Realization of Safe Autonomous Driving using Randomized Model Predictive Control

Arun Muraleedharan

Department of Mechanical Systems Engineering
Graduate School of Engineering, Nagoya University

This dissertation is submitted for the degree of
Doctor of Philosophy

Nagoya University September 2023

I would like to dedicate this thesis to my loving family . . .

Table of contents

List of figures

List of tables

List of Algorithms

Nomenclature

Roman Symbols

β	Slip angle
χ_{ref}	Reference path
δ	Steering angle
ρ	Curvature of reference path
C_f	Cornering stiffness - Front tire
C_r	Cornering stiffness - Rear tire
$F_{c/o}$	Cut-off frequency
I_z	Yaw moment of inertia
l_f	Distance from gravity to front axis
l_r	Distance from gravity to rear axis
m	Mass of the vehicle
N	Prediction horizon length
N_s	Number of samples
V	Forward velocity of vehicle

List of abbreviations

ACC
adaptive cruise control 72

AD
autonomous driving 1

AI
artificial intelligence 47

ALUs
arithmetic logic units 34

BHMIP
bayesian human motion intentionality predictor 46

C/GMRES
continuation/ generalized minimal residual method 8

CPU
central processing unit 20

CUDA
compute unified device architecture 35

DARPA
defense advanced research projects agency 3

DBM
dynamic bayesian network 46

DMC
dynamic matrix control 13

GHMMs
growing hidden Markov models 46

GMRES
generalized minimal residual method 79

GPU
graphics processing unit 10

IDCT
inverse discrete cosine transform 24

IMM-KF
interacting multiple-model kalman filters 47

KKT
Karush–Kuhn–Tucker 78

LSTM
long short term memory 46

MIMO
multiple input multi output 8

MP&C
motion planning and control 6

MPC
model predictive control 7

MSE
mean standard error 53

NHTSA
national highway traffic safety administration 1

NMPC	
nonlinear MPC	15
PID	
proportional integral derivative	7
PrARX	
probability weighted ARX	45
PWM	
pulse width modulation	36
RC	
radio-controlled	11
RMPC	
randomized MPC	9
RRT	
rapidly exploring random tree	19
SISO	
single input single output	7
SMPLC	
simultaneous motion planning and control	19
WHO	
world health organization	1

Chapter 1

Introduction

1.1 Background

Autonomous driving (AD) has been advancing significantly over the last few decades. It is impossible not to wonder why it has been a major research interest all around the globe. Moving people and things from point to point is an essential part of daily life, and the major share of such needs are met by on-road transportation. While on-road transportation is essential, according to the world health organization (WHO), approximately 1.3 million people die each year as a result of road accidents [7]. Adding injuries and economical loss to this, the number gets even bigger. A recent technical report published by the national highway traffic safety administration (NHTSA) reveals that 94% of traffic accidents result from human errors [8]. AD systems are currently in development, aiming to mitigate these accidents. Additionally, as illustrated in Figure 1.1, the implementation of autonomous driving technology is anticipated to bring about numerous other benefits. They include environmental gains, higher productivity and greater independence [5]. In the case of widespread AD deployment, the annual social benefits due to it are expected to cross $800 billion by 2050. This includes gains from congestion reduction, accident reduction, lower energy consumption and increased productivity by utilising time otherwise spent in driving [9].

1.2 History of autonomous driving

The first mention of autonomous driving machines can be dated back to the 1500s. The original concept for an autonomous vehicle was created decades before the first automobile. Leonardo da Vinci invented a cart that could move without being pulled or pushed in the

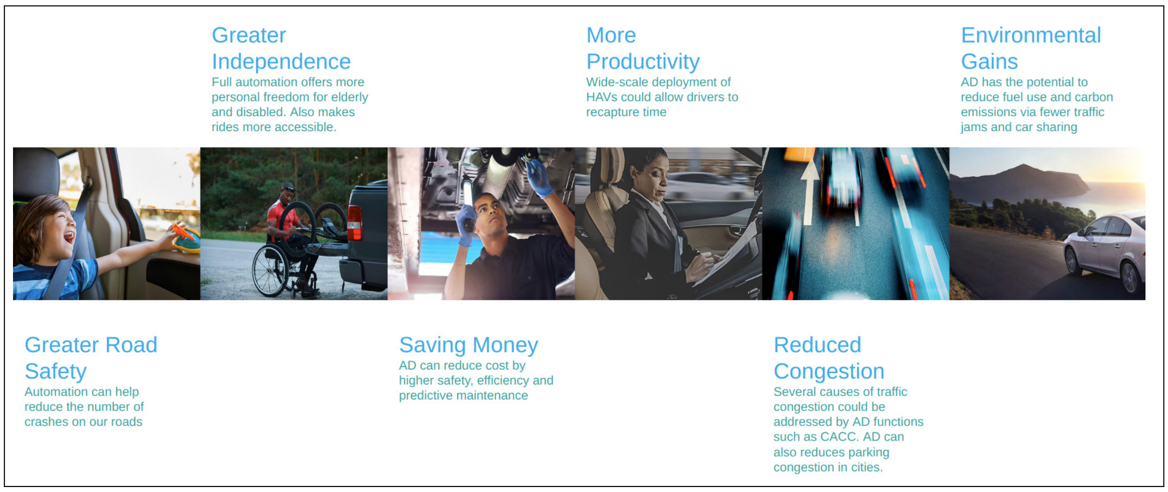

Greater Independence
Full automation offers more personal freedom for elderly and disabled. Also makes rides more accessible.
More Productivity
Wide-scale deployment of HAVs could allow drivers to recapture time
Environmental Gains
AD has the potential to reduce fuel use and carbon emissions via fewer traffic jams and car sharing
Greater Road Safety
Automation can help reduce the number of crashes on our roads
Saving Money
AD can reduce cost by higher safety, efficiency and predictive maintenance
Reduced Congestion
Several causes of traffic congestion could be addressed by AD functions such as CACC. AD can also reduces parking congestion in cities.

Fig. 1.1 The advantages of autonomous driving (adapted from [1])

1500s [10]. Power was given by highly compressed springs, and the steering was pre-set so that the cart could follow a specified path. This item is also occasionally referred to as the original robot.

In 1925, inventor Francis Houdina achieved a remarkable feat by driving a radio-controlled car through the streets of Manhattan, notable for its absence of a human driver behind the wheel. This car's capabilities included the ability to start the engine, shift gears, and even sound the horn [10]. This vehicle gave a glimpse into the autonomous future, but it was swiftly shut down after the driver lost control twice and collided with another automobile. The industry didn't give up on remote-controlled cars despite this early setback.

General Motors developed the first self-driving car model at the 1939 World's Fair [10]. This pioneering vehicle was an electric car that operated by running on magnetized metal spikes embedded in the roadway and was controlled through radio-controlled electromagnetic fields. In 1958, this model became a reality [10]. The vehicle was equipped with sensors that could identify the current moving through a wire buried in the pavement. The direction of the steering wheel could be altered by adjusting the current, allowing it to move either to the left or to the right.

During the height of the space race in 1961, scientists began contemplating the challenge of landing vehicles on the moon's surface. As a result, James Adams developed the Stanford Cart, a vehicle equipped with cameras and designed to autonomously identify and track a line on the ground [11]. Cameras, a crucial component of modern autonomous vehicles, were used for the first time in these vehicles.

Building upon this idea, the Tsukuba Mechanical Engineering Lab in Japan advanced the concept in 1977 by introducing a camera system that transmitted information to a computer.

This computer processed images of the road, paving the way for the testing of the first autonomous passenger car capable of reaching speeds of up to 20 mph [12].

By incorporating neural networks into image processing and steering controls, Carnegie Mellon University started developing self-driving cars in 1990 [13]. In 1995, researchers from Carnegie Mellon accomplished a remarkable feat by driving their self-driving vehicle, named NavLab 5, a distance of 2,797 miles from Pittsburgh to San Diego. Although the vehicle was otherwise autonomous, it was responsible for managing its own speed and braking [13].

The autonomous vehicle sector was booming by the early 2000s. Several competitions were sponsored by defense advanced research projects agency (DARPA), the research branch of the U.S. Department of Defense, to advance autonomous vehicles. They staged a competition in 2004 wherein vehicles had to cross 150 miles of arid terrain on their own [10]. No vehicles finished the course. In 2007, a 60-mile urban setting was replicated for the challenge; this time, four cars completed the journey [10].

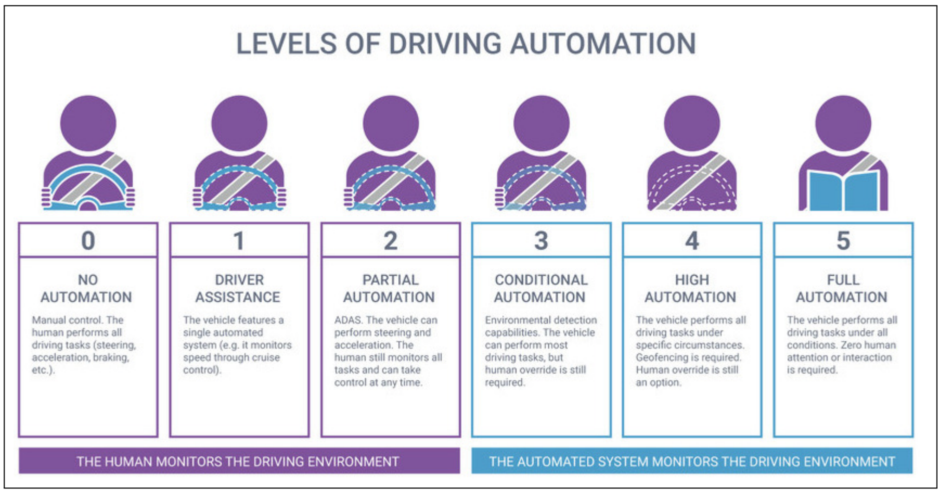
Major automakers like Ford, Mercedes-Benz, and BMW, as well as ride-hailing services like Uber, started competing with self-driving technology by the middle of the decade [11]. True autonomy turned out to be trickier to attain than first anticipated, and many of these businesses eventually shut down. Notably, Uber declared in 2020 that they were abandoning their efforts to create self-driving cars due to concerns about safety, legal action, and financial loss [14].

With their Full Self-Driving package, which enables autonomous hands-free operation for highway and freeway driving, Tesla is the firm that will be the closest to releasing autonomous vehicles on the market as of 2021 [11]. The vehicles aren't autonomous, though, by any standard. In reality, Tesla was requested by the German authorities to stop using this phrase.

There are currently no commercially accessible completely autonomous vehicles, not even among these ground-breaking inventions. However, hundreds of autonomous cars are already in use in significant areas like mining. In under 7 years of operation as of May 2021, Caterpillar's autonomous trucks have securely transported more than 3 billion tonnes of material [11]. The success of autonomous vehicles in the mining sector offers some encouragement for the challenges that lie ahead for these vehicles.

1.3 State of the art and general challenges in Autonomous Driving

The development of deep learning, advances in computer vision due to it, and the availability of novel sensor modalities, including lidar, all sparked interest in AD research and its commercial use. The emergence of AD with various degrees of automation was also prompted by a rise in the public interest and market potential [5]. Such degrees of automation are standardised by the Society of Automotive Engineers (SAE) as depicted in fig 1.2.


LEVELS OF DRIVING AUTOMATION
0
NO AUTOMATION
1
DRIVER ASSISTANCE
2
PARTIAL AUTOMATION
3
CONDITIONAL AUTOMATION
4
HIGH AUTOMATION
5
FULL AUTOMATION
THE HUMAN MONITORS THE DRIVING ENVIRONMENT
THE AUTOMATED SYSTEM MONITORS THE DRIVING ENVIRONMENT

Manual control. The human performs all driving tasks (steering, acceleration, braking, etc.).

The vehicle features a single automated system (e.g. it monitors speed through cruise control).

ADAS. The vehicle can perform steering and acceleration. The human still monitors all tasks and can take control at any time.

Environmental detection capabilities. The vehicle can perform most driving tasks, but human override is still required.

The vehicle performs all driving tasks under specific circumstances. Geofencing is required. Human override is still an option.

The vehicle performs all driving tasks under all conditions. Zero human attention or interaction is required.

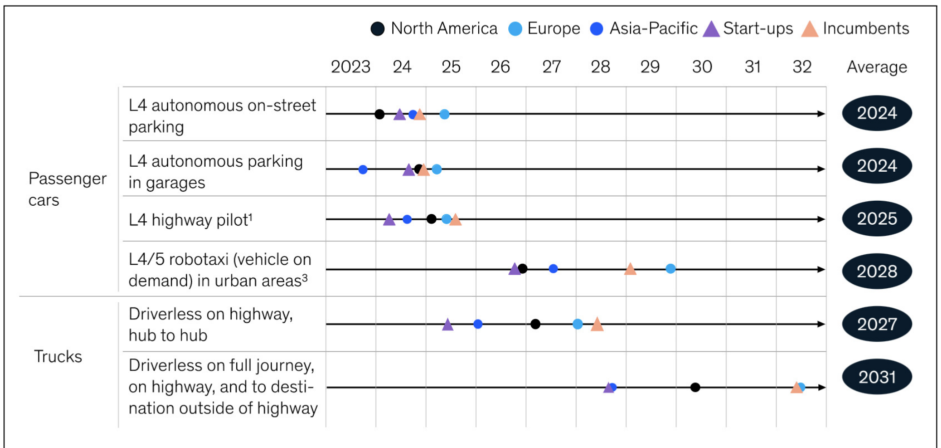
Fig. 1.2 The 6 Levels of Vehicle Autonomy (reproduction from [2])

The five degrees of driving automation established by SAE start at Level 0. This category denotes a complete lack of automation. Level one systems are the most basic and include stability control, anti-lock braking, and adaptive cruise control. Level two is a level of partial automation that integrates advanced aid technologies like emergency braking and accident avoidance. Level two automation became a practical technology as a result of industry experience and the body of knowledge in the field of vehicle control. Being able to drive hands-free on closed highways is becoming an expected level 2 (fig. 1.2) feature in modern automobiles. Above this point, the true challenge begins [5].

Level three automation represents conditional automation, where under normal conditions, the driver can divert their attention away from driving tasks but must remain prepared to promptly intervene in case of an emergency. Moreover, level three autonomous driving systems are limited to specific operational design domains (ODDs), primarily encompassing highway driving scenarios. In these constrained highway conditions, Audi asserts its position as the first automaker to achieve level three automation in a production vehicle.

However, transitioning from automated mode to manual mode poses a significant challenge. Recent research on this subject has identified an elevated risk of collisions with nearby vehicles during the takeover process. An unresolved issue pertains to the heightened accident risk associated with this transition [5].

Even though highway AD up to level 3 is on the verge of being a reality, there are multiple domains in which autonomous driving is still facing challenges to guarantee safety. At level four and level five, no degree of human attention is required. Level four, however, can only function in specific ODDs with specialized infrastructure or comprehensive maps. If the car leaves specified regions, it must automatically park itself to end the trip. The level five completely automated system can function in any type of road network and any kind of weather. As of now, achieving level four or level five driving automation in production vehicles remains unattainable. Moreover, a recent announcement from the Toyota Research Institute underscores that no company within the industry has made significant progress toward realizing level five automation [5].



[1]Driver can use the time on highways for work or leisure activities using in-car or own solutions but needs to take over at highway exits.
[2]Driver can use the time on highways in urban environments for work or leisure activities using in-car or own solutions but may require some driver intervention.
[3]Robotaxis drive everywhere fully automated with no driver and accept and conduct transportation requests (goods, passengers). Passenger can use the travel time for work or leisure activities.
Question: In your estimation, what is the rollout (ie, commercial availability of vehicles/service) timeline for autonomous driving across use cases in your region?
Source: 75 respondents (North America, n = 31; Europe, n = 33; Asia–Pacific, n = 11)

Fig. 1.3 What's next for autonomous vehicles - A McKinsey survey (reproduction from [3])

Driving automation at levels four and up on urban road networks is an unsolved and difficult issue. The environmental factors, which include everything from the weather to the nearby population's behavior, are exceedingly unpredictable. This is reflected in fig.1.3, according to this McKinsey survey, level 4 AD in highways is expected by 2025 and urban AD is expected to take another 3 years to realize. A third of the causes behind such a delay is due to technical challenges that hinder the safety guarantee [3]. Additionally, system flaws result in accidents. For example, one of the AD cars in the Hyundai competition crashed due to rain [15], Google's AD car struck a bus while changing lanes because it underestimated the velocity of a bus [16], also Tesla's Autopilot collided with a white truck after failing to recognize it, killing the driver [17].

Deaths brought on by underdeveloped technology reduce public support for AD. A recent survey [18] found that the majority of the consumers are concerned about the safety of the technology and desire substantial control over its creation and application. On the other hand, AD vehicles that are overly cautious can come across poorly. Another set of difficulties arises from ethical quandaries. How should the system respond in the event of an accident, which is unavoidable? Regarding this subject, experimental ethics have been suggested [5].

Another challenge is the certification of risk and reliability. AD systems must be designed with high redundancy to reduce the likelihood of a catastrophic failure, just like in aircraft. Although there are interesting projects in this area, such as DeepTest [19], neither the rule-makers nor the industry have yet developed the design-simulation-test-redesign certification system.

The complexity of an already challenging problem is increased by several optimization objectives like time to destination, fuel efficiency, comfort, and ride-sharing optimization. Consequently, the safe execution of all dynamic driving responsibilities within strict parameters remains a formidable challenge when operating beyond a well-defined geofenced region [5].

1.4 Challenges in autonomous driving from motion planning and control perspective

The major blocks in any autonomous driving stack can be roughly represented as shown in fig. 1.4. Everything starts with a set of sensors including cameras, LiDARs, GPS etc. The sensors observe the surroundings and find the car's position using localization. Sensors also detect and predict the positions and trajectories of other agents around the ego car. With its own position and other agent details, a path planner creates a global path for the AD to achieve its motion target. The next step is the motion planning and control (MP&C). The task of the MP&C is to create and achieve a motion plan which follows the target as accurately as possible. Sometimes the MP&C can also modify the global path locally to achieve targets such as obstacle avoidance. Our research focuses on the MP&C side of autonomous driving. The challenges faced by AD in this domain are discussed in this section. It is also interesting to mention that there have been some interesting works on end-to-end autonomous driving using deep learning methods. Such an approach often connects the sensing section to actuators directly with an end-to-end deep learning model. Fig 1.5 shows the key differences between these two approaches.

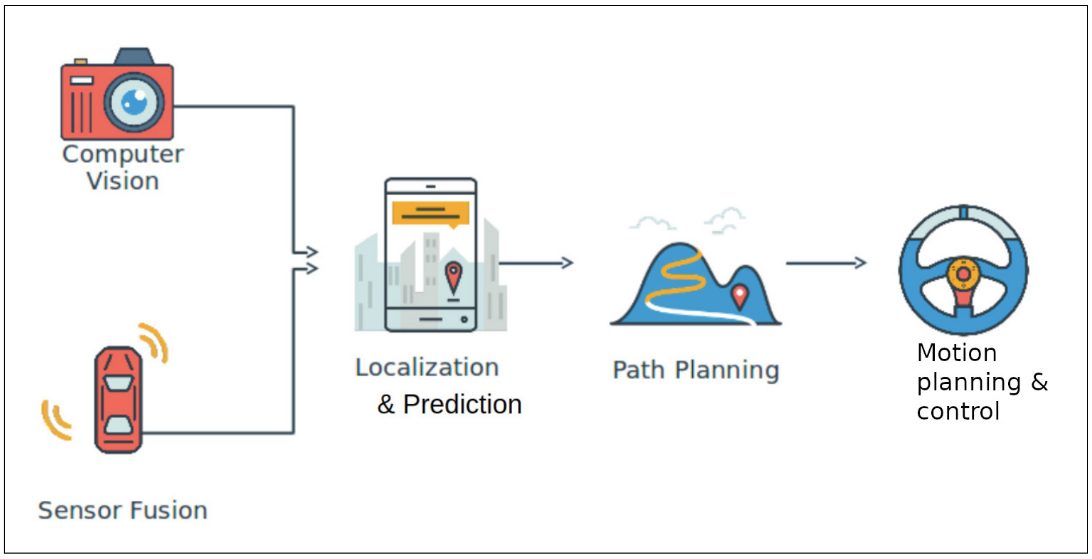


Fig. 1.4 How Control Works for Self-Driving Cars (reproduction from [4])

As discussed in section 1.3, even though highway AD up to level 3 is on the verge of being a reality, there are multiple domains in which autonomous driving is still facing challenges to guarantee safety. Most of these challenges can be characterized by the presence of an external agent that disturbs such stable autonomous driving. In the case of highway driving at high speeds, the disturbance could be sudden cut-ins or obstacles that partly cover the driving area. Once you leave the highway, any task that includes interaction with decision-making agents such as pedestrians, cyclists, or other cars are currently considered as challenging. All these scenarios call for fast controllers that can deal with such disturbances without compromising safety. Since AD promises and needs to deliver a higher level of safety than human driving, guaranteeing safety under such scenarios is of utmost importance.

