

Trajectory Tracking Control and
Mechanism Development of an
Omnidirectional Mobile Robot with
Slidable Wheels

XU HUANG

2026

Trajectory Tracking Control and Mechanism Development of an Omnidirectional Mobile Robot with Slidable Wheels

XU HUANG

Department of Mechanical Engineering and Science
Graduate School of Engineering, Kyoto University

A thesis submitted for the degree of
Doctor of Engineering

2026

Contents

Chapter 1 Introduction.....	1
1.1 Background and motivation.....	1
1.2 Relative researches.....	4
1.2.1 Trajectory tracking and model predictive control	4
1.2.2 Deep reinforcement learning.....	6
1.3 Thesis organization	7
Chapter 2. Trajectory tracking control of SWOM using linear model predictive control	9
2.1 Introduction.....	9
2.2 Kinematic analysis of slidable-wheel omnidirectional mobile robot.....	9
2.3 Reference path generation.....	12
2.4 Trajectory tracking for SWOM.....	14
2.4.1 Linearization of kinematic equation.....	14
2.4.2 Trajectory tracking control using model predictive control	15
2.5 Experiment.....	21
2.5.1 Prototype of SWOM	21
2.5.2 Experiments results	22
2.6 Conclusion	29
Chapter 3. Deep-reinforcement-learning-based controller design for SWOM.....	31
3.1 Introduction.....	31
3.2 DRL-based hierarchical control method for trajectory tracking	32
3.2.1 Problem description	32
3.2.2 Upper-level controller	34
3.2.3 Lower-level controller.....	35
3.3. Numerical simulations	39
3.3.1 Numerical simulation of agents trained with different reward designs	40
3.3.2 Trajectory tracking simulation of SWOM without considering wheel slippage	40
3.3.3 Trajectory tracking simulation of SWOM with wheel slippage consideration	42
3.4 Conclusion	44
Chapter 4. Novel omnidirectional mobile robot with slidable base	46
4.1 Introduction.....	46
4.2 Structure and basic movement	46
4.2.1 Synthesis of slidable-base omnidirectional mobile robot.....	46
4.2.2 Motion of slidable-base omnidirectional mobile robot	48
4.2.3 Characteristics of slidable-base omnidirectional mobile robot	50
4.3 Kinematic analysis of slidable-base omnidirectional mobile robot	51
4.3.1 Modeling of slidable-base omnidirectional mobile robot	51
4.3.2 Omnidirectional movement of slidable-base omnidirectional mobile robot.....	52
4.4 Stabilization of slider position on rail	57
4.5. Experiments and discussion	60
4.5.1 Prototype of slidable-base omnidirectional mobile robot	60
4.5.2 Experiments on unlimited translation	62
4.6 Conclusion	65

Chapter 5. Conclusion	66
5.1 Summary of contributions.....	66
5.2 Future work.....	67
Reference.....	69
List of publication	77
Acknowledgment.....	78

Chapter 1 Introduction

1.1 Background and motivation.

In recent decades, societies across the globe, particularly in Japan, Europe, and increasingly in China and other parts of Asia, are facing a profound demographic shift characterized by declining birth rates, shrinking working-age populations, and rapid aging. This demographic transition has led to chronic labor shortages in sectors that rely heavily on physical work, such as manufacturing, transportation, retail, and eldercare. In response, there is an urgent and growing demand for automation technologies that can supplement or replace human labor in routine, repetitive, or physically demanding tasks.

Mobile robots have emerged as a key enabler of this automation wave and are already widely deployed for material transportation in factories and warehouses, where they enhance operational efficiency while reducing labor costs. However, the demand for automation is no longer limited to highly structured industrial environments. In retail settings, such as supermarkets and convenience stores, tasks including shelf restocking, customer guidance are increasingly being targeted for robotic implementation. Similarly, in last-mile delivery and indoor service applications, mobile platforms are being integrated with robotic manipulators or humanoid upper bodies to create versatile, multi-functional service agents.

For such applications, the mobile base must support agile navigation through dynamic, crowded, and spatially constrained indoor environments. In this context, omnidirectional mobile robots offer significant advantages over conventional differential-drive counterparts. Unlike differential-drive robots, which require rotational maneuvers to reorient before lateral movement, an omnidirectional platform can instantaneously translate in any direction. This capability enables seamless navigation through narrow aisles, rapid obstacle avoidance, and precise alignment with shelves, counters, or interaction points, all without the need for additional maneuvering space. Consequently, omnidirectional mobility is not merely a performance enhancement but a critical enabler for effective operation in complex human-centric environments.

Omnidirectional mobile robots can be roughly divided based on the moving parts into four types: wheeled robots [1–8], legged robots [9–12], wheel-legged robots [13–15], and tracked robot[16,17]. Wheeled omnidirectional mobile robots are the most common because of their low energy consumption, simple structure, and high stability. The non-holonomic constraints of conventional wheeled mobile robots can be removed with mechanisms such as Omni wheels [18,19], Mecanum wheels [20,21], caster wheels [22–25], and spherical wheels [26–30].

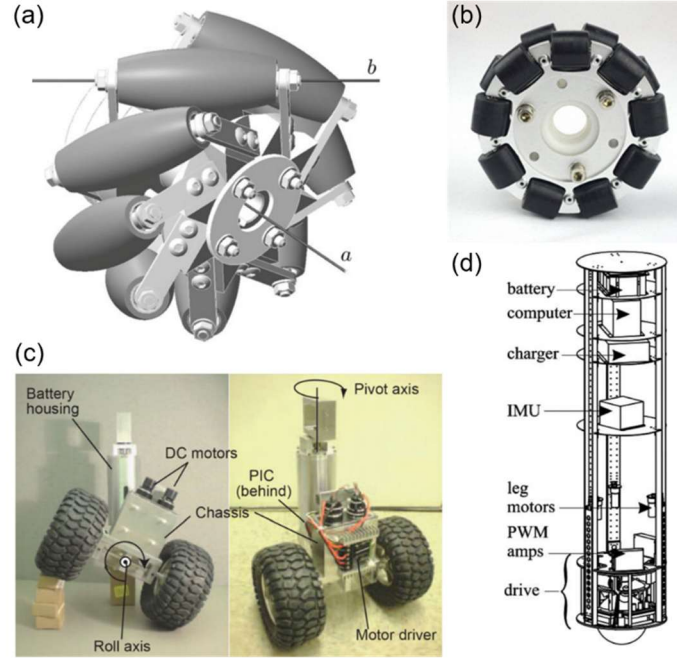


Fig. 1.1 Omnidirectional mechanisms. (a) Mecanum wheel, adopted from [31]; (b) Omni wheel[18]; (c) active split offset caster [32]; (d) spherical wheel robot called Ballbot [33];

The Mecanum wheel shown in Fig. 1.1(a) consists of a hub and several free rollers attached to its periphery at a 45-degree angle. Compared to robots equipped with conventional wheels, robots equipped with Mecanum wheels often have a lower load capacity due to the complicated structure of such wheels. The discontinuous contact between the rollers and the ground results in vibrations, especially on uneven ground. In contrast, the omni wheel, depicted in Fig. 1.1(b), comprises a central hub with multiple small passive rollers mounted orthogonally along its circumference. Compared to Mecanum wheels, omni wheels generally offer higher side-slip efficiency and lower mechanical complexity, which can lead to reduced manufacturing cost and improved energy efficiency during transverse movement. However, the discrete nature of the roller-ground contact similarly introduces vibrations on rough surfaces, and the load-bearing capacity remains limited due to the reliance on small-diameter rollers for support. Furthermore, the exposed rollers are susceptible to entanglement with debris or fibers, which may compromise reliability in unstructured environments. A caster wheel, shown in Fig. 1.1(c), has a higher load capacity but its rotation around the steering axis has a small amount of pause [34]. In addition, it may cause shock or slippage and increase the uncertainty in the positioning of the robot. Figure 1.1(d) shows a robot that uses a spherical wheel. This type of wheel has a continuous contact point with the ground, so it does not produce vibrations. However, since it is necessary to use friction to rotate the spherical wheel, compensation for slippage is necessary. Moreover, the large volume of a spherical wheel makes the movement device larger.

To overcome the disadvantages mentioned above, a slidable-wheeled omnidirectional mobile robot (SWOM) was previously proposed by Terakawa et al. [35], as shown in Fig. 1.2.

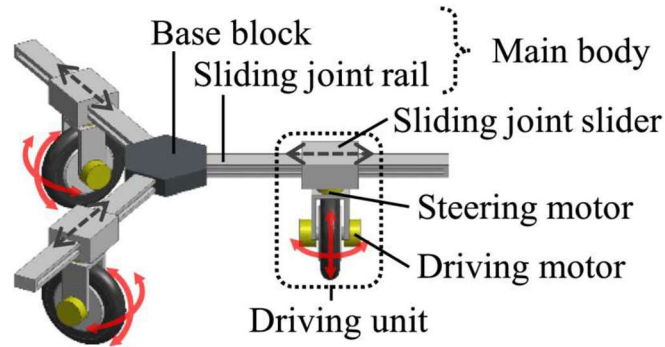


Fig. 1.2 Diagram of slidable-wheel omnidirectional mobile robot (SWOM) [35]

Structurally, SWOM comprises a rigid central base block connected to three identical driving units via three linear sliding rails arranged radially at 120° intervals. Each driving unit incorporates a standard conventional wheel equipped with independent actuators for active rotation and active steering. By utilizing conventional wheels instead of specialized omnidirectional mechanisms such as Mecanum or omni wheels, SWOM can inherit their inherent advantages, including high load capacity, mechanical robustness, low vibration during operation, and compact footprint. Critically, each driving unit is also capable of passive sliding along its dedicated rail. This additional degree of freedom (DOF) decouples the wheel's orientation from the robot's body frame, thereby eliminating the non-holonomic constraints typically associated with conventional wheeled systems. As a result, SWOM achieves true omnidirectional mobility. As schematically illustrated in Fig. 1.3, coordinated control of the wheel orientations and driving torques enables the robot to execute a full range of planar motions, including forward and backward translation, lateral strafing, diagonal movement, and in-place rotation, without requiring reorientation maneuvers.

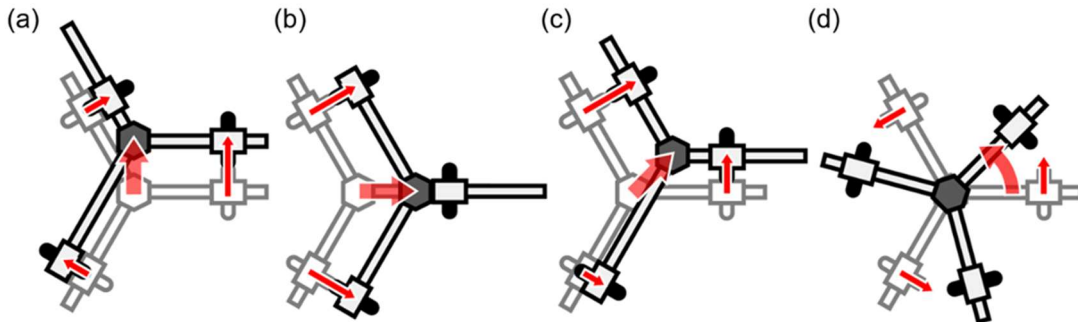


Fig. 1.3 Omnidirectional movement of SWOM when the movement direction of each wheel is perpendicular to the sliding joint. (a) Moving forward. (b) Moving to the right. (c) Moving diagonally forward right. (d) Turning. The upward direction in the figure is the forward direction. [35]

Despite these advantages, SWOM introduces significant control challenges that hinder its practical deployment. First, the system requires six actuators (three for wheel rotation, three for steering), exceeding the three DOFs needed for planar omnidirectional motion, resulting in a redundant and over-actuated system. Second, the passive sliding motion is not directly controlled; it emerges from interactions between wheel forces, body inertia, and rail constraints. This can lead to unintended motions

or instability if not properly managed. Third, the overall systems are strongly nonlinear, and each conventional wheel remains subject to non-holonomic constraints, complicating trajectory planning and feedback control [36]. Most critically, the sliding joints must remain within bounded ranges on their rails; if a joint reaches an end-stop, the robot enters a singular configuration where omnidirectional mobility is lost. Therefore, ensuring that all sliding positions stay within safe operational limits during motion is a fundamental requirement for stable control.

Consequently, the primary challenge addressed in this thesis is how to achieve accurate and robust control for SWOM despite its structural complexity, redundancy, and inherent nonlinearities. From a software perspective, this necessitates advanced control strategies capable of handling high-dimensional, constrained, and nonlinear systems. From a hardware perspective, it raises the question of whether the mechanical architecture itself can be rethought to reduce complexity while preserving omnidirectional capability.

1.2 Relative researches.

1.2.1 Trajectory tracking and model predictive control

Trajectory tracking control has been extensively investigated in the literature [37–39]. By definition, trajectory tracking refers to the capability of controlling a system such that it accurately follows a predefined reference trajectory, which includes not only spatial position but also temporal information. For instance, in mobile robotics, this involves tracking the desired trajectory of a robot's position and orientation in a plane or three-dimensional space; similarly, for robotic manipulators, it entails controlling the end-effector's pose (position and orientation) to follow a specified trajectory [40,41]. Recently, dynamic behaviors of humanoid robots [42,43]—such as dancing, standing, and walking—can also be interpreted as pose trajectory tracking tasks. The primary challenges in trajectory tracking problems arise from system nonlinearities, external and internal noise, and the increasing dimensionality of the state space.

Due to the broad scope of systems and applications involved in trajectory tracking problems, a wide variety of control methodologies have been employed. For relatively simple systems, classical control theory—such as PID control or lead-lag compensation—can be directly applied to achieve satisfactory tracking performance. Sliding mode control has been applied to trajectory tracking in various fields, including underwater vehicles [44,45], flapping-wing micro aerial vehicles [46], and mobile robots [47]. A trajectory tracking control method for automated guided vehicles based on the Udwadia-Kalaba approach was proposed by Yu et al. [48]. Trajectory tracking from the perspective of port-controlled Hamiltonian systems was discussed by Fujimoto et al. [49].

Additionally, Model Predictive Control (MPC) is also widely employed for trajectory tracking tasks [5,50,51]. The fundamental concept of MPC is to predict the process output along a future time horizon using the system model and calculate a control sequence to optimize a performance index [52]. Therefore, the performance of the controller depends on how well the dynamics of the system being captured by the input–output model that is used for the design of the controller [53].

The overall procedure of MPC can be illustrated with Fig. 1.4.

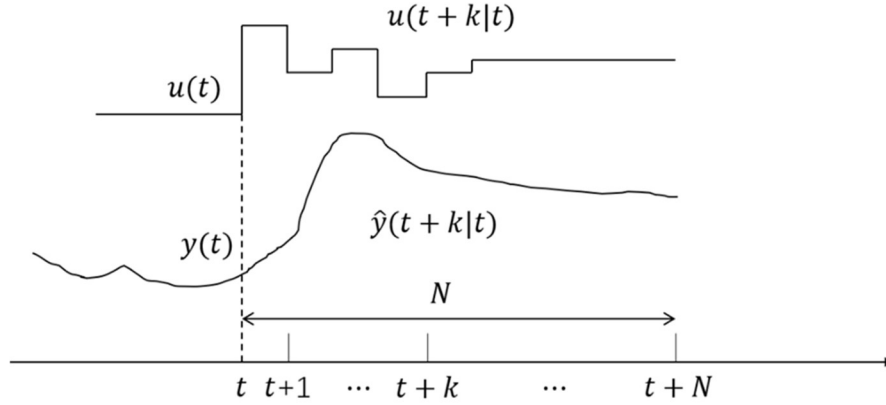


Fig. 1.4 Model predictive control strategy [54].

At each discrete time step t , MPC performs the following sequence of operations: first, the current observation $y(t)$ of the system is measured. Subsequently, based on a dynamic model of the plant, a finite-horizon optimal control problem is formulated and solved over the prediction horizon $[t, t + N]$, subject to system constraints on states and control inputs. The cost function to be minimized typically includes weighted terms for tracking error and control effort. The solution yields an optimal sequence of future control inputs $\{u(t), u(t + 1|t), \dots, u(t + N - 1|t)\}$. Only the first element of this sequence, $u(t)$, is implemented and applied to the system as the control input at time t . At the next time step $t + 1$, the process is repeated with the updated state measurement $y(t + 1)$, resulting in a receding horizon implementation that provides inherent feedback and robustness to disturbances and modeling inaccuracies. Due to its conceptual simplicity and ability to handle complex systems with strict control constraints and multiple inputs and outputs [55], MPC is widely used in various industries, such as process control [56], power electronics [57], and manufacturing [58,59].

Depending on whether the optimization solver employed can handle nonlinear programming problems, MPC can be classified into Nonlinear MPC (NMPC) and Linear MPC (LMPC). For linear systems, LMPC is typically sufficient for effective control. However, for nonlinear systems, NMPC is generally required to adequately capture the system dynamics. In NMPC, the associated optimization problems are inherently nonlinear and must be solved in real time, leading to significantly high computational costs. To mitigate this burden, an alternative approach is to linearize the system model around the current operating point or along the reference trajectory, and subsequently apply LMPC based on the linearized model, which offers a favorable trade-off between performance and computational efficiency.

While MPC has been successfully applied to a wide range of mobile robots, including those equipped with Mecanum or omni wheels, its application to mechanisms such as SWOM remains largely unexplored. A primary challenge stems from the strong nonlinear coupling among steering motion, wheel driving, and passive sliding, which significantly complicates real-time implementation. If NMPC is employed, substantial computational resources are required to solve the optimization problem within the

prescribed control interval. On the other hand, LMPC demands carefully designed linearization strategies that accurately preserve the essential kinematic characteristics of the system. Addressing these challenges constitutes a key motivation for adapting and applying MPC techniques to SWOM platform in this work.

1.2.2 Deep reinforcement learning

Reinforcement learning (RL) enables the construction of autonomous decision-making agents that learn optimal policies through trial and error during their interaction with a dynamic environment. The specific interaction paradigm is illustrated in Fig.1.5: the agent selects an action based on the current state, the environment transitions to a new state accordingly, and returns an immediate reward signal. The agent then improves its behavioral policy by evaluating the long-term cumulative reward.

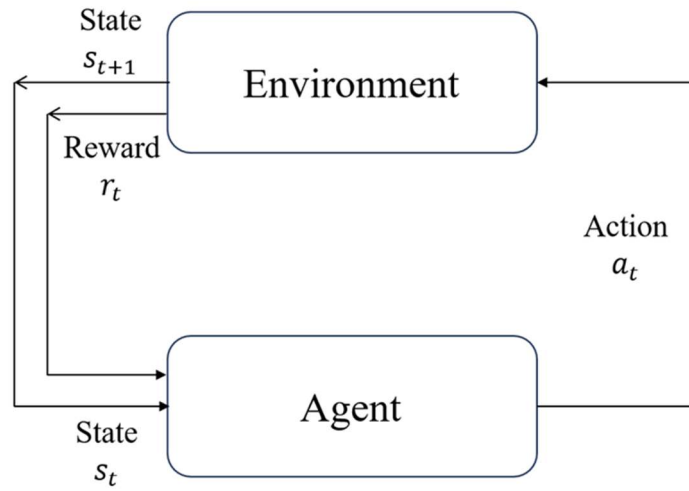


Fig. 1.5 Illustration of the agent-environment interaction loop

Traditional RL methods, such as Q-learning, SARSA, and policy gradient algorithms, have achieved considerable success in tasks with low-dimensional, discrete, and well-structured state and action spaces. However, traditional RL approaches face significant challenges when applied to problems involving high-dimensional and continuous state and action spaces. A key bottleneck lies in the limited representational capacity of conventional function approximators, which struggle to capture complex, nonlinear patterns in large-scale input spaces. To overcome this limitation, deep neural networks have been integrated into the reinforcement learning framework, giving rise to Deep RL (DRL).

DRL exploits the powerful nonlinear approximation and hierarchical feature extraction capabilities of deep networks, enabling agents to learn both state representations and control policies directly from raw, high-dimensional observations. This facilitates end-to-end learning of perception and control. A landmark advancement was the Deep Q-Network (DQN) [60] introduced by DeepMind, which combined convolutional neural networks with Q-learning to achieve human-level performance on a variety of Atari games using only pixel inputs. Since then, numerous advanced DRL algorithms, including Deep Deterministic Policy Gradient (DDPG) [61], Soft Actor-Critic (SAC) [62], and Proximal Policy Optimization (PPO) [63] have been developed, significantly enhancing the adaptability and decision-

making capabilities of agents in complex and uncertain environments.

Unlike model-based control methods such as MPC or Linear Quadratic Regulators (LQR) which rely heavily on accurate mathematical models of system dynamics and are sensitive to modeling errors, DRL adopts a model-free paradigm. This allows DRL to operate effectively in scenarios where system identification is difficult or impractical. Moreover, DRL can inherently adapt to environmental changes and unmodeled disturbances through online learning. However, this flexibility comes at a cost: DRL typically suffers from high sample complexity, substantial computational demands, and limited interpretability. Training processes are often unstable and prone to poor generalization if not carefully designed. Furthermore, the design of reward functions remains a critical challenge. There is currently no universal methodology for crafting effective rewards, and in practice, it requires extensive manual tuning and domain expertise, entailing significant engineering effort.

In recent years, DRL has achieved remarkable progress in the field of robotics, demonstrating strong capabilities in enabling autonomous control and adaptive decision-making across a wide range of platforms. Successful applications have been reported in humanoid robots [64,65], robotic manipulators [66,67], unmanned aerial vehicles (UAVs) [68,69], and mobile robotic systems [70–72].

Building on this foundation, several studies have specifically focused on trajectory tracking tasks using DRL. For instance, DRL-based approaches have been developed for end-effector trajectory tracking in robotic arms [73–75], as well as for path following in autonomous underwater vehicles [76], UAVs [77,78], and wheeled mobile robots [79,80]. These efforts highlight the growing potential of DRL in achieving precise and robust motion control under complex and uncertain environments. However, for wheeled mobile robots, most current research uses relatively simple models, which are simplified versions of existing vehicles with two or three DOFs. For example, Tan et al. [81] employed a bicycle model, Srikonda et al. [82] studied skid-steered wheeled mobile robots, Liu et al. [83] focused on distributed drive vehicles, and Chai et al. [84] and Lian et al. [85] considered a four-wheeled mobile robot with passive front wheels and differential-driven rear wheels, respectively. There is relatively little research on trajectory tracking problems that involve wheeled mobile robots with complex structures and higher DOFs, which motivates my investigation on the use of DRL-based trajectory tracking for such robots.

1.3 Thesis organization

The objective of this thesis is to advance the practical application of SWOM from both software and hardware perspectives, with a primary focus on the three themes outlined above. The research outcomes are presented in the subsequent chapters. A brief review of this thesis is as follows:

Chapter 2 presents the trajectory tracking controller for SWOM based on the LMPC approach. An original method for generating a reference path that includes not only state variables but also inputs is proposed. The linearized kinematic model of SWOM is obtained using a first-order Taylor expansion around reference points on the reference path. Simulations considering slippage are conducted and the results show that SWOM can well track the reference path. Experiments conducted on a prototype also

validate the effectiveness of the proposed control method.

Chapter 3 proposes a control framework that integrates DRL with hierarchical control to achieve DRL-based trajectory tracking for SWOM. The upper-level controller computes the required velocity of the robot body based on the error between the current and target positions, thereby regulating the wheel's rotational speed. The lower-level controller consists of three DRL-trained sub-controllers, each responsible for steering one of SWOM's three wheels, ensuring that the passive components of SWOM remain within an acceptable range. The design of the reward function in reinforcement learning, as well as the impact of parameter variations on training outcomes, is discussed. The effectiveness of the proposed controller is validated through numerical simulations conducted under two scenarios: one without considering wheel slippage and the other with wheel slippage taken into account. The proposed controller demonstrates satisfactory performance in both cases.

Chapter 4 proposes an omnidirectional mobile robot with a simpler structure based on SWOM. The synthesis of the proposed robot, including the basic structure and basic movement patterns, is first presented. A kinematic model is established, and the singular configuration of the robot is analyzed. An original movement strategy for avoiding singularities is developed considering the periodic input and stabilization of the slidable base. The feasibility of the new structure and its movement patterns is validated through simulations and experiments.

Chapter 5 provides a summary of each chapter and discusses directions for future research.

Chapter 2. Trajectory tracking control of SWOM using linear model predictive control

2.1 Introduction

In the previous chapter, I outlined the challenges of trajectory tracking for robots and reviewed commonly adopted approaches. This chapter focuses on an MPC-based scheme. Compared with intelligent control, model-based methods generally offer superior interpretability. Owing to its inherent advantage in handling constraints, MPC has been extensively applied in mobile robot control. For SWOM, constructing its kinematic model and designing an MPC controller accordingly can provide a reference baseline performance for subsequent learning-based approaches.

Due to its low computational complexity, ease of implementation, and ability to handle constraints on inputs and outputs, LMPC was adopted to achieve trajectory tracking for SWOM. Building on the kinematic equations of SWOM proposed in previous research, this study introduces a novel method for generating reference trajectories. The generated reference trajectories encompass not only all state variables of SWOM but also the corresponding reference inputs. Subsequently, the linear motion equations of SWOM were derived through a first-order Taylor expansion at the reference points along the trajectory. Finally, the LMPC method was successfully applied to achieve trajectory tracking for SWOM, and its effectiveness was validated through both simulations and prototype experiments.

The rest of this chapter is organized as follows: Section 2 reviews the kinematic model and singularity for SWOM. Section 3 presents the proposed methodology for generating reference trajectories and motion planning. Section 4 describes the linearization of the kinematic equations for SWOM and the application of LMPC to the trajectory tracking problem. A numerical simulation is conducted to verify the effectiveness of the proposed method. Section 5 presents and discusses the experimental results and Section 6 concludes this chapter.

2.2 Kinematic analysis of slidable-wheel omnidirectional mobile robot

The global reference frame and local reference frame are shown in Fig. 2.1(a). The origin of the local reference frame is fixed at the center of the base block, with the X_R axis parallel to a sliding rail. To distinguish the three driving units, the unit that slides on the rail parallel to the X_R axis is labeled as driving unit 1. The other two driving units are labeled as driving units 2 and 3, respectively, in the counterclockwise direction. Figure 2.1(b) shows more detailed parameters describing SWOM's configuration in the local reference frame. The angle between each sliding rail and the X_R axis is denoted as α . l is the displacement of the slider, r is the wheels' radius, $\dot{\phi}$ is the rotation speed of the wheel, and β is the angle between the wheel axle and the sliding rail indicating the orientation of

the wheel. I use the subscript i to indicate the parameters belonging to driving unit i ($i = 1, 2, 3$).

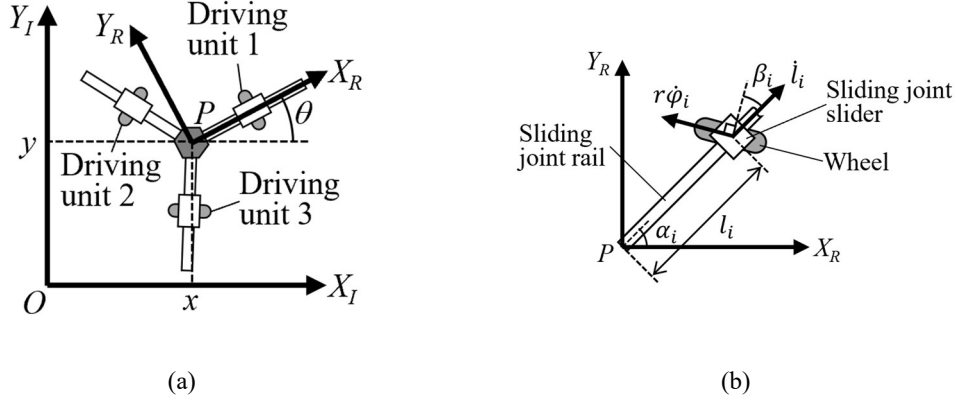


Fig. 2.1 (a) Global and local reference frames for SWOM. (b) Wheel parameters in the local reference frame [36].

