Nonlinear Model Predictive Control for Autonomous Vehicles: Enhancement via Simplified Physics-Aware Prediction and Decomposed Optimization

Mizuho Aoki

Nagoya University

Ph.D. Dissertation

Doctor of Engineering

2025

Abstract

Autonomous driving technologies are becoming increasingly widespread, as both vehicle configurations and usage scenarios continue to diversify. This trend calls for control methods that address diverse control requirements by effectively utilizing vehicle capabilities. Model Predictive Control (MPC), which predicts future system behavior and selects optimal actions, is a promising approach for leveraging the performance of controlled systems. This study aims to improve both the prediction and optimization aspects of MPC to enhance control performance and broaden its applicability. On the prediction side, vehicle models are rationally simplified by incorporating physical characteristics to reduce computational burden and support clearer formulation. On the optimization side, a multi-objective control problem is decomposed hierarchically according to task priorities, alleviating the need for difficult trade-off tuning and facilitating easier controller design. In navigation tasks involving cluttered environments, the proposed method demonstrated more efficient and smoother motion compared to conventional approaches.

Supervisor:

Prof. Tatsuya Suzuki

Examination Committee:

Prof. Tatsuya Suzuki

Prof. Toshiyuki Ohtsuka

Prof. Ichiro Takeuchi

Prof. Hiroyuki Okuda

Contents

List of Figures

List of Tables

Notations

Notations used in this dissertation are listed as follows:

Notations	Descriptions
$\mathbb{N}$	Set of positive integers, i.e., $\mathbb{N} = \{1, 2, 3, \dots\}$
$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	n-dimensional real vector space
$\mathcal{X}$	Set
$\mathbf{x}^\top$	Transpose of $\mathbf{x}$
$\|\mathbf{x}\|_n$	l_n norm of $\mathbf{x}$
$\mathrm{diag}(\mathbf{x})$	Diagonal matrix whose diagonal is $\mathbf{x}$

Chapter 1

Introduction

1.1 Background and Motivation

Mobility has greatly contributed to the progress of civilization. Not only has it enabled humanity to access distant regions across the Earth, but it has also served as a driving force for economic growth, cultural exchange, and humanitarian assistance by efficiently transporting people and goods.

Autonomous driving intelligence represents a transformative advancement that significantly enhances the contributions of mobility. With no need for human operation, it reduces the burden of effort and time, enabling people to benefit from convenient and safe transportation. Moreover, it allows mobility to extend beyond environments that are uninhabitable for humans, facilitating exploration and utilization in extreme conditions such as space, underwater, and hazardous zones. Improving autonomous control methods to fully leverage the motion capabilities of mobility systems broadens their range of applications and helps address the diverse needs of people around the world.

In this context, this dissertation focuses on developing motion planning and control methods that can adapt to increasingly complex and flexible mobility platforms.

1.2 Diversifying Mobility Configurations and Difficulties in Planning and Control

The diversification of mobility configurations has been driven by cumulative advances in vehicle technology over the years. Among these developments, the increase in operational degrees of freedom (DoF) —the number of independently controllable axes—has had a particularly direct impact on motion capabilities. The growing feasibility of high DoF vehicles is driven by the miniaturization of electric motors and improvements in their torque

(a) 2 DoF Vehicle [1]

(b) 5 DoF Vehicle [2]

(c) 8 DoF Vehicle [3]

Figure 1.1: Examples of vehicles with different degrees of freedom

capabilities. These advances have made it increasingly feasible to embed individual motors directly into each wheel, enabling the practical realization of in-wheel motor systems. Conventional four-wheeled vehicles (e.g., 1.1a) typically have 2 DoF: front-wheel steering and longitudinal acceleration. By equipping each wheel with an in-wheel motor while retaining front-wheel steering, vehicles can attain 5 DoF (e.g., 1.1b). The additional DoF enable enhanced cornering performance and the ability to rotate in place. Furthermore, Four-Wheel Independent Drive and Steering (4WIDS) vehicles (e.g., 1.1c), which allow independent control of both the steering angle and driving force at each wheel, possess 8 DoF. Such configurations support not only in-place rotation but also omnidirectional translation, making them particularly suitable for navigating in narrow environments.

A fundamental characteristic of high DoF vehicles is the existence of multiple feasible motion patterns for achieving a given objective. Consider parallel parking as an example (Fig. 1.2): a 2 DoF vehicle must rely exclusively on turning maneuvers to reach the target position, necessitating repeated back-and-forth movements. A 5 DoF vehicle, while capable of similar maneuvers, can achieve sharper turns by coordinating front-wheel steering with independent wheel torque control, thereby reducing the number of back-and-forth movements. Additionally, it can rotate in place after reaching the target position. An 8 DoF vehicle further expands these capabilities by enabling lateral movements, providing an even more diverse set of motion patterns for accomplishing objectives.

The multiplicity of feasible motion patterns for achieving a given objective introduces both advantages and complexities. Vehicles with higher DoF can optimize secondary objectives while satisfying a primary goal. In parallel parking scenarios, these vehicles can minimize travel distance while reaching the target position. This flexibility, however, creates a challenge: solutions that achieve the primary objective may not represent the optimal motion pattern. Control algorithms must therefore identify the best solution among all feasible motion patterns that satisfy the primary objective to fully utilize the capabilities of high DoF vehicles.

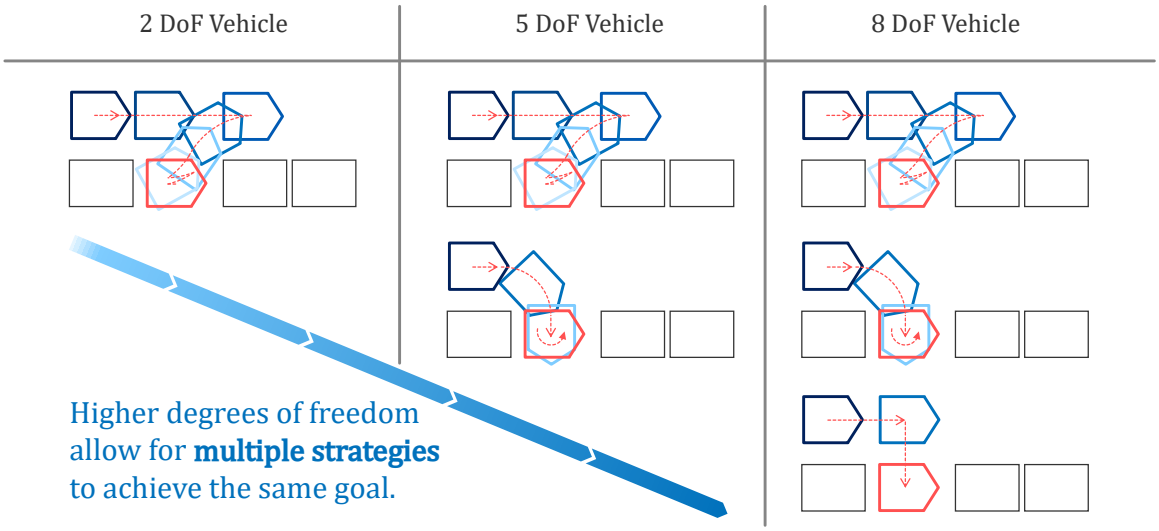


Figure 1.2: Diversity of parallel parking strategies enabled by increased degrees of freedom

1.3 Vehicle Control Architecture for Maximizing Mobility Performance

1.3.1 Comparison of Control Methods for Nonlinear Vehicle Systems

What control methods are suitable for exploiting performance of diverse vehicle configurations? Considering the need to handle nonlinear, multi-input multi-output systems, this section examines four candidate approaches: Control Lyapunov Function (CLF), Model Predictive Control (MPC), Reinforcement Learning (RL), and Imitation Learning (IL), analyzing their respective advantages and limitations. A comparative overview is provided in Table 1.1.

Control Lyapunov Functions (CLFs) establish a theoretical foundation for ensuring stability in nonlinear control systems [4, 5]. CLFs focus on system stabilization rather than exploiting performance capabilities. This approach has been successfully applied to systems where stability is critical, such as multicopters and vehicles [6–8]. The primary challenge of CLF-based methods stems from the need to identify system-specific control Lyapunov functions. This requirement limits their applicability across diverse systems. Recent advances have integrated CLFs with Control Barrier Functions (CBFs) to create constrained quadratic programming formulations (CLF-CBF-QP), to improve safety consideration while maintaining stability guarantees [9].

Model Predictive Control (MPC) is a control methodology that predicts future system behavior and optimizes control actions accordingly [10–12]. The method offers broad applicability across diverse systems and enables explicit handling of state and input con-

straints, although theoretical stability guarantees is challenging to establish due to the numerical optimization process. MPC has demonstrated successful implementation across various complex systems, including vehicles, quadruped robots, and humanoid robots [13–15]. The optimality of control inputs is defined through the minimization of an objective function, making MPC particularly suitable for fully exploiting the performance capabilities of controlled systems. The primary limitations of MPC stem from its computational complexity, which arises from the need to perform high-dimensional optimization based on future state predictions. Additionally, the requirement for a mathematical model of the controlled system for prediction purposes can present implementation challenges in certain applications.

Reinforcement Learning (RL) enables controlled systems to learn optimal operational policies through environmental interactions to maximize rewards [16, 17]. This approach can be applied without prior knowledge of the system's mathematical model, as learning occurs through automated trial and error. The integration of deep neural networks has significantly expanded RL's capabilities and applications [18, 19]. RL has demonstrated success across diverse domains, including legged robot locomotion [20], robotic manipulation [21,22], control of quadruped robots with manipulators [23], and even manipulation of deformable objects such as cloth [24]. A significant challenge in RL implementation lies in the requirement for extensive trial and error, satisfying safety constraints on real systems at an industry-applicable level difficult to achieve [25]. While simulation-to-real (sim-to-real) approaches have been developed to address this challenge, they still face limitations when the gap between simulation and reality is substantial [26].

Imitation Learning (IL) enables systems to learn operational policies by imitating recorded expert demonstrations [27, 28]. While not framed as an optimization problem with explicit performance metrics, this characteristic allows IL to be applicable in scenarios where defining clear performance objectives is challenging. The method has demonstrated successful applications in various domains, including vision-based manipulation [29], dexterous hand manipulation [30], and autonomous driving [31, 32]. A significant limitation of IL lies in its reliance on human-recorded expert data for training, which inherently restricts its effectiveness in scenarios where human operators cannot achieve satisfactory performance.

Among the candidate control methods, **MPC** emerges as the most suitable approach for exploiting the performance capabilities of diversifying mobility systems. While both MPC and RL can achieve optimal solutions under arbitrary objective functions, safety considerations are paramount for mobility applications involving human passengers on public roads. Although constraint satisfaction has been investigated in RL frameworks [33, 34], MPC offers superior practical applicability through its inherent ability to handle time-varying

scenarios and explicitly enforce constraint satisfaction for both system states and control inputs in the near future. The requirement for a mathematical model in MPC is mitigated for mobility systems, as these systems have been extensively studied and well-documented in the literature [35].

Table 1.1: Qualitative comparison of control methods for vehicle control

Method	**Performance Optimality**	**Computation Efficiency**	**Stability Analysis**	**Constraint Handling**	**Applicability to Complex Systems**
CLF	C	A	A	B	C
MPC	A	C	B	A	B
RL	A	B	C	B	A
IL	C	B	C	C	A

1.3.2 Model Predictive Control (MPC)

Model Predictive Control (MPC) is a control methodology that computes optimal control actions through prediction and optimization in real-time. The prediction process employs a mathematical model of the controlled system, termed the prediction model, which forecasts future system states based on current state and future control inputs. The prediction model projects the system's behavior across a defined prediction horizon by simulating the evolution of system states under various control input sequences. The optimization process evaluates these predicted trajectories to determine the optimal control input sequence that minimizes a specified performance metric. The evaluation of control performance is typically formulated as a scalar-valued cost function that maps both control input sequences and corresponding state trajectories to a quantitative measure of system performance.

The application of MPC to mobility control requires careful design of its two fundamental components: prediction and optimization. The prediction and optimization components are further elaborated in Sections 1.3.3 and 1.3.4, respectively.

1.3.3 Mathematical Modeling of Mobility Systems

To control mobility systems using MPC, it is necessary to predict the future behavior of the target system under given control input sequences, which requires a mathematical model of the vehicle.

A common approach to modeling four-wheeled vehicles involves approximating them as two-wheel systems. The Dynamic Bicycle Model [36, 37] represents a relatively precise dynamic model that accounts for tire slip and friction forces, enabling accurate representation of vehicle behavior during high-speed operation. The Kinematic Bicycle Model [36,37] offers a more simplified kinematic representation that neglects wheel friction, making it suitable for modeling vehicle behavior during low-speed operation [38].

Using overly complex prediction models in MPC is undesirable. This approach increases optimization difficulty and requires more physical parameters for prediction. Therefore, it is desirable to use the simplest possible model while maintaining sufficient prediction accuracy. Selecting appropriate prediction models based on the control scenario is a practical approach, and the selection of vehicle prediction models is discussed in detail in Chapter 3.

1.3.4 Multi-Objective Control Problem

Multi-objective control refers to a control methodology that aims to achieve multiple control objectives simultaneously. This approach is commonly required in practical applications, such as maintaining smooth motion while tracking a desired reference path.

The challenge in multi-objective control lies in the potential conflicts between different control objectives, where achieving one objective may lead to suboptimal outcomes for others due to inherent trade-offs. While the trade-offs between multiple performance metrics have been studied extensively through Pareto optimality analysis [39–42], Pareto optimality-based optimization becomes particularly challenging for real-time control applications especially when dealing with high-dimensional optimization problems due to its computational complexity.

For multi-objective control in real-time, many studies adopts the approach of scalarizing multiple task-related terms into a weighted sum within a cost function. While this approach is computationally tractable and effective for systems with few control objectives, tuning the weights becomes increasingly challenging as the number of objectives grows. The need for autonomous driving control that exploits the performance potential of diversifying mobility systems requires the simultaneous achievement of multiple objectives, necessitating a more tractable multi-objective control framework.

A key idea of this dissertation is that while acknowledging the importance of soft balancing the task importance, enabling more explicit consideration of task priorities hierarchically would make multi-objective control more tractable. This concept leads to the proposal of **Nullspace MPC**, a novel optimization approach proposed in Chapter 5.

1.4 Research Questions and Dissertation Structure

This dissertation addresses the central research question: **[Main Q] How can autonomous driving control be realized to exploit the performance potential of diversifying mobility systems?**

The investigation follows a structured approach through several key questions. First, **[Q1] Which control method is suitable for achieving the research goal?** is addressed through a comprehensive literature review, which identifies Model Predictive Control (MPC) as the most appropriate approach (**Chapter 1**).

The analysis then addresses **[Q2] What improvements are needed for MPC?**, revealing two critical areas requiring enhancement: **prediction** and **optimization** aspects (**Chapter 1** and **Chapter 2**).

These aspects are further explored through two specific questions: **[Q2-α] How can practical prediction models be designed?** (addressed in **Chapter 3**) and **[Q2-β] How can the handling of multiple tasks be enhanced?** (examined through an improved version of an existing method in **Chapter 4** and a novel approach in **Chapter 5**). The structure of this investigation is summarized in Fig. 1.3.

This dissertation contributes to advancing research on harnessing the performance potential of diversified mobility systems as mentioned in the main research question. Although substantial progress has been achieved, several limitations remain. The limitations and future research directions are thoroughly examined in **Chapter 6**.

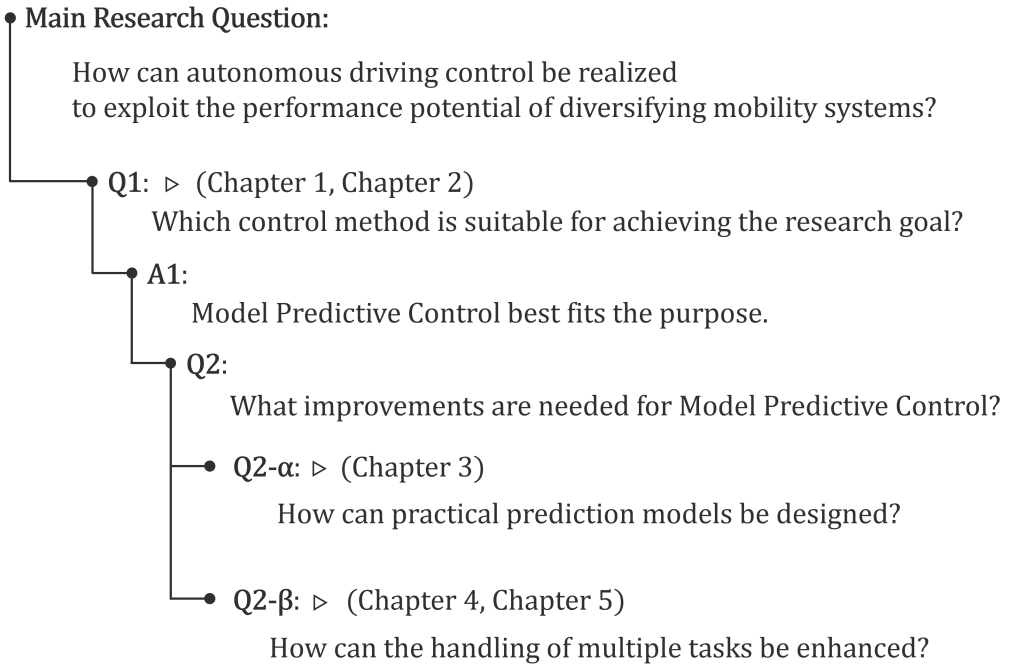


Figure 1.3: Structure of the Research Questions

Chapter 2

Preliminaries

2.1 Chapter Overview and Contribution to Q2-α, Q2-β

This chapter provides the foundational knowledge required for a comprehensive understanding of the dissertation. Section 2.2 defines the coordinate frames used throughout the dissertation. Section 2.3 presents an overview of Model Predictive Control (MPC), which is a central methodology in this research. Section 2.5 introduces a representative study on the application of MPC to 5 DoF vehicle control. This section serves as a basis for discussing the necessary improvements to MPC required to address the main research question of this dissertation.

2.2 Definition of Coordinate Frames

This section defines the coordinate frames used to express vehicle models and problem formulations.

The following provides the definition of coordinate frames used in this dissertation. Fig.2.1 shows the geometrical relationship among these coordinate frames. For a detailed explanation of these coordinate systems, refer to [44]. The descriptions of Global Frame, Base Frame, and Frenet-Serret Frame are given below.

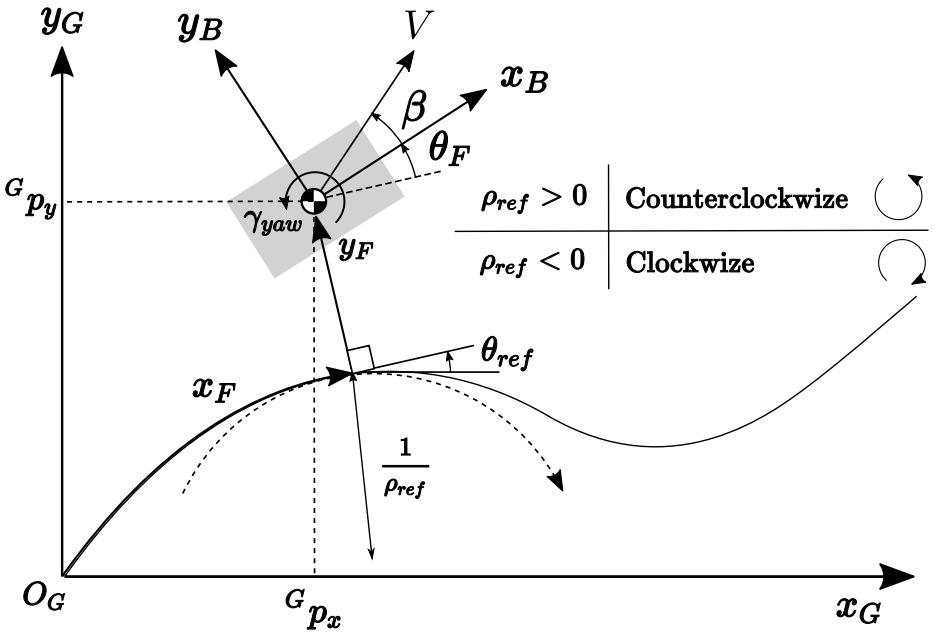


Figure 2.1: Geometrical Relationship Among Coordinate Frames

Global Frame

The cartesian coordinate system in a fixed inertial frame. The position vector in this frame is defined as ${}^G P = [{}^G p_x, {}^G p_y]$.

Base Frame

The frame fixed on the moving frame at the position of the center of gravity (CG) of the car. The position vector in this frame is defined as ${}^B P = [{}^B p_x, {}^B p_y]$. The x_G axis direction is the same as the front of the vehicle.

Frenet-Serret Frame

The frame fixed on the moving frame at the nearest point on the reference path from the car. The position vector is defined as ${}^F P = [{}^F p_x, {}^F p_y]$. x_F axis aligns with the tangent of the reference path and ${}^F p_x$ means the trajectory length from the origin. y_F axis aligns with the reference path, and ${}^F p_y$ value shows the lateral error between the car and the reference path.

2.3 Model Predictive Control (MPC)

Model Predictive Control (MPC) is a control strategy that combines prediction of future system behavior with optimization of control inputs at each time step. Fig.2.2 illustrates

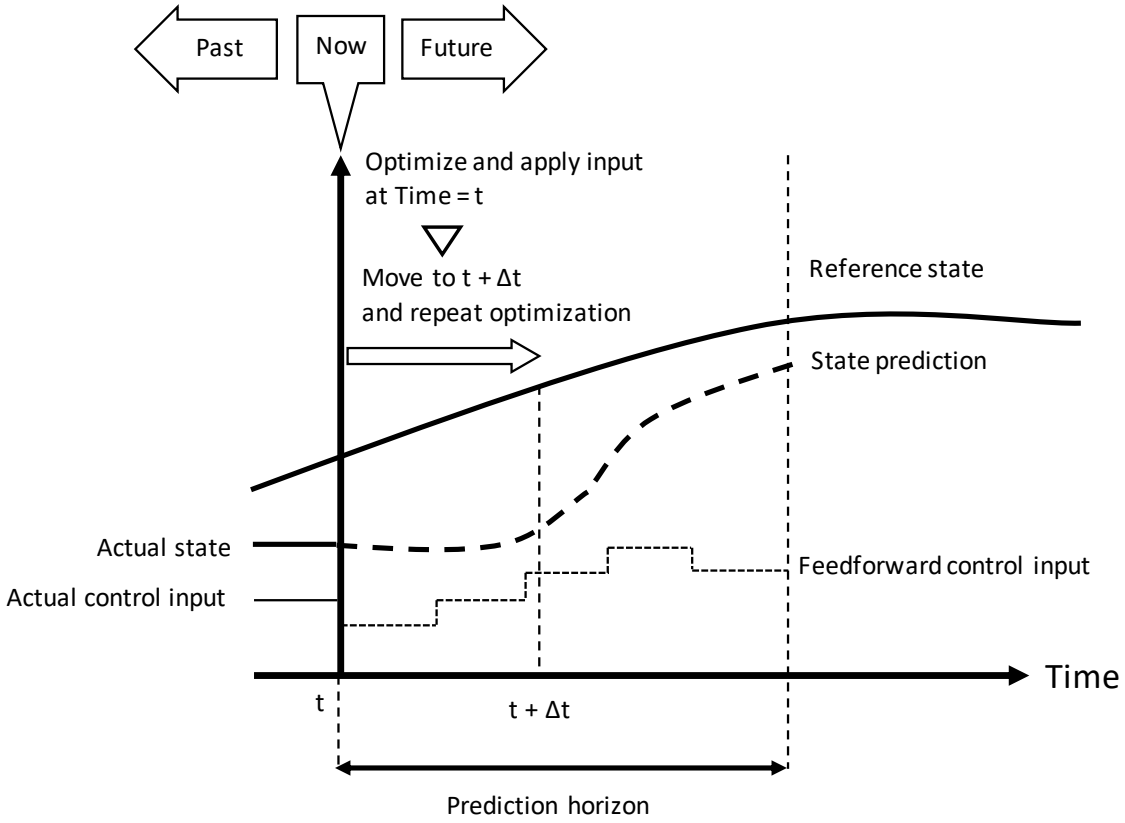


Figure 2.2: Overview of Model Predictive Control (MPC)

the basic concept of MPC. At each time step t, the controller solves a finite-horizon optimal control problem to compute a sequence of control inputs and corresponding predicted states. Only the first input in the sequence is applied to the system, and the process is repeated at the next time step $t + \Delta t$. This scheme, known as the receding horizon approach, enables continuous feedback and allows the controller to react to disturbances and model inaccuracies, thereby improving robustness and adaptability in dynamic environments.

The advantages and limitations of MPC are summarized below. MPC offers several benefits over conventional control methods. It explicitly optimizes a cost function over a finite horizon, enabling the computation of control inputs that minimize the specified cost. A key advantage is its ability to handle constraints on both state and input variables, which is generally difficult to achieve with traditional feedback control approaches. However, MPC also has inherent drawbacks. It requires solving an optimization problem at every control cycle, which can be computationally demanding, particularly for nonlinear or high-dimensional systems. Furthermore, because the optimization is performed over a finite horizon, analyzing and guaranteeing closed-loop stability can be challenging.

The general structure of a MPC problem is summarized below. At each time step t,

the real system state $x(t)$ is obtained through measurement or estimation, and used as the initial condition for optimization. The controller predicts future states $\hat{x}(k \mid t)$ and computes control inputs $\hat{u}(k \mid t)$ over a finite prediction horizon $k = 0, \ldots, N$, where k denotes the step index within the horizon and t indicates the current time at which the optimization is performed. The cost function J to be minimized consists of a terminal cost $\Phi(\hat{x}(N \mid t))$ and a sum of stage costs $L(\hat{x}(k \mid t), \hat{u}(k \mid t))$ along the horizon. The optimization is subject to discrete-time system state transition based on a prediction model f, as well as state and input constraints. This general formulation is detailed in Eqs. (2.1)–(2.4).

Given

$$\begin{aligned} &\hat{x}(0 \mid t) = x(t), \quad N \in \mathbb{N}, \quad \Delta t > 0, \\ &\Phi : \mathbb{R}^n \to \mathbb{R}, \quad L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}, \\ &f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n \quad \text{(prediction model)} \end{aligned} \tag{2.1}$$

Find

$$\begin{aligned} &\hat{x}(k \mid t) \in \mathbb{R}^n, \quad \forall k \in \{1, \ldots, N\}, \\ &\hat{u}(k \mid t) \in \mathbb{R}^m, \quad \forall k \in \{0, \ldots, N-1\} \end{aligned} \tag{2.2}$$

Objective: Minimize Cost Function J

$$J = \Phi\big(\hat{x}(N \mid t)\big) + \sum_{k=0}^{N-1} L\big(\hat{x}(k \mid t), \hat{u}(k \mid t)\big)\Delta t \tag{2.3}$$

Subject to

$$\begin{aligned} &\hat{x}(k+1 \mid t) = f\big(\hat{x}(k \mid t), \hat{u}(k \mid t)\big), \quad \forall k \in \{0, \ldots, N-1\}, \\ &\hat{x}(k \mid t) \in \mathcal{X}, \quad \hat{u}(k \mid t) \in \mathcal{U}, \quad \forall k \end{aligned} \tag{2.4}$$

Various algorithms have been developed to solve the optimization problem formulated above. Section 2.4 introduces representative algorithms classified into two categories: Gradient-Based Algorithms and Sampling-Based Algorithms.

2.4 Optimization Algorithms

As MPC requires solving optimization problems in real-time, the role of optimization algorithms is crucial in determining whether practical control can be achieved. If the computation time exceeds the control period or if the optimization fails to converge, it directly leads to control failure. Various optimization algorithms have been proposed, each with its strengths and weaknesses. The controller designer must precisely understand the properties

of the optimization problem to be solved and select the appropriate algorithm. Numerous optimization algorithms exist, and each has its own strengths and weaknesses. The selection of an appropriate algorithm requires a thorough understanding of the optimization problem's characteristics.

This section classifies optimization algorithms into two main categories: Gradient-Based Algorithms and Sampling-Based Algorithms. Section 2.4.1 introduces gradient-based algorithms, while Section 2.4.2 presents sampling-based algorithms, along with their representative methods and characteristics.

The optimization method **SA-HQP** proposed in Chapter 5 of this dissertation can be interpreted as a framework that combines both gradient-based and sampling-based algorithms to leverage their respective advantages. The content of this section serves as prerequisite knowledge for understanding the framework.

2.4.1 Gradient-Based Algorithms

Gradient-based algorithms utilize gradient information to iteratively update the solution toward the optimal point. Representative algorithms include gradient descent [45, 46], Newton's method [45–47], C/GMRES [48], ADMM [49], and PANOC [50, 51].

Gradient-based algorithms demonstrate superior performance for convex and differentiable optimization problems. These methods achieve rapid convergence and scale effectively to high-dimensional spaces. On the other hand, these algorithms may converge to local optima when applied to non-convex problems. The quality of the solution in such cases exhibits strong dependence on the initial guess.

2.4.2 Sampling-Based Algorithms

Sampling-based algorithms generate multiple candidate solutions through random sampling and evaluate them to find optimal or near-optimal solutions [52]. These methods include Monte Carlo Model Predictive Control (MC-MPC) [53], Cross-Entropy Method (CEM) [54], and Model Predictive Path Integral (MPPI) [55].

Sampling-based algorithms offer several advantages, including broad applicability to non-convex and non-differentiable optimization problems. These methods exhibit lower sensitivity to the choice of initial guess compared to gradient-based approaches, resulting in reduced susceptibility to local optima. However, these benefits come at the cost of significantly higher computational complexity, particularly when applied to high-dimensional search spaces.

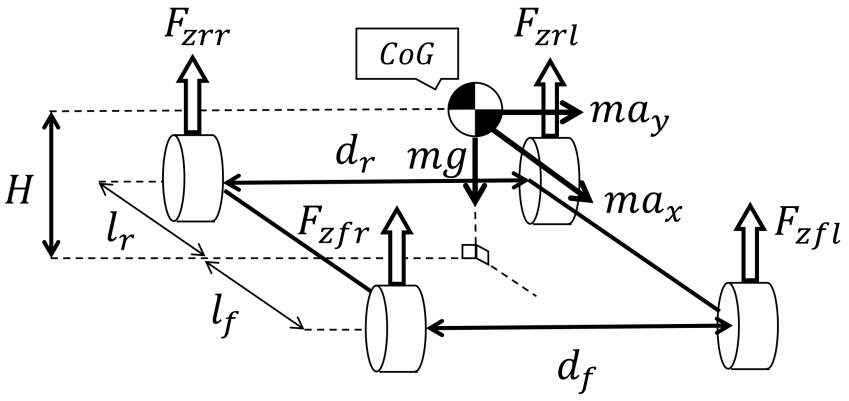


Figure 2.3: Tire Load

2.5 Application of MPC to 5 DoF Vehicle

This section presents a standard application of Model Predictive Control (MPC) to a five-degree-of-freedom (5 DoF) vehicle control system. The 5 DoF vehicle features independent drive torque control for each wheel in addition to front-wheel steering. This configuration enables vehicle turning through both steering angle control and wheel torque distribution. The following demonstrates how MPC can effectively utilize this high degree of freedom to achieve sophisticated control objectives. Additionally, this section addresses practical challenges in MPC implementation from both prediction and optimization perspectives.

2.5.1 Mathematical Modeling of Vehicle Dynamics for Future Prediction

In this section, a nonlinear full vehicle model [56] is introduced to predict the future states of a 5 DoF vehicle. In the following, this vehicle model is derived using the frenet-serret coordinate system shown in Fig.2.1. All definitions of the variables in the following are shown in Table. 2.1.

First, assuming that the road surface is flat, the load on each wheel F_{z**} can be calculated as follows:

$$F_{zfl} = \frac{m_f g}{2} - \frac{mHa_x}{2l} - \frac{m_f Ha_y}{d_f}, \tag{2.5}$$

$$F_{zfr} = \frac{m_f g}{2} - \frac{mHa_x}{2l} + \frac{m_f Ha_y}{d_f}, \tag{2.6}$$

$$F_{zrl} = \frac{m_r g}{2} + \frac{mHa_x}{2l} - \frac{m_r Ha_y}{d_r}, \tag{2.7}$$

$$F_{zrr} = \frac{m_r g}{2} + \frac{mHa_x}{2l} + \frac{m_r Ha_y}{d_r}, \tag{2.8}$$

Table 2.1: Definition of Variables

Variable	Definition	Value [Unit]
${}^{G}p_x, {}^{G}p_y$	X/Y axis of the Global Frame	- [m]
${}^{B}p_x, {}^{B}p_y$	X/Y axis of the Base Frame	- [m]
${}^{F}p_x, {}^{F}p_y$	X/Y axis of the Frenet-Serret frame	- [m]
θ_F	Yaw Angle between the Ref. Path	- [rad]
β	Slip Angle of Vehicle Body	- [rad]
γ	Yaw Rate of Vehicle Body	- [rad/s]
δ	Front Wheel Angle	- [rad]
T	Torque generated by a tire.	- [Nm]
ρ_{ref}	Curvature of Ref. Path	- [m^{-1}]
θ_{ref}	Yaw Angle of the Ref. in Global Frame	- [rad]
V	Vehicle velocity	- [m/s]
a_x	Longitudinal acceleration	- [m/s^2]
a_y	Lateral acceleration	- [m/s^2]
l_f	Distance from CoG to the Front Axle	1.04 [m]
l_r	Distance from CoG to the Rear Axle	1.56 [m]
d_f, d_r	Front / Rear tread	2.082 [m]
m	Mass of vehicle body	1270 [kg]
I_z	Inertia of vehicle yaw moment	1343 [kgm^2]
H	Height of CoG	0.540[m]
R	Tire radius	0.3[m]
e, f	Tire parameter coefficients	4.15, 855.0
g	Gravitational acceleration	9.807 [m/s^2]

CoG: center of gravity

where, $m_f = ml_r/(l_f + l_r)$, $m_l = ml_f/(l_f + l_r)$ and the definitions of the the variables are shown in Table. 2.1. Next, each tire slip angle is derived from the geometric relationships by

$$\beta_{fl} = \tan^{-1}\left(\frac{V\sin\beta + l_f\gamma}{V\cos\beta - \frac{d_f\gamma}{2}}\right) - \delta, \tag{2.9}$$

$$\beta_{fr} = \tan^{-1}\left(\frac{V\sin\beta + l_f\gamma}{V\cos\beta + \frac{d_f\gamma}{2}}\right) - \delta, \tag{2.10}$$

$$\beta_{rl} = \tan^{-1}\left(\frac{V\sin\beta - l_r\gamma}{V\cos\beta - \frac{d_r\gamma}{2}}\right), \tag{2.11}$$

$$\beta_{rr} = \tan^{-1}\left(\frac{V\sin\beta - l_r\gamma}{V\cos\beta + \frac{d_r\gamma}{2}}\right). \tag{2.12}$$

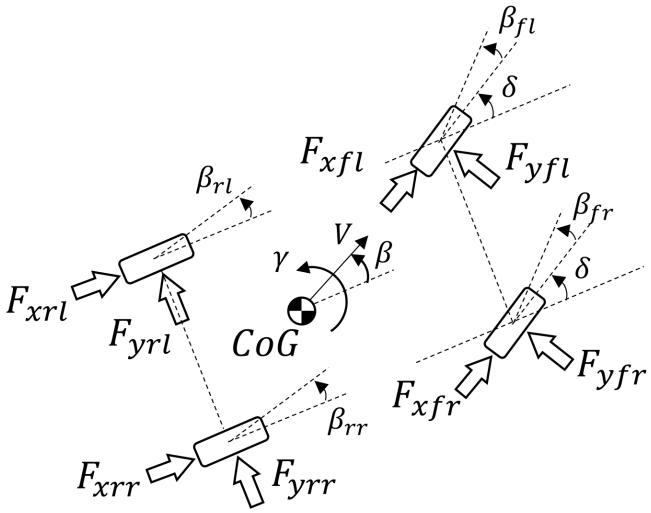


Figure 2.4: Four Wheel Model

The lateral and longitudinal forces generated by each tire are given by

$$F_{y**} = -(e \cdot F_{z**} + f)\beta_{**} \tag{2.13}$$

$$F_{x**} = T_{**}/R \tag{2.14}$$

where, ** is replaced by the indices fl (front-left), fr (front-right), rl (rear-left), and rr (rear-right) indicating each tire, respectively. e, f are parameters for expressing cornering stiffness as a function of F_{z**}, and R is a tire radius.