Once any of the above-mentioned tasks are formulated into a control problem, it can be seen that they are nonlinear control problems. The vehicle models that can accurately represent high-speed driving are nonlinear. So are the models that can represent decision-making agents such as pedestrians that respond to the car's actions. The controllers should also be able to accept certain risk levels while guaranteeing safety, just like us human drivers does. The current control implementations that deal with such scenarios often simplify these problems to linear formulations.

Table 1.1 compares various controller types that are presently used to control self-driving cars. The classical control contains methods such as proportional integral derivative (PID) controllers. These are easy to implement linear controllers that consume very low computational resources. They are capable of handling single input single output (SISO) and are not capable of handling constraints. Such limitations make it necessary to approximate all the nonlinear models and constraints present in the original AD control problem into linear models. Such approximations make a classical control solution inaccurate for AD control. Such approximations, also being far from accurate, calls for higher safety margins. This leads to slower and conservative control behaviour. An accurate controller that can process such non-linearity and stochastic constraints without approximations could provide similar safety margins at higher speeds. This is where a nonlinear model predictive control (MPC) that can process stochastic models shines as the best choice of control.

MPC belongs to the second category in the table 1.1. Modern control solutions such as MPC can handle multiple input multi output (MIMO) systems. They can also contain nonlinear models and constraints in its formulation. Such controllers can also be used for both discrete and continuous control formulations. Such benefits make them much more accurate than classic control solutions. This comes at the cost of challenging computational problems that are at times difficult to solve in a timely manner for fast real-time systems like AD.

There is a third category of control that is a deep learning based. These methods have shown great results under certain limited driving conditions. Since they are learning-based, the training data has a great influence on their performance. Also, the resulting models are black-box models. This makes it often difficult to predict the reasons behind the actions of the system. AD being safety-critical, and having the threat of injury, being unable to analyze causes of collisions or unexpected actions makes it extremely difficult to use such black box models in production. In this study, the nonlinear MPC formulations that are a part of modern control methods are chosen as our control method of choice.

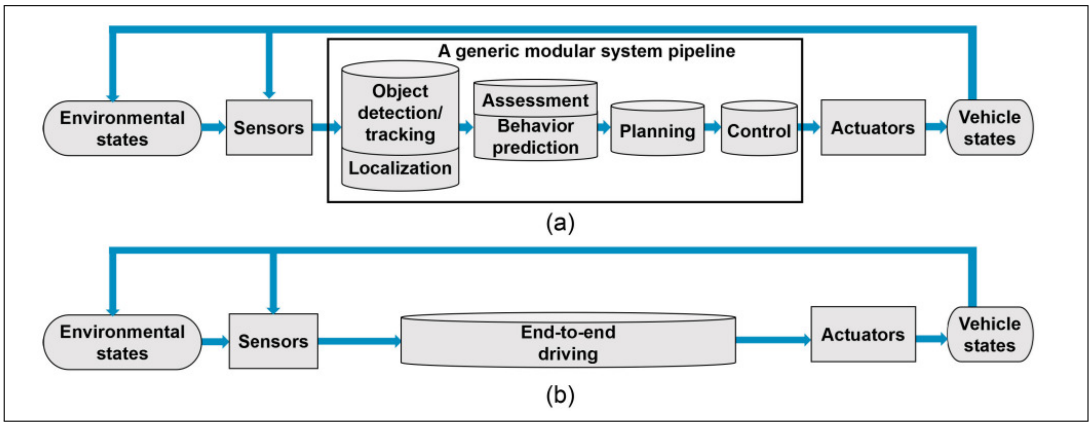


Fig. 1.5 Flow diagrams of generic and end-to-end self driving (reproduction from [5])

Table 1.1 Comparison between various control methods

Classical control	**Modern control**	**Deep learning**
SISO	MIMO	MIMO
Linear	Linear and non linear	Learning based
Continuous	Continuous and discrete	Black box model
Unconstrained	Constrained	-
Fast computation	Computationally challenging	-
Less accurate	Accurate	Training data dependant

Most of the literature on nonlinear MPCs uses a continuation-based method such as interior point or continuation/ generalized minimal residual method (C/GMRES) to quickly solve the optimal control problem. Even though they process the nonlinear components pretty well, they often approximate the stochastic constraints with other continuous functions. Such an approximation makes them behave conservatively in the presence of stochastic elements such as pedestrians in the driving scenario. Another drawback of continuation-based optimizers is that they are continuation based as the name itself implies. They track the past solutions to quickly converge to the next solution. This fact makes them unreliable at unexpected changes in driving scenarios.

Randomized MPC (RMPC) (2.2) is a type of nonlinear MPC that uses a randomized optimization algorithm. Since RMPC pick the minimum cost sample among randomly sampled candidate solutions, it does not bias the solution towards past solutions. With the right samples, it is possible to arrive at a semi-global solution that could be away from the past minimum point. A method to represent the stochastic constraints without modifications based on the number of samples that violate the constraints is also proposed. Hence, RMPC is expected to provide a better performance in comparison to continuation-based nonlinear MPCs. RMPC also has certain drawbacks that are identified and solved in this work. They are the noisiness of the input signal, slow computational time at high sample sizes and a lack of parameter selection methodology. Solutions to all three of these challenges are proposed in chapter 3. Hence, the RMPC proposed in this work overcomes the major drawbacks of existing nonlinear controllers and realizes safe AD. This work also addresses the current implementation challenges faced by RMPC.

1.5 Goal of research

Various challenges faced by AD have been discussed, particularly in the control perspective in section 1.4. Reflecting on our motivation (1.1), our research goal is set as follows. The safety levels of autonomous driving need to be raised to guarantee safe operation in various environments. RMPC with its various advantages is a strong candidate as a fast and flexible nonlinear controller, but largely unexplored towards this goal. This study has shown that the advantages of RMPC are particularly suited to tackle challenging tasks in AD control. That too at a higher safety level in comparison to the current state of the art control solutions. This work starts by addressing the issues that are thought to be preventing RMPC from being widely used. As seen in fig 1.6, this includes noisy input signals, high computational burden, lack of a parameter selection methodology, method of expressing stochastic constraints and the capability to supplement other optimization methods.

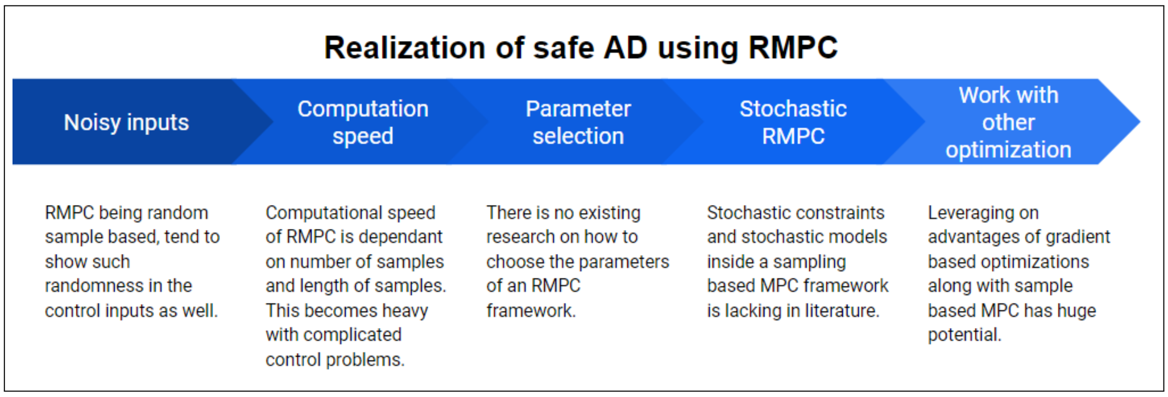

Realization of safe AD using RMPC
Noisy inputs
Computation speed
Parameter selection
Stochastic RMPC
Work with other optimization
RMPC being random sample based, tend to show such randomness in the control inputs as well.
Computational speed of RMPC is dependant on number of samples and length of samples. This becomes heavy with complicated control problems.
There is no existing research on how to choose the parameters of an RMPC framework.
Stochastic constraints and stochastic models inside a sampling based MPC framework is lacking in literature.
Leveraging on advantages of gradient based optimizations along with sample based MPC has huge potential.

Fig. 1.6 Challenges in realizing AD using RMPC

Following this, this work demonstrates that RMPC can realize safe AD during various challenging driving tasks. Examples of high-speed obstacle avoidance, considerate interaction with a crossing pedestrian and a sudden cut-in scenario during highway driving as some examples where RMPC has shown to be safer and better in performance. In the task of enabling reliable high-speed obstacle avoidance, the research goals are threefold. Firstly, sampling from the frequency domain has been implemented that gives smoother random samples in comparison to sampling in the time domain. Second, the symmetry property of the samples is leveraged to implement the algorithm in parallel in multiple graphics processing unit (GPU) cores. This reduces the computational complexity dramatically. The third goal of this task is to propose a systematic parameter selection methodology for RMPC. In the task of considerate interaction with pedestrians, safe autonomous driving has been enabled among dynamic agents in urban AD. A sample-based representation of collision risk constraint is proposed, this enables the tuning of the risk levels the controller admits. This part also proposes a novel stochastic pedestrian model in this section, this model allows us to model the interaction between the car and pedestrian. It also enables us to put a cost on the internal confusion levels of the pedestrian, thereby enabling considerate driving. The last task that is demonstrated is the handling of sudden cut-ins in highway driving. In this task, RMPC is combined with a continuation-based optimizer. Under sudden state changes, when the continuation-based optimisation becomes unstable, RMPC helps to bring the optimizer back to stability faster and provide safe control inputs in its place. Throughout this work, the RMPC framework is built from its basic form. This has been done to make sure that this work act as a reference to the researchers and practitioners who would like to explore RMPC and its possibilities.

1.6 Organisation of the thesis

The rest of this thesis is organized as follows. Chapter 3 identifies and addresses major bottlenecks in using RMPC to address challenging control problems in AD. This includes the input signals being noisy, RMPC being not useful at higher sample counts because of the computational burden and an efficient method of parameter selection for RMPC. The effectiveness of RMPC is demonstrated by high-speed obstacle avoidance driving of a radio-controlled (RC) car. In chapter 4, a method of representing stochastic constraints in an RMPC framework is presented. This is demonstrated by enabling human-like considerate driving in an interactive scenario between a crossing pedestrian and a self-driving car. Even though RMPC can handle most of the nonlinear control problems well, there are certain cases where a common gradient-based optimisation provides a stable and smooth solution. In chapter 5, a combination of such a commonly used nonlinear MPC framework with the RMPC is proposed. In the following chapter 6, a summary with some concluding remarks is presented along with some possible directions for future work.

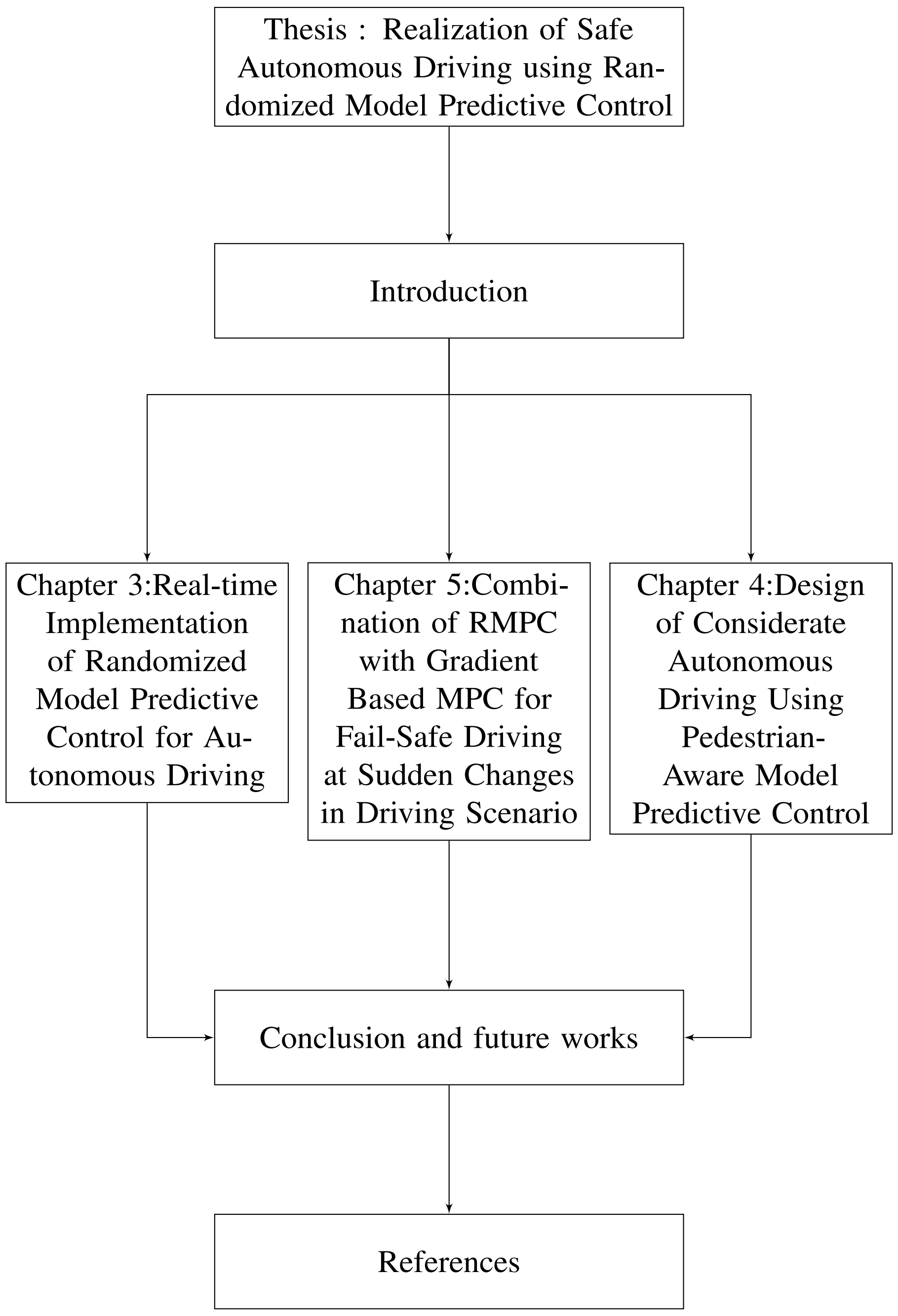

Thesis : Realization of Safe Autonomous Driving using Randomized Model Predictive Control
Introduction
Chapter 3:Real-time Implementation of Randomized Model Predictive Control for Autonomous Driving
Chapter 5:Combination of RMPC with Gradient Based MPC for Fail-Safe Driving at Sudden Changes in Driving Scenario
Chapter 4:Design of Considerate Autonomous Driving Using Pedestrian-Aware Model Predictive Control
Conclusion and future works
References

Fig. 1.7 Flow of the thesis

Chapter 2

Model predictive control (MPC)

Model Predictive Control (MPC), as referenced in [20–22], stands out as a widely employed control strategy for addressing multi-variable constrained control challenges. Despite its initial introduction dating back to [23] in the 1960s, MPC's substantial adoption and expansion in practical applications primarily stemmed from advancements in computational capabilities. This includes the works such as [24, 25] in the late 1970s, which have independently laid a strong foundation for MPC theory. They were able to effectively regulate complex tasks with the new digital controllers, showcasing a huge economic potential. Model predictive heuristic control (MPHC), which was first introduced by [24] in 1978, already had all the characteristics of an MPC, including an explicit process model, impulse response functions (IRFs) that describe the process, a receding horizon, input and output constraints, and an iterative determination of the controls (value of the manipulated variable u). But [24] didn't assert to have the best optimum controls. Instead, until the future controls satisfied the constraints, they were determined repeatedly. Heuristic was used to emphasize the lack of defined control law. The process sector, with its multiple input multiple output (MIMO) systems, noticeable delays, and lengthy processing times, was the target market for the technology [24]. Even the idea of identifying the process model online was investigated, but only for adjustments to the set points. Dynamic matrix control (DMC) was developed by [25] from Shell Oil Company around the same period. To forecast the behavior of a catalytic cracking unit in the future, the DMC employed a piecewise linear model. The controller was able to understand the plant's time delay and dynamic system behavior as a result. Cutler and Ramaker adjusted the model coefficients based on the discrepancy between the previously anticipated output and the current measured output using a receding prediction horizon. They demonstrated that DMC outperforms traditional cascaded PID control and claimed that DMC has been used at Shell Oil since 1974 to solve control issues. The primary distinction between DMC and MPHC is the calculation of optimal control variables. However, the control problem's matrix formulation limits DMC to linear process models. Both of these efforts set the stage for the widespread and quick adoption of MPC in the petrochemical process sector. The sampling periods were many hours even with linear models [26]. To enable the technology to be applied in industry, the first emphasis was on making the controller design simple and developing a thorough theory [24, 25, 27]. MPC's potential was not just dependent on prediction, but also on the capability of using non-linear models, both of which were unsupported by conventional control. The process model formulation, including the impulse response formulation (IRF) [105], piecewise linear step response functions [25], ARMA models [28, 29], and state space formulations [30], was a hot topic at the inception of the MPC theory. One of the main factors contributing to MPC's quick success was its flexibility in the model formulation options.

Because the majority of chemical engineering processes were open-loop stable, the earliest techniques simply ignored model uncertainties and process instabilities [23]. The research community around Manfred Morari focused particularly on MPC robustness and stability beginning in the late 1980s [31–35].

The (linear) estimation problem might be stated as a quadratic programming problem [36] with a finite horizon, i.e. a fixed moving window, which proved advantageous computationally. The "explicit MPC" that was introduced by computation pushing [37] switches the computation to large a priori optimization [38].

With the turn of the 2000 and the increasing capability of computers, research switched towards application. The trend was moving away from problems with many control variables and lengthy computation requirements and toward issues with few control variables and significantly shorter computation time requirements.

Within MPC, "model" refers to the foundational representation of the system's dynamics under control. This representation facilitates forecasting future states of the controlled system. Furthermore, MPC continuously monitors the system's state during each control cycle, granting it the ability to adapt to real-time environmental variations. A graphical representation of a discrete MPC system is shown in fig.2.1. Please note that the green line representing predicted output is discrete in the formulation. The primary advantage of Model Predictive Control (MPC) lies in its ability to optimize the present time period while simultaneously considering future time periods. It achieves this by optimizing over a finite time horizon but implementing only the current time period's control actions before iteratively re-optimizing, in contrast to a linear-quadratic regulator (LQR). MPC also has the ability to predict future events and take control actions in response. PID controllers lack this ability to predict. Even though the models used by MPC can represent both complex and simple dynamic systems, such additional complexity is not necessary to control simple systems adequately. In such cases, generic PID controllers will suffice. Typically, large time delays and higher order dynamics necessitate a controller like MPC.