The kinematic model of SWOM is derived as:

$$[\sin(\alpha_i + \beta_i) \quad -\cos(\alpha_i + \beta_i) \quad -l_i \cos \beta_i] R(\theta) \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} + r \dot{\phi}_i + \dot{l}_i \sin \beta_i = 0 \quad (2.1)$$

$$[\cos(\alpha_i + \beta_i) \quad \sin(\alpha_i + \beta_i) \quad l_i \sin \beta_i] R(\theta) \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} + \dot{l}_i \cos \beta_i = 0 \quad (2.2)$$

where

$$R(\theta) = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

and $\dot{X}$ represents the velocity of SWOM along the OX_I axis in the global coordinate system, while $\dot{Y}$ represents its velocity along the OY_I axis in the same system.

These equations are derived under the assumption of no-slip conditions, based on the relationship between SWOM's body velocity and the velocities of its three wheels. Equation (2.1) describes the scenario where, under no-slip conditions, the actual rotational speed of the wheel equals the velocity component of the point on the sliding rail coinciding with the wheel in the local coordinate system, projected in the direction parallel to the wheel orientation, plus the displacement velocity of the wheel along the sliding rail in the same direction. Under no-slip conditions, the wheel's velocity component perpendicular to its orientation is zero. Accordingly, Eq. (2.2) states that the velocity component of the

point on the sliding rail coinciding with the wheel in the local coordinate system, projected in the direction perpendicular to the wheel orientation, plus the displacement velocity of the wheel along the sliding rail in the same perpendicular direction, must sum to zero.

By combining Eqs. (2.1) and (2.2), $\dot{l}$ can be eliminated. The forward and inverse kinematics can be expressed by Eqs. (2.4) and (2.5), respectively.

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} = R(\theta)^{-1} \begin{bmatrix} -\sin \alpha_1 & \cos \alpha_1 & l_1 \\ -\sin \alpha_2 & \cos \alpha_2 & l_2 \\ -\sin \alpha_3 & \cos \alpha_3 & l_3 \end{bmatrix}^{-1} \begin{bmatrix} r \cos \beta_1 & 0 & 0 \\ 0 & r \cos \beta_2 & 0 \\ 0 & 0 & r \cos \beta_3 \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} \quad (2.4)$$

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} = \begin{bmatrix} r \cos \beta_1 & 0 & 0 \\ 0 & r \cos \beta_2 & 0 \\ 0 & 0 & r \cos \beta_3 \end{bmatrix}^{-1} \begin{bmatrix} -\sin \alpha_1 & \cos \alpha_1 & l_1 \\ -\sin \alpha_2 & \cos \alpha_2 & l_2 \\ -\sin \alpha_3 & \cos \alpha_3 & l_3 \end{bmatrix} R(\theta) \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\theta} \end{bmatrix} \quad (2.5)$$

The robot will fall into a singular configuration if the orientation of any of the wheels is parallel to the sliding rail. In this paper, I constrain β to the range of $(-\pi/2, \pi/2)$ to avoid the singularity. Substituting $\alpha_1 = 0, \alpha_2 = 2\pi/3$, and $\alpha_3 = 4\pi/3$ into the inverse kinematic equation yields

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} = \begin{bmatrix} r \cos \beta_1 & 0 & 0 \\ 0 & r \cos \beta_2 & 0 \\ 0 & 0 & r \cos \beta_3 \end{bmatrix}^{-1} * \begin{bmatrix} -\dot{X} \sin \theta + \dot{Y} \cos \theta + \dot{\theta} l_1 \\ \dot{X} \left(-\frac{\sqrt{3}}{2} \cos \theta + \frac{1}{2} \sin \theta \right) + \dot{Y} \left(-\frac{\sqrt{3}}{2} \sin \theta - \frac{1}{2} \cos \theta \right) + \dot{\theta} l_2 \\ \dot{X} \left(\frac{\sqrt{3}}{2} \cos \theta + \frac{1}{2} \sin \theta \right) + \dot{Y} \left(\frac{\sqrt{3}}{2} \sin \theta - \frac{1}{2} \cos \theta \right) + \dot{\theta} l_3 \end{bmatrix} \quad (2.6)$$

Due to the limitations of the motors and the restricted length of the sliding rail, SWOM has constraints on both inputs and outputs. The constraints on the inputs can be satisfied by directly controlling their magnitude. From Eq. (2.2), when $-\pi/2 < \beta_i < \pi/2$, the time dependence of l_i can be expressed as:

$$\dot{l}_i = -\dot{X}(\cos(\theta + \alpha_i) - \tan \beta_i \sin(\theta + \alpha_i)) - \dot{Y}(\tan \beta_i \cos(\theta + \alpha_i) + \sin(\theta + \alpha_i)) - l_i \dot{\theta} \tan \beta_i \quad (2.7)$$

From Eq. (2.7), it is possible to keep l_i within the constraint limits by controlling the orientation of the wheel.

2.3 Reference path generation

Since the kinematic system is linearized using a first-order Taylor expansion at reference points on the reference path, a reference trajectory for SWOM that includes not only its configuration but also the corresponding inputs is necessary. I first define the target velocity for SWOM and then calculate the reference value for the wheels' rotational speed. Based on this, the orientation of each wheel is adjusted to ensure that l_i remains within the specified range. Consequently, reference values for l_i , β_i , and $\dot{\beta}_i$ are obtained. The procedures for reference path generation are as follows.

The inputs of the system are the rotation and steering velocity of the wheels, $\dot{\phi}_i$ and $\dot{\beta}_i$, which means the steering angle of the wheel cannot be directly controlled. At the k -th time step, the reference translation velocity $(\dot{X}_{ref}(k), \dot{Y}_{ref}(k))$ and rotation velocity $\dot{\theta}_{ref}(k)$ of SWOM are first specified. The configurations of SWOM, including $X(k), Y(k), \theta(k), l_i(k)$, and $\beta_i(k)$, are measured by sensors. The necessary rotation velocity of the three wheels $\dot{\phi}_{id}(k)$ can be calculated using Eq. (2.6).

Next, the reference steering velocity of the wheels $\dot{\beta}_{iref}(k)$ is designed to stabilize the displacement of the slider. Let $E_{li} = l_{iref} - l_{id}$ be the distance between the slider's position and the desired position, where l_{id} denotes the desired position of the slider. Generally, it is set to be at the middle of the sliding rail. If $\dot{l}_{iref} = -K_l E_{li}$, l_{iref} will converge to l_{id} . With $\dot{l}_{iref} = -K_l E_{li}$, from Eq. (2.7), the reference orientation of the wheel β_{iref} can be obtained as:

$$\beta_{iref} = \arctan \left(\frac{\dot{X}_{ref} \cos(\theta + \alpha_i) + \dot{Y}_{ref} \sin(\theta + \alpha_i) - K_l E_{li}}{\dot{X}_{ref} \sin(\theta + \alpha_i) - \dot{Y}_{ref} \cos(\theta + \alpha_i) - l_i \dot{\theta}_{ref}} \right) \quad (2.8)$$

At the k -th time step, the rate of change of the position of the slider $\dot{l}_{iref}(k)$ is first derived according to Eq. (2.7). Then, the position of the slider at the next time step $l_{ir}(k+1)$ and the distance between the slider and its desired position at the next time step $E_{li}(k+1)$ can be predicted. Using Eq. (2.8), the reference steering angle of the wheel for the next time step $\beta_{ref}(k+1)$ is derived to control the displacement of the slider. Thus, the reference angular velocity $\dot{\beta}_{iref}(k)$ can be calculated from the rate of change of the steering angle between two steps to ensure the angular velocity at the next step reaches the desired value (i.e., $\dot{\beta}_{iref}(k) = (\beta_{iref}(k+1) - \beta_{iref}(k))/dt$). If $\dot{X}_{ref} \sin(\theta + \alpha_i) - \dot{Y}_{ref} \cos(\theta + \alpha_i) - l_i \dot{\theta}_{ref} = 0$, $\beta_{iref}(k+1)$ becomes $\pi/2$ or $-\pi/2$, which violates the constraints placed on the steering angle of the wheel. For this situation, I simply set $\beta_{iref}(k+1)$ to the value at the previous step and $\dot{\beta}_{iref}(k) = 0$.

The reference position for SWOM can be calculated by integrating $\dot{X}_{ref}$, $\dot{Y}_{ref}$, and $\dot{\theta}_{ref}$. Finally, I obtain the reference path for SWOM, which consists of the reference configuration and the corresponding inputs.

An example of a reference trajectory and the corresponding reference position of the sliders are shown in Fig. 2.2. To demonstrate the robot's omnidirectional mobility, the reference trajectories are set to be unsmooth and non-differentiable. The target velocity of the robot is ($\dot{X} = 0.05$ m/s, $\dot{Y} = 0.00$ m/s)

for the first 20.0 s and ($\dot{X} = 0.00$ m/s, $\dot{Y} = 0.05$ m/s) for the next 20.0 s. The target rotation speed is 0.08 rad/s. The target position of the sliders is set to $l_{id} = 0.20$ m and the limits on the position of the sliders are $l_{\min} = 0.12$ m and $l_{\max} = 0.30$ m. K_i ($i = 1, 2, 3$) is set as 1. In the subsequent simulations and experiments, the value of K_i has also been set to 1. Figure 2.2(b) shows the reference position of the sliders as the robot moves along the reference trajectory. From the graph, it can be concluded that the position of the sliders can be limited to a small range from the specified position using the proposed method.

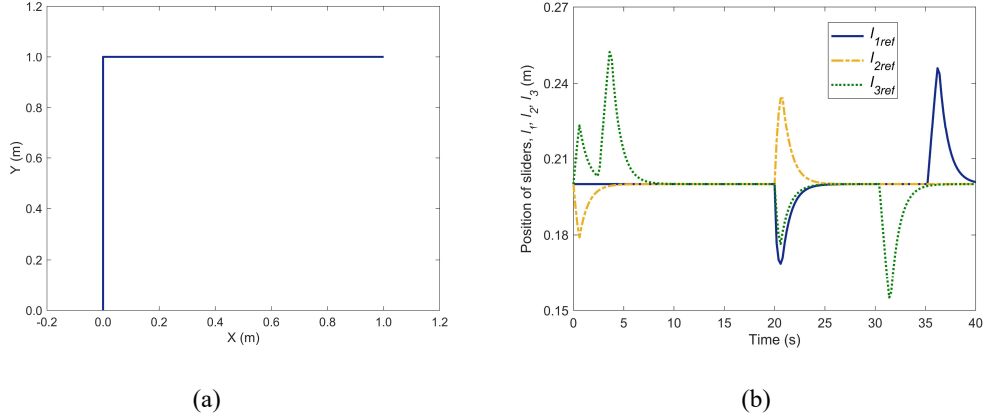


Fig. 2.2 (a) Reference path and (b) reference position for sliders on sliding rails.

The graph of the position of the sliders has some peaks. There are two main reasons for this phenomenon. First, when the velocity of the robot suddenly changes a lot, the orientation of the wheels β also needs to change a lot. Since I set an upper limit on the acceleration of β , the desired value is not reached immediately, so the displacement of the slider continues to change. Second, the direction of the velocity at the point on the robot that is in contact with an arbitrary slider is parallel to the direction of its sliding rail (i.e., $\dot{X}_{\text{ref}}\sin(\theta + \alpha_i) - \dot{Y}_{\text{ref}}\cos(\theta + \alpha_i) - l_i\dot{\theta}_{\text{ref}} = 0$). At this point, β_i needs to be $\pi/2$ or $-\pi/2$, which is prohibited, and Eq. (7) becomes:

$$\dot{l}_i = -\dot{X}_{\text{ref}}\cos(\theta + \alpha_i) - \dot{Y}_{\text{ref}}\sin(\theta + \alpha_i) \quad (2.9)$$

The rate of change of the displacement of the slider $\dot{l}_i$ equals the velocity of the robot in the direction of the sliding rail. Therefore, the displacement of the slider continues to change and the speed becomes the same as that of the robot in the direction of the sliding rail.

2.4 Trajectory tracking for SWOM

2.4.1 Linearization of kinematic equation

I first define the state variables and inputs for SWOM as:

$$\begin{aligned} q &= [X, Y, \theta, l_1, l_2, l_3, \beta_1, \beta_2, \beta_3], \\ U &= [\dot{\phi}_1, \dot{\phi}_2, \dot{\phi}_3, \dot{\beta}_1, \dot{\beta}_2, \dot{\beta}_3] \end{aligned} \quad (2.10)$$

The derivatives of X, Y , and θ can be calculated from U using Eq. (2.4). By substituting Eq. (2.4) into Eq. (2.2), I can calculate $[\dot{l}_1, \dot{l}_2, \dot{l}_3]$ from U as

$$\begin{aligned} \begin{bmatrix} \dot{l}_1 \\ \dot{l}_2 \\ \dot{l}_3 \end{bmatrix} &= - \begin{bmatrix} \cos \beta_1 & 0 & 0 \\ 0 & \cos \beta_2 & 0 \\ 0 & 0 & \cos \beta_3 \end{bmatrix}^{-1} \begin{bmatrix} \cos \beta_1 & \sin \beta_1 & l_1 \sin \beta_1 \\ \cos(\beta_2 + \alpha_2) & \sin(\beta_2 + \alpha_2) & l_2 \sin \beta_2 \\ \cos(\beta_3 + \alpha_3) & \sin(\beta_3 + \alpha_3) & l_3 \sin \beta_3 \end{bmatrix} * \\ &\quad \begin{bmatrix} -\sin \alpha_1 & \cos \alpha_1 & l_1 \\ -\sin \alpha_2 & \cos \alpha_2 & l_2 \\ -\sin \alpha_3 & \cos \alpha_3 & l_3 \end{bmatrix}^{-1} \begin{bmatrix} r \cos \beta_1 & 0 & 0 \\ 0 & r \cos \beta_2 & 0 \\ 0 & 0 & r \cos \beta_3 \end{bmatrix} \begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} \end{aligned} \quad (2.11)$$

In addition, β_i can be calculated by integrating $\dot{\beta}_i$. Therefore, the derivative of q can be expressed as a function of q and U .

$$\dot{q} = f(q, U) \quad (2.12)$$

I discretize the system at the k -th time step as

$$q(k+1) = q(k) + \dot{q}(k)dt = q(k) + f(q(k), U(k))dt \quad (2.13)$$

The state variables at the next time step can be represented by a function of the state variables and inputs of the current time step.

$$q(k+1) = F(q(k), U(k)) = q(k) + f(q(k), U(k))dt \quad (2.14)$$

At the point on the reference path at time step $(q_d(k), U_d(k))$, the system also satisfies

$$q_d(k+1) = F(q_d(k), U_d(k)) = q_d(k) + f(q_d(k), U_d(k))dt \quad (2.15)$$

For $q(k), U(k)$ close to $q_d(k), U_d(k)$, from Taylor's theorem, $F(q(k), U(k))$ can be calculated as

$$\begin{aligned} q(k+1) &= F(q(k), U(k)) \\ &= F(q_d(k), U_d(k)) + \left. \frac{\partial F}{\partial q} \right|_{q=q_d(k)} * (q(k) - q_d(k)) + \left. \frac{\partial F}{\partial U} \right|_{U=U_d(k)} * (U(k) - U_d(k)) \end{aligned} \quad (2.16)$$

where $\partial F / \partial q|_{q=q_d(k)}$ and $\partial F / \partial U|_{U=U_d(k)}$ are the partial derivatives of the function $F(q(k), U(k))$ at $(q_d(k), U_d(k))$ with respect to q and U , respectively. The linear and discrete kinematic equations for SWOM are thus obtained.

2.4.2 Trajectory tracking control using model predictive control

In Eq. (2.16), $F(q_d(k), U_d(k)) = q_d(k+1)$. I define the state error as $e^q(k) = q(k) - q_d(k)$ and the input error as $e^U(k) = U(k) - U_d(k)$. Then,

$$e^q(k+1) = \left. \frac{\partial F}{\partial q} \right|_{q=q_d(k)} e^q(k) + \left. \frac{\partial F}{\partial U} \right|_{U=U_d(k)} e^U(k) \quad (2.17)$$

$e^q(k+1)$ is the predicted state error of the $(k+1)$ -th time step at the k -th time step, so I denote it as $e^q(k+1|k)$. $\partial F / \partial q|_{q=q_d(k)}$ and $\partial F / \partial U|_{U=U_d(k)}$, which are the partial derivatives of the function $F(q(k), U(k))$ at $(q_d(k), U_d(k))$, can be rewritten as $A(k)$ and $B(k)$, respectively.

The state error for the next N time steps can be calculated using Eq. (2.17), giving

$$\begin{aligned} e^q(k+1|k) &= A(k)e^q(k) + B(k)e^U(k) \\ e^q(k+2|k) &= A(k+1|k)e^q(k+1|k) + B(k+1|k)e^U(k+1) \\ &= A(k+1|k)A(k)e^q(k) + A(k+1|k)B(k)e^U(k) + B(k+1|k)e^U(k+1) \\ &\vdots \\ e^q(k+N|k) &= A(k+N-1|k)A(k+N-2|k) \dots A(k)e^q(k) \\ &\quad + (k+N-1|k)A(k+N-2|k) \dots A(k+1|k)B(k)e^U(k) \\ &\quad + (k+N-1|k)A(k+N-2|k) \dots A(k+2|k)B(k+1|k)e^U(k+1) \end{aligned}$$

$$\begin{aligned}
& + \dots \\
& + B(k+N-1|k)e^U(k+N-1)
\end{aligned} \tag{2.18}$$

where $e^q(k+N|k)$ is the predicted state error of the $(k+N)$ -th time step at the k -th time step, $e^U(k+N-1)$ is the input error at the $(k+N-1)$ -th time step, and $A(k+N-1|k) = \partial F / \partial q|_{q=q_d(k+N-1)}$ and $B(k+N-1|k) = \partial F / \partial U|_{U=U_d(k+N-1)}$ are respectively the predicted partial derivatives of the function $F(q(k), U(k))$ with respect to q and U at the k -th time step at $(q_d(k+N-1), U_d(k+N-1))$.

At time step k , the predicted state and input error for the next N steps can be written as

$$\bar{X}(k+1) = \begin{bmatrix} e^q(k+1|k) \\ e^q(k+2|k) \\ e^q(k+3|k) \\ \vdots \\ e^q(k+N|k) \end{bmatrix}, \bar{U}(k) = \begin{bmatrix} e^U(k) \\ e^U(k+1) \\ e^U(k+2) \\ \vdots \\ e^U(k+N-1) \end{bmatrix} \tag{2.19}$$

Equation (2.17) becomes

$$\bar{X}(k+1) = \bar{A}e^q(k) + \bar{B}\bar{U}(k) \tag{2.20}$$

where

$$\bar{A} = \begin{bmatrix} A(k) \\ A(k+1|k)A(k) \\ \vdots \\ A(k+N-1|k) \dots A(k) \end{bmatrix} \tag{2.21}$$

$$\bar{B} =$$

$$\begin{pmatrix} & B(k) & \dots & \dots & 0 \\ & A(k+1|k)B(k) & B(k+1|k) & \dots & 0 \\ & \vdots & \vdots & \ddots & \vdots \\ A(k+N-1|k) \dots A(k+1|k)B(k) & A(k+N-1|k) \dots A(k+2|k)B(k+1|k) & \dots & B(k+N-1|k) \end{pmatrix} \tag{2.22}$$

Then, the cost function is defined as

$$\bar{J} = \bar{X}(k+1)^T \bar{Q} \bar{X}(k+1) + \bar{U}(k)^T \bar{R} \bar{U}(k) \tag{2.23}$$

where $\bar{Q}$ and $\bar{R}$ are weight coefficient matrices.

Substituting Eq. (2.20) into Eq. (2.23) yields

$$\begin{aligned}
 \bar{J} &= \left(\bar{A}e^q(k) + \bar{B}\bar{U}(k) \right)^T \bar{Q} \left(\bar{A}e^q(k) + \bar{B}\bar{U}(k) \right) + \bar{U}(k)^T \bar{R}\bar{U}(k) \\
 &= \bar{U}(k)^T (\bar{B}^T \bar{Q} \bar{B} + \bar{R}) \bar{U}(k) + e^q(k)^T \bar{A}^T \bar{Q} \bar{B} \bar{U}(k) + \bar{U}(k)^T \bar{B}^T \bar{Q} \bar{A} e^q(k) + e^q(k)^T \bar{A}^T \bar{Q} \bar{A} e^q(k) \\
 &= \frac{1}{2} \bar{U}(k)^T H(k) \bar{U}(k) + 2f(k)^T \bar{U}(k) + d(k)
 \end{aligned} \tag{2.24}$$

where $H(k) = 2(\bar{B}^T \bar{Q} \bar{B} + \bar{R})$, $f(k)^T = e^q(k)^T \bar{A}^T \bar{Q} \bar{B}$, and $d(k) = e^q(k)^T \bar{A}^T \bar{Q} \bar{A} e^q(k)$.

The minimization of the cost function becomes a quadratic programming problem with constraints on the inputs and outputs:

$$\begin{aligned}
 \min_{\bar{U}(k)} \bar{J} &= \frac{1}{2} \bar{U}(k)^T H(k) \bar{U}(k) + 2f(k)^T \bar{U}(k) + d(k) \\
 \text{s.t.} \\
 l_{\min} &\leq l_i \leq l_{\max}, \\
 -\frac{\pi}{2} &< \beta_i < \frac{\pi}{2}, \\
 \dot{\phi}_{\min} &\leq \dot{\phi}_i \leq \dot{\phi}_{\max}, \\
 \dot{\beta}_{\min} &\leq \dot{\beta}_i \leq \dot{\beta}_{\max}, \quad i \in \{1, 2, 3\}
 \end{aligned} \tag{2.25}$$

At the k -th time step, the quadratic programming problem is solved and the optimal input error $\bar{U}(k)$ is derived. I only use the optimal input for the present time step and then advance to the next step.

A simulation of SWOM tracking the reference path shown in Fig. 2.2 was conducted to verify the effectiveness of the LMPC-based trajectory tracking method. Both the acceleration and deceleration of the robot were considered. Initially, the robot started with $\dot{X}_d = 0.00$ m/s, $\dot{Y}_d = 0.00$ m/s, and $\dot{\theta}_d = 0.00$ rad/s, and accelerated at $\ddot{Y}_d = 0.01$ m/s² and $\ddot{\theta}_d = 0.02$ rad/s². After 5.0 s, the robot moved at a constant speed of $\dot{X}_d = 0.00$ m/s, $\dot{Y}_d = 0.05$ m/s, and $\dot{\theta}_d = 0.1$ rad/s. When the robot approached the corner, $\ddot{Y}_d$ became -0.01 m/s² and $\ddot{\theta}_d$ became -0.02 rad/s². After 4.0 s, $\ddot{\theta}_d$ became 0 and $\ddot{Y}_d$ remained at -0.01 m/s². The robot finally stopped at the corner after one more second and rotated at $\dot{\theta}_d = 0.02$ rad/s. After that, the robot accelerated again at $\ddot{X}_d = 0.01$ m/s² and $\ddot{\theta}_d = 0.02$ rad/s² one second later. The accelerated rotation lasted 4.0 s and the accelerated movement in the X direction lasted 5.0 s. Then, the robot started moving at a constant speed of $\dot{X}_d = 0.05$ m/s, $\dot{Y}_d = 0.00$ m/s, and $\dot{\theta}_d = 0.1$ rad/s until the end of the simulation.

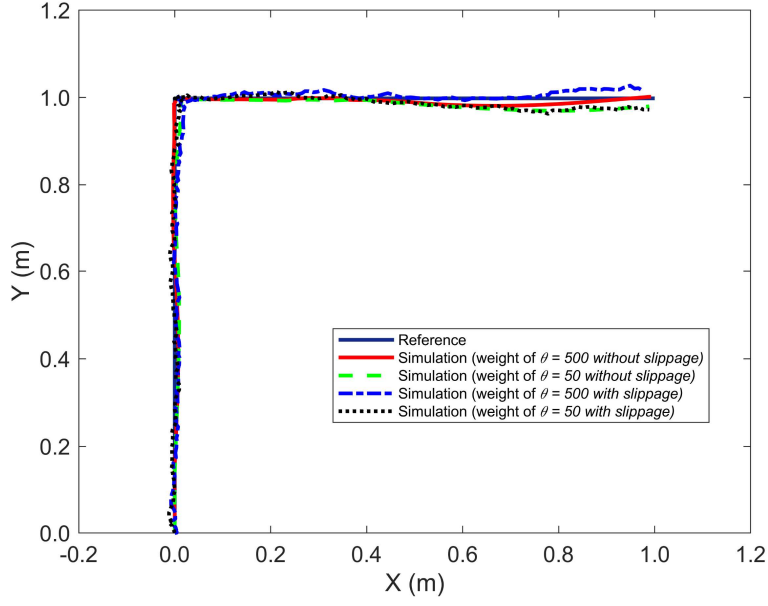
The reference position for the sliders was set to $l_{id} = 0.20$ m and the limits on the position of the

sliders were $l_{\min} = 0.12$ m and $l_{\max} = 0.30$ m. The constraints on the inputs were $\dot{\phi}_{\min} = -1.00$ rad/s, $\dot{\phi}_{\max} = 1.00$ rad/s, $\dot{\beta}_{\min} = -2.50$ rad/s, and $\dot{\beta}_{\max} = 2.50$ rad/s. In the optimization problem of MPC, the state weight matrix is designed as $\bar{Q} = \text{diag}(1000, 1000, 500, 100, 100, 100, 1, 1, 1)$, and the control weight matrix is $\bar{R} = I_{3 \times 3}$.

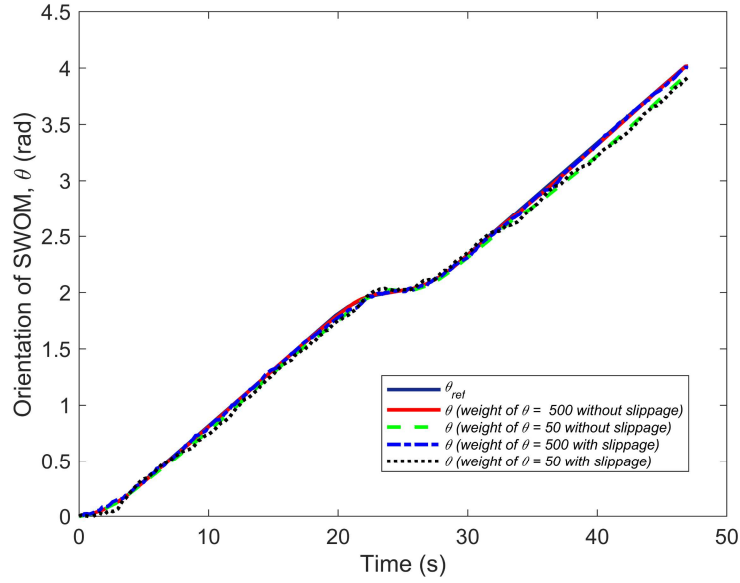
In the simulation, I simulate the effect of wheel slippage by adding random disturbances with an amplitude of 0.005 m to the position coordinates of the three wheels in the global coordinate system. Additionally, I use two different weights (500 and 50) for the errors related to θ to observe their impact on the simulation results. The tracking error of each simulation is recorded in Table. 3.1. Figure 2.3 presents the results of the four simulations, it can be observed that SWOM is able to track the reference path effectively in the simulation. In Fig. 2.3(b), it is evident that, after 40 seconds, when the weights of the errors related to θ are the same, the difference between considering or not considering slipping is minimal. However, when the weight of the error related to θ is larger, SWOM is able to track the reference value of θ more effectively.