The state vector of the MPC controller for the vehicle is

$$x = [{}^F p_y, \theta_F, V, \gamma, \beta, a_x, a_y]^\top, \tag{2.15}$$

where the state equations for the state variables are given as follows:

$$\mathrm{d}{}^F p_y/\mathrm{d}t = V\cos\beta\sin\theta_F + V\sin\beta\cos\theta_F, \tag{2.16}$$

$$\mathrm{d}\theta_F/\mathrm{d}t = \gamma - {}^F\dot{p}_x\rho_{ref}, \tag{2.17}$$

$$\mathrm{d}{}^F p_x/\mathrm{d}t = \frac{V\cos\beta\cos\theta_F - V\sin\beta\sin\theta_F}{1-\rho_{ref}{}^F p_y}, \tag{2.18}$$

$$\mathrm{d}V/\mathrm{d}t = a_y\sin\beta + a_x\cos\beta, \tag{2.19}$$

$$\mathrm{d}\beta/\mathrm{d}t = (a_y\cos\beta - a_x\sin\beta)/V - \gamma, \tag{2.20}$$

$$\begin{aligned}\mathrm{d}\gamma/\mathrm{d}t = {} & (l_f((F_{xfl}+F_{xfr})\sin\delta) + (F_{yfl}+F_{yfr})\cos\delta) \\ & + d_f(F_{xfr}-F_{xfl})\cos\delta)/2 + d_f(F_{yfl}-F_{yfr})\sin\delta/2 \\ & - l_r(F_{yrl}+F_{yrr}) + d_r(F_{xrr}-F_{xrl})/2)/I_{zz},\end{aligned} \tag{2.21}$$

Note that the case of zero velocity cannot be calculated due to the 1/V term in Eq. (2.20). Since it is impossible to obtain all the state variables at once from the equations, a_x and a_y are taken from the previously given values. The derivatives of a_x and a_y are obtained by assuming a first-order delay system. It is unlikely to cause practical problems if the time delay T_{delay} is small enough. $T_{\text{delay}} = 0.05$s is used in this study.

$$\begin{aligned}\tilde{a}_x = \frac{1}{m}(&-F_{yfl}\sin\delta - F_{yfr}\sin\delta \\ &+ F_{xfl}\cos\delta + F_{xfr}\cos\delta + F_{xrl} + F_{xrr})\end{aligned} \tag{2.22}$$

$$\begin{aligned}\tilde{a}_y = \frac{1}{m}(&F_{yfl}\cos\delta + F_{yrl}\cos\delta \\ &+ F_{yrl} + F_{yrr} + F_{xfl}\sin\delta + F_{xfr}\sin\delta)\end{aligned} \tag{2.23}$$

$$\frac{\mathrm{d}a_x}{\mathrm{d}t} = \frac{1}{T_{\text{delay}}}(\tilde{a}_x - a_x) \tag{2.24}$$

$$\frac{\mathrm{d}a_y}{\mathrm{d}t} = \frac{1}{T_{\text{delay}}}(\tilde{a}_y - a_y) \tag{2.25}$$

This study uses time-state control [57] [58] to simply describe the location of obstacles. Let ξ be the state variable, then the conversion of the time-axis state control is expressed by the following chain rule.

$$\frac{\mathrm{d}\xi}{\mathrm{d}^F p_x} = \frac{\mathrm{d}\xi}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}^F p_x} = \frac{\mathrm{d}\xi}{\mathrm{d}t} \Big/ \frac{\mathrm{d}^F p_x}{\mathrm{d}t} \tag{2.26}$$

2.5.2 Formulation of Optimization Problem

Since the torque inputs for all four in-wheel motors and the front wheel steering angle can be controlled independently, a 5 DoF vehicle is over-actuated and redundant. This section formulates the optimization problem for MPC to determine the distribution of the driving force for each tire and steering angle directly. In the case of applying the controller to a real system, it is necessary for the in-wheel motors to follow the given torque commands.

The optimization problem solved in each control interval in MPC is formulated as follows, putting ${}^F p_x$ with s for the convenience of visibility,

Given

$$\begin{aligned}&\hat{x}(0 \mid s) = x(s) = [{}^F p_y,\, \theta_F,\, V,\, \gamma,\, \beta,\, a_x,\, a_y]^\top, \\ &x_{\text{ref}} = [0,\, 0,\, V_{\text{ref}},\, 0,\, 0,\, 0,\, 0]^\top, \\ &S_f,\ Q,\ R,\ R',\ C_x,\ C_y,\ C_r,\ N_{\text{obj}},\ W_{\text{obj}},\ \delta_{\max},\ T_{\max}\end{aligned}$$

Find

$$
\begin{aligned}
&\hat{x}(k \mid s), \quad \forall k \in \{1, \ldots, N\}, \\
&\hat{u}(k \mid s) = \left[\hat{\delta}(k \mid s), \hat{T}_{fl}(k \mid s), \hat{T}_{fr}(k \mid s), \hat{T}_{rl}(k \mid s), \hat{T}_{rr}(k \mid s)\right]^T, \\
&\forall k \in \{0, \ldots, N-1\}
\end{aligned} \tag{2.27}
$$

Objective: Minimize Cost Function J

$$J = \Phi(\hat{x}(N \mid s)) + \sum_{k=0}^{N-1} L(\hat{x}(k \mid s), \hat{u}(k \mid s))\,\Delta s, \tag{2.28}$$

$$\Phi(\hat{x}(N \mid s)) = \tfrac{1}{2}(\hat{x}(N \mid s) - x_{\text{ref}})^\top S_f (\hat{x}(N \mid s) - x_{\text{ref}}), \tag{2.29}$$

$$\begin{aligned}
L(\hat{x}(k \mid s)) = {} & \tfrac{1}{2}(\hat{x}(k \mid s) - x_{\text{ref}})^\top Q (\hat{x}(k \mid s) - x_{\text{ref}}) \\
& + \hat{u}(k \mid s)^\top R \hat{u}(k \mid s) \\
& + (\hat{u}(k \mid s) - \hat{u}(k-1 \mid s))^\top R' (\hat{u}(k \mid s) - \hat{u}(k-1 \mid s)) \\
& + P(\hat{x}(k \mid s)),
\end{aligned} \tag{2.30}$$

$$P(\hat{x}(k \mid s)) = \sum_{j=1}^{N_{\text{obj}}} \frac{W_{\text{obj}}}{({}^F p_x(k \mid s) - C_{jx})^2 + ({}^F p_y(k \mid s) - C_{jy})^2} \tag{2.31}$$

Subject to

$$\hat{x}(k+1 \mid s) = \hat{x}(k \mid s) + \Delta s \cdot \frac{dx}{ds} \tag{2.32}$$

$$|\delta| < \delta_{\max} \tag{2.33}$$

$$|T_{fl}|, |T_{fr}|, |T_{rl}|, |T_{rr}| < T_{\max} \tag{2.34}$$

$$C_r^2 - \left((p_{x**} - C_{jx})^2 + (p_{y**} - C_{jy})^2\right) < 0 \tag{2.35}$$

where $x(s)$ is a state vector defined by Eq. (2.15), S_f, Q, R and R' are the weight matrices. J is the cost function to be minimized, which consists of stage cost L, terminal cost Φ, and potential field penalty P. See Table.2.1 for the definitions of the variables.

Hard constraints (2.35) work to guarantee that the vehicle never collides with obstacles. Moreover, the artificial potential field (APF) [59] added to the stage cost allow the vehicle to drive keeping certain distances away from obstacles. W_{obj} in Eq. (2.31) is a parameter that determines the level of influence of obstacles, and the value shown in Table.2.2 is used in this study.

Note that ** indicates every four points of the vehicle body rectangle. It is assumed that the vehicle does not collide if all of the four points are outside the circular areas of the obstacles.

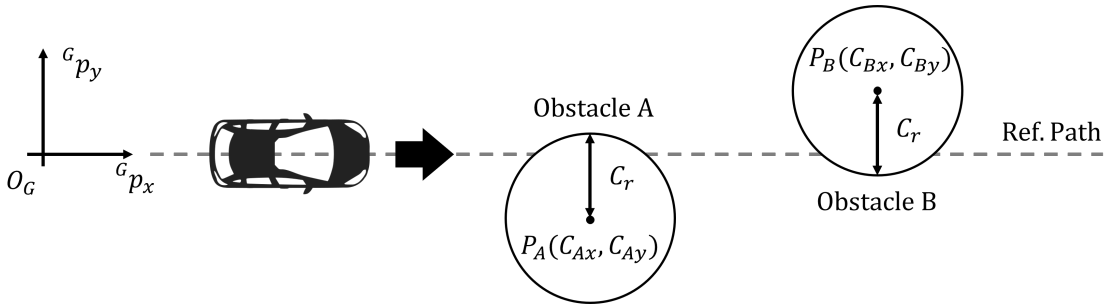


Figure 2.5: Obstacle Avoidance Situation

2.5.3 Simulation Results of Obstacle Avoidance Scenario

Here, two types of simulation scenarios are presented. The first one (Section. 2.5.3.2) is an obstacle avoidance task using both steer and each four tire torque, and the other one (Section. 2.5.3.3) is emergency avoidance behavior in the case of a steering failure. The full vehicle model (Section. 2.5.1) in the Global Frame is used as a simulator with the assumption that the time delay is 0 s in following the reference value of torque generation.

2.5.3.1 Driving Scenario

The target task is set as obstacle avoidance as shown in Fig.2.5 in the simulation. The vehicle tries to follow the given straight reference path as closely as possible while avoiding two circular obstacles. The radius C_r and positions of the two obstacles (C_{*x}, C_{*y}) are defined as follows.

$$
\begin{aligned}
C_r &= 2.0\ [\mathrm{m}] \\
P_A(C_{Ax}, C_{Ay}) &= (10\ [\mathrm{m}], -1.5\ [\mathrm{m}]) \\
P_B(C_{Bx}, C_{By}) &= (25\ [\mathrm{m}], 1.5\ [\mathrm{m}])
\end{aligned}
$$

The initial values of the state variables is,
$x(0) = [{}^{F}p_y, \theta_F, V, \gamma, \beta, a_x, a_y] = [0.5, 0.0, 3.0, 0.0, 0.0, 0.0, 0.0]^{\top}$. The parameters in Table.2.2 are used in both cases with/without the steering input. The simulations were done using the Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, RAM 24.0 GB. The optimization problem was solved using PANOC [50], a fast algorithm suitable for real-time nonlinear control applications. The control interval was set to 0.05 s, and as shown in Fig.2.6, both scenarios achieved real-time computation with sufficient margin.

Table 2.2: Common Parameters in Simulations

Control interval	0.05 [s]
Δt	0.05 [m]
N	50 [step]
V_{ref}	6.95 [m/s] (25 [km/h])
W_{obj}	45.0 [-]

Table 2.3: Parameters in Simulation With Steering Input

S_f	diag $[0, 0, 1, 10^{-3}, 10^{-7}, 10^{-3}, 10^{-3}]$
Q	diag $[7.5, 0.5, 1, 10^{-8}, 10^{-7}, 10^{-3}, 10^{-3}]$
R	diag $[0.0, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}]$
R'	diag $[0.1, 10^{-5}, 10^{-5}, 10^{-5}, 10^{-5}]$
δ_{max}	30 [deg]
T_{max}	1000[Nm]

Table 2.4: Parameters in Simulation Without Steering Input

S_f	diag $[0, 0, 1, 10^{-3}, 10^{-7}, 10^{-3}, 10^{-3}]$
Q	diag $[7.5, 0.5, 1, 10^{-8}, 10^{-7}, 10^{-3}, 10^{-3}]$
R	diag $[0, 5 \times 10^{-8}, 5 \times 10^{-8}, 5 \times 10^{-8}, 5 \times 10^{-8}]$
R'	diag $[0.1, 10^{-4}, 10^{-4}, 10^{-4}, 10^{-4}]$
δ_{max}	0 [deg]
T_{max}	1000[Nm]

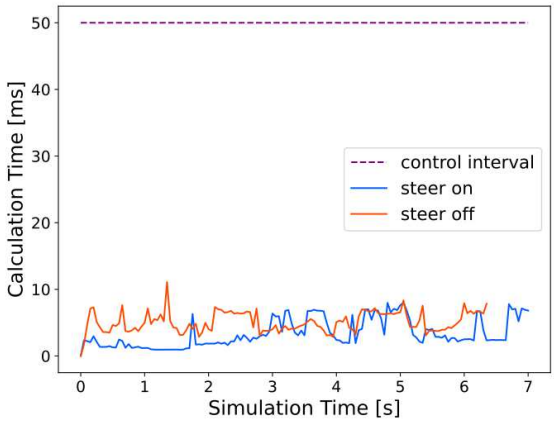


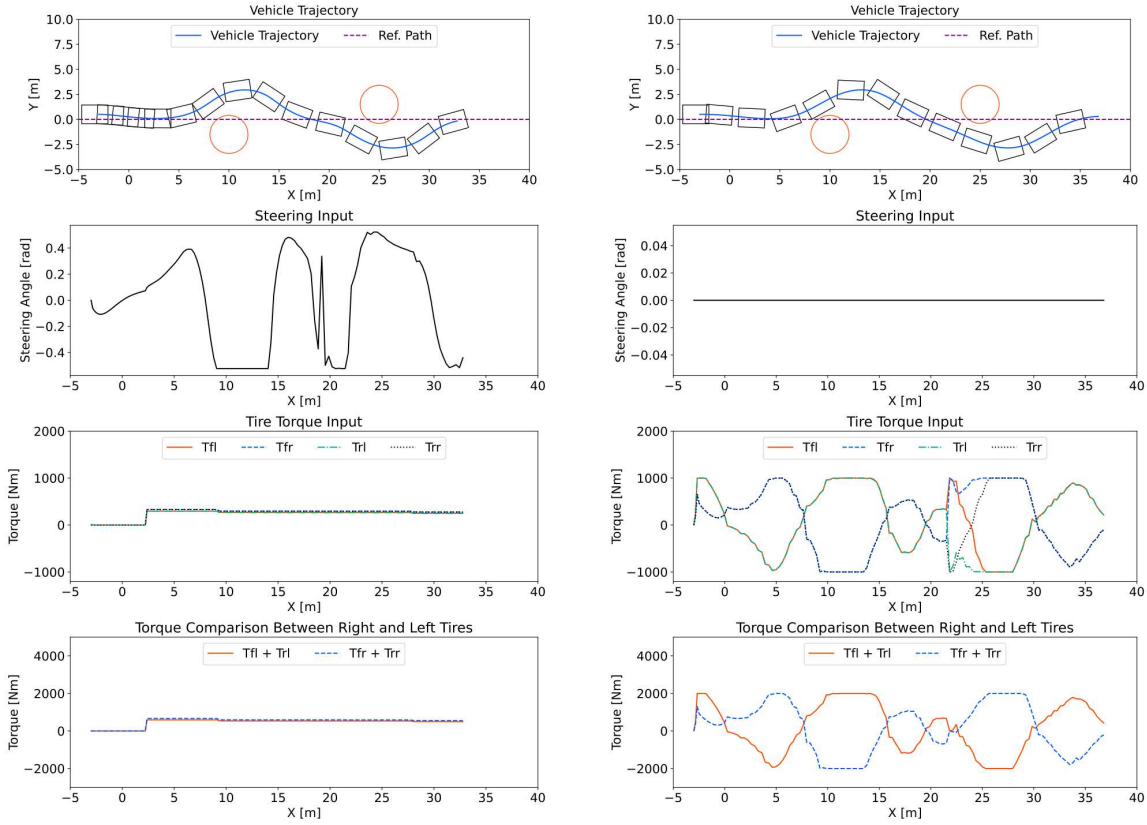
Figure 2.6: Computation Time

2.5.3.2 Obstacle avoidance simulation with steering input

First, obstacle avoidance with steering input is tested using the parameters listed in Table.2.3. In this scenario, the vehicle successfully avoided the obstacles. See Fig. 2.7a to check the simulation result. Basically, the vehicle turned itself using steering input. The torque inputs were mainly used to accelerate the car to follow the reference velocity. This is because steering input has a greater influence on the vehicle motion, and is more effective in reducing the penalty in terms of the cost function. However, right tire torques are slightly larger than left ones where the vehicle is right in front of the obstacle A. This means that torque distribution also contributes to generate yaw-moment of the vehicle, especially in severe situations where it is difficult to make a turn by steering input alone.

2.5.3.3 Obstacle avoidance simulation without steering input

Next, obstacle avoidance without steering input is tested setting parameters in Table.2.4. The vehicle was successful in avoiding collisions even though steering input is always zero as shown in Fig.2.7b. Comparing the results between Fig.2.7a and Fig.2.7b, much higher torque input was needed to avoid collision without steering input. The vehicle's direction change through torque distribution represents a natural control approach. The system demonstrates effective utilization of individual tire torques. The results indicate successful obstacle avoidance through torque distribution control alone. Since the turning performance is reduced when turning without steer, the closest distance from the obstacle is smaller than in the case with normal steering input. The bottom figure in Fig. 2.7b is helpful to grasp how the torque allocation contributes to the vehicle motion. If greater torque is generated by the left tires, the vehicle receives a yaw moment causing it to turn right, and the opposite is also true (See Fig.2.8.) The larger the difference in torque generated by the left and right tires, the larger the rotational moment given to the car.



(a) With Steering Input

(b) Without Steering Input

Figure 2.7: Obstacle Avoidance Performance Comparison

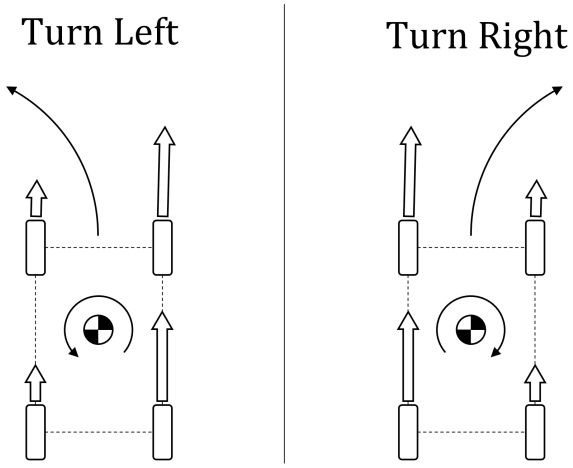


Figure 2.8: Vehicle Turn with Torque Distributions

2.6 Discussion and Insights on Q2-α, Q2-β

This chapter first provided an overview of Model Predictive Control (MPC), followed by a case study involving its application to a 5 DoF vehicle control system. In the application to 5 DoF vehicle control, obstacle avoidance was achieved under both normal and steering-failure conditions by maintaining the same control framework. In particular, during steering failure, turning maneuvers were accomplished by independently controlling the in-wheel motors installed on each wheel of the vehicle. This case demonstrates an example of how MPC can exploit the performance potential of high-degree-of-freedom vehicles.

However, the universal applicability of this standard strategy to all types of vehicles remains questionable. This issue should be addressed from both the prediction and optimization perspectives.

Regarding prediction modeling, vehicle models based on Newton's laws require the specification of friction parameters (c.f. Eq. (2.14)) and cannot accommodate stopping behavior (c.f. Eq. (2.20)), which imposes significant limitations on their practical applicability. The development of simplified models that can achieve comparable control performance is therefore desirable for practical implementation. Furthermore, it is essential to systematically compare and evaluate the limitations of such simplified models. A discussion on the preparation and development of practical prediction models is provided in Chapter 3, leading an insight into the Research Question Q2-α.

With respect to optimization, the adopted approach involved minimizing a scalarized objective function, constructed by weighting terms corresponding to multiple tasks, using a gradient-based method. This approach presents several challenges. Tuning of the balancing weights is inherently difficult for controller designers, and this difficulty increases as the number of tasks grows, resulting in scalability issues. Furthermore, solutions to non-convex optimization problems exhibit significant sensitivity to the initial guess. Given the characteristic of high-degree-of-freedom vehicles to allow multiple strategies for achieving a given objective, non-convexity in the optimization problem is inevitable. Chapters 4 and 5 discuss approaches to address these challenges, and an insight to the Research Question Q2-β is provided.

Chapter 3

Comparison and Improvement of Prediction Models for 2 DoF Vehicle Path-Tracking

3.1 Chapter Overview and Contribution to Q2-α

This chapter investigates the appropriate design of prediction models for model predictive control (MPC) in the context of 2 DoF vehicle path-tracking. Specifically, the study formulates a path-tracking control problem for a 2 DoF vehicle using MPC and evaluates tracking performance by switching the prediction model used in MPC for each experiment. In addition to three existing prediction models, a novel model is proposed to address the limitations identified in a conventional vehicle model. The comparative analysis demonstrates that the path-tracking performance of each prediction model varies according to the vehicle's driving speed. The results highlight the importance of selecting prediction models based on the driving speed range, and suggest that employing the most detailed model is not always the best choice.

3.2 Background and Related Work

Path tracking is a crucial function for autonomous vehicles, and Model Predictive Control (MPC) has emerged as a promising approach that has been extensively studied. The 2 DoF vehicle represents a typical configuration for passenger cars, and has already been successfully implemented for path tracking in real-world autonomous driving demonstrations [61].

The prediction model, which forecasts the future state of the controlled vehicle, significantly influences MPC performance. Without accurate future predictions, selecting the optimal action becomes difficult. Excessively complex models compromise practical implementation through increased computational complexity in optimization and the requirement for numerous physical parameters that may be challenging to measure or estimate accurately.

Several candidate vehicle prediction models exist for MPC-based path tracking, but their appropriate selection criteria have not been thoroughly validated. The candidate models include the Kinematic Ackermann Model (KAM), Kinematic Bicycle Model (KBM), and Dynamic Bicycle Model (DBM) [36, 37]. These models represent relatively simple formulations of vehicle motion, each based on different underlying assumptions. Since these assumptions are significantly influenced by vehicle speed, the most suitable model is expected to vary depending on the driving speed range [62].

This chapter focuses on two main aspects of vehicle prediction models for MPC-based path tracking: (1) performance evaluation of existing models and (2) proposing a new model that improves a weakness of an existing model. Evaluation results demonstrate that driving speed is a significant factor in selecting the appropriate model. In addition, the proposed Dynamic Bicycle Model improved in Low-speed range(DBM-L) further extends DBM's applicability to low-speed driving.

3.3 2 DoF Vehicle Prediction Models for Path-Tracking Control

3.3.1 Definitions of Variables

Before introducing the target vehicle models, the variables and parameters used to describe the models are defined in Table 3.1. The coordinate frames used in vehicle models are defined in Fig.2.1 in Chapter 2, which includes the Global Frame, Base Frame, and Frenet-Serret Frame.

3.3.2 Kinematic Ackermann Model (KAM)

The first model tested in this study is Kinematic Ackermann Model (KAM). In conditions where the car turns along the stable circle at low speed, the Ackermann Geometry [37] (Fig. 3.1) exists in the Global Frame assuming no wheel slipping in the lateral direction.

Table 3.1: Definition of Variables

Variable	Definition	Value [Unit]
β	Slip Angle of Vehicle Body	- [rad]
β_f, β_r	Slip Angles of Front, Rear Wheel	- [rad]
θ_F	Yaw Angle between the Ref. Path	- [rad]
γ_{yaw}	Yaw Rate of Vehicle Body	- [rad/s]
δ	Front Wheel Angle	- [rad]
ρ_{ref}	Curvature of Ref. Path	- [m^{-1}]
θ_{ref}	Yaw Angle of the Ref. in the Global Frame	- [rad]
${}^{B}v_y$	Vehicle Lateral Velocity in the Base Frame	- [m/s]
V	Vehicle Velocity	- [m/s]
a	Vehicle Acceleration	- [$\mathrm{m/s}^2$]
l_f	Distance from CG to the Front Axle	1.04 [m]
l_r	Distance from CG to the Rear Axle	1.56 [m]
m	Mass of Vehicle Body	1110 [kg]
I_z	Vehicle Moment of Inertia	1343 [kgm^2]
K_f	Cornering Stiffness of Front Wheel	56023[N/rad]
K_r	Cornering Stiffness of Rear Wheel	37942[N/rad]

CG: center of gravity

From geometrical relationship,

$$\rho = (l_f + l_r)/\delta, \tag{3.1a}$$

$$\gamma_{yaw} = V/\rho = \delta V/(l_f + l_r), \tag{3.1b}$$

$$\beta = l_r/\rho = \delta l_r/(l_f + l_r). \tag{3.1c}$$

For path tracking, KAM is transformed into Frenet-Serret frame as follows:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} x_{\mathrm{KAM}} &= \frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} {}^{F}p_y & \theta_F & {}^{F}p_x & V \end{bmatrix}^{\top} \\ &= \begin{bmatrix} V\theta_F + \dfrac{l_r}{(l_f + l_r)} V\delta \\ -\rho_{ref} V + \dfrac{1}{(l_f + l_r)} V\delta \\ V \\ a \end{bmatrix}. \end{aligned} \tag{3.2}$$

Note that the simple point mass model is used to express the simple longitudinal dynamics of the car in Eq. (3.2).

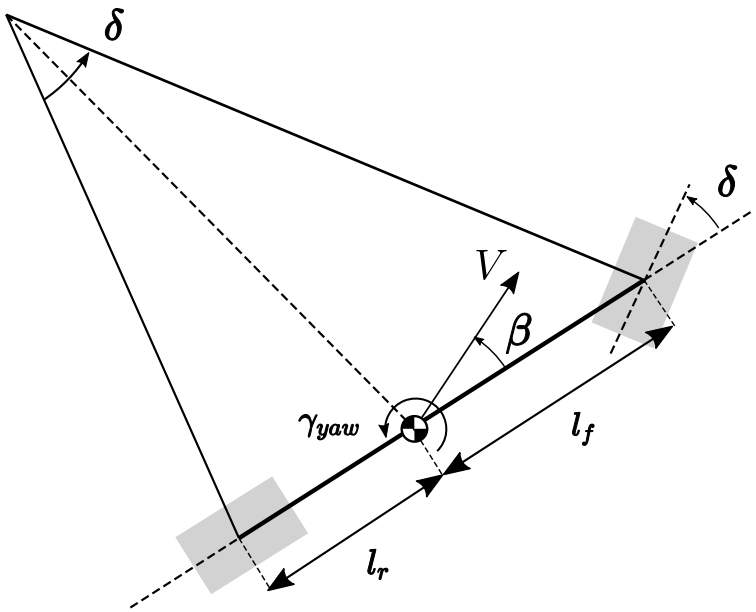


Figure 3.1: Kinematic Ackermann Model

3.3.3 Kinematic Bicycle Model (KBM)

The motion of the car at low speed can be described as Kinematic Bicycle Model(Fig. 3.2) in an inertial frame. Driving speed in the Global Frame can be written as follows:

$$ {}^{G}\dot{p}_x = V\cos(\theta_G+\beta), \;\; {}^{G}\dot{p}_y = V\sin(\theta_G+\beta), \tag{3.3a} $$

$$ \dot{\theta_G} = \frac{V}{l_r}\sin\beta, \;\; \beta = \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan\delta\right). \tag{3.3b} $$

For path-tracking control purpose, Eq. (3.3) is described in Frenet-Serret frame as follows:

$$ \begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t}x_{\mathrm{KBM}} &= \frac{\mathrm{d}}{\mathrm{d}t}\left[\begin{array}{cccc} {}^{F}p_y & \theta_F & {}^{F}p_x & V \end{array}\right]^{\top} \\ &= \left[\begin{array}{c} V\sin(\theta_F+\beta) \\ \dfrac{V}{l_r}\sin\beta - \dfrac{V\rho_{ref}}{1-\rho_{ref}\,{}^{F}p_y}\cos(\theta_F+\beta) \\ \dfrac{V}{1-\rho_{ref}\,{}^{F}p_y}\cos(\theta_F+\beta) \\ a \end{array}\right]. \end{aligned} \tag{3.4} $$

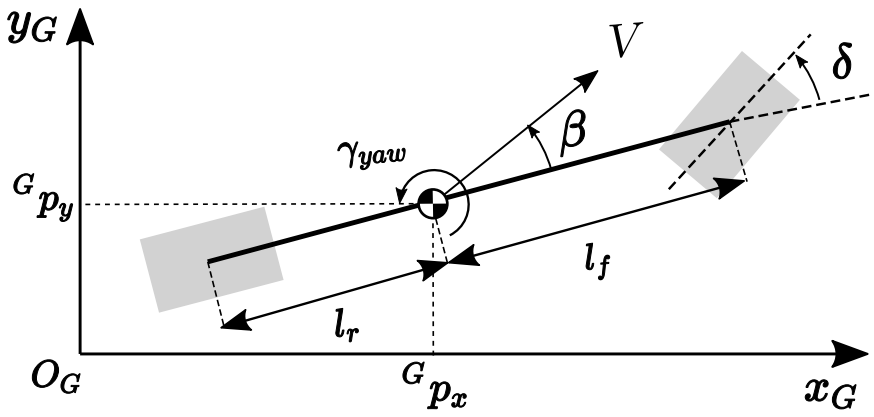


Figure 3.2: Kinematic Bicycle Model

3.3.4 Dynamic Bicycle Model (DBM)

Dynamic Bicycle Model (DBM) is a well known lateral vehicle dynamics model in higher speed range which considers the effect of wheel slip angles. The equations of DBM in the base coordinate can be written as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} {}^B v_y \\ \gamma_{yaw} \end{bmatrix} = \begin{bmatrix} -\frac{a_{11}}{V} & \frac{a_{12}}{V} - V \\ -\frac{a_{21}}{V} & \frac{a_{22}}{V} \end{bmatrix} \begin{bmatrix} {}^B v_y \\ \gamma_{yaw} \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}. \tag{3.5}$$

For path tracking, DBM can be described in Frenet-Serret coordinate as follows:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} x_{\mathrm{DBM}} &= \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} {}^F p_y & {}^F \dot{p}_y & \theta_F & \dot{\theta}_F & {}^F p_x & V \end{bmatrix}^\top \\ &= \begin{bmatrix} {}^F \dot{p}_y \\ -\frac{a_{11}}{V} {}^F \dot{p}_y + (a_{11}+a)\theta_F + \frac{a_{12}}{V}\dot{\theta}_F \\ \dot{\theta}_F \\ -\frac{a_{21}}{V} {}^F \dot{p}_y + a_{21}\theta_F + \frac{a_{22}}{V}\dot{\theta}_F \\ V \\ a \end{bmatrix} \\ &\quad + E^\top \rho_{ref} + B^\top \delta, \end{aligned} \tag{3.6}$$

where

$$a_{11} = \frac{2(K_f + K_r)}{m}, \quad a_{12} = -\frac{2(K_f l_f - K_r l_r)}{m}, \tag{3.7}$$

$$a_{21} = \frac{2(K_f l_f - K_r l_r)}{I_z}, \; a_{22} = -\frac{2(K_f l_f^2 + K_r l_r^2)}{I_z}, \tag{3.8}$$

$$b_1 = \frac{2K_f}{m}, \;\; b_2 = \frac{2K_f l_f}{I_z}, \tag{3.9}$$

$$E = \left[0, \rho_{ref}(a_{12} - V^2), 0, a_{22} - a, 0, 0\right]^\top, \tag{3.10}$$

$$B = [0, b_1, 0, b_2 - a, 0, 0]^\top. \tag{3.11}$$

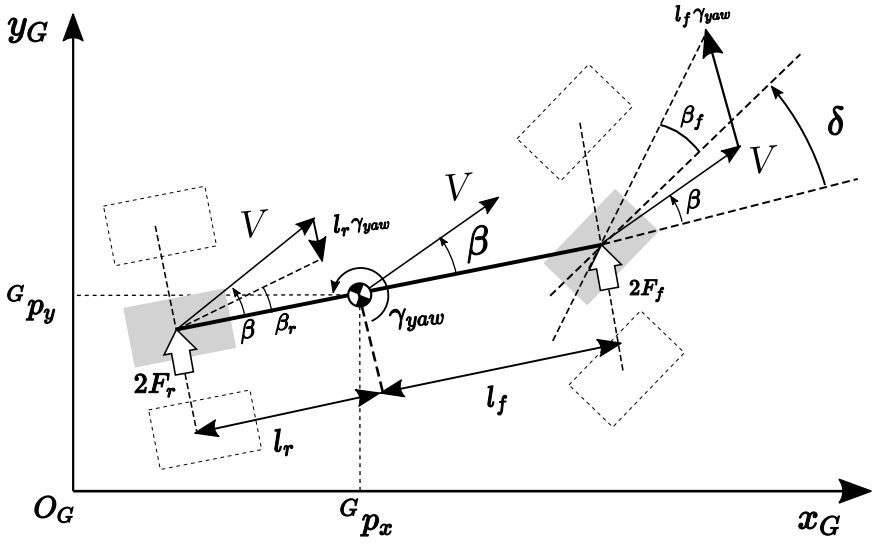


Figure 3.3: Dynamic Bicycle Model

3.3.5 Dynamic Bicycle Model improved in Low-speed range (DBM-L)

DBM suffers from inapplicability when the vehicle speed V is close to 0 due to singularity of the term $1/V$ in its formulation.

