectories as control inputs at each control time ΔT_c.

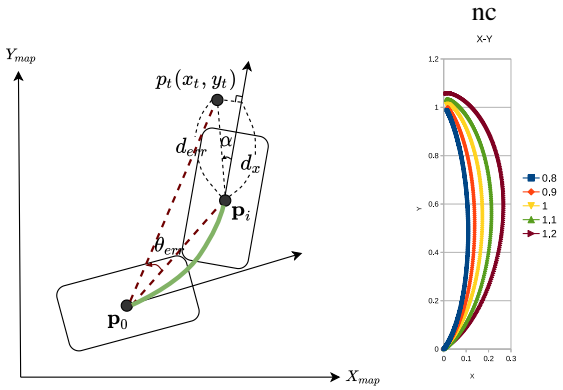
A. Basic trajectory

Three types of basic trajectories are defined as follow: *Driving Trajectory, Focusing Trajectory, and Stopping Trajectory*. The driving trajectory refers to a trajectory connecting the current state of the robot to the target point. The focusing trajectory represents a trajectory which makes the robot rotate to reach a desired angle while decreasing the linear velocity. The stopping trajectory indicates a trajectory with a consistent deceleration to come to a complete stop or a trajectory decelerating to reach a specific position.

These basic trajectories are planned in the same way. First, the next velocity, $\mathbf{v}_{i+1}$, is determined to have a trapezoidal velocity profile within (3). Subsequently, the next robot pose, $\mathbf{P}_{i+1}$, is calculated through (1) using these velocities at every control time step.

1) Driving trajectory: *drivingTrajectory*$(\mathbf{z}_s, \mathbf{p}_t)$ generates the driving trajectory $\mathbf{Q}_{dr}$ with the given the current robot state $\mathbf{z}_s$ and the current target position $\mathbf{p}_t = (x_t, y_t)$ which connects $\mathbf{z}_s$ to $\mathbf{p}_t$ satisfying (1)-(3) with the driving acceleration $\mathbf{a}_{dr} = [a^v_{dr}\ a^w_{dr}]^T$. The Algorithm 1 represents how the driving trajectory is generated.

At each control time step ΔT_c, the next linear velocity v_{i+1} and the angular velocity w_{i+1} are determined by increasing or decreasing the current linear v_i or angular velocity w_i, respectively. The velocity, $\mathbf{v} = [v,\ w]^T$, is decreased by its acceleration $\mathbf{a}_{dr}$ described as *decrease*$(\mathbf{v}, \mathbf{a}_{dr})$ in Algorithm 1 if the remaining x distance of the robot to $\mathbf{p}_t$ referred as d_x, is less than or equal to d_{stop} which is the required distance for the robot to stop as shown in Fig. 2a. In case of $d_x > d_{stop}$, it determines whether to increase or decrease v_i and w_i depending on the rotational angle α needed for the robot to face $\mathbf{p}_t$. v_i is decreased if α is bigger than the θ_{dr} which is an adjustable parameter; otherwise it is increased by a^v_{dr} or saturated by v^{max}_{dr} described as *increase*(v_i, a^v_{dr}, v^{max}). w_i is increased with $sign(\alpha) \cdot a^w_{dr}$ or saturated by $sign(\alpha) \cdot w^{max}_{dr}$ until α becomes less than or equal to the stopping angle θ_{stop} which is the required rotating angle for the robot to stop with the current angular velocity w_i.



(a) (b)

Fig. 2: (a) The path with green is a part of the driving trajectory which is planned up to i-th step with the target point $\mathbf{p}_t$. (b) The example of the driving trajectories obtained by incrementally increasing θ_{dr} from 0.8 to 1.2 when $\mathbf{z}_s = [0,0,0,0,0]^T$ and $\mathbf{p}_t = [0,1]^T$.

The robot's pose at the next time step $\mathbf{P}_{i+1}$ is then calculated using the obtained velocities by (1). This procedure is repeated until the distance error d_{err} is within the thresholds of δ_g.

θ_{dr} is an adjustable parameter which determines the increase or decrease of linear velocity. The driving trajectory algorithm tends to generate more detour trajectories as θ_{dr} increases, as the robot rotates at a higher linear velocity. Fig.2b shows an example of the driving trajectories obtained by incrementally increasing θ_{dr} from 0.8 to 1.2 when $\mathbf{z}_s = [0,0,0,0,0]^T$ and $\mathbf{p}_t = [0,1]^T$. Typically, it is not guaranteed that, given other constraints such as the acceleration limits, maximum velocity, and θ_{dr}, we can reach the goal precisely. But it is acceptable because the driving trajectory will guide the robot towards the goal, making its best efforts at each planning iteration within the constraints.