Table 2.1 Tracking error of simulations

Weight of $\theta = 500$, without slippage	Mean squared error	0.0004
	Max position error	0.0385
	Max angle error	0.0201
Weight of $\theta = 50$, without slippage	Mean squared error	0.0036
	Max position error	0.0361
	Max angle error	0.1156
Weight of $\theta = 500$, with slippage	Mean squared error	0.0009
	Max position error	0.0572
	Max angle error	0.0502
Weight of $\theta = 50$, with slippage	Mean squared error	0.0050
	Max position error	0.0417
	Max angle error	0.1421



(a)



(b)

Fig. 2.3 (a) Simulated and reference paths and (b) simulated and reference orientations for SWOM.

Figure 2.4 shows the simulated position of the sliders as SWOM moved along the single-corner reference path. The tracking error is small and the simulation results remain in a small range from the reference position of the sliders. Compared to Fig. 2.2, there are fewer peaks in reference value of l_i because when I consider the acceleration and deceleration of the robot, there are no rapid changes in the velocity of the robot. The peaks only occur when the direction of the velocity at the point on the robot that is in contact with the slider is parallel to the direction of the sliding rail (i.e., when β_i needs to be $\pi/2$ or $-\pi/2$). Compared to the case with wheel slippage, when slippage is not considered, the position

of the sliders changes more smoothly because there are no disturbances in the position of the wheels.

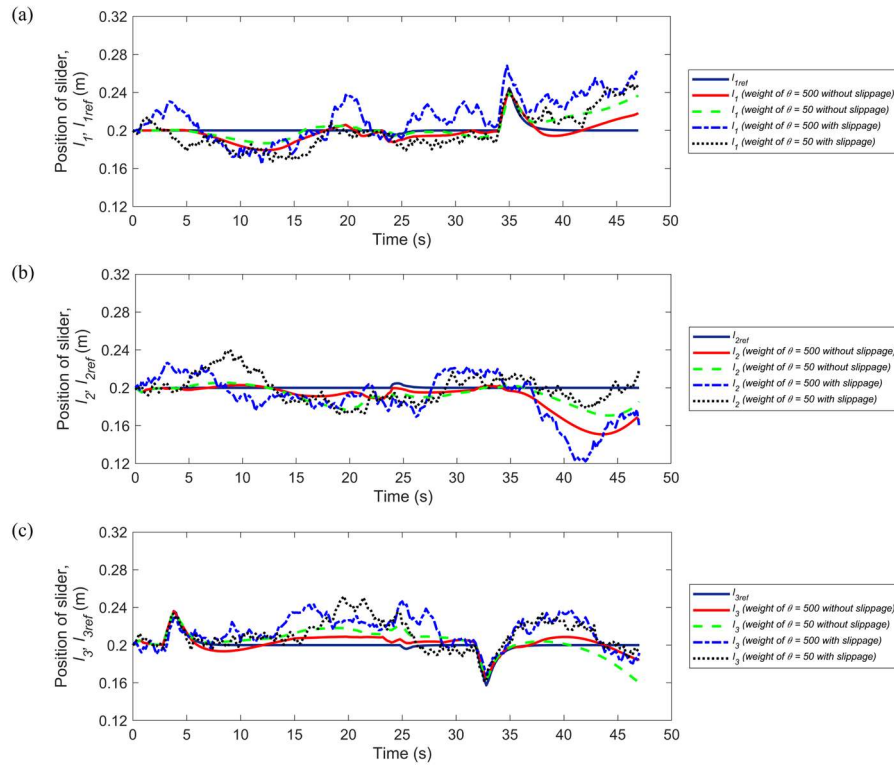


Fig. 2.4 Simulated and reference positions of sliders on SWOM.

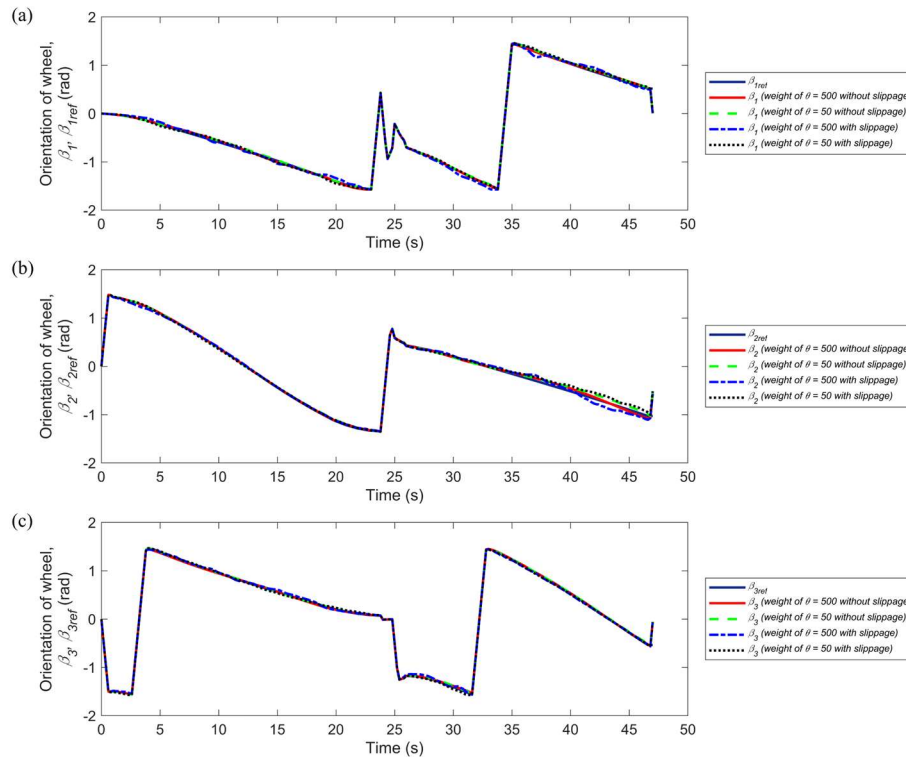


Fig. 2.5 Simulated and reference orientations of wheels on SWOM.

Figure 2.5 shows the orientation of the robot's wheels during the simulation compared to the reference values. The results show that the controller works well and that the wheels' orientations are almost the same as the reference values. It can also be seen that whether or not wheel slippage is considered, as well as the weight of the error related to θ , does not affect the tracking of the reference value for the wheel orientation.

2.5 Experiment

The previous section demonstrated that the proposed controller works well in a simulation. In this section, I conduct trajectory tracking experiments on a prototype to verify the effectiveness of the proposed controller.

2.5.1 Prototype of SWOM

The prototype, whose basic structure is similar to that in Fig. 2.1, is shown in Fig. 2.6. A robot computer and a monitor were placed on the top of the base block. Three sliding joints were connected to the base block. The angles between the joints were 120° . Each sliding joint consisted of two linear guides. Three driving units were attached to the three sliding joints, respectively. The units could move freely in the direction of the linear guides. A driving unit mainly consisted of two motors and a wheel. The upper motor was responsible for wheel steering and the lower motor was responsible for wheel rotation. In addition, each wheel was equipped with two ball rollers as safety wheels to prevent the robot from tipping over. These rollers did not affect the movement of SWOM because they rarely touched the ground.

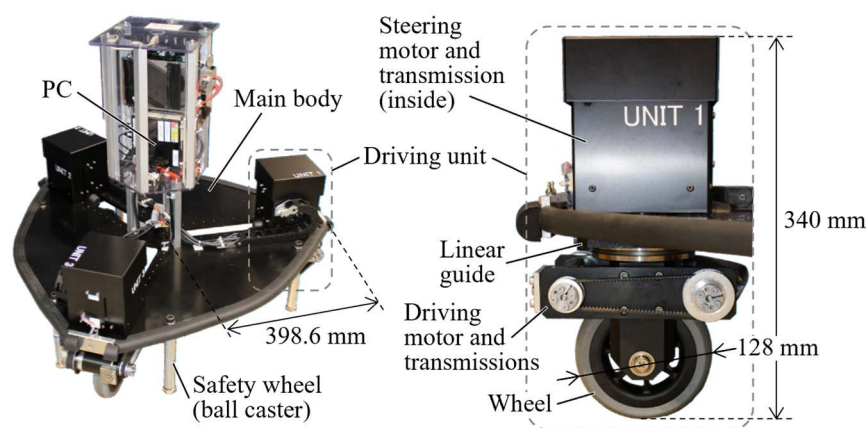


Fig. 2.6 Prototype of SWOM [36].

Figure 2.7 shows the control system of SWOM prototype. To measure the state variables for the robot to be used as inputs to the controller, a motion capture system (Bonita cameras, VICON) was used to measure the positions of the base block and the driving units and the displacement of the linear guides was calculated from the measured positions. The angle and speed of the wheels were measured by rotary encoders mounted on the motors. These measurements were sent to the robot computer, which then sent

appropriate input commands to the motors via motor drivers. Details of SWOM parameters are shown in Table 2.2.

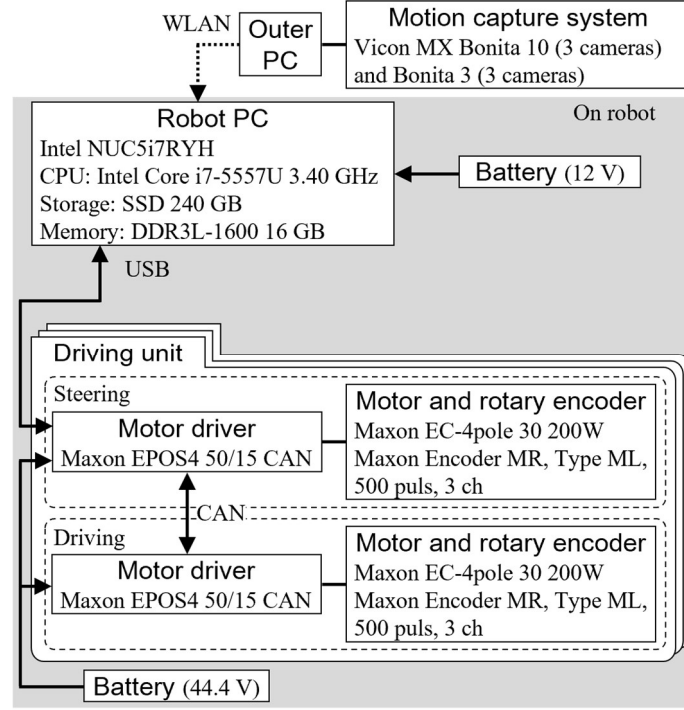


Fig. 2.7 Control system of SWOM prototype [36].

Table 2.2 Specifications of SWOM prototype [36].

Whole system	Weight	33kg
Driving unit (Driving / Steering)	Maximum speed	9.32 / 10.7 rad/s
	Maximum torque	38.0 / 41.7 N·m
	Resolution of encoder	1.35×10^{-5} / 1.54×10^{-5} rad
Linear guide	Movable range	$0.122 \text{ m} \leq l_i \leq 0.314 \text{ m}$
Motion capture	Operation period	1000 Hz

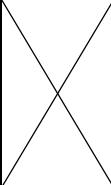
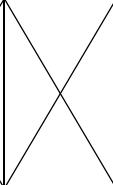
2.5.2 Experiments results

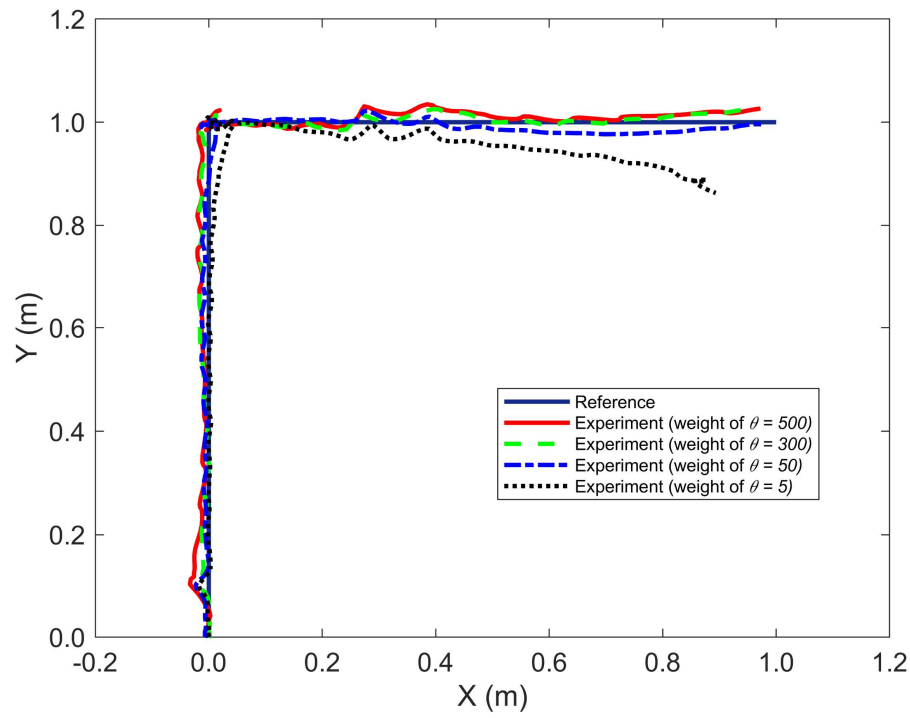
The prototype was tested on two different reference paths considering different weights of error of θ . The first path is a single-corner path, similar to that used in the simulation. Such a non-smooth path can only be traced by an omnidirectional mobile robot. The acceleration and deceleration of the robot were the same as those for the reference path in the simulation. The reference position of the sliders l_{ref} was set to 0.200 m and the limitation on the position of the sliders was $0.122 \text{ m} \leq l_i \leq 0.314 \text{ m}$. The constraints on the inputs were $\dot{\phi}_{\min} = -1.00 \text{ rad/s}$, $\dot{\phi}_{\max} = 1.00 \text{ rad/s}$, $\dot{\beta}_{\min} = -1.50 \text{ rad/s}$, and

$\dot{\beta}_{\max} = 1.50$ rad/s. The initial conditions were $X = 0.00$ m, $Y = 0.00$ m, $\theta = 0.00$ rad, and $\beta_1 = \beta_2 = \beta_3 = 0.00$ rad. The initial value of l_i was set randomly so that the slider was located near the middle of the slide rail.

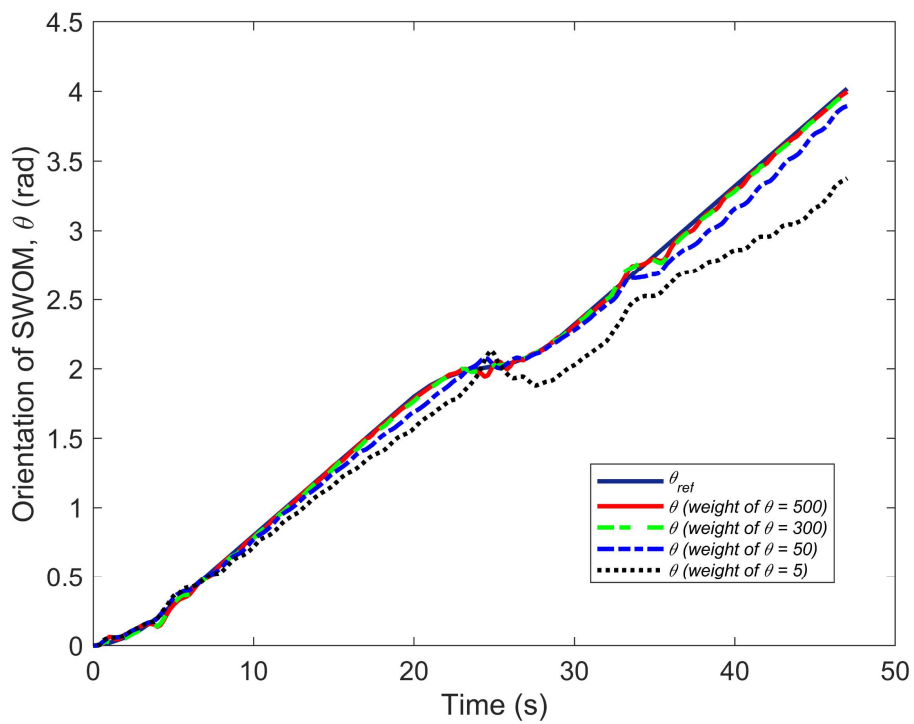
Multiple experiments were conducted for different weights of the error of θ , with the tracking errors recorded in Table 2.3. When the weights of the error of θ are not too small, SWOM performs well in tracking the reference trajectory. Figure 2.8 shows the robot's path and the reference path during trial No. 2, as well as the robot's orientation compared to the reference orientation. From the figure, it can be observed that when the weights of the error of θ are 500, 300, and 50, the tracking performance of the reference position is quite good. Moreover, when the weights of the error of θ are 500 and 300, the tracking of the robot's reference orientation is better compared to when the weights are 50 and 5. When the weight of the error of θ is 50 or 5, the orientation angle error may persist until the end. From the above, in this case, increasing the weights regarding the error of θ allows SWOM to better track the reference value of θ . Furthermore, when the weight of the error of θ becomes sufficiently large, further increasing the weight does not improve the tracking performance.

Table 2.3 Tracking error of each experiment (single-corner path).

		No. 1	No. 2	No. 3	No. 4	No. 5
Weight of $\theta = 500$	Mean squared error	0.0016	0.0015	0.0014	0.0019	0.0018
	Max position error	0.0565	0.0578	0.0502	0.0624	0.0663
	Max angle error	0.1020	0.0874	0.1031	0.1060	0.1132
Weight of $\theta = 300$	Mean squared error	0.0015	0.0013	0.0022	0.0015	0.0018
	Max position error	0.0518	0.0467	0.0687	0.0589	0.0627
	Max angle error	0.0880	0.0968	0.1084	0.0982	0.0955
Weight of $\theta = 50$	Mean squared error	0.0057	0.0087	0.0082	0.0112	0.0077
	Max position error	0.0395	0.0455	0.0437	0.0522	0.0398
	Max angle error	0.1336	0.1917	0.1633	0.1955	0.1778
Weight of $\theta = 5$	Mean squared error	0.0610	0.0911	0.1152		
	Max position error	0.1365	0.1738	0.1975		
	Max angle error	0.6021	0.6740	0.7273		



(a)



(b)

Fig. 2.8 (a) Real and reference paths and (b) real and reference orientations of SWOM in single-corner path experiment.

Figure 2.9 shows the displacement of the sliders and the reference displacement during movement. The displacement of the sliders in the experiment did not match the reference value as closely as in the simulation, but it was consistently within the allowable range and remained close to the reference value. Figure 2.10 shows the orientation of the robot's wheels during the experiment compared to the reference values. As can be seen in Fig. 2.10, the experimental values oscillate around the reference values. This may be due to the controller's inability to precisely control the actual value of β . First, in the prototype, the wheel orientation can be measured using a rotary encoder; however, the zero point was set manually and might be affected by measurement errors or assembly errors. As a result, the measured value of β contains errors compared to the actual value. Secondly, when solving the optimization problem in the controller, I set the time step to 0.2 seconds, expecting the motor to rotate for 0.2 seconds at the specified speed each time it receives a command. However, in the experiment, the time step typically ranges from 0.187 seconds to 0.203 seconds, meaning that I cannot transmit commands with perfect accuracy every 0.2 seconds and have the motor run exactly for 0.2 seconds. Consequently, the actual value of β cannot reach the intended value accurately. These two factors lead to the controller's inability to precisely control the actual value of β , causing β to oscillate near the reference value to ensure that the other state variables of SWOM can track their reference values.

Comparing the results in Table 2.3 and Table 2.1, under no-slip conditions, when the weight of the error related to θ is the same, the maximum position error and maximum angle error in the simulation are both smaller than those in the experiment. When slippage is considered, with the same weight for the error related to θ , the maximum position and angle errors in the simulation are sometimes comparable to the experimental results. However, for the mean squared error, regardless of whether slippage is considered or not, the simulation results are better than the experimental results when the weight of the error related to θ is the same. This indicates that the controller's performance in the simulation is better than in the actual experiment. The differences between the experiment and the simulation may be attributed to several factors, including the inability of the controller to send commands at fixed time steps as in the simulation, assembly errors present in the robot prototype, and discrepancies between the linearized motion model used in the simulation and the actual motion model. Since I derived the linear kinematic equations of the robot using a first-order Taylor expansion at reference points on the reference path, they become imprecise when the robot's state deviates significantly from the reference value, making the controller less effective.

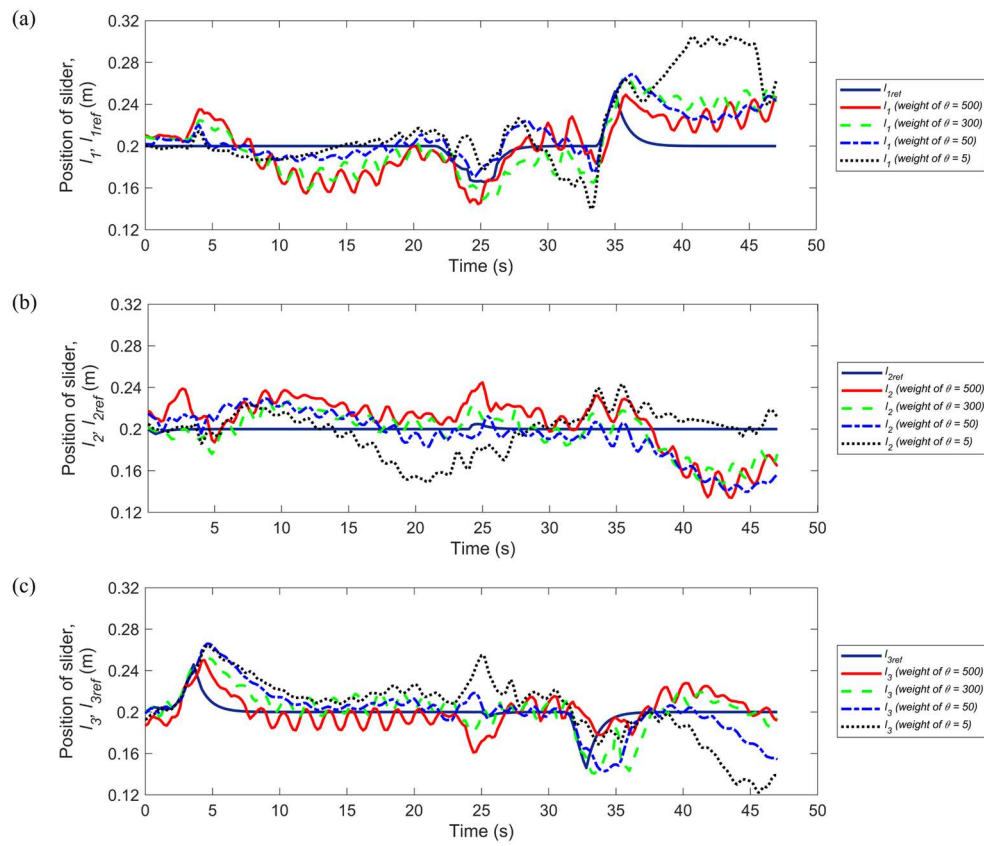


Fig. 2.9 Real and reference positions of sliders on SWOM in single-corner path experiment.

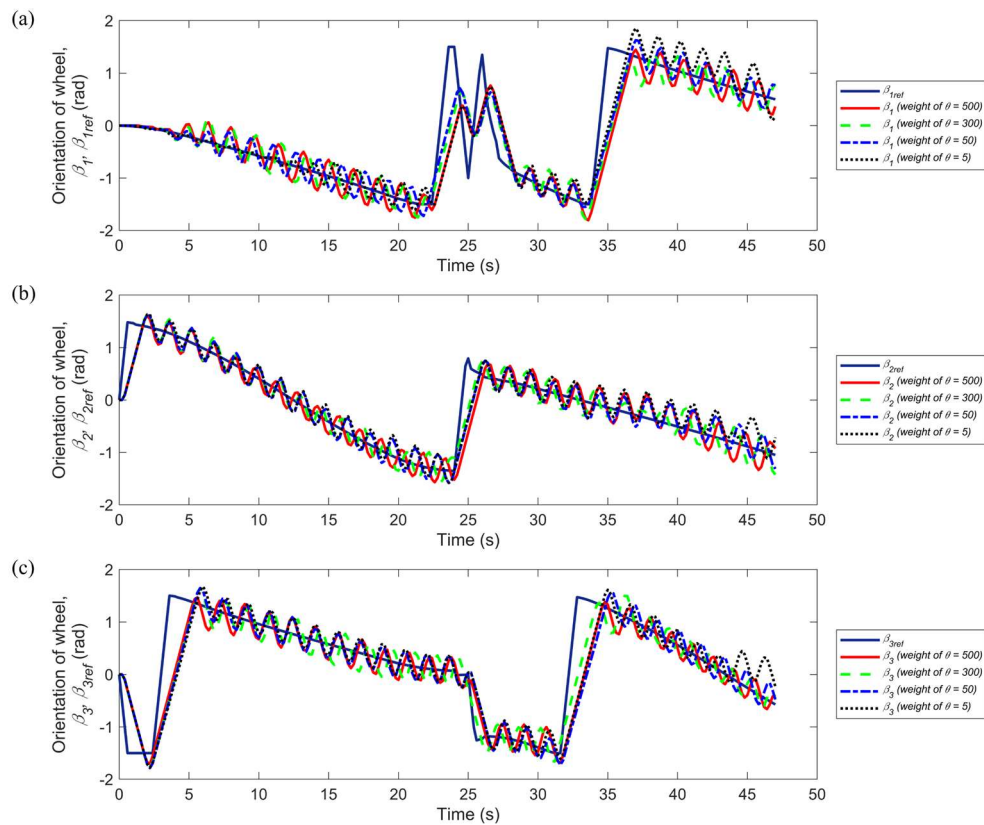


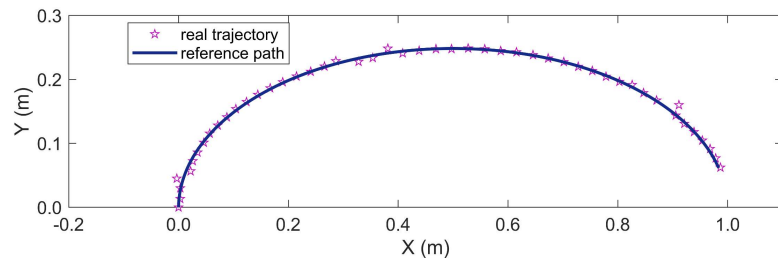
Fig. 2.10 Real and reference orientations of wheels on SWOM in single-corner path experiment.

The second reference path was smooth and circular. The robot needed to move along an arc with a radius of 0.5 m for 48 s at a speed of 0.03 m/s while rotating at 0.05 rad/s. The constraints on the inputs and outputs were the same as those for the single-corner path. The initial conditions were $X = 0.00$ m, $Y = 0.00$ m, $\theta = 0.00$ rad, and $\beta_1 = \beta_2 = \beta_3 = 0.00$ rad. The initial value of l_i was set randomly near the middle of the slide rail.

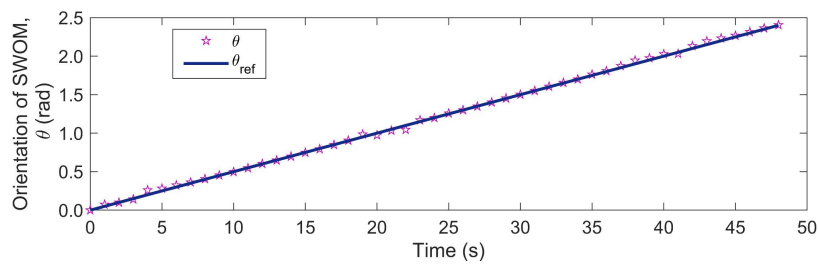
I also conducted three experiments and recorded the tracking errors from each experiment in Table 2.4. Comparing the data in Table 2.3, it can be observed that under the same parameter conditions, SWOM performs better in tracking the circular path than the single-corner path.