Therefore, we approximate the $1/V$ using soft normalization function [63] as Eq. (3.12), and define DBM improved in Low-speed range (DBM-L) as Eq. (3.13) in order to improve its behavior at extremely low-speed range. Fig.3.4 shows the visualization of V_{inv} compared with $1/V$.

$$V_{\text{inv}} = \frac{1}{V + \alpha \ln(1 + \exp(-2\alpha V))}, \tag{3.12}$$

where the constant $\alpha = 1.0$ is used in this study. According to Eq. (3.12), V_{inv} equals $1/(\alpha \ln 2)$ when $V = 0$. Therefore, as shown in Fig. 3.4, adjusting the parameter α changes the intercept value. The parameter α must be positive, and decreasing its value causes V_{inv} to more closely approximate $1/V$.

Consequently, DBM-L can be described as follows:

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t} x_{\text{DBM-L}} &= \frac{\mathrm{d}}{\mathrm{d}t} \left[\begin{array}{ccccccc} {}^F p_y & {}^F \dot{p}_y & \theta_F & \dot{\theta}_F & {}^F p_x & V \end{array} \right]^\top \\
&= \begin{bmatrix} {}^F \dot{p}_y \\ -a_{11} V_{\text{inv}} {}^F \dot{p}_y + (a_{11} + a)\theta_F + a_{12} V_{\text{inv}} \dot{\theta}_F \\ \dot{\theta}_F \\ -a_{21} V_{\text{inv}} {}^F \dot{p}_y + a_{21} \theta_F + a_{22} V_{\text{inv}} \dot{\theta}_F \\ V \\ a \end{bmatrix} \\
&+ E^\top \rho_{ref} + B^\top \delta.
\end{aligned} \tag{3.13}$$

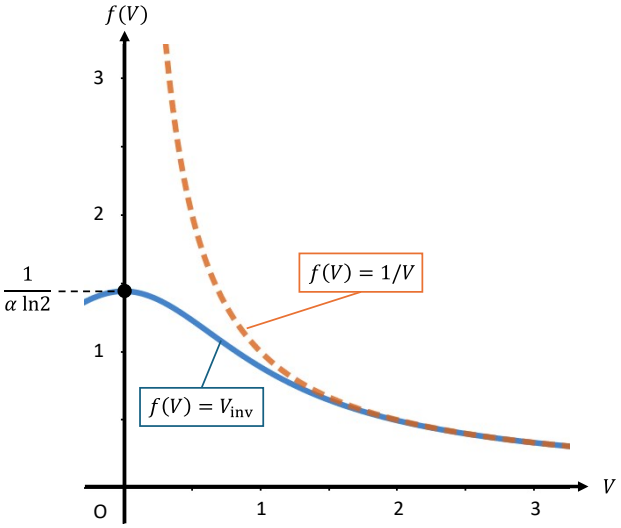


Figure 3.4: Visualization of V_{inv} compared with $1/V$

3.4 Formulation of MPC for Path-Tracking Control

This study examines how different prediction models affect path-tracking performance in model predictive control (MPC). The MPC framework incorporates the prediction models developed in Section 3.3, and their applicability is assessed through path-tracking performance evaluation.

Figure 3.5 shows an architecture overview of the model predictive path-tracking controller. The normal passenger car in Carsim software (Mechanical Simulation Corp.) including a high-fidelity vehicle dynamics model is used as the control target. The optimization problem is solved using the C/GMRES [48] method, an extremely fast gradient-based approach suitable for real-time execution.

The optimization problem solved in each control interval is formulated as follows:

Given

$$\begin{aligned} &\hat{x}(0 \mid t) = x(t), \quad x_{\text{ref}} \in \mathbb{R}^n, \quad N \in \mathbb{N}, \quad \Delta t > 0, \\ &S_f \in \mathbb{R}^{n \times n}, \quad Q \in \mathbb{R}^{n \times n}, \quad R \in \mathbb{R}^{m \times m} \end{aligned} \tag{3.14}$$

Find

$$\begin{aligned} &\hat{x}(k \mid t) \in \mathbb{R}^n, \quad \forall k \in \{1, \ldots, N\}, \\ &\hat{u}(k \mid t) = \begin{bmatrix} \hat{\delta}(k \mid t) \\ \hat{a}(k \mid t) \end{bmatrix} \in \mathbb{R}^m, \quad \forall k \in \{0, \ldots, N-1\} \end{aligned} \tag{3.15}$$

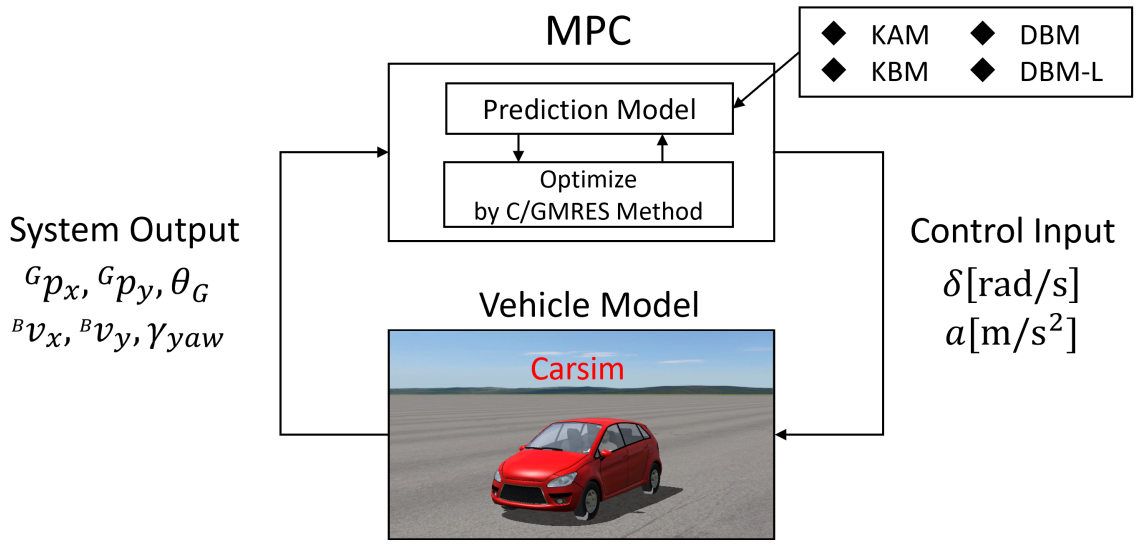


Figure 3.5: Path-tracking simulation architecture using MPC

Table 3.2: Parameters in Simulations

	KAM, KBM	DBM, DBM-L
Δt	0.01s	0.01s
N	100	100
S_f	diag[10, 10, 0, 10]	diag[10, 1, 10, 1, 0, 10]
Q	diag[1, 1, 0, 1]	diag[1, 0.1, 1, 0.1, 0, 1]
R	diag[1, 1]	diag[0.1, 1]

Objective: Minimize Cost Function J

$$J = \Phi\big(\hat{x}(N \mid t)\big) + \sum_{k=0}^{N-1} L\big(\hat{x}(k \mid t), \hat{u}(k \mid t)\big)\Delta t \tag{3.16}$$

$$\Phi\big(\hat{x}(N \mid t)\big) = \frac{1}{2}\big(\hat{x}(N \mid t) - x_{\text{ref}}\big)^{\top} S_f \big(\hat{x}(N \mid t) - x_{\text{ref}}\big), \tag{3.17}$$

$$L\big(\hat{x}(k \mid t), \hat{u}(k \mid t)\big) = \frac{1}{2}\big(\hat{x}(k \mid t) - x_{\text{ref}}\big)^{\top} Q \big(\hat{x}(k \mid t) - x_{\text{ref}}\big) + \hat{u}(k \mid t)^{\top} R \hat{u}(k \mid t) \tag{3.18}$$

Subject to

$$\hat{x}(k+1 \mid t) = \hat{x}(k \mid t) + \Delta t \cdot \frac{\mathrm{d}}{\mathrm{d}t} x_M, \quad \forall k \in \{0, \ldots, N-1\} \tag{3.19}$$

where $x(t)$ is a state vector, S_f, Q, R are weight matrices. The dimension of $x(t)$ and weight matrices differ depending on prediction models. The state equation $\frac{\mathrm{d}}{\mathrm{d}t} x_M$ (Eq. (2.32)) is also replaced depending on the target vehicle model for the evaluation.

Table 3.3: Driving speed and curvature for evaluation

v_{ref} [km/h]	5	10	20	30	40	50	60	80	100	120
R [m]	5	10	15	30	60	100	150	280	460	710

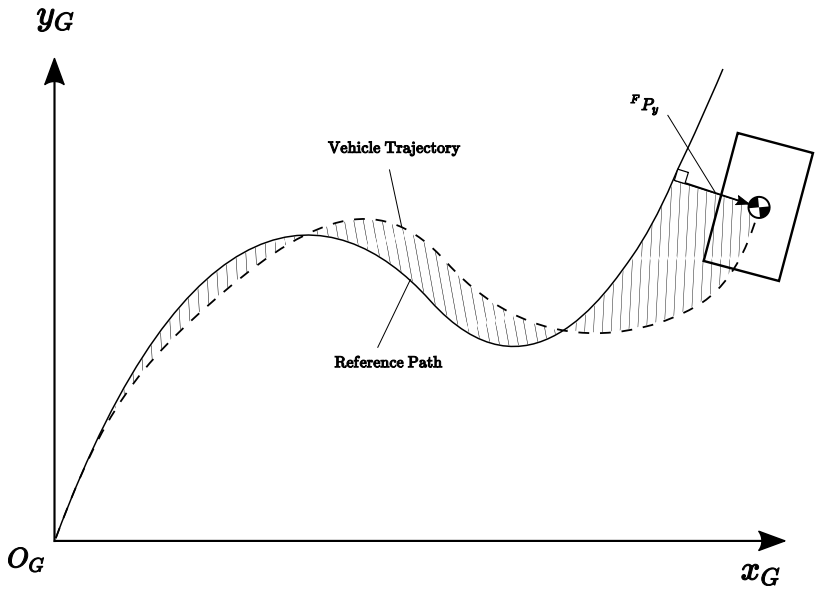


Figure 3.6: Visualization of tracking error definition

3.5 Evaluation

3.5.1 Evaluation metrics

In the path-tracking simulation on Carsim, a virtual car is controlled by MPC to follow the reference paths shown in Fig.3.7 and 3.8. One of the vehicle models prepared in Section 3.3.1 is used as a prediction model of MPC. The parameters of MPC are shown in Table 3.2, which are manually tuned to achieve a good tracking performance, and the parameters are different depending on the prediction model. Performance of path tracking is evaluated by mean and maximum values of lateral tracking error as shown in Eq. (3.20), where L is the total path length. The tracking error is defined as the area enclosed by the vehicle trajectory and the reference path, divided by the total path length, as illustrated in Fig. 3.6.

$$E_{\mathrm{ave}} = \frac{1}{L}\int_0^L |^F p_y|\, d^F p_x, \quad E_{\mathrm{max}} = \max|^{\mathrm{F}}\mathrm{p_y}|. \tag{3.20}$$

3.5.2 Driving conditions

In this study, 10 driving speed ranges listed in Table 3.3 (5[km/h] to 120[km/h]) are tested for each prediction model in order to cover the daily driving scene including low-speed

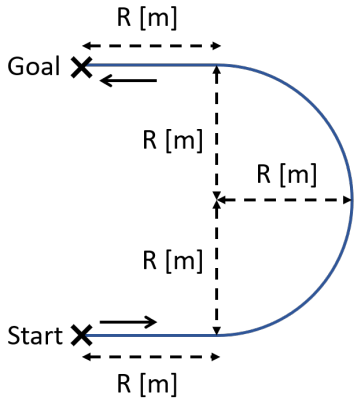


Figure 3.7: Oval-Shaped Path

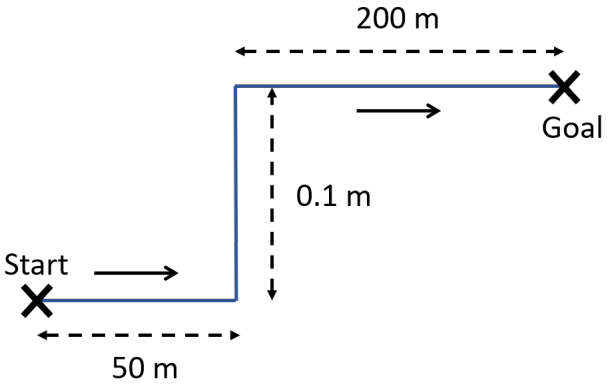


Figure 3.8: Step-Shaped Path

urban driving and highway driving.

Two test tracks are used to evaluate the path-tracking performance: (a). a half of oval track (Fig. 3.7), called ”oval-shaped path”, and (b). a rectangular wave-like driving path (Fig. 3.8), called ”step-shaped path”.

In (a), the curvature of the reference is changed depending on the driving speed as specified in Table 3.3 for the oval-shaped path. The applied curvature corresponding to a certain speed range is decided by referring to the standard of maximum curvature for road construction in Japan. In (b), the reference path is in a stepping manner to evaluate the step response performance of the path tracking.

3.5.3 Availability of models in various speed ranges

The experimental results are presented. Figures 3.9 to 3.12 show the control performances of the MPC path tracking using different prediction models in various driving speeds. Note that all depicted points show the cases the car was able to follow the path, and the points are lacking if the car failed to track the path.

The experimental results (Figs. 3.9 to 3.12) demonstrate that the prediction models show varying levels of tracking performance across different driving speeds. KAM and KBM maintain feasible tracking accuracy only below 40 km/h on the oval-shaped path, and below 50 km/h on the step-shaped path. In contrast, DBM and DBM-L can follow both of the reference paths in all tested driving speed from 5 to 120 km/h. The difference in these results stems from whether tire slip is considered in the model. KAM and KBM do not account for tire slip, which leads to significant discrepancies between predicted and actual vehicle behavior at high speeds where slip effects become more pronounced. This mismatch ultimately causes the path tracking to fail at higher velocities.

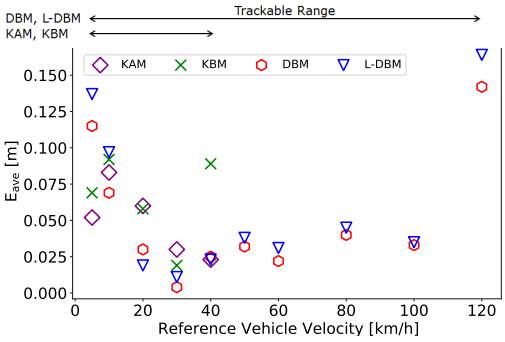


Figure 3.9: E_{ave} of Oval-Shaped Path [m]

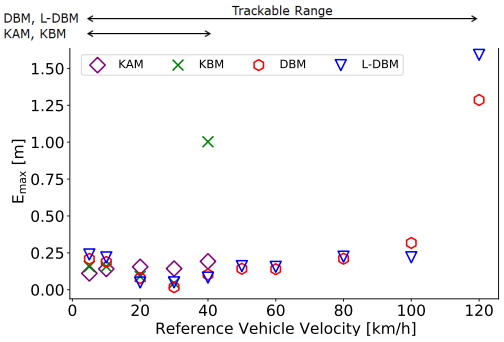


Figure 3.10: E_{max} of Oval-Shaped Path [m]

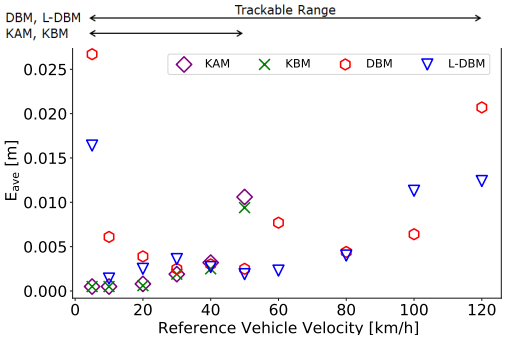


Figure 3.11: E_{ave} of Step-Shaped Path [m]

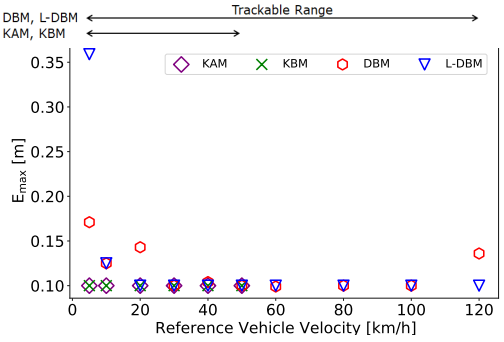


Figure 3.12: E_{max} of Step-Shaped Path [m]

Since DBM-L is approximately equivalent to DBM in the high-speed range, DBM-L is available in a wide speed range as well as the conventional DBM. It is found that DBM-L follows the tendency of the conventional DBM in the high-speed range from the results. Thanks to the soft normalization of the $1/V$ term in DBM-L, the model prevents the divergence of the computation in an extremely low-speed range successfully. As a result, DBM-L enables path tracking from standstill, which is not possible with the conventional DBM. Figure 3.13 shows the time profiles of the states in a oval-shaped path tracking with 5km/h starting from a stop state ($V = 0$ km/h).

As the summary of model comparison, Tab. 3.4 classified the evaluation of each model regarding the availability and accuracy. If a vehicle is required to run in the full vehicle speed range, the proposed DBM-L is the most applicable model among the tested models. Although DBM-L shows the availability in all driving speed range, the tracking accuracy of DBM and DBM-L are degraded in the low-speed range compared to KAM and KBM. This result implies that KAM and KBM are more suitable for low-speed driving, especially in the speed range of 0 to 30 km/h. Not only the tracking accuracy, KAM and KBM has advantages in the real-world application that the model does not need any friction coefficient or tire slip angle to be estimated.

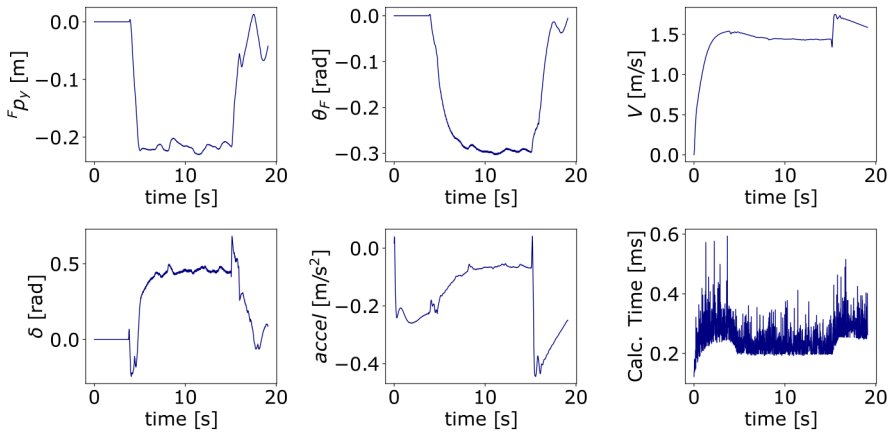


Figure 3.13: Simulation results using DBM-L model with initial velocity $V = 0$ km/h

Table 3.4: Evaluation of each vehicle model

Evaluation Items		KAM	KBM	DBM	DBM-L
Tracking Accuracy	Low speed	A	A	C	B
	Mid. speed	C	C	A	A
	High speed	D	D	B	B
Ave. Calculation Time [ms]		0.219	1.011	1.684	0.573
Num. of Physical Parameters		2	2	5	5

The evaluations in the table are defined as follows:
A : High-accuracy tracking is possible in the speed range,
B : Tracking is possible in the speed range,
C : Tracking is possible only in a part of the speed range,
D : Tracking is impossible in the speed range.

3.6 Discussion and Insights on Q2-α

This chapter evaluated how sensitive the path tracking performance of MPC is to selection of prediction models. This result provides insights on the research question **[Q2-α] How can practical prediction models be designed?** for MPC to exploit the performance potential of mobility systems.

In the low-speed driving scenarios, KAM and KBM exhibit superior performance despite their simplified structure that neglects tire slipping effects compared to DBM and DBM-L.

For high-speed scenarios, the use of DBM or DBM-L is essential since the performance of KAM and KBM deteriorates significantly. While DBM enables motion prediction considering friction, it has a weakness of prediction failure at zero vehicle speed. DBM-L achieves improvement by mathematical modification to suppress unstable prediction at zero vehicle speed without compromising the original performance of DBM. In terms of covering a wide speed range, this newly proposed model has the highest applicability.

These results suggest that selecting a detailed prediction model (DBM, DBM-L) is not always the best choice, and the selection of the prediction model should be made considering its compatibility with the target application. Since vehicle speed significantly influences tracking performance, prediction models should be chosen based on the target speed range of operation. For autonomous vehicles designed to operate at low speeds on public roads, using KAM or KBM not only provides satisfactory path tracking performance but also offers practical advantages by eliminating the need to determine vehicle mass and tire friction parameters. These insights directly address **[Q2-α]**, confirming that prediction model design for MPC should be tailored to the vehicle's operational context – favoring simpler models at low speeds and more detailed models at high speeds. While DBM-L does not achieve the highest accuracy at low speeds, it enables stable control across the full speed range, making it the most widely usable model among those tested.

Chapter 4

MPPI Controller as a Sampling-Based Optimizer for Multi-Objective Control via Weighted Scalarization in 8 DoF Vehicle Navigation

 Portions of this chapter first appeared in "Switching Sampling Space of Model Predictive Path-Integral Controller to Balance Efficiency and Safety in 4WIDS Vehicle Navigation," *Proceedings of the 2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3196–3203, 2024 [64].

4.1 Chapter Overview and Contribution to Q2-β

This chapter presents an application of Model Predictive Path Integral (MPPI) control to 8 DoF vehicle navigation. The flexible motion capabilities of 8 DoF vehicles offer significant potential for efficient navigation in dense obstacle environments. Achieving this potential requires sophisticated control methods that can simultaneously process real-time obstacle information for collision avoidance and maintain smooth steering operations. While previous research on 8 DoF vehicles has primarily addressed path following tasks, navigation in obstacle-rich environments remains challenging. This study proposes a novel approach using MPPI, a sampling-based Model Predictive Control (MPC) method, to achieve smooth and safe navigation in complex environments. The broad applicability of sampling-based optimization enables a practical framework that can handle real-world navigation scenarios,

including real-time obstacle detection from LiDAR point cloud data. This work represents the first implementation of MPPI for 8 DoF vehicles, demonstrating effective navigation performance. Regarding the relationship with Q2-β, this study explores the performance capabilities of the weighted scalarization approach in multi-objective control. It serves as a baseline for Chapter 5, which introduces enhanced optimization algorithms for improved multi-objective control performance.

4.2 Background and Related Work

Four-wheel independent drive and steering (4WIDS) vehicles has 8 DoF and has high mobility and flexibility. By enabling independent control of the drive torque and steering angle at each wheel, 4WIDS systems support holonomic movements such as rotating in place or moving diagonally. Compared to other holonomic platforms like omnidirectional or mecanum-wheel robots, 4WIDS vehicles can better handle rough terrain and maintain high-speed stability [65–67].

However, the enhanced mobility and flexibility of 4WIDS come at the cost of increased control complexity. The system's high-dimensional input space—8 DoF in total—makes direct control challenging. Moreover, to avoid mechanical failure, steering commands must be executed smoothly and in coordination across wheels.

Several approaches have been proposed to tackle the control problem of 4WIDS vehicles. A common strategy is to reduce the system's degrees of freedom by applying constraints. For instance, in [68], the front and rear steering angles are constrained to be mirror images, allowing the application of the pure pursuit algorithm for path tracking. While this method is computationally efficient and easy to implement, it limits the vehicle ' s ability to exploit its quick rotation and diagonal movement. Another class of methods adopts fuzzy logic to handle the entire 4WIDS input space. These methods design multiple driving modes that are switched based on context [69]. While effective in some target environments, such approaches require extensive domain expertise to define the modes and rules, and the rules are not easily extendable to novel or unstructured environments. Control designs based on Lyapunov stability theory [70] aim to ensure system stability under various

conditions. Stable path tracking can be achieved based on theoretical guarantees, but considering other tasks such as smooth steering and obstacle avoidance is challenging for the framework.

This study addresses these limitations by leveraging model predictive control (MPC), which is well-suited for handling high-dimensional, constrained systems. By optimizing control inputs over a prediction horizon, MPC offers a principled way to manage redundancy while satisfying physical constraints and promoting smooth vehicle behavior. Our approach aims to bridge the gap between maneuverability and practicality, advancing the capabilities of 4WIDS control systems.

4.3 Navigation Scenario Description

The goal of our system is to navigate a 4WIDS vehicle to a given goal while avoiding collision with surrounding obstacles (Fig. 4.1). We assume that a global planner generates a reference trajectory, which is a sequence of positions and orientations in a 2D plane. Therefore, the main focus of our study is to design a local planner that generates a vehicle motion to follow the reference trajectory while safely avoiding obstacles. The local planner sends an eight DoF vehicle command consisting of a wheel speed and a steering angle for each wheel to the vehicle actuators. Especially in our navigation environment where the obstacles are densely distributed, the high maneuverability of 4WIDS vehicles, such as small turning radius and diagonal movement, is an important factor for achieving smooth and efficient driving.

4.4 Modeling of 4WIDS Vehicle Motion

To achieve accurate vehicle control, it is important to understand the characteristics of vehicle motion. In this section, we formulate the model of 4WIDS vehicle behavior, mainly used for sampling future vehicle poses in our model predictive controller.

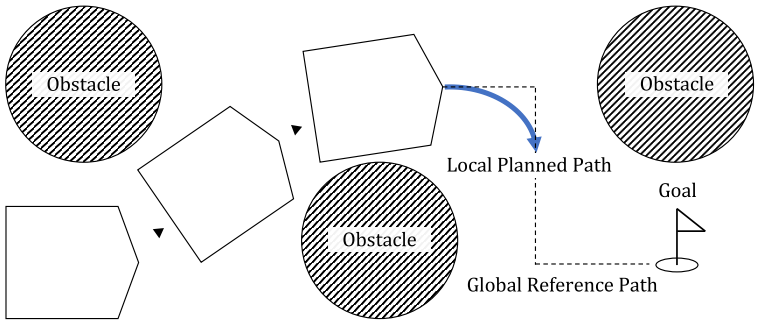


Figure 4.1: Navigation task for a 4WIDS vehicle: the goal is to follow a reference trajectory quickly and safely while avoiding dense obstacles.

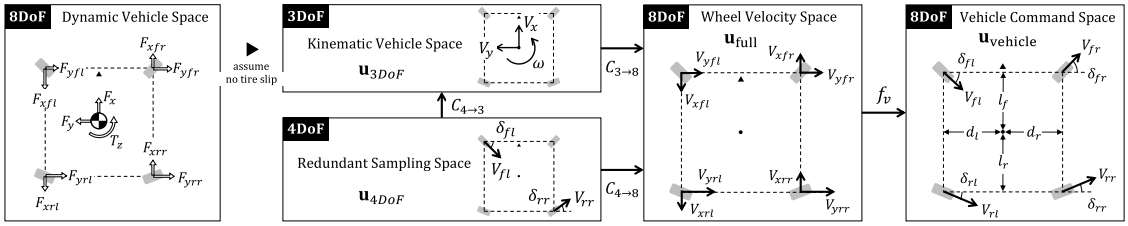


Figure 4.2: Relationship between spaces; Dynamics(8DoF) can be simplified to Kinematics(3DoF) with assuming no tire slip. Two types of sampling space are tested in this study; $\mathbf{u}_{3\text{DoF}}$ and $\mathbf{u}_{4\text{DoF}}$ to compare the navigation performance. Both spaces can be converted to the vehicle command space $\mathbf{u}_{\text{vehicle}}$ with conversion matrix $C_{n\to 8}(n \in \{3,4\})$ and nonlinear projection f_v.

4.4.1 4WIDS Vehicle Dynamics

Although it is a straightforward approach to model vehicle behavior using Newton's laws of motion, this study does not focus on this approach. To formulate the full vehicle dynamics, it is necessary to consider the eight tire forces generated by the four wheels, as shown in Fig.4.2. However, to obtain the forces, it is necessary to observe tire slip angles, tire slip ratios, and tire physical parameters, which are not easy to measure accurately in real-world applications [36]. Even if we have true dynamics and explore the full eight-dimensional input space, most of the solutions will include large tire slips, which cause unstable vehicle behavior. Therefore, predicting vehicle motion with full dynamics is not always practical, especially for real-world applications.

4.4.2 4WIDS Vehicle Kinematics

An efficient way to easily model the vehicle motion is to focus on vehicle kinematics [71]. Essentially, the difficulty of controlling 4WIDS vehicles comes from the redundancy of the control degrees of freedom, which means that there are multiple solutions to achieve a certain vehicle behavior. The kinematic formulation adds assumptions that the vehicle is moving at a constant velocity, and the tire slip is negligible. The assumptions can be interpreted as a reduction of the control input space and focusing on considering more practical vehicle motion.

Under the kinematic assumptions, the vehicle's motion can be simplified as a model with 3 DoF $\mathbf{u}_{\text{3DoF}}$; longitudinal velocity V_x, lateral velocity V_y, and angular velocity ω of the vehicle center. The relationship between the vehicle center velocity $\mathbf{u}_{\text{3DoF}}$ and the eight-wheel velocities $\mathbf{u}_{\text{full}}$ is formulated as follows and shown in Fig.4.2.

$$\mathbf{u}_{\text{3DoF}} = [V_x, V_y, \omega]^\top, \tag{4.1}$$

$$\mathbf{u}_{\text{full}} = [V_{xfl}, V_{xfr}, V_{xrl}, V_{xrr}, V_{yfl}, V_{yfr}, V_{yrl}, V_{yrr}]^\top, \tag{4.2}$$

$$\mathbf{u}_{\text{full}} = \mathbf{C}_{3\to 8}\, \mathbf{u}_{\text{3DoF}}, \tag{4.3}$$

$$\mathbf{C}_{3\to 8} = \begin{bmatrix} 1 & 0 & -d_l \\ 1 & 0 & d_r \\ 1 & 0 & -d_l \\ 1 & 0 & d_r \\ 0 & 1 & l_f \\ 0 & 1 & l_f \\ 0 & 1 & -l_r \\ 0 & 1 & -l_r \end{bmatrix}. \tag{4.4}$$

The wheel velocity $\mathbf{u}_{\text{full}}$ is converted to the vehicle command $\mathbf{u}_{\text{vehicle}}$. $\mathbf{u}_{\text{vehicle}}$ consists of the steering angle δ_* and translational velocity V_* for each wheel, where $* \in \{fl, fr, rl, rr\}$ denotes the front-left, front-right, rear-left, and rear-right wheels, respectively.

$$\mathbf{u}_{\text{vehicle}} = [\delta_{fl}, \delta_{fr}, \delta_{rl}, \delta_{rr}, V_{fl}, V_{fr}, V_{rl}, V_{rr}]^\top \tag{4.5}$$

The nonlinear transformation, denoted as $\mathbf{u}_{\text{vehicle}} = f_v(\mathbf{u}_{\text{full}})$, is defined by the following

component-wise conversions for each wheel:

$$\delta_* = \arctan2(V_{y*}, V_{x*}) \tag{4.6}$$

$$V_* = \sqrt{V_{x*}^2 + V_{y*}^2} \tag{4.7}$$

Note that the kinematic formulation only requires the geometric relationship of the tires which is easy to obtain, and is easily applicable to real-world applications.

4.4.3 Exploring Solution in Redundant Control Input Space

Sampling based controller needs to get a variety of solutions from the control input space $\mathbf{u}_{\text{3DoF}}$. In this study, we found that exploring a slightly redundant space $\mathbf{u}_{\text{4DoF}}$ and converting it to $\mathbf{u}_{\text{3DoF}}$ was more effective in achieving smooth and stable vehicle motion with MPPI. This idea is inspired by methods such as Koopman Operator [72] and Dynamic Mode Decomposition [73], which expand the state space to express the complex dynamics as a linear system in a higher dimensional space.

Since $\mathbf{u}_{\text{4DoF}}$ is a 4-dimensional redundant space, it does not always satisfy kinematic constraints formulated in Eq. (4.3). Therefore, projection matrix $\mathbf{C}_{4\to8}$ is used to map $\mathbf{u}'_{\text{4DoF}}$ to $\mathbf{u}_{\text{vehicle}}$, so that the sampled vehicle motion always satisfies the kinematic constraints.

$$\mathbf{u}_{\text{4DoF}} = [V_{fl}, V_{rr}, \delta_{fl}, \delta_{rr}]^\top, \tag{4.8}$$

$$\begin{aligned}\mathbf{u}'_{\text{4DoF}} &= f(\mathbf{u}_{\text{4DoF}}) \\ &= [V_{fl}\cos\delta_{fl}, V_{rr}\cos\delta_{rr}, V_{fl}\sin\delta_{fl}, V_{rr}\sin\delta_{rr}]^\top \\ &= [V_{xfl}, V_{xrr}, V_{yfl}, V_{yrr}]^\top,\end{aligned} \tag{4.9}$$

$$\mathbf{u}_{\text{vehicle}} = \mathbf{C}_{4\to8}\mathbf{u}'_{\text{4DoF}}, \tag{4.10}$$

$$\mathbf{C}_{4\to8} = \mathbf{C}_{3\to8}\mathbf{C}_{4\to3}, \tag{4.11}$$

$$\mathbf{C}_{4\to3} = \begin{bmatrix} \frac{d_r}{d_l+d_r} & \frac{d_l}{d_l+d_r} & 0 & 0 \\ 0 & 0 & \frac{l_r}{l_f+l_r} & \frac{l_f}{l_f+l_r} \\ \frac{-1}{2(d_l+d_r)} & \frac{1}{2(d_l+d_r)} & 0 & 0 \end{bmatrix}. \tag{4.12}$$

The conversion to reduce a dimention in Eq. (4.12) is based on the idea of taking the average of the control inputs between front wheel space (V_{fl}, δ_{fl}) and the rear wheel space (V_{rr}, δ_{rr}) to obtain the compromised vehicle center angular velocity ω.