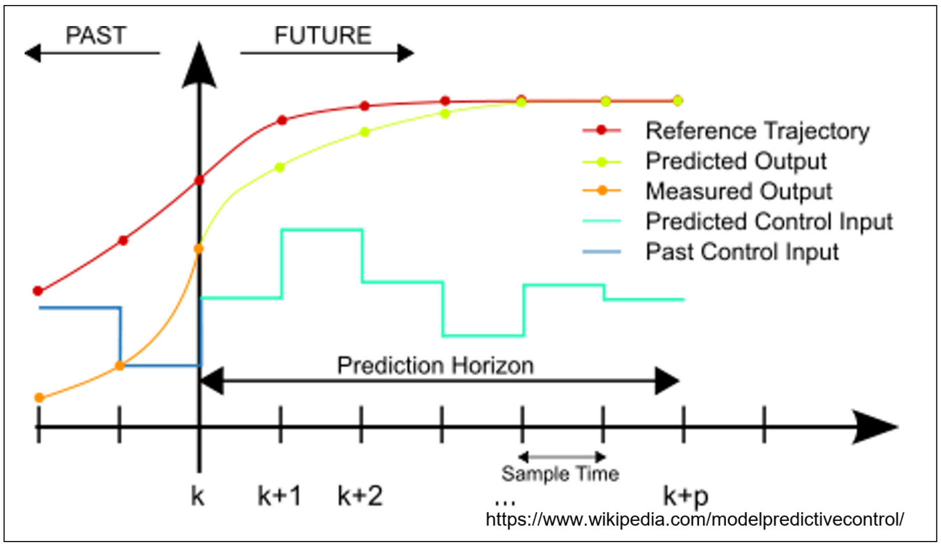


Fig. 2.1 A discrete time MPC scheme

2.1 Linear and nonlinear model predictive control

Although many real-world processes are not inherently linear, they often exhibit approximate linearity within a limited operating range. Consequently, linear Model Predictive Control (MPC) methods are commonly applied in the majority of practical applications. In these cases, the MPC's feedback mechanism plays a crucial role in compensating for prediction errors stemming from the inherent mismatch between the model and the actual process. One advantage of using linear models in MPC is the application of the superposition principle from linear algebra. This principle allows the combined effect of changes in multiple independent variables to be computed, enabling the prediction of the response of dependent variables within the framework of model predictive controllers that rely on linear models. As a result, the control problem is simplified into a series of efficient and dependable direct matrix algebra calculations.

When linear models are unable to accurately capture real-world nonlinearities, several approaches can be employed. One method involves mitigating the nonlinearity by transforming the process variables, both before and after the linear MPC model, as suggested by [39]. Another approach is to utilize nonlinear MPC (NMPC) [39], which directly incorporates a nonlinear model into the control application. This nonlinear model can take the form of a high-fidelity dynamic model based on fundamental mass and energy balances or be derived from empirical data fitting techniques, such as artificial neural networks [40].

NMPC uses nonlinear system models to make predictions. The iterative solution of optimal control problems on a finite prediction horizon is required in NMPC, just as it is in linear MPC. While these problems are convex in linear MPC, they aren't necessarily convex in nonlinear MPC [41]. Both the NMPC stability theory and the numerical solution face difficulties as a result of this.

The numerical solution of NMPC optimum control problems is usually based on direct optimal control methods. These are based on Newton-type optimization algorithms in one of three variants: direct single shooting, direct multiple shooting, or direct collocation. The fact that successive optimal control problems are similar to one another is often leveraged by NMPC algorithms. This permits the usage of an appropriately shifted guess from the previously computed optimal solution to efficiently initialize the Newton-type solution technique. Such initialization saves significant computing time [42]. The similarity of subsequent problems are further exploited by path following algorithms (or "real-time iterations"), which never attempt to iterate any optimization problem to convergence, but instead, take a few iterations towards the solution of the most recent NMPC problem before moving on to the next, which has been properly initialized; see, for example, [42].

With advancements in controller hardware and computational algorithms, such as preconditioning, NMPC is increasingly being applied to applications with high sampling rates, such as in the automotive industry, or even when the states are distributed in space (Distributed parameter systems)[43]. NMPC has recently been utilized for real-time control of autonomous systems in various scenarios, such as optimal trajectory planning and tracking of aircraft [44].

2.2 Randomized MPC

A variation of MPC known as randomised MPC (RMPC) [45–49] makes use of a randomised optimisation technique. The production of samples is the first step in this optimisation technique. In the solution space, N_s sample sequences with length N are created at random. The next stage is to determine the plant's future state, assuming it adopts each of these sample sequences. At this stage, the samples that are impractical are filtered out using the constraints that have been provided. This is followed by the cost calculation of each of the samples using a cost function. The first input in the lowest cost series is then chosen, and it serves as the ego car's input. For each control cycle, the controller repeats these actions. Among the previous works on RMPC, [46] applies RMPC to control a stochastic linear system and [48] uses RMPC to guide an autonomous vehicle. A study on effect of sample size while solving a convex programs with uncertainty using a randomized approach is discussed in [49].

The ability to adjust the number of samples, denoted as N_s, provides us with the flexibility to strike a balance between computational demands and accuracy. Despite its straightforward implementation, RMPC delves into the global solution space and conducts rigorous checks to ensure constraint satisfaction during each iteration. The work in [45] demonstrates the infinite horizon stability of RMPC, while [47] establishes its optimality. However, a challenge in RMPC application persists in the absence of a clear optimality index, resulting in a semi-optimal status. Additionally, the randomness inherent in sampling can impact the smoothness of control inputs, presenting another challenge.

Chapter 3

Real-time Autonomous Driving using Randomized Model Predictive Control

3.1 Introduction

Real-time management of the motion control issue has proven to be difficult. However, MPC effectively addresses it because of the benefits described in section 2. When it comes to motion control issues, autonomous driving is on the challenging end of the scale. It needs precise control inputs to be calculated in real time because, it is very dynamic and safety-critical. In real-time implementations, it has been difficult to compute the steps necessary for the MPC to solve the model-based optimization problem. Utilizing Randomized MPC is one strategy to overcome the computational complexity (RMPC) (section 2.2).

On the other hand, path planning and path following can be considered as two separate tasks in the behavior control of an autonomous vehicle. They are performed alternately. Because the planner must take into account safety limits like collision avoidance and speed restriction, planning has to be repeated. This goes in addition to the primary target of path planning for maintaining a lane, overtaking, etc. In robotics and automotive applications, numerous path planning strategies have been effectively demonstrated, as seen in [50–53]. While [50] presents a survey on MP&C strategies for urban self driving vehicles, [52] and [53] put forward techniques for path planning by utilizing bezier curves and rapidly exploring random tree (RRT)s, respectively. A motion planner for non-holonomic mobile robots that is based on recursive subdivision of a collision-free path produced by a lower-level geometrical planner that disregards the motion constraints is presented by [51]. The next stage is to precisely follow the modified path that the car has prepared. A controller must keep track of the car's current condition and give the right steering and acceleration instructions for accurate path following. There have also been many works on path following [54, 55, 20–22]. Different control techniques, such as proportional-integral-derivative (PID) control, state feedback controllers, MPC, and others, can be used to perform path following control. The MPC-based strategy is described in depth in [20] and [21]. Some advances in path following under sliding effects and reflection of driver characteristics in path following can be found in [54] and [55], respectively. A common strategy in the literature is to separate the path planning and tracking operations. It follows that if vehicle dynamics limitations are taken into account independently for each stage, then this consideration becomes redundant. It is obvious that combining these two into a single problem formulation reduces the computational load by eliminating redundant vehicle dynamics calculations. The term simultaneous motion planning and control (SMPLC) problem is frequently used to describe such conjoined issues [56–58].

During the studies on SMPLC by the writers in [56] and [57], a number of difficulties were discovered with real-time SMPLC implementation. The main difficulty was brought on by how difficult it was to solve an SMPLC problem computationally.Some notable examples that had demonstrated real time steering control with non-linear MPC are [22] and [59]. The work [22] has reported an average computation time of more than 150 ms, restricting the speed of their stable experiment at 7 m/s. While, [59] exhibits a processing time of 60ms on average (using pre-calculated invariant sets), allowing for the avoidance of several obstacles at a speed of 14m/s. Both [22] and [59] demonstrate obstacle avoidance while driving on a straight road. Finding an ideal solution in real-time was challenging due of the stringent time constraints involved with a safety-critical problem like autonomous driving. Even sampling-based optimization approaches like RMPC were unable to achieve the required control frequency with a conventional central processing unit (CPU) based implementation. Another problem was also discovered while simulating RMPC for vehicle control. It was discovered that the sampling process' randomness had a negative effect on how smoothly the car was driven. This will be highlighted further when the RMPC is applied to passenger cars rather than autonomous robots. This necessitates the utilization of a smoothing approach during the sampling phase in order to maintain the smoothness of the control action.

The observer would become aware of the parallelism the RMPC framework by looking at it more closely. The sequence of processes like series creation and calculation of cost could be performed on all sample points simultaneously. The parallel processing capability of a Graphics Processing Unit (GPU) is closely related to this exciting characteristic of RMPC. GPUs are currently attracting interest from every application with a abundant parallelism. GPUs are interesting because they have a much larger computational throughput than CPUs, which are designed for higher latency. This gives GPUs a tremendous potential to speed up applications with a lot of parallel processing [60].

Two categories can be made out of earlier works on sampling-based MPC. Construction of a sampling-based MPC solver falls under the first category [45, 61, 62]. The second is a study that focuses on the implementation of sampling-based MPC [63–67]. In the first category, [61] developed a method that searches the control space by seeding a tree from the previous best sample and expanding it in a manner similar to rapidly exploring random tree (RRT). The paper [45] suggested a sampling-based MPC employing a potential field model of obstacles for one-leg robot navigation. The authors asserted that they could only increase performance at the expense of computational burden, even if implementation is not the main focus of this work. The paper [62] applied a modified version of RMPC to the drive of high-speed RC cars, which enabled faster real-time computation.

In the second category, the cmanagement of a drinking water network is discussed in [63]. A GPU-based implementation was suggested due to the network's size and plenty of configurable parts. Even though the problem size is large, this application adopted longer control intervals than the case of managing a car due to its slow dynamics. In [64], Using an embedded GPU, the authors were able to successfully manage a mobile robot for obstacle avoidance. Here, the robot speed was very low and the robot dynamics were considered as a point mass model. Various methods for nonlinear model predictive control that utilizes GPU are presented in these papers [65] and [66]. The application domain of the optimization algorithms was very different from the control of an autonomous vehicle, and they were not sampling-based in nature. Finally, in the survey paper [67], A summary of different MPC parallel implementations was given. The issues that MPC currently confronts are covered in detail in this work, along with how parallel computing aids in their resolution. This study comes to the conclusion that while most GPU-based solutions are faster at performing calculations, they have a significant overhead in terms of memory transfers and kernel construction. Any improvement over the CPU case is negligible for common use cases if the time of overheads and calculation are combined.

Because no earlier works specifically addressed autonomous driving using RMPC, the authors investigated the feasibility of RMPC for autonomous driving [56]. The paper [56] has utilized RMPC in simulation to achieve obstacle avoidance driving of a car. This research offered several suggestions for smoothening the randomness using frequency domain sampling. This approach demonstrated more smooth steering input, but the controller could only drive in a straight line at a fixed speed. Additionally, only a simulation using pre-calculated inputs was used to confirm the effectiveness. In addition , this study came to the conclusion that greater computational power is required in order to use the RMPC in real-time, despite the ego car being represented with a bicycle model. In our earlier works, this requirement for computational improvement was also explored. [68] and [69]. Computation speed of linear MPC problems was accelerated using GPU in [68], however, this was limited to linear MPC problems. On the other hand, [69] highlighted some preliminary results showcasing the GPU's potential to speed up RMPC. Additionally, this study was restricted to simulations and constant speed straight-line driving. Based on these considerations, this work addresses the RMPC as a solution for autonomous driving control. First, it extends the previous works [56] and [69] with an updated vehicle model, thereby eliminating the restrictions of driving only in straight path and maintaining a steady speed. The randomization issue is then resolved using frequency domain sampling, and the RMPC is put into use along with the new sampling technique. Second, the suggested RMPC with frequency domain sampling is excecuted on GPU. Each sample is handled by a distinct GPU thread simultaneously in order to transfer the algorithm to the GPU. With this increased computing capability, RMPC is able to process more samples or even take into account a longer prediction horizon within strict real-time constraints. The proposed GPU implementation idea removes the requirement for large data transfers between the GPU and CPU. By not dividing the MPC into subproblems, this approach also minimizes the kernel generation overhead. This eliminates the time overhead problems that are frequently experienced in GPU-based systems [67]. Third, a distinct approach of step-by-step parameter selection for RMPC is proposed. Simulations and experimentations show how this new sampling technique paired with GPU realization enhances the control performance. An RC car with the same kinematics as a typical road car that is 1:10 scale is used to show the validity of the ideas that are suggested.

This study not only contributes to improving autonomous driving control but also highlights the potential of RMPC, frequency-domain sampling, and GPU implementation in various applications requiring fast real-time control of mechanical systems.

3.2 Problem setting

3.2.1 Task description

In-order to demonstrate RMPC and its improvements, a collision avoidance task as shown in Fig. 3.1 is chosen. There is an ego car controlled by the controller following a pre-defined reference path x^{ref}. The reference path, expressed in a local coordinate frame attached to the

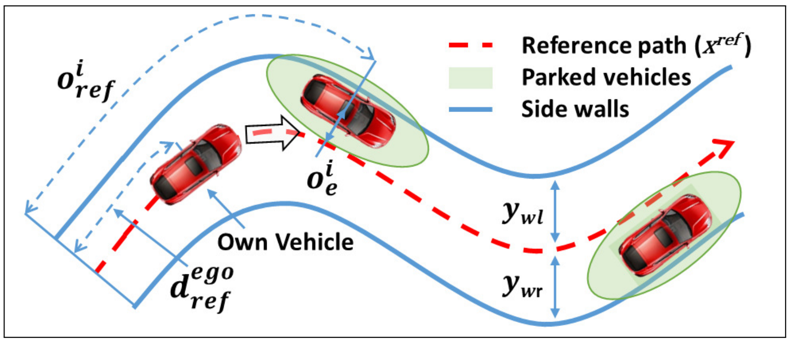


Fig. 3.1 Target environment

car ($\Sigma^l(t)$) as seen in Fig. 3.2 is defined as follows [68];

$$^l\chi_{ref}(t) = \{^l p_{ref,i}(t) | i = 0, 1, \cdots, N_p\} \tag{3.1}$$

$$^l p_{ref,i}(t) = \{^l x^p_{ref,i}(t), ^l y^p_{ref,i}(t)\}. \tag{3.2}$$

Here, $^l\chi_{ref}(t)$ is the reference path at time t, it contains N_p path nodes $^l p_{ref,i}(t)$, where $^l p_{ref,i}(t)$ denotes the relative position of the path node, $^l x^p_{ref,i}(t)$ and $^l y^p_{ref,i}(t)$, from the car's origin. The symbol l indicates local coordinate frame attached to the car and the symbol p indicates the path. Here the deviation of the car from the reference path, $y_e(t)$, is computed as the distance from the car to $O^p(t)$, the origin of the path coordinate frame $\Sigma^p(t)$ at the time of t, which is computed by applying the suitable interpolation between path nodes. The car ideally should drive along the center-line of this 6m wide path. It starts at the point $(d^{ego}_{ref}(t=0), y_e(t=0), \theta_e(t=0)) = (0,0,0)$, which is the origin. Variable t is the time index, d^{ego}_{ref} represents the distance covered by ego car along the reference path and y_e, θ_e represent lateral error and yaw error of ego car from the reference path. The car should also make necessary steering maneuver not to collide with the L parked cars on both sides of the street whose position is represented by $(o^i_{ref}, o^i_e), i \in \{1, 2, \cdots, L\}$ for the ith parked car. The controller controls the ego car motion by providing speed and steering angle commands.

The simultaneous motion planning and control (SMPLC) approach is used to approach this problem. An MPC with the required constraints serves as the controller. There is no need to plan a path first and then follow it in order to avoid obstacles because SMPLC combines planning and tracking into one problem while taking constraints and the car dynamic model into account.

An equivalent bicycle model (section3.2.2) is used to represent the dynamics of the car. The point-mass model was used in earlier works that implemented MPC for navigation using GPU to make the issue simpler. The preferred model is an equivalent bicycle model, given the faster rate of motion. This framework makes use of some input and safety constraints,

Table 3.1 Definition of parameters and variables

C_f	Cornering stiffness - Front tire	N/rad
C_r	Cornering stiffness - Rear tire	N/rad
m	Mass of the vehicle	kg
I_z	Yaw moment of inertia	kgm^2
l_f	Distance from gravity to front axis	m
l_r	Distance from gravity to rear axis	m
δ	Steering angle	rad
V	Forward velocity of vehicle (const.)	m/s
ρ	Curvature of reference path	1/m

as demonstrated in section 3.2.3, to maintain a safe distance from the obstructions and the sidewalls.

The nonlinear optimization problem is solved using a sample-based approach for reliable real-time performance. The sample-based approach used in this study is referred to as RMPC and has recently gained more popularity.

Methods based on random sampling frequently result in random variations in the control input. For tasks requiring precision, like driving, this is not advised. Preliminary findings from the co-earlier author's work [56] point to a significant benefit of frequency domain sampling for smooth driving control. An inverse discrete cosine transform (IDCT) was used in the frequency domain sample generation process in [56]. We show how the IDCT method operates on a GPU while smoothly operating our radio-controlled car.