Table 2.4 Tracking error of each experiment (circular path).

		No. 1	No. 2	No. 3
Weight of $\theta = 50$	Mean squared error	0.0012	0.0005	0.0007
	Max position error	0.0236	0.0262	0.0212
	Max angle error	0.1138	0.0662	0.0926



(a)



(b)

Fig. 2.11 (a) Real and reference paths and (b) real and reference orientations for SWOM in circular path experiment.

The following figures show the results in trial No. 2. Figure 2.11 shows the reference and actual paths of the robot as it tracked a circular path, along with the reference and actual orientations. Figure 2.12 shows the displacement of the sliders and the reference values. The results show that the robot

worked properly in this situation. In Fig. 2.12, the peaks in the reference paths of the sliders are smoother than those observed for the single-corner path. This is because the robot moved along a circular arc while rotating. As a result, the robot's direction of motion changed more rapidly than it would moving in a straight line, preventing it from lingering too long in a direction parallel to the sliding rail.

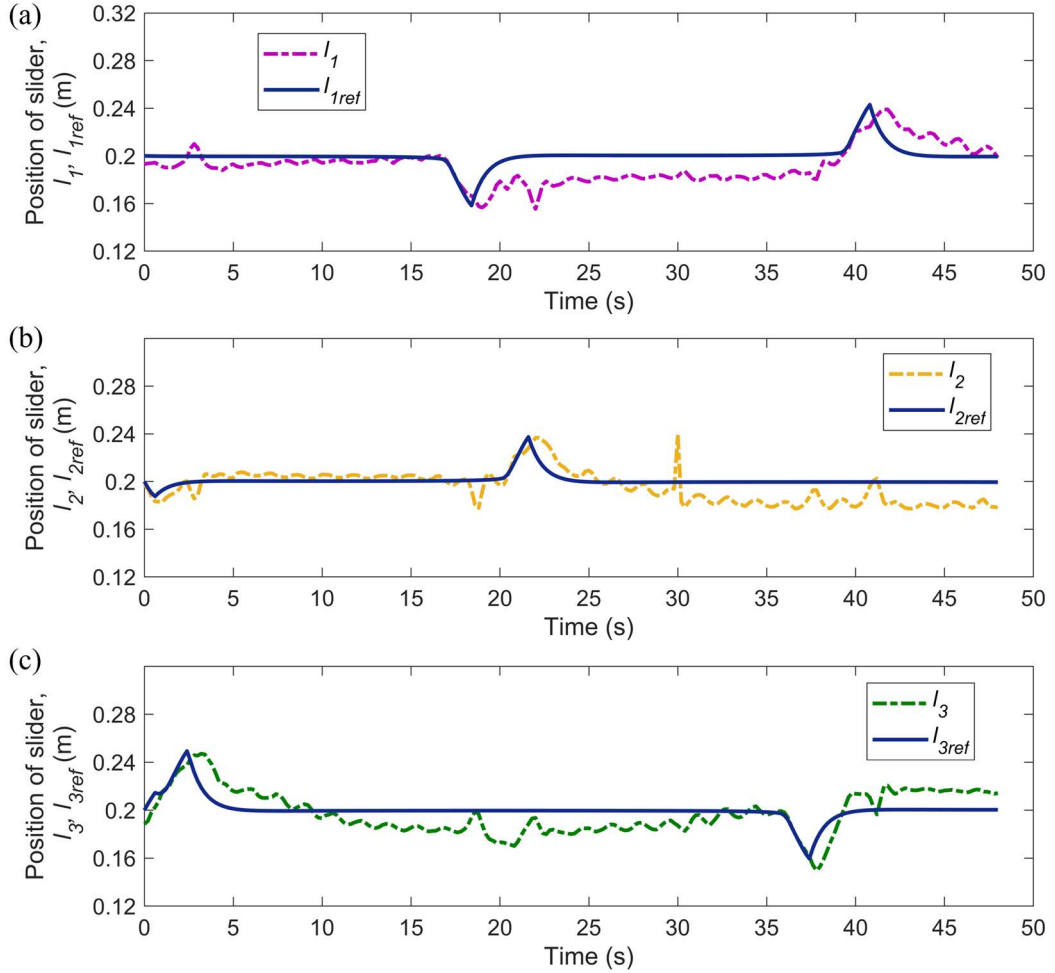


Fig. 2.12 Real and reference positions of sliders on SWOM in circular path experiment.

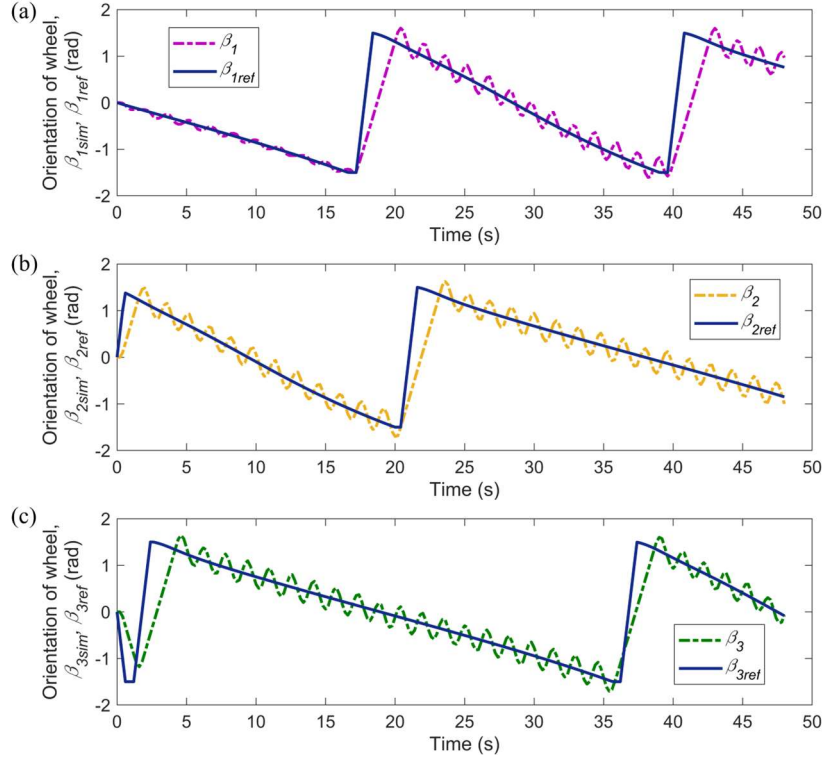


Fig. 2.13 Real and reference orientations of wheels on SWOM in circular path experiment.

Figure 2.13 shows the orientation of the robot's wheels during the experiment compared to the reference values. Similar to the single-corner path experiment, the experimental values oscillate around the reference values. This oscillation causes the wheels to rotate back and forth continuously, which can increase the likelihood of slippage. As explained in the discussion of the results from the single-corner path, if the position of β_i could be measured and set more accurately, this oscillation could be reduced.

The experiments using smooth and unsmooth reference paths confirm that the proposed controller works well, the tracking error is small, and the displacements of the sliders are controlled within the set limit.

2.6 Conclusion

In this chapter, the LMPC method is applied to achieve trajectory tracking for SWOM. Building on the previous research, the linearized kinematic model of SWOM was obtained by performing a first-order Taylor expansion at reference points on the reference trajectory. I used only the reference trajectory to generate the linear kinematic model of SWOM and then used LMPC to compute the robot's inputs during subsequent control. To perform a Taylor expansion on the reference trajectory, the trajectory must include not only the displacement and orientation of the robot but also the displacement of the sliders, the orientation of the wheels, the wheels' rotation, and steering velocities. I proposed a simple and effective method for generating such a reference trajectory. By observing the generated reference trajectory, I found that when the target speed changes suddenly, the reference value of the sliders'

displacement exhibits a peak. Similarly, when the target speed at the point on the robot that is in contact with an arbitrary slider is parallel to the corresponding sliding rail, the slider's displacement on that rail shows a peak. The peaks caused by sudden changes in speed can be mitigated by considering acceleration and deceleration. The peaks caused by the speed being parallel to the sliding rail can be reduced by increasing the rate of change in the robot's speed direction, such as by accelerating the robot's rotation or enabling it to move along a curve.

I conducted simulations considering wheel slippage and different weights for SOWM's orientation error in LMPC to validate the effectiveness of the proposed method. The results indicate that SWOM's displacement, orientation, and wheel orientation are all able to effectively track the reference values. The displacement of the slider on the sliding rail is also kept within the allowable range. Experiments were also conducted on a prototype for both a single-corner path and a circular path. Due to the prototype's inability to precisely control the actual value of β , the wheel orientation oscillated around the reference values and the slider position did not exactly match the reference values. Nevertheless, as long as the weight for SOWM's orientation error is not too small, the robot's displacement and orientation still tracked the reference values well, confirming the effectiveness of the proposed control method.

Chapter 3. Deep-reinforcement-learning-based controller design for SWOM

3.1 Introduction

In the previous chapter, I implemented trajectory tracking control for SWOM using LMPC and verified its effectiveness. However, the performance of LMPC is highly dependent on model accuracy and is subject to certain limitations in terms of real-time applicability. To overcome these issues, this chapter introduces a control approach based on DRL. Unlike model-based methods, DRL can directly learn control policies from interaction data, thereby avoiding complex modeling processes while offering greater adaptability and robustness. The trained controller has very low computational costs, which is very suitable for real time control of the robot.

When applying DRL for the trajectory tracking control of SWOM, if the robot's velocity in the global coordinate system is used as the action, the robot's motion model can be simplified to a planar motion model of a point mass without any constraints, making the task appear straightforward. However, due to the presence of passive sliding joints, movement in a specific direction at a given velocity may result in unintended movements of the passive joints, potentially causing the robot to enter singular configurations. As a result, this seemingly simple action becomes a complex task that requires a dedicated controller. The overall trajectory tracking task is thus not simplified. Therefore, a controller trained using the robot's velocity in the global coordinate system as the action is impractical. Some studies have used motor velocity commands as the action in DRL tasks [86]. However, since SWOM is equipped with six motors and has 12 DOFs, the action space and observation space dimensions are large. In addition, SWOM has an inherently complex motion model (due to its unique structure) and is subject to certain constraints on its state variables. These factors make it challenging to train a DRL controller for SWOM that can achieve effective trajectory tracking.

Hierarchical control is often applied to the control of mobile robots. It addresses complex control problems by dividing the controller into several layers, each responsible for a specific level of control. Different layers can employ distinct controllers, allowing for the advantages of various control methods to be leveraged while mitigating their drawbacks. In mobile robot navigation in unknown environments [87], the upper-level controller typically serves as the decision-making layer. It generates a reference path based on the current state of the robot and the environment. The lower-level controller utilizes this reference path to output motor inputs or wheel speed commands, enabling the robot to track the reference path. Xinyi et al. [88] proposed a hierarchical structure for controlling a mobile robot when the system cannot be accurately modeled and the operating environment varies in real time. The upper-level controller is based on the soft actor-critic (SAC) algorithm and the lower-level controller employs an incremental proportional-integral-derivative approach. The upper-level controller determines the optimal

parameters for the lower-level controller based on the robot's state and the reference path, enabling the robot to effectively follow the reference path. Yan et al. [89] proposed a hierarchical control method for the trajectory tracking of mobile robots that accounts for system uncertainties. The upper-level controller utilizes kinematic equations to determine the velocities required for the robot to achieve trajectory tracking and the lower-level controller regulates the robot's velocity through robust adaptive three-dimensional torque control.

In this chapter, I propose a trajectory tracking controller for SWOM based on DRL integrated with hierarchical control. The trajectory tracking controller is divided into two layers. The upper-level controller computes the velocity required for SWOM body to follow the reference trajectory based on the current state of SWOM and the reference state. The required rotational speeds of the individual wheels can be calculated using SWOM's kinematic equations. The lower-level controller takes the velocity of SWOM body output by the upper-level controller as input. Based on the state of SWOM, it outputs the steering speeds of the wheels to regulate their displacements along the sliding rails within feasible ranges. I trained a DRL-based controller specifically for a single slidable wheel. Due to the symmetry of the three wheels, the same controller can be used to control each wheel. Consequently, the lower-level controller is composed of three identical controllers, which operate independently and take inputs from the upper-level controller. This reduces the dimensions of the action space and observation space of the agent trained in DRL, significantly decreasing the complexity of the task and training time. Moreover, the controller trained using the proposed method is generalizable and applicable to any robot that utilizes slidable wheels. This control method, which combines hierarchical control and DRL, also offers valuable insights for the application of DRL to other robots with complex structures.

This chapter is organized as follows: Section 2 describes the design of the proposed trajectory tracking controller, which integrates DRL with hierarchical control. The task formulation, reward design, as well as the action and state spaces of the reinforcement learning framework are also presented in this section. Section 3 presents numerical simulation results to validate the effectiveness of the trained controller in regulating wheel displacement, as well as to demonstrate the overall performance of the proposed controller in SWOM trajectory tracking control. Section 4 concludes this chapter.

3.2 DRL-based hierarchical control method for trajectory tracking

3.2.1 Problem description

When applying DRL to the trajectory tracking control of SWOM, directly controlling the entire robot leads to challenges. Specifically, the action space and observation space dimensions are six and nine, respectively, which results in prohibitively long training times or failure to achieve satisfactory results.

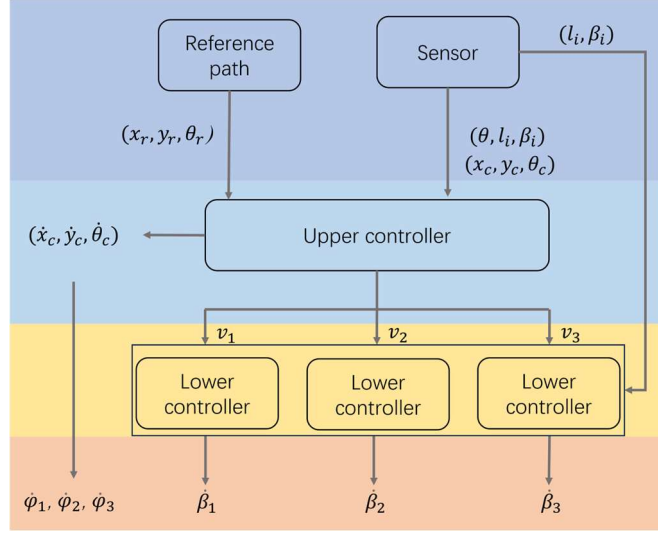


Fig. 3.1 Overview of SWOM control system.

To address these limitations, I propose a controller design based on DRL integrated with hierarchical control by taking advantage of features of SWOM's system. From Eq. (2.5), it can be seen that once SWOM body's translational and rotational velocities $(\dot{X}, \dot{Y}, \dot{\theta})$ are determined and the state variables are known, the required rotational speeds of the three wheels (ϕ_1, ϕ_2, ϕ_3) can be uniquely determined. Based on this fact, the control of SWOM is divided into two levels, as shown in Fig. 3.1. At the upper level, a controller computes the desired translational and rotational velocities of SWOM body based on its current state and the offset from the target position and orientation to achieve trajectory tracking. Using the target translational and rotational velocities of SWOM body, the required rotational speeds of the wheels are easily determined through SWOM's kinematic equations.

The lower-level controller controls the sliding motion of each wheel along the sliding rail. According to Eq. (2.7), when SWOM body's translational and rotational velocities $(\dot{X}, \dot{Y}, \dot{\theta})$ are determined, the displacements of the three wheels along their respective sliding rails (l_1, l_2, l_3) are independent of each other and solely determined by their orientations $(\beta_1, \beta_2, \beta_3)$. I divide the lower-level controller into three parts. I train a controller for a single wheel using the proximal policy optimization (PPO) algorithm to regulate its steering velocity $\dot{\beta}_i$, ensuring that the wheel stays as close as possible to the center of the sliding rail regardless of the target translational and rotational velocities of SWOM body. This single-wheel controller can be applied to all three wheels without additional tuning because all wheels have the same mechanism. The condition in Eq. (2.7) is generally challenging to manage with conventional control methods because it is a nonholonomic constraint in which the input $\dot{\beta}_i$ does not appear explicitly. However, the proposed DRL-based method obviates the necessity to solve the equation.

Through this two-level control framework, I can implement a DRL-based trajectory tracking controller for SWOM that generalizes to various reference paths. The details of the controller design, including the state space, action space, initial conditions, and reward design used during training, are described in subsequent sections.

3.2.2 Upper-level controller

As shown in Fig. 3.2, the actual position of the robot is denoted as (x_c, y_c, θ_c) and the reference path is denoted as (x_r, y_r, θ_r) . The positional and orientation errors of the robot are defined as (x_e, y_e, θ_e) , where $x_e = x_c - x_r$, $y_e = y_c - y_r$, and $\theta_e = \theta_c - \theta_r$.

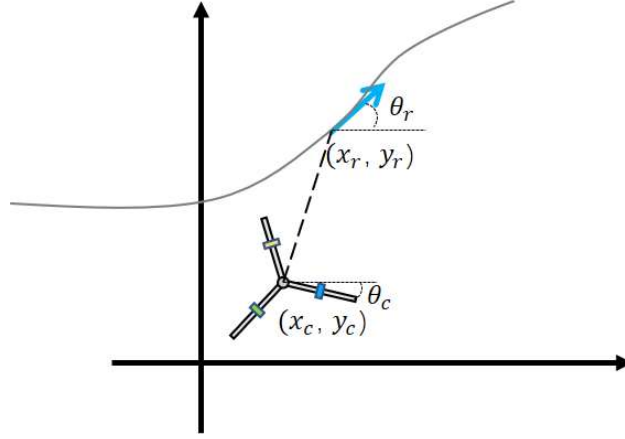


Fig. 3.2 Actual and reference positions of SWOM.

For an omnidirectional mobile robot such as SWOM, the velocities along the three DOFs can be independently controlled. I thus utilize a straightforward feedback method to design the velocity and angular velocity of SWOM for trajectory tracking.

The actual velocities of the robot ($\dot{x}_c$, $\dot{y}_c$, and $\dot{\theta}_c$) are designed as follows:

$$\begin{aligned}\dot{x}_c &= \dot{x}_r - K_x * x_e \\ \dot{y}_c &= \dot{y}_r - K_y * y_e \\ \dot{\theta}_c &= \dot{\theta}_r - K_\theta * \theta_e\end{aligned}\tag{3.1}$$

Here, $\dot{x}_r$, $\dot{y}_r$, and $\dot{\theta}_r$ represent the reference velocities and reference rotational velocity of the robot, respectively.

From Eq. (3.1), it can be obtained that

$$\begin{aligned}\dot{x}_e &= \dot{x}_c - \dot{x}_r = -K_x * x_e \\ \dot{y}_e &= \dot{y}_c - \dot{y}_r = -K_y * y_e \\ \dot{\theta}_e &= \dot{\theta}_c - \dot{\theta}_r = -K_\theta * \theta_e\end{aligned}\tag{3.2}$$

The Lyapunov function is designed as:

$$V = x_e^2 + y_e^2 + \theta_e^2 \geq 0 \quad (3.3)$$

Differentiating the Lyapunov function with respect to time yields:

$$\dot{V} = 2(x_e\dot{x}_e + y_e\dot{y}_e + \theta_e\dot{\theta}_e) = -2K_x x_e^2 - 2K_y y_e^2 - 2K_\theta \theta_e^2 \leq 0 \quad (3.4)$$

Therefore, the positional and orientation errors of the robot converge to zero.

3.2.3 Lower-level controller

After determining the target translational and rotational velocities for SWOM, the required rotational speeds for each wheel can be calculated based on their orientations using Eq. (2.5). To prevent the displacements of the wheels along the sliding rails from exceeding their limits, a controller must be trained to regulate the steering of the wheels, thereby controlling their displacements. From the kinematic equations governing the displacements of the slidable wheels, it can be observed that once the overall target translational and rotational velocities of SWOM are determined, the displacements of the three wheels can be decoupled. Therefore, during controller training using DRL, a task targeting a single wheel can be designed. The trained controller can then be applied independently to each of the three wheels. In this section, I first present the kinematic analysis of wheel displacement, followed by a detailed description of the task used to train the agent in the DRL framework and the design of the reward function. Finally, the training of the agent using the PPO algorithm is implemented with Gymnasium [90] and Stable Baselines3 [91].

3.2.3.1 Kinematic analysis of wheel displacement

As shown in Fig. 3.3, a local coordinate system is established on each sliding rail, denoted as OX_1Y_1 , OX_2Y_2 , and OX_3Y_3 , where OX_1 , OX_2 , and OX_3 are parallel to the respective sliding rail. Using SWOM's target translational and rotational velocities $(\dot{X}, \dot{Y}, \dot{\theta})$, the velocity magnitude and direction of the point on each sliding rail that overlaps with its corresponding wheel can be calculated in the respective local coordinate systems.

$$\begin{aligned} v_i &= \dot{x}_i^2 + \dot{y}_i^2 \\ \sigma_i &= \arctan\left(\frac{\dot{y}_i}{\dot{x}_i}\right) \\ \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} &= \begin{bmatrix} \cos(\theta + \alpha_i) & \sin(\theta + \alpha_i) \\ -\sin(\theta + \alpha_i) & \cos(\theta + \alpha_i) \end{bmatrix} \begin{bmatrix} \dot{X} \\ \dot{Y} \end{bmatrix} + \begin{pmatrix} 0 \\ l_i \dot{\theta} \end{pmatrix} \end{aligned} \quad (3.5)$$

where $\dot{x}_i$ and $\dot{y}_i$ ($i = 1, 2, 3$) represent the velocity magnitudes along the X_i - and Y_i -axis directions,

respectively, of the point on the sliding rail that overlaps with the wheel in their respective local coordinate systems.

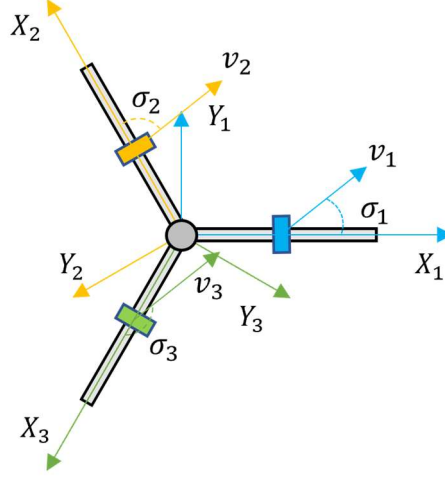


Fig. 3.3 Local coordinate system established on each of three sliding rails. The positive direction of σ_i is defined as counterclockwise. The scenario shown in the figure corresponds to the case where SWOM's rotational speed is zero, and $(\dot{x}_1, \dot{y}_1)$, $(\dot{x}_2, \dot{y}_2)$, and $(\dot{x}_3, \dot{y}_3)$ are parallel.

By substituting Eq. (3.5) into Eq. (2.7), the kinematic equations of displacement for each wheel in their respective local coordinate systems can be derived.

$$\dot{l}_i = -v_i \sin(\sigma_i) \tan(\beta_i) - v_i \cos(\sigma_i) \quad (3.6)$$

Define $\sigma_i > 0$, which means that the velocity of the point on the sliding rail that overlaps with the wheel points in the positive direction of the OY_i axis, it can be observed that as long as the wheel's orientation is parallel to the velocity direction of the point on the sliding rail overlapping with the wheel, the wheel's displacement along the rail will remain unchanged. When the wheel's orientation points to the left of the velocity direction of the overlapping point, the wheel will move leftward along the sliding rail, and vice versa. I express this property of the rate of change of wheel displacement using the following equation.

$$\begin{aligned} \dot{l}_i &> 0 \quad \text{when} \quad \beta_i + \frac{\pi}{2} < \sigma_i \\ \dot{l}_i &= 0 \quad \text{when} \quad \beta_i + \frac{\pi}{2} = \sigma_i \\ \dot{l}_i &< 0 \quad \text{when} \quad \beta_i + \frac{\pi}{2} > \sigma_i \end{aligned} \quad (3.7)$$

When $\sigma_i < 0$, the relationship between the rate of change of wheel displacement and the wheel orientation is expressed as follows:

$$\begin{aligned} \dot{l}_i &< 0 \quad \text{when } \beta_i - \frac{\pi}{2} < \sigma_i \\ \dot{l}_i &= 0 \quad \text{when } \beta_i - \frac{\pi}{2} = \sigma_i \\ \dot{l}_i &> 0 \quad \text{when } \beta_i - \frac{\pi}{2} > \sigma_i \end{aligned} \tag{3.8}$$

3.2.3.2 Task for reinforcement learning

Next, with the target translational and rotational velocities of SWOM body already determined, I use DRL to train a controller for regulating the displacement of the wheels. The task for DRL is illustrated here in OX_1Y_1 coordinate system. As shown in Fig. 3.4, L is the displacement of the wheel along the sliding rail. Unlike in the kinematic equations, here I define the zero point of L at the center of the sliding rail. β is the angle between the wheel axle and the sliding rail. v and σ are the magnitude and direction, respectively, of the velocity of the point on the sliding rail that overlaps with the wheel in the local coordinate system. The state space and action space are then defined as: $[L, \beta, v, \sigma]$ and $[\dot{\beta}]$, respectively. Compared to directly applying end-to-end reinforcement learning to train SWOM trajectory tracking controller, the dimensions of both the state space and action space have been significantly reduced. The goal of the trained agent is to output an appropriate $\dot{\beta}$ such that L is maintained near zero and the orientation of the wheel aligns with the direction of v by observing the value of state in state space.

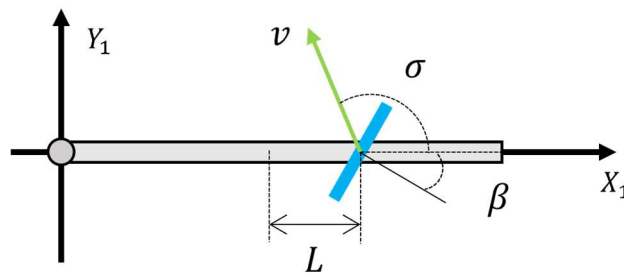


Fig. 3.4 Slidable wheel model in OX_1Y_1 coordinate system.

In the task, the direction of v is defined as to point above the sliding rail for simplicity, which implies that $\sigma > 0$. For the case where the direction of v points below the sliding rail, due to its symmetry with the case where the velocity points above the sliding rail, the controller trained when v points above the sliding rail can be applied. For example, if the state variables are $[L = 0.02, \beta = -1, v = 0.08, \sigma = -2]$, I can simply change the signs of certain values to obtain $[L = 0.02, \beta = 1, v =$

$0.08, \sigma = 2]$, input this into the controller trained under $\sigma > 0$, and then take the opposite of the output action as the actual control action. Therefore, during training, I only need to consider the scenario where the velocity points above the sliding rail.

Moreover, when the direction of the velocity of the point on the sliding rail that overlaps with the wheel is parallel to the rail, the wheel's displacement will inevitably increase or decrease. Therefore, during training, I constrain σ to the range $(\pi/9, 8\pi/9)$. To prevent SWOM from entering a singular configuration, I stipulate that the wheel's orientation must not be parallel to the sliding rail. As a result, the ranges of variation of four variables in state space are defined as

$$\begin{cases} -0.1 < L < 0.1, \\ -\frac{4\pi}{9} < \beta < \frac{4\pi}{9}, \\ 0 < v < 0.1, \\ \frac{\pi}{9} < \sigma < \frac{8\pi}{9} \end{cases} \quad (3.9)$$

To ensure that the trained controller can handle various conditions, I randomly assign values within predefined ranges to the four state variables as the initial conditions for each round of training. Their respective ranges are:

$$\begin{cases} -0.05 < L < 0.05, \\ -\frac{4\pi}{9} < \beta < \frac{4\pi}{9}, \\ 0 < v < 0.1, \\ \frac{\pi}{9} < \sigma < \frac{8\pi}{9} \end{cases} \quad (3.10)$$

The design of the reward function during training is critical for obtaining a controller that meets the desired performance. A detailed description of the reward function is provided in the next section.