The driving trajectory is employed to generate a trajectory from the current state to the next waypoint in Algorithm 2 or to generate obstacle avoidance trajectories in Section IV-C.

2) Focusing trajectory: We define the focusing trajectory $\mathbf{Q}_{fc}$ as a trajectory ensuring that the last angle of the trajectory becomes the target angle θ_t, which is described as *focusingTrajectory*$(\mathbf{z}_s, \theta_t)$, by only adjusting the angular velocity with an acceleration of a^w_{fc} and a angular velocity limit of w^{max}_{fc} while reducing the linear velocity with a deceleration of a^v_{fc}. The angular velocity is decreased if $\theta_{err} \leq \theta_{stop}$, where $\theta_{err} = \theta_t - \theta_i$ and θ_{stop} represents the required angle to be stopped at the current angular velocity;otherwise, it is increased.

Algorithm 1: Driving Trajectory

Input: $\mathbf{z}_s, \mathbf{p}_t, \delta_g, \theta_{dr}, \mathbf{v}_{dr}^{max}, \mathbf{a}_{dr}$
Output: $\mathbf{Q}_{dr}$
Initialization: $i \leftarrow 0$

```
Calculate d_err, θ_err, α, d_x, d_stop
while d_err > δ_g do
    if d_x ≤ d_stop then
        v_{i+1} ← decrease(v_i, a^v_dr)
        w_{i+1} ← decrease(w_i, a^w_dr)
    else
        if α > θ_dr then
            v_{i+1} ← decrease(v_i, a^v_dr)
        else
            v_{i+1} ← increase(v_i, a^v_dr, v^max_dr)
        end if
        if α ≤ θ_stop then
            w_{i+1} ← decrease(w_i, a^w_dr)
        else
            w_{i+1} ← increase(w_i, sign(α)a^w_dr, sign(α)w^max_dr)
        end if
    end if
    P_{i+1} = P_i + f(θ_i) · v_{i+1} · ΔT_c
    i ← i + 1
    Update d_err, θ_err, α, d_x, d_stop
end while
```

Algorithm 2: KMA Overall Algorithm

Input: $\mathbf{z}_s, \mathbf{z}_g, \mathbf{M}_l, \mathbf{W}_p, \theta_{fc}, \delta_g, \delta_{wp}$
Output: $\mathbf{Q}_r$
Initialization: $k \leftarrow 0$

```
Calculate d_sg, d_st, θ_st, θ^st_err.
if d_sg > δ_g then
    if d_sg ≤ d_stop then
        Q_r = stoppingTrajectory(z_s, p_g) return
    else if d_st ≤ δ_wp then
        k ← k + 1
    end if
    p_t = w_p[k]
    if θ^st_err > θ_fc then
        Q_fc ← focusingTrajectory(z_s, θ_st)
        if no collision in Q_fc then
            Q_r = Q_fc return
        end if
    end if
    Q_dr ← drivingTrajectory(z_s, p_t)
    if no collision in Q_dr then
        Q_r = Q_dr return
    end if
    Q_lc ← obstacleAvoidanceTrajectory(z_s, p_t)
    if Q_lc exists then
        Q_r = Q_lc
    else
        Q_r = stoppingTrajectory(z_s)
    end if
else
    Q_fc = focusingTrajectory(z_s, θ_g)
    if no collision in Q_fc then
        Q_r = Q_fc
    else
        Q_r = stoppingTrajectory(z_s)
    end if
end if
```

The focusing trajectory is utilized in Algorithm 2 when the angle θ_{err}^{st} required for the robot to face the target point exceeds a certain threshold θ_{fc}, or when the robot needs to rotate at the destination to match the final angle.

3) Stopping trajectory: There are two types of the stopping trajectories. *stoppingTrajectory*$(\mathbf{z}_s)$ generates a trajectory that decelerates at a constant deceleration to come to a stop, and *stoppingTrajectory*$(\mathbf{z}_s, \mathbf{p}_t)$ generates a trajectory that adjusts deceleration to come to a stop at a specific point $\mathbf{p}_t$. When encountering obstacles, a stopping trajectory with a constant deceleration $\mathbf{a}_{st} = [a_{st}^v, a_{st}^w]^T$ generated by *stoppingTrajectory*$(\mathbf{z}_s)$ is employed in Algorithm 2. In order to reach the destination, a stopping trajectory is generated by *stoppingTrajectory*$(\mathbf{z}_s, \mathbf{p}_t)$ with deceleration $\mathbf{a}_{st} = [min\left(a_{st}^v, \frac{v_i^2}{2min(d_{err}, \delta_g)}\right) a_{st}^w]^T$ where d_{err} is the distance between the current robot position and the target position and δ_g is the arriving margin of the goal.