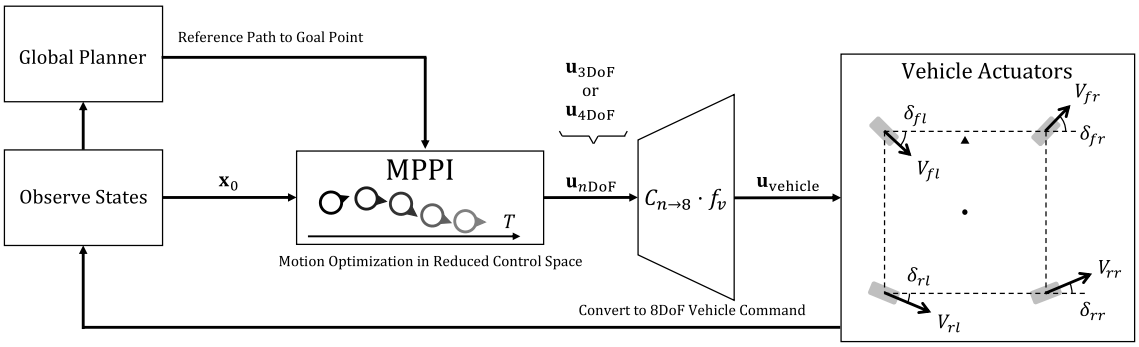


Figure 4.3: Overview of the control architecture (See Section 4.5.1). (a) global planner generates a reference trajectory based on current vehicle pose and the given map, (b) MPPI generates the optimal control input in a reduced dimensional space, (c) the n DoF ($n \in \{3,4\}$) control input is converted to the 8DoF vehicle command space, and (d) the vehicle actuators execute the command.

4.5 Navigation Architecture for 4WIDS Vehicle

4.5.1 System Overview

The navigation system for the 4WIDS vehicle is composed of four main components below in this work (See Fig.4.3)

State Observation

The map of the environment is given, and the vehicle localizes itself using 2D LiDAR point cloud data and odometry information.

Global Path Planning

The global planner calculates a path from the current vehicle position to the goal using Dijkstra's algorithm.

Local Path Planning

In this work, a Model Predictive Path-Integral Controller (MPPI) is used to plan the local path and calculate the next optimal control input. Since exploring the full vehicle command

space which has 8 DoF is difficult and inefficient, MPPI samples solutions in a dimension that is reasonably reduced by kinematic constraints such as $\mathbf{u}_{3\mathrm{DoF}}$ and $\mathbf{u}_{4\mathrm{DoF}}$ defined in Section 4.4.

Vehicle Command Calculation

Since MPPI calculates the control input in a reduced dimension, the control input is converted to the 8DoF vehicle command and sent to the vehicle actuators. The optimal control input calculated by MPPI is converted to the 8DoF vehicle command and sent to the actuators. The conversion is done by Eqs. (4.4), (4.12) and (4.7) in Section 4.4.

4.5.2 Algorithm of MPPI Controller

In this study, local planning is performed using the Model Predictive Path-Integral (MPPI) Controller [74] shown in Algorithm 1. MPPI is an optimal control algorithm that uses a sample-based approach to compute the optimal control input sequence in the near future. The algorithm is based on the idea of sampling the control input sequences as normal distributions centered at the previous optimal input sequence, and getting the optimal sequence as weighted sum so that better sequences are heavily weighted and vice versa.

For a discrete-time, continuous state-action system $\mathbf{x}_t = \mathbf{F}(\mathbf{x}, \mathbf{u})$, K samples of input sequences $V = \{\mathbf{v}_t\}_{t=0}^{T-1}$ are generated by adding Gaussian noise to mean control input sequence $U = \{\mathbf{u}_t\}_{t=0}^{T-1}$ with covariance matrix Σ. After preparing the samples, the optimal control input sequence is easily obtained by calculating the weighted sum of the samples. The stage cost function $c(\mathbf{x}, \mathbf{u})$ and terminal cost function $\phi(\mathbf{x})$ are defined to evaluate the quality of the samples. Let S_k be the total cost of the k-th sample whose input sequence is V_k, the weight for the sequence is

$$w(V_k) = \frac{1}{\eta}\left(-\frac{1}{\lambda}S(V_k) + \lambda \sum_{\tau=0}^{T-1} \mathbf{u}_t^\top \Sigma^{-1} \mathbf{v}_t - \rho\right) \tag{4.13}$$

$$\eta = \sum_{k=1}^{K} \exp\left(-\frac{1}{\lambda}S(V_k) + \lambda \sum_{\tau=0}^{T-1} \mathbf{u}_\tau^\top \Sigma^{-1} \mathbf{v}_\tau - \rho\right) \tag{4.14}$$

where η is the normalization factor, λ is the constant temperature parameter, and ρ is the minimum cost among the samples.

The notable benefits of MPPI are that it can be applied to a wide range of optimization problems such as those involving non-linear dynamics or cost functions that are non-convex or non-differentiable. In our application, this versatility allows the system to perform real-time obstacle avoidance using LiDAR sensor input. Figure 4.4 illustrates the point cloud obtained from the LiDAR sensor, showing how the robot perceives nearby obstacles during navigation. Since the environment can change rapidly and analytical gradients of the cost function are not available, a sample-based approach like MPPI is essential to ensure responsive and reliable planning.

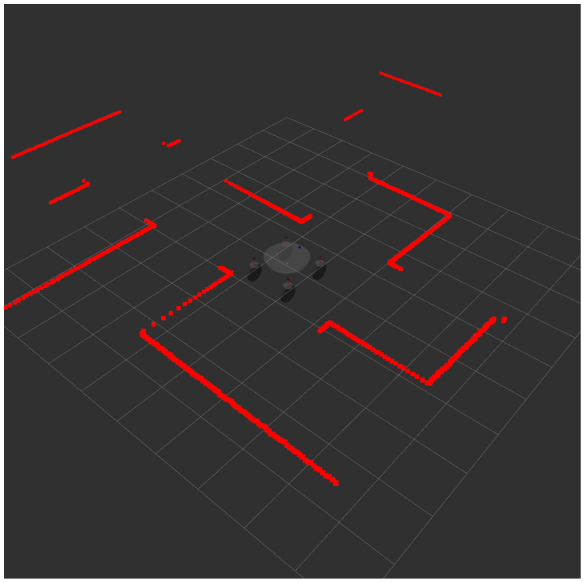


Figure 4.4: Real-time obstacle detection by LiDAR sensor. The red points represent the measured positions of surrounding obstacles.

Algorithm 1 Model Predictive Path-Integral Control in 4WIDS Vehicle Navigation

Given: F, g: Transition Model;
K: Number of samples;
T: Number of timesteps;
$U \leftarrow (\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{T-1})$: Initial control sequence;
$\Sigma, \phi, c, \gamma, \lambda, \alpha, \Delta t$: Cost functions and parameters;
while task not completed **:**
  $\mathbf{x}_0 \leftarrow$ ObserveSystemState()
  for k = 0 to K-1 **:**
    $\mathbf{x} \leftarrow \mathbf{x}_0$;
    Sample $\mathcal{E} = (\varepsilon_0^{\mathrm{k}} \ldots \varepsilon_{\mathrm{T}-1}^{\mathrm{k}})$, $\varepsilon \in \mathcal{N}(0, \Sigma)$;
    for t = 1 to T-1 **:**
      if $k < (1-\alpha)K$ **:**
        $\mathbf{v}_{t-1} = \mathbf{u}_{t-1} + \varepsilon_{t-1}^k$; ▷ samples for exploitation
      else
        $\mathbf{v}_{t-1} = \varepsilon_{t-1}^k$; ▷ samples for exploration
      $x \leftarrow \mathbf{F}(\mathbf{x}, g(\mathbf{v}_{t-1}), \Delta t)$;
      $S_k += c(\mathbf{x}, \mathbf{u}) + \gamma \mathbf{u}_{t-1}^{\top} \Sigma^{-1} \mathbf{v}_t$ ▷ add stage cost
    $S_k += \phi(\mathbf{x})$ ▷ add terminal cost
  $\rho \leftarrow \min_k[S_k]$;
  $\eta \leftarrow \sum_{k=1}^{K} \exp\left(-\frac{1}{\lambda}(S_k - \rho)\right)$;
  for k = 0 to K-1 **:**
    $\mathbf{w}_k \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} S_k\right)$; ▷ calculate sample weights
  for t = 0 to T-1 **:**
    $U \leftarrow U + \left(\sum_{k=1}^{K} \mathbf{w}_k \mathcal{E}^k\right)$; ▷ calculate weighted sum
  $\mathbf{u}_{\text{vehicle}} \leftarrow$ ConvertTo8DoFVehicleCommand($\mathbf{u}_0$);
  SendToVehicleActuators($\mathbf{u}_{\text{vehicle}}$);
  for t = 1 to T-1 **:**
    $\mathbf{u}_{t-1} \leftarrow \mathbf{u}_t$; ▷ shift the control sequence
  $\mathbf{u}_{T-1} \leftarrow$ Initialize($\mathbf{u}_{\mathrm{T}-1}$);

4.5.3 MPPI Switching Multiple Control Spaces

Through analysis of the experimental results in Section 4.6.4, we found that selecting the control input space to explore the solution affects the navigation performance significantly. To take advantage of the strengths and mitigate the weaknesses of each control input space, we propose a method to switch between multiple spaces in real time according to the situation explained in Algorithm 2. If two control input spaces are defined as $\mathbf{u}^A$ and $\mathbf{u}^B$, both spaces need to update the control input sequence in each time step as a preparation for the next calculation. The conversion functions ConvertToSpaceA and ConvertToSpaceB are needed to convert the control input sequence to the other space. If we switch $\mathbf{u}_{3\mathrm{DoF}}$ to $\mathbf{u}_{4\mathrm{DoF}}$, the conversions are done using Eqs. (4.4), (4.12), and (4.7) in Section 4.4.

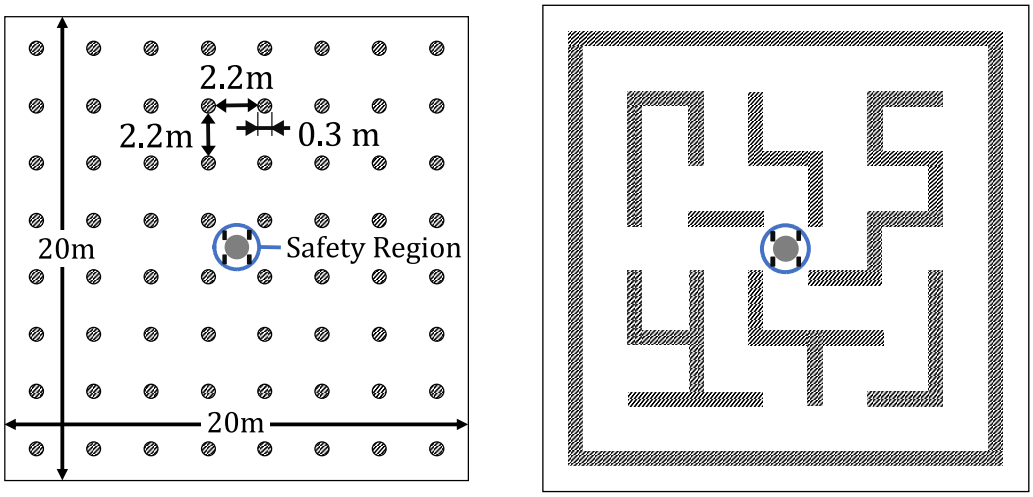
Algorithm 2 MPPI Switching Multiple Control Input Spaces

$U_0^A \leftarrow (u_0^A, \ldots, u_{T-1}^A)$: Initial control sequence of space A;
$U_0^B \leftarrow (u_0^B, \ldots, u_{T-1}^B)$: Initial control sequence of space B;
while task not completed **:**
    mode $\leftarrow$ SelectMode(); ▷ select control input space
    if mode is A **:**
        $U_{t+1}^A \leftarrow \mathrm{SolveMPPI}(\mathrm{U_t^A})$;
        $U_{t+1}^B \leftarrow \mathrm{ConvertToSpaceB}(\mathrm{U_{t+1}^A})$;
    else if mode is B **:**
        $U_{t+1}^B \leftarrow \mathrm{SolveMPPI}(\mathrm{U_t^B})$;
        $U_{t+1}^A \leftarrow \mathrm{ConvertToSpaceA}(\mathrm{U_{t+1}^B})$;

4.6 Experiments

4.6.1 Simulation Setup

To evaluate the navigation performance of the proposed architecture, we set up a simulation environment using Gazebo simulator. Two types of fields are prepared, "Cylinder Garden" and "Maze" (Fig. 4.5). The vehicle need to reach 10 goals sequentially in a episode as fast as possible while avoiding densely placed obstacles. The four wheel positions are set symmetrically as $l_f = l_r = d_l = d_r = 0.5$ [m] (Fig. 4.2).



(a) Cylinder Garden (Easier Environment) (b) Maze (Harder Environment)

Figure 4.5: Simulation environment. The 8 DoF vehicle needs to navigate through dense obstacles and narrow passages.

Our evaluation system was developed using ROS and C++. Calculation is performed on a desktop computer with an Intel Core i7-13700KF CPU and 32GB of RAM. For faster computation, we used CPU multi-threading with OpenMP [75].

4.6.2 Cost Formulation for MPPI

In this section, the cost functions $c(\mathbf{x},\mathbf{u})$ and $\phi(\mathbf{x})$ (See Algorithm 1 are defined for the MPPI controller.

The stage cost $c(\mathbf{x},\mathbf{u})$ is defined as the weighted sum of the following terms,

$$\begin{aligned} c(\mathbf{x},\mathbf{u}) = 40\, c_{\text{dist}}({}^{G}p_x, {}^{G}p_y) + 30\, c_{\text{angle}}(\theta) + 10\, c_{\text{speed}}(v) \\ + 50\, c_{\text{collision}}({}^{G}p_x, {}^{G}p_y) + c_{\text{cmd}}(\mathbf{u}), \end{aligned} \tag{4.15}$$

where ${}^{G}p_x$ and ${}^{G}p_y$ are the vehicle's position, θ is the vehicle's yaw angle, and v is the vehicle's velocity. $c_{\text{dist}}({}^{G}p_x, {}^{G}p_y)$, $c_{\text{angle}}(\theta)$, and $c_{\text{speed}}(v)$ are quadratic error from the reference path and constant target velocity v_{des}, respectively. $c_{\text{collision}}({}^{G}p_x, {}^{G}p_y)$ is a binary

cost function that returns 1 if the vehicle is in collision, and 0 otherwise;

$$c_{\text{collision}}({}^{G}p_x, {}^{G}p_y) = \begin{cases} 0 & \text{if no collision} \\ 1 & \text{if collision} \end{cases}, \tag{4.16}$$

Collision is determined based on a two-dimensional obstacle map that is updated in real time using LiDAR point cloud data. A collision is considered to have occurred if the center of the vehicle is closer to the nearest obstacle than its body radius of 0.6 m. $c_{\text{cmd}}(\mathbf{u})$ is used to smooth the vehicle actuator commands. In either control space $\mathbf{u}_{\text{3DoF}}$ or $\mathbf{u}_{\text{4DoF}}$ to be explored, it can be converted to 8-dimensional vehicle actuator commands $\mathbf{u}_{\text{vehicle}}$ with Eq. (4.3) and Eq. (4.10). With previous vehicle command $\mathbf{u}^{\text{prev}}_{\text{vehicle}}$ in the control sequence, minimizing the penalty term

$$c_{\text{cmd}}(\mathbf{u}) = \|\mathbf{u}_{\text{vehicle}} - \mathbf{u}^{\text{prev}}_{\text{vehicle}}\|_2 \tag{4.17}$$

can smooth the vehicle actuator commands.

the terminal cost $\phi(\mathbf{x})$ is added only for preventing reverse driving,

$$\phi(\mathbf{x}) = 50\ \phi_{\text{goal}}({}^{G}p_x, {}^{G}p_y), \tag{4.18}$$

where $\phi_{\text{goal}}({}^{G}p_x, {}^{G}p_y)$ is the quadratic error from the goal position.

4.6.3 Preparing MPPI Controllers for Comparison

To investigate the characteristics of the MPPI controller depending on the choice of the control space, MPPI-3D and MPPI-4D are prepared to explore two different control spaces, $\mathbf{u}_{\text{3DoF}}$ and $\mathbf{u}_{\text{4DoF}}$.

Since the variance parameter affects the controller's behavior, setting the variance of the control space carefully is important for fair comparison. Changing the variance parameters, two types of MPPI-3D are prepared, MPPI-3D(a) and MPPI-3D(b). The variance parameters of MPPI-3D(a) and MPPI-4D are set systematically as shown in Table 4.3 following the rule that the variance is half of the maximum value of the control space defined in Table 4.2. This consideration is to explore a wide range of the control space, as the normal distribution contains about 95% of the data within twice the standard deviation.

Table 4.1: MPPI Params

Param	Value	Unit
K	3000	sample
T	30	step
Δt	0.033	sec
α	0.1	-
λ	250	-
γ	6.25	-

Table 4.2: Controller Params

Param	Value	Unit
Control Interval Δt_i	0.05	sec
Target Velocity v_{des}	2.00	m/s
Max. Velocity v_{max}	2.00	m/s
Max. Yawrate ω_{max}	1.58	rad/s
Max. Steering Angle	1.58	rad

Table 4.3: MPPI Variance Params

Name	Control Space	Variance Σ
MPPI-3D(a)	$[V_x, V_y, \omega]$	$[1.00, 1.00, 0.78]$
MPPI-3D(b)	$[V_x, V_y, \omega]$	$[0.55, 0.55, 0.96]$
MPPI-4D	$[V_{fl}, V_{rr}, \delta_{fl}, \delta_{rr}]$	$[1.00, 1.00, 0.78, 0.78]$

Another approach is to fit a normal distribution numerically close to the sampled results of MPPI-4D. MPPI-3D(b) follows this procedure, and the variance parameters are calculated with maximum likelihood estimation of the normal distribution, when the vehicle is stopped and all the steering angles are zero in the space of $\mathbf{u}_{\text{4DoF}}$.

Additionally, a hybrid MPPI controller, MPPI-H, is prepared to switch between MPPI-3D and MPPI-4D depending on the situation. From the verification results, we determined that the mode selection should be based on the target path tracking error. The mode switching function in Algorithm 2 is defined as follows with constant parameter $d_{\text{thresh}} = 0.3$ [m] and $\theta_{\text{thresh}} = 0.3$ [rad],

$$\begin{aligned} &\text{SelectMode}() \\ &= \begin{cases} \mathbf{u}_{\text{3DoF}} & \text{if } c_{\text{dist}}({}^{G}p_x, {}^{G}p_y) < d_{\text{thresh}} \text{ and } \mathrm{c}_{\text{angle}}(\theta) < \theta_{\text{thresh}} \\ \mathbf{u}_{\text{4DoF}} & \text{otherwise} \end{cases}. \end{aligned} \tag{4.19}$$

4.6.4 Definition of Evaluation Metrics

Here the evaluation metrics are defined to compare the performance of the MPPI controllers. All metrics are calculated for all episodes, and the mean value is used for evalua-

tion.

Cost

”Cost” is the sum of stage cost and terminal cost of optimal trajectory output from MPPI. This metric indicates how well the MPPI controller can minimize the cost function and get close to the optimal behavior.

Vehicle Command Change

To evaluate the smoothness of the vehicle actuator commands, two metrics are defined. ”Steering Rate” is the absolute value of the steering angular velocity. ”Wheel Acceleration” means the absolute change of the wheel velocity. For both metrics, the mean values of four wheels are used for evaluation.

Navigation Efficiency

two metrics are defined to evaluate the navigation efficiency. ”Trajectory Length” is selected to evaluate how the vehicle could reach the goal with a short path. ”Episode Time” is also used to know how fast the vehicle could reach the goal.

Success Rate

Success rate is the percentage of episodes that the vehicle reached all the given goal points. Collision with obstacles and getting stuck in the field are major factors of failure.

4.6.5 Evaluation Results Comparison

100 episodes of navigation are performed for each field, and the evaluation results are shown in Table 4.4 and Table 4.5. For each MPPI controller, the mean calculation time is less than 30ms and works in real-time with the control interval $\Delta t_i = 50$ [ms]. Even though the cost function and algorithmic parameters are common, the results drastically changed by exploring different control spaces.

MPPI-3D(a) has a short episode time, which means it can drive efficiently. A lower success rate (76% in Cylinder Garden, 33% in Maze) means that it fails to complete episodes

frequently, and has difficulty in safe and stable navigation. Since MPPI-3D(b) has a smaller variance in the vehicle velocity space than MPPI-3D(a), it moves slowly and appears more conservative behavior. The episode time is the worst among the four controllers in both fields. However, the success rate is only slightly improved (89% in Cylinder Garden, 58% in Maze) from MPPI-3D(a) and still has low stability.

On the other hand, MPPI-4D showed an extremely high success rate (100% in Cylinder Garden, 99% in Maze) and stable navigation behavior. MPPI-4D achieved the lowest cost and shortest trajectory length among the four controllers, indicating its capability to find the optimal solution.

Comparing the episode time in Cylinder Garden, MPPI-4D (38.4s) is faster than MPPI-3D(b) (41.3s), but MPPI-3D(a) (36.4s) surpasses MPPI-4D in terms of driving efficiency.

Summarizing the characteristics of MPPI-3D and MPPI-4D, MPPI-3D is good at high-speed driving but has low stability, and MPPI-4D is more conservative and is good at stable navigation but has lower efficiency. Then it is reasonable to switch control spaces depending on the situation to balance the efficiency and stability. In the case the vehicle is in a difficult situation (*i.e.*when the path tracking error is large), MPPI-4D is selected to ensure stable navigation. Otherwise, MPPI-3D is selected to drive efficiently.

As a result, the hybrid MPPI-H switching control spaces showed the best episode time (31.2s in Cylinder Garden, 44.8s in Maze), keeping the high success rate (99% in Cylinder Garden, 96% in Maze) compared to MPPI-4D. The fact that MPPI-H has no worst score in any of the metrics is also evidence that MPPI-H has a balanced performance.

4.6.6 Trajectory Comparison

To understand the characteristics of the controllers more deeply, the trajectories of the four controllers are compared in Fig. 4.6. This example scenario is a hard situation where the vehicle receives the next goal point and needs to turn sharply.

In Fig. 4.6, the MPPI-3D(b) turns with a larger radius than the other controllers, and a part of the trajectory is close to the collision with surrounding obstacles. It is a typical dangerous behavior and shows the reason of the lower success rate of MPPI-3D controllers.

On the other hand, MPPI-4D can turn in a small radius and run safely keeping a distance from the obstacles, showing the reason of the high success rate of MPPI-4D.

Table 4.4: Evaluation Results of 100 Navigation Episodes in Cylinder Garden
Blue: best, Red: worst among controllers

Metric	MPPI-3D(a) $[V_x, V_y, \omega]$	MPPI-3D(b) $[V_x, V_y, \omega]$	MPPI-4D $[V_{fl}, V_{rr}, \delta_{fl}, \delta_{rr}]$	MPPI-H 3D(a)/4D
Cost $[-] \downarrow$	3241.7	1900.5	1455.8	2425.4
Calc. Time [ms] $\downarrow$	24.1	23.0	27.6	26.6
Steering Rate [rad/s] $\downarrow$	4.5	3.1	3.6	4.0
Wheel Acc. $[\mathrm{m/s^2}] \downarrow$	5.03	3.36	4.08	4.98
Trajectory Length [m] $\downarrow$	51.9	46.0	40.8	42.6
Episode Time [s] $\downarrow$	36.4	41.3	38.4	31.2
Success Rate [%] $\uparrow$	76	89	100	99

Note that minor collisions that do not hinder navigation are not considered failures.

Table 4.5: Evaluation Results of 100 Navigation Episodes in Maze
Blue: best, Red: worst among controllers

Metric	MPPI-3D(a) $[V_x, V_y, \omega]$	MPPI-3D(b) $[V_x, V_y, \omega]$	MPPI-4D $[V_{fl}, V_{rr}, \delta_{fl}, \delta_{rr}]$	MPPI-H 3D(b)/4D
Cost $[-] \downarrow$	10030.4	3918.8	2452.3	2887.6
Calc. Time [ms] $\downarrow$	19.7	19.9	24.0	21.0
Steering Rate [rad/s] $\downarrow$	6.0	3.6	5.0	3.5
Wheel Acc. $[\mathrm{m/s^2}] \downarrow$	6.02	3.77	4.85	4.02
Trajectory Length [m] $\downarrow$	72.1	64.8	55.2	55.3
Episode Time [s] $\downarrow$	49.6	55.9	52.1	44.8
Success Rate [%] $\uparrow$	33	58	98	96

Note that minor collisions that do not hinder navigation are not considered failures.

In the situation where the vehicle should turn sharply, MPPI-H activates MPPI-4D to drive carefully, resulting in the same turning behavior as MPPI-4D.

4.6.7 Effect of Control Input Space Selection

In this section, why the performance changes by selecting control input space for sampling is discussed. Specifically, the explanation of why the solution search in the control space $\mathbf{u}_{\mathrm{4DoF}}$ is more likely to find a more optimal solution than in $\mathbf{u}_{\mathrm{3DoF}}$ is provided.

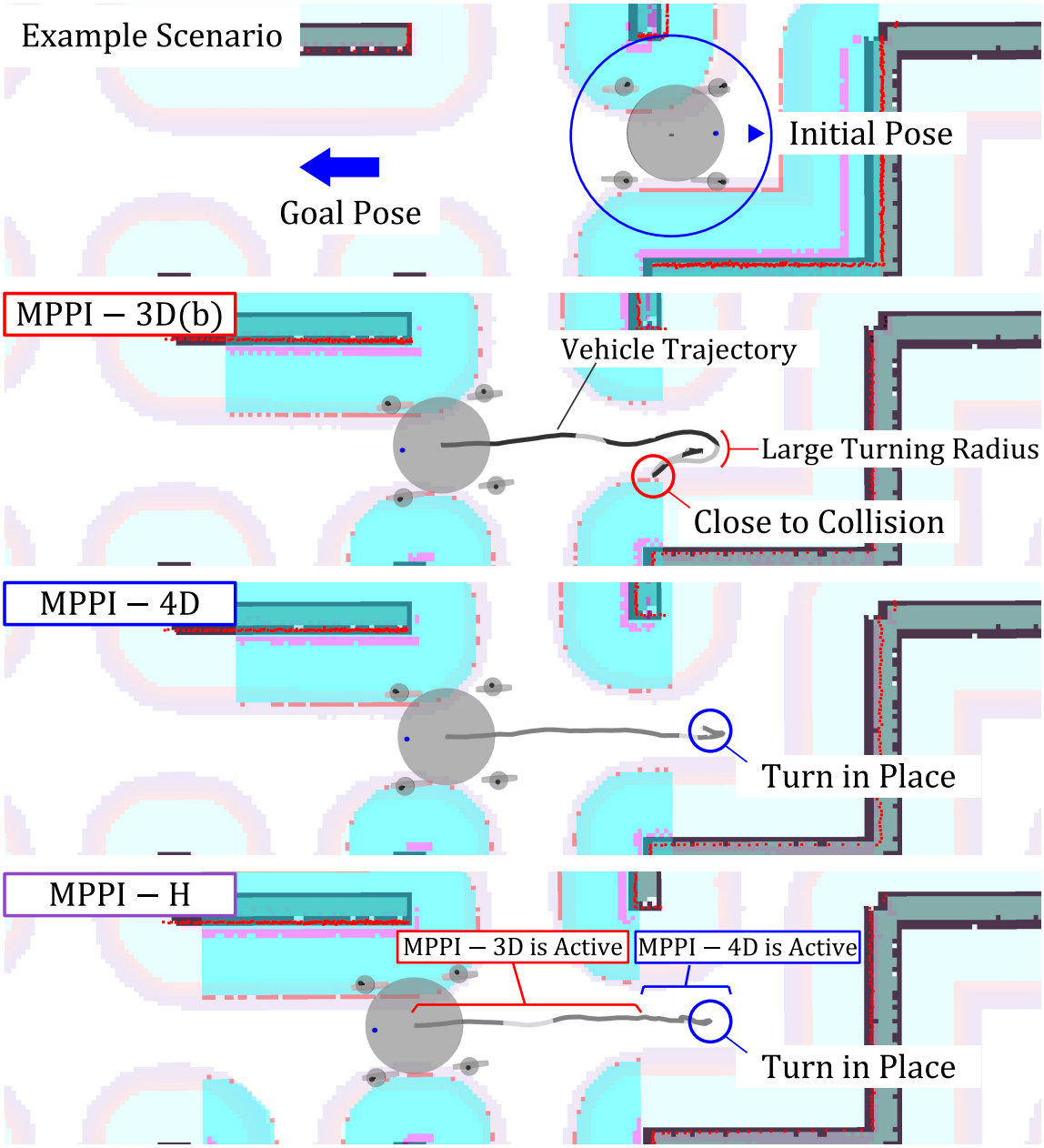


Figure 4.6: Trajectory comparison under a challenging navigation scenario.
While MPPI-3D shows dangerous behavior close to the collision, MPPI-4D and MPPI-H drive safely with turning in a small radius.

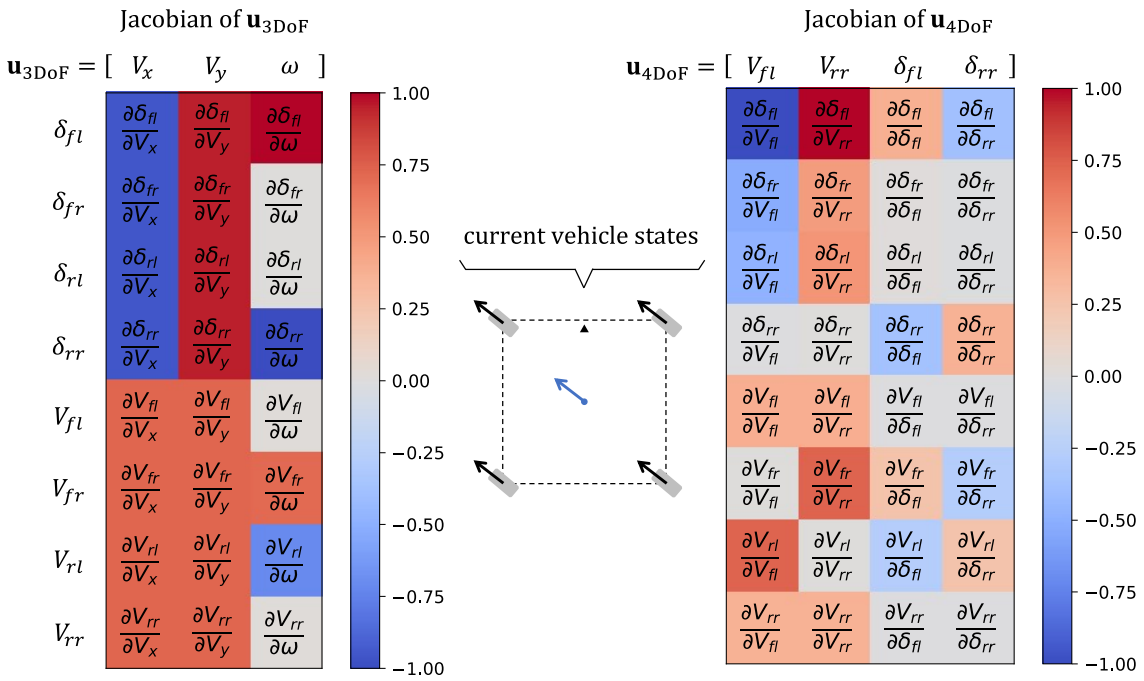


Figure 4.7: Jacobian matrices comparison between $\mathbf{u}_{3\mathrm{DoF}}$ and $\mathbf{u}_{4\mathrm{DoF}}$ when $\delta_{fl} = \delta_{fr} = \delta_{rl} = \delta_{rr} = 0.25\pi$ [rad] and $V_{fl} = V_{fr} = V_{rl} = V_{rr} = 0.7$ [m/s]. J_B has a more sparse structure than J_A, which makes the optimal solution search easier.

In the navigation task, one of the most difficult situation is when the next goal point is specified in the opposite direction to the vehicle's heading. In this case, the optimal behavior is to quickly decelerate the vehicle and rotate the vehicle body to the target direction.

Here we consider the Jacobian matrices J_A, J_B with respect for the projection from each control input space $\mathbf{u}_{3\mathrm{DoF}}, \mathbf{u}_{4\mathrm{DoF}}$ to the vehicle command space $\mathbf{u}_{\mathrm{vehicle}}$. The Jacobian matrices are normalized to the range of -1 to 1 and plotted as color maps in Fig.4.7.

As for the control input space $\mathbf{u}_{3\mathrm{DoF}}$, deceleration of the vehicle speed V_x is strongly affects the vehicle steering angles. Therefore, even if a control sequence that includes a rapid deceleration is sampled, it is likely to be penalized due to the large change in the steering angle. As a result, the deceleration behavior is less likely to occur and the vehicle tries to rotate with the current speed, which often causes collision with obstacles.

On the other hand, as for the control input space $\mathbf{u}_{4\mathrm{DoF}}$, the Jacobian matrix J_B has a more sparse structure than J_A. This means that the change in the vehicle speed and the

steering angles are relatively independent. This feature makes the optimal solution search easier, and more likely to find the better solution including rapid deceleration and rotation of the vehicle at the same time.

4.7 Discussion and Insights on Q2-β

This chapter establishes a framework for autonomous navigation of 8 DoF vehicles in obstacle-rich environments using MPPI. The implementation successfully achieves autonomous navigation in complex environments by leveraging the flexible motion capabilities of 8 DoF vehicles while maintaining practical applicability through real-time obstacle detection from LiDAR point cloud data.

This chapter establishes a baseline framework for addressing research question **[Q2-β] How can the handling of multiple tasks be enhanced?** The weighted scalarization approach that combines multiple task objectives represents the mainstream methodology, and this chapter follows the standard approach.

The straightforward implementation of MPPI (MPPI-3D) demonstrated insufficient success rates for autonomous navigation. However, safety performance improved through modifications to the vehicle action sampling space. The enhanced sampling space in MPPI-4D achieved higher success rates despite somewhat conservative behavior. The hybrid approach (MPPI-H), which switches sampling spaces based on the situation, maintained high success rates while enabling efficient navigation.

This finding indicates that simply applying MPPI to a system may not yield optimal results, suggesting that further improvements can be achieved by refining the optimization algorithm. This chapter lays the groundwork for Chapter 5, where we explore enhanced optimization algorithms to further improve the performance of 8 DoF vehicles.

Chapter 5

Nullspace MPC: Priority-Based Decomposition of Multi-Objective Control with Application to 8 DoF Vehicle Navigation

5.1 Chapter Overview and Contribution to Q2-β

This chapter proposes a novel optimization algorithm **SA-HQP** (Sampling-Augmented Hierarchical Quadratic Programming) and its receding horizon extension **Nullspace MPC** to achieve multi-objective control and demonstrates its application to 8 DoF vehicle navigation. This work addresses **[Q2-β] How can the handling of multiple tasks be enhanced?** and demonstrates the advantages of the proposed approach over the baseline method MPPI presented in Chapter 4.

Conventional approaches like MPPI minimize a scalarized cost function where multiple task-related terms are weighted. This approach relies solely on soft constraints through weight parameters. High-degree-of-freedom vehicles often have multiple control input patterns that can achieve a given task. This characteristic makes the optimization problem non-

convex, potentially leading to local optima that fail to produce desirable driving behavior. While well-tuned weight parameters can yield satisfactory performance, finding such parameters is often challenging. Even when suitable parameters are found, their effectiveness across diverse environments remains uncertain.