3.2.3.3 Reward function

In reinforcement learning, the objective of training is to maximize the cumulative reward within an episode. Therefore, the design of the reward function has a significant impact on the performance of the trained agent. I not only provide the agent with rewards or penalties upon task success or failure, but also impose penalties at each time step based on the current state to guide the agent toward successfully completing the task.

If the wheel's displacement along the sliding rail exceeds the limit or the wheel's orientation goes beyond the allowable range, the task fails and a penalty of 1000 is applied. When the wheel's orientation aligns with the direction of v and the wheel is positioned at the center of the sliding rail, the task succeeds and a reward of 100 is given.

$$\begin{cases} \text{Reward} -= 1000, \text{ if } |L| > 0.1 \text{ or } |\beta| > \frac{\pi}{2} \\ \text{Reward} += 100, \text{ if } |L| < 0.01 \text{ and } \left| \beta + \frac{\pi}{2} - \theta \right| < 0.01 \end{cases} \quad (3.11)$$

For each time step, I also design the following reward function:

$$\begin{aligned} \text{Reward} &= R_1 + R_2 + R_3, \\ \begin{cases} R_1 = -L^2 \\ R_2 = -0.0025 \times \dot{L}^2, \\ R_3 = -0.0001 \times \dot{\beta}^2 \end{cases} \end{aligned} \quad (3.12)$$

The reward consists of three components. The first component R_1 is the negative square of the displacement between the wheel and the center of the sliding rail. This means that the farther the wheel is from the center, the greater the penalty it receives, thereby encouraging the agent to stabilize the wheel at the center of the rail.

The second component R_2 is the negative square of the wheel's moving speed along the rail. Based on the typical ranges of β , σ and v , as well as Eq. (3.6), the estimated range of $\dot{L}$ is approximately $[-2, 2]$, while the range of L is about $[-0.1, 0.1]$. Therefore, a normalization factor of 2.5×10^{-3} is applied. This component is designed to limit the magnitude of $\dot{L}$, otherwise, the wheel may overshoot the target during the stabilization. I discuss this overshooting behavior in detail in the next section.

The third component R_3 is the negative square of the wheel's steering velocity. It penalizes the agent based on the magnitude of the action it outputs, promoting more efficient control actions. Since the range of $\dot{\beta}$ is $[-10, 10]$, a normalization factor of 1.0×10^{-4} is introduced accordingly.

3.3. Numerical simulations

To validate the effect of R_2 in the reward function designed in the previous section, I first trained two agents with different parameter values and compared their performance on the same task through simulation. To verify the effectiveness and accuracy of the proposed trajectory tracking method, I conducted simulations to control SWOM to follow a target trajectory. To better approximate real-world scenarios, I further simulated a trajectory tracking task that considers wheel slippage. Specifically, during the simulation of SWOM trajectory tracking, I introduced noise to the X and Y coordinates of each wheel at every time step when updating SWOM configuration. To account for this slippage, I also added noise to the wheel displacement and orientation at each step when updating the state in the step function during training.

3.3.1 Numerical simulation of agents trained with different reward designs

To verify the role of R_2 in the reward at each step, I trained agents using two different formulations: $R_2 = -0.001 \times \dot{L}^2$ and $R_2 = -0.0025 \times \dot{L}^2$. I then simulated the behavior of these two agents under the same task conditions. The initial conditions for the task were set as follows: $L_{init} = 0.04$ m, $\beta_{init} = -1$, $\sigma_{init} = 2$, $v_{init} = 0.08$ m/s.

Figure 3.5(a) and (b) quantitatively depict the variations in wheel displacement and orientation during the simulation under the control of the two aforementioned agents. Figure 3.5(c) presents a schematic illustration of the wheel motion.

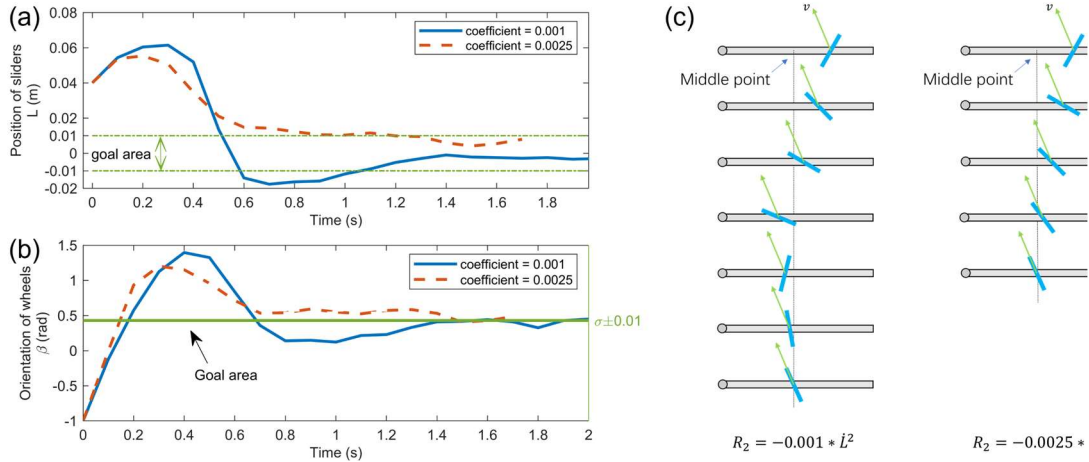


Fig. 3.5 Comparison of agents trained with reward functions using different parameters. (a) Variation in slider positions during the task. (b) Variation in wheel orientations during the task. (c) Schematic illustration of the behaviors exhibited by two different agents.

As shown in the figure, in this task, a relatively small coefficient of R_2 leads to an excessively large value of $\dot{L}$. Consequently, after the wheel initially reaches the center of the rail, it overshoots to the left, then reverses direction to the right, and finally stabilizes at the center. When the coefficient of R_2 is increased, the wheel begins to change direction toward the target velocity as it approaches the center of the rail. This behavior effectively reduces $\dot{L}$, prevents overshooting, and facilitates the alignment of the wheel with the target position, thereby improving task performance.

3.3.2 Trajectory tracking simulation of SWOM without considering wheel slippage

Numerical simulations were conducted to validate the effectiveness of the proposed method for SWOM control. The target trajectory was determined by specifying the reference velocity and rotational velocity. As shown in Fig. 3.6, the reference path is a figure-eight-shaped trajectory starting from the origin. Its equation can be expressed as:

$$X = -\cos(0.02t), Y = -\sin(0.04t), \theta = 0.3t \quad (3.13)$$

To verify whether the controller can effectively enable the robot to track the reference path despite initial position errors, the initial conditions of SWOM were set as $X_{init} = 0.1, Y_{init} = -0.1, \theta_{init} = 0, L_{1ini} = 0.02, L_{2init} = 0.01, L_{3init} = -0.01$, and $\beta_{1ini} = \beta_{2init} = \beta_{3ini} = 0$.

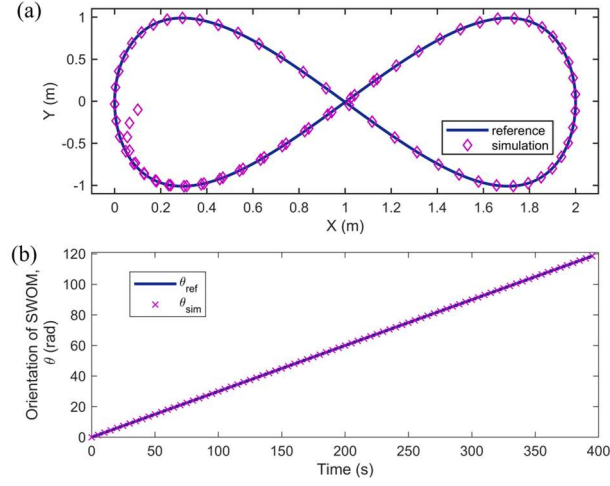


Fig. 3.6 (a) Real and reference path and (b) real and reference orientation of SWOM for figure-eight-shaped path.

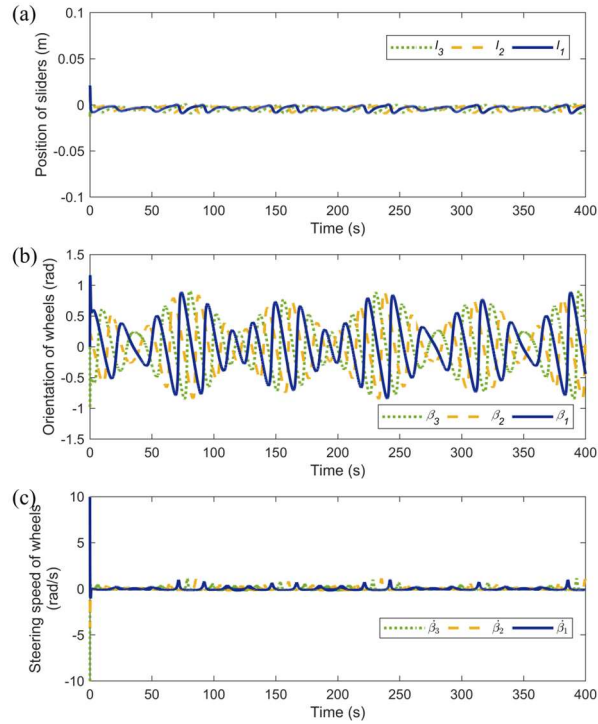


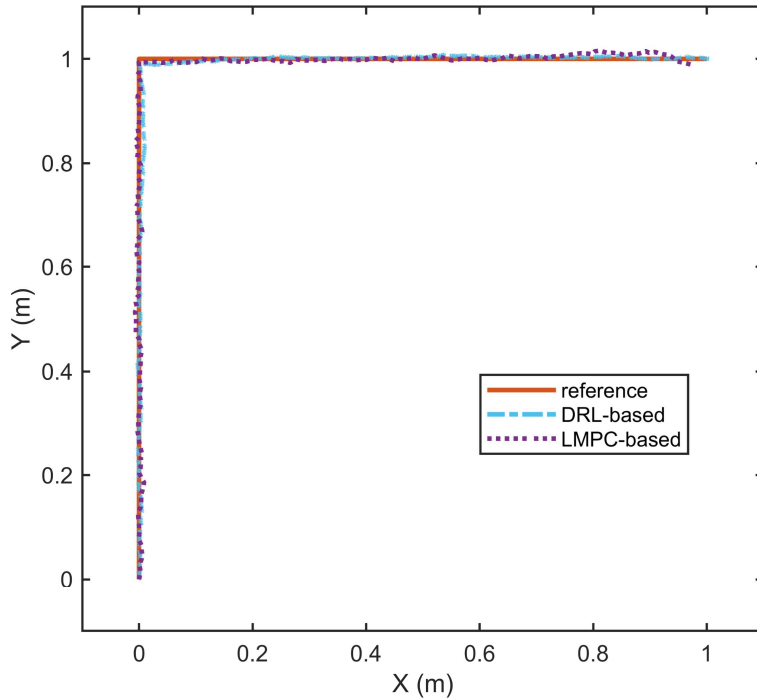
Fig. 3.7 (a) Position of three sliders, (b) orientation of three wheels, and (c) steering speed of three wheels of SWOM in simulation with figure-eight-shaped path.

As shown in Figs. 4.6 and 4.7, SWOM effectively tracks the reference trajectory in terms of both

position and orientation. The slider's displacement is quickly reduced to near zero and remains close to the midpoint of the sliding rail throughout the rest of the motion. The wheel orientation exhibits periodic changes during the entire movement process, as the reference trajectory involves SWOM rotating at a constant speed. The only instances where a sudden change in wheel orientation occurs are at the start of the motion (at 0 seconds). At the beginning, due to the initial conditions of the displacements of the three wheels being set as $L_{1init} = 0.02, L_{2ini} = 0.01, L_{3init} = -0.01$ and $\beta_{1init} = \beta_{2init} = \beta_{3init} = 0$. The velocity direction of the point on the sliding rail that overlaps with the wheel is not aligned with the wheel's direction. This situation causes changes in the displacement of the wheel along the sliding rail. In addition, it is also necessary for the wheel to be stabilized to the center of the sliding rail. The controller needs to promptly adjust the wheel's steering to minimize the displacement and ensure that the wheel's direction aligns with the velocity direction of the overlapping point on the sliding rail.

3.3.3 Trajectory tracking simulation of SWOM with wheel slippage consideration

In the previous trajectory tracking simulations, the system's kinematic model was assumed to be ideal, with no measurement errors or environmental noise. In this section, I simulate the effect of wheel slippage by adding random disturbances with an amplitude of 0.005 m to the position coordinates of the three wheels in the global coordinate system during trajectory tracking. To address this issue, I trained the agent in a noisy environment. Specifically, at each step of the state update, random disturbances with an amplitude of 0.0025 were added to the displacement and orientation of the wheels. I then simulated the performance of the controller in controlling SWOM for trajectory tracking. The results were compared with those obtained using a linear model predictive control (LMPC) approach in last chapter.



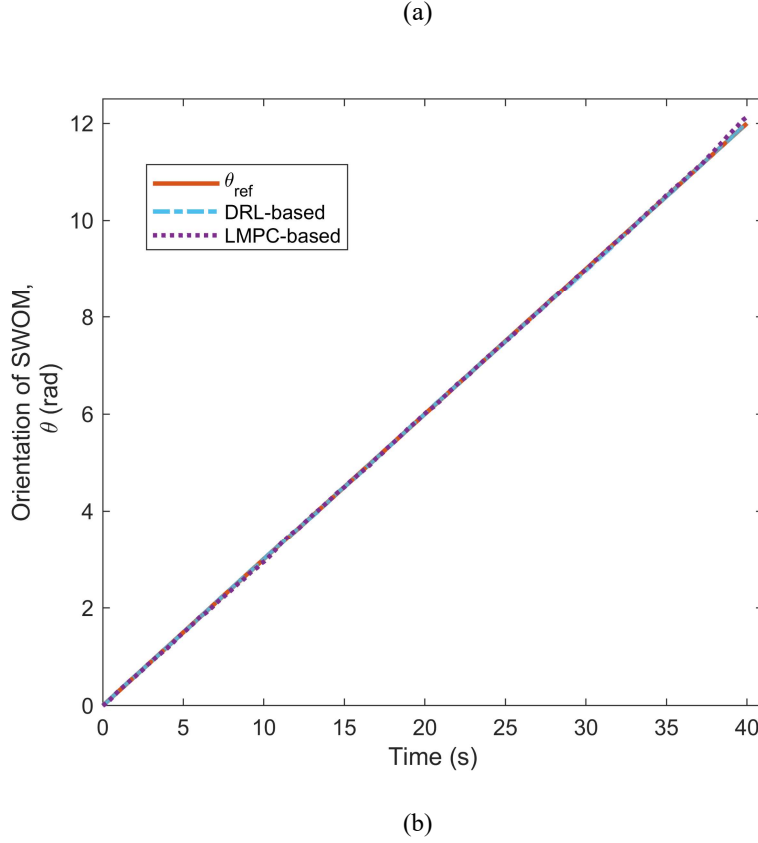


Fig. 3.8 Comparison of trajectory tracking results of SWOM using different control methods. (a) Simulated and reference trajectories of SWOM, (b) Simulated and reference orientations of SWOM. (Dash-dotted lines represent results from the DRL-based method, while dashed lines represent results from the LMPC-based method.)

As shown in Fig. 3.8, both methods achieve satisfactory trajectory tracking performance even in the presence of noise, with the DRL-based approach yielding slightly better results. However, since the LMPC method requires solving an optimization problem at each step, its computation time per step is longer than that of the DRL-based method. When applied to real robotic systems, the minimum feasible time step for LMPC was approximately 0.2 seconds, whereas the DRL-based method can instantly output actions based on the current target state.

Figure 3.9 illustrates the variations in wheel displacement, orientation, and steering velocity during the trajectory tracking simulation. It can be observed that the wheel is quickly moved to the center of the sliding rail and remains near the center throughout the simulation, indicating that the DRL-trained controller is effective in regulating wheel displacement even under noisy conditions. Before and after the turning point in the reference trajectory, the wheel's orientation changes periodically along each segment of the path and consistently remains within the predefined limits.

As shown in Fig. 3.9(c), the wheel's steering velocity remains within a relatively small range for most of the time, which demonstrates that the penalty term R_3 in reward function, associated with the magnitude of the action, effectively contributes to suppressing unnecessary high-speed movements.

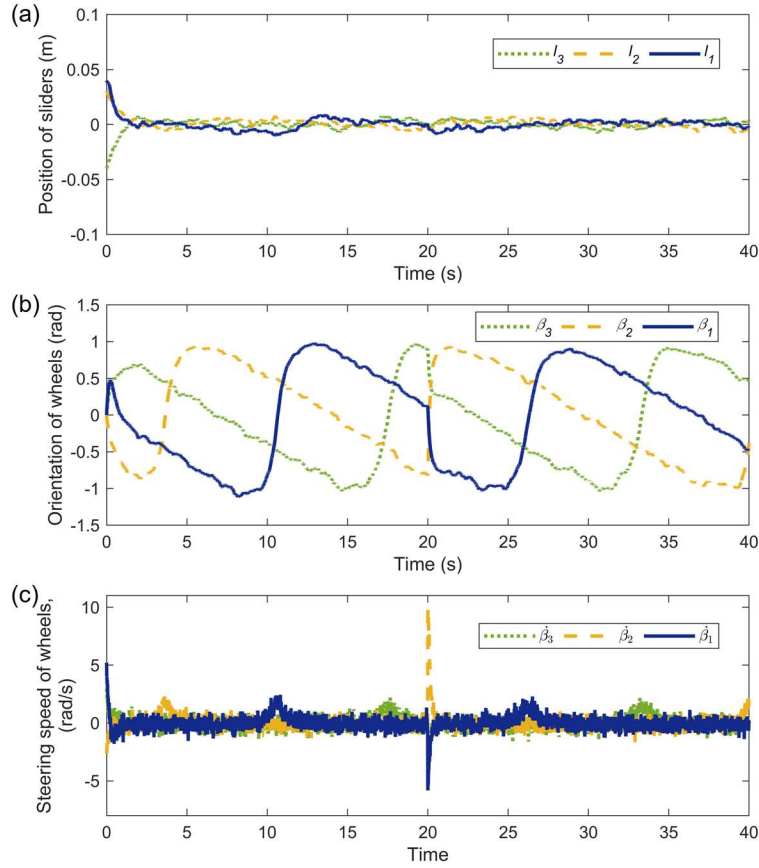


Fig. 3.9 (a) Position of three sliders, (b) orientation of three wheels, and (c) steering speed of three wheels of SWOM in simulation with single corner path.

3.4 Conclusion

In this chapter, I integrated the concepts of hierarchical control to implement DRL-based trajectory tracking control for SWOM. The proposed controller consists of an upper-level controller and a lower-level controller. The upper-level controller employs a traditional Lyapunov-stability-based approach to compute the translational and rotational velocities required for the robot to track the reference path based on the robot's state and the reference state. Using the kinematic equations of SWOM, the required rotational speeds for the three wheels are then calculated. The lower-level controller consists of three controllers trained using the PPO algorithm. These controllers regulate the steering speed of each wheel to maintain the wheels' positions near the center of their respective sliding rails. Each lower-level controller operates independently, taking inputs from the upper-level controller.

Numerical simulations were conducted to validate the effectiveness of the proposed method as well as the effect of the reward function designed in DRL. The term in the reward function related to the wheel displacement velocity has been shown to reduce overshoot during the control process, while the term

associated with the action encourages the agent to produce more efficient control actions. In the simulations of trajectory tracking, SWOM tracked two different reference paths: a figure-eight-shaped trajectory and a single corner trajectory. In the single corner trajectory, I considered the slippage of wheels by adding random disturbances with an amplitude of 0.005 m to the position coordinates of the three wheels. The initial conditions were set such that the wheels' displacements on their sliding rails were away from the center. The results demonstrate that SWOM effectively tracked the reference paths and that the lower-level controllers successfully maintained the wheels near the center of their sliding rails throughout the motion.

Chapter 4. Novel omnidirectional mobile robot with slidable base

4.1 Introduction

In the previous two chapters, I explored MPC-based and DRL-based control strategies for trajectory tracking of SWOM, achieving notable improvements in control performance. Nevertheless, the complex structure of SWOM imposes significant manufacturing costs and complicates practical implementation. To address these challenges, this chapter focuses on robot structural design and introduces a simplified configuration, termed the Slidable-Base Omnidirectional Mobile Robot (SBOM). This design reduces system complexity while retaining essential omnidirectional mobility, thereby providing a more cost-effective and implementable platform. Here, I present the design principles, modeling methodology, and motion capabilities of SBOM, and validate its omnidirectional locomotion performance through both simulations and experiments.

The rest of this chapter is organized as follows. Section 2 describes the structure of the proposed mobile robot and its omnidirectional movement method. Section 3 presents the kinematic analysis and movement strategy. Section 4 describes the proposed approach for avoiding the singular configuration into which the proposed mechanism is likely to fall. Section 5 presents the results of experiments conducted on the prototype to verify its movement and avoidance of singular configuration. Finally, Section 6 concludes the chapter.

4.2 Structure and basic movement

In this section, I describe the proposed omnidirectional robot, its mechanism, and movement method.

4.2.1 Synthesis of slidable-base omnidirectional mobile robot

To improve SWOM, I removed the redundant motors and passive joints. First, I fixed the steering angle of the three wheels at 90 degrees, as shown in Fig. 4.1(a), which removed three motors. These motors are mainly used for controlling the relative position of the wheel on the rail, so removing them does not directly affect the omnidirectional mobility of the robot. Then, as shown in Fig. 4.1(b), I reduced the number of wheels to two. To maintain the active DOFs, one of the remaining sliders needs to be set as the active joint. This mechanism retains one passive joint and one redundant DOF, namely the variability of the distance between the two wheels. I thus combined the active and passive sliding joints into a single joint by swapping the arrangement of the rail and slider. As a result, the base block is connected to an active slider that moves on the rail between the two wheels, as shown in Fig. 4.1(c). A

more refined omnidirectional mobility mechanism is thus achieved.

However, as described in detail in the next section, the omnidirectional mobility of this mechanism is restricted to a certain range of motion due to the limitation of the sliding rail length. To overcome this limitation, I added an active rotating turntable on the top of the base block. This robot is named the slidable-base omnidirectional mobile robot (SBOM).

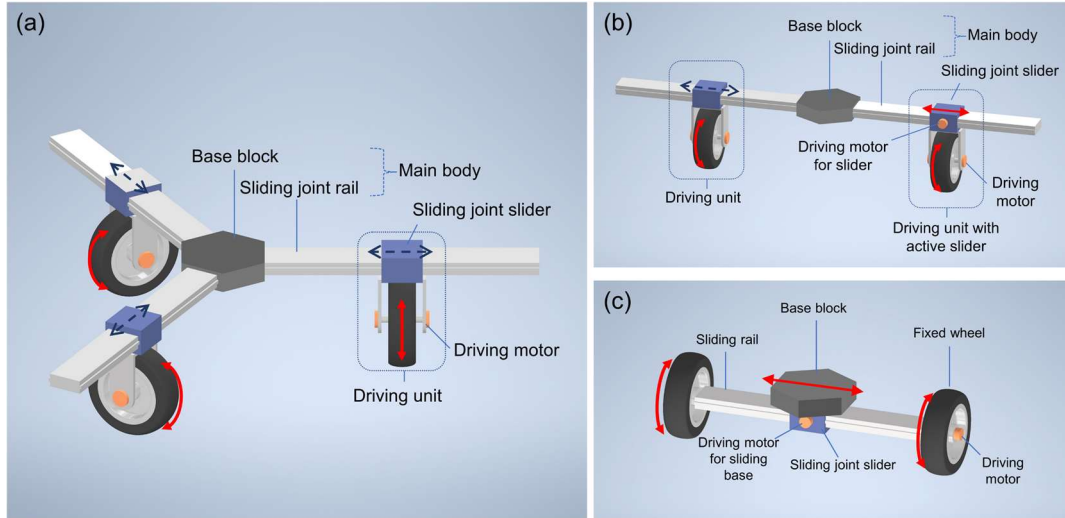


Fig. 4.1 Synthesis of slidable-base omnidirectional mobile robot (SBOM). A solid (dashed) arrow indicates that the joint can actively (passively) move in the direction indicated by the arrow. (a) Steering angle of three wheels fixed at 90 degrees. (b) Number of wheels reduced to two. (c) Base block connected to active slider that moves on rail between two wheels. Auxiliary wheels should be added in practice to maintain its balance for (b) and (c) but omitted here.

The basic structure of the proposed robot is shown in Fig. 4.2. It consists of two fixed wheels, a sliding rail, an active slider, and a turntable that can be actively rotated. The parallel fixed wheels can move independently so the whole robot can move forward, move backward, or turn. The turntable is fixed on the active slider such that it can both actively rotate around its axis and actively move along the rail with the slider. As a result, the output platform can realize omnidirectional movement.

In the following parts of the paper, the turntable and slider are referred to as the base and the other parts of SBOM are referred to as the mobile part. It's worth noting that, this paper aims to achieve omnidirectional movement of the base part of SBOM, i.e., I want both the translational velocity and the rotational velocity of the base part relative to the ground to be able to be in any direction. Since the base can actively rotate in any direction through the turntable, to verify the omnidirectional mobility of the base, I mainly discuss whether the translational velocity of the base can be in any direction.

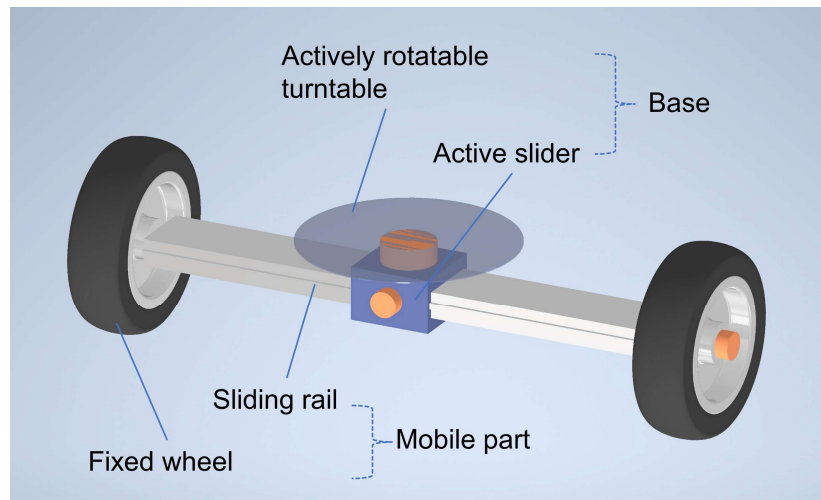


Fig. 4.2 Proposed omnidirectional mobile robot SBOM. Auxiliary wheels should be added in practice to maintain its balance but omitted here.

4.2.2 Motion of slidable-base omnidirectional mobile robot

Figure 4.3 schematically shows one of the omnidirectional movement methods of SBOM. Note that the upward direction in the figure is the forward direction. The mobile part can move forward or backward and turn like a conventional nonholonomic vehicle but cannot move to the left or right. However, the relative movement between the base and the mobile part can be controlled by the active slider, which enables the base to move to the left or right. As a result, the base can move in any direction immediately. In addition, the turntable can actively rotate relative to the base, as mentioned above. Thus, the output platform can realize omnidirectional movement. However, the omnidirectional movement of SBOM shown in Fig. 4.3 is effective only when the relative movement between the base and the mobile part is feasible (i.e., when the active slider is away from the end of the rail). Therefore, SBOM can move in lateral and diagonal directions only within the length of the rail.