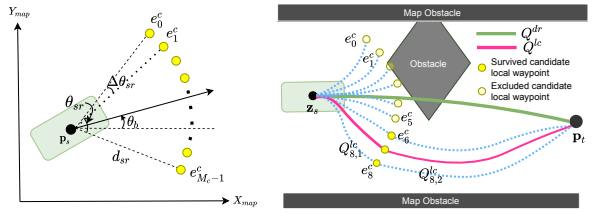
B. Overall algorithm

The overall algorithm is presented in Algorithm 2. The KMA local planner generates a local trajectory at each planning period ΔT_p that avoids obstacles reflected in $\mathbf{M}_l$ from $\mathbf{z}_s$ to $\mathbf{z}_g$ through $\mathbf{W}_p = \{w_p[k] | k = 0, ..., N_{wp} - 1\}$, where N_{wp} is the number of waypoints.

The first waypoint, $w_p[0]$, is selected as the current target point $\mathbf{p}_t$. Initially, the KMA planner checks whether a focusing trajectory is required or not by comparing the angle error, θ_{err}^{st}, needed to face $\mathbf{p}_t$ from the current angle, with a certain threshold θ_{fc}, determined by the user during the initial stages of the algorithm. If the focusing trajectory is necessary, $\mathbf{Q}_{fc}$ is generated by *focusingTrajectory*$(\mathbf{z}_s, \theta_{st})$ where θ_{st} is the angle formed by the line connecting the center of the robot to $\mathbf{p}_t$. If no collision occurs in $\mathbf{Q}_{fc}$, it becomes the resulting trajectory $\mathbf{Q}_r$; otherwise, the planner resorts to the driving trajectory $\mathbf{Q}_{dr}$ generated by *drivingTrajectory*$(\mathbf{z}_s, \mathbf{p}_t)$, provided it is collision-free.

If neither $\mathbf{Q}_{fc}$ nor $\mathbf{Q}_{dr}$ is collision-free, we utilize the trajectory passing through the local waypoint to reach $\mathbf{p}_t$ as the resulting trajectory. We refer to it as Q_{lc} or the obstacle avoidance trajectory, which can be generated through *obstacleAvoidanceTrajectory*$(\mathbf{z}_s, \mathbf{p}_t)$ to avoid obstacles, as described in the following Section IV-C.

Depending on the current state of the robot or the location of obstacles, there might be no collision-free trajectory passing through the generated local waypoints. In such cases, *stoppingTrajectory*$(\mathbf{z}_s)$ is generated and employed as the resulting trajectory.



(a) (b)

Fig. 3: (a) Generation of candidate local waypoints. (b) Selection of the obstacle avoidance trajectory.

This procedure continues until the robot reaches the destination $\mathbf{p}_g$ within the distance of δ_g, while updating the $\mathbf{p}_t$ as the next waypoint when the robot is close to $\mathbf{p}_t$ within δ_{wp}, where δ_g and δ_{wp} represent the arrival margin of the goal and a waypoint, respectively. Especially, *stoppingTrajectory*$(\mathbf{z}_s, \mathbf{p}_g)$ is employed to precisely reach the destination $\mathbf{p}_g$ after the distance to the destination d_{sg} becomes less than the stop distance d_{stop}.

After arriving at $\mathbf{p}_g$, the focusing trajectory is generated using *focusingTrajectory*$(\mathbf{z}_s, \theta_f)$ to match the final angle θ_f, or the stop trajectory is generated using *stoppingTrajectory*$(\mathbf{z}_s)$ in case a collision occurs in the focusing trajectory.

C. Generation of the obstacle avoidance trajectory

When a collision occurs in the driving trajectory in Algorithm 2, a trajectory for avoiding obstacles $\mathbf{Q}_{lc}$ must be created by *obstacleAvoidanceTrajectory*$(\mathbf{z}_s, \mathbf{p}_t)$. To generate this trajectory, candidate local waypoints are created around the robot, and candidate trajectories that connect the current robot state to the destination passing these waypoints are generated. The costs of the generated candidate trajectories are calculated, and the trajectory with the lowest cost is selected.