3.2.2 State space equation for car dynamics

The vehicle behavior in this paper is assumed to be represented by an approximate bicycle model with its coordinate system along the reference path (Fig. 3.2). The state equation of this model is expressed as follows [70].

$$\begin{bmatrix} {}^{l}\dot{v}_Y \\ \dot{r} \end{bmatrix} = \begin{bmatrix} -\frac{a_{11}}{{}^{l}v_X} & \frac{a_{12}}{{}^{l}v_X} - {}^{l}v_X \\ -\frac{a_{21}}{{}^{l}v_{X^l}} & \frac{a_{22}}{{}^{l}v_X} \end{bmatrix} \begin{bmatrix} {}^{l}v_Y \\ r \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \delta, \tag{3.3}$$

where

$$a_{11} = (C_f + C_r)/m, \qquad a_{12} = -(l_f C_f - l_r C_r)/m,$$
$$a_{21} = (l_f C_f - l_r C_r)/I_z, \qquad a_{22} = -(l_f^2 C_f + l_r^2 C_r)/I_z,$$
$$b_1 = C_f m, \qquad b_2 = l_f C_f / I_z.$$

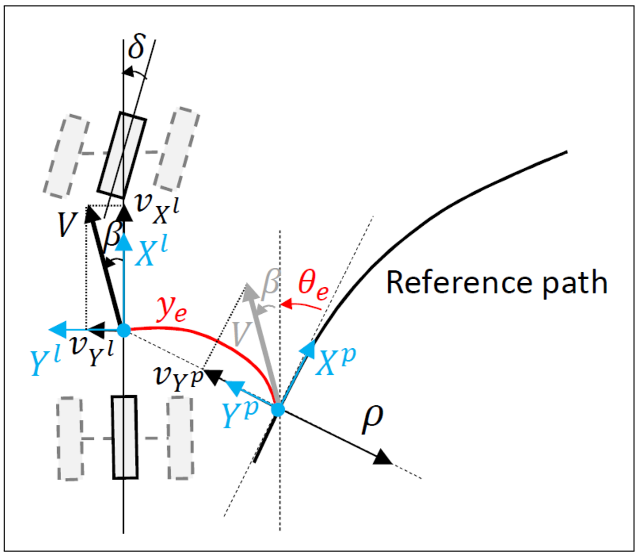


Fig. 3.2 Car represented using an approximate bicycle model.

where ${}^{l}v_X$ and ${}^{l}v_Y$ represents the longitudinal and the lateral velocity in local coordinate frame, respectively. These values are available from the Carsim software used for simulation or the motion capture system where the RC car experiments were conducted. Please refer to Table 3.1 for other variables. The nonlinear state equation that is given above is further transformed into an approximate bicycle model with it's coordinate system along the reference path. With the assumption of vehicle speed V being constant within prediction horizon , the slip angle of the vehicle β and θ_e to be small enough, the dynamics of the lateral tracking error y_e and the angle θ_e are summarized as follows (discretized with forward Euler's Method):

$$x(k+1) = A_d(k)x(k) + B_d(k)u_t(k) + W_d(k)\rho(k), \tag{3.4}$$

$$y(k) = Cx(k), \tag{3.5}$$

$$x(k) = \begin{bmatrix} y_e(k) & \dot{y}_e(k) & \theta_e(k) & \dot{\theta}_e(k) & \delta(k) \end{bmatrix}^T \tag{3.6}$$

$$u_t(k) = \delta^*(k) \tag{3.7}$$

$$A_d(k) = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 \\ 0 & 1-\frac{a_{11}}{V(k)}\Delta t & a_{11}\Delta t & \frac{a_{12}}{V(k)}\Delta t & b_1\Delta t \\ 0 & 0 & 1 & \Delta t & 0 \\ 0 & -\frac{a_{21}}{V(k)}\Delta t & a_{21}\Delta t & 1+\frac{a_{22}}{V(k)}\Delta t & b_2 \\ 0 & 0 & 0 & 0 & 1-\alpha\Delta t \end{bmatrix} \tag{3.8}$$

$$B_d(k) = \begin{bmatrix} 0 & 0 & 0 & 0 & \Delta t \end{bmatrix}^T \tag{3.9}$$

$$W_d(k) = \begin{bmatrix} 0 & (a_{12} - V^2(k))\Delta t & 0 & a_{22}\Delta t & 0 \end{bmatrix}^T \tag{3.10}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{3.11}$$

Where ρ represents the curvature of the reference path at the nearest point and δ^* represents the tire angle reference. Actual tire angle δ is considered to have a first order delay from δ^* to represent physical delay in steering system. Readers are referred to our previous work [68] for further details.

3.2.3 Input and safety constraints

There are two constraints in this formulation, one is the hard constraint to ensure feasible input commands and the other one is to prevent entry into the prohibited area.

The range of control input u_t and its rate of change Δu_t are constrained as follows to maintain physical limits of steering system and to prevent urgent steering action, respectively.

$$|\, u_t \,| < 0.1745 \ (\approx 10 \text{ [degrees]}) \tag{3.12}$$

$$|\, \Delta u_t \,| < 0.35 \text{ radians/sec } (\approx 20 \text{ [degrees/sec]}) \tag{3.13}$$

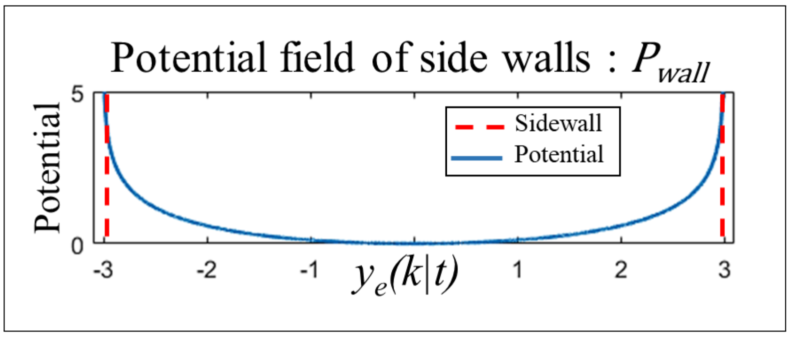


Fig. 3.3 Potential field of sidewalls.

For each parked car, an ellipse around them defines the prohibited area with a safety margin. This area for ith parked car is expressed by an inequality

$$\left(\frac{d_{ref}^{ego}(t)-o_{ref}^{i}}{r_a^i}\right)^2+\left(\frac{y_e(t)-o_e^i}{r_b^i}\right)^2>1$$
$$\forall i \in \{1,2,\cdots,L\} \tag{3.14}$$

where L is the number of cars parked, (o_{ref}^i, o_e^i) is the position of i th parked car. The length of the major and minor axis of the ellipse around ith parked car are r_a^i and r_b^i, respectively.

The areas beyond sidewalls are also expressed as prohibited areas represented with a logarithmic function as explained in the next subsection (3.19).

3.2.4 Cost function and reference state

The controller's objective is to keep the car in the middle of the road and avoid hitting any obstacles. When examining the given case in greater detail, it becomes clear that there is a conflict between the requirements to follow the street center and avoid obstacles. In addition to avoiding obstacles, it's crucial to maintain a safe distance from them rather than taking the route that passes closest to the obstruction. By including a potential field term in the cost function, this is met. Weight parameters are used to balance terms that adhere to the reference path and terms that avoid colliding. Finally, to maintain a safe distance from the side walls, another potential field is also added.

The cost function $J(u_t)$ for an input series $u_t=[u(0|t),u(1|t),\cdots,u(N-1|t)]$ to be optimized at time t is

$$\begin{aligned}J(u_t) = &\sum_{k=1}^{N-1} s_0(k|t)\left(\phi^T(k|t)Q\phi(k|t) + \Delta u(k|t)^T R\Delta u(k|t)\right)\\ &+\phi^T(N|t)Q_f\phi(N|t)\\ &+Q_{obs}\sum_{k=1}^{N-1} s_j P^j_{obs}(k|t) + Q_{wall}\sum_{k=1}^{N} P_{wall}(k|t) \end{aligned} \quad (3.15)$$

$$\begin{aligned}\text{where}\ \ \phi(k|t) =& y(k|t), & (3.16)\\ \Delta u(0|t) =& 0,\ \Delta u(k|t) = |u(k|t) - u(k-1|t)|_1 \\ & \forall k \in \{1,2,\cdots,N-1\}, & (3.17)\end{aligned}$$

where N, ϕ and Δu represents prediction horizon length of MPC, state error and the time difference in control input, respectively. The representation $u(k|t)$ is the value of u at prediction horizon step k at time t. $Q = diag(Q_{y_e}, Q_{\theta_e})$ and R being weight parameters, balance the control performance to the effort. Q_f acts on the last step in the prediction horizon as a penalty to the residue.

This framework enables us to follow any reference path x^{ref} that is needed, the reader can refer to our previous work [68] for a detailed study on the path tracking performance of this dynamic model. Further details on the weight parameter matrices and the terms $s_i(k|t)$, $s_0(k|t)$ can be found in author's previous work [56]. When the ego car approaches a parked car i, the switching parameter s_i approaches to 1. This makes the repulsive force more significant. Otherwise, following the reference state is of higher priority.

Q_{obs} and Q_{wall} works as weight parameters for the potential field around parked cars and sidewalls. Potential function around jth parked car is expressed by P^j_{obj}:

$$P^j_{obj}(k|t) = C_j \exp\left(-\left(\frac{d^{ego}_{ref}(k|t) - o^j_{ref}}{r^j_a}\right)^2 - \left(\frac{y_e(k|t) - o^j_e}{r^j_b}\right)^2\right) \quad (3.18)$$

where $[d^{ego}_{ref}(k|t), y_e(k|t)]$ are the ego car position along reference path predicted for prediction horizon step k at time t. C_j, r^j_a and r^j_b allows obstacle-specific adjustments in potential field area and magnitude. While r^j_a and r^j_b adjusts the proportions of the elliptical obstacle area around the obstacle, C_j adjusts it's potential. Side walls have their own potential field P_{wall}:

$$\begin{aligned} P_{wall}(k|t) =& (\log|y_{wl}| + \log|y_{wr}|) \\ & - (\log(y_{wl} - y_e(k|t) + \log(y_e(k|t) - y_{wr})) \end{aligned} \tag{3.19}$$

where $y_{wl} = 3, y_{wr} = -3$ are side wall locations. P_{wall} is designed to increase steeply close to the walls, as shown in fig 3.3. With assumption that the car is travelling parallel to the side walls, it is sufficient to use a simpler log function for P_{wall} while P^j_{obj} needs a slightly complex exponential representation.

3.2.5 Formulation of input optimization problem

The optimization problem to be solved in every control step in order to generate optimum control input series can be formulated as follows:

given

$$x(0|t) = x(t), x^{ref}, \tag{3.20}$$

find

$$u(k|t), \quad (k \in \{0, 1, \cdots, N-1\}) \tag{3.21}$$

which minimize

$$J(u_t = \{u(k|t)\}), \quad (k \in \{0, 1, 2, \cdots, N-1\}) \tag{3.22}$$

subject to

$$\begin{aligned} &\text{Input constraints (3.12), (3.13)} \\ &\text{Prohibited area constraint (3.14), (3.19)} \\ &\text{Car dynamics (3.4).} \end{aligned} \tag{3.23}$$

3.3 Frequency domain sampling for semi optimal solution

The issue at hand is a non-linear optimization problem that needs to be resolved in real-time. A non-linear vehicle model and non-linear constraints on safe driving are both present. Approximating the non-linear constraints for the solver in some way is a typical method to find the solution [62]. This paper tries to eliminate these approximations by the application of a random sample based approach. This method is proven to provide a semi optimal solution which is close enough to the optimal solution, once the necessary sample size is considered [47]. The generation of samples initiates the process of this optimization technique. A predetermined distribution of random numbers is used to select N_s sample sequences with length N. The car's future trajectory must then be determined, assuming it will adhere to each of these model sequences. The samples that are impractical are now removed using constraint-based filtering. There is a threat to safety if the majority of the samples are discovered to be impractical. The controller should then either request manual driving or switch to an emergency stop mode. This comes after using a cost function to determine the cost related to each sample. The element at the start of the lowest cost series is then chosen, and it serves as the ego car's input. For each control cycle, the controller repeats these steps.

3.3.1 Generating smoother input samples by sampling in the frequency domain

It is anticipated that randomly generating samples will also generate randomness in the control input. This randomness is not advised for safe driving performance because the given control input directs the car's steering. The samples produced by the suggested sampling technique are produced in the frequency domain. An Inverse Discrete Cosine Transform is then used to transform these into the time domain (IDCT). It is possible to get samples that drive the car more smoothly by following this process.

The generation of samples is thoroughly described in the following set of equations.

$$u_{IDCT}(0|t) = u(t-1) \tag{3.24}$$

$$u_{IDCT}(k|t) = u_{IDCT}(k-1|t) + \Delta u_{IDCT}(k|t), \quad \forall k \in \{1,\cdots,N\} \tag{3.25}$$

$$\Delta u_{IDCT}(t)^T = \gamma D U_{IDCT}(t)^T \tag{3.26}$$

$$U_{IDCT}(f|t) \begin{cases} \sim \mathscr{U}(-1,1) & \text{if } f \leq F_{c/o} \\ = 0 & \text{other} \end{cases} . \tag{3.27}$$

Here, $D \in R^{N \times N}$ is the coefficient of IDCT, f is the index representing the frequency component and N is the prediction horizon length. Each element of D is calculated as follows:

$$
\begin{aligned}
D_{i_i j_i} = & \sqrt{\frac{2}{N}} k_{i_i} \cos\left(\frac{(i_i - 1)(j_i - 1/2)\pi}{N}\right), \\
& i_i \in \{1, 2, \cdots, N\}, j_i \in \{1, 2, \cdots, N\}
\end{aligned}
\tag{3.28}
$$

where k_i $(i_i = 1, 2, ..., N)$ is a normalizing factor that has two values

$$
k_i = \left\{ \begin{array}{cc} \frac{1}{\sqrt{2}} & i_i = 1 \\ 1 & i_i \neq 1 \end{array} \right\}. \tag{3.29}
$$

$U_{IDCT}(t)$ is sampled from a uniform distribution $\mathscr{U}$ ranging (-1,1) in this case. γ is a parameter that can be used to adjust the resulting input rate of change according to (3.13). $F_{c/o}$ is a cut-off threshold that prevents higher frequency components in the resulting input sequence. The smaller the $F_{c/o}$ the smoother in the resulting input series. Unless specified, an $F_{c/o}$ of 15 is used in this paper. If any of the element u_t generated in (3.25) is found to violate the constraint (3.12), a new random number is generated to replace the $U_{IDCT}(f|t)$ until u_t satisfies (3.12). The control performance with and without IDCT can be seen in Fig. 3.4 (Ns=500). The graphs show visible improvement in control signal smoothness and steering performance. In summary, the contents of IDCT is essentially random number generation followed by some matrix multiplications and series making. The following sections will demonstrate how the above-mentioned operations were efficiently ported to GPU.

3.3.2 Validation of proposed sampling method

Before using RMPC for real experiments, its performance is evaluated in simulation-based experiments. Since the experiments are performed in a 1:10 scale car, a high fidelity car simulator is necessary for simulations. Carsim by Virtual Mechanics Inc. is used owing to its excellent vehicle and environment models. The controller issues an updated control input in every control cycle. In the simulation experiments, the control interval dT is set as 0.1 seconds. The prediction horizon length is set as $N = 50$. This means that the controller calculates the optimal input considering vehicle behavior predictions of 5 seconds into the future. The following combination of parameters resulted in the best possible tracking performance. $Q_f = 1$, $Q = \text{diag}(10, 10)$, $R = 3000$, $Q_{obs} = 3000$ and $Q_{wall} = 5$, respectively. The parameter γ is set to 1. The parked cars are at the positions $o^1 = (50, 0.85)$ and $o^2 = (80, -0.85)$, respectively. Since the optimization is based on random samples, there

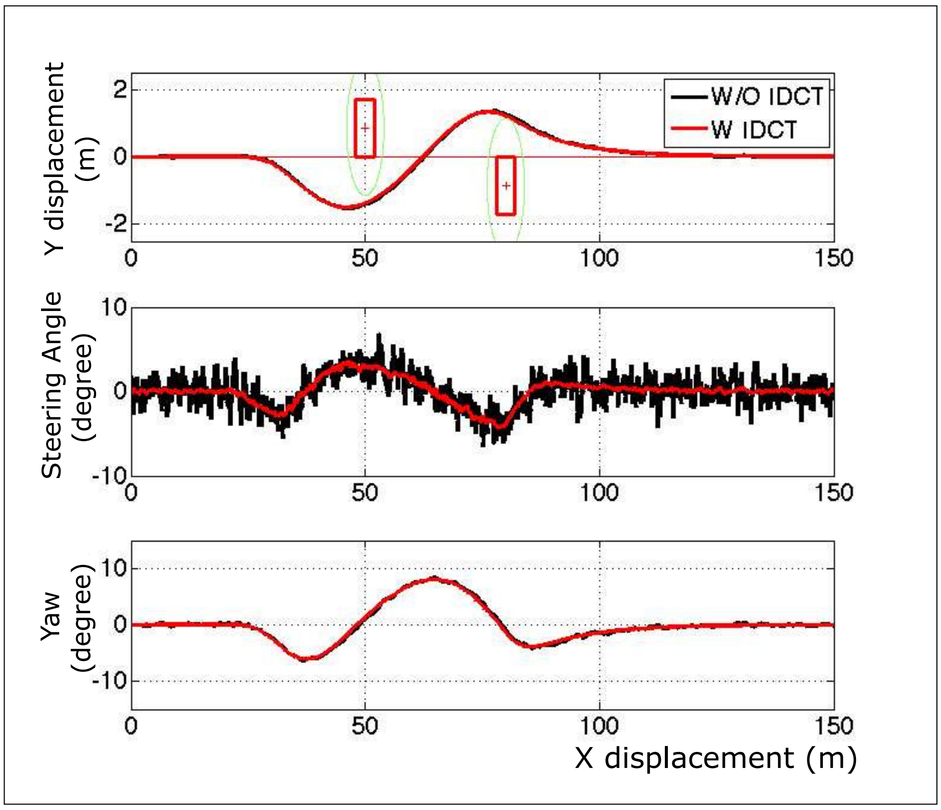


Fig. 3.4 Comparison of control performance - IDCT vs No IDCT

is an obvious question of its closeness to the global optimum. There have been some previous works [47] that quantifies the minimum sample size for the solution to be global with a defined probability level. Referring to [47], it is calculated that a sample size of $N_s > 458$ is necessary for the solution to have confidence level of $99\%(\alpha_{believe} = 0.01)$. Hence, the controller is tested at a sample size of $N_s = 500$. Fig. 3.5 demonstrates the performance. Tracking performance is very accurate without any intrusion to the obstacle area which is prohibited. While the author's previous works [56] were limited to the performance of the controller at sample sizes less than 500, this paper tries to go beyond this number. Even though $N_s = 500$ provides sufficient performance at the simulation conditions, sample sizes beyond 500 are found to provide better controller performance. This can be seen in Fig. 3.5, where at 1000 samples, the control performance is clearly superior than at 500 samples. The majority of control tasks have multiple control variables and stricter constraints. Such cases would clearly demand a higher number of samples to be processed in real-time. Another interesting trend in the case of driving is related to driving speed. The faster the car is, the shorter the control interval has to be. Both these conditions demand higher computational performance. But due to hardware limits, the sample size $N_s < 500$ was the limit that a CPU

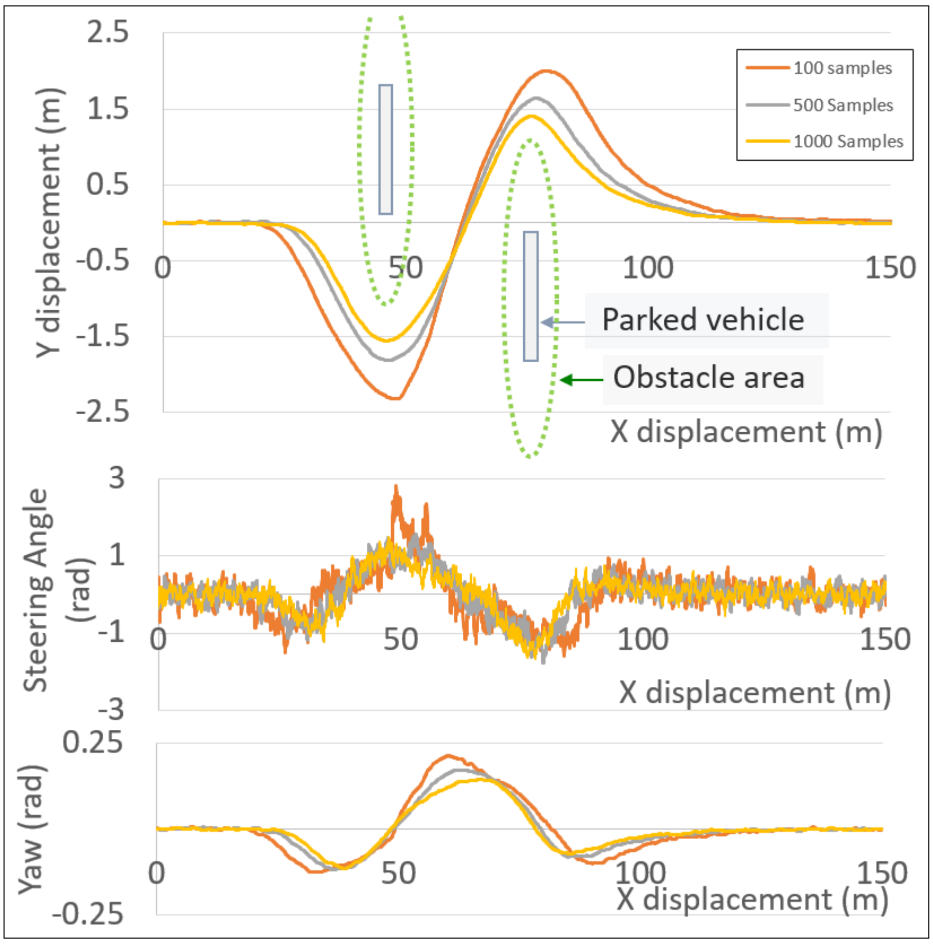


Fig. 3.5 Control performance at various sample sizes

based implementation could process in real-time. This limitation in computation power was also expressed in [56] and [62].