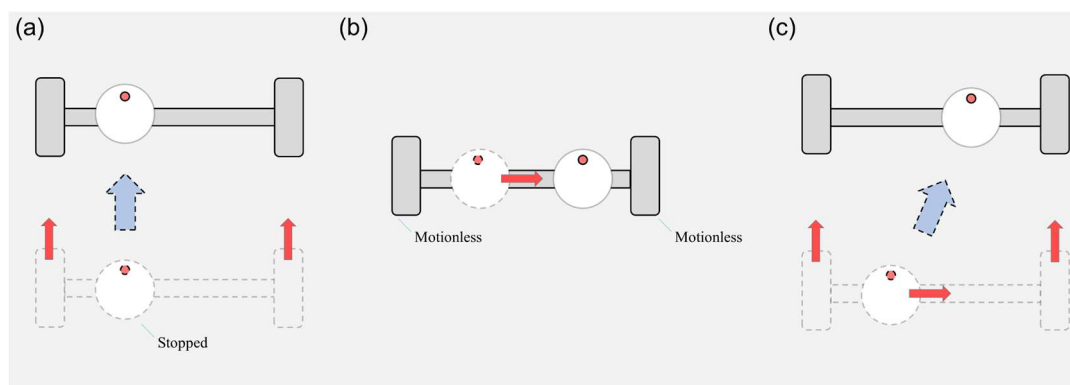


Fig. 4.3 Omnidirectional movement of SBOM. (a) Movement forward. (b) Movement to the right. (c)

Movement forward and to the right. The red spot on the turntable indicates the orientation of the output platform.

To overcome this problem, I propose another method for omnidirectional movement in which the mobile part keeps rotating at an appropriate speed, as shown in Fig. 4.4. With this method, the base does not reach the limit even if it keeps moving in a given direction.

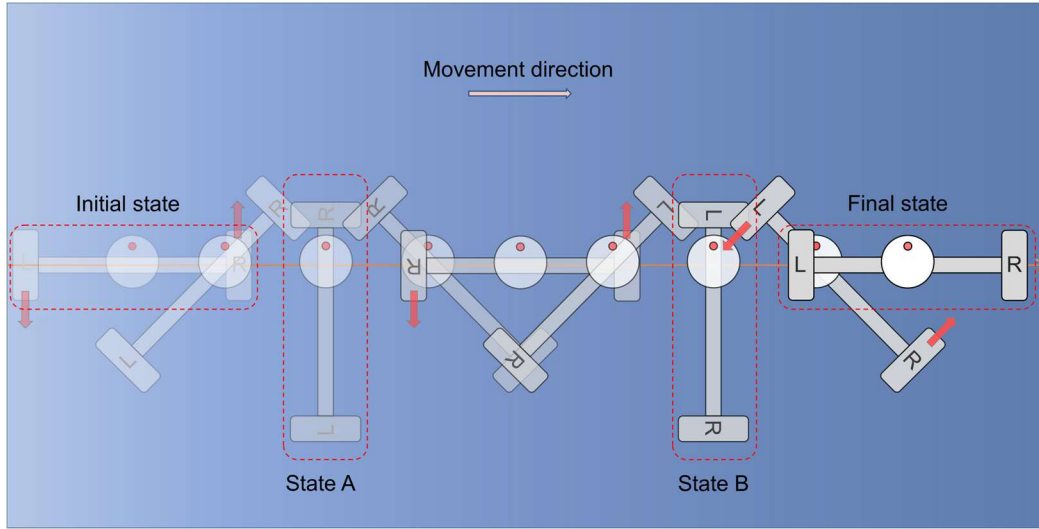


Fig. 4.4 Movement of SBOM with rotating mobile part.

As shown in Fig. 4.4, when the base part keeps moving to the right, since the mobile part keeps rotating, the base part will not lose its omnidirectional mobility by reaching the limit on the sliding rail. In addition, it should be noted that the base rotates at the same speed as the mobile part but in the opposite direction so that it does not rotate relative to the ground throughout this progress. In the global coordinate system, it only has a translational movement to the right. I refer to the wheels located on the left and right in the initial state as the left and right wheels, respectively. In the global coordinate system, the base only has a translational movement to the right, but on the rail, the base initially moves to the right-wheel side and then starts to move to the left-wheel side when it reaches state A. After state B, it starts to move to the right-wheel side again. Finally, the robot returns to the initial state, with only a lateral translation of its position under the global coordinate system. It can be thus concluded that no matter how long the base moves to the right in the global coordinate system, it will never reach the ends of the rail. This holds for all directions.

A disadvantage of the above method is that it takes energy to keep the mobile part rotating. Here I combine the movement methods in Figs. 4.3 and 4.4 as a new strategy, Fig. 4.5 is an example to illustrate it.

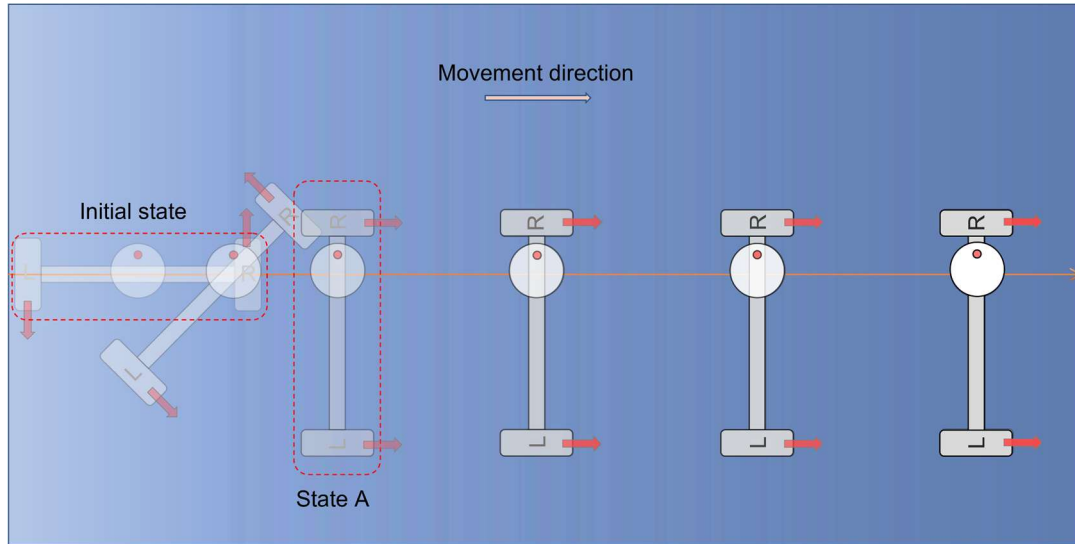


Fig. 4.5 Movement of SBOM with the proposed strategy.

In Fig. 4.5, the robot uses the movement method in Fig. 4.4 at the initial state, the mobile part is rotating while the base is moving towards the right wheel side and rotating at the same speed as the mobile part but in the opposite direction. In the global coordinate system, it only has a translational movement to the right. When the mobile part is oriented towards the desired direction at state A, it is possible to move the base to the right by relying only on the movement of the mobile part, so the robot switches to the movement method in Fig. 4.3(a). Thus, SBOM can efficiently perform omnidirectional movement with no limits.

4.2.3 Characteristics of slidable-base omnidirectional mobile robot

Unlike other omnidirectional mobile robots, SBOM uses conventional fixed wheels. This avoids the problems associated with Mecanum wheels (vibrations), caster wheels (shocks and slippage), and spherical wheels (large wheel volume). In addition, conventional wheels potentially have a greater load capacity, lower manufacturing cost, and higher applicability to various environments. For the structure of SBOM other than wheels, most of the weight applied to the base is carried by the sliding joint. Therefore, the load capacity of SBOM is considered to be dependent on that of the sliding joint. In practice, if I select a linear guide that is strong enough as a sliding joint, SBOM could maintain a high load capacity.

Compared to SWOM, which also uses conventional wheels, SBOM has a simpler structure without redundant motors and passive joints. It thus uses less energy and has a simpler controller design. In addition, SBOM has fewer wheels, which reduces the area it occupies.

Regarding the number of motors, at least three motors are required to give a robot omnidirectional mobility. SBOM uses four motors and has one extra DOF (i.e., rotation of the mobile part) in addition to omnidirectional movement that allows it to control the relative movement between the base and the mobile part. SBOM can thus divide the track at the desired ratio, which is useful not only for singularity avoidance but also for obstacle avoidance or traction control. It is noted that the turntable shown in Fig.

4.2 is small for the sake of illustration, but in practice, it may be designed to be larger depending on the object to be carried, and the object's width may exceed it. Even when the object's width exceeds the length of the sliding rail, as far as I can take the size of the object into account when controlling, SBOM can work properly. The arrangement of the four motors in SBOM allows the turntable's orientation to be decoupled and controlled independently, thus simplifying the control of the robot.

SBOM achieves omnidirectional mobility despite its simple structure. It can thus improve the efficiency of transportation tasks in narrow spaces in factories and warehouses.

4.3 Kinematic analysis of slidable-base omnidirectional mobile robot

To theoretically demonstrate the omnidirectional mobility, this section conducts a kinematic analysis of SBOM.

4.3.1 Modeling of slidable-base omnidirectional mobile robot

The coordinate system and necessary parameters for the kinematic analysis of SBOM are defined in Figs. 4.6 and 4.7, respectively. The global reference frame $O-X_I Y_I$ is fixed on the ground and the configuration of the robot is represented by x, y, θ . Local reference frame $P-X_R Y_R$ is placed on the output platform, with the Y_R axis parallel to the sliding rail. The orientation of SBOM is denoted by θ , which is the angle between the X_R axis and the X_I axis. Note that I deal with the orientation of only the mobile part because that of the turntable can be controlled independently. When the turntable rotates at an appropriate speed by considering the rotation of the robot, it can turn in any direction at any time.

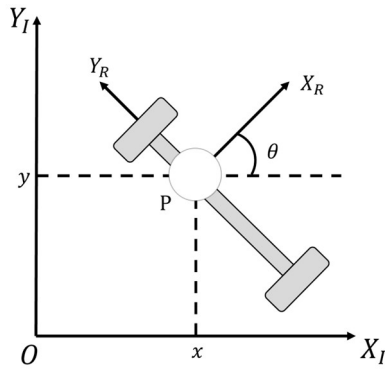


Fig. 4.6 SBOM and reference frames.

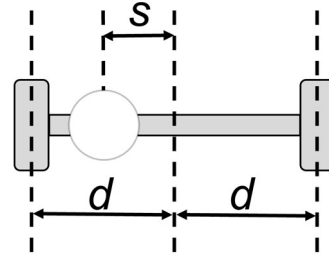


Fig. 4.7 Parameters of SBOM.

In Fig. 4.7, s represents the displacement of the slider from the middle of the rail, where the Y_R -axis direction is defined to be the positive direction, and d is the distance between the middle of the rail and the wheel plane, which is a constant.

With the coordinate systems and parameters defined above, the forward and inverse kinematics of SBOM can be derived as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{d-s}{2d} \cos \theta & \frac{d+s}{2d} \cos \theta & -\sin \theta \\ \frac{d-s}{2d} \sin \theta & \frac{d+s}{2d} \sin \theta & \cos \theta \\ \frac{1}{2d} & -\frac{1}{2d} & 0 \end{bmatrix} \begin{bmatrix} v_R \\ v_L \\ \dot{s} \end{bmatrix} \equiv J \begin{bmatrix} v_R \\ v_L \\ \dot{s} \end{bmatrix} \quad (4.1)$$

$$\begin{bmatrix} v_R \\ v_L \\ \dot{s} \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & d+s \\ \cos \theta & \sin \theta & s-d \\ -\sin \theta & \cos \theta & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = J^{-1} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} \quad (4.2)$$

where v_R is the traveling speed of the right wheel, v_L is the traveling speed of the left wheel, and $\dot{s}$ is the movement speed of the slider.

From the kinematics, SBOM never falls into the singular configuration kinematically because the determinant of Jacobian J is never zero under the condition $-d < s < d$. However, due to the limit of the sliding rail, the robot will fall into the singular configuration when the slider reaches the end of the rail, which can be represented by

$$s = S_{\text{MAX}}, S_{\text{MIN}} \quad (4.3)$$

where S_{MAX} and S_{MIN} are the maximum and minimum limits of the slider's displacement, respectively.

4.3.2 Omnidirectional movement of slidable-base omnidirectional mobile robot

In this section, the methods used to realize omnidirectional movement (see Section 4.2.2) are discussed and simulations are conducted to verify their effectiveness.

Considering the situation where SBOM moves in a certain direction at a constant speed while rotating:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} V \cos(\theta_0 + \gamma) \\ V \sin(\theta_0 + \gamma) \\ \Omega(t) \end{bmatrix} \quad (4.4)$$

where θ_0 is the initial orientation, γ is the designated movement direction relative to the X_R axis at the initial state (see Fig. 4.8), V is the magnitude of the desired translational velocity of the base, and $\Omega(t)$ is the rotational speed of the mobile part.

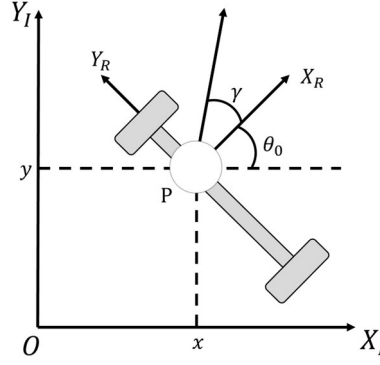


Fig. 4.8 SBOM movement in given direction.

Substituting Eq. (4.4) into Eq. (4.2) yields

$$\begin{bmatrix} v_R(t) \\ v_L(t) \\ \dot{s}(t) \end{bmatrix} = \begin{bmatrix} \Omega(t)(d + s(t)) + V \cos\left(\gamma - \int_0^t \Omega(\tau) d\tau\right) \\ -\Omega(t)(d - s(t)) + V \cos\left(\gamma - \int_0^t \Omega(\tau) d\tau\right) \\ V \sin\left(\gamma - \int_0^t \Omega(\tau) d\tau\right) \end{bmatrix} \quad (4.5)$$

Equation (4.5) can be used to obtain the input of SBOM for moving in a certain direction.

4.3.2.1 Movement method without rotation

First considering SBOM moving without rotation (see Fig. 4.3). When $\Omega(t) = 0$, Eq. (4.5) becomes

$$\begin{bmatrix} v_R(t) \\ v_L(t) \\ \dot{s}(t) \end{bmatrix} = \begin{bmatrix} V \cos \gamma \\ V \cos \gamma \\ V \sin \gamma \end{bmatrix} \quad (4.6)$$

Because the velocity of the slider $\dot{s}$ is a constant, the displacement of the slider will monotonically increase or decrease. Under this situation, SBOM can realize omnidirectional movement only when $S_{\text{MIN}} < s(t) = Vt \sin \gamma + s_0 < S_{\text{MAX}}$.

4.3.2.2 Movement method with rotation at constant speed

If the mobile part of SBOM rotates at a constant speed (see Fig. 4.4), such as $\Omega(t) = \Omega_c = \text{constant}$, Eq. (4.5) becomes

$$\begin{bmatrix} v_R(t) \\ v_L(t) \\ \dot{s}(t) \end{bmatrix} = \begin{bmatrix} \Omega_c(d + s(t)) + V \cos(\gamma - \Omega_c t) \\ -\Omega_c(d - s(t)) + V \cos(\gamma - \Omega_c t) \\ V \sin(\gamma - \Omega_c t) \end{bmatrix} \quad (4.7)$$

When V and γ are constant, the displacement of the slider $s(t)$ can be calculated as

$$s(t) = \frac{V}{\Omega_c} \cos(\gamma - \Omega_c t) - \frac{V}{\Omega_c} \cos \gamma + s_0 \quad (4.8)$$

where s_0 is the initial displacement of the slider. From this equation, it can be seen that the motion of s is simple harmonic vibration, with an initial value $-(V/\Omega_c) \cos \gamma + s_0$, amplitude V/Ω_c , and frequency Ω_c . Therefore, even though the length of the sliding rail is limited, appropriately adjusting the rotation speed Ω_c based on the required velocity ensures that the slider does not exceed its boundaries. The condition on the rotational speed to guarantee this constraint can be derived as follows.

When $\Omega_c > 0$,

$$|\Omega_c| > \max \left(\left| \frac{V(1+\cos \gamma)}{S_{\text{MIN}} - s_0} \right|, \left| \frac{V(1-\cos \gamma)}{S_{\text{MAX}} - s_0} \right| \right) \quad (4.9)$$

When $\Omega_c < 0$,

$$|\Omega_c| > \max \left(\left| \frac{V(1-\cos \gamma)}{S_{\text{MIN}} - s_0} \right|, \left| \frac{V(1+\cos \gamma)}{S_{\text{MAX}} - s_0} \right| \right) \quad (4.10)$$

Simulations were conducted to verify the effectiveness of this movement method. The parameters were set as follows: $V = 40 \text{ mm/s}$, $\gamma = \pi/3$, $S_{\text{MAX}} = 200 \text{ mm}$, $S_{\text{MIN}} = -200 \text{ mm}$, $s_0 = 0 \text{ mm}$, $d = 250 \text{ mm}$, $\Omega_c > 0$. The condition given by Eq. (9) becomes $\Omega_c > 0.3 \text{ rad/s}$, so Ω_c is chosen to be 0.5 rad/s to meet the constraint. The initial condition of SBOM is $x = 0, y = 0, \theta_0 = \pi/2$. These are the parameter values in the following simulations of the movement of SBOM unless otherwise specified.

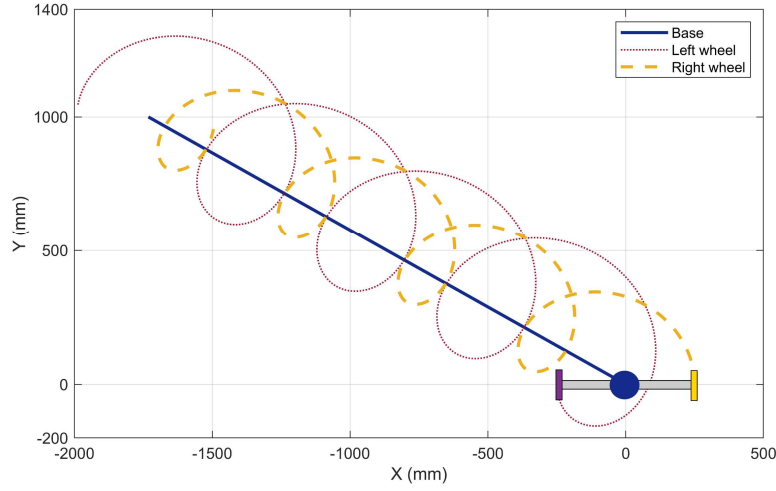


Fig. 4.9 Paths of base and left and right wheels for SBOM moving in given direction.

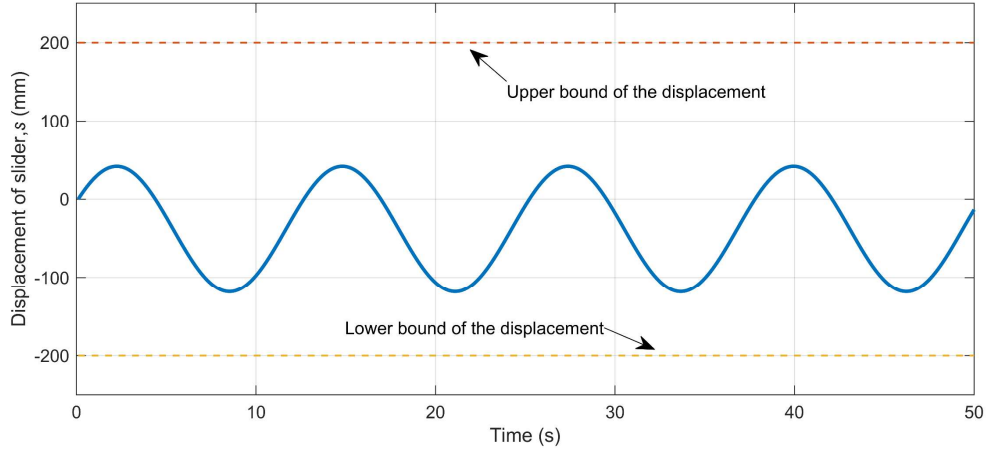


Fig. 4.10 Displacement of slider for SBOM moving in given direction.

Figure 4.9 shows the paths of the base, left wheel, and right wheel. It is confirmed that the base moves straight in the designated direction. Figure 4.10 shows the displacement of the slider s . It is confirmed that s never reaches the limit.

4.3.2.3 Movement strategy that combines two methods

As mentioned in Section 4.2.2, unlimited omnidirectional movement can be realized by first rotating the mobile part to the desired direction and then moving toward the target direction without rotation. The switch from the former movement method to the latter movement method can be done immediately after the orientation of the mobile part matches the desired direction because I can ensure that the singular configuration will not appear during the whole process by setting the proper rotation velocity of SBOM, as shown in the previous section.

When the mobile part of SBOM initially rotates at a constant speed Ω_c , at $t = \gamma/\Omega_c$, the change in the orientation of the mobile part $\Delta\theta$ reaches γ . At this moment, the rotation speed is set to zero (see Fig. 4.5). The kinematics of this motion are given as

$$\begin{aligned}
 \begin{bmatrix} v_R(t) \\ v_L(t) \\ \dot{s}(t) \end{bmatrix} &= \begin{cases} \begin{bmatrix} \Omega_c(d + s(t)) + V \cos(\gamma - \Omega_c t) \\ -\Omega_c(d - s(t)) + V \cos(\gamma - \Omega_c t) \\ V \sin(\gamma - \Omega_c t) \end{bmatrix} & 0 < t < \frac{\gamma}{\Omega_c} \\ \begin{bmatrix} V \\ V \\ 0 \end{bmatrix} & t > \frac{\gamma}{\Omega_c} \end{cases} \\
 (4.11)
 \end{aligned}$$

A numerical simulation was conducted to verify the effectiveness of this method. SBOM was set to initially rotate at $\Omega_c = 0.5$ rad/s.

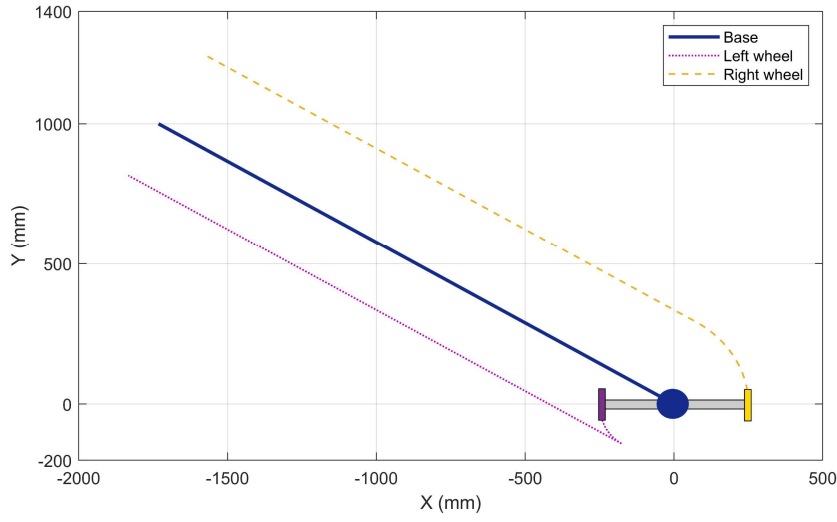


Fig. 4.11 Paths of base and left and right wheels for SBOM moving in given direction using combined method.

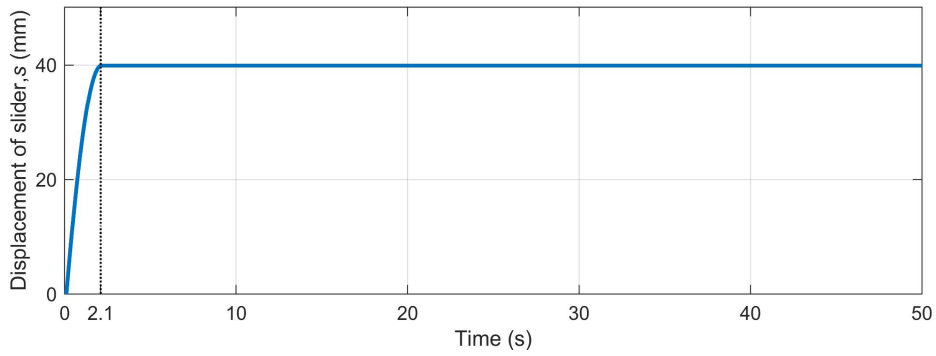


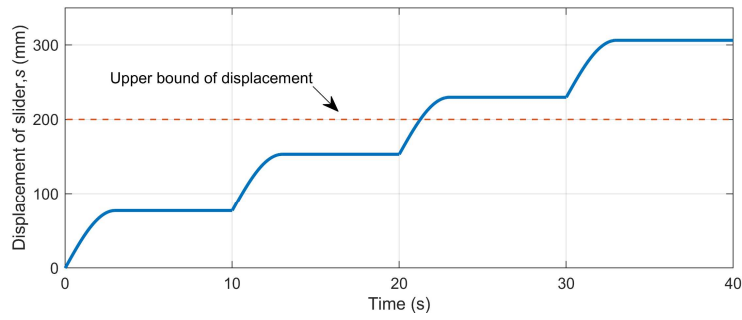
Fig. 4.12 Displacement of slider for SBOM moving in given direction using combined method.

Figure 4.11 shows the paths of the base, left wheel, and right wheel. The robot moved in the designated direction, indicating that the proposed method performs well. Figure 4.12 shows that the

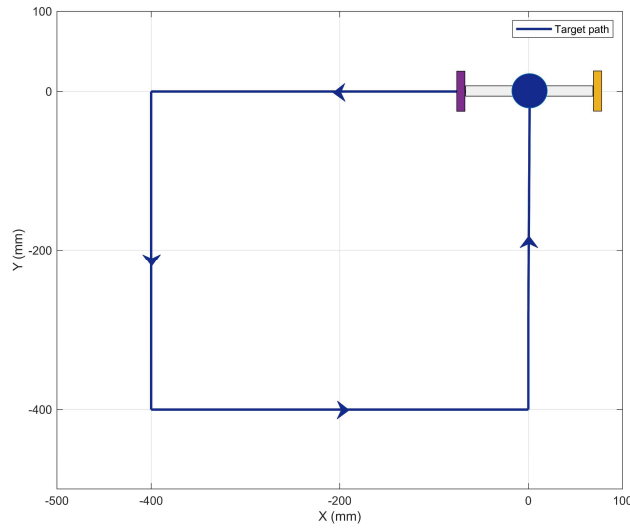
displacement of the slider first increases and then becomes constant after the robot switches to the movement method without rotation. SBOM can move without limits in an arbitrary direction without falling into the singular configuration.

4.4 Stabilization of slider position on rail

As shown in Fig. 4.12, the displacement of the slider first increases and then becomes constant when SBOM moves in a certain direction using the proposed movement method. As a result, the slider may reach its limit if SBOM repeatedly changes its traveling direction. For example, Fig. 4.13(a) shows how the displacement of the slider changes over time when SBOM moves along the path shown in Fig. 4.13(b). Specifically, the robot moves counterclockwise along the sides of a square with a side length of 0.4 m at a velocity of 40 mm/s, its initial position is the origin, and it is oriented in the positive direction of the y axis. The displacement of the slider periodically increases and eventually reaches the left end of the sliding rail.



(a)



(b)

Fig. 4.13 (a) Displacement of slider and (b) simulated path for SBOM moving along square.