The proposed approach takes a hierarchical decomposition strategy by explicitly considering task priorities in multi-objective control problems. In trajectory planning for holonomic vehicles as a proof of concept, the proposed method demonstrated superior performance compared to MPPI by achieving higher-priority tasks more certainly while generating trajectories that were more optimal in the context of MPPI's cost function. Validation through narrow-space navigation with 8 DoF vehicles showed advantages in both trajectory length and time-to-goal, achieving more efficient driving performance compared to MPPI.

5.2 Background and Related Work

Holonomic vehicles possess particularly high degrees of freedom among mobile systems, enabling flexible movements such as in-place rotation and lateral motion. This characteristic makes them well-suited for efficient navigation in confined spaces and environments with dense obstacles [77]. However, the high number of degrees of freedom results in numerous possible motion choices, making optimal action selection challenging.

A multi-objective optimization framework provides a suitable approach for developing action plans that leverage the performance capabilities of high-degree-of-freedom systems. This methodology requires the designer to define a set of objectives (tasks), and the optimization process then aims to find solutions that satisfy these objectives to the greatest possible extent.

A prevalent method for addressing multi-objective optimization problems involves minimizing a scalarized cost function, which is typically formulated as a weighted sum of terms representing individual tasks [78]. This strategy demonstrates effectiveness when managing a limited number of tasks and has found numerous practical applications. Various optimization techniques, such as gradient-based and sampling-based methods, have been extensively investigated to identify optimal solutions within this framework [54, 79, 80].

Among these, Model Predictive Path-Integral Control (MPPI) [80] stands out as a prominent sampling-based optimization technique. It significantly expands the applicability of optimization methods by utilizing a gradient-free algorithm for solution searching and has proven effective across diverse applications [77, 81–83]. However, this approach of balancing multiple task objectives through weighting parameters encounters limitations. Specifically, it is less suitable for scenarios involving a large number of tasks because it is challenging to find an appropriate set of weights to achieve a satisfactory trade-off among multiple tasks.

When addressing problems where achieving all tasks through single-objective optimization is challenging, problem decomposition emerges as an effective strategy [84–88]. This study focuses on hierarchical decomposition based on task priorities as a rational approach to problem partitioning. The hierarchical framework establishes explicit priorities among tasks and aims to achieve them sequentially, from highest to lowest priority. While this approach excludes solutions that sacrifice higher-priority tasks for lower-priority ones, it simplifies the optimization problem at each hierarchical level. For instance, in a scenario involving obstacle avoidance and smooth motion as objectives, solutions that achieve smooth motion at the cost of collision are practically undesirable. In such cases, a reasonable problem decomposition - selecting the smoothest motion from the set of collision-free solutions - facilitates problem-solving without compromising solution quality.

While hierarchical multi-objective optimization problems with task priorities can be solved using Hierarchical Quadratic Programming (HQP) [87, 88], this approach has limitations in its applicability. Specifically, HQP requires both the system model and optimization objectives to be expressed in linear form, making it difficult to handle nonlinear constraints such as avoiding obstacles with arbitrary shapes.

This study first proposes SA-HQP (Sampling-Augmented HQP), a novel optimization framework for solving motion planning problems for holonomic vehicles. SA-HQP extends the hierarchical optimization framework formulated by HQP by incorporating insights from sampling-based optimization, enabling the handling of nonlinear expressions. Furthermore, SA-HQP is extended to a receding horizon controller, proposing Nullspace MPC that successfully achieves 8 DoF vehicle navigation. Nullspace MPC demonstrates

efficient navigation capabilities, particularly in narrow-space navigation tasks, where it achieves faster goal-reaching times and shorter paths compared to MPPI.

Lastly, we discuss the relationship between SA-HQP and lexicographic optimization, a concept in the field of Operations Research. Lexicographic optimization is a multi-objective framework that assigns explicit task priorities and ensures that no lower-level task can degrade a higher-priority one, sharing the same motivation as SA-HQP. It has been applied in diverse areas such as liquid-tank regulation [89], aircraft trajectory design [90], and worker-assignment scheduling [91]. Lexicographic problems are often solved efficiently as linear-programming or with gradient-based methods when the objectives and constraints can be written as linear or smoothly convex functions. This structural requirement limits the class of task formulations: non-differentiable objectives—such as obstacle avoidance with arbitrary shapes or discrete cost maps—cannot be handled directly. To address this restriction, some approaches discretize the state space via sampling or graph search [92, 93], but they suffer from the curse of dimensionality [94]. SA-HQP can be viewed as a hybrid that keeps the fast, priority-preserving HQP core (gradient-based) while delegating non-smooth tasks to a low-dimensional sampling layer. As a result it (1) runs in real time for vehicle navigation, (2) handles nonlinear or non-differentiable task representations, and (3) mitigates the high-dimensional exploration by sampling only a small parameter subspace. The trade-off is that sampling relaxes the strict optimality guarantee of pure HQP and complicates stability analysis.

5.3 Formulation of Optimization Problem Considering Task Priority

This section formulates the framework that decomposes optimization problems by considering task priorities, which is the main contribution of this work.

Let $\boldsymbol{s}$ be a vector containing all variables to be determined in the motion planning of a system, and let $\mathbb{S}$ be the solution set of $\boldsymbol{s}$. The objective of this method is to select an optimal solution $\boldsymbol{s}_*$ from the solution set $\mathbb{S}$ that satisfies multiple tasks specified by the designer as much as possible.

The optimization variable $\boldsymbol{s}$ for the motion planning problem of a holonomic vehicle is defined as follows. For the vehicle state, let p_x and p_y denote the x and y components of the vehicle position in the global coordinate frame, and θ represent the vehicle heading angle. For the control inputs, let v_x and v_y denote the vehicle velocities in the global coordinate frame, and ω represent the angular velocity. Given a planning horizon of T time steps, the state vector $\boldsymbol{p}_k \in \mathbb{R}^3$ and control input vector $\boldsymbol{v}_k \in \mathbb{R}^3$ at time step k are defined as,

$$\begin{aligned} \boldsymbol{p}_k &= [p_x^k, p_y^k, \theta^k] && (k = 0, \cdots, T), \\ \boldsymbol{v}_k &= [v_x^k, v_y^k, \omega^k] && (k = 0, \cdots, T-1), \end{aligned}$$

and the vector $\boldsymbol{s} \in \mathbb{R}^{6T+3}$ that represents the solution to the motion planning problem is defined as,

$$\boldsymbol{s} = [\boldsymbol{p}_0, \boldsymbol{v}_0, \cdots, \boldsymbol{p}_k, \boldsymbol{v}_k, \cdots, \boldsymbol{p}_{T-1}, \boldsymbol{v}_{T-1}, \boldsymbol{p}_T]^\top. \tag{5.1}$$

First, the conditions that $\boldsymbol{s}$ must satisfy are defined as *tasks*. A task $\mathcal{T}$ is expressed as a set of nonlinear equality and inequality constraints, as shown in Eq. (5.2). A task can be defined using either equality constraints or inequality constraints alone. The set of solutions $\boldsymbol{s}$ that satisfy task $\mathcal{T}$ is called the task set and is denoted as $\mathcal{S}_\mathcal{T}$.

$$\mathcal{T} : F(\boldsymbol{s}) = \boldsymbol{0},\ G(\boldsymbol{s}) \leq \boldsymbol{0} \tag{5.2}$$

$$\mathcal{S}_\mathcal{T} = \{\boldsymbol{s} \in \mathcal{S} \mid F(\boldsymbol{s}) = \boldsymbol{0}, G(\boldsymbol{s}) \leq \boldsymbol{0}\} \tag{5.3}$$

This framework prepares multiple prioritized tasks and aims to achieve them in order of priority. From this perspective, a motion planning problem $\mathcal{P}$ is expressed as an ordered list of N hierarchical tasks $\mathcal{T}_1 \cdots \mathcal{T}_N$ (Eq. (5.4)). The integers from 1 to N assigned to each task represent the task priorities, where smaller values indicate higher priority ($\mathcal{T}_1$ has the highest priority).

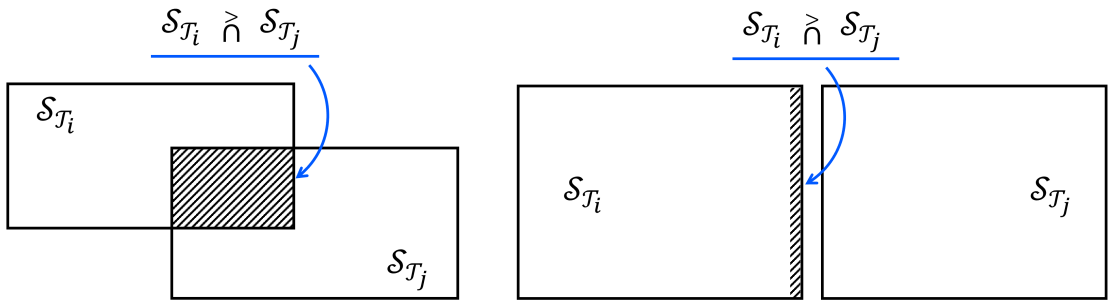
$$\mathcal{P} = \begin{cases} \mathcal{T}_1 : F_1(\boldsymbol{s}) = \boldsymbol{0},\ G_1(\boldsymbol{s}) \leq \boldsymbol{0} \\ \quad \vdots \\ \mathcal{T}_N : F_N(\boldsymbol{s}) = \boldsymbol{0},\ G_N(\boldsymbol{s}) \leq \boldsymbol{0} \end{cases} \tag{5.4}$$

One approach to solving the prioritized control problem (Eq. (5.4)) is to sequentially add tasks in order of priority while progressively constraining the solution set where the

optimal solution exists. Qualitatively, this can be understood as lower priority tasks aiming to achieve their objectives only within the constraints that do not interfere with the achievement of higher priority tasks.

Consider a procedure to constrain the solution set by adding a lower priority task $\mathcal{T}_j$ $(i < j)$ to a higher priority task set $\mathcal{S}_{\mathcal{T}_i}$ that satisfies task $\mathcal{T}_i$ $(i \in 1, 2, \cdots, N-1)$. This means selecting a solution set $\mathcal{S}_{\mathcal{T}_i} \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_j}$ from within the higher priority task set $\mathcal{T}_i$ that also achieves the lower priority task. The operator $\overset{>}{\cap}$ is newly defined to narrow down the solution set while considering priorities. While it fundamentally takes the intersection of the sets on both sides like the $\cap$ operator, when the intersection is an empty set, it returns the solution set from the higher priority task set on the left side that minimizes deviation from the lower priority task (Eq. (5.5), Fig.5.1).Here, w_p represents a penalty term added when inequality constraints are not satisfied. Depending on the task formulation, the penalty term w_p can either be set as a constant or defined as a function of the magnitude of constraint violation $G_j(\boldsymbol{s})$.

$$\mathcal{S}_{\mathcal{T}_i} \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_j} = \begin{cases} \text{if} \quad \mathcal{S}_{\mathcal{T}_i} \cap \mathcal{S}_{\mathcal{T}_j} \neq \phi : \\ \quad \mathcal{S}_{\mathcal{T}_i} \cap \mathcal{S}_{\mathcal{T}_j} \\ \text{if} \quad \mathcal{S}_{\mathcal{T}_i} \cap \mathcal{S}_{\mathcal{T}_j} = \phi : \\ \quad \{\boldsymbol{s} \in \mathcal{S}_{\mathcal{T}_i} \mid \min_{\boldsymbol{s}}(\|F_j(\boldsymbol{s})\|_2 + G_b(\boldsymbol{s}))\}, \\ \quad G_b(\boldsymbol{s}) = \begin{cases} 0 & \text{if } G_j(\boldsymbol{s}) \leq \boldsymbol{0} \\ w_p & \text{otherwise} \end{cases}. \end{cases} \tag{5.5}$$



(a) if $\mathcal{S}_{\mathcal{T}_i} \cap \mathcal{S}_{\mathcal{T}_j} \neq \phi$ (b) if $\mathcal{S}_{\mathcal{T}_i} \cap \mathcal{S}_{\mathcal{T}_j} = \phi$

Figure 5.1: Definition of $\mathcal{S}_{\mathcal{T}_i} \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_j}$. Check Eq. (5.5).

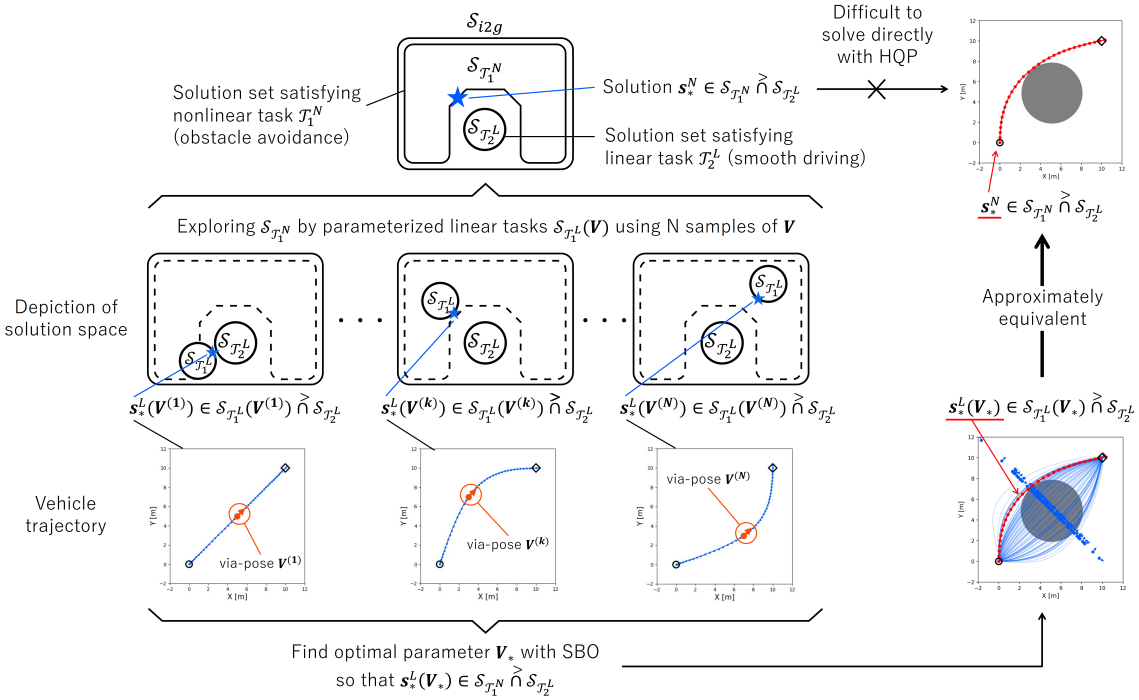


Figure 5.2: Overview of the proposed approach to solve a simple example problem. $\mathcal{S}_{i2g}$ is a solution set of vehicle trajectories connecting the initial and goal poses. Two tasks ($\mathcal{T}_1^N$ and $\mathcal{T}_2^L$) are additionally taken into account to achieve obstacle avoidance and smooth driving. The nonlinear task $\mathcal{T}_1^N$ is alternatively expressed as a linear task parametrized at the $\boldsymbol{V}$ which means a via-pose to be passed on the way, as depicted by the arrows in the vehicle trajectory figures. The appropriate $\boldsymbol{V}$ to satisfy obstacle avoidance is explored by a sampling-based optimizer, and the smooth trajectory avoiding obstacles is finally obtained as a solution of the motion planning problem.

The optimal solution $\boldsymbol{s}_*$ that achieves all tasks according to their priorities satisfies Eq. (5.6). When multiple operators $\hat{\cap}$ exist in a single equation, they are evaluated from left to right.

$$\boldsymbol{s}_* \in \mathcal{S}_{\mathcal{T}_1} \hat{\cap} \cdots \hat{\cap} \mathcal{S}_{\mathcal{T}_N} \tag{5.6}$$

When multiple solutions $\boldsymbol{s}_*$ exist, any one can be selected since all the solutions already satisfy all task constraints.

5.4 How to Solve Optimization Problem Considering Task Priority

This section describes the Sampling-Augmented Hierarchical Quadratic Programming (SA-HQP) method for solving multi-objective optimization problems with task priorities, yielding the solution $\boldsymbol{s}_*$ (Eq. (5.6)). A schematic overview of the method is presented in Fig. 5.2.

As a special case, when all tasks can be expressed as linear tasks $\mathcal{T}^L$ (where the subscript L denotes "linear") comprising sets of linear equality and inequality constraints as shown in Eq. (5.7), the problem can be efficiently solved using Hierarchical Quadratic Programming (HQP) [87, 88]. The details of HQP are described in Section 5.5.

$$\mathcal{T}^L : \mathbf{A}\boldsymbol{s} - \boldsymbol{b} = \mathbf{0},\ \mathbf{D}\boldsymbol{s} - \boldsymbol{f} \leq \mathbf{0} \tag{5.7}$$

However, when the problem contains even one nonlinear task $\mathcal{T}^{\mathbf{N}}$ (where the subscript $\mathbf{N}$ denotes "nonlinear") that cannot be expressed in the form of Eq. (5.7), HQP cannot be applied. In practical applications, problems often involve nonlinear tasks, which significantly limits the applicability of the basic HQP approach.

This work proposes a framework that extends the applicability of the method by incorporating sampling-based optimization to handle nonlinear tasks. The problem formulation for cases involving both linear and nonlinear tasks is shown in Eq. (5.8). For simplicity, this analysis focuses on the case in which only task m $(1 \leq m \leq N)$ is nonlinear, though the approach can be extended to address multiple nonlinear tasks.

$$\mathcal{P}^{\mathbf{N}} = \begin{cases} \mathcal{T}_1^L : F_1^L(\boldsymbol{s}) = \mathbf{0},\ G_1^L(\boldsymbol{s}) \leq \mathbf{0} \\ \quad \vdots \\ \mathcal{T}_m^{\mathbf{N}} : F_m^{\mathbf{N}}(\boldsymbol{s}) = \mathbf{0},\ G_m^{\mathbf{N}}(\boldsymbol{s}) \leq \mathbf{0} \\ \quad \vdots \\ \mathcal{T}_N^L : F_N^L(\boldsymbol{s}) = \mathbf{0},\ G_N^L(\boldsymbol{s}) \leq \mathbf{0} \end{cases} \tag{5.8}$$

The solution $\boldsymbol{s}_*^{\mathbf{N}}$ that satisfies Eq. (5.9) must be found.

$$\boldsymbol{s}_*^{\mathbf{N}} \in \mathcal{S}_{\mathcal{T}_1^L} \overset{>}{\cap} \cdots \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_m^{\mathbf{N}}} \overset{>}{\cap} \cdots \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_N^L} \tag{5.9}$$

Since directly solving the problem $\mathcal{P}^{\mathbf{N}}$ containing nonlinear tasks is challenging, this work proposes an approach that introduces a parameter $\boldsymbol{V}$ to transform the nonlinear tasks into locally linear tasks $\mathcal{P}^{L}(\boldsymbol{V})$ (Eq. (5.10)), which can then be solved using HQP.

$$\mathcal{P}^{L}(\boldsymbol{V}) = \begin{cases} \mathcal{T}_1^L : F_1^L(\boldsymbol{s}) = \mathbf{0},\ G_1^L(\boldsymbol{s}) \leq \mathbf{0} \\ \quad \vdots \\ \mathcal{T}_m^L : F_m^L(\boldsymbol{s};\boldsymbol{V}) = \mathbf{0},\ G_m^L(\boldsymbol{s};\boldsymbol{V}) \leq \mathbf{0} \\ \quad \vdots \\ \mathcal{T}_N^L : F_N^L(\boldsymbol{s}) = \mathbf{0},\ G_N^L(\boldsymbol{s}) \leq \mathbf{0} \end{cases} \tag{5.10}$$

The proposed approach is illustrated using a concrete example, as shown in Fig. 5.2. In vehicle motion planning, obstacle avoidance with arbitrary shapes is a nonlinear task. The method introduces a parameter $\boldsymbol{V} = [\boldsymbol{p}_{\text{mid}}]^{\top}$ that includes the vehicle state $\boldsymbol{p}_{\text{mid}}$ to be achieved at a specific time step T_{mid}. In the example shown in Fig. 5.2, $\boldsymbol{V}$ corresponds to a via pose through which the vehicle must pass on its way to the goal location. The constraint requiring the vehicle state at time T_{mid} to match the target value specified by parameter $\boldsymbol{V}$ can be expressed as a linear task. By varying $\boldsymbol{V}$, diverse vehicle trajectories $\boldsymbol{s}$ can be generated. By selecting an appropriate parameter $\boldsymbol{V}$ that enables obstacle avoidance, the nonlinear task of avoiding arbitrarily shaped obstacles can be achieved.

The solution $\boldsymbol{s}_*^L(\boldsymbol{V})$ to the problem $\mathcal{P}^{L}(\boldsymbol{V})$ for a given parameter $\boldsymbol{V}$ can be obtained using HQP, and the resulting solution $\boldsymbol{s}_*^L(\boldsymbol{V})$ satisfies Eq. (5.11).

$$\boldsymbol{s}_*^L(\boldsymbol{V}) \in \mathcal{S}_{\mathcal{T}_1^L} \overset{>}{\cap} \cdots \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_m^{\mathbf{L}}} \overset{>}{\cap} \cdots \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_N^L} \tag{5.11}$$

The solution $\boldsymbol{s}_*^L(\boldsymbol{V})$ represents the result of considering all linear tasks in order of priority under a given parameter $\boldsymbol{V}$. When exploring the parameter $\boldsymbol{V}$, it is necessary to consider all tasks that are affected by this parameter selection. Specifically, the nonlinear task $\mathcal{T}_m^{\mathbf{N}}$ and all linear and nonlinear tasks with lower priority must be considered as constraints. Let $\boldsymbol{V}_*$ denote the parameter that satisfies these conditions and $\mathcal{V}$ represent the set of possible

values for $\boldsymbol{V}$. Then, $\boldsymbol{V}_*$ is expressed by Eq. (5.12).

$$\begin{aligned} \boldsymbol{V}_* = \boldsymbol{V} \in \mathcal{V} \\ s.t. \\ \begin{cases} \mathcal{T}_m^{\mathbf{N}} : F_m^{\mathbf{N}}(\boldsymbol{s}_*^L(\boldsymbol{V})) = \mathbf{0},\ G_m^{\mathbf{N}}(\boldsymbol{s}_*^L(\boldsymbol{V})) \leq \mathbf{0} \\ \mathcal{T}_{m+1}^L : F_{m+1}^L(\boldsymbol{s}_*^L(\boldsymbol{V})) = \mathbf{0},\ G_{m+1}^L(\boldsymbol{s}_*^L(\boldsymbol{V})) \leq \mathbf{0} \\ \quad \vdots \\ \mathcal{T}_N^L : F_N^L(\boldsymbol{s}_*^L(\boldsymbol{V})) = \mathbf{0},\ G_N^L(\boldsymbol{s}_*^L(\boldsymbol{V})) \leq \mathbf{0} \end{cases} \end{aligned} \tag{5.12}$$

To explore $\boldsymbol{V}_*$, this work introduces Sampling-Based Optimization (SBO), which estimates solutions by randomly sampling parameter $\boldsymbol{V}$ and identifying those that satisfy the required conditions. The details of SBO are described in Section 5.6.

Finally, the solution $\boldsymbol{s}_*^L(\boldsymbol{V}_*)$ obtained using the optimal parameter $\boldsymbol{V}_*$ from SBO serves as an approximation of the true solution $\boldsymbol{s}_*^{\mathbf{N}}$ (Eq. (5.9)) to the multi-objective control problem (Eq. (5.8)) that contains both linear and nonlinear tasks.

5.5 Hierarchical Quadratic Programming (HQP)

This section describes an efficient method for solving multi-objective optimization problems $\mathcal{P}^L$ where all tasks are expressed solely through linear equalities and inequalities as shown in Eq. (5.13). The solution $\boldsymbol{s}_*^L \in \mathcal{S}_{\mathcal{T}_1^L} \overset{>}{\cap} \cdots \overset{>}{\cap} \mathcal{S}_{\mathcal{T}_N^L}$ can be obtained by sequentially solving hierarchical quadratic programming problems [87, 88].

$$\mathcal{P}^L = \begin{cases} \mathcal{T}_1^L : \mathbf{A_1}\boldsymbol{s} - \boldsymbol{b_1} = \mathbf{0},\ \mathbf{D_1}\boldsymbol{s} - \boldsymbol{f_1} \leq \mathbf{0} \\ \quad \vdots \\ \mathcal{T}_N^L : \mathbf{A_N}\boldsymbol{s} - \boldsymbol{b_N} = \mathbf{0},\ \mathbf{D_N}\boldsymbol{s} - \boldsymbol{f_N} \leq \mathbf{0} \end{cases} \tag{5.13}$$

The fundamental approach to obtaining the solution $\boldsymbol{s}_*^L$ involves achieving lower-priority tasks while preserving the satisfaction of higher-priority task constraints. Specifically, at the hierarchical level aiming to achieve task $\mathcal{T}_{p+1}$, the solution space is modified while preserving the optimal solutions $\boldsymbol{s}_p$ already obtained for higher-priority tasks $\mathcal{T}_1, \cdots, \mathcal{T}_p$. This process is repeated N times in order of task priority, ultimately yielding the final solution $\boldsymbol{s}_*^L$.

For linear equality constraints, the solution space that does not interfere with task achievement can be explicitly defined. For a matrix $\mathbf{X}$, a matrix $\mathcal{N}(\mathbf{X})$ composed of basis vectors of its nullspace is defined. The nullspace basis matrix $\mathcal{N}(\mathbf{X})$ satisfies the relationship shown in Eq. (5.14).

$$\mathbf{X}\mathcal{N}(\mathbf{X}) = \mathbf{0} \tag{5.14}$$

When a vector $\boldsymbol{s}$ satisfies the linear equality constraint $\mathbf{A}\boldsymbol{s} - \boldsymbol{b} = \mathbf{0}$, the vector $\boldsymbol{s} + \mathcal{N}(\mathbf{A})\boldsymbol{z}$ also satisfies the same equality constraint for any vector $\boldsymbol{z}$, as demonstrated by Eq. (5.15).

$$\begin{aligned} \mathbf{A}(\boldsymbol{s} + \mathcal{N}(\mathbf{A})\boldsymbol{z}) - \boldsymbol{b} &= \mathbf{A}\boldsymbol{s} + \mathbf{A}\mathcal{N}(\mathbf{A})\boldsymbol{z} - \boldsymbol{b} \\ &= \mathbf{A}\boldsymbol{s} + \mathbf{0} - \boldsymbol{b} \\ &= \mathbf{0} \end{aligned} \tag{5.15}$$

This property can be extended to multiple tasks. When $\mathbf{Z}_p$ is defined as the nullspace basis matrix that preserves the equality constraints of tasks 1 through p, the nullspace basis matrix $\mathbf{Z}_{p+1}$ for tasks 1 through $p+1$ can be given as Eq. (5.16) [88].

$$\mathbf{Z}_{p+1} = \mathbf{Z}_p \mathcal{N}(\mathbf{A}_{p+1}\mathbf{Z}_p) \tag{5.16}$$

For linear inequality constraints, error variables are introduced for each task. Given the linear inequality constraint $\mathbf{D}_p\boldsymbol{s} - \boldsymbol{f}_p \leq \mathbf{0}$ for task $\mathcal{T}_p$, an error variable $\boldsymbol{e}_p$ is introduced to transform it into $\mathbf{D}_p\boldsymbol{s} - \boldsymbol{f}_p \leq \boldsymbol{e}_p$. In the hierarchical optimization problem, the objective becomes minimizing the error variable $\boldsymbol{e}_p$ while satisfying the inequality constraints of higher-priority tasks. When solving the optimization problem to determine $\boldsymbol{e}_p$ for task $\mathcal{T}_p$, the error variables $\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{p-1}$ for higher-priority tasks $\mathcal{T}_1, \cdots, \mathcal{T}_{p-1}$ are fixed to their previously optimized values. This ensures that the search for a solution satisfying task $\mathcal{T}_p$ is performed within a region that does not degrade the performance of the higher-priority tasks. It should be noted that the inequalities are not converted into equality constraints using slack variables, precisely to maintain a strict priority among the task inequalities.

The restriction of the solution set considering task priorities defined in Eq. (5.5) is achieved through two mechanisms: (1) The introduction of nullspace basis matrices ensures that higher-priority tasks are not compromised (Eqs. (5.15) and (5.16)), (2) The objective function aims to minimize deviations from both equality and inequality constraints

(Eq. (5.17)).

$$
\begin{aligned}
\underset{\boldsymbol{z}_{p+1},\boldsymbol{e}_{p+1}}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{A}_{p+1}(\boldsymbol{s}_p^L + \mathbf{Z}_p\boldsymbol{z}_{p+1}) - \boldsymbol{b}_{p+1}\|^2 + \frac{1}{2}\|\boldsymbol{e}_{p+1}\|^2 \\
\text{subject to} \quad & \mathbf{D}_{p+1}(\boldsymbol{s}_p^L + \mathbf{Z}_p\boldsymbol{z}_{p+1}) - \boldsymbol{f}_{p+1} \leq \boldsymbol{e}_{p+1} \\
& \mathbf{D}_p(\boldsymbol{s}_p^L + \mathbf{Z}_p\boldsymbol{z}_{p+1}) - \boldsymbol{f}_p \leq \boldsymbol{e}_p^* \\
& \qquad \vdots \\
& \mathbf{D}_1(\boldsymbol{s}_p^L + \mathbf{Z}_p\boldsymbol{z}_{p+1}) - \boldsymbol{f}_1 \leq \boldsymbol{e}_1^* \\
& \mathbf{0} \leq \boldsymbol{e}_{p+1}
\end{aligned}
\tag{5.17}
$$

The sequential solution of this optimization problem for $p = 1 \cdots N$ yields the solution $\boldsymbol{s}_*^L$ to the multi-objective optimization problem $\mathcal{P}^L$.

5.6 Sampling-Based Optimization (SBO)

This section explains how to solve Eq. (5.12) using sampling-based optimization to find the optimal parameter $\boldsymbol{V}_*$. The content of this section follows the theory of MPPI, and theoretical details regarding optimality proofs can be found in [80].

To handle the constraints in Eq. (5.12) more effectively, the problem is transformed into minimizing the expected value of a cost function $C\colon \boldsymbol{V} \to \mathbb{R}$ as shown in Eq. (5.18). Note that in this section, the variable $\boldsymbol{V}$ to be optimized is treated as a random variable. The random variable $\boldsymbol{V}$ follows a probability distribution $\mathbb{Q}_{\tilde{\boldsymbol{V}}}$ parameterized by its expected value $\tilde{\boldsymbol{V}}$.

$$
\boldsymbol{V}_* = \underset{\tilde{\boldsymbol{V}} \in \mathcal{V}}{\operatorname{argmin}} \, \mathbb{E}_{V \sim \mathbb{Q}_{\tilde{\boldsymbol{V}}}} [\, C(\boldsymbol{V}) \,], \tag{5.18}
$$

The cost function $C_l(\boldsymbol{V})$ for achieving a task $\mathcal{T}_l$ is defined using a constant w_b that adjusts the importance of equality and inequality constraints as follows:

$$
C_l(\boldsymbol{V}) = \|F_l^{\mathbf{N}}(\boldsymbol{s}_*^L(\boldsymbol{V}))\|_2 + G_b^{\mathbf{N}}(\boldsymbol{s}_*^L(\boldsymbol{V})), \tag{5.19}
$$

$$
G_b^{\mathbf{N}}(\boldsymbol{s}) = \begin{cases} 0 & \text{if } G_l^{\mathbf{N}}(\boldsymbol{s}) \leq 0 \\ w_b & \text{otherwise} \end{cases}. \tag{5.20}
$$

The cost function $C(\boldsymbol{V})$ for tasks $\mathcal{T}_m \cdots \mathcal{T}_N$ is defined in Eq. (5.21) as a weighted sum of individual task cost functions.

$$C(\boldsymbol{V}) = \sum_{l=m}^{N} w_l C_l(\boldsymbol{V}), \quad w_l \in \mathbb{R}. \tag{5.21}$$

However, since task priorities are handled through weighting, it is important to note that strict prioritization like HQP cannot be applied here. The sampling-based approach offers the advantage of handling diverse task representations, but introduces a limitation in the process of determining the parameter $\boldsymbol{V}$ by restricting prioritization to soft constraints. While accepting increased computational complexity, it is possible to find $\boldsymbol{V}$ that strictly satisfy critical tasks. This case can be interpreted as applying an infinite penalty for task violations.