3.4 Implementation using GPU

3.4.1 Computational steps involved

The process of calculating the optimum control input can be expressed in a sequence of 6 steps as described by Table 3.2. Once the present state variables are obtained from the car, the controller performs steps 1 to 6. The final step does a minimization of all the samples based on a cost function and comes up with the best sample sequence. The second element in this sequence will be the optimum input for the car in the next step. This will be then sent to the car. This process is repeated in every control cycle. The following subsections are

Table 3.2 Steps in computation

Step	Function	Equation reference
1	Random Number Generation	(3.27)
2	IDCT	(3.26)
3	Series Generation	(3.25)
4	State Matrix Calculation	(3.4)
5	Cost Calculation	(3.15)
6	Minimization of cost	–

Algorithm 1 Randomized MPC

```
1: for Every time step do
2:     Generate $Ns$ input samples of length $N$ acc. to 2.
3:     Initialize state matrix.
4:     for $Sample = 1,2,\dots,Ns$ do
5:         for $Length = 1,2,\dots,N$ do
6:             Calculate the extended state matrix acc. to 1.
7:         end for
8:     end for
9:     for $Sample = 1,2,\dots,Ns$ do
10:        for $Length = 1,2,\dots,N$ do
11:            Calculate the cost for each sample acc. to 3.
12:        end for
13:    end for
14:    Find minimum cost sample $Smin$.
15:    Return second element of $Smin$ as next control input.
16: end for
```

dedicated to the detailed contents of these steps and the methods in moving them to support parallel computation using GPU.

3.4.2 CPU baseline

In order to have a better understanding about the improvements by parallel computing, The controller was first implemented using a CPU based algorithm. The steps are expressed in Algorithm 1. The CPU used in this study is a 2.2GHz Intel Core i5. CPU program uses a random device function of the C++ Numerics library [71] to generate random numbers. Please note that the steps from 2 to 5 in Table 3.2 are performed on every element, one element at a time using two 'for loops'. First one is a 'for loop' that index the sample number Ns, the second one for the length of the sample N. This makes up $Ns \times N$ calls to the CPU one after the other.

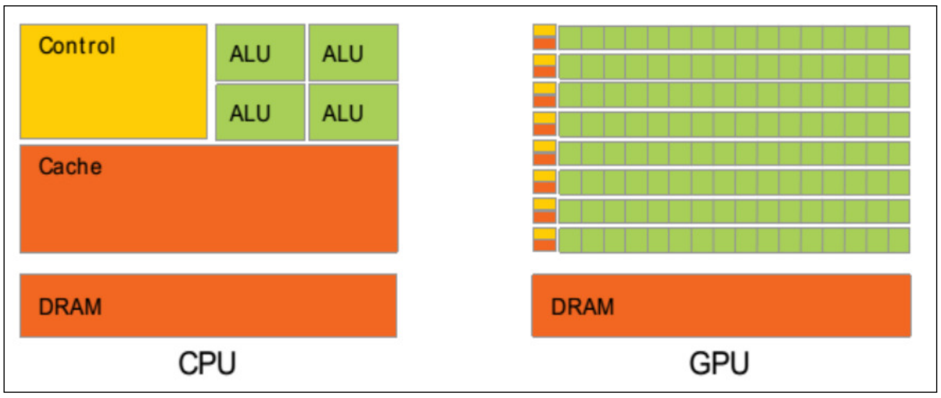


Fig. 3.6 CPU vs GPU architecture

3.4.3 Code design for GPU

The MPC algorithm's various steps were segmented into six distinct phases, as outlined in Table 3.2. These phases underwent a thorough examination to identify potential enhancements through adaptation to a 2560-core GPU (specifically, the NVidia GeForce GTX1080 running at 1.7GHz). The fundamental hardware distinction between CPUs and GPUs is illustrated in Fig. 3.6, as described in [72]. Unlike a typical CPU, which possesses numerous fast arithmetic logic units (ALUs) sharing a common cache memory and being controlled by a unified controller, GPUs feature thousands of ALUs, each equipped with its own controller and cache memory. Given that all the ALUs within a GPU can simultaneously execute a thread of operations in parallel, the primary implementation concept involves assigning a sample point to each GPU thread, as depicted in the illustrative representation shown in Fig. 3.7. Random samples are stored within an array of size $N_s \times N$, and an equivalent number of GPU threads are allocated as a block, sized at $N_s \times N$.

The first step of random number generation is performed with Pseudo-random number generation function in CURAND library [73], which is a part of NVidia's compute unified device architecture (CUDA) library [72]. Step 2 that performs the IDCT operation (As seen in (22)) and Step 5 that calculates cost (As seen in (10)), are independent at each discrete sample points. Being independent, they can use as many GPU threads as the hardware allows simultaneous operation. Matrix multiplication operations associated with Step 2 can also use the maximum possible capacity of GPU. These steps while using one sample point per GPU thread gives $N_s \times N$ times faster computation. The remaining steps in the computation like Steps 3 and 4 depend on previous prediction horizon/series value to compute next. These steps have to be performed in N steps, calculating all Ns samples at a given time using Ns number of threads. Even here Ns times improvement in computational time is obtained over CPU.

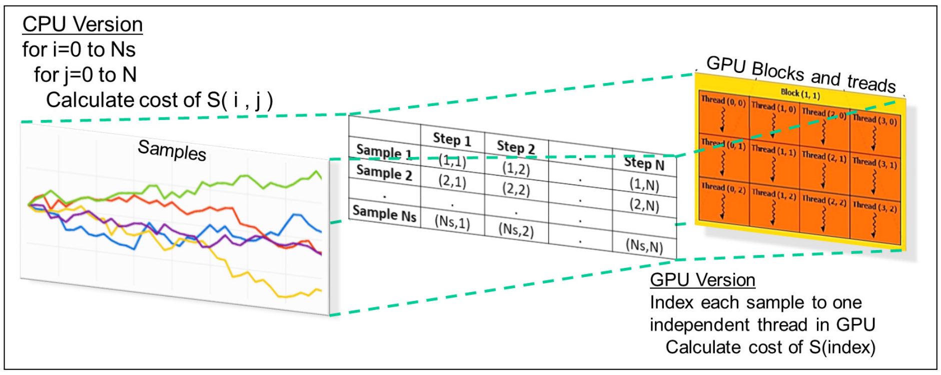


Fig. 3.7 GPU implementation image

As described in the survey paper [67], the majority of GPU based implementations are faster in calculation but suffer from a high overhead of memory transfer time and kernel creation time. The GPU implementation presented here avoid these problems in following ways:

- By generating the random numbers within the GPU memory (step 1 in table 3.2), all the steps of computation are kept within the GPU and its memory. Only the present state of the car is transferred to GPU and the optimum input is sent back.
- Certain problem-parallel approaches in literature splits the MPC to smaller problems by dividing the prediction horizon and assign each section to different GPU kernels. Thanks to the random sample based method, our data-parallel approach does not split the MPC into subproblems, limiting the kernel creation time to the minimum.

3.5 Evaluation using RC car

It is anticipated that using GPU, control signals of the same quality as those produced by CPU will require significantly less computation time. This has a number of advantages when used for vehicle control. Previous simulation-based works did not account for localization, error estimation, or communication delay delays. By relying solely on simulations, model errors and various other noises were also disregarded. As a result, extensive testing of the controller is done with an RC car in environments that are as close to real life as possible.

3.5.1 Experiment setup

The RC car used is a Tamiya 1:10 scale car as shown in Fig. 3.8. The control pulse width modulation (PWM) signals are sent from the control PC via Wi-Fi to the on-board "Raspberry

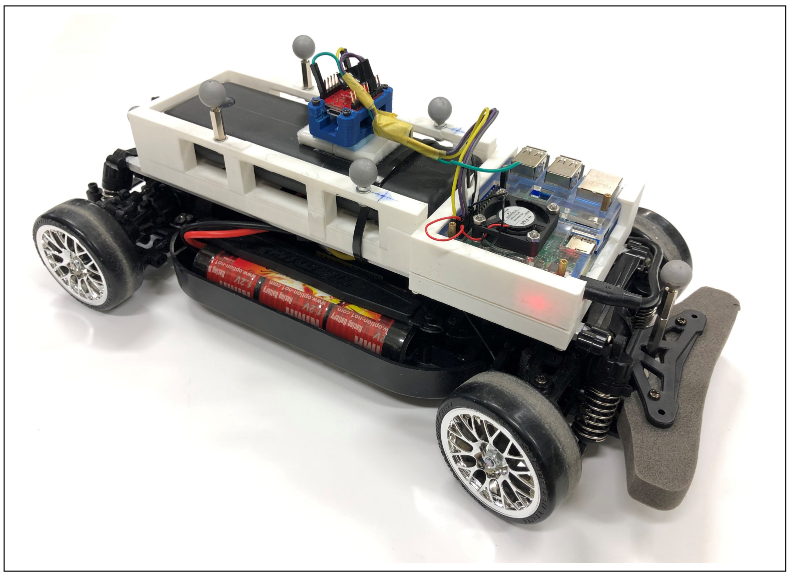


Fig. 3.8 RC car with attached computer and markers for localization

Pi" computer in the vehicle. The framework for communication is ROS. A camera-based motion capture system called "Motive" is used to capture the position of the car in real-time. The maximum control frequency is less than the 200Hz frequency at which the motion capture system operates. The length N of the prediction horizon is set at 30. (to predict 3 metres ahead).

3.5.2 Benefits of running higher control frequency

Running the controller at a higher frequency is one benefit of using a GPU to perform computations more quickly. As a result, obstacle avoidance is made possible at speeds faster than the CPU. Under the condition of $N = 30$, and minimum Ns of 500, the maximum control frequency using CPU is 30Hz. This controller is found to work fine at a speed of 3.6 km/h[1]. However, it is discovered that the CPU-based system strikes the obstacles at a speed of 5.1 km/h. This is because CPU-based implementations are limited to operating at a certain control frequency. The GPU performs smooth avoidance up to a speed of 11.5 km/h while operating at 200 Hz and 1000 samples. The details of the experiment is shown in Fig. 3.9 and the video [75].

[1]In a real car, this correspond to $3.6 \times 10 = 36$km/h [74].

Table 3.3 Cost function at different sample sizes

Sample Number	Cost $(\times 10^4)$	Improvement over CPU $(\%)$
100	16.30	_
500	13.62	16.4
1000	12.43	23.7
5000	11.19	31.4
10000	10.37	36.4
20000	9.77	40.1
30000	9.70	40.5

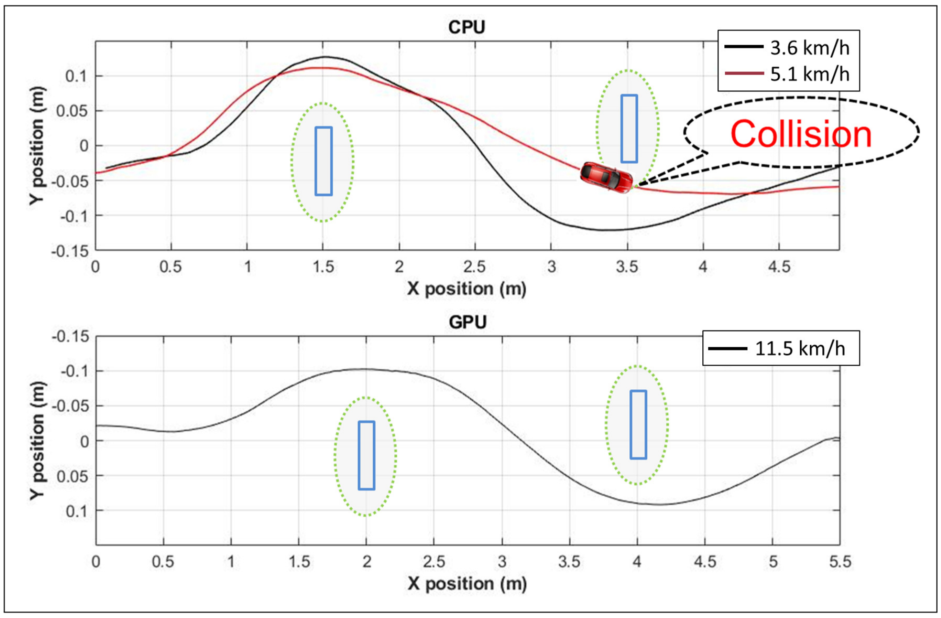


Fig. 3.9 Speed limit comparison

3.5.3 Benefits of running higher sample counts

The ability to consider more samples at each control interval is another benefit of using the faster computation speed. A larger sample size is anticipated to bring the solutions closer to the global optimum, as was previously discussed in section 3.3.2 The control frequency in the RC car tests was set to 100Hz, and the sample size was increased. The results are indicated in Table 3.3 and Fig. 3.10. Only 100 samples can be processed in real time by the CPU at 100Hz. The GPU based controller can have lower cost values by using more real-time samples. Higher sample sizes show a low reduction in cost value, but the car behavior is still noticeably smoother and maintains a better safety distance.

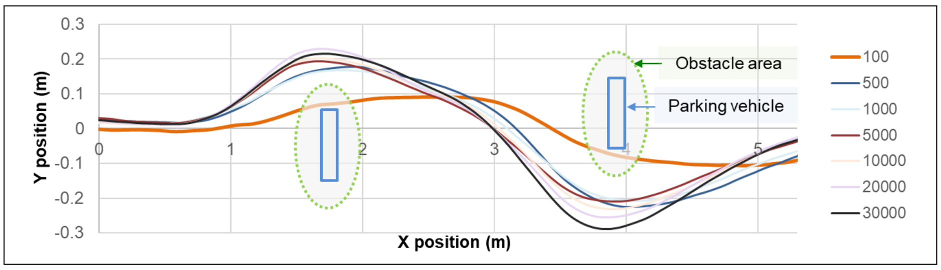


Fig. 3.10 Sample size comparison

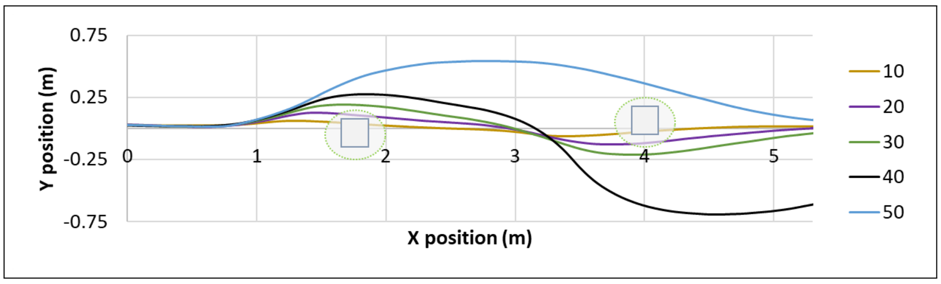


Fig. 3.11 Tracking performance at various prediction horizon lengths

3.6 Discussion on parameter selection for RMPC

RMPC controllers are characterized by the samples used and the frequency at which they are updated. The samples are characterized by the number of samples Ns and length of each sample N. Let us denote the control interval as dT. The selection of these parameters is not as easy as it seems due to their relationship with each other. Increasing Ns or N forces dT to increase. For a fixed N, there could be multiple combinations of Ns and dT that satisfy computational limits.

The recommended procedure to identify these parameters starts with the identification of N. N is decided first because of it's unique effect on control behavior. Value of N has to be bounded with both upper and lower limits as seen in Fig. 3.11. On the other hand, acceptable values of Ns and dT could only be bounded on one side, minimum Ns and maximum dT.

N can be identified with a few experiments as demonstrated in Fig. 3.11. The values for Ns and dT are chosen intuitively for this step. It can be seen that the RC car, while performing the obstacle avoidance task as explained in Section 3.5, has an optimal range of prediction horizon length N for best tracking performance. Here, the best N value was found to be 30 steps.

Once the value of N is identified, it is recommended to choose the optimal dT corresponding to the speed of the system. Rule of thumb is that a car moving at double speed will cover double the distance in the given time. Hence control frequency should also be doubled. It is always recommended to have some tolerance in the dT value considering the computational time slightly varies depending on situation.

After choosing the N and dT, it is recommended that the maximum sample number is chosen considering given computational resources. This is because the cost function is found to decrease monotonously as the sample size increases as seen in table 3.3. This method of selection was employed during parameter selection for the RC car experiments and simulations presented in this work.

3.7 Modification for cooperation with speed control on curved roads

The simulations and experiments covered up until section 3.6 deal with an obstacle avoidance task on a straight road at constant speed. It is determined that this task is the best option for demonstrating the impacts of different control parameters and computational speed on the suggested RMPC controller.

This controller performs flawlessly on all types of roads, even those with sharp turns. The ego car can be seen making a sharp turn while facing a parked car at the curve's exit in Fig. 3.12. Control parameters have been selected to match section 3.3.2. It is seen that the ego car avoids collisions with ease.

With the addition of a speed controller that slows the vehicle down when higher steering angles are ordered, the constant speed premise is replaced. The input vector (3.7) becomes

$$u_t(k) = [\delta^*(k), v(k)]^T \tag{3.30}$$

Where,

$$v(k) = C_f(\delta^*_{avg}(k)), \;\; \delta^*_{avg}(k) = \frac{1}{n}\sum_{i_{ma}=1}^{n} \delta^*(k - i_{ma}).$$

C_f is a cubic function of δ^*_{avg}. δ^*_{avg} is averaged over n steps in the past (here, $n = 10$). k represents the control step and the index i_{ma} is used for indexing the moving average. In Fig. 3.12, the speed profile during collision avoidance is seen to be smooth. Speed control can be easily replaced with any other type of controller because the proposed lateral controller is independent. Please take note that in the example provided, the RMPC is only in charge of steering angle; the vehicle speed will be a state variable for the RMPC.

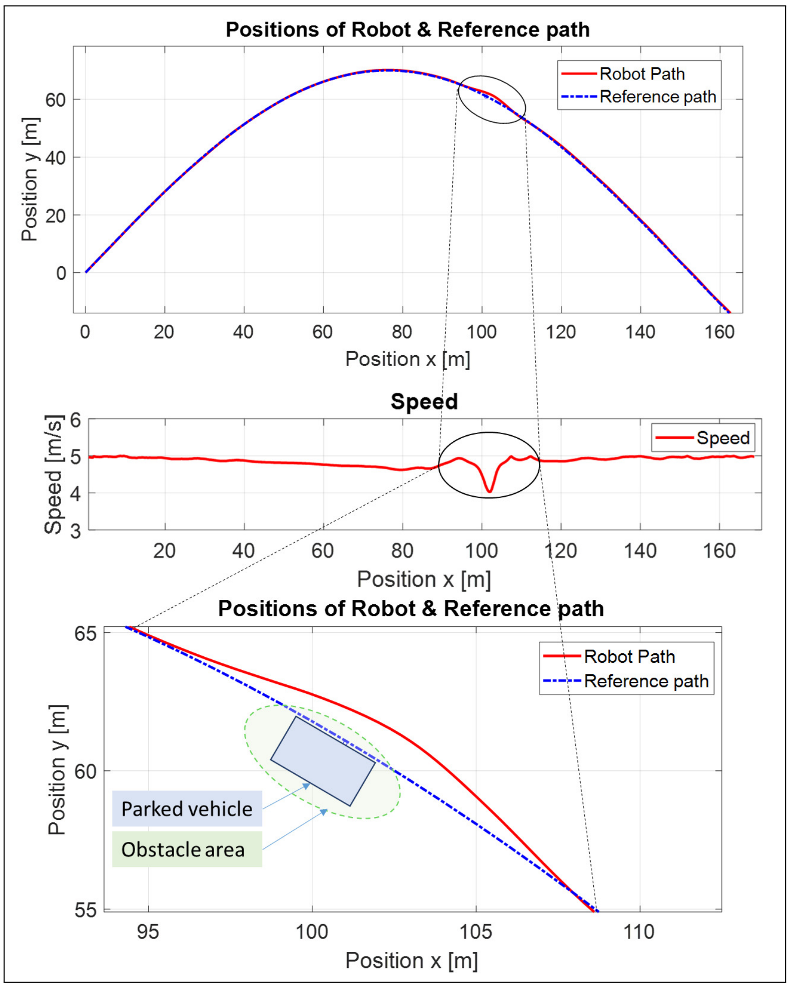


Fig. 3.12 Path tracking with curves and speed control

3.8 Conclusion

A randomized nonlinear model predictive controller for autonomous driving has been presented in this chapter, with a focus on the obstacle avoidance task. Despite the nonlinear nature of the vehicle model and the obstacle constraints, the proposed scheme allowed for direct consideration of nonlinear constraints by employing a randomized optimization method. The IDCT method was used to create random samples from the frequency domain for optimization. This avoids unfavorable control input oscillation, which is particularly crucial for autonomous driving. This chapter also looked into the effects of using a graphics processing unit to implement randomized MPC. The real-time performance ceiling was raised by using GPU. Additionally, a method for randomized MPC parameter selection has been discussed. The proposed scheme and improvements were confirmed in simulation and experiments using an RC-car. In comparison to a CPU-based controller, the RC-car experiment has demonstrated higher driving speeds and better control performance. Other types of control systems that require high control performance under nonlinear constraints can use our suggested GPU implementation scheme.