1) Generation of candidate local waypoints: Given a base angle θ_b, the candidate local waypoints, $E_c = \{e_m^c | m = 0, \ldots, M_c - 1\}$, are generated by sampling an arc of a circle with a radius d_{sr} centered at the robot position $\mathbf{p}_s$ with an angle step $\Delta\theta_{sr}$ where M_c is the total number of the candidate local waypoints. The central angle θ_{sr} is based on θ_b as illustrated in Fig. 3a. d_{sr} is determined by

$$d_{sr} = min(d_{sr}^{min} + r_v \cdot v_s,\ d_{sr}^{max}) \tag{4}$$

where d_{sr}^{min} and d_{sr}^{max} are the minimum distance and the maximum distance of d_{sr}, r_v is the ratio considering the robot's speed and v_s is the current robot's linear velocity. By increasing d_{sr} in accordance with v_s, it becomes possible to generate an avoidance trajectory that considers obstacles that are further away as the speed increases. θ_b is set as α which is the heading angle to the current target point.

2) Selection of the obstacle avoidance trajectory : Fig.3b shows an example of how the obstacle avoidance trajectory is selected using the candidate local waypoints e_m^c generated at Section IV-C. First, the trajectories from $\mathbf{z}_s$ to e_m^c denoted as $Q_{m,1}^{lc}$ are created using the driving trajectory. The candidate waypoints belonging to trajectories that collide with obstacles, are excluded. The remaining candidate waypoints are then connected to the target point using the driving trajectory, and we refer to these trajectories as $Q_{m,2}^{lc}$.
These survived trajectories are evaluated using a cost function that takes into account curvature and safety considerations. Low curvature results in reduced execution time and minimized redundant movement. The curvature cost C_m^{cv} of the trajectory passing through e_m^c is calculated as

$$C_m^{cv} = \frac{1}{N_m^{lc}} \sum_{i=0}^{N_m^{lc}-1} \frac{|w_i|}{max(|v_i|, v_{dz})} \tag{5}$$

where v_{dz} is a dead-zone velocity of the robot and N_m^{lc} is the number of states consisting of Q_m^{lc} with time stamp ΔT_c. The cost about safety is determined based on whether a collision has occurred and the distance values from obstacles. Since we have already excluded e_m^c where collisions occur, the collision cost C_m^{col} is only computed about $Q_{m,2}^{lc}$ such as

$$C_m^{col} = \begin{cases} 0 & \text{if obstacle-free in } Q_{m,2}^{lc} \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

The obstacle cost about $Q_{m,n}^{lc}$ where $n \in \{1, 2\}$, is calculated as the average cost of the robot's footprint on the trajectory such as

$$C_{m,n}^{obs} = \frac{1}{N_m^{lc}} \Sigma_{i=0}^{N_m^{lc}-1} e^{-\alpha_f \cdot d_i} \tag{7}$$

where α_f is the cost scaling factor and d_i is the minimum distance from i-th footprint to obstacles. To differentiate the weights of $Q_{m,1}^{lc}$ and $Q_{m,2}^{lc}$, the final obstacle cost C_m^{obs} is calculated as

$$C_m^{obs} = w_1^{obs} \cdot C_{m,1}^{obs} + w_2^{obs} \cdot C_{m,2}^{obs} \tag{8}$$

where w_1^{obs} and w_2^{obs} are the weighting coefficients of $C_{m,1}^{obs}$ and $C_{m,1}^{obs}$, respectively. As the costs units of C_m^{cv} and C_m^{obs} are different, we use the normalization technique to make them dimensionless using $\bar{C} = \frac{C - min(C)}{max(C) - min(C)}$. Finally, the trajectory is selected as the one minimizing the total cost C_m such as

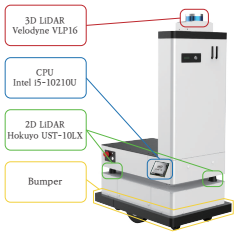
$$C_m = w^{cv} \cdot \bar{C}_m^{cv} + w^{col} \cdot C_m^{col} + w^{obs} \cdot \bar{C}_m^{obs} \tag{9}$$

where w^{cv}, w^{col} and w^{obs} are the weighting coefficients of curvature, collision and distance to obstacles, respectively.

V. Experiments

To evaluate our proposed KMA local planner, we conducted a comprehensive set of experiments, comparing its performance against that of the TEB local planner. To ensure fair comparisons, both local planners were tested within the identical navigation system described in Section II and used the same hardware limits, such as $\mathbf{v}^{max}$=[1.0 m/s, 1.0 rad/s]^T and $\mathbf{a}^{max}$=[0.5 m/s^2, 1.0 rad/s^2]^T. Specific details regarding the parameter settings of KMA are outlined in Table. I. Experiments in the Gazebo simulator and the real-world were conducted with the NarGo60[1] (refer to Fig. 4).