To overcome this problem, I propose an algorithm for stabilizing the slider position that considers the additional motion after the direction of the mobile part reaches the desired direction. The total time for the adjustment is denoted as T . The whole process is split into two phases with rotation speeds Ω_1 and $-\Omega_1$, respectively. The duration of each phase is $T/2$. By using Eq. (4.11), the displacement of the slider after T can be calculated as

$$s(t_1 + T) = \frac{2V}{\Omega_1} \left(\cos\left(\frac{\Omega_1 T}{2}\right) - 1 \right) + s(t_1) \quad (4.12)$$

where t_1 is the time when the adjustment starts. It is noted that the orientation of the mobile part after T is the initial orientation; that is, $\theta(t_1 + T) = \theta(t_1)$. Under the assumption that the maximum rotation angle of the mobile part during the adjustment is Φ , which means $\Omega_1 T/2 = \Phi$, the displacement of the slider after T can be derived as

$$s(t_1 + T) = \frac{2V}{\Omega_1} (\cos \Phi - 1) + s(t_1) \quad (4.13)$$

If the desired displacement of the slider is given as s_{ref} and substituted into Eq. (4.13), the corresponding rotation speed Ω_1 and adjustment time T can be derived. Through the series of motions, the displacement of the slider s can be set to the desired value s_{ref} after time T without affecting the omnidirectional movement of the base.

Here, I conduct a simulation of the unlimited translation of SBOM to verify that the proposed method is feasible. The robot was required to complete the task shown in Fig. 4.13. During the motion, the slider was required to return to $s_{ref} = 0$ after each change in the movement direction. The maximum rotation angle of the mobile part during the adjustment was $\Phi = \pi/3$. The results of the simulation are shown in Figs. 4.14 and 4.15.

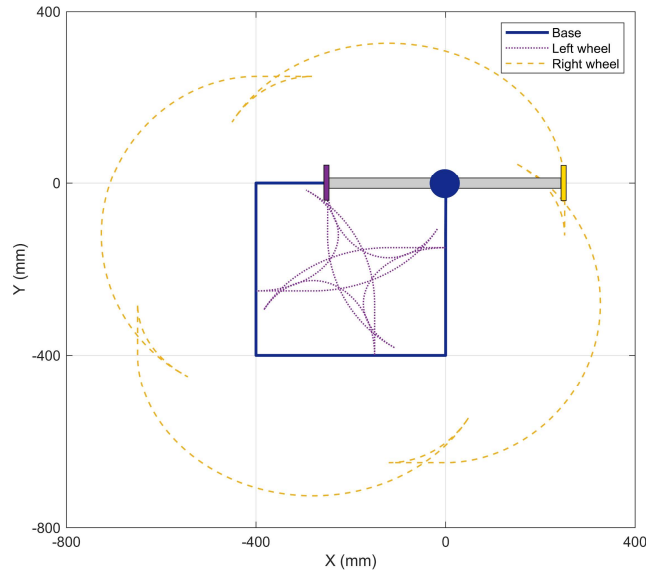


Fig. 4.14 Paths of base and left and right wheels during movement.

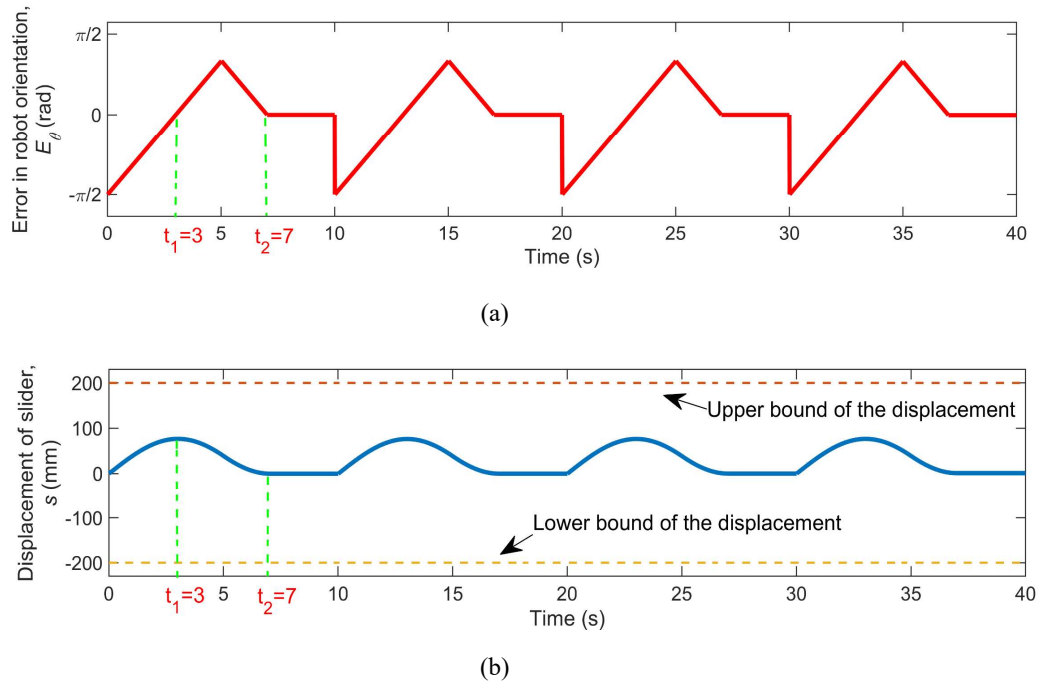


Fig. 4.15 Changes in parameters for SBOM moving along square path. (a) Error in robot orientation with respect to specified direction (E_θ) and (b) displacement of slider.

Figure 4.14 shows the paths of the base, left wheel, and right wheel. As shown, the base moved as expected. Figure 4.15(a) shows the error between the orientation of the robot and the specified direction during the movement E_θ . The specified direction changed at the corners of the path every 10 s, so there are sudden changes in the graph at 10, 20, and 30 s. The displacement of the slider while the robot was moving along the desired path is shown in Fig. 4.15(b). For the first 10 s, Fig. 4.15(a) and (b) shows that

the orientation of the robot meets the specified direction at $t_1 = 3\text{ s}$ and that the displacement of the slider reaches its maximum at this time. The proposed algorithm then starts to adjust the displacement of the slider, and hence E_θ first increases and then decreases, reaching zero again at $t_2 = 7\text{ s}$. The displacement of the slider keeps decreasing and returns to zero at the end of the adjustment. With the proposed algorithm, the slider returns to its original position every time the robot's orientation matches the specified orientation. Therefore, the proposed method is feasible and can prevent SBOM from falling into the singular configuration.

4.5. Experiments and discussion

To confirm that SBOM moves as expected, a prototype was produced and experiments were conducted.

4.5.1 Prototype of slidable-base omnidirectional mobile robot

Figures 4.16 and 4.17 respectively show a photograph and a 3D CAD drawing of the SBOM prototype. The basic structure is the same as that in Fig. 4.2. The left and right wheels are connected to independent motors and can actively rotate. The slider is driven by a rack-and-pinion mechanism. The rack is fixed to the main body of the mobile part parallel to the rails, and the pinion attached to the slider can be rotated by the motor. Another motor is mounted on top of the slider, enabling the turntable to actively rotate. Two ball rollers are mounted at the front and rear of the main body of the mobile part as auxiliary wheels to ensure the robot's stability. The auxiliary wheels are omnidirectional wheels (ball rollers), and their friction is small, so their effect on the motion of the robot is negligible. Table 4.1 shows the specifications of the prototype.

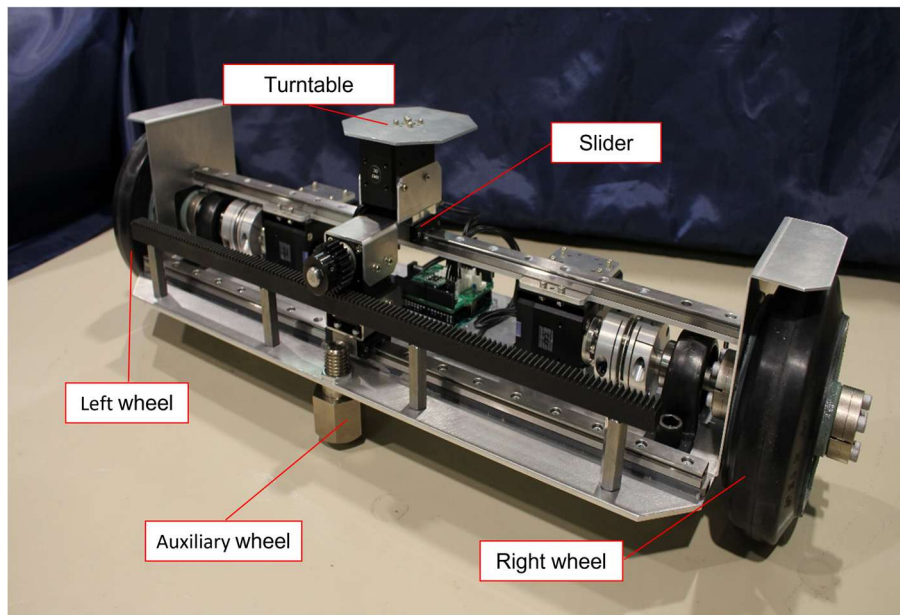


Fig. 4.16 Photograph of SBOM prototype.

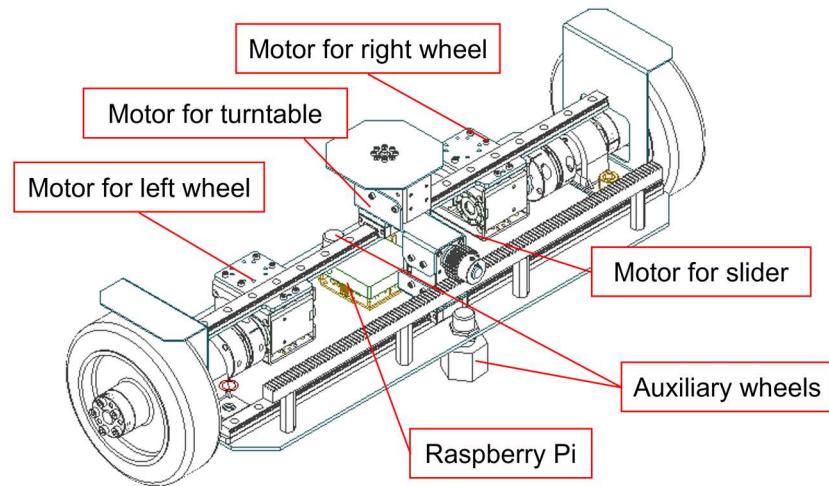


Fig. 4.17 3D CAD drawing of SBOM prototype.

Table 4.1 Parameters of prototype SBOM

Height	155 mm
Width	580 mm
Length	160 mm
Track	500 mm
Wheel diameter	126 mm
S_{MAX}, S_{MIN}	200 mm, -200 mm
Weight	6.5 kg
Motors for wheels	Dynamixel XH540-W150-R (holding torque: 7.1 Nm)
Motors for slider and turntable	Dynamixel XH430-W210-R (holding torque: 2.5 Nm)
Controller	Raspberry Pi 4 Model B 8GB

In the experiments, a Raspberry Pi 4 (Model B, 8 GB) single-board computer was used as the controller for the motors. The basic control system is shown in Fig. 4.18.

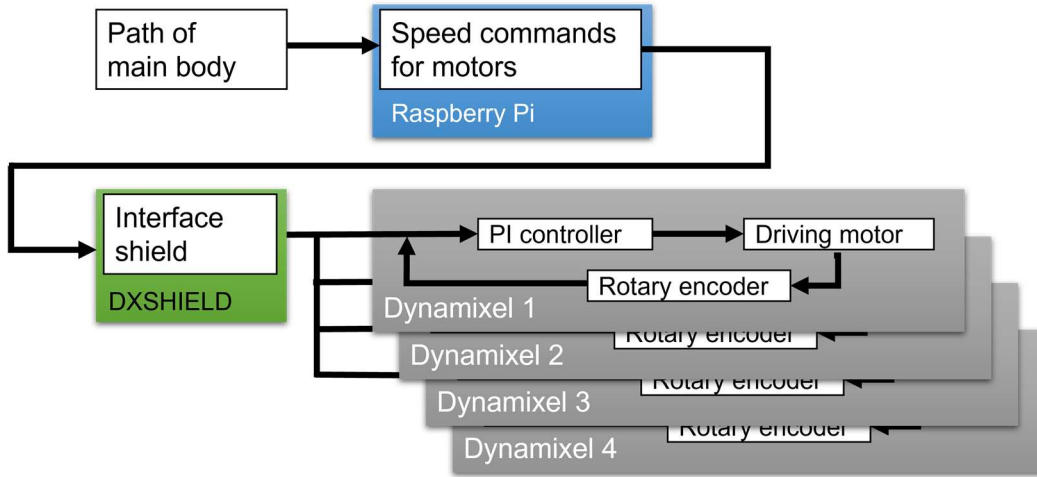


Fig. 4.18 Control system based on Raspberry Pi 4.

4.5.2 Experiments on unlimited translation

To verify the feasibility of the proposed robot, experiments were conducted on the prototype to test the movement method discussed in Section 4.2.2 and the stabilization of the slider. The positions of the base, left wheel, and right wheel were measured by a motion capture system (VICON MX Bonita).

In the first experiment, to show the fundamental omnidirectional mobility, a single-corner path was used. Here, the prototype was required to first move toward the left at 40 mm/s for 10 s and then turn left and move at 40 mm/s for another 20 s. The initial position of the robot was the origin and the robot was oriented in the positive direction of the y axis. The displacement of the slider was required to be adjusted to $s_{ref} = 0$ and the maximum rotation angle of the mobile part during the adjustment was required to be $\Phi = \pi/3$.

The results of the experiment are shown in Figs. 4.19 and 4.20. Figure 4.19 shows the paths of the base, left wheel, and right wheel in the simulation and experiment. The results of the experiment match the simulation relatively well and the robot moves as expected.

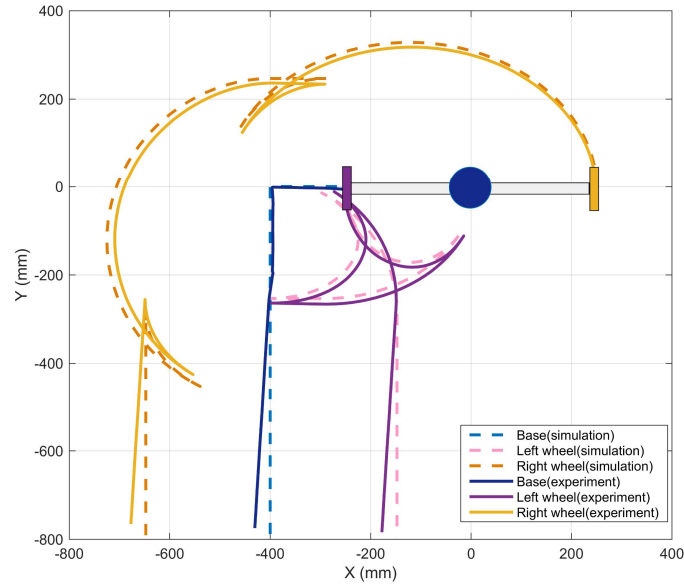


Fig. 4.19 Actual and simulated paths of SBOM for single-corner path.

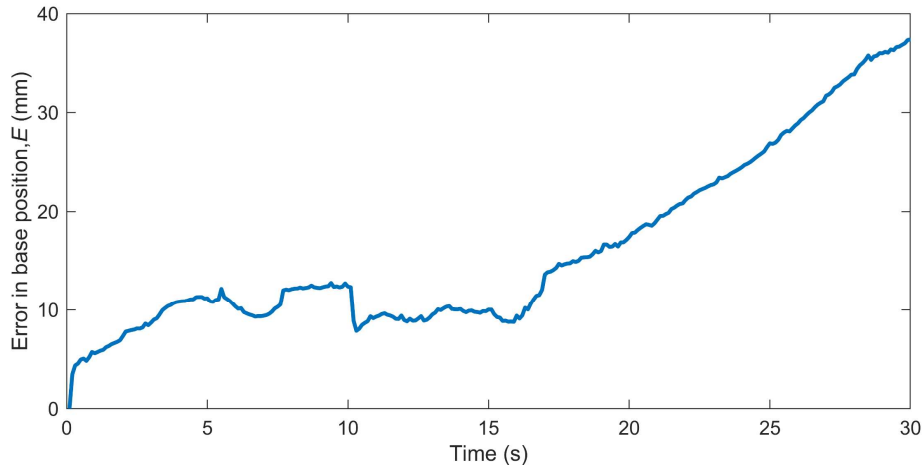


Fig. 4.20 Error in base position during experiment (with respect to simulation results) for single-corner path.

Figure 4.20 shows the error in base position in the experiment, with the simulation results used as the reference. The error increases with time. This error may be due to the slippage between the wheels and the ground, or the acceleration limit not being considered in the design of the inputs. The robot is prone to experience a sudden shock when the rotation direction of the mobile part changes, which may cause unnecessary slippage of the robot. Feedback control can be used as a countermeasure.

Next, to verify the method used for the stabilization of the slider position, the robot was required to make the movements in the simulation shown in Fig. 4.14 from the same initial condition. That is, the prototype was required to move counterclockwise along the side of a square with a side length of 0.4 m at a velocity of 40 mm/s. The initial position of the robot was the origin, and the robot was oriented in the positive direction of the y axis.

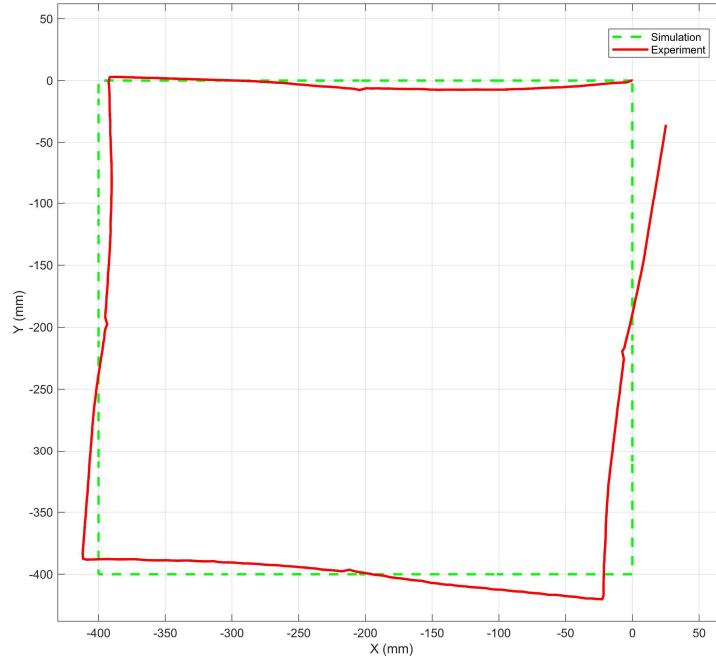


Fig. 4.21 Actual and simulated paths of SBOM for path along square.

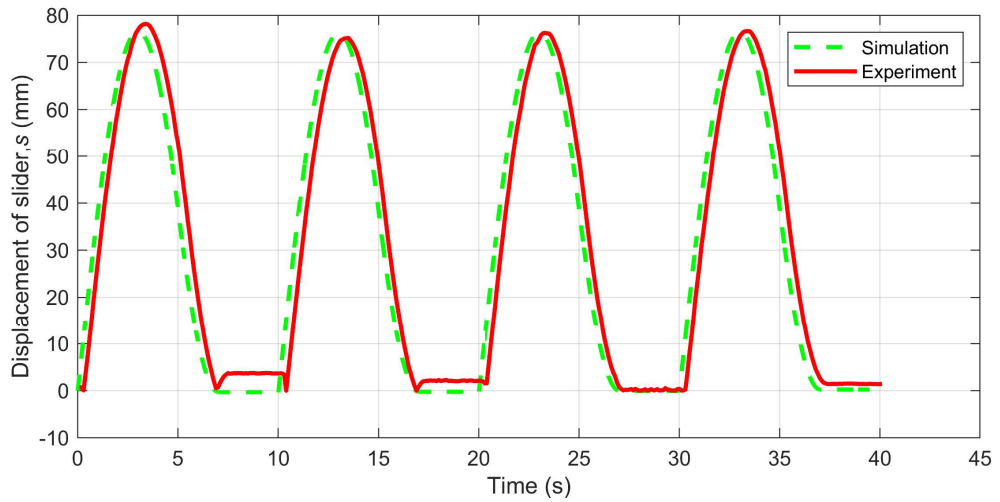


Fig. 4.22 Displacement of slider for path along square.

Figure 4.21 shows the path of the base in the experiment compared to that in the simulation. As in the previous experiment, the error (difference between experimental and simulation results) increases with time. This is considered to be due to slippage and shock. As more turns are made, the error accumulates. Figure 4.22 shows that the displacement of the slider returned to the specified value after each change in direction. Although the position of the slider does not return to the exact specified value due to errors in measurement and the robot itself, it is guaranteed that the slider will not exceed the limit, confirming the effectiveness of the proposed method.

4.6 Conclusion

This chapter proposed a novel two-wheel omnidirectional mobile robot with a Slidable Base, termed SBOM. Unlike conventional designs, SBOM utilizes standard non-omni wheels mounted on a movable base, offering potential advantages in load capacity, mechanical simplicity, and reduced vibration. Its actuation mechanism avoids passive elements and actuator coupling, simplifying controller design and improving reliability.

The fundamental structure and basic motion patterns of SBOM were introduced, followed by a comprehensive kinematic analysis that theoretically verified its omnidirectional mobility. A singularity condition was identified, arising when the slider reached the physical limits of the sliding rail due to accumulated unidirectional motion.

A systematic movement strategy was developed to enable continuous omnidirectional locomotion while avoiding singular configurations. The strategy involves an initial phase where the mobile unit rotates to align with the desired direction while the base translates accordingly, followed by synchronized forward motion of both units. This two-phase approach ensures that the robot maintains controllability even during prolonged travel in a fixed direction.

However, repeated directional changes may lead to irreversible slider drift, potentially causing saturation at the rail ends. To address this issue, a dedicated slider position stabilization algorithm was proposed, actively regulating the slider's position without compromising overall mobility.

Experimental results on a prototype platform confirmed the feasibility of the SBOM concept and the effectiveness of the proposed motion strategy and stabilization algorithm. The robot successfully achieved singularity-free omnidirectional movement, and the stabilization mechanism properly adjusted the slider position in real time. These outcomes highlight SBOM as a promising alternative in the design of robust, high-performance omnidirectional mobile platforms.

Chapter 5. Conclusion

5.1 Summary of contributions.

This dissertation has addressed key challenges in the development and control of omnidirectional mobile robots, with a focus on enhancing mechanical simplicity, load capacity, and control performance. As introduced in Chapter 1, conventional omnidirectional solutions such as Mecanum wheels, caster wheels, and spherical wheels suffer from various drawbacks—including low load capacity, vibration, mechanical complexity, and slippage uncertainty—motivating the exploration of alternative designs. Slidable-Wheeled Omnidirectional Mobile robot (SWOM), which employs conventional wheels mounted on sliding rails, presents a promising direction due to its high load capacity, low vibration, and structural robustness. However, its practical application is hindered by structural complexity—requiring six motors for three DOFs—and the presence of passive sliding components that introduce nonlinearity and control difficulty. Moreover, the inherent non-holonomic constraints of individual wheels and coupling between system states make trajectory tracking a nontrivial task.

To overcome these challenges, this thesis adopted a dual-track approach: (i) developing advanced control strategies to improve the performance of the existing SWOM architecture, and (ii) proposing a novel, simplified robot design that retains the advantages of slidable mechanisms while reducing complexity. These two directions form the core of this work, explored in depth across Chapters 2–4.

In Chapter 2, a Linear Model Predictive Control (LMPC) framework was developed to achieve precise trajectory tracking for SWOM. By linearizing the kinematic model around reference trajectory points via first-order Taylor expansion, the method enables real-time optimization of control inputs. A key contribution lies in the formulation of a comprehensive reference trajectory that includes not only robot pose but also internal states such as slider displacements and wheel orientations. The analysis revealed two critical phenomena: transient peaks in slider displacement caused by sudden velocity changes or alignment of wheel velocity with the rail direction. Practical mitigation strategies—such as incorporating acceleration profiles and increasing rotational rates during direction changes—were proposed and validated. Simulation and experimental results on both single-corner and circular paths confirmed the effectiveness of the LMPC approach in achieving accurate tracking under wheel slippage and actuation imprecision, demonstrating its robustness despite hardware limitations.

Building on the limitations of purely model-based control, Chapter 3 introduced a hybrid hierarchical control architecture that integrates classical control theory with deep reinforcement learning (DRL). The upper-level controller computes desired translational and rotational velocities using a Lyapunov-stability-based method, while the lower-level PPO-trained agents independently regulate each wheel's steering speed to maintain central positioning on the sliding rails. This approach effectively decouples the high-level trajectory tracking task from the low-level mechanical stabilization problem. The reward function design was shown to significantly influence control performance: penalizing wheel displacement velocity reduces overshoot, while action regularization promotes smooth and energy-

efficient behavior. Simulations under disturbance conditions—including initial off-center wheel positions and random positional noise—demonstrated the adaptability and resilience of the DRL-enhanced controller. This work illustrates the potential of combining model-free learning with model-based planning in complex robotic systems.

Recognizing the hardware-level limitations of SWOM, Chapter 4 proposed a conceptually novel two-wheel omnidirectional mobile robot with a Slidable Base, termed SBOM. By relocating the sliding mechanism from the wheels to the base, SBOM achieves omnidirectional mobility using only two conventional wheels and fewer actuators, thereby reducing mechanical complexity, energy consumption, and control coupling. A complete kinematic analysis confirmed the robot’s theoretical omnidirectional capability and identified a singularity condition arising when the slider reaches the end of its rail. To ensure long-term operational stability, a two-phase motion strategy was developed: first aligning the robot’s orientation through differential base-wheel motion, then moving forward synchronously. Furthermore, an active slider position stabilization algorithm was proposed to prevent cumulative drift and avoid singular configurations. Experimental validation on a prototype demonstrated that SBOM can achieve continuous, singularity-free omnidirectional movement. The stabilization algorithm effectively regulates the slider position in real time, confirming the feasibility and practicality of this new design paradigm. SBOM represents a significant step toward simpler, more reliable, and scalable omnidirectional platforms.

5.2 Future work

Despite the progress made, several limitations remain. First, while the proposed hierarchical DRL framework effectively reduces the complexity of the learning task by decoupling high-level navigation from low-level stabilization, it relies on a partially model-based structure. However, given the increasing computational capabilities of modern embedded systems, there is potential to explore end-to-end learning approaches that directly map sensory inputs to actuator commands without explicit decomposition. Second, although the SBOM design demonstrates promising omnidirectional mobility in basic motion tasks, its performance has only been validated under relatively simple scenarios. More complex tasks such as trajectory tracking under disturbances, or payload-carrying navigation require further simulation and experimental validation to fully assess the structural effectiveness and practical applicability of the SBOM concept.

Future research directions include:

- 1) End-to-End DRL for SWOM: Given the sufficient computational resources available in modern robotic platforms, an important direction is to develop and evaluate end-to-end DRL controllers for SWOM. Unlike the current hierarchical approach, such a policy would learn to directly generate motor commands from high-dimensional observations (e.g., robot pose, slider positions, and reference trajectory). Comparative studies between hierarchical and end-to-end architectures can provide valuable insights into the trade-offs between sample efficiency, generalization, and interpretability in complex robotic systems.
- 2) Advanced Task Validation for SBOM: To comprehensively evaluate the SBOM design, future work will involve designing and testing the robot in more challenging scenarios. These include

trajectory tracking under external disturbances, motion planning in cluttered environments, and payload transport tasks to assess load capacity and stability. High-fidelity simulations using physics engines (e.g., MuJoCo or PyBullet) will precede physical experiments to accelerate development and risk assessment.