Chapter 4

Design of Considerate Autonomous Driving Using Pedestrian-Aware Model Predictive Control

4.1 Introduction

Self driving on closed highways has reached very close to realization. Major automobile makers are set to provide autonomous driving on such highways in the 2020's. Autonomous driving on the highway can be controlled if certain attributes are fixed, such as compliance to traffic rules and lane navigation. There have been previous works that express them mathematically as deterministic functions or constraints built on vehicle state perceived by various sensors. Some achievements can be found in [22] and [76–78]. The works [76] and [77] proposed rule-based highway autonomous driving (AD) solutions, while [22] and [78] propose model predictive control solutions to highway AD. Considering the fact that majority of our everyday driving includes driving on roads in shopping and residential areas, fully autonomous driving in downtown areas is much more difficult than on the highway. It is still a challenging task because the driving environment is often shared with other agents like pedestrians. Even if the priorities of pedestrians and cars in shared scenarios are generally defined by rules, such negotiations are very common, particularly in residential areas, parking lots, and so on [79]. While considering developing markets like China and India [80], such interactions are very much part of daily life. In China, the person who comes to the way first has the legal right of way. This means that the motorist does not need to yield to the pedestrian [81].

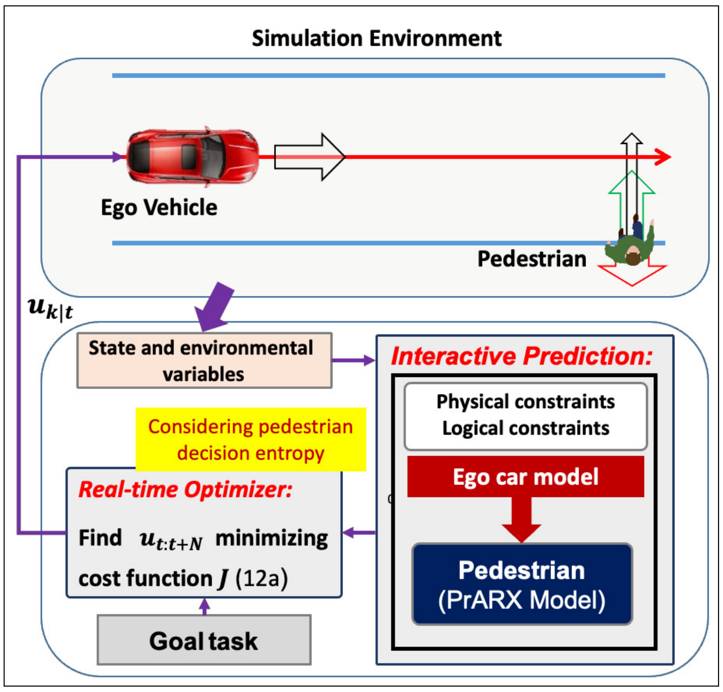


Fig. 4.1 Control architecture for considerate driving

It is generally very hard to predict the movement of pedestrians mathematically. The action of the pedestrian, however, can sometimes be implicitly controlled by changing the action of the ego vehicle. This can be regarded as a behavioral interaction between the car and pedestrians. Behavioral interaction implies that a driver tries to convey his or her intentions to pedestrians by doing an action for the purpose of achieving a reaction that the driver expects [82]. Such behavioral interaction reflects the balance between the driver's own intention to keep driving and their consideration of the comfort of the pedestrians. For example, human drivers often reduce the speed in advance for a pedestrian who intends to cross so that he/she may cross before the driver proceeds. In the same way, human drivers often accelerate and pass a yielding pedestrian when they expect us to pass. In both cases, it can be seen that the consideration of the pedestrian's stressless decision-making made it easier to reach a fast behavioral consensus. In this paper, autonomous driving that can realize such behavioral interaction is defined as "Considerate Driving", where the driver is considerate towards the comfort of the pedestrians and drives accordingly. In comparison to being conservative and yielding all the time, or being aggressive to force a pedestrian to yield, it is better to be considerate. This achieves a safe behavior consensus faster, thereby enabling the car to realize natural driving. Generally speaking, the car makes the pedestrian's decision-making easier by reducing his or her internal confusion level.

To enable stressless behavioral interaction, an index to measure the comfort level of the pedestrian during the negotiation is needed. The comfort levels are generally related to how confused the pedestrian is. The confusion levels inside a traffic participant has been studied by author's previous works [6] and [83]. These works has proven the parameter "entropy" to be an effective measure of internal confusion of human traffic participants. The term "decision entropy" is used in this work to represent the internal confusion of the pedestrian. Trying to minimize the entropy of pedestrians as a part of car's control objective is expected to enable faster behavior consensus and thereby facilitate faster interactions. Calculating the decision entropy necessitates a pedestrian model that explicitly provides the decision making process with probabilistic measure for different possible predefined motion modes.

In this paper, a novel pedestrian-aware MPC approach has been proposed considering the vehicle-pedestrian interaction in a shared road environment. The major contributions can be summarized as follows:

- Proposes a novel multi-mode probability weighted ARX (PrARX) model [6] that includes the interaction behavior between car and pedestrian.
- Proposes an MPC framework that can realize considerate driving using the proposed pedestrian model.
- The level of aggressiveness and the level of consideration to pedestrian's decision making comfort are tunable.
- Optimization method being random sample based, is compatible with accelerated computation on GPUs, hence scalable to multiple pedestrians.

The proposed PrARX model considers multiple pedestrian modes, hence predicts multiple future pedestrian trajectories with corresponding probabilities in each step. A risk parameter that is tunable based on collision probability helps to customize the level of risk that the controller can take, making it more personal to different drivers. The pedestrian's decision entropy is introduced as a component of the MPC cost function. By varying the weight on this entropy cost, the vehicle behaves considerately making the pedestrian's decision making easier. The MPC problem is formulated and solved using a randomized optimization algorithm ([47, 56, 84]). Driving is a task where safety is crucial, so real-time computation is required. The calculation time required to determine the cost corresponding to samples for each mode a pedestrian can use is one anticipated bottleneck of the suggested method. Our earlier research [69] has shown that using a GPU rather than a CPU can increase the real time calculation limit for the randomized algorithm. The proposed MPC has been created to use a GPU for computation.

The remaining part of this paper is organized as follows. In section 4.2, we discuss the relevant related works. An overall control architecture for considerate driving is explained in 4.3. This section also discusses the interactive situation addressed in this work. In 4.4, the dynamic models used to represent the car and the pedestrian behavior are described. The sections 4.5 and 4.6 explain how the safety constraints are formulated, and how the decision entropy of the pedestrian is incorporated into the MPC, respectively. Section 4.7 discusses how the pedestrian-aware MPC problem is formulated to consider the probabilistic nature of the collision avoidance. In section 4.8, extensive simulations are carried out to demonstrate the novelty of our proposed controller.

4.2 Related Works

4.2.1 Considerate driving

The term considerate driving as proposed here has been noted in [85] and [86]. [85] defines considerate driving as a form of prosocial behavior. While prosocial behaviors are defined as "broad range of actions intended to benefit one or more people other than oneself" [87]. Prosocial driving is defined as "driving behaviors that potentially protect the well being of passengers, other drivers, and pedestrians, and that promote effective cooperation with others in the driving environment" [88]. While the previously mentioned works discuss considerate driving by human drivers, [86] discusses a considerate system design approach for the driving environment with interactions among the ego-car and the other traffic participants. This work follows the definition of considerate driving from [85].

4.2.2 Prediction models

The most important component to enable considerate driving is an accurate behavior prediction model with two essential features. Firstly, it must include the dynamics of interaction between pedestrian and another agent, as seen in [89–94]. Secondly, the structure of the model should enable entropy calculation during decision making, as seen in [95, 96, 6, 83]. Among the works on interaction between pedestrian and another agent, [89] presents a novel dynamic bayesian network (DBM) for pedestrian path prediction, [90] approaches the same task as an unsupervised learning problem. Growing hidden Markov models (GHMMs) and bayesian human motion intentionality predictor (BHMIP) are used along with a social forces based motion model in [91] and [92], respectively. Both the works claims to yield a significant performance gain in comparison with the standard constant velocity-based models. In order to predict human crowds behavior, [93] develops an approach that models the joint distribution over future trajectories of all interacting agents in the crowd, through a local interaction model that they train using real human trajectory data. An long short term memory (LSTM) model is proposed by [94] for such crowd interactions.

Earlier works on pedestrian model with car interaction can be broadly divided into three categories based on the modeling methods used. They are physics-based methods, pattern-based methods and planning-based methods [97]. Recent advances in modeling of human behavior using neural networks [98, 99], are found to perform well in prediction accuracy. The neural network based techniques are usually referred to as “black-box” models. This is due to the fact that artificial intelligence (AI) developers cannot fully explain some decisions taken by the neural network. This makes them particularly problematic during cause investigations of incidents involving autonomous vehicles and their liability determination [100]. This motivates us to use a subset of physics based methods. Physics-based models are further divided into single mode and multi-mode models [97]. Since it is known that the multi-mode model can represent the decision making by focusing on the mode switching condition, the multi-mode model is considered as a behavior model in this paper. The models proposed in our own previous works [95, 96] and [101–103], employed probabilistic functions to predict the assorted paths that the pedestrian might choose. This was achieved with the use of interacting multiple-model kalman filters (IMM-KF). While [95] and [96] use IMM-KF for human trajectory prediction, [101] highlights the advantages of multiple models in a kalman filter for prediction. A recursive bayesian filter based approach to pedestrian path prediction at short time horizons is proposed by [102]. A combined intention recognition and path prediction is proposed by [103] using an IMM filter. Even though the decision making in terms of modes and their probabilities were explicitly accessible, these models did not consider the pedestrian-vehicle behavioral interaction. Another multi-mode model method that has interaction [89] uses two simple models of stop and go, which is insufficient for an accurate representation of pedestrian dynamics.

Based on these considerations, a multi-mode probability weighted ARX (PrARX) model [6] is adopted for pedestrian crossing behavior modeling under the influence of a car in this paper. Real-world vehicle-pedestrian interaction data is used to identify this model [104]. The identified PrARX model gives explicit access to pedestrian decision making in terms of modes and their probabilities. Each of the modes considered also includes the pedestrian-vehicle interaction in them. With this model, it becomes possible to calculate the predicted mode probabilities and entropy within long prediction horizons at a small computational cost.

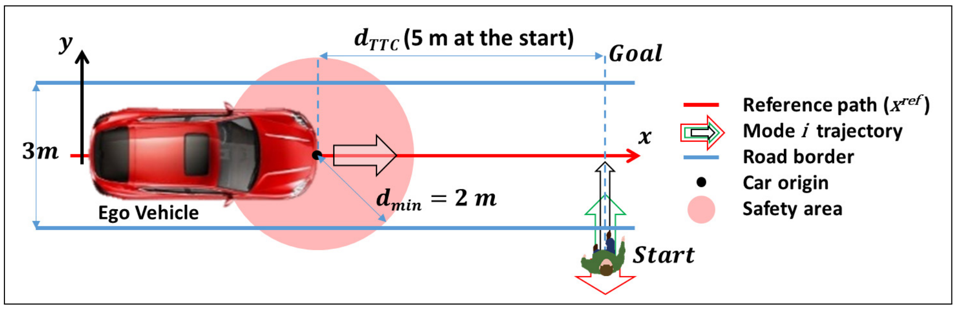


Fig. 4.2 Target use case that contains one vehicle and one pedestrian.

4.2.3 Controller design

An MPC that can consider probabilistic scenarios is necessary to incorporate with the proposed pedestrian model. Due to the inability to address probabilistic elements, a conventional MPC considers the worst case scenario and hence acts conservatively. MPC with probabilistic scenarios can perform in a less conservative way without compromising the safety of pedestrians. To the best of our understanding, there has not been enough number of significant works addressing shared road driving with pedestrians using MPC containing probabilistic elements. Our own previous works [95] and [96] have tried to address such a problem using an IMM-KF model for pedestrian behavior. The work [96] employed a random sample based optimization [69] with a sampling-based representation of probabilistic constraints. The work [95] proposed an alternate method of optimization using IPOPT optimizer [105] by creating a novel formulation for the probabilistic constraints. Both of these works lacked the pedestrian-vehicle interaction dynamics inside the pedestrian model, hence unaware of the vehicle's influence on pedestrians. In this work, along with the proposed PrARX pedestrian model, a pedestrian-aware MPC framework is proposed. The sampling-based solver and a sampling-based representation of probabilistic constraints are chosen for the MPC.

4.3 Control architecture for considerate driving

The control architecture of the proposed MPC controller is explained in Fig. 4.1. The behavior of the ego vehicle and the traffic participants in its vicinity are predicted by their corresponding models. The interaction between them is also considered since the pedestrian model includes the ego car's state. The variable u represents the control input that is obtained by optimizing the cost function J in real-time. The cost function J contains safety constraints along with decision entropy and reference path and speeds. The prediction models including interaction act as constraints in the optimization. The proposed MPC framework with pedestrian model described by PrARX model is expected to realize a natural consensus between car and pedestrians. To demonstrate this, a shared-road driving situation is considered where a car has to negotiate a crossing pedestrian as shown in Fig. 4.2. The ego car intends to drive straight keeping a reference velocity v_{ref} as close as possible. The road under consideration is narrow (3m wide) which is typical for a suburban road for one-way traffic [106]. The road being narrow, steering action is almost negligible. The pedestrian starts from one side of the road and his/her intention is identified as trying to cross to the other side (identification of this intention is considered as external to this framework, hence not in the scope of this work). Pedestrians crossing path is also assumed to be straight, but his/her motion is influenced by the car. The ego car is capable of estimating pedestrian's crossing dynamics. Since the influence of the ego car on the pedestrian's motion is modeled, the controller tries to make the pedestrian's crossing as easy as possible by minimizing the decision entropy of the pedestrian. Since the pedestrian behavior is modeled with multiple models, the predicted trajectory following each mode is different. They are highlighted as arrows of different colors in Fig. 4.2.

4.4 Modeling of pedestrian and vehicle for state prediction

Pedestrians being the most vulnerable traffic participants, predicting their motion accurately can greatly improve the safety in intelligent transportation. As the pedestrians are most dynamic in nature, their ability to suddenly and randomly switch direction and speed makes their predictions difficult. Such dynamics makes it inaccurate to represent it using a single dynamic model. Instead, a multi-mode approach is used, which identifies and combines several basic models as a PrARX model. This framework has been proven to be useful to obtain correct estimation and prediction of human decision making [6, 83]. This section explains the details of the chosen dynamic models and path prediction using them.

4.4.1 Data for pedestrian model identification

Clean and continuous data which is free of noise is always recommended to be used as an input to the PrARX model identification phase. Vehicle-Crowd Interaction (VCI) - CITR Dataset [104] is used in this work, which is available in open source from the Ohio state university. This data set was collected from controlled experiments executed in a university parking lot. Experiment participants were instructed to cross a road from a particular starting area to a goal area. A manually driven golf cart approaches perpendicular to the crossing participants and create an interaction scenario. A drone at a fixed distance above experiment

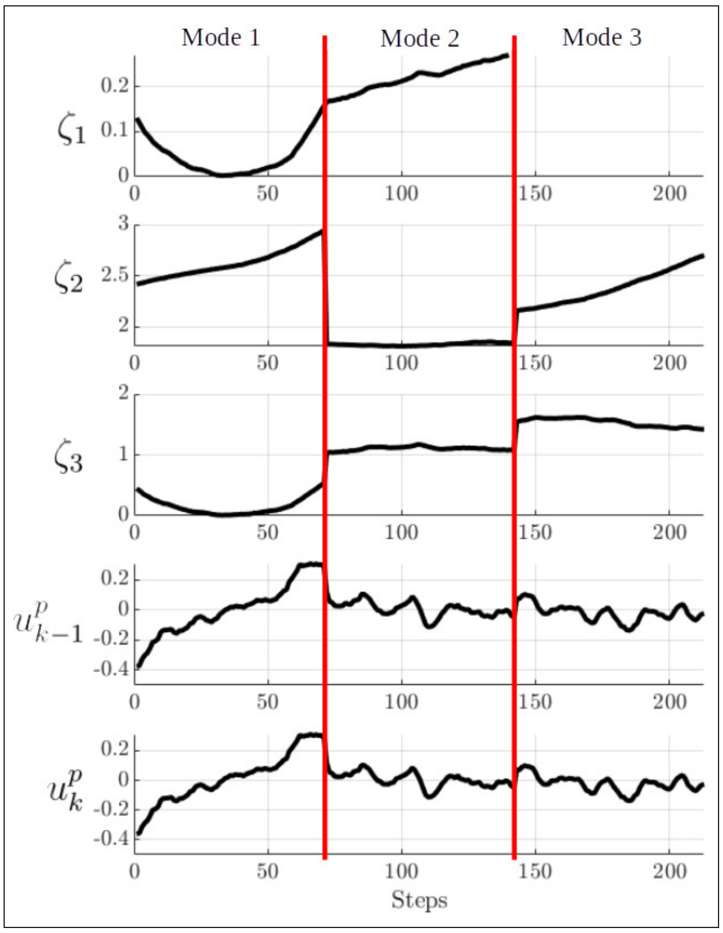


Fig. 4.3 Sample profiles from the data.

area was used to record the interaction data by logging positions of each entity. The details of the experiment, along with videos and filtered data can be found in [104] and their GitHub repository [107]. The data set contains position and velocity data along x and y axis for the pedestrians. For the car, position along x and y axis, longitudinal velocity and heading angle are available. Among the 6 scenarios available in the data set we choose only the ”Lateral interaction (Unidirectional)" data as seen in Fig. 4.2. This contains the data for 32 pedestrians split into 8 scenes that are 20 seconds long.

From this data, the following variables are calculated as input and output:

Input variables[1].

- ζ_1: $1/TTC$ (Inverse of time to collision) [1/s]
- ζ_2: Pedestrian velocity [m/s]

[1]The variable U is also used to represent control inputs in chapter 3 and 5.

- ζ_3: Car velocity [m/s]

Output variable

- u^p: Pedestrian acceleration [m/s^2]

The sample profiles of these data are shown in Fig.4.3. The x axis shows the steps in time, and the y axis shows 1/TTC, pedestrian velocity, vehicle speed, recursive term, and pedestrian model output acceleration from top to bottom. We will explain the vertical red line and the term 'Mode' later in this chapter.