[1]https://twinny.ai/amr



TWINNY NarGo60

Size(LWH)	500 x 740 x 1240 mm
Max. Speed	1.2 m/s
Payload	60 kg
Operating Hours	Max. 8hrs

Fig. 4: The NarGo60 is a DWMR model developed by TWINNY Co. Ltd., specifically designed for autonomous logistics operations. It features obstacle detection capabilities with a 3D LiDAR and two 2D LiDARs.

Parameters	Values	Parameters	Values	Parameters	Values
$v_{dr}^{max}, v_{fc}^{max}(m/s)$	1.0	$\Delta T_c(s)$	0.01	α_f	1.0
$w_{dr}^{max}, w_{fc}^{max}(rad/s)$	1.0	$\Delta T_p(s)$	0.2	w_1^{obs}	0.97
$a_{dr}^{v}, a_{fc}^{v}, a_{st}^{v}(m/s^2)$	0.5	$\delta_g(m)$	0.1	w_2^{obs}	0.03
$a_{dr}^{w}, a_{fc}^{w}, a_{st}^{w}(rad/s^2)$	1.0	$\delta_{wp}(m)$	1.4	w^{cv}	0.1
$d_{sr}^{min}(m)$	0.6	$\theta_{dr}(rad)$	0.9	w^{obs}	1.0
$d_{sr}^{max}(m)$	3.0	$\theta_{fc}(rad)$	0.5π	w^{col}	100
r_v	1.0	$\theta_{sr}(rad)$	1.7		
$v_{dz}(m/s)$	0.05	$\Delta\theta_{sr}(rad)$	$\frac{\pi}{35}$		

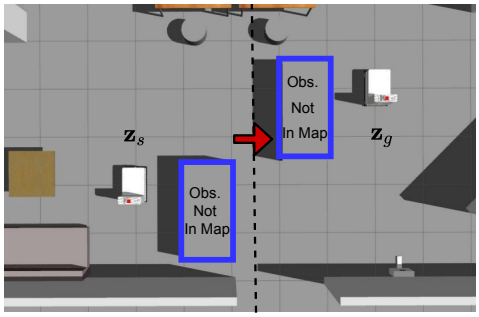
TABLE I: KMA parameters for the experiments.

Method	Travel Time (s)		Linear Dist. (m)		Angular Dist. (rad)	
	Mean	Std.	Mean	Std.	Mean	Std.
KMA	20.5	0.44	8.2	0.16	11.0	0.21
TEB	25.1	4.02	9.5	0.69	13.7	2.0

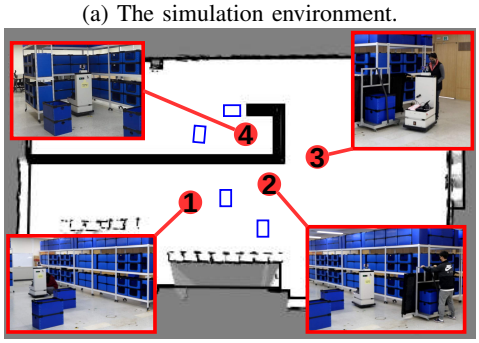
TABLE II: Comparison of the mean and standard deviation of the travel time, linear and angular moving distance results over 100 repetitions in simulations conducted by KMA and TEB.

A. Experiments in Gazebo simulator

The simulation was conducted 100 times repeatedly in an environment with two static obstacles not represented on the map, using the Gazebo simulator as shown in Fig. 5a with $\mathbf{z}_s = [-3, -2, -\frac{\pi}{2}, 0, 0]^T$ and $\mathbf{z}_g = [2, 0, -\frac{\pi}{2}, 0, 0]^T$. The Fig. 6 illustrates the results for one of the 100 repetitions, displaying RViz snapshots captured at particular moments that highlight features of the local planner, as well as the velocity and acceleration profiles of both KMA and TEB. The first and third snapshots in Fig. 6a and Fig. 6d show the different characteristics of KMA and TEB movements when a significant change of direction is required. In such cases, while KMA adjusts its angle through in-place rotation, TEB performs erratic movements, such as rotating while moving backward due to simultaneously aligning the angle, and reaching the destination. Moreover, in cases where the robot needs to avoid obstacles, KMA generates smooth and intuitive trajectories for obstacle avoidance using algorithms described in Section IV-C, while TEB hesitates to find solutions that satisfy both obstacle avoidance and



(a) The simulation environment.