To efficiently solve Eq. (5.18) through sampling, the problem is reformulated as a KL divergence minimization problem using variational inference. Let the prior distribution of random variable $\boldsymbol{V}$ be parameterized by mean U and covariance matrix $\boldsymbol{\Sigma}$ as constants,

$$\boldsymbol{V} = U + \mathcal{E}, \quad \mathcal{E} \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{\Sigma}), \tag{5.22}$$

where $\mathcal{E}$ follows a normal distribution with zero mean and covariance $\boldsymbol{\Sigma}$. This probability distribution is denoted as $\mathbb{Q}_{U,\boldsymbol{\Sigma}}$. In this case, the optimal distribution $q^*(\boldsymbol{V})$ of the random variable $\boldsymbol{V}$ is given by the product of the prior distribution $\mathbb{Q}_{U,\boldsymbol{\Sigma}}$ of $\boldsymbol{V}$ and a Boltzmann distribution with temperature parameter λ, as shown in Eqs. (5.23) and (5.24).

$$q^*(\boldsymbol{V}) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} C(\boldsymbol{V})\right) q(\boldsymbol{V}|U,\Sigma), \tag{5.23}$$

$$\eta = \int \exp\left(-\frac{1}{\lambda} C(\boldsymbol{V})\right) q(\boldsymbol{V}|U,\Sigma) d\boldsymbol{V}. \tag{5.24}$$

However, computing the optimal distribution $q^*(\boldsymbol{V})$ shown in Eqs. (5.23) and (5.24) is computationally intractable. Therefore, this approach transforms the problem into finding a normal distribution that is closest to $q^*(\boldsymbol{V})$ in terms of KL divergence. Let $\tilde{\boldsymbol{V}}$ denote the expected value of the random variable $\boldsymbol{V}$ to be optimized, $\mathbb{Q}_{\tilde{V}}$ represent the probability distribution Normal$(\tilde{\boldsymbol{V}}, \boldsymbol{\Sigma})$ of $\boldsymbol{V}$, and $\mathbb{Q}^*$ denote the optimal distribution $q^*(\boldsymbol{V})$. With this

notation, Eqs. (5.12) and (5.18) reduce to the KL divergence minimization problem shown in Eq. (5.25).

$$\boldsymbol{V}_* \approx \underset{\tilde{\boldsymbol{V}}}{\operatorname{argmin}} \operatorname{KL}(\mathbb{Q}^* \mid\mid \mathbb{Q}_{\tilde{\boldsymbol{V}}}). \tag{5.25}$$

However, it is important to note that $\boldsymbol{V}_*$ obtained from Eq. (5.25) differs from the solution that purely minimizes the expected value of the cost function $C(\boldsymbol{V})$ as shown in Eq. (5.18). Strictly speaking, it is the solution to the optimization problem shown on the right side of Eq. (5.26). Here, $\mathbb{P}$ refers to Normal$(\mathbf{0}, \boldsymbol{\Sigma})$. This transformation is a practical approach to approximately obtain $\boldsymbol{V}_*$ as a solution to a more tractable optimization problem, since directly solving Eq. (5.18) is computationally difficult.

$$\begin{aligned} \underset{\tilde{\boldsymbol{V}}}{\operatorname{argmin}} \operatorname{KL}(\mathbb{Q}^* \mid\mid \mathbb{Q}_{\tilde{\boldsymbol{V}}}) = & \underset{\tilde{\boldsymbol{V}} \in \mathcal{V}}{\operatorname{argmin}} \mathbb{E}_{V \sim \mathbb{Q}_{\tilde{\boldsymbol{V}}}} [\, C(\boldsymbol{V}) \,] \\ & + \lambda \cdot \operatorname{KL}(\mathbb{Q}_{\tilde{\boldsymbol{V}}} \mid\mid \mathbb{P}). \end{aligned} \tag{5.26}$$

Eq. (5.25) can be solved using importance sampling. For K samples $\boldsymbol{V}^{(k)} \sim \mathbb{Q}_{U,\boldsymbol{\Sigma}}, (k = 0, 1, \cdots, K-1)$, the weight for each sample is obtained as shown in Eq. (5.27).

$$w(\boldsymbol{V}^{(k)}) = \frac{1}{\eta} \exp\left(-\frac{1}{\lambda} C(\boldsymbol{V}^{(k)}) \right). \tag{5.27}$$

Finally, the optimal parameter $\boldsymbol{V}_*$ is obtained by taking the weighted average of K samples, as shown in Eq. (5.28).

$$\boldsymbol{V}_* = U + \sum_{k=1}^{K} w(\boldsymbol{V}^{(k)}) \mathcal{E}^{(k)}, \tag{5.28}$$

while $\boldsymbol{V}^{(k)}$ and $\mathcal{E}^{(k)}$ represent the k-th sample and noise, respectively.

MPPI was selected as the sampling-based optimization algorithm in this section due to its computational efficiency and practical applicability. However, since MPPI's inferred posterior distribution is constrained to a normal distribution, alternative optimization methods [95, 96] should be considered when dealing with more complex probability distributions.

5.7 Proposed Algorithm: SA-HQP

This section presents the proposed algorithm based on the content discussed from Section 5.3 to Section 5.6. Specifically, it details the procedure for solving the prioritized optimization problem that combines linear and nonlinear tasks, as shown in Eq. (5.8).

The solution procedure consists of four main steps. The detailed algorithm is presented in Algorithms 3 and 4.

Step 1: Sample Parameter $\boldsymbol{V}$

The parameter $\boldsymbol{V}$ for locally linearizing nonlinear tasks is sampled to obtain K samples $\boldsymbol{V}^{(k)}$ $(k = 0, 1, \ldots, K-1)$. These samples follow a normal distribution Normal$(U, \boldsymbol{\Sigma})$ (see Eq. (5.22)). The mean $\boldsymbol{U}$ and covariance matrix $\boldsymbol{\Sigma}$ are constant parameters.

Step 2: Calculate Evaluation Values for Each Sample

By treating parameter $\boldsymbol{V}$ as a precondition, the multi-objective optimization problem containing nonlinear tasks can be handled as a problem containing only local linear tasks. For each sample $\boldsymbol{V}^{(k)}$, the problem containing only linear tasks is solved using HQP, and the evaluation value $C(\boldsymbol{V}^{(k)})$ to be minimized is calculated. HQP is the procedure shown in Algorithm 4, which repeatedly solves the optimization problem (5.17) defined in Section 5.5 as many times as there are tasks. The solution obtained by solving HQP satisfies the task priorities based on the definitions in Eqs. (5.5) and (5.6). Note that this step can be computed independently for each sample, allowing for parallel processing to improve computational efficiency.

Step 3: Determine Optimal Parameter $\boldsymbol{V}_*$

The optimal parameter $\boldsymbol{V}_*$ is obtained by calculating weights $\mathbf{w}_k$ for each sample $\boldsymbol{V}^{(k)}$ based on their evaluation values $C(\boldsymbol{V}^{(k)})$, and then computing their weighted average.

Step 4: Obtain Final Solution $\boldsymbol{s}_*^L(\boldsymbol{V}_*)$

The final solution $\boldsymbol{s}_*^L(\boldsymbol{V}_*)$ of the multi-objective optimization is obtained by solving HQP using the optimal parameter $\boldsymbol{V}_*$ as a precondition.

Algorithm 3 Sampling-Augmented HQP (SA-HQP)

K: Number of samples;
$\boldsymbol{U}, \boldsymbol{\Sigma}$: Mean and covariance of sampling distribution;
λ, Cost: Parameter and cost function for SBO;
for $k \leftarrow 0\ to\ K-1$ **:**
  Sample $\mathcal{E}^{(k)} \sim \text{Normal}(\boldsymbol{0}, \boldsymbol{\Sigma})$;
  $\boldsymbol{V}^{(k)} \leftarrow \boldsymbol{U} + \mathcal{E}^{(k)}$;
  $\boldsymbol{s}_k \leftarrow \text{SolveHQP}(\boldsymbol{V}^{(k)})$; (Algorithm 4)
  $C_k \leftarrow \text{Cost}(\boldsymbol{s}_k)$;
$\rho \leftarrow \min_k[C_k]$; ▷ for avoiding numerical instability
$\eta \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}(C_k - \rho)\right)$; ▷ normalization factor
for $k \leftarrow 0\ to\ K-1$ **:** ▷ calculate sample weights
  $\mathrm{w}_k \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(C_k - \rho)\right)$;
$\boldsymbol{V}_* \leftarrow \boldsymbol{U} + \sum_{k=0}^{K-1} \mathrm{w}_k \mathcal{E}^{(k)}$; ▷ obtain the optimal via pose
$\boldsymbol{s}_* \leftarrow \text{SolveHQP}(\boldsymbol{V}_*)$;

Algorithm 4 Solve HQP

N: Number of tasks;
$\mathcal{T}_n$: Task n $(n = 1, \ldots, N)$ represented by A_n, b_n, D_n, f_n;
$\mathcal{L}$: initiailze list of v^* as empty;
$\boldsymbol{s}_0 \leftarrow \boldsymbol{0}$;
$\boldsymbol{Z}_0 \leftarrow \boldsymbol{I}$;
for $p \leftarrow 0\ to\ N-1$ **:**
  $\boldsymbol{z}_{p+1}^*, \boldsymbol{e}_{p+1}^* \leftarrow \text{SolveQP}(\boldsymbol{Z}_p, \mathcal{L}, \mathcal{T}_1, ..., \mathcal{T}_{p+1})$; (Prob.(5.17))
  $\mathcal{L} \leftarrow (\mathcal{L}, \boldsymbol{e}_{p+1}^*)$;
  $\boldsymbol{s}_{p+1} \leftarrow \boldsymbol{s}_p + \boldsymbol{Z}_p \boldsymbol{z}_{p+1}^*$;
  $\boldsymbol{Z}_{p+1} \leftarrow \boldsymbol{Z}_p \mathcal{N}(\boldsymbol{A}_{p+1} \boldsymbol{Z}_p)$;
return $\boldsymbol{s}_N$;

5.8 Comparison with Conventional MPC Formulation

This section clarifies the relationship between the conventional model predictive control (MPC) formulation and the proposed Sampling-Augmented Hierarchical Quadratic Programming (SA-HQP) framework. Additionally, the discussion addresses problem settings in which the proposed method demonstrates significant advantages, as well as current limitations associated with its application.

Section 2.3 presents the general formulation of MPC in Eqs. (2.1)–(2.4). To facilitate comparison with the proposed SA-HQP framework, a more concrete formulation of MPC is provided here. MPC determines an optimal control input sequence $\{\mathbf{u}_k\}_{k=0}^{T-1}$ along with the corresponding state trajectory $\{\mathbf{x}_k\}_{k=0}^{T}$ by solving an optimization problem. The objective is to minimize the sum of a terminal cost $\Phi(\mathbf{x}_T)$ and stage costs $L(\mathbf{x}_k, \mathbf{u}_k)$, subject to the system dynamics and inequality constraints $g(\mathbf{x}_k, \mathbf{u}_k) \leq 0$ and $g(\mathbf{x}_T) \leq 0$. The optimization problem is formulated as follows:

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \Phi(\mathbf{x}_T) + \sum_{k=0}^{T-1} L(\mathbf{x}_k, \mathbf{u}_k)\Delta t \\
\text{s.t.} \quad & \mathbf{x}_0 = \mathbf{x}_{\text{init}}, \\
& \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k), \quad \forall\, k \in \{0, 1, \ldots, T-1\} \\
& g(\mathbf{x}_k, \mathbf{u}_k) \leq 0, \quad \forall\, k \in \{0, 1, \ldots, T-1\} \\
& g(\mathbf{x}_T) \leq 0.
\end{aligned}
\tag{5.29}
$$

In the context of the SA-HQP framework, the conventional MPC formulation can be equivalently represented as shown in Eqs. (5.30)–(5.31). A key aspect of this representation is that the minimization of the objective function can be reformulated as an equality constraint by introducing as an ideal minimum cost value C_{min}. This approach is justified because the SA-HQP framework is designed to solve optimization problems defined in Eq. (5.17) that try to minimize deviations from equality constraints, even when these constraints cannot be satisfied exactly. Consequently, the conventional MPC formulation, which seeks to minimize an objective function subject to constraints, is interpreted within the SA-HQP framework as a problem involving two hierarchically prioritized tasks. Specifically, the first-priority task ($\mathcal{T}_1$) enforces the system dynamics and inequality constraints,

while the second-priority task ($\mathcal{T}_2$) addresses the minimization of the original cost function, as shown in Eq. (5.30).

$$\mathcal{P} = \begin{cases} \mathcal{T}_1 : F_1(\boldsymbol{s}) = \mathbf{0},\ G_1(\boldsymbol{s}) \leq \mathbf{0} \\ \mathcal{T}_2 : F_2(\boldsymbol{s}) = \mathbf{0}, \end{cases} \tag{5.30}$$

where the decision variable $\boldsymbol{s}$ stacks the entire state and control trajectories, i.e., $\boldsymbol{s} = [\mathbf{x}_0^\top, \mathbf{u}_0^\top, \ldots, \mathbf{x}_{T-1}^\top, \mathbf{u}_{T-1}^\top, \mathbf{x}_T^\top]^\top$. The tasks are then defined in detail as:

$$\begin{aligned} F_1(\boldsymbol{s}) &= \begin{bmatrix} \mathbf{x}_0 - \mathbf{x}_{\text{init}} \\ f(\mathbf{x}_0, \mathbf{u}_0) - \mathbf{x}_1 \\ \vdots \\ f(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) - \mathbf{x}_T \end{bmatrix}, \\ G_1(\boldsymbol{s}) &= \begin{bmatrix} g(\mathbf{x}_0, \mathbf{u}_0) \\ \vdots \\ g(\mathbf{x}_{T-1}, \mathbf{u}_{T-1}) \\ g(\mathbf{x}_T) \end{bmatrix}, \\ F_2(\boldsymbol{s}) &= \left[\left\{ \Phi(\mathbf{x}_T) + \textstyle\sum_{k=0}^{T-1} L(\mathbf{x}_k, \mathbf{u}_k) \right\} - C_{\min} \right]. \end{aligned} \tag{5.31}$$

In summary, the SA-HQP framework extends the conventional MPC formulation by introducing multi-level prioritized constraints. This approach generalizes conventional MPC and is not limited to strict prioritization. The framework also allows for defining an integrated task by weighting multiple objectives and formulating the result as a single equality constraint. This approach still supports nuanced trade-offs within a single priority level, much like a traditional cost function.

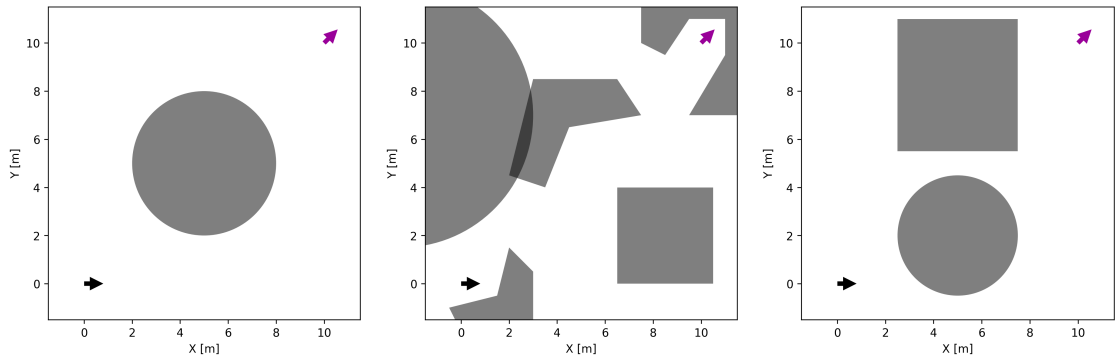
SA-HQP is particularly effective in problem settings where multiple control solutions exist to satisfy a critical, high-priority objective. This typically occurs in systems with a non-trivial null-space for the primary task, leaving room for optimization of secondary objectives. A primary example is a system with high degrees of freedom (DoF). For instance, an 8 DoF vehicle can follow a designated path while independently controlling its orientation. Path following is the high-priority task, and the vehicle's orientation can be managed as a lower-priority task by utilizing the redundant DoF. Another suitable scenario involves tasks with diverse temporal patterns, such as reaching a target state at a future time T. If the primary task for a robot on a 2D plane is to reach a target position in T seconds, numerous trajectories can fulfill this requirement. This redundancy allows for the consideration of

lower-priority tasks, such as selecting a smoother path. Conversely, in problem settings that lack such redundancy, few degrees of freedom remain for lower-priority tasks. In such cases, adding low-priority tasks may not change the final solution. The only viable approach is then to reduce the number of task hierarchies, causing the optimization problem to become similar to the conventional MPC formulation.

While the SA-HQP formulation is more general than conventional MPC, its current implementation does not necessarily outperform it across all problem types. A key limitation is that the strict null-space projection for prioritizing tasks is only applicable when tasks are expressed as linear equality or inequality constraints. Addressing nonlinear tasks requires the introduction of sampling-based optimization, which brings the drawbacks of increased computational cost and a soft consideration of task priorities. Specifically, to handle nonlinear system dynamics, the current framework must treat the entire state trajectory through sampling, offering little advantage over sampling-based MPC methods like MPPI. The SA-HQP framework is therefore most effective for problems characterized by linear dynamics that require simultaneous handling of multiple tasks, including those with nonlinear expressions. The navigation of the holonomic and 8-DoF vehicle models discussed in later sections meets this requirement, as their vehicle-centric motion can be predicted with linear expressions, making them prime use cases for the SA-HQP framework.

5.9 Motion Planning of Holonomic Vehicle with SA-HQP

This section presents motion planning simulations for holonomic vehicles to validate the effectiveness of the proposed method. Figure 5.3 shows three different maps with varying obstacle shapes and configurations. The vehicle aims to navigate from its initial position (black arrow, bottom left) to the goal position (purple arrow, top right) while avoiding obstacles. The motion planning optimization targets the sequence of vehicle states and velocities up to time step T (i.e., vector $\boldsymbol{s}$ defined in Eq. (5.1)). The target arrival time T is set to $T = 50$ for Map A, $T = 40$ for Map B, and $T = 60$ for Map C. These time horizons were selected as the minimum feasible values within achievable ranges.



(a) Map A (b) Map B (c) Map C

Figure 5.3: Three types of navigation maps. The gray regions are obstacles. The black arrows and the purple arrows indicate the initial poses and the goal poses, respectively.

5.9.1 Overview of Evaluation

To validate the effectiveness of the proposed method, the simulation results of three approaches are compared to address the following two evaluation questions(EQs):

[EQ1] Does considering nonlinear tasks contribute to performance improvement?

[EQ2] Does the framework of solving optimization problems by considering task priorities outperform methods that minimize a single cost function defined by weighted sums?

For comparison, the following three methods were selected:

[SA-HQP] Priority-based optimization handling nonlinear tasks

[HQP] Strict priority-based optimization handling only linear tasks

[MPPI] Optimization minimizing a single cost function expressed as a weighted sum of multiple tasks

The evaluation of EQ1 is conducted by comparing SA-HQP and HQP, while the evaluation of EQ2 is conducted by comparing SA-HQP and MPPI.

5.9.2 Setting Requirements, Tasks, and Evaluation Metrics

The vehicle requirements are to efficiently navigate through narrow spaces while avoiding obstacles of arbitrary shapes and reaching the target terminal state at the specified time. The definition of vector $\boldsymbol{s}$, which represents the solution to the motion planning problem, can be found in Eq. (5.1).

The following presents a list of requirements for vehicle motion planning and their corresponding evaluation metrics. All evaluation values are scalar quantities, where smaller values indicate better performance.

[Requirement 1] Satisfy the state update law
[Requirement 2] Satisfy velocity constraints
[Requirement 3] Satisfy acceleration constraints
[Requirement 4] Avoid collisions with arbitrary obstacles
[Requirement 5] Reach the terminal state
[Requirement 6] Minimize the trajectory length
[Requirement 7] Minimize the acceleration for smooth motion

The evaluation metric for Requirement 1 employs the L_2 norm of the state updating error. The discrete-time state update law for the omnidirectional vehicle with time step Δt is expressed as:

$$\boldsymbol{p}_{k+1} = \boldsymbol{p}_k + \boldsymbol{v}_k \Delta t \quad (k = 0, \cdots, T-1). \tag{5.32}$$

To ensure the motion planning results are feasible for the control system, minimal deviation from the state update law is desirable. The evaluation metric is defined as follows. When the motion planning solution $\boldsymbol{s}$ perfectly satisfies the state update law, the evaluation value $c_{\text{sys}}(\boldsymbol{s})$ becomes 0:

$$c_{\text{sys}}(\boldsymbol{s}) = \sum_{k=0}^{T-1} \|\boldsymbol{p}_{k+1} - (\boldsymbol{p}_k + \boldsymbol{v}_k \Delta t)\|_2 \tag{5.33}$$

The evaluation metric for Requirement 2 employs a penalty value $v_{\text{pen}} \in \mathbb{R}$ that is applied to control inputs $\boldsymbol{v}_k (k = 0, \cdots, T-1)$ when they exceed the defined velocity bounds

$\boldsymbol{v}_{\min} \in \mathbb{R}^3$ and maximum velocity $\boldsymbol{v}_{\max} \in \mathbb{R}^3$.

$$c_{v\text{lim}}(\boldsymbol{s}) = \sum_{k=0}^{T-1} \left(\begin{cases} 0 & \text{if } \boldsymbol{v}_{\min} \leq \boldsymbol{v}_k \leq \boldsymbol{v}_{\max} \\ v_{\text{pen}} & \text{otherwise} \end{cases} \right) \tag{5.34}$$

The evaluation metric for Requirement 3 implements a penalty system that assigns $a_{\text{pen}} \in \mathbb{R}$ to control inputs $\boldsymbol{v}_k (k = 1, \cdots, T-1)$ when they exceed the specified acceleration bounds $\boldsymbol{a}_{\min} \in \mathbb{R}^3$ and $\boldsymbol{a}_{\max} \in \mathbb{R}^3$.

$$c_{a\text{lim}}(\boldsymbol{s}) = \sum_{k=1}^{T-1} \left(\begin{cases} 0 & \text{if } \boldsymbol{a}_{\min} \leq \boldsymbol{a}_k \leq \boldsymbol{a}_{\max} \\ a_{\text{pen}} & \text{otherwise} \end{cases} \right). \tag{5.35}$$

Since the acceleration $\boldsymbol{a}_k$ is not included in the motion planning solution $\boldsymbol{s}$, it is calculated from the velocity $\boldsymbol{v}_k$ as follows:

$$\boldsymbol{a}_k = \frac{\boldsymbol{v}_k - \boldsymbol{v}_{k-1}}{\Delta t} \tag{5.36}$$

where Δt represents the discrete time step length.

The evaluation metric for Requirement 4 implements a binary penalty system, where o_{pen} is assigned for collision cases and 0 for collision-free trajectories. This simulation employs a simplified collision model that evaluates only the vehicle's center point position, disregarding the vehicle's physical dimensions. A collision is registered when the center point enters the obstacle region $\mathcal{O}_{\text{obs}}$ defined in the global coordinate frame.

$$c_{\text{obs}}(\boldsymbol{s}) = \begin{cases} 0 & \text{if } \boldsymbol{p}_k \notin \mathcal{O}_{\text{obs}} \quad (k = 0, \cdots, T) \\ o_{\text{pen}} & \text{otherwise} \end{cases} \tag{5.37}$$

The evaluation metric for Requirement 5 employs the L_2 norm to quantify the discrepancy between the motion plan's terminal state $\boldsymbol{p}_T$ and the desired goal pose $\boldsymbol{p}_g$.

$$c_{\text{goal}}(\boldsymbol{s}) = \|\boldsymbol{p}_T - \boldsymbol{p}_g\|_2 \tag{5.38}$$

The evaluation metric for Requirement 6 employs the total path length c_{len} of the planned vehicle trajectory.

$$c_{\text{len}}(\boldsymbol{s}) = \sum_{k=1}^{T} \sqrt{(p_x^k - p_x^{k-1})^2 + (p_y^k - p_y^{k-1})^2} \tag{5.39}$$

The evaluation metric for Requirement 7 employs the squared sum of the vehicle's center acceleration $\boldsymbol{a}_k$ calculated using Eq. (5.36).

$$c_{\mathrm{smo}}(\boldsymbol{s}) = \sum_{k=0}^{T-1} \boldsymbol{a}_k^\top \boldsymbol{a}_k \tag{5.40}$$

5.9.3 Motion Planner Design

SA-HQP: Proposed Method

In implementing SA-HQP, the requirements defined in Section 5.9.2 were prioritized and decomposed into tasks, resulting in the task list presented in Table 5.1. The prioritization strategy focuses on excluding solutions that need not be explored. For instance, the highest priority task $\mathcal{T}_1^L$ excludes solutions that are infeasible for the controlled system. By assigning higher priority to $\mathcal{T}_2^{\boldsymbol{N}}$ than $\mathcal{T}_3^L$, solutions that reach the target terminal state while colliding with obstacles are excluded.

The next step involves transforming the problem in Table 5.1 to obtain an approximate solution, as the original formulation contains nonlinear tasks that are difficult to address directly. The nonlinear tasks of obstacle avoidance ($\mathcal{T}_2^{\boldsymbol{N}}$) and trajectory length minimization ($\mathcal{T}_5^{\boldsymbol{N}}$) are converted to linear tasks by introducing vehicle states $\boldsymbol{p}_a^{\boldsymbol{V}}, \boldsymbol{p}_b^{\boldsymbol{V}}, \boldsymbol{p}_c^{\boldsymbol{V}}$ as parameters that should be matched at specific time steps a, b, c $(0 \leq a < b < c \leq T)$. The transformed task list is presented in Table 5.2, and the specific definitions of each task

$\mathcal{T}_1^L, \mathcal{T}_2^{\boldsymbol{N}\to L}, \mathcal{T}_3^L, \mathcal{T}_4^{\boldsymbol{N}\to L}$ are shown below. $\boldsymbol{p}_{\text{init}}$ represents the vehicle's initial state.

$$
\begin{aligned}
\mathcal{T}_1^L: \quad & \begin{cases} \boldsymbol{p}_0 = \boldsymbol{p}_{\text{init}}, \\ \boldsymbol{p}_k = \boldsymbol{p}_{k-1} + \boldsymbol{v}_{k-1}\Delta t, \\ \qquad (k = 1, \cdots, T). \\ \boldsymbol{v}_{\min} \le \boldsymbol{v}_k \le \boldsymbol{v}_{\max}, \\ \qquad (k = 0, \cdots, T-1). \\ \boldsymbol{a}_{\min} \le \frac{\boldsymbol{v}_k - \boldsymbol{v}_{k-1}}{\Delta t} \le \boldsymbol{a}_{\max}, \\ \qquad (k = 1, \cdots, T-1). \end{cases} \\
\mathcal{T}_2^{\boldsymbol{N}\to L}: \quad & \begin{cases} \boldsymbol{p}_a = \boldsymbol{p}_a^{\boldsymbol{V}} \\ \boldsymbol{p}_b = \boldsymbol{p}_b^{\boldsymbol{V}} \\ \boldsymbol{p}_c = \boldsymbol{p}_c^{\boldsymbol{V}} \end{cases} \\
\mathcal{T}_3^L: \quad & \begin{cases} \boldsymbol{p}_T = \boldsymbol{p}_g \end{cases} \\
\mathcal{T}_4^{\boldsymbol{N}\to L}: \quad & \begin{cases} \frac{\boldsymbol{v}_k - \boldsymbol{v}_{k-1}}{\Delta t} = 0 \quad (k = 1, \cdots, T-1) \end{cases}
\end{aligned}
$$

Eq. (5.41) presents the cost function used to search for parameters that satisfy the specified conditions. The cost function $\text{Cost}(\boldsymbol{s})$ referenced in Algorithm 3 corresponds to $c_{\text{sbo}}(\boldsymbol{s})$ in Eq. (5.41).

$$
\begin{aligned}
c_{\text{sbo}}(\boldsymbol{s}) = w_{\text{obs}} c_{\text{obs}}(\boldsymbol{s}) + w_{\text{goal}} c_{\text{goal}}(\boldsymbol{s}) \\
+ w_{\text{len}} c_{\text{len}}(\boldsymbol{s}) + w_{\text{smo}} c_{\text{smo}}(\boldsymbol{s}).
\end{aligned} \tag{5.41}
$$

As an implementation strategy, instead of directly sampling the vehicle state $[p_x, p_y, \theta] \in \mathbb{R}^3$, parameters $[\alpha, \beta, \gamma] \in \mathbb{R}^3$ are sampled relative to the straight line connecting the initial and terminal states, defining parameter $\boldsymbol{V}$. The relationship between the search parameters and vehicle state is shown in Eqs. (5.42) – (5.44). This transformation enables efficient sampling of the state space where waypoints are more likely to exist.

$$p_x = p_x^0 + \alpha(p_x^\top - p_x^0), \tag{5.42}$$

$$p_y = p_y^0 + \beta(p_y^\top - p_y^0), \tag{5.43}$$

$$\theta = \tan^{-1} \frac{p_y^\top - p_y^0}{p_x^\top - p_x^0} + \gamma. \tag{5.44}$$

Table 5.1: Task list of SA-HQP for Trajectory Optimization

Task	Description	Requirements
$\mathcal{T}_1^L$	State update law, velocity bounds, acceleration bounds	1, 2, 3
$\mathcal{T}_2^{\boldsymbol{N}}$	Collision avoidance with arbitrary obstacles	4
$\mathcal{T}_3^L$	Reach the terminal state	5
$\mathcal{T}_4^{\boldsymbol{N}}$	Minimize acceleration and trajectory length	6, 7

Table 5.2: Linearized task list of SA-HQP for Trajectory Optimization

Task	Description
$\mathcal{T}_1^L$	State update law, velocity bounds, acceleration bounds
$\mathcal{T}_2^{\boldsymbol{N}\to L}$	Via state $\boldsymbol{V}$ tracking
$\mathcal{T}_3^L$	Reach the terminal state
$\mathcal{T}_4^{\boldsymbol{N}\to L}$	Minimize acceleration

HQP: Conventional Method 1

To implement HQP, a prioritized list of linear tasks must be prepared. The list containing nonlinear tasks (Table 5.1) is converted into a list of linear tasks only (Table 5.3) through simple local linear approximation. The collision avoidance with arbitrary obstacles is linearized into a task of avoiding straight wall surfaces. The trajectory length minimization is linearized into a task of zeroing the velocity input.

MPPI: Conventional Method 2

MPPI [80] is a sampling-based optimization method that determines control input sequences by minimizing an cost function through variational inference. To implement MPPI, an cost function must be prepared to quantify the quality of control inputs. The cost function $c_{\mathrm{mppi}}(\boldsymbol{s})$ is constructed by weighted summation of the evaluation metrics defined in Section 5.9.2.

$$\begin{aligned} c_{\mathrm{mppi}}(\boldsymbol{s}) = {} & w_{\mathrm{goal}} c_{\mathrm{goal}}(\boldsymbol{s}) + w_{v\mathrm{lim}} c_{v\mathrm{lim}}(\boldsymbol{s}) + w_{a\mathrm{lim}} c_{a\mathrm{lim}}(\boldsymbol{s}) \\ & + w_{\mathrm{obs}} c_{\mathrm{obs}}(\boldsymbol{s}) + w_{\mathrm{len}} c_{\mathrm{len}}(\boldsymbol{s}) + w_{\mathrm{smo}} c_{\mathrm{smo}}(\boldsymbol{s}). \end{aligned} \tag{5.45}$$

Table 5.3: Linear task list of HQP

Task	Description
$\mathcal{T}_1^L$	State update law, velocity bounds, acceleration bounds
$\mathcal{T}_2^{\boldsymbol{N}\to L}$	Obstacle avoidance (linearized)
$\mathcal{T}_3^L$	Reach the terminal state
$\mathcal{T}_4^{\boldsymbol{N}\to L}$	Acceleration minimization, trajectory length minimization (linearized)

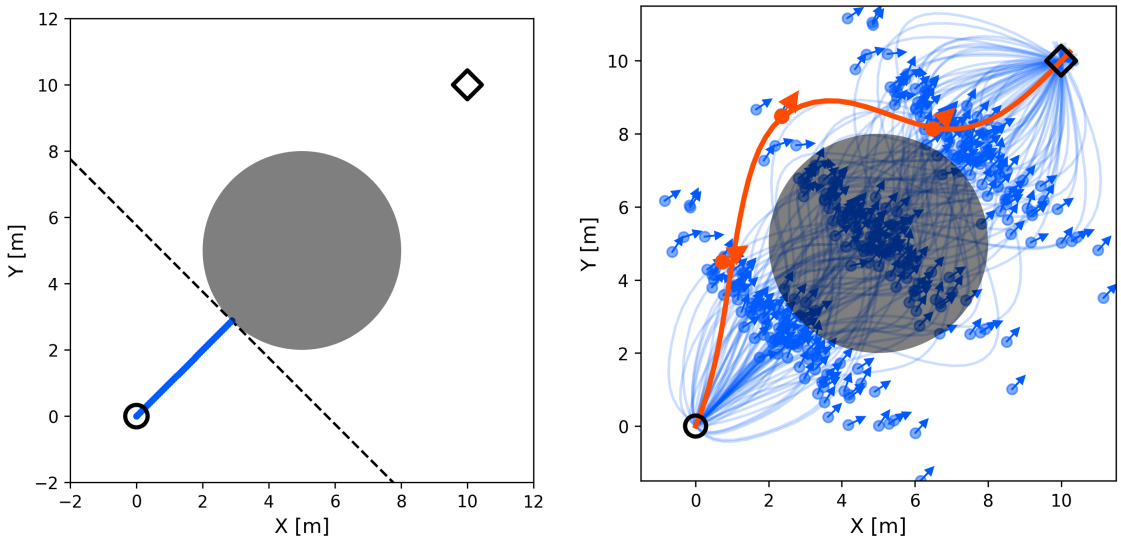
Table 5.4: Comparison of SA-HQP and MPPI Solutions. Smaller costs are better, and better values are highlighted in bold letters.

Cost	Map B		Map C	
	SA-HQP	MPPI	SA-HQP	MPPI
c_{sys} [↓]	0.0	0.0	0.0	0.0
c_{goal} [↓]	**0.38**	2.3	**0.32**	5.4
$c_{v\text{lim}}$ [↓]	0.0	0.0	0.0	0.0
$c_{a\text{lim}}$ [↓]	**0.0**	50.0	**0.0**	2700.0
c_{obs} [↓]	0.0	0.0	0.0	0.0
c_{len} [↓]	14.8	**12.3**	**16.7**	20.7
c_{smo} [↓]	10.6	**5.9**	**14.6**	73.2
c_{mppi} [↓]	**901.7**	3839.4	**477.3**	53319.7

5.9.4 Comparison of Motion Planning Results

To verify the main interests of the verification, the motion planning results calculated by the three methods are compared, and the answers to evaluation questions EQ1 and EQ2 are obtained.

To verify the advantages of SA-HQP in handling nonlinear tasks, the motion planning results of SA-HQP and HQP in Map A (Fig. 5.4) are compared. Since HQP can only handle linear tasks, it approximates circular obstacles as linear walls, resulting in failure to reach the target terminal state. In contrast, SA-HQP achieves more accurate approximation of nonlinear tasks through the introduction of sampling methods, enabling it to obtain a solution that reaches the target terminal state while avoiding obstacles. These results demonstrate that considering nonlinear tasks contributes to meeting the motion planning requirements (answer to EQ1).



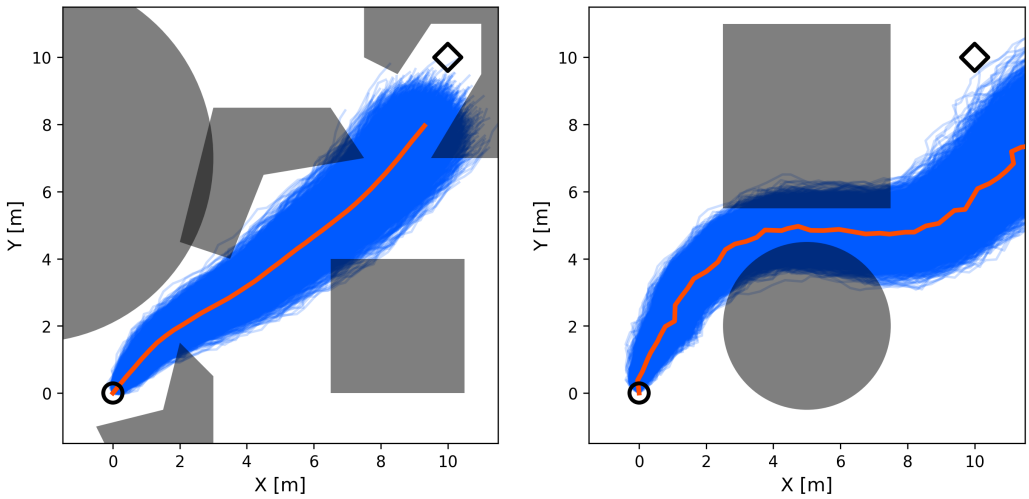
(a) HQP (b) SA-HQP

Figure 5.4: Comparison of HQP and SA-HQP in Map A.