4.4.2 Pedestrian modeling by PrARX model

In this work, a Probability weighted ARX (PrARX) model is used as a pedestrian model, wherein the multiple ARX models are composed by the probabilistic weighting functions. The PrARX model is defined by the form

$$u_k^p = f_{PrARX}(r_k^p) + e_k, \tag{4.1}$$

where $k \geq 0$ denotes the sampling index, $u_k \in \mathbb{R}^q$ is the pedestrian acceleration, e_k is an error term. r_k is a regressor vector containing by input $\zeta_k \in \mathbb{R}^p$ and past outputs.

$$r_k^p = \begin{bmatrix} u_{k-1}^p & \cdots & u_{k-n_a}^p & \zeta_{k-1}{}^\top & \cdots & \zeta_{k-n_b}{}^\top \end{bmatrix}^\top \tag{4.2}$$

where $r_k \in \mathbb{R}^n$, $n = q \cdot n_a + p \cdot n_b$, $\zeta_k = [\zeta_{1k}, \cdots, \zeta_{3k}]^\top \in \mathbb{R}^3$ is the input variable vector with number of inputs $p = 3$. Number of outputs $q = 1$. For simplicity, n_a and n_b are set as 1 for a first order ARX model. $f_{PrARX}(r_k)$ is a function of the form

$$f_{PrARX}(r_k) = \sum_{i=1}^{M} \mu_i \theta_i^T \varphi_k, \tag{4.3}$$

where $\varphi_k = [r_k^T \ 1]^T \in \mathbb{R}^{n+1}$. $\theta_i \in \mathbb{R}^{(n+1)\times q}$ ($i = 1, \cdots, M$) is an unknown parameter matrix of each mode. M is the number of modes and is supposed to be known. μ_i denotes the probability that the corresponding regressor vector r_k belongs to the mode i, and is given by the softmax function as follows:

$$\mu_i = \frac{\exp(\eta_i^T \varphi_k)}{\sum_{j=1}^{M} \exp(\eta_j^T \varphi_k)}, \tag{4.4}$$
$$\eta_M = 0,$$

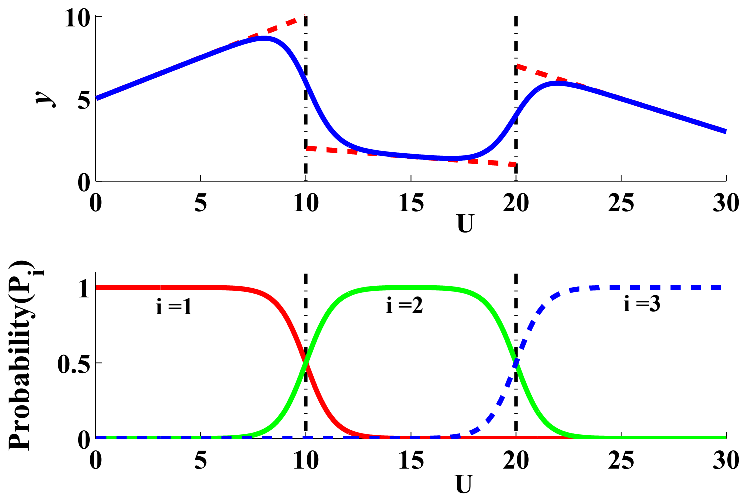


Fig. 4.4 Sample model of the single output PrARX model with 3 modes (reproduction of fig.1 in [6]).

where η_i ($i = 1, \cdots, M-1$) is an unknown parameter that characterizes the probabilistic partition between regions corresponding to each mode.

The sample model is shown in Fig.4.4. This model is the single output PrARX model with three modes (the red lines associated with the outputs shows the linear model). The model parameters are given by

$$\begin{aligned} \theta_1 = [0.5 \quad -5]^T, \quad \theta_2 = [-0.1 \quad 3]^T, \\ \theta_3 = [-0.4 \quad 15]^T, \\ \eta_1 = [-3 \quad 45]^T, \quad \eta_2 = [-1.5 \quad 30]^T, \\ \eta_3 = [0 \quad 0]^T. \end{aligned} \tag{4.5}$$

It can be seen that the three ARX models are smoothly connected at $U = 10$ and 20. These connecting points, i.e., the partitions can be calculated from the η_1 and η_2. Since the partitions defined by η is the switching point of pedestrian's dynamic model, the parameter η in given PrARX model characterizes the decision making in the pedestrian. (See [6] for details). In order to identify the parameters in the given PrARX model, the steepest descent method is used. The cost function is de fined as the square norm of the output. Details of the identification scheme can be found in author's previous work [6]. The pedestrian is assumed to move straight parallel to the y axis shown in Fig. 4.2. Hence, the pedestrian state is updated using a point mass model. Once the pedestrian acceleration y_k is calculated using

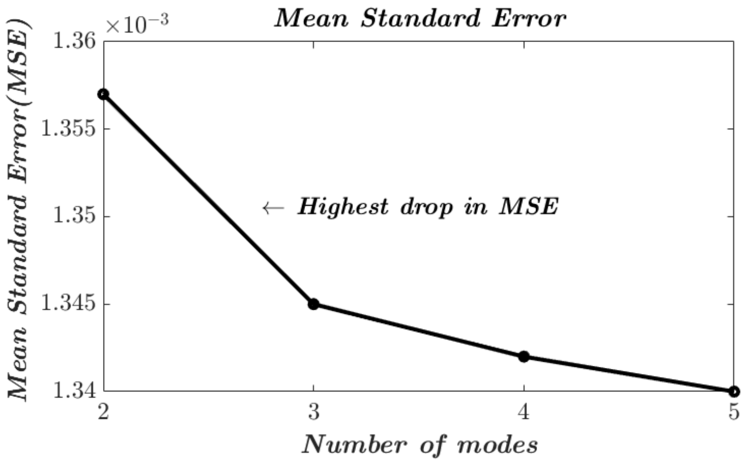


Fig. 4.5 Mean standard errors at various mode numbers

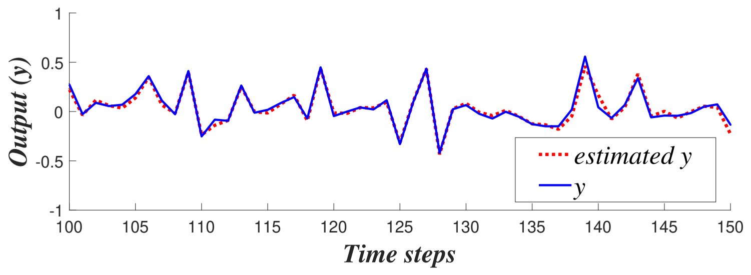


Fig. 4.6 Fit of PrARX model on data

PrARX model (4.1), the pedestrian state is updated as follows:

$$\xi^p_{k+1} = A\xi^p_k + Bu^p_k,$$
$$A = \begin{bmatrix} 1 & \Delta_T \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \Delta_T^2/2 \\ \Delta_T \end{bmatrix} \tag{4.6}$$

$\xi^p_k = \begin{bmatrix} y^p_k & v^p_k \end{bmatrix}^\top$ represents the state of the pedestrian, index p refers to pedestrian. y^p_k, v^p_k and u^p_k represent the position, the velocity and the acceleration of the pedestrian along y-axis, at step k respectively. The time interval corresponding to one control step is represented by T. The pedestrian is assumed to move along y-axis, hence the pedestrian position along the x-axis (x^p_k), is considered to be constant.

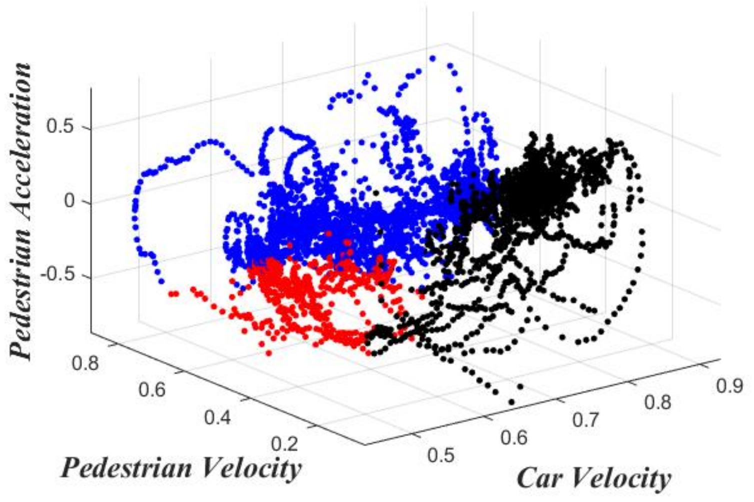


Fig. 4.7 Segmentation of the data in 3D

Table 4.1 Identified parameters

	θ				
Mode (i)	θ_{i1}	θ_{i2}	θ_{i3}	θ_{i4}	θ_{i5}
1	$-4.82e-4$	0.03	-0.05	0.97	$1.61e-4$
2	0.02	-0.15	-0.08	0.95	-0.06
3	0.02	-0.02	-0.02	0.95	0.18
	η				
Mode (i)	η_{i1}	η_{i2}	η_{i3}	η_{i4}	η_{i5}
1	0.07	8.61	-8.37	-0.50	-2.82
2	$3.24e-4$	-0.76	-0.56	-0.42	.59

4.4.3 Verification of proposed model

The data from VCI-CSTR data set as discussed in section 4.4.1 was used to identify the parameters for the PrARX pedestrian model. The given data is randomly divided into two, for training and testing. Generally speaking, it is desirable that the number of modes is as small as possible from the viewpoint of computational complexity. In addition, as seen in Fig. 4.5 the difference of the mean square error between the two-mode model and the three-mode model is much larger than that between the three-mode model and the four-mode model. Therefore, it can be concluded that the optimal number of the modes here is three. In the remaining part of this paper, the number of modes is considered as three. In the previously discussed Fig.4.3, the input and estimated output according to 3 modes are shown, the vertical red lines indicates the mode switching time. The list of identified parameters for the three

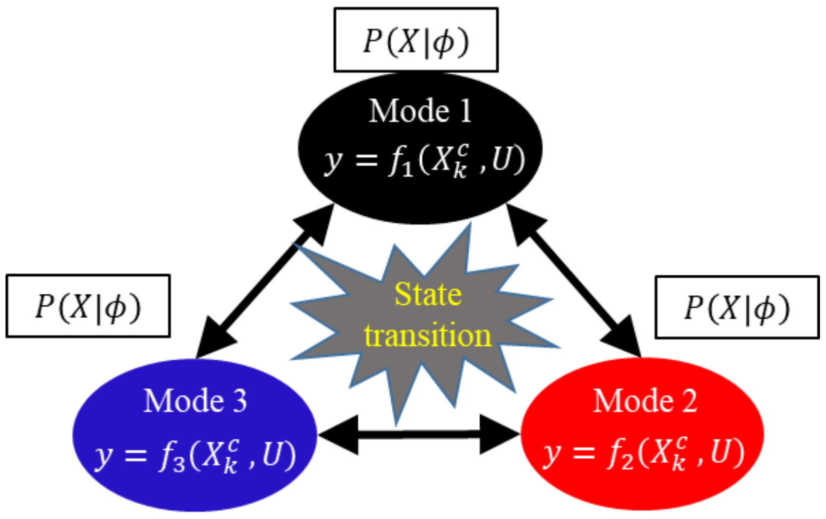


Fig. 4.8 State transition in the model

mode model are listed in Table 4.1. Figure 4.6 shows the fitting of the data regenerated by the model on the original training data. They coincide well with each other, and the mean standard error (MSE) value was 0.0013. The identification process does not contain any intentional scheme to find each mode and separations between modes. The parameters in the PrARX model was automatically found by applying identification scheme. Nevertheless, the data was clearly separated into three clusters as shown in Fig.4.7. The behavioral meaning behind these modes are as explained below:

- BLACK: The black points belong to Mode 1. This mode is active at higher car velocities. This mode represents the pedestrian's yielding behavior to the car (hence low pedestrian velocities).
- RED: The red points belong to Mode 2. This mode is only active at low car velocities. In this mode, Pedestrian decides and starts to cross the road. Pedestrian acceleration is found to be positive in this mode.
- BLUE: The blue points belong to Mode 3. This mode is significant in higher pedestrian velocities. This mode shows pedestrian's steady state crossing behavior, once he/she reaches maximum walking pace. This is also the predominant mode once the pedestrian or the car passes each other.

The transitions between these three modes are illustrated in Fig.4.8. Such transitions specified by η can be interpreted to represent the decision making aspect of the pedestrian. Furthermore, having the probability measure along with every discrete state, the decision entropy can be easily quantified.

4.4.4 Longitudinal vehicle model

Due to the constrained lane width in the shared road scenario, it is noticed that the driver hardly steers the car to miss a pedestrian. Instead, he or she slows down the car when a danger is approaching or speeds up and overtakes a pedestrian before getting too close. As a result, only the longitudinal dynamics of the ego car is taken into account in this work. At the specified speeds, the longitudinal dynamics of the ego car can be adequately expressed by a point mass model similar to eqn. 4.6.

$$X^c_{k+1} = AX^c_k + Bu^c_k \tag{4.7}$$

$X^c_k = \begin{bmatrix} x^c_k & v^c_k \end{bmatrix}^\top$ represents the vehicle state. x^c_k, v^c_k and u^c_k represent the position, velocity and acceleration of ego car along x-axis, at step k respectively. The time interval corresponding to one control step is represented by T. The car position along y-axis, y^c_k, is considered to be constant.

4.5 Safety constraints

4.5.1 Collision avoidance with pedestrian

In order to prevent the collision between the vehicle and the pedestrian, the distance between them must be kept over a specified safety distance $d_{\min}$. In Section 4.4.3, M predicted trajectories are considered corresponding to M pedestrian model. Note that those trajectories have different occurrence probabilities as mentioned in Section 4.4.3. When estimating the collision risk, it is needed to predict the pedestrian behavior. In the pedestrian model described in Section 4.4.2, probabilistic weighting of multiple modes We used the PrARX model to express pedestrian behavior. When considering the collision risk, it is not possible to predict the average behavior of each mode hence certain dangerous behavior can be missed. On the other hand, we could use a Markov process for mode transition. In such a case, verifying all transition possibilities of a mode sequence will be difficult to implement in real-time control from the point of view of computational complexity. Therefore, in this study, the prediction horizon in model predictive control is kept short and the mode of choice is assumed to be constant in this interval. Prediction is limited to M scenarios, which is the number of modes in the pedestrian model. Then, the risk is estimated by using the maximum occurrence probability of the relevant mode at the step where the collision is most probable. This is predicted as the representative value of the probability of occurrence of the scenario.

The chance of collision at time t, $\mathrm{P}_\mathrm{t}^\mathrm{col}$, can be estimated as follows:

$$\begin{aligned} \mathrm{P}_t^{\mathrm{col}} &= \sum_{m=1}^{M} D_t^m, \\ D_t^m &= \mu_{k_{max}|t}^m C_{k_{max}|t}^m, \\ k_{max} &= \underset{k\in(0,N-1)}{\arg\max}\ \mu_{k|t}^m C_{k|t}^m, \end{aligned} \tag{4.8}$$

where $\mu_{k|t}^m$ and $C_{k|t}^m$ are the mode probability and collision indicator at time t, corresponding to prediction step k and the pedestrian mode m. The bar subscript $*_{k|t}$ indicates information about the k-th prediction step of the prediction horizon at time t. The collision indicator can receive the value of 0 or 1 as follows:

$$C_{k|t}^m = \begin{cases} 1 & \text{if } d_{k|t}^m - d_{\min} \leq 0 \\ 0 & \text{if } d_{k|t}^m - d_{\min} > 0 \end{cases}, \tag{4.9}$$

where $d_{k|t}^m$ is the distance between the pedestrian and the vehicle at given time t corresponding to prediction step k and the pedestrian mode m. The car's state at time t is predicted for N future steps assuming the car follows a given input sequence. Pedestrian states corresponding to the car's action are also predicted for N future steps, with M such states of the pedestrian corresponding to the M modes. The choice of safety distance $d_{\min}$ is discussed in Section 4.8.

Remark 1 *Even if the collision indicator returns a value of 1 for a particular pedestrian mode m, it does not imply that the pedestrian and the vehicle actually collide. Additionally, the mode probability, which is derived from the PrARX, must be considered.*

In order to prevent the vehicle from colliding with the pedestrian, the safety constraint is imposed on the chance of collision (4.8) as below:

$$\mathrm{P}_t^{\mathrm{col}} \leq r, \tag{4.10}$$

where $r \in (0,1)$ specifies an allowed risk threshold. Generally speaking, large r might lead to a high chance of collision. Small r, however, might lead to a conservative behavior of the vehicle. The effect of the parameter r on the behavior of the vehicle is investigated in Section 4.8.

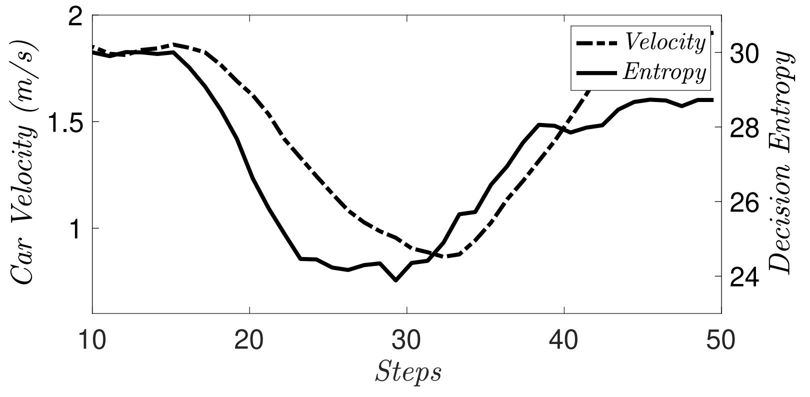
4.5.2 Velocity and acceleration constraints

The velocity and acceleration constraints of the vehicle are specified as follows:.

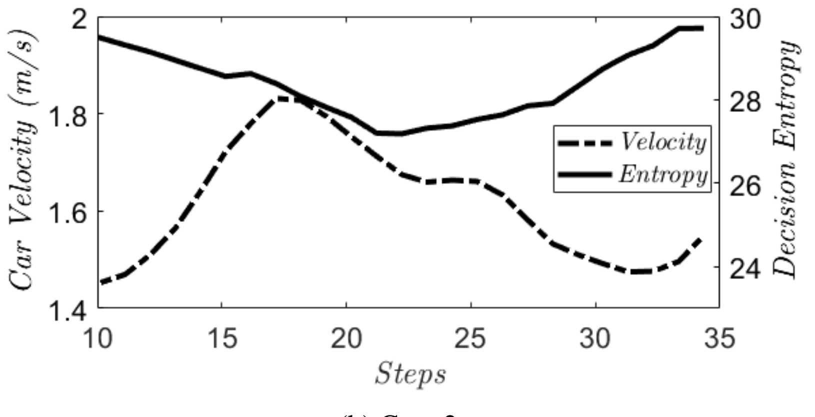
$$v_{\min} \leq v_k^c \leq v_{\max}, \qquad a_{\min} \leq u_k^c \leq a_{\max}. \tag{4.11}$$

Here, $v_{\max}$ is set at $8m/s$ to match with the speed limits on the roads where such interactions were observed, $v_{\min}$ is set as $0m/s$ to avoid driving in reverse. Limits on acceleration $a_{\min}$ and $a_{\max}$ are set at $3m/s^2$ and $-3m/s^2$, these numbers are typical for passenger cars in such scenarios [108].