(b) The real-world environment.

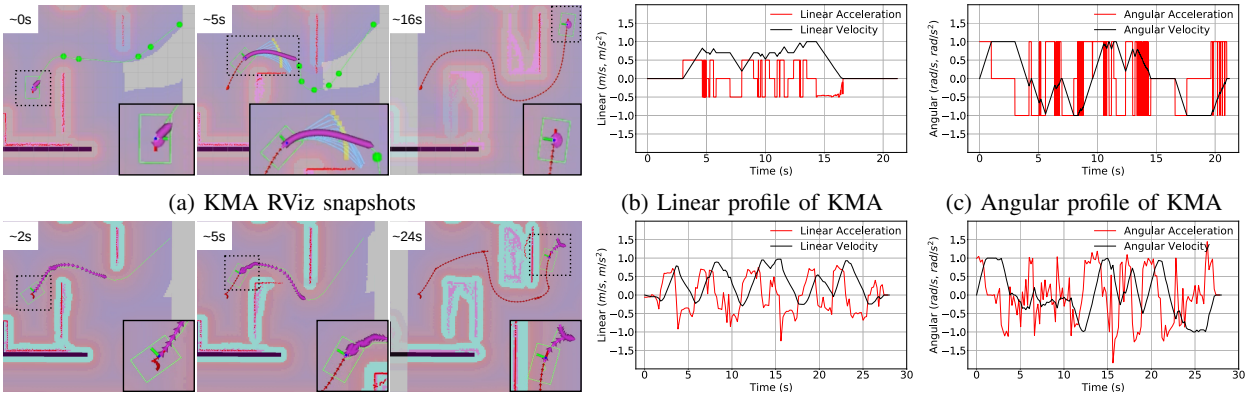
Fig. 5: Scenarios for the experiments.

following the global planner. Such features can also be observed in Fig. 7, which displays the result trajectories of KMA and TEB obtained through 100 repeated experiments. It is evident that KMA generates more consistent driving paths than TEB, thereby enabling humans to predict the movement of the robot. These observations are consistent with the quantitative analysis presented in Table. II. KMA's simple movements and smooth obstacle avoidance behaviors resulted in an average travel time reduction of approximately 4.6 seconds compared to TEB. Furthermore, the shorter standard deviations of travel time (0.44 s), linear distance (0.16 m), and angular distance (0.21 rad) of KMA compared to TEB (4.02 s, 0.69 m, 2.0 rad) reaffirm KMA's consistency in movement relative to TEB. Comparing the velocity and acceleration profiles depicted as black and red lines in Fig. 6, both planners adhere well to the velocity limits $\mathbf{v}^{max}$=[1.0 m/s, 1.0 rad/s]^T. However, regarding acceleration limits $\mathbf{a}^{max}$=[0.5 m/s^2, 1.0 rad/s^2]^T, KMA consistently adheres to acceleration limits demonstrating its feasibility, while TEB exceeds its limits by up to 24% and 83% over the linear and angular acceleration limits, respectively, due to the use of soft constraints with a penalty function for acceleration.

The oscillations that stand out in KMA's angular acceleration are attributed to the temporary fluctuations in θ_s at each planning iteration. Nevertheless, these oscillations are so fleeting that they do not have a significant impact on velocity profiles, making them hardly noticeable in the actual movement of the robot.

B. Experiments in the real-world

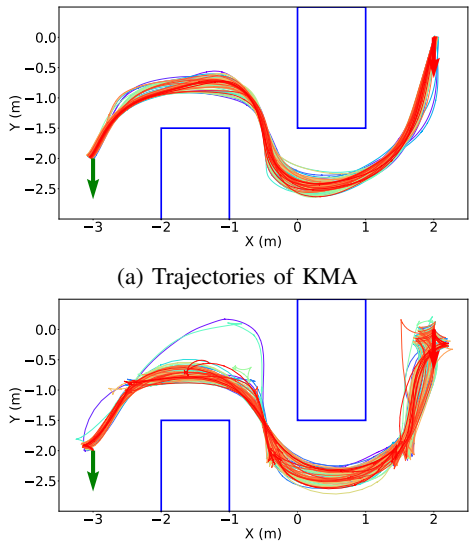
We also conducted experiments in the real environment with both static and dynamic obstacles, as illustrated in