To evaluate the effectiveness of optimization problem decomposition through prioritization, the motion planning results of SA-HQP and MPPI in Map B and Map C are compared. Table 5.4 presents the evaluation metrics for the obtained motion planning solutions. The trajectories of SA-HQP and MPPI motion planning are visualized in Fig. 5.6 and Fig. 5.5, respectively. The results of SA-HQP motion planning for Map B and Map C are shown in Fig. 5.9 and Fig. 5.10, respectively. The results of MPPI motion planning for Map B and Map C are shown in Fig. 5.7 and Fig. 5.8, respectively. The evaluation values in Table 5.4 indicate that both methods satisfy the requirements for state update rules, velocity constraints, and obstacle avoidance, with no significant differences. A notable difference emerges in achieving the target terminal state, where SA-HQP achieves more precise attainment compared to MPPI (see Fig. 5.6 and Fig. 5.5). While SA-HQP never violates acceleration constraints, MPPI fails to maintain acceleration limits (see Fig. 5.9, Fig. 5.10, Fig. 5.7, and Fig. 5.8). Comparing the temporal evolution of SA-HQP solutions (Fig. 5.9, Fig. 5.10) and MPPI solutions (Fig. 5.7, Fig. 5.8), SA-HQP exhibits smooth value transitions, whereas MPPI shows discontinuous and abrupt changes.

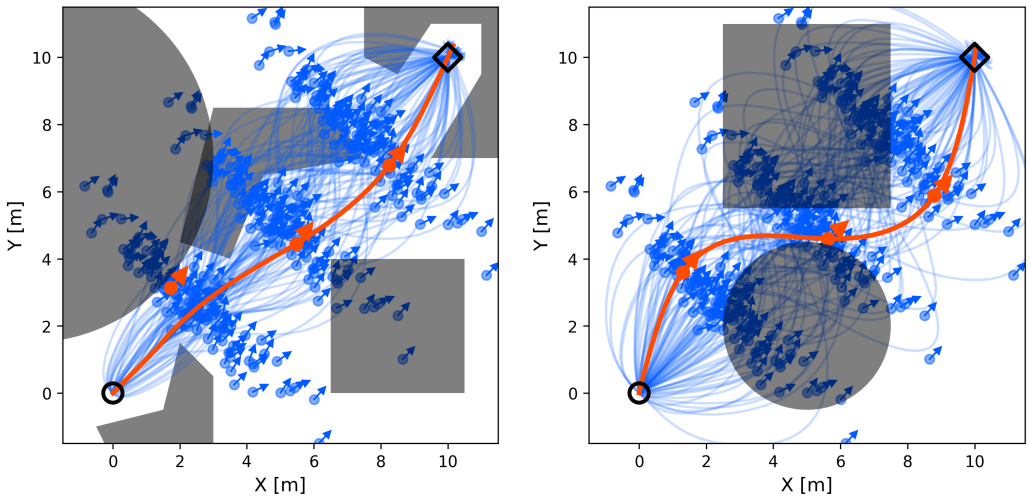
When comparing the optimality of both methods using MPPI's cost function c_{mppi}, SA-

HQP, which solves problems decomposed by prioritization, obtained a better solution than MPPI, which directly minimizes c_{mppi} (answer to EQ 2).



(a) Map B (b) Map C

Figure 5.5: MPPI Trajectories in Map B and Map C.



(a) Map B (b) Map C

Figure 5.6: SA-HQP Trajectories in Map B and Map C.

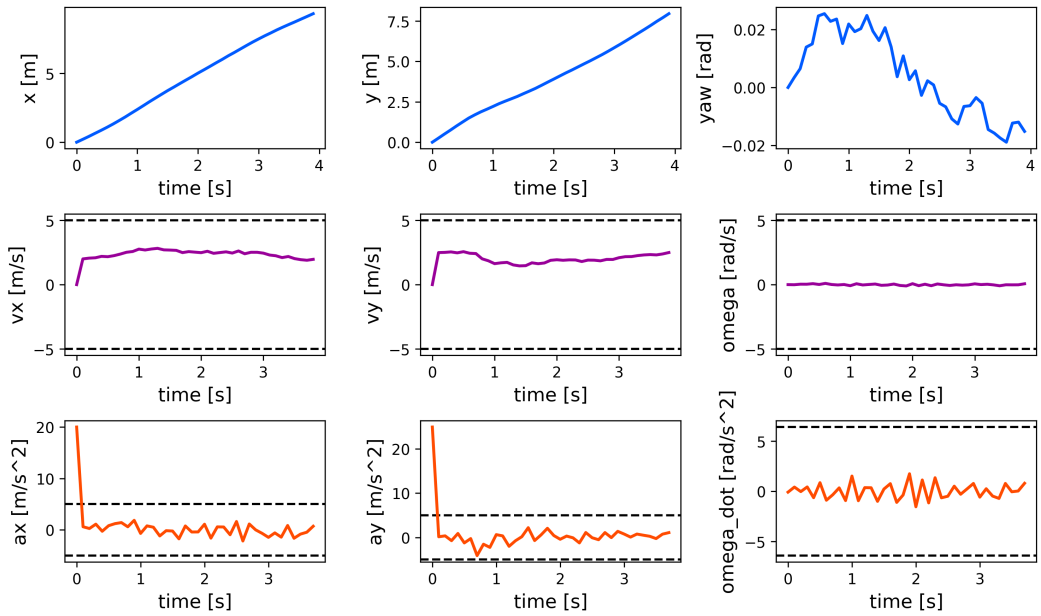


Figure 5.7: MPPI Motion Planning Result in Map B

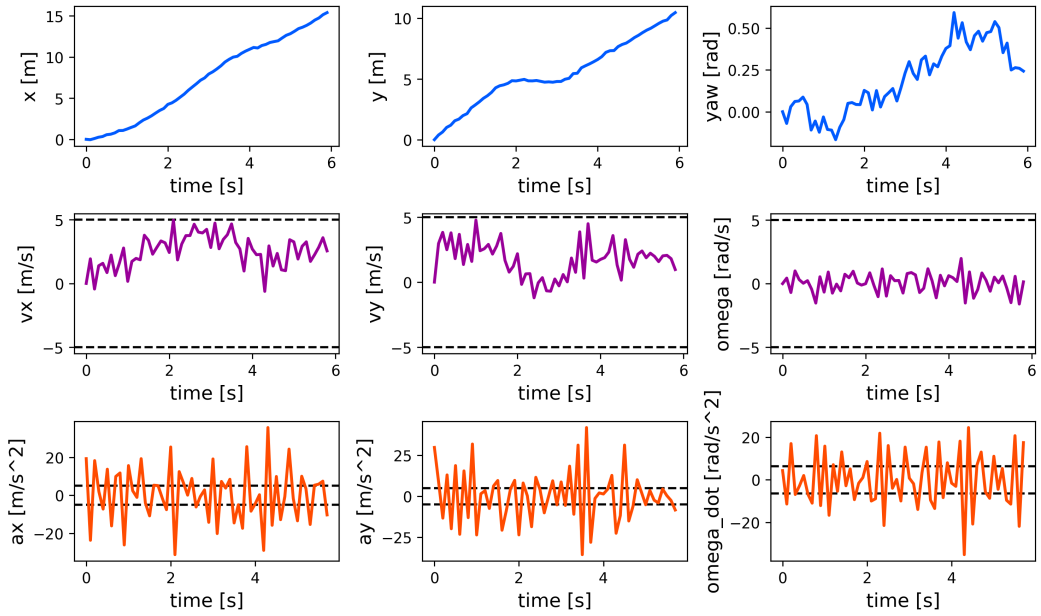


Figure 5.8: MPPI Motion Planning Result in Map C

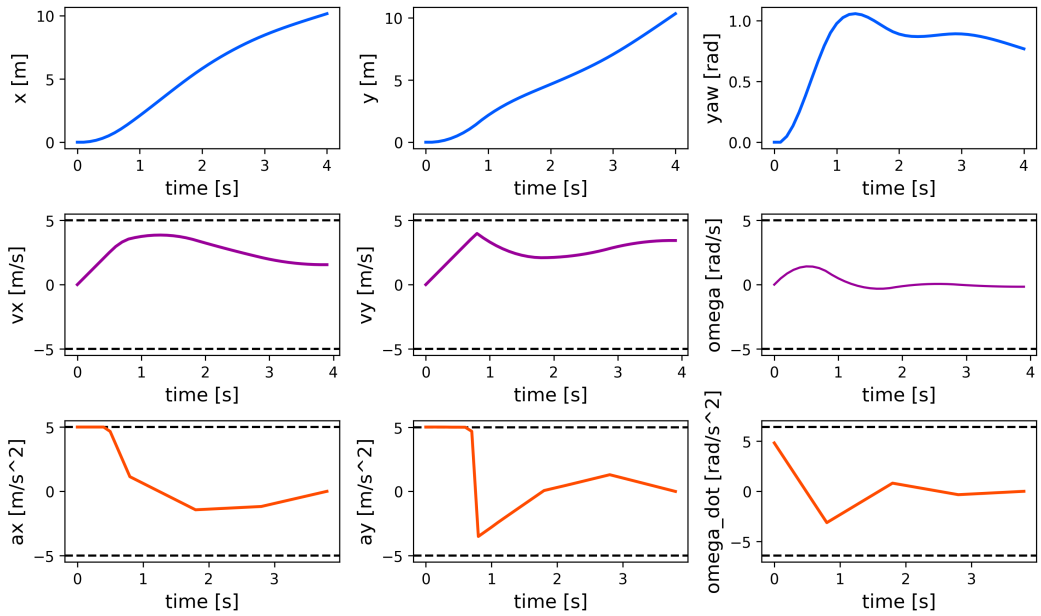


Figure 5.9: SA-HQP Motion Planning in Map B

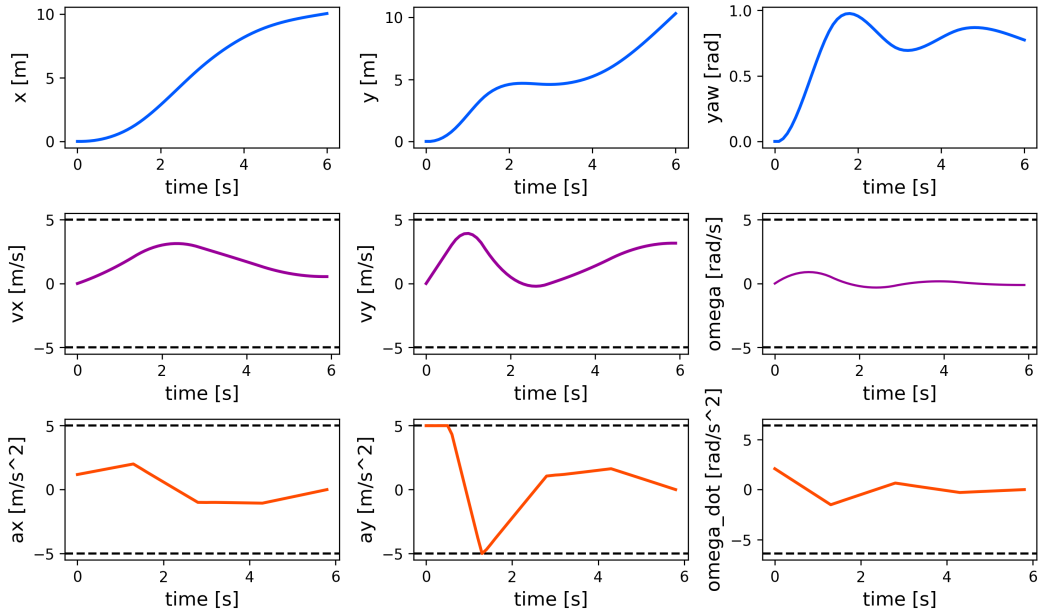
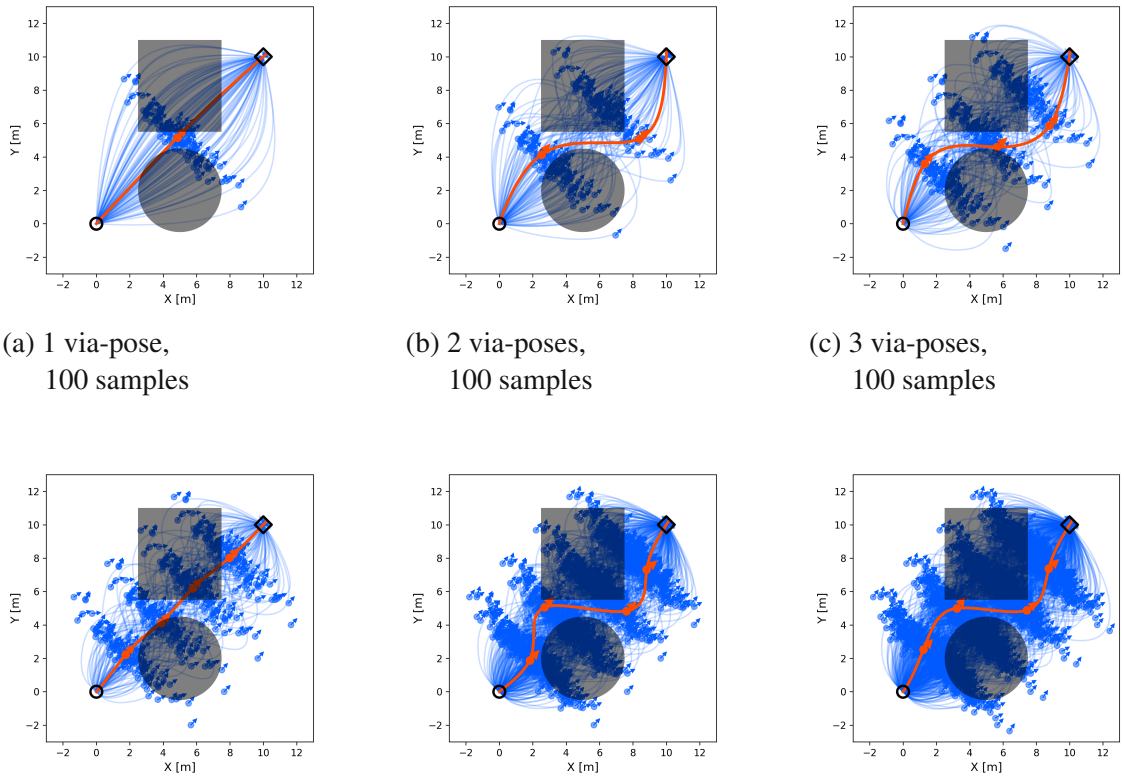


Figure 5.10: SA-HQP Motion Planning in Map C



(a) 1 via-pose, 100 samples

(b) 2 via-poses, 100 samples

(c) 3 via-poses, 100 samples

(d) 4 via-poses, 100 samples

(e) 4 via-poses, 300 samples

(f) 4 via-poses, 500 samples

Figure 5.11: SA-HQP trajectories in Map C with different numbers of via-poses and samples.

An analysis of SA-HQP's superior performance compared to MPPI reveals key insights. The motion planning simulation demonstrates that SA-HQP achieved more faithful requirement satisfaction than MPPI while utilizing only one-hundredth of the samples (100 versus 10,000). This sample efficiency advantage stems from SA-HQP's substantially reduced sampling space dimensionality. MPPI must explore control inputs across all time steps ($\mathbb{R}^{3T}$), whereas SA-HQP samples only three via-points ($\mathbb{R}^9$). The critical finding is that this constrained sampling space provided sufficient expressiveness for the given problem requirements. This dimensionality reduction was accomplished through strategic problem decomposition via task prioritization. Notably, SA-HQP incurs higher per-sample compu-

tational cost due to the need to solve multiple QP problems per sample.

The number of via-points presents a trade-off in the optimization process. Too few via-points make it difficult to adequately approximate nonlinear tasks, while too many unnecessarily expand the sampling space and degrade exploration efficiency. To investigate this relationship, Fig. 5.11 presents the results of applying SA-HQP to Map C with varying numbers of via-points and samples. With insufficient via-points in Fig. 5.11a, the system failed to avoid obstacles. However, increasing to two (Fig. 5.11b) and three (Fig. 5.11c) via-points enabled successful obstacle avoidance. Fig. 5.11d demonstrates that 100 samples proved insufficient for exploring the solution space with four via-points, resulting in failed obstacle avoidance. As shown in Figs. 5.11e and 5.11f, increasing the sample count to 300 or 500 enabled successful obstacle avoidance even with four via-points.

Sampling-based optimization methods provide versatility across diverse problem domains but face inherent limitations in computational efficiency as dimensionality increases (curse of dimensionality [94]). Gradient-based optimization approaches exhibit limitations when addressing non-convex or non-differentiable problems but demonstrate superior performance in high-dimensional spaces when appropriate conditions are satisfied. These gradient-based methods excel particularly in managing hard constraint satisfaction. The SA-HQP framework achieves computational efficiency through systematic decomposition of multi-objective control problems using task prioritization. This strategic decomposition enables the assignment of each subproblem to the appropriate algorithm based on problem characteristics.

5.10 Nullspace MPC as Receding Horizon Extension of SA-HQP and Application to 4WIDS Vehicle Navigation

This section introduces Nullspace MPC, a receding horizon extension of SA-HQP, for real-time navigation of 4WIDS vehicles. The navigation performance of Nullspace MPC is evaluated through comparison with the MPPI controller presented in Chapter 4.

5.10.1 Setting Requirements and Tasks

The control requirements are first enumerated to serve as the basis for defining the control objectives. The controller design follows a similar approach to the trajectory optimization solved by SA-HQP in Section 5.9.2. However, unlike the previous trajectory optimization, the following requirements are modified to accommodate real-time navigation demands and computational performance considerations.

[Requirement 1] Satisfy the state update law
[Requirement 2] Satisfy velocity constraints
[Requirement 3] Avoid collisions with arbitrary obstacles
[Requirement 4] Follow a reference path
[Requirement 5] Minimize the trajectory length
[Requirement 6] Minimize the acceleration for smooth motion

The requirements are transformed into the task list shown in Table 5.5. Subsequently, nonlinear tasks are converted into local linear tasks listed in Table 5.6 by parameterizing the vehicle via states $\boldsymbol{V}$. The tasks listed in Table 5.6 are defined as follows. For $\mathcal{T}_1^L$, the initial state is set to the observed vehicle state $\boldsymbol{p}_{\text{obs}}$, and the vehicle's state update law is constrained to follow holonomic motion while respecting velocity limits. For $\mathcal{T}_2^{\boldsymbol{N}\to L}$, the states that the vehicle should pass through are determined based on parameter $\boldsymbol{V}$. The number of waypoints is set to three, with a, b, c $(0 \leq a < b < c \leq T)$ representing their respective time steps. For $\mathcal{T}_3^{\boldsymbol{N}\to L}$, the vehicle's acceleration and trajectory length are minimized while balancing their weights. Note that trajectory length minimization is linearly approximated as vehicle center velocity minimization for ease of computation.

Table 5.5: Task list of Nullspace MPC for 4WIDS vehicle navigation

Task	Description	Requirements
$\mathcal{T}_1^L$	State update law, velocity bounds	1, 2
$\mathcal{T}_2^{\boldsymbol{N}}$	Collision avoidance with arbitrary obstacles	3
$\mathcal{T}_3^{\boldsymbol{N}}$	Follow a reference path, Short Trajectory, Smooth motion	4, 5, 6

Table 5.6: Linearized task list of Nullspace MPC for 4WIDS vehicle navigation

Task	Description
$\mathcal{T}_1^L$	State update law, velocity bounds
$\mathcal{T}_2^{\boldsymbol{N}\to L}$	Via state $\boldsymbol{V}$ tracking
$\mathcal{T}_3^{\boldsymbol{N}\to L}$	Minimize acceleration and trajectory length

$$
\mathcal{T}_1^L: \quad \begin{cases} \boldsymbol{p}_0 = \boldsymbol{p}_{\text{obs}}, \\ \boldsymbol{p}_k = \boldsymbol{p}_{k-1} + \boldsymbol{v}_{k-1}\Delta t, \quad (k = 1, \cdots, T). \\ \boldsymbol{v}_{\min} \le \boldsymbol{v}_k \le \boldsymbol{v}_{\max}, \quad (k = 0, \cdots, T-1). \end{cases}
$$

$$
\mathcal{T}_2^{\boldsymbol{N}\to L}: \quad \begin{cases} \boldsymbol{p}_a = \boldsymbol{p}_a^{\boldsymbol{V}} \\ \boldsymbol{p}_b = \boldsymbol{p}_b^{\boldsymbol{V}} \\ \boldsymbol{p}_c = \boldsymbol{p}_c^{\boldsymbol{V}} \end{cases}
$$

$$
\mathcal{T}_3^{\boldsymbol{N}\to L}: \quad \left\{ w_{\text{acc}} \cdot \left(\frac{\boldsymbol{v}_k - \boldsymbol{v}_{k-1}}{\Delta t} \right)^2 + w_{\text{len}} \cdot \left(v_x^k \right)^2 + w_{\text{len}} \cdot \left(v_y^k \right)^2 \quad (k = 0, \cdots, T-1). \right.
$$

Eq. (5.46) presents the cost function used to search for optimal parameters $\boldsymbol{V}$ (representing via states) through Sampling-based Optimization (SBO). The cost function c_{sbo} corresponds to Cost($\boldsymbol{s}$) in Algorithm 5. The definitions of each cost function are described in Section 4.6.2.

$$
\begin{aligned}
c_{\text{sbo}}(\boldsymbol{s}) = {} & w_{\text{collision}} c_{\text{collision}}(\boldsymbol{s}) + w_{\text{cmd}} c_{\text{cmd}}(\boldsymbol{s}) \\
& + w_{\text{dist}} c_{\text{dist}}(\boldsymbol{s}) + w_{\text{angle}} c_{\text{angle}}(\boldsymbol{s}) + w_{\text{speed}} c_{\text{speed}}(\boldsymbol{s})
\end{aligned} \tag{5.46}
$$

Algorithm 5 Nullspace MPC Applied to 4WIDS Vehicle Navigation

K: Number of samples;
$\boldsymbol{U}, \boldsymbol{\Sigma}$: Mean and covariance of sampling distribution;
λ, Cost: Parameter and cost function for SBO;
$\mathcal{R}$: Reference path;
$\boldsymbol{p}_{\text{obs}} \leftarrow$ ObserveVehicleState();
for $k \leftarrow 0\ to\ K-1$ **:**
    $\boldsymbol{V}^{(k)} \leftarrow$ SampleViaStates();
    $\boldsymbol{s}_k \leftarrow$ SolveHQP($\boldsymbol{V}^{(k)}, \boldsymbol{p}_{\text{obs}}$); (Algorithm 4)
    $C_k \leftarrow$ Cost($\boldsymbol{s}_k, \mathcal{R}$);
$\rho \leftarrow \min_k[C_k]$; ▷ for avoiding numerical instability
$\eta \leftarrow \sum_{k=0}^{K-1} \exp\left(-\frac{1}{\lambda}(C_k - \rho)\right)$; ▷ normalization factor
for $k \leftarrow 0\ to\ K-1$ **:** ▷ calculate sample weights
    $\mathbf{w}_k \leftarrow \frac{1}{\eta} \exp\left(-\frac{1}{\lambda}(C_k - \rho)\right)$;
$\boldsymbol{V}_* \leftarrow \boldsymbol{U} + \sum_{k=0}^{K-1} \mathbf{w}_k \mathcal{E}^{(k)}$; ▷ obtain the optimal via pose
$\boldsymbol{s}_* \leftarrow$ SolveHQP($\boldsymbol{V}_*$);
$\boldsymbol{u}_{\text{vehicle}} \leftarrow$ ConvertTo8DoFVehicleCommand($\boldsymbol{s}_*$);
SendToVehicleActuators($\boldsymbol{u}_{\text{vehicle}}$); ▷ send the control input to vehicle actuators.

5.10.2 Algorithm Description for 4WIDS Vehicle Navigation

This section presents the algorithmic framework of Nullspace MPC, which extends SA-HQP to a receding horizon formulation. The algorithm consists of sequential steps that transform a navigation problem into a series of local trajectory optimization tasks. At each control cycle, the algorithm generates an optimal control sequence for the vehicle while considering the prioritized tasks defined in Section 5.10.1. The following describes each step of the algorithm, from state observation to control input generation. The detailed algorithm is shown in Algorithm 5.

Step 1: Observe Current State

The step 1 involves observing the current state in the Global Frame of the vehicle. The vehicle state at time t consists of position p_x, p_y and heading angle θ in the Global Frame, along with translational velocity v_x, v_y and angular velocity ω in the Base Frame.

Step 2: Set a Local Goal

Unlike trajectory optimization where the goal state is predetermined, navigation tasks lack a clear terminal goal state. Instead, a reference path $\mathcal{R}$ is given as a sequence of waypoints:

$$\mathcal{R} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{N_{\text{ref}}}\} \tag{5.47}$$

where each waypoint $\mathbf{p}_i = [p_x^i, p_y^i, \theta_i]^\top$ contains position and orientation information in the Global Frame. The reference path is generated using the A* algorithm [97] to find the shortest obstacle-avoiding path reaching a specified goal location.

Nullspace MPC sets a local goal on the reference path $\mathcal{R}$ as the ideal terminal state at the end of the prediction horizon. Given a reference velocity v_{ref}, prediction horizon T, and time step Δt, the local goal is set at a distance $L = v_{\text{ref}} \cdot T \cdot \Delta t$ ahead of the current position along the reference path.

Step 3: Sample Via Poses

Candidate via poses for the vehicle trajectory are sampled randomly. If a probability distribution with unbounded support is selected and an infinite number of samples are drawn, the optimal via points will inevitably be included among the sampled candidates. However, in practical applications with a finite number of samples, it is essential to center the sampling distribution as close as possible to the optimal via points in order to ensure efficient exploration.

Specifically, a Gaussian distribution is employed with its mean centered on the poses that would be achieved through ideal reference path tracking (See Fig. 5.12c).

When an ideal via pose on the reference path is $\boldsymbol{p}_{\text{ideal}} = [p_x^{\text{ideal}}, p_y^{\text{ideal}}, \theta^{\text{ideal}}]^\top$, the probability distribution of the corresponding via pose is given by:

$$\boldsymbol{p}_{\text{via}} \sim \mathcal{N}\left(\begin{bmatrix} p_x^{\text{ideal}} \\ p_y^{\text{ideal}} \\ \theta^{\text{ideal}} \end{bmatrix}, \text{diag}\left(\sigma_x \cos\theta^{\text{ideal}} - \sigma_y \sin\theta^{\text{ideal}}, \sigma_x \sin\theta^{\text{ideal}} + \sigma_y \cos\theta^{\text{ideal}}, \sigma_\theta\right)\right), \tag{5.48}$$

where $\sigma_x, \sigma_y, \sigma_\theta$ are the standard deviations of the positions and heading angle, respectively.

The number of via poses per trajectory represents a trade-off between computational efficiency and solution accuracy. While increasing the number of via poses may yield more precise solutions, it also requires a larger number of samples for exploration. A detailed analysis of the number of via poses is presented in Section 5.9.4.

Step 4: Sample Trajectories Passing Through Via Poses

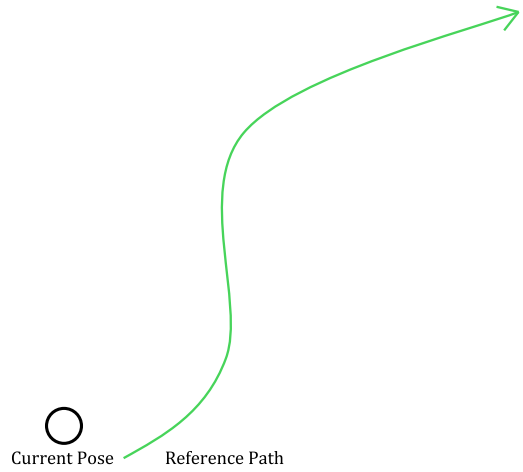
Smooth vehicle trajectories passing through the sampled via poses are computed using HQP to satisfy the prioritized linear tasks in Table 5.6. As shown in Fig. 5.12d, smooth trajectories are computed that pass through the sampled via points. For detailed algorithm of HQP, refer to Algorithm 4.

Step 5: Evaluate Trajectories and Calculate Weight for Each Sample

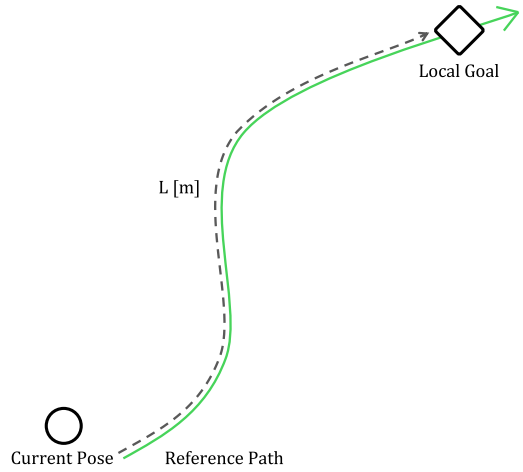
The sampled trajectories are evaluated and normalized weights are calculated for each sample. These weights are used to infer the optimal probability distribution of via points that minimizes the expected value of the cost c_{sbo} defined in Eq. (5.46). This inference process is called as variational inference, which infers a posterior distribution (Fig. 5.13b) that minimizes the cost from a prior distribution (Fig. 5.13a) where via points were sampled in advance. The specific procedure utilizes importance sampling, as detailed in Algorithm 5.

Step 6: Obtain Optimal Trajectory and Control Input

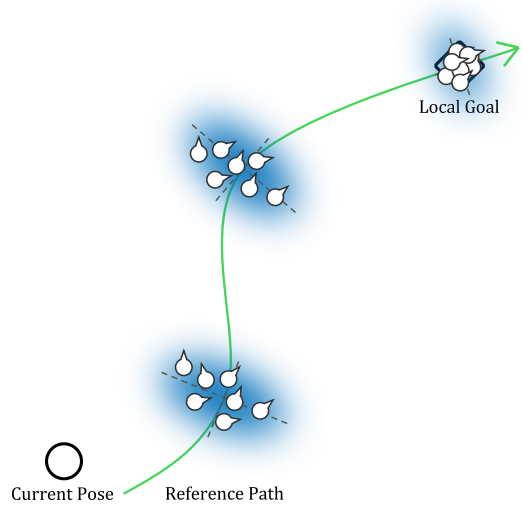
Once the distribution of via points that minimizes the expected cost is obtained, the mean of this distribution is treated as the optimal via point (See Fig. 5.13c). Finally, the optimal trajectory passing through these optimal via poses is computed using HQP, which constitutes the optimal solution obtained through Nullspace MPC (See Fig. 5.13d). The obtained vehicle center velocity is converted to the 8 DoF control input using the method described in Section 4.4.2, and applied to the vehicle.



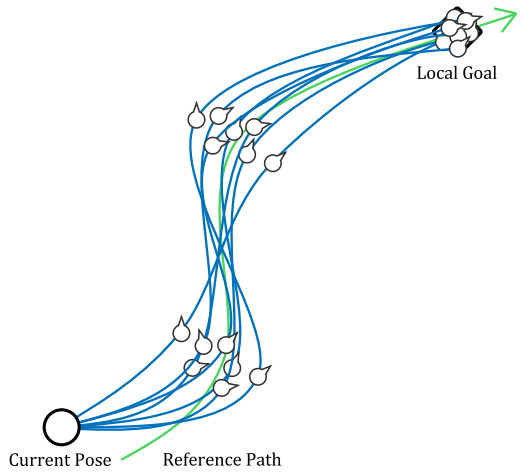
(a) Step 1, current state and reference path



(b) Step 2, local goal setting

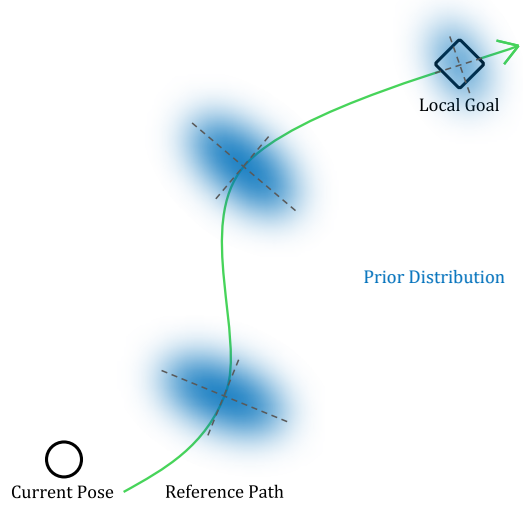


(c) Step 3, sample via poses

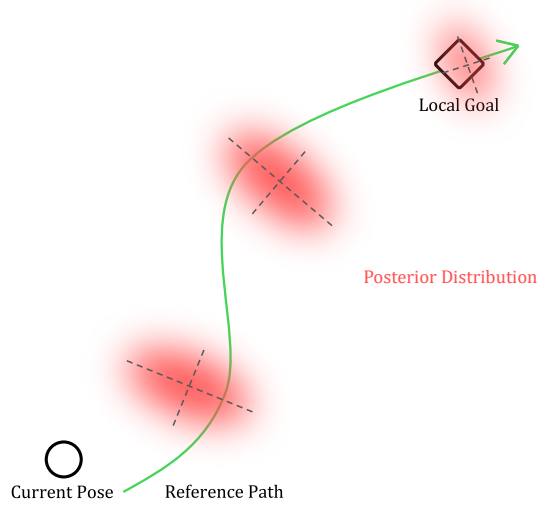


(d) Step 4, sample trajectories

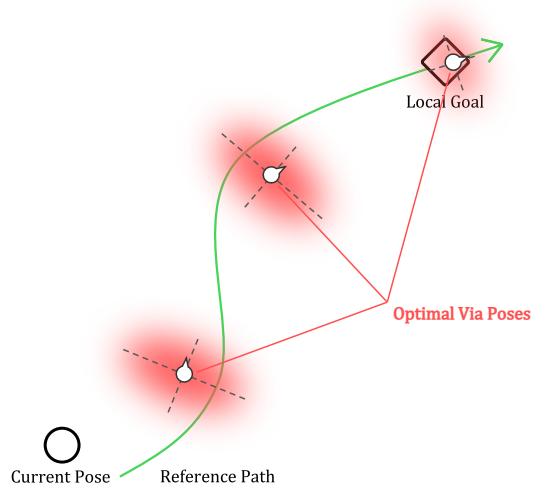
Figure 5.12: Visualization of the navigation algorithm with Nullspace MPC (Step 1-4)



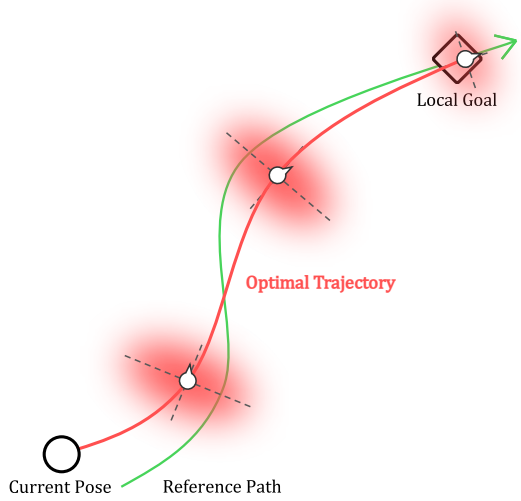
(a) Step 5, prior distribution



(b) Step 5, posterior distribution

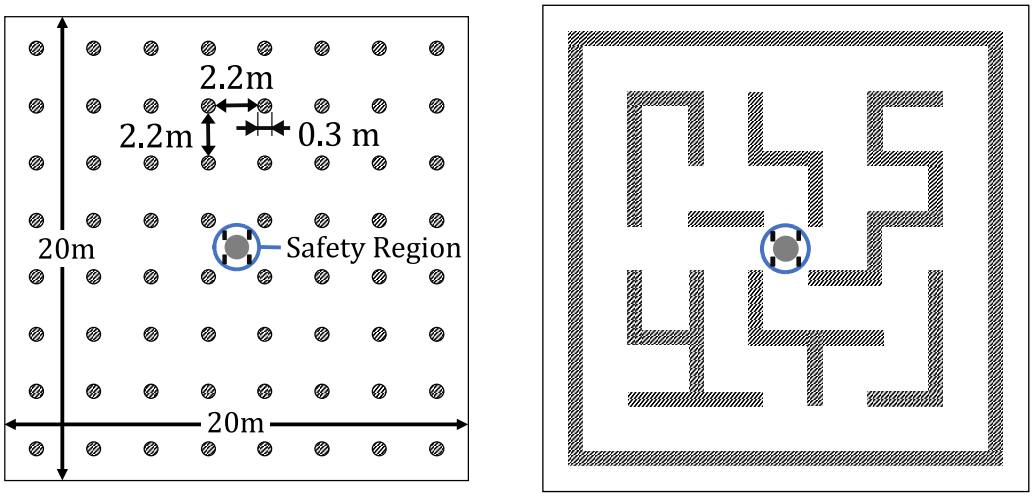


(c) Step 6, optimal via poses



(d) Step 6, optimal trajectory

Figure 5.13: Visualization of the navigation algorithm with Nullspace MPC (Step 5-8)



(a) Cylinder Garden (Easier Environment) (b) Maze (Harder Environment)

Figure 5.14: Simulation environments for evaluating navigation performance (reproduced from Chapter 4). The 8 DoF vehicle must navigate through dense obstacle fields and narrow passages while reaching sequential goals.