- 3) Multi-Robot Systems: The control frameworks should be extended to cooperative scenarios involving multiple SBOM/SWOM units for payload transport or formation navigation.

Reference

- [1] Ferriere, L., Raucourt, B. and Campion, G., Design of omnimobile robot wheels. Proceedings of IEEE International Conference on Robotics and Automation (1996), pp. 3664-3670.
DOI: 10.1109/ROBOT.1996.509271.
- [2] Brunhorn, J., Tenchio, O, Rojas, R., A Novel Omnidirectional Wheel Based on Reuleaux-Triangles. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds) RoboCup 2006: Robot Soccer World Cup X. RoboCup 2006. DOI: 10.1007/978-3-540-74024-7_53.
- [3] Djebrani, S., Benali, A. and Abdessemed, F., Modelling and control of an omnidirectional mobile manipulator, International Journal of Applied Mathematics and Computer Science, Vol. 22, (2012), pp. 601-616. DOI: 10.2478/v10006-012-0046-1.
- [4] Komori, M., Matsuda, K., Terakawa, T., Takeoka, F., Nishihara, H. and Ohashi, H., Active omni wheel capable of active motion in arbitrary direction and omnidirectional vehicle, Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol. 10, No. 6 (2016), DOI: 10.1299/jamdsm.2016jamdsm0086.
- [5] Wang, C., Liu, X., Yang, X., Hu, F., Jiang, A. and Yang, C., Trajectory tracking of an omnidirectional wheeled mobile robot using a model predictive control strategy, Applied Sciences, Vol. 8, No. 2 (2018), DOI: 10.3390/app8020231.
- [6] Long, S., Terakawa, T., Yogou, M., Koyano, R. and Komori, M., Kinetic analysis of active omni wheel with barrel-shaped rollers for avoiding slippage and vibration, Journal of Mechanisms and Robotics, Vol. 16, No. 5 (2023), DOI: 10.1115/1.4062608.
- [7] Long, S., Terakawa, T., Komori, M., Nishida, Y., Ougino, T. and Hattori, Y., Effect of double-row active omni wheel on stability of single-track vehicle in roll direction, Mechanism and Machine Theory, Vol. 163 (2021), DOI: 10.1016/j.mechmachtheory.2021.104374.
- [8] Terakawa, T., Komori, M., Yamaguchi, Y. and Nishida, Y., Active omni wheel possessing seamless periphery and omnidirectional vehicle using it, Precision Engineering, Vol. 56 (2019b), pp. 466-475. DOI: 10.1016/j.precisioneng.2019.02.003.
- [9] Feller, D. and Siemers, C., Mechanical design and analysis of a novel three-legged, compact, lightweight, omnidirectional, serial-parallel robot with compliant agile legs. Robotics, Vol. 11, No. 2 (2022), DOI: 10.3390/robotics11020039.
- [10] Gao, Y., Wei, W., Wang, X., Wang, D., Li, Y. and Yu, Q., Trajectory tracking of multi-legged robot based on model predictive and sliding mode control. Information Sciences, Vol. 606 (2022), pp. 489-511.
- [11] Kalouche, S. GOAT: a legged robot with 3D agility and virtual compliance, 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017), pp. 4110-4117. DOI: 10.1109/IROS.2017.8206269.

- [12] Zhu, Y., Chen, L., Liu, Q., Qin, R. and Jin, B., Omnidirectional jump of a legged robot based on the behavior mechanism of a jumping spider, *Applied Sciences*, Vol. 8, No.1 (2018), DOI: 10.3390/app8010051.
- [13] Cao, R., Gu, J., Yu, C. and Rosendo, A., Omniwheg: an omnidirectional wheel-leg transformable robot, 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2022), pp. 5626-5631.
- [14] Ma, J., Zhu, M., Zhang, T. and Yue, X., The wheel-legged robot for granular terrain: Guardian, 28th International Conference on Mechatronics and Machine Vision in Practice (M2VIP) (2022), pp.1-6. DOI: 10.1109/M2VIP55626.2022.10041044.
- [15] Yun, S.H., Park, J., Seo, J. and Kim, Y.J., Development of an agile omnidirectional mobile robot with GRF compensated wheel-leg mechanisms for human environments, *IEEE Robotics and Automation Letters*, Vol. 6, No.4 (2021), pp. 8301-8308. DOI: 10.1109/LRA.2021.3098954.
- [16] Fiedeń, M., Kinematic Analysis of an Omnidirectional Tracked Vehicle, *Applied Sciences*, Vol. 15, No.11 (2025), DOI: 10.3390/app15116111.
- [17] Zhang, Y., Huang, T., Research on a tracked omnidirectional and cross-country vehicle, *Mechanism and Machine Theory*. Vol. 87 (2015), pp.18-44. DOI: 10.1016/j.mechmachtheory.2014.12.016.
- [18] Typiak, A., Łopatka, M.J., Rykała, Ł. and Kijek, M., Dynamics of omnidirectional unmanned rescue vehicle with mecanum wheels, In *AIP Conference Proceedings*, Vol. 1922, No. 1 (2018), pp. 120005. DOI: 10.1063/1.5019120.
- [19] Wampfler, G., Salecker, M. and Wittenburg, J., Kinematics, dynamics, and control of omnidirectional vehicles with Mecanum wheels, *Mechanics of Structures and Machines*, Vol.17, No. 2 (1989), pp. 165-177. DOI: 10.1080/15397738909412814.
- [20] Holmberg, R. and Khatib, O., Development and control of a holonomic mobile robot for mobile manipulation tasks, *The International Journal of Robotics Research*, Vol. 19, No. 11 (2000), pp. 1066-1074. DOI: 10.1177/02783640022067977.
- [21] Jung, E.J., Yi, B.J. and Kim, W., Kinematic analysis and motion planning for a planar multiarticulated omnidirectional mobile robot, *IEEE/ASME Transactions on Mechatronics*, Vol. 20, No. 6 (2015), pp. 2983-2995. DOI: 10.1109/TMECH.2015.2416153.
- [22] Kato, K. and Wada, M., ACROBAT-S omnidirectional mobile robot prototype : and study on ball drive mechanism, 2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM) (2020), pp. 105-110. DOI: 10.1109/AIM43001.2020.9158909.
- [23] Wada, M., Takagi, A. and Mori, S., Caster drive mechanisms for holonomic and omnidirectional mobile platforms with no over constraint, *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, Vol. 2 (2000), pp. 1531-1538. DOI: 10.1109/ROBOT.2000.844814.
- [24] Chen, W.-H., Chen, C.-P., Tsai, J.-S., Yang, J. and Lin, P.-C., Design and implementation of a ball-driven omnidirectional spherical robot, *Mechanism and Machine Theory*, Vol. 68 (2013), pp. 35-48. DOI: 10.1016/j.mechmachtheory.2013.04.012.

- [25] Ok, S., Kodama, A., Matsumura, Y. and Nakamura, Y., SO(2) and SO(3), omni-directional personal mobility with link-driven spherical wheels, 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2011), pp.268-273. DOI: 10.1109/IROS.2011.6095179.
- [26] Ostrovskaya, S., Angeles, J. and Spiteri, R., Dynamics of a mobile robot with three ball-wheels, The International Journal of Robotics Research, Vol. 19, No. 4 (2000), pp. 383-393. DOI: 10.1177/02783640022066923.
- [27] Runge, G., Borchert, G., Henke, P. and Raatz, A., Design and testing of a 2-DOF ball drive, Journal of Intelligent and Robotic Systems, Vol. 81, No. 2 (2016), pp. 195-213. DOI: 10.1007/s10846-015-0247-6.
- [28] West, M. and Asada, H., Design and control of ball wheel omnidirectional vehicles, In Proceedings of 1995 IEEE International Conference on Robotics and Automation (Nagoya), Vol. 2 (1995), pp. 1931-1938. DOI: 10.1109/ROBOT.1995.525547.
- [29] Gferrer, A., Geometry and kinematics of the Mecanum wheel, computer aided geometric design, Vol. 25, No. 9 (2008), pp. 784-791. DOI: 10.1016/j.cagd.2008.07.008.
- [30] Ishigami, G., Iagnemma, K., Overholt, J. and Hudas, G., Design, Development, and mobility evaluation of an omnidirectional mobile robot for rough terrain, Journal of Field Robotics, Vol. 32, No.6 (2015), pp. 880-896. DOI: 10.1002/rob.21557.
- [31] Lauwers, T. B., Kantor, G. A. and Hollis, R. L., A dynamically stable single-wheeled mobile robot with inverse mouse-ball drive. Proceedings 2006 IEEE International Conference on Robotics and Automation (ICRA) (2006), pp. 2884-2889. DOI: 10.1109/ROBOT.2006.1642139.
- [32] Terakawa, T., Komori, M., Matsuda, K. and Mikami, S., A novel omnidirectional mobile robot with wheels connected by passive sliding joints. IEEE/ASME Transactions on Mechatronics, Vol. 23, No. 4 (2018), pp. 1716-1727. DOI: 10.1109/TMECH.2018.2842259.
- [33] Taheri, H. and Zhao, C. X., Omnidirectional mobile robots, mechanisms and navigation approaches, Mechanism and Machine Theory, Vol. 153 (2020), DOI: 10.1016/j.mechmachtheory.2020.103958.
- [34] Terakawa, T., Komori, M. and Fujimoto, K., Control of an omnidirectional mobile robot with wheels connected by passive sliding joints, Journal of Advanced Mechanical Design, Systems, and Manufacturing, Vol. 13, No. 1 (2019a), DOI: 10.1299/jamdsm.2019jamdsm0006.
- [35] Amoozgar, M.H., Sadati, S.H. and Alipour, K., Trajectory tracking of wheeled mobile robots using a kinematical fuzzy controller, International Journal of Robotics and Automation, Vol. 27, No.1 (2012), pp. 49-59. DOI: 10.2316/Journal.206.2012.1.206-3476.
- [36] Li, Z., Deng, J., Lu, R., Xu, Y., Bai, J. and Su, C.Y., Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 46, No. 6 (2016), pp. 740-749. DOI: 10.1109/TSMC.2015.2465352.
- [37] Wang, K., Liu, Y. and Li, L., Visual servoing trajectory tracking of nonholonomic mobile robots without direct position measurement, IEEE Transactions on Robotics, Vol. 30. No. 4 (2014), pp. 1026-1035. DOI: 10.1109/TRO.2014.2317891.

- [38] He, Z., Fu, J., Zhang, M., Zhang, J. and Dai, L., Predictive End-Effector Control for Trajectory Tracking with Obstacle Avoidance on a Mobile Manipulator, In 2025 IEEE 20th Conference on Industrial Electronics and Applications (ICIEA), (2025), pp1-6.
DOI: 10.1109/ICIEA65512.2025.11148850.
- [39] Wang, Y., Sifferman, C., Gleicher, M., IKLink: End-Effector Trajectory Tracking with Minimal Reconfigurations, In 2024 IEEE International Conference on Robotics and Automation (ICRA), (2024), pp. 12165-12171. DOI: 10.1109/ICRA57147.2024.10611510.
- [40] Xie, W., Han, J., Zheng, J., Li, H., Liu, X., Shi, J., Zhang, W., Bai, C. and Li, X., KungfuBot: Physics-Based Humanoid Whole-Body Control for Learning Highly-Dynamic Skills. (2025)
arXiv preprint arXiv:2506.12851
- [41] Zhang, Z., Guo, J., Chen, C., Wang, J., Lin, C., Lian, Y., Xue, H., Wang, Z., Liu, M., Lyu, J. and Liu, H., Track any motions under any disturbances, (2025), arXiv preprint arXiv:2509.13833.
- [42] Qiao, L.,Zhang, W., Double-Loop Integral Terminal Sliding Mode Tracking Control for UUVs with Adaptive Dynamic Compensation of Uncertainties and Disturbances, IEEE Journal of Oceanic Engineering, Vol. 44 (2019),pp. 29-53. DOI:10.1109/JOE.2017.2777638.
- [43] Qiao, L. and Zhang, W., Trajectory tracking control of AUVs via adaptive fast nonsingular integral terminal sliding mode control, IEEE Transactions on Industrial Informatics, Vol. 16, No. 2 (2019), pp.1248-1258. DOI: 10.1109/TII.2019.2949007.
- [44] He, W., Mu, X., Zhang, L. and Zou, Y., Modeling and trajectory tracking control for flapping-wing micro aerial vehicles, IEEE/CAA Journal of Automatica Sinica, Vol. 8, No.1 (2021), pp.148-156. DOI: 10.1109/JAS.2020.1003417.
- [45] Wendemagegn, Y.A., Asfaw, W.A., Abdissa, C.M. and Lemma, L.N., Enhancing trajectory tracking accuracy in three-wheeled mobile robots using backstepping fuzzy sliding mode control, Engineering Research Express, Vol, 6, No. 4 (2024), DOI: 10.1088/2631-8695/ad79b9.
- [46] Yu, R., Zhao, H., Zhen, S., Huang, K., Chen, X., Sun, H. and Shao, K., A novel trajectory tracking control of AGV based on Udwadia-Kalaba approach, IEEE/CAA Journal of Automatica Sinica, Vol. 11, No. 4 (2024), pp. 1069-1071. DOI: 10.1109/JAS.2016.7510139.
- [47] Fujimoto, K., Sakurama, K. and Sugie, T., Trajectory tracking control of port-controlled Hamiltonian systems via generalized canonical transformations, Automatica, Vol. 39, No. 12 (2003), pp. 2059-2069. DOI: 10.1016/j.automatica.2003.07.005.
- [48] Neunert, M., Crousaz, C. de, Furrer, F., Kamel, M., Farshidian, F., Siegwart, R. and Buchli, J., Fast nonlinear model predictive control for unified trajectory optimization and tracking, Proceedings of 2016 IEEE International Conference on Robotics and Automation (ICRA) (2016), pp. 1398-1404. DOI: 10.1109/ICRA.2016.7487274.
- [49] Shen, C., Shi, Y. and Buckham, B., Trajectory tracking control of an autonomous underwater vehicle using lyapunov-based model predictive control, IEEE Transactions on Industrial Electronics, Vol. 65, No. 7 (2018), pp. 5796-5805. DOI: 10.1109/TIE.2017.2779442.
- [50] Holkar, K.S. and Waghmare, L.M., An overview of model predictive control, International Journal

of control and automation, Vol. 3, No. 4 (2010), pp. 47-63.

- [51] Mohanty, S., Artificial neural network based system identification and model predictive control of a flotation column, *Journal of Process Control*, Vol. 19, No. 6 (2009), pp. 991-999.
DOI: 10.1016/j.jprocont.2009.01.001.
- [52] Eduardo F. Camacho, Carlos Bordons, *Model Predictive Control*, Springer publication, London, 2007. DOI: 10.1007/978-1-4471-3398-8
- [53] Mayne, D.Q., Model predictive control: Recent developments and future promise, *Automatica*, Vol. 50, No. 12 (2014), pp. 2967-2986. DOI: 10.1016/j.automatica.2014.10.128.
- [54] Steyn, C. W. and Sandrock, C., Benefits of optimisation and model predictive control on a fully autogenous mill with variable speed, *Minerals Engineering*, Vol. 53 (2013), pp. 113-123.
DOI:10.1016/j.mineng.2013.07.012.
- [55] Dragicevic, T. and Novak, M., Weighting factor design in model predictive control of power electronic converters: an artificial neural network approach, *IEEE Transactions on Industrial Electronics*, Vol. 66, No. 11 (2019), pp. 8870-8880. DOI: 10.1109/TIE.2018.2875660.
- [56] Liu, Y. K. and Zhang, Y. M., Model-based predictive control of weld penetration in gas tungsten arc welding, *IEEE Transactions on Control Systems Technology*, Vol. 22, No.3 (2014), pp. 955-966.
DOI: 10.1109/TCST.2013.2266662.
- [57] Wehr, M., Schätzler, S., Abel, D. and Hirt, G., Model predictive control of an overactuated roll gap with a moving manipulated variable, *Proceedings of 2020 American Control Conference (ACC)* (2020), pp. 1931-1936. DOI: 10.23919/ACC45564.2020.9147360.
- [58] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., Human-level control through deep reinforcement learning, *Nature*, Vol. 518, No. 7540 (2015), pp. 529-533. DOI: 10.1038/nature14236.
- [59] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., Continuous control with deep reinforcement learning, (2015), arXiv preprint arXiv:1509.02971.
- [60] Haarnoja, T., Zhou, A., Abbeel, P. and Levine, S., Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, In *International conference on machine learning (PMLR)*, pp. 1861-1870.
- [61] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O., Proximal Policy Optimization Algorithms, (2017), DOI: 10.48550/arXiv.1707.06347
- [62] Li, G. and Gomez, R., Realizing full-body control of humanoid robots, *Nature Machine Intelligence*, Vol. 6 (2024), pp. 990-991.
- [63] Wang, C., Chen, X., Yu, Z., Dong, Y., Chen, K. and Gergondet, P., Robust humanoid robot vehicle ingress with a finite state machine integrated with deep reinforcement learning, *International Journal of Machine Learning and Cybernetics*, (2024), DOI: 10.1007/s13042-024-02407-w.
- [64] Jiang, D., Wang, H. and Lu, Y., Mastering the complex assembly task with a dual-arm robot based on deep reinforcement learning: a novel reinforcement learning method, *IEEE Robotics and Automation Magazine*, Vol. 30, No. 2 (2023), pp. 57-66. DOI: 10.1109/MRA.2023.3262461.

- [65] Li, J., Shi, H. and Hwang, K.-S., Using goal-conditioned reinforcement learning with deep imitation to control robot arm in flexible flat cable assembly task, *IEEE Transactions on Automation Science and Engineering*, Vol. 21, No. 4 (2024), pp. 6217-6228. DOI: 10.1109/TASE.2023.3323307.
- [66] Kaufmann, E., Bauersfeld, L., Loquercio, A., Mueller, M., Koltun, V. and Scaramuzza, D., Champion-level drone racing using deep reinforcement learning, *Nature*, Vol. 620 (2023), pp. 982-987. DOI: 10.1038/s41586-023-06419-4.
- [67] Song, Y., Steinweg, M., Kaufmann, E. and Scaramuzza, D., Autonomous drone racing with deep reinforcement learning, In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2021), pp. 1205-1212. DOI: 10.1109/IROS51168.2021.9636053.
- [68] Chen, X., Ghadirzadeh, A., Folkesson, J., and Jensfelt, P., Deep reinforcement learning to acquire navigation skills for wheel-legged robots in complex environments, *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 3110-3116.
- [69] Zuo, L., Xu, X., Liu, C. and Huang, Z., A hierarchical reinforcement learning approach for optimal path tracking of wheeled mobile robots, *Neural Computing and Applications*, Vol. 23 (2013), pp. 1873-1883. DOI: 10.1007/s00521-012-1243-4.
- [70] Zhu, K. and Zhang, T., Deep reinforcement learning based mobile robot navigation: a review, *Tsinghua Science and Technology*, Vol. 26, No. 5 (2021), pp. 674-691. DOI: 10.26599/TST.2021.9010012.
- [71] Li, X., Shang, W. W. and Cong, S., Offline reinforcement learning of robotic control using deep kinematics and dynamics, *IEEE-ASME Transactions On Mechatronics*, Vol. 29 (2024), pp. 2428-2439. DOI: 10.1109/TMECH.2023.3336316.
- [72] Pavlichenko, D. and Behnke, S., Real-robot deep reinforcement learning: improving trajectory tracking of flexible-joint manipulator with reference correction, In *2022 IEEE International Conference on Robotics and Automation (ICRA)* (2022), pp. 2671-2677. DOI: 10.1109/ICRA46639.2022.9812023.
- [73] Zhang, S., Sun, C., Feng, Z. and Hu, G., Trajectory-tracking control of robotic systems via deep reinforcement learning, In *2019 IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)* (2019), pp. 386-391. DOI: 10.1109/cis-ram47153.2019.9095802.
- [74] Yu, R., Shi, Z., Huang, C., Li, T. and Ma, Q., Deep reinforcement learning based optimal trajectory tracking control of autonomous underwater vehicle, In *2017 36th Chinese Control Conference (CCC)* (2017), pp. 4958-4965. DOI: 10.23919/ChiCC.2017.8028138.
- [75] Barzegar, A. and Lee, D.-J., Deep reinforcement learning-based adaptive controller for trajectory tracking and altitude control of an aerial robot, *Applied Sciences*, Vol. 12, No. 9 (2022), DOI: 10.3390/app12094764
- [76] Xie, J., Peng, X., Wang, H., Niu, W. and Zheng, X., UAV autonomous tracking and landing based on deep reinforcement learning strategy, *Sensors*, Vol. 20, No.19 (2020), DOI: 10.3390/s20195630
- [77] Deng, R., Zhang, Q., Gao, R., Li, M., Liang, P. and Gao, X., A trajectory tracking control algorithm

- of nonholonomic wheeled mobile robot, In 2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM) (2021), pp. 823-828.
DOI: 10.1109/ICARM52023.2021.9536154.
- [78] Shansi, Z. and Weiming, W., Tracking control for mobile robot based on deep reinforcement learning, In 2019 2nd International Conference on Intelligent Autonomous Systems (ICOIAS) (2019), pp. 155-160. DOI: 10.1109/ICoIAS.2019.00034.
- [79] Tan, G. W., Li, M. S., Hou, B. Y., Zhu, J. H. and Guan, L. X., Achieving accurate trajectory predicting and tracking for autonomous vehicles via reinforcement learning-assisted control approaches, Engineering Applications of Artificial Intelligence, Vol. 135 (2024), DOI: 10.1016/j.engappai.2024.108773
- [80] Srikonda, S., Norris, W. R., Nottage, D. and Soylemezoglu, A., Deep reinforcement learning for autonomous dynamic skid steer vehicle trajectory tracking. Robotics, Vol. 11, No. 5 (2022), DOI: 10.3390/robotics11050095.
- [81] Liu, Y., Ding, W., Yang, M., Zhu, H., Liu, L. and Jin, T., Distributed drive autonomous vehicle trajectory tracking control based on multi-agent deep reinforcement learning, Mathematics, Vol. 12, No. 11 (2024), DOI: 10.3390/math12111614
- [82] Chai, R. Q., Niu, H. L., Carrasco, J., Arvin, F., Yin, H. J. and Lennox, B., Design and experimental validation of deep reinforcement learning-based fast trajectory planning and control for mobile robot in unknown environment. IEEE Transactions on Neural Networks and Learning Systems, Vol. 35, No. 4 (2024), pp. 5778-5792. DOI: 10.1109/TNNLS.2022.3209154.
- [83] Lian, C., Xu, X., Chen, H. and He, H., Near-optimal tracking control of mobile robots via receding-horizon dual heuristic programming, IEEE Transactions on Cybernetics, Vol. 46, No. 11 (2016), pp. 2484-2496. DOI: 10.1109/TCYB.2015.2478857.
- [84] Yu, C., Yang, Y., Cheng, Y., Wang, Z. and Shi, M. M., Trajectory tracking control of an unmanned aerial vehicle with deep reinforcement learning for tasks inside the EAST, Fusion Engineering and Design, Vol. 194 (2023), DOI: 10.1016/j.fusengdes.2023.113894.
- [85] Hu, J., Niu, H., Carrasco, J., Lennox, B. and Arvin, F., Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning, IEEE Transactions on Vehicular Technology, Vol. 69, No. 12 (2020), pp. 14413-14423. DOI: 10.1109/TVT.2020.3034800.
- [86] Xinyi, Y., Fan, Y., Xu, S. and Ou, L., A self-adaptive SAC - PID control approach based on reinforcement learning for mobile robots, International Journal of Robust and Nonlinear Control, Vol. 32 (2021), DOI: 10.1002/rnc.5662
- [87] Yan, Y., Guo, F., Mo, H. and Huang, X., Hierarchical tracking control for a composite mobile robot considering system uncertainties, In 2024 16th International Conference on Computer and Automation Engineering (ICCAE) (2024), pp. 512-517,
DOI: 10.1109/ICCAE59995.2024.10569297
- [88] Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., Cola, G. D., Deleu, T., Goulão, M., Kallinteris, A., Krimmel, M., Kg, A., Perez-vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Tan, H. and Younis,

O. G., Gymnasium: A Standard Interface for Reinforcement Learning Environments, (2024), DOI: 10.48550/arXiv.2407.17032.

- [89] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M. and Dormann, N., Stable-Baselines3: reliable reinforcement learning implementations, Journal of Machine Learning Research, Vol. 22 (2021), pp. 1-8.

List of publication

Chapter 2 consists of:

H. Xu, T. Terakawa, M. Komori, Trajectory tracking control of slidable-wheel omnidirectional mobile robot based on linear model predictive control. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol. 19, (2025), No. 1, pp. JAMDSM0011.

<DOI: 10.1299/jamdsm.2025jamdsm0011>

The original article is under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license (CC BY-NC-ND 4.0) (<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en>)

Chapter 3 consists of:

H. Xu, T. Terakawa, M. Komori, Deep-reinforcement-learning based trajectory tracking control of slidable-wheel omnidirectional mobile robot. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol. 19, (2025), No. 4, pp. JAMDSM0031.

<DOI: 10.1299/jamdsm.2025jamdsm0031>

The original article is under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license (CC BY-NC-ND 4.0) (<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en>)

Chapter 4 consists of:

H. Xu, T. Terakawa, K. Ohyama, K. Shinno, M. Komori, Novel omnidirectional mobile robot with slidable base. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*, Vol. 18, (2024), No. 5, pp. JAMDSM0063. <DOI: 10.1299/jamdsm.2024jamdsm0063>

The original article is under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International license (CC BY-NC-ND 4.0) (<https://creativecommons.org/licenses/by-nc-nd/4.0/deed.en>)

Acknowledgment

Time flies. The memory of my quarantine in a hotel near Kyoto Station upon first arriving in Japan remains vivid—now, in my fifth year as a doctoral student, I would like to express my heartfelt gratitude to everyone who has supported, guided, and cared for me throughout this journey.

First and foremost, I am deeply, deeply grateful to my advisors, Professor Komori and Professor Terakawa. They have provided me with invaluable advice, patiently mentored me, and consistently offered their support and encouragement. They have helped me grow into a better researcher in every possible way. Whenever I struggled to grasp complex concepts, they never grew impatient; instead, they explained things to me again and again until I understood. This truly means a lot to me.

I also extend my sincere thanks to my lab mates—both seniors and juniors. Mr. Siying Long and Mr. Qiang Yao warmly welcomed me when I first arrived and offered tremendous help with everyday life. Mr. Gen Lin, Mr. Fengrui Liu, and Mr. Yang Li provided me with generous support in my research. Playing basketball with Mr. Kojima and Mr. Mamiya was always great fun, and the homemade sweets shared by Mr. Takuya Nakamura were absolutely delicious. Additionally, Mr. Zhongzheng Liang and Mr. Aokun Chen gave me crucial assistance when I was stuck in China and unable to return to Japan to handle administrative procedures.

I'm also grateful to all the members of the Kyoto University Chinese basketball team, who brought so much joy into my daily life. Special thanks go to Mr. Yu Wang, Mr. Zhibo Yang, and Mr. Runze Mao—hanging out and sharing meals with them was my only source of happiness during the pandemic.

Lastly, I want to thank my parents. They not only supported me financially but also gave me unwavering emotional encouragement. They respected my choices and never blamed me for not finishing my degree sooner. I'm also profoundly thankful to my wife, for her understanding and endless patience throughout this long wait for my graduation—thankfully, we've finally made it through together. And of course, I must thank my cat, Mimi, who has stayed by my side all along—even now, she's curled up right next to me.

Without their collective support, I can hardly imagine having completed this academic journey.

Looking back over these past five years, the world has changed so much—and so have I. Now, I'm eager and ready to embrace the unknowns and challenges that lie ahead. Let's go!!