(a) KMA RViz snapshots (b) Linear profile of KMA (c) Angular profile of KMA

(d) TEB RViz snapshots (e) Linear profile of TEB (f) Angular profile of TEB

Fig. 6: The results of KMA(a-c) and TEB(d-f) from simulation experiments: (a,d) RViz snapshots illustrating trajectories with $\mathbf{M}_l$, (b,e) linear velocity and acceleration and, (c,f) angular velocity and acceleration. Each snapshot displays global paths as light green lines, the robot's footprint as a light green rectangle, local paths as pink arrows, and the robot's trajectory as red arrows. In KMA snapshots, $\mathbf{W}_p$ is depicted in light green circular markers, candidate local waypoints in yellow circles and their corresponding trajectories in blue lines same as in Fig. 3. Additionally, for better understanding, each snapshot includes an approximate timestamp at the upper left and an inset image showing a magnified view of the robot's surroundings at the bottom right.



(a) Trajectories of KMA

(b) Trajectories of TEB

Fig. 7: Trajectory results from the simulation : Each trajectory is depicted in a different color across 100 repetitions. Initial and destination poses are marked with green and red arrows, and blue boxes represent static obstacles not represented on the map.

Fig. 5b, where specific scenarios are numbered from 1 to 4. Scene 1 and 4 depicts the initial and destination pose near static obstacles marked by blue boxes, while scene 2 illustrates a scenario where a dynamic obstacle suddenly appears from a corner. In scene 3, a person blocks the robot's path, approaching so closely that the robot cannot avoid.

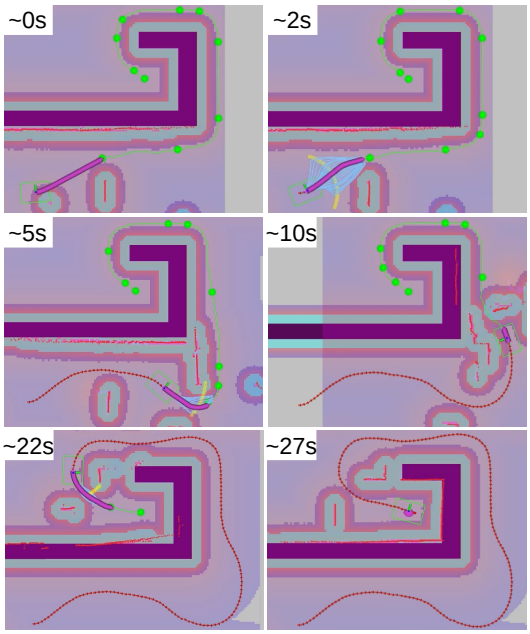
Fig. 8a shows RViz snapshots in accordance with each scenarios and Fig. 8b demonstrates its velocity and angular profiles. Obstacle avoidance trajectories were used in the snapshots around 2 and 22 seconds to navigate around the static obstacles, and also in the snapshot around 5 seconds to avoid a dynamic obstacle emerging from a blind spot. At around 10 seconds, a stop trajectory was generated as intended when no feasible trajectory could be made. Finally, at around 27 seconds, a focusing trajectory was generated to align with the destination orientation.

From the repeatedly conducted simulation results, we can validate the features of the KMA local planner compared to TEB local planner and demonstrate its applicability through the real-world experiment. Further exploration of the experiment's results can be found in the attached video[2], which includes comparative results with TEB in real-world experiments, along with additional experiments conducted in environments similar to simulation conditions.

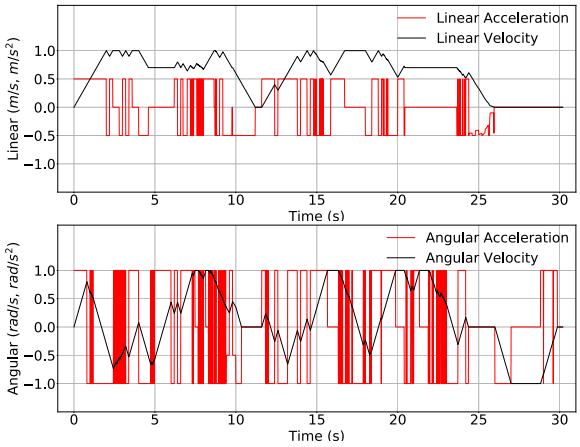
VI. CONCLUSIONS

In this paper, we presented a local trajectory planning method that is smooth and easy to predict the robot's movement by defining three types of basic trajectories considering the kinodynamic constraints of DWMRs and using them appropriately according to the situation. The suggested approach involves a obstacle avoidance trajectory generation by creating local waypoints around the robot. The proposed local planner algorithm can generate consistent trajectories without unnecessary movements compared to the TEB local planner. Additionally, it has been demonstrated through both simulation and real-world experiments that the proposed local planner produces straightforward and smooth movements while generating an obstacle avoidance trajectory in both unknown static and dynamic obstacle environment.