5.10.3 Comparison of Navigation Performance Between MPPI and Nullspace MPC

The navigation performance of MPPI and Nullspace MPC is evaluated through comparative experiments with 4WIDS vehicles. The objective is to leverage the vehicle's high degrees of freedom to achieve efficient and safe navigation through narrow environments. As the baseline method, MPPI-H is employed, which was proposed in Chapter 4 and achieves a balance between safety and efficiency through its sampling space switching mechanism.

The simulation environments consist of two maps identical to those used in Chapter 4 (see Fig. 5.14). Each episode requires reaching 10 sequential goals, and 100 episodes are conducted for each map. The success rate is evaluated as the proportion of episodes where all goals are reached without any failure. Other metrics are calculated using the average values from episodes where both MPPI and Nullspace MPC succeed, ensuring a fair comparison. An episode is considered failed if the vehicle experiences prolonged immobilization, collision, or loss of position estimation due to tipping. Note that minor collisions that do not hinder navigation are not considered failures.

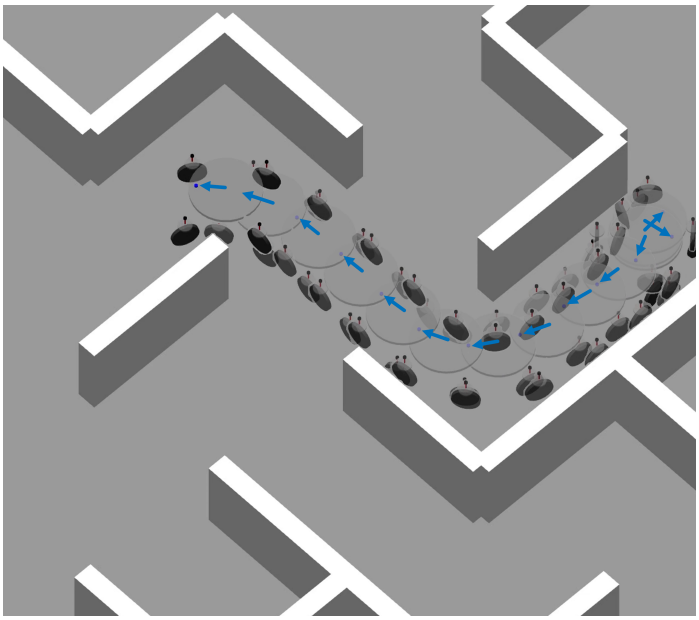


Figure 5.15: Frame-by-frame visualization of 4WIDS vehicle navigation using MPPI-H

The motion patterns of the 4WIDS vehicle exhibit distinct characteristics between the two control approaches. A significant behavioral divergence emerges when the vehicle must navigate in a direction opposite to its current orientation (see Figs. 5.15 and 5.16). MPPI-H employs a sequential approach, it first executes an in-place rotation to align the vehicle's orientation with the reference path, followed by forward motion. Nullspace MPC implements a concurrent strategy, simultaneously translating the vehicle's center along the reference path while progressively adjusting its orientation along the reference path. This simultaneous execution of translation and rotation allows Nullspace MPC to achieve higher velocities by eliminating orientation-related motion interruptions.

The evaluation results are summarized in Tables 5.7 and 5.8. The performance trends show similar patterns across both the Cylinder Garden and Maze environments.

Nullspace MPC demonstrates superior performance across three key metrics: Trajectory Length, Episode Time, and Success Rate. In the Cylinder Garden environment, Nullspace MPC achieves a 13.7% reduction in trajectory length and a 52.1% reduction in episode time compared to MPPI. In the Maze environment, Nullspace MPC achieves a 10.7% reduction in trajectory length and a 35.7% reduction in episode time. While

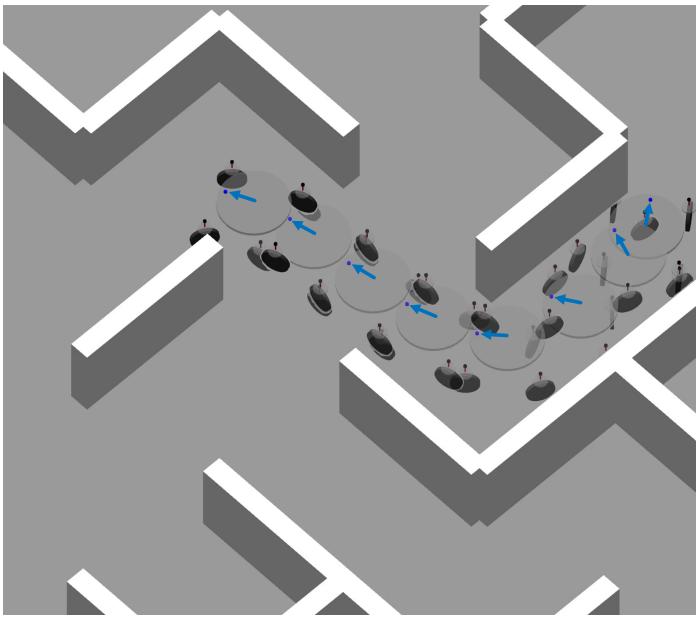


Figure 5.16: Frame-by-frame visualization of 4WIDS vehicle navigation using Nullspace MPC

Nullspace MPC maintains perfect 100% success rates in both environments, MPPI achieves 88% in Cylinder Garden and 94% in Maze, indicating occasional navigation failures.

MPPI-H exhibits superior computational efficiency. Both methods maintain real-time control within the 0.05s control interval, but MPPI-H achieves computation times over 10 times faster than Nullspace MPC. The computational advantage of MPPI-H stems from its simpler trajectory evaluation process. In contrast, Nullspace MPC requires solving multiple quadratic programming problems for each sample trajectory, despite its ability to reduce the sampling space. This computational overhead represents a key limitation that warrants further investigation.

Table 5.7: Evaluation Results of 100 Navigation Episodes in Cylinder Garden
bold value: better result

Metric	MPPI-H (3D(b)/4D)	Nullspace MPC
Calc. Time [ms] ↓	**3.6**	41.9
Trajectory Length [m] ↓	116.5	**100.5**
Episode Time [s] ↓	122.4	**58.6**
Success Rate [%] ↑	88	**100**

Note that minor collisions that do not hinder navigation are not considered failures.

Table 5.8: Evaluation Results of 100 Navigation Episodes in Maze
bold value: better result

Metric	MPPI-H (3D(b)/4D)	Nullspace MPC
Calc. Time [ms] ↓	**3.2**	39.3
Trajectory Length [m] ↓	142.5	**127.2**
Episode Time [s] ↓	111.9	**71.9**
Success Rate [%] ↑	94	**100**

Note that minor collisions that do not hinder navigation are not considered failures.

5.11 Discussion and Insights on Q2-β

This chapter presents an enhanced framework addressing research question **[Q2-β] How can the handling of multiple tasks be enhanced?**. The proposed Sampling Augmented Hierarchical Quadratic Programming (SA-HQP) algorithm decomposes optimization problems based on task priorities. Extension of this algorithm to a receding horizon framework, termed Nullspace MPC, enables successful navigation for 8 DoF vehicles. Nullspace MPC achieves comparable safety performance to MPPI while demonstrating superior navigation speed through narrow spaces.

In the comparison of one-shot trajectory optimization between MPPI and SA-HQP, SA-HQP demonstrated superior performance in achieving critical tasks such as reaching the goal and satisfying the acceleration constraint. MPPI attempts to accomplish multiple tasks by minimizing a weighted scalar objective function, but this approach becomes increasingly challenging to balance as the number of tasks grows. A significant limitation of this approach is that when balance adjustment fails, lower-priority tasks can interfere with the achievement of higher-priority tasks. The SA-HQP approach, which explicitly decomposes tasks based on priority levels, guarantees the achievement of important tasks even as the number of tasks increases. This hierarchical decomposition strategy suggests potential advantages for multi-objective control problems with many tasks.

A significant limitation of Nullspace MPC lies in its computational complexity. While the decomposition of optimization problems typically reduces the number of samples compared to MPPI, the requirement to solve multiple Quadratic Programming problems per sample leads to increased computational overhead.

Chapter 6

Conclusion

This chapter summarizes the key findings and contributions of this dissertation, and discusses future research directions.

6.1 Summary of Contributions to All Research Questions

This section revisits the main research question of this dissertation and summarizes the key findings and contributions that address it. The main research question of this dissertation is: **[Main Q] How can autonomous driving control be realized to exploit the performance potential of diversifying mobility systems?** Chapter 1 suggests that Model Predictive Control (MPC) is the most appropriate approach to achieve the research goal, and the dissertation focused on improving both the prediction and optimization aspects of MPC.

As for the prediction side, **[Q2-α] How can practical prediction models be designed?** is addressed in Chapter 3. The quantitative comparison of path-following performance using multiple prediction models demonstrates that more complex models do not always yield better results. Models that ignores tire slip (KAM, KBM) demonstrated superior path-following performance at low speeds compared to models that explicitly consider tire slip (DBM, DBM-L). This finding suggests the importance of utilizing appropriately simplified models based on the target task requirements. For models that explicitly consider slip, a solution was proposed to address the invalidity at zero velocity by modifying the

function structure. This approach enables the proposed model DBM-L to achieve broader applicability in low-speed scenarios compared to the original DBM model.

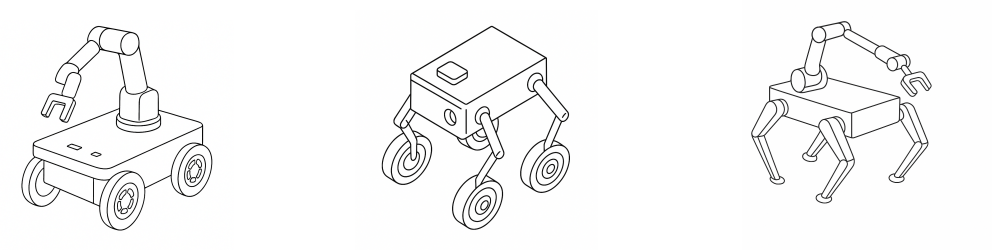
As for the optimization side, **[Q2-β] How can the handling of multiple tasks be enhanced?** is addressed in Chapter 4 and Chapter 5. For narrow-space navigation with an 8 DoF vehicle, two approaches were implemented: an improved MPPI-based method and a novel optimization algorithm called Nullspace MPC. Regarding multi-task achievement, MPPI employs an approach that minimizes a weighted sum of cost functions, while Nullspace MPC adopts a strategy of dividing the optimization problem using task priorities. The results demonstrate that Nullspace MPC enables faster navigation through narrow spaces compared to MPPI.

6.2 Discussion and Future Work

This dissertation emphasizes the development of practical MPC prediction models by focusing on simplified model designs that is complex enough to achieve the target task. Certain tasks such as vehicle drift control [98,99], however, necessitate precise vehicle dynamics and modeling of tire slip. These implementations remain challenging problems to be realized in practical applications. One potential solution involves developing technologies for accurate real-time physical parameter estimation [100, 101]. An alternative approach is to implement motion planning that avoids situations with high prediction uncertainty. For instance, while predicting vehicle motion on icy roads is difficult, the system can choose alternative routes with better traction conditions. This represents an integrated approach combining state estimation and motion planning [102, 103], which will be crucial for addressing diverse real-world environments.

While Nullspace MPC demonstrated superior performance in narrow-space navigation for 8 DoF vehicles, its application to more complex control systems(ex. Fig. 6.1) remains challenging within the current framework. Sampling-based approaches for handling nonlinear tasks provide a powerful means of expanding applicability, but they increase computational complexity and introduce difficulties in strictly maintaining task priorities. Therefore, tasks that can be expressed linearly should be solved within the Hierarchical Quadratic Programming (HQP) framework without relying on sampling when possible.

Even for nonlinear tasks, it would be beneficial to handle them within the HQP framework without sampling if appropriate mappings to linear spaces can be established. Learning such beneficial nonlinear mappings through neural networks represents a highly promising direction for expanding the applicability of the framework.



(a) Vehicle with a manipulator (b) Legged robot with wheels (c) Legged robot with a manipulator

Figure 6.1: Examples of complex systems with multiple degrees of freedom.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my academic advisor, Prof. Tatsuya Suzuki. I could not have come this far without his encouragement, thoughtful guidance, and considerate support throughout my doctoral journey. I am also deeply thankful to Prof. Hiroyuki Okuda, whose invaluable advice and generous support has enriched my research and academic growth. I am sincerely grateful to all my co-authors — Tatsuya Suzuki, Hiroyuki Okuda, Kohei Honda, Akira Ito, and Daisuke Nagasaka — for their invaluable collaboration and contributions to the core research presented in this dissertation. Working together with them has been an inspiring experience. Special thanks are due to Ms. Noriko Takada for her dedicated administrative support, which greatly eased the daily logistics of my research life. I would also like to express my appreciation to all the reviewers whose thoughtful feedback helped refine my publications and elevate the quality of my research. My sincere thanks go to the organizations that supported my research through grants and scholarships —Nagoya University, Rinnai Scholarship Foundation, Japan Society for the Promotion of Science (JSPS), and the Graduate Program for Lifestyle Revolution based on Transdisciplinary Mobility Innovation (TMI). Lastly, I am deeply grateful to the professors, research staff, and lab members of the Suzuki Laboratory for their daily support, inspiring discussions, and for providing a warm and stimulating research environment throughout my time as a doctoral student.

References

[1] Toyota Motor Corporation. Lexus RZ Series. https://lexus.jp/models/rz/, 2025. Accessed: 2025-05-08.

[2] Mercedes-Benz. G-TURN of electric G-Class Vehicle. https://www.youtube.com/watch?v=Yp_Kjyghckc, 2024. Official YouTube video by Mercedes-Benz. Accessed: 2025-05-08.

[3] Hyundai Mobis. Hyundai Mobis' e-Corner System Demonstration. https://www.youtube.com/watch?v=Bbw_smfOgVA, 2023. Official YouTube video by Hyundai Mobis. Accessed: 2025-05-08.

[4] Zvi Artstein. Stabilization with relaxed controls. *Nonlinear Analysis: Theory, Methods & Applications*, 7(11):1163–1173, 1983.

[5] Eduardo D. Sontag. A 'universal' construction of artstein's theorem on nonlinear stabilization. *Systems & Control Letters*, 13(2):117–123, 1989.

[6] Aydin Yesildirek and Bara Imran. Nonlinear control of quadrotor using multi lyapunov functions. In *2014 American Control Conference*, pages 3844–3849, 2014.

[7] Hai Zhao, Hongjiu Yang, Zhengyu Wang, and Hongbo Li. Nonlinear switched model predictive control with multiple lyapunov functions for trajectory tracking and obstacle avoidance of nonholonomic systems. *International Journal of Robust and Nonlinear Control*, 33(11):6171–6187, 2023.

[8] Hind Laghmara, Moustapha Doumiati, Reine Talj, and Ali Charara. Yaw moment lyapunov based control for in-wheel-motor-drive electric vehicle. *IFAC-PapersOnLine*, 50(1):13828–13833, 2017.

[9] Boqian Li, Shiping Wen, Zheng Yan, Guanghui Wen, and Tingwen Huang. A survey on the control lyapunov function and control barrier function for nonlinear-affine control systems. *IEEE/CAA Journal of Automatica Sinica*, 10(3):584–602, 2023.

[10] Manfred Morari and Jay H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4):667–682, 1999.

[11] Max Schwenzer, Muzaffer Ay, Thomas Bergs, and Dirk Abel. Review on model predictive control: an engineering perspective. *The International Journal of Advanced Manufacturing Technology*, 117(5):1327–1349, 2021.

[12] Sébastien Gros, Mario Zanon, Rien Quirynen, Alberto Bemporad, and Moritz Diehl and. From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, 93(1):62–80, 2020.

[13] Davor Hrovat, Stefano Di Cairano, H Eric Tseng, and Ilya V Kolmanovsky. The development of model predictive control in automotive industry: A survey. In *2012 IEEE International Conference on Control Applications*, pages 295–302. IEEE, 2012.

[14] Michael Neunert, Markus Stäuble, Markus Giftthaler, Carmine D. Bellicoso, Jan Carius, Christian Gehring, Marco Hutter, and Jonas Buchli. Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465, 2018.

[15] Nicola Scianca, Daniele De Simone, Leonardo Lanari, and Giuseppe Oriolo. Mpc for humanoid gait generation: Stability and feasibility. *IEEE Transactions on Robotics*, 36(4):1171–1188, 2020.

[16] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[17] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[18] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015.

[20] Guillaume Bellegarda, Yiyu Chen, Zhuochen Liu, and Quan Nguyen. Robust high-speed running for quadruped robots via deep reinforcement learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10364–10370, 2022.

[21] Yuanpei Chen, Tianhao Wu, Shengjie Wang, Xidong Feng, Jiechuan Jiang, Zongqing Lu, Stephen McAleer, Hao Dong, Song-Chun Zhu, and Yaodong Yang. Towards human-level bimanual dexterous manipulation with reinforcement learning. *Advances in Neural Information Processing Systems*, 35:5150–5163, 2022.

[22] Jennifer Andersson, Kenneth Bodin, Daniel Lindmark, Martin Servin, and Erik Wallin. Reinforcement learning control of a forestry crane manipulator. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2121–2126, 2021.

[23] Mike Zhang, Yuntao Ma, Takahiro Miki, and Marco Hutter. Learning to open and traverse doors with a legged manipulator. *arXiv preprint arXiv:2409.04882*, 2024.

[24] Rishabh Jangir, Guillem Alenyà, and Carme Torras. Dynamic cloth manipulation with deep reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4630–4636, 2020.

[25] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

[26] Erica Salvato, Gianfranco Fenu, Eric Medvet, and Felice Andrea Pellegrino. Crossing the reality gap: A survey on sim-to-real transferability of robot controllers in reinforcement learning. *IEEE Access*, 9:153171–153187, 2021.

[27] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2), April 2017.

[28] Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54(12):7173–7186, 2024.

[29] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, pages 1199–1210. PMLR, 2023.

[30] Yuzhe Qin, Hao Su, and Xiaolong Wang. From one hand to multiple hands: Imitation learning for dexterous manipulation from single-camera teleoperation. *IEEE Robotics and Automation Letters*, 7(4):10873–10881, 2022.

[31] Eli Bronstein, Mark Palatucci, Dominik Notz, Brandyn White, Alex Kuefler, Yiren Lu, Supratik Paul, Payam Nikdel, Paul Mougin, Hongge Chen, Justin Fu, Austin Abrams, Punit Shah, Evan Racah, Benjamin Frenkel, Shimon Whiteson, and Dragomir Anguelov. Hierarchical model-based imitation learning for planning in autonomous driving. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8652–8659, 2022.

[32] Anthony Hu, Gianluca Corrado, Nicolas Griffiths, Zachary Murez, Corina Gurau, Hudson Yeo, Alex Kendall, Roberto Cipolla, and Jamie Shotton. Model-based imitation learning for urban driving. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave,

K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 20703–20716. Curran Associates, Inc., 2022.

[33] Yunho Kim, Hyunsik Oh, Jeonghyun Lee, Jinhyeok Choi, Gwanghyeon Ji, Moonkyu Jung, Donghoon Youm, and Jemin Hwangbo. Not only rewards but also constraints: Applications on legged robot locomotion. *IEEE Transactions on Robotics*, 40:2984–3003, 2024.

[34] Yiming Zhang, Quan Vuong, and Keith Ross. First order constrained optimization in policy space. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15338–15349. Curran Associates, Inc., 2020.

[35] Tantan Zhang, Yueshuo Sun, Yazhou Wang, Bai Li, Yonglin Tian, and Fei-Yue Wang. A survey of vehicle dynamics modeling methods for autonomous racing: Theoretical models, physical/virtual platforms, and perspectives. *IEEE Transactions on Intelligent Vehicles*, 9(3):4312–4334, 2024.

[36] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[37] M. Abe. *Automotive Vehicle Dynamics*. Tokyo Denki University Press, 2008.

[38] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. pages 1094–1099, 06 2015.

[39] Sanyapong Petchrompo, David W. Coit, Alexandra Brintrup, Anupong Wannakrairot, and Ajith Kumar Parlikad. A review of pareto pruning methods for multi-objective optimization. *Computers & Industrial Engineering*, 167:108022, 2022.

[40] Xi Lin, Hui-Ling Zhen, Zhenhua Li, Qing-Fu Zhang, and Sam Kwong. Pareto multi-task learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[41] Zhengliang Liu and Matthias Ehrgott. Primal and dual algorithms for optimization over the efficient set. *Optimization*, 67(10):1661–1686, 2018.

[42] Debabrata Mahapatra and Vaibhav Rajan. Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6597–6607. PMLR, 13–18 Jul 2020.

[43] Mizuho Aoki, Kohei Honda, Hiroyuki Okuda, Tatsuya Suzuki, Akira Ito, and Daisuke Nagasaka. Obstacle avoidance control based on nonlinear mpc for all wheel driven in-wheel ev in steering failure. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2863–2868, 2022.

[44] Yu-lei Liao, Ming-jun Zhang, and Lei Wan. Serret-frenet frame based on path following control for underactuated unmanned surface vehicles with dynamic uncertainties. *Journal of Central South University*, 22:214–223, 01 2015.

[45] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[46] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[47] Yu-Hong Dai. A perfect example for the bfgs method. *Mathematical Programming*, 138:501–530, 2013.

[48] Toshiyuki Ohtsuka. A continuation/gmres method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.

[49] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011.

[50] Lorenzo Stella, Andreas Themelis, Pantelis Sopasakis, and Panagiotis Patrinos. A simple and efficient algorithm for nonlinear model predictive control. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1939–1944, 2017.

[51] Elias Small, Pantelis Sopasakis, Emil Fresk, Panagiotis Patrinos, and George Nikolakopoulos. Aerial navigation in obstructed environments with embedded nonlinear model predictive control. In *2019 18th European Control Conference (ECC)*, pages 3556–3563, 2019.

[52] Ali Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[53] Nikolas Kantas, JM Maciejowski, and A Lecchini-Visintini. Sequential monte carlo for model predictive control. In *Nonlinear model predictive control: Towards new challenging applications*, pages 263–273. Springer, 2009.

[54] Marin Kobilarov. Cross-entropy motion planning. *International Journal of Robotic Research (IJRR)*, 31:855–871, 05 2012.

[55] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[56] Sumioka Tadashi, Hosoya Tomoyuki, and Mori Yoshihiro. Vehicle motion control under limit state and fastest speed control by using nonlinear model predictive control. *Transactions of the Society of Instrument and Control Engineers*, 53(7):385–397, 2017.

[57] M. Sampei. A control strategy for a class of nonholonomic systems - time-state control form and its application. In *Proceedings of 1994 33rd IEEE Conference on Decision and Control*, volume 2, pages 1120–1121 vol.2, 1994.

[58] M. Sampei, H. Kiyota, H. Koga, and M. Suzuki. Necessary and sufficient conditions for transformation of nonholonomic system into time-state control form. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 4, pages 4745–4746 vol.4, 1996.

[59] Shibata Koji, Shibata Naoki, Nonaka Kenichiro, and Sekiguchi Kazuma. Model predictive obstacle avoidance control for vehicles with automatic velocity suppression using artificial potential field. *IFAC-PapersOnLine*, 51(20):313–318, 2018. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[60] Mizuho Aoki, Kohei Honda, Hiroyuki Okuda, and Tatsuya Suzuki. Comparative study of prediction models for model predictive path- tracking control in wide driving speed range. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 1261–1267, 2021.

[61] Jean Pierre Allamaa, Petr Listov, Herman Van Der Auweraer, Colin Jones, and Tong Duy Son. Real-time nonlinear mpc strategy with full vehicle validation for autonomous driving. In *2022 American Control Conference (ACC)*, pages 1982–1987, 2022.

[62] Philip Polack, Florent Altché, Brigitte d'Andréa Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? In *2017 IEEE Intelligent Vehicles Symp. (IV)*, pages 812–818. IEEE, 2017.

[63] Nathan D. Ratliff, Jan Issac, Daniel Kappler, Stan Birchfield, and Dieter Fox. Riemannian motion policies, 2018.

[64] Mizuho Aoki, Kohei Honda, Hiroyuki Okuda, and Tatsuya Suzuki. Switching sampling space of model predictive path-integral controller to balance efficiency and safety in 4wids vehicle navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3196–3203, 2024.

[65] Kiattisin Kanjanawanishkul. Omnidirectional wheeled mobile robots: Wheel types and practical applications. *International Journal of Advanced Mechatronic Systems*, 6:289, 02 2015.

[66] Xiaolong Zhang, Yuanlong Xie, Liquan Jiang, Gen Li, Jie Meng, and Yu Huang. Trajectory tracking of a 4wis4wid robot using adaptive receding horizon control based

on neurodynamics optimization. In *2019 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 565–570, 2019.

[67] Eko Henfri Binugroho, Andri Setiawan, Yudha Sadewa, Prishandy Hamami Amrulloh, Kafin Paramasastra, and Rahardita Widyatra Sudibyo. Position and orientation control of three wheels swerve drive mobile robot platform. In *2021 International Electronics Symposium (IES)*, pages 669–674, 2021.

[68] Lijun Xu, Yankun Yang, Qinhan Chen, Fengcheng Fu, Bihang Yang, and Lijian Yao. Path tracking of a 4wis-4wid agricultural machinery based on variable look-ahead distance. *Applied Sciences*, 12(17), 2022.

[69] Tzuu-Hseng S. Li, Ming-Han Lee, Chia-Wei Lin, Guan-Hong Liou, and Wei-Chung Chen. Design of autonomous and manual driving system for 4wis4wid vehicle. *IEEE Access*, 4:2256–2271, 2016.

[70] Widagdo Purbowaskito and Chung-Hao Hsu. Kinematics model based motion control of a 4wis mobile robot with lyapunov stability. In *2020 12th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 68–73, 2020.

[71] Ming-Han Lee and Tzuu-Hseng S. Li. Kinematics, dynamics and control design of 4wis4wid mobile robots. *The Journal of Engineering*, 2015(1):6–16, 2015.

[72] Igor Mezić and Andrzej Banaszuk. Comparison of systems with complex behavior. *Physica D: Nonlinear Phenomena*, 197(1):101–133, 2004.

[73] PETER J. SCHMID. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010.

[74] Grady Williams, Paul Drews, Brian Goldfain, James M Rehg, and Evangelos A Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[75] Rohit Chandra. *Parallel programming in OpenMP*. Morgan kaufmann, 2001.

[76] Mizuho Aoki, Kohei Honda, Hiroyuki Okuda, and Tatsuya Suzuki. Priority-based decomposition of multi-objective control problem for holonomic vehicle motion planning in arbitrary obstacle avoidance. *Transactions of the Society of Instrument and Control Engineers*, 61(9), 2025.

[77] Mizuho Aoki, Kohei Honda, Hiroyuki Okuda, and Tatsuya Suzuki. Switching sampling space of model predictive path-integral controller to balance efficiency and safety in 4wids vehicle navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[78] Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1):1502242, 2018.

[79] Anders Forsgren, Philip E Gill, and Margaret H Wright. Interior methods for nonlinear optimization. *SIAM review*, 44(4):525–597, 2002.

[80] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-theoretic model predictive control: Theory and applications to autonomous driving. *IEEE Transactions on Robotics (T-RO)*, 34(6):1603–1622, 2018.

[81] Lucas Streichenberg, Elia Trevisan, Jen Jen Chung, Roland Siegwart, and Javier Alonso-Mora. Multi-agent path integral control for interaction-aware motion planning in urban canals. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1379–1385, 2023.

[82] Ermano Arruda, Michael J Mathew, Marek Kopicki, Michael Mistry, Morteza Azad, and Jeremy L Wyatt. Uncertainty averse pushing with model predictive path integral control. In *International Conference on Humanoid Robotics (Humanoids)*, pages 497–502. IEEE, 2017.

[83] Nicklas Hansen, Xiaolong Wang, and Hao Su. Temporal difference learning for model predictive control. In *International Conference on Machine Learning (ICML)*, 2022.

[84] Qian Xu, Zhanqi Xu, and Tao Ma. A survey of multiobjective evolutionary algorithms based on decomposition: Variants, challenges and future directions. *IEEE Access*, 8:41588–41614, 2020.

[85] Hideo HANAFUSA, Tsuneo YOSHIKAWA, and Yoshihiko NAKAMURA. Redundancy analysis of articulated robot arms and its utilization for tasks with priority. *Transactions of the Society of Instrument and Control Engineers*, 19(5):421–426, 1983.

[86] Hakan Girgin and Sylvain Calinon. Nullspace structure in model predictive control. *arXiv preprint arXiv:1905.09679*, 2019.

[87] Escande Adrien, Mansard Nicolas, and Wieber Pierre-Brice. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research (IJRR)*, 33(7):1006–1028, 2014.

[88] C. Dario Bellicoso, Christian Gehring, Jemin Hwangbo, Péter Fankhauser, and Marco Hutter. Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 558–564, 2016.

[89] Defeng He, Haiping Li, and Haiping Du. Lexicographic multi-objective mpc for constrained nonlinear systems with changing objective prioritization. *Automatica*, 125:109433, 2021.

[90] Xavier Prats, Vicenç Puig, Joseba Quevedo, and Fatiha Nejjari. Lexicographic optimisation for optimal departure aircraft trajectories. *Aerospace Science and Technology*, 14(1):26–37, 2010.

[91] Marrisa Kimaporn and Wuttinan Nunkaew. Solving clustering and allocation problems of human-robot collaboration in smart industry 5.0 applications using fis-gra integration-based multi-objective programming model. In *Proceedings of the 2024 13th International Conference on Informatics, Environment, Energy and Applications*, IEEA '24, page 29–39, New York, NY, USA, 2024. Association for Computing Machinery.

[92] Tixiao Shan and Brendan Englot. A lexicographic search method for multi-objective motion planning. *arXiv preprint arXiv:1909.02184*, 2019.

[93] Tixiao Shan, Wei Wang, Brendan Englot, Carlo Ratti, and Daniela Rus. A receding horizon multi-objective planner for autonomous surface vehicles in urban waterways. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 4085–4092, 2020.

[94] Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[95] Masashi Okada and Tadahiro Taniguchi. Variational inference mpc for bayesian model-based reinforcement learning. In *Proceedings of the Conference on Robot Learning*, volume 100, pages 258–272. PMLR, 30 Oct–01 Nov 2020.

[96] Haoru Xue, Chaoyi Pan, Zeji Yi, Guannan Qu, and Guanya Shi. Full-order sampling-based mpc for torque-level locomotion control via diffusion-style annealing, 2024.

[97] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[98] Franck Djeumou, Thomas Jonathan Lew, Nan Ding, Michael Thompson, Makoto Suminaka, Marcus Greiff, and John Subosits. One model to drift them all: Physics-informed conditional diffusion model for driving at the limits. In *8th Annual Conference on Robot Learning*, 2024.

[99] Trey P. Weber and J. Christian Gerdes. Modeling and control for dynamic drifting trajectories. *IEEE Transactions on Intelligent Vehicles*, 9(2):3731–3741, 2024.

[100] Dasol Jeong, Seungtaek Kim, Jonghyup Lee, Seibum B. Choi, Mintae Kim, and Hojong Lee. Estimation of tire load and vehicle parameters using intelligent tires combined with vehicle dynamics. *IEEE Transactions on Instrumentation and Measurement*, 70:1–12, 2021.

[101] Taku Okawara, Kenji Koide, Shuji Oishi, Masashi Yokozuka, Atsuhiko Banno, Kentaro Uno, and Kazuya Yoshida. Tightly-coupled lidar-imu-wheel odometry with online calibration of a kinematic model for skid-steering robots. *IEEE Access*, 12:134728–134738, 2024.

[102] Martim Brandão, Yukitoshi Minami Shiguematsu, Kenji Hashimoto, and Atsuo Takanishi. Material recognition cnns and hierarchical planning for biped robot locomotion on slippery terrain. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 81–88, 2016.

[103] Mustafa Mukadam, Jing Dong, Frank Dellaert, and Byron Boots. Steap: simultaneous trajectory estimation and planning. *Autonomous Robots*, 43(2):415–434, 2019.