[2]Video Link: https://www.youtube.com/watch?v=03Vw4ZqFMeQ



(a) RViz Snapshots



(b) Linear and angular profile

Fig. 8: The results of the KMA experiment in the real-world scenario. See the caption in Fig. 6 for further details of the results.

In the current algorithm, there is no consideration for cases where obstacles occur during in-place rotation at the destination to align with the destination angle, or when obstacles are too close and there exist no obstacle avoidance trajectories. For future work, we plan to incorporate an supplementary basic trajectory for parking scenarios and implement backward local trajectory generation to evade obstacles that are too close in front.

REFERENCES

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik 1*, vol. 1, no. 1, pp. 269-271, 1959.

[2] P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[3] A. Stentz, "The focussed D* algorithm for real-time replanning," *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 95, pp. 1652–1659, 1995.

[4] L. Kavraki, P. Svestka, J.-C. Latombe and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transitions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[5] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Department of Computer Science, Iowa State University, America, IA, UAS, Technical Report No. 98-11, 1998.

[6] T. Jianping and S. X. Yang, "Genetic algorithm based path planning for a mobile robot," *IEEE International Conference on Robotics and Automation*, vol. 1, pp 1221-1226, 2003.

[7] E. Magid, D. Keren, E. Rivlin and I. Yavneh, "Spline-Based Robot Navigation," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2296–2301, 2006.

[8] M. Elbanhawi, M. Simic and R. N. Jazar, "Continuous path smoothing for car-like robots using b-spline curves," *Journal of Intelligient & Robotic Systems*, vol. 80, no. 1, pp. 23–56, 2015.

[9] D. Lee, H. Song and D. H. Shim, "Optimal path planning based on spline-RRT for fixed-wing UAVs operating in three-dimensional environments," *Proceedings of International Conference on Control, Automation and Systems*, pp. 835–839, 2014

[10] D. J. Webb and J. van den Berg, "Kinodyanmic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5054-5061, 2013.

[11] Y. Kim and B. K. Kim, "Time-optimal trajectory planning based on dynamics for differential-wheeled mobile robots with a geometric corridor," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 7, pp. 5502-5512, 2017.

[12] Y. Kim and B. K. Kim, "Global Trajectory Planning Based on DWMR Dynamics in Circular C-space." *International Journal of Control, Automation and Systems*, vol. 17, no. 10, pp. 2624-2633, 2019.

[13] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The international journal of robotics research*, pp. 396-404, 1986.

[14] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179-1187, 1989.

[15] X. Zhang, T. Zhu, L. Du, Y. Hu and H. Liu, "Local Path Planning of Autonomous Vehicle Based on an Improved Heuristic Bi-RRT Algorithm in Dynamic Obstacle Avoidance Environment," *Sensors*, vol. 22, no. 20, pp. 7968, 2022.

[16] M. Duguleana and G. Mogan, "Neural networks based reinforcement learning for mobile robots obstacle avoidance," *Expert Systems with Applications*, vol. 62, pp. 104-115, 2016.

[17] C. Rosmann, W. Feiten, T.Wosch, F. Hoffmann and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," *Proceedings of the ROBOTIK 2012, 7th German Conference on Robotics*, pp. 1–6, 2012.

[18] C. Rosmann, W. Feiten, T. Wosch, F. Hoffmann and T. Bertram, "Efficient trajectory optimization using a sparse model", *European Conference on Mobile Robots*, pp. 138-143, 2013.

[19] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," *Proceedings of IEEE International Conference on Robotics and Automation*, vol 2, pp. 802-807, 1993.

[20] D. Fox, W. Burgard and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23-33, 1997.

[21] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Transition on Robotics*, vol. 25, no. 4, pp. 902-911, 2009.

[22] A. Ravankar, A. A. Ravankar, Y. Kobayashi, Y. Hoshino and C.-C. Peng, "Path smoothing techniques in robot navigation: State-of-the-art, current and future challenges," *Sensors*, vol. 18, no. 9, p. 3170, Sep. 2018.

[23] https://gazebosim.org/home

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler and A. Ng, "Ros: An open-source robot operating system," *ICRA workshop on open source software*, vol. 3, pp. 5, 2009.

[25] D. Fox, W. Burgard, H. Kruppa and S. Thrun, "Monte Calro localization: Efficient position estimation for mobile robots", *Aaai/iaai*, pp. 343-349, 1999.