4.6 Decision Entropy



(a) Case 1



(b) Case 2

Fig. 4.9 Effect of the car on pedestrians decision entropy

This work uses a quantitative index, “decision entropy” to evaluate the vagueness in pedestrian’s decision. Decision entropy was previously proposed to successfully represent vagueness in human decision making [83]. Since the decision entropy is calculated from probabilities of the available modes that an agent can choose, the usage of PrARX model facilitates easy definition of this decision entropy. This is due to the fact that the PrARX model always provide the probability of the pedestrian to choose each one of the defined modes. Decision entropy is defined as follows:

$$H(r_k) = \sum_{i=1}^{M} \mu_i \log(\mu_i), \tag{4.12}$$

where $H(r_k)$ is the decision entropy of the pedestrian at sample index k corresponding to the regressor vector r_k. M is the number of modes and μ_i denotes the probability that the corresponding regressor vector r_k belongs to the mode i (eq.(4.4)). The larger the decision entropy is, the more the vagueness in the decision making is. This entropy is added as a new cost into the MPC cost function as explained in section 4.7.1. With the help of PrARX model, the MPC controller is able to find the action of the ego vehicle, which minimizes the pedestrian’s entropy levels, and thereby, is able to provide a stressless interaction between car and pedestrian. Figure 4.9 shows the pedestrian’s entropy levels at two common interaction scenarios. Fig. 4.9a shows an example scenario in which a car slows down to give priority to pedestrians. In this case, we can see that the pedestrian’s decision entropy decreases from around Step 15 as the vehicle approaches, but there is a steeper drop in entropy as the vehicle decelerates significantly. Similarly, Fig. 4.9b shows a situation in which the car accelerates and passes first. Here, by increasing the vehicle speed, the pedestrian’s decision entropy can be reduced. In this case, it can be said that it became easier for pedestrians to make a decision to wait. (In this case, Mode 1, pedestrian is yielding).

4.7 Realization by pedestrian-aware model predictive control

n this section, we are developing a Model Predictive Control (MPC) speed controller for a vehicle that can safely navigate a shared road environment, which includes pedestrians. The key feature of this controller is its ability to handle the uncertainties associated with the pedestrian’s behavior within the prediction horizon of the controller. The MPC minimizes a cost function which includes the decision entropy of the pedestrians, and calculates the optimal control input. The MPC also keeps the collision probability below a specified risk threshold r.

4.7.1 Problem formulation

To address this challenge, we employ models for the vehicle, pedestrian, and safety constraints to frame the receding horizon control problem. At each time step, we tackle a finite time horizon optimal control problem, seeking to generate a sequence of control inputs that minimize a predefined cost function. The first control input from this sequence is then implemented on the vehicle. This process is reiterated with updated measurement data in the subsequent time step. Below, we provide an explanation of how the optimization problem is formulated at each time step.

given: $X_t^c,\ \xi_t^p,\ v_{\text{ref}}\ \mathit{and}\ \ \mu_t^m,$

find: $u_{k|t},\quad k=0,\dots,N-1.$

Which minimizes:

$$J_{\text{cost}} = \Phi(X_{N|t}^c, v_{\text{ref}}) + \sum_{k=0}^{N-1}\sum_{m=1}^{M} \mu_{k|t}^m \mathscr{L}^m(X_{k|t}^c, u_{k|t}). \tag{4.13a}$$

$$\Phi(X_{N|t}^c, v_{\text{ref}}) = S_c(v_{\text{ref}}^c - v_{N|t}^c)^2, \tag{4.13b}$$

$$\begin{aligned}\mathscr{L}^m(X_{k|t}^c, u_{k|t}, \xi_{k|t}^p) = {} & Q_c(v_{\text{ref}}^c - v_{k|t}^c)^2 + R_c(u_{k+1|t} - u_{k|t})^2 \\ & + E_c(\mu_{k|t}^m \log(\mu_{k|t}^m))^2 + \mathscr{B}^m(X_{k|t}^c, \xi_{k|t}^p),\end{aligned} \tag{4.13c}$$

$$\mathscr{B}^m(X_{k|t}^c, \xi_{k|t}^p) = P_c \exp\left(-\alpha\left(d_{k|t}^m - d_{\min}\right)\right). \tag{4.13d}$$

Subject to:

$$X_{k+1|t}^c = FX_{k|t}^c + Gu_{k|t},\ \ k=0,\dots,N-1\ , \tag{4.13e}$$

$$\xi_{k+1|t}^p = F\xi_{k|t}^c + Gy_{k|t},\ \ k=0,\dots,N-1\ , \tag{4.13f}$$

Safety constraint for collision avoidance

$$\mathrm{P}_t^{\text{col}} \le r \quad (refer\ (4.8)\ in\ 4.5.1), \tag{4.13g}$$

$$v_{\min} \le v_k^c \le v_{\max},\quad a_{\min} \le u_k \le a_{\max}, \tag{4.13h}$$

$$X_{0|t}^c = X_t^c,\quad \xi_{0|t}^p = \xi_t^p, \tag{4.13i}$$

where $N=20$ shows the step count in the prediction horizon, $T=0.1$ (s) gives the time step of each interval, $d_{\min}$ represents the safety distance. $\alpha=10$ is defined as a constant, S_c, Q_c, R_c, P_c and E_c are parameters of weight, and d_k^m is the distance between the pedestrian and the vehicle at time k given that the pedestrian is following the Mode m. The bar subscript $*_{k|t}$ indicates a variable related to the k-th prediction step of the prediction horizon at time t, as described in Section 4.5. Note that each individual pedestrian's predicted trajectory owns its unique probability μ_t^m calculated from the PrARX model. Hence, if the constraint of distance between pedestrian and vehicle is violated as per Mode m whose probability μ_t^m is very small, this implies that the probability of collision between pedestrian and vehicle is also negligible.

4.7.2 Semi-global optimal solution using a randomized approach

The predicted motion of the pedestrian is a result of the weighted composition of the output of multiple modes. The problem of optimization in use should also consider such pedestrian modes directly. Having authored some previous works about developing real-time sample-based optimization using GPU [75], such sample-based optimization method was exploited for this problem. The past work [56] has demonstrated that a sample based method can provide accurate solutions for nonlinear problems without needing a linearization or approximations. Also, it is proven by [47] that a sample count more than a certain number guarantees provided solution to be sufficiently near to the global optimum. The detailed steps involved in this optimization process are as follows.

Algorithm 2 Sample-based optimization

```
1: Generate $N_s$ number of of control input series.
2: for $Series = 1,2,\dots,N_s$ do
3:     for $Modes = 1,2,\dots,M$ do
4:         Calculate the predicted positions of car and pedestrian based on the control input
           series and $M$ possible pedestrian modes.
5:         Calculate mode probabilities and entropy.
6:         Calculate the cost (4.13a) and and the chance-constraint (4.13g) corresponding
           to each series.
7:     end for
8: end for
9: Filter and remove the samples having a probability of collision higher than $r$.
10: Find the minimum cost control input series. The first element of this is passed to the
    system.
```

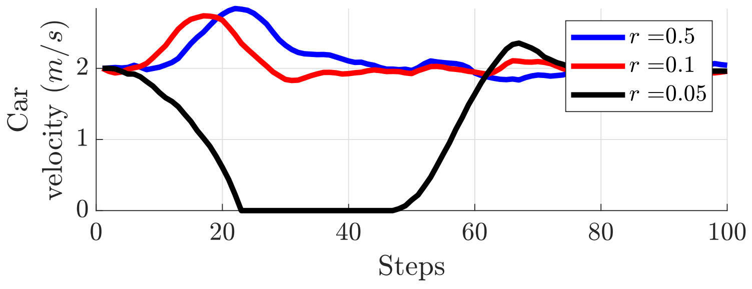
The first step is that of generating N_s number of control input series u^i_{IDCT} following (3.24-3.29) in section 4.7.3. It is also to be confirmed that such series do not cross the physical maximum acceleration bounds of given car. Then the extended state matrices are computed for each of the series according to the vehicle dynamic model (4.7). The pedestrian path is also predicted using the pedestrian's model (4.6). This step is performed M times for the M number of modes that may be followed by the pedestrian within the horizon. After, the cost function (4.13a) is computed for each individual control input series corresponding to all given models. In order to get the optimum sample, there is a step of filtering the samples that have a higher probability of collision than r. The first element of the control input series with minimum cost is the control input to be chosen. This is implemented to the controlled car. These steps are performed repeatedly in every control step (Algorithm 2).

4.7.3 Sampling of random input in the frequency domain

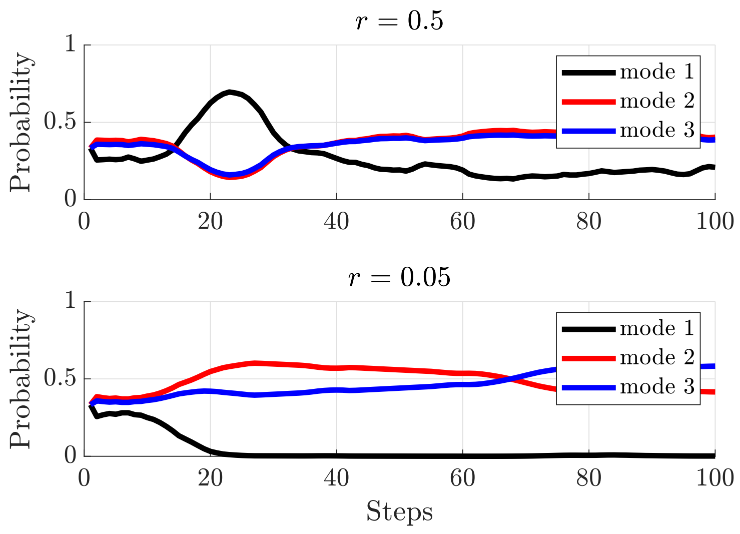
Random number based sample generation cannot ensure that the samples produced are smooth enough to be preferred for driving a car. The samples that are being used here were first created in the frequency domain and then transformed into the time domain. Inverse Discrete Cosine Transform is used for this (IDCT). This method of generation produces samples that are smoother, and consequently, driving performance is smoother. The author's earlier works contain the benefits from the viewpoint of control performance [56]. The details of generation of input series u^i_{IDCT} using IDCT transform are explained in detail in section 3.3. In this work, the smoothing factor $F_{c/o}$ is assumed to be 10.

4.8 Simulation results

MATLAB is used to conduct simulations evaluating the effectiveness of the proposed framework. The sample count N_s in Section 4.7.2 is decided to be 200 samples. In the simulations discussed below, the pedestrian is assumed to wait at one side of the given road, 5 m ahead of the vehicle. Pedestrian has an intention to cross the road. The target speed of the vehicle is set to be 2 m/s (7.2 km/h) unless mentioned otherwise, which is typical for a narrow residential street under consideration. The pedestrian starting point is kept 3.5 m away from the car center along the y-axis. This has been chosen to create a transition in interaction behavior as seen in Fig. 4.10(a) and Fig. 4.12(a). The pedestrian has a chance to change the intention at any time by responding to the car's behavior. As for the pedestrian model, the parameters shown in section 4.4.3 were used for simulations. The same parameters are also used for the pedestrian model within MPC. The closest distance between the car and pedestrian in the training data is observed to be 1 m in the lateral direction and 2 m in the forward direction. Considering that the car is roughly 2 m wide, the origin of the car is set in the front and the safety distance $d_{\min}$ is set as 2 (m) as seen in Fig. 4.2 to maintain these limits.



(a) Car speed profile



(b) PrARX mode probabilities

Fig. 4.10 Interaction at different risk threshold r

4.8.1 Test scenario 1: Different levels of allowed risk threshold

In this test scenario, the risk threshold r is varied and the performance of the controller is observed. The entropy weight is set at $E_c = 3$. In a deterministic case where the uncertainty in prediction models is not considered, the vehicle will either keep a fixed speed expecting the pedestrian to not cross or it will assume the pedestrian to cross and wait indefinitely. Because of the behavior of the pedestrian being highly dynamic, however, the vehicle must anticipate sudden changes in the intention of the pedestrian and act accordingly. This is necessary to keep the collision risk lower than a certain amount. In Fig. 4.10, Fig. 4.10(a) shows the car velocities at different values of r. Fig. 4.10(b) shows the PrARX mode probabilities at two of those cases. In Fig. 4.10(a), It can be seen that the car passes the pedestrian first at a higher risk threshold, but decides to wait for the pedestrian at a lower risk threshold. It can also be seen that the car reacts earlier to the pedestrian's presence as the risk threshold gets lower. At a higher risk threshold r, the vehicle is seen to accelerate. This is a risky choice, but, as seen in Fig. 4.10(b) this influences the pedestrian to choose the yielding mode of motion and wait for the car to pass first. By taking a higher risk, this action enables faster passing. The possibility of changing risk threshold r allows this framework to be personalized to suit the driving styles of different drivers.

While choosing r, it is necessary to consider the upper limit to r after which the car can be found to violate the safety distance. Such values of r are usually very high, as seen in Fig. 4.11 at an entropy level of $E_c = 20$. At $r = 0.9$, there is a safety distance violation. The maximum limit on r should be identified on a case by case basis.

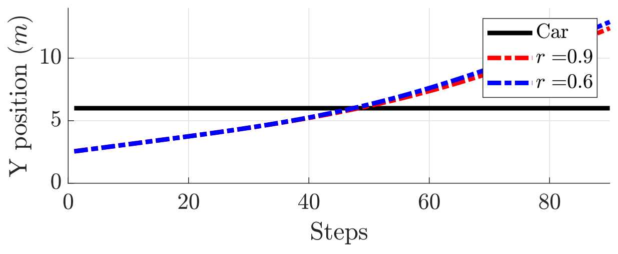
Fig. 4.11 shows the interaction at $E_c = 20$, where $r = 0.9$ and 0.6. The horizontal axis in each figure is the control step index, and the vertical axis is the y coordinates of the pedestrian, the speed of the vehicle, and the distance between the pedestrian and the vehicle, respectively. Looking at Fig. 4.11(a) and Fig. 4.11(b), the pedestrian is passing the y coordinates of the vehicle while the vehicle is almost stopped near Step 47. It can be confirmed that the pedestrian crosses first in either setting. On top of that, when $r = 0.9$, the approach distance is less than d_{min} (even in this case, the safety probability constraint (eqn. 4.10) is satisfied).), higher risk-taking behavior is observed. The setting of the upper limit of r must be determined while considering other performance specifications.

4.8.2 Test scenario 2: Different levels of weight on pedestrian's entropy

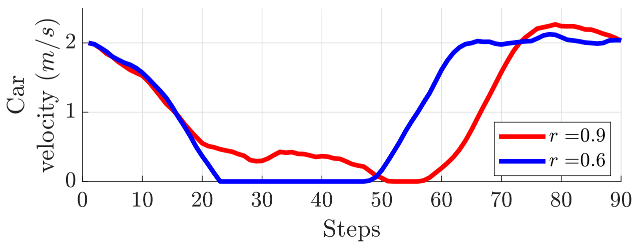
In this section, the risk parameter r is set to be constant as $r = 0.2$. The investigation is on the difference of the performance due to varying cost on pedestrian's decision making entropy. The effect without considering the entropy is realized by simply setting the parameter $E_c = 0$ in (4.13a). The figures Fig. 4.12 and 4.13 corresponds to car velocities of 2 and 2.5 m/s, respectively.

The figures Fig. 4.12(a)-4.12(c) and Fig. 4.13(a)-4.13(c) show the speed of the car, decision entropy and distance between car and pedestrian. In the Fig. 4.12, by assigning a higher weight on the entropy, the car decelerates to a full stop to enable easy decision making of the pedestrian. On the contrary, when there is no cost on entropy, the car accelerates before the pedestrian gets close. This makes the passing faster with minimum deviation from the target velocity. Obviously, the pedestrian goes through a harder decision-making process because of the sudden acceleration of the car.

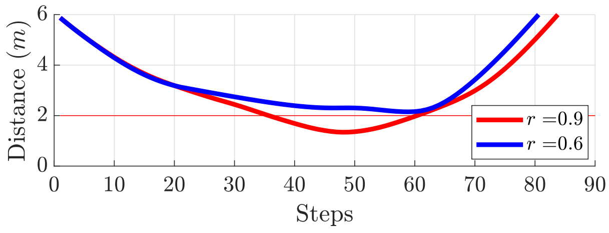
On the contrary in Fig. 4.13, at a higher weight on entropy, the car is found to accelerate and pass the pedestrian first. It might sometimes seem to sacrifice the safety, however, even in this case, a firm consensus is made between car and pedestrian. This is reflected as reduced entropy in Fig. 4.13(c). This behavior is acceptable and has the benefit of stressless interaction.



(a) profile of pedestrian



(b) Car speed profile



(c) Distance between car and pedestrian

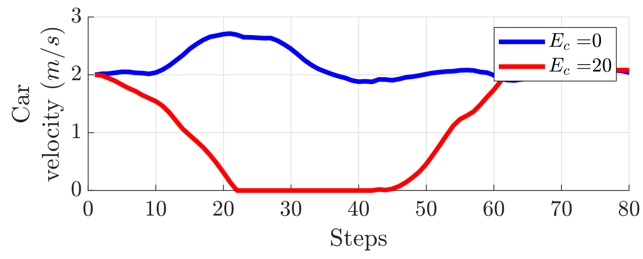
Fig. 4.11 Safety distance violation at high risk threshold ($E_c = 20$)

These comparisons show the considerate driving behavior that was enabled by the addition of entropy cost in this framework. In both cases, the minimum safety distance is not violated, as can be seen in the distance plot in Fig. 4.12(c) and 4.13(c).

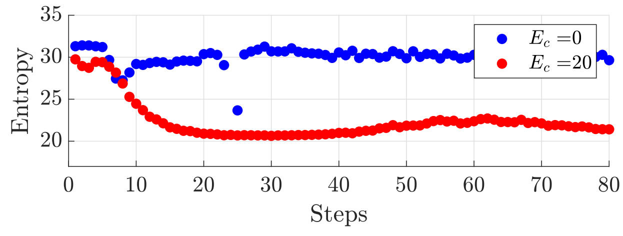
The Fig. 4.12(d), 4.12(e), 4.13(d) and 4.13(e) show car acceleration and the mode probabilities of the pedestrian's dynamic modes in the control duration which is calculated from the PrARX model. As seen in Fig. 4.12(e) and 4.13(e), at a higher weight on entropy, the pedestrian behavior modes are strongly separated, indicating lesser internal confusion. It is also clear in Fig. 4.12(d) and 4.13(d) that the car's acceleration is steeper and clearer at a higher entropy cost. This is expected to clearly convey the decision of the car to the pedestrian early, so that he/she can make clear decisions and prepare for the interaction. This again indicates clear communication of intention, identical to considerate driving by humans.

4.8.3 Comments on computational performance

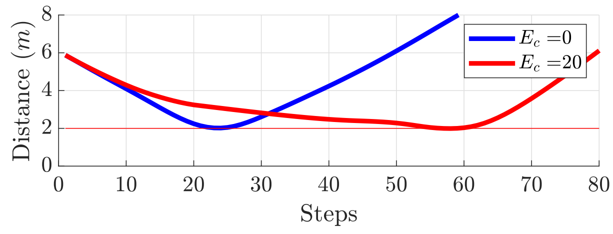
The mean computation time taken by the proposed MPC controller per step is 120 (ms) running in MATLAB. Keep in mind that the proposed controller's structure lends itself very well to parallel processing. Consequently, implementing the proposed controller in GPU, as discussed by [69], can result in a significant reduction in the calculation time.



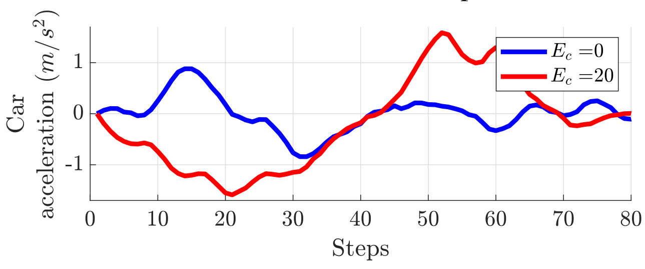
(a) Car speed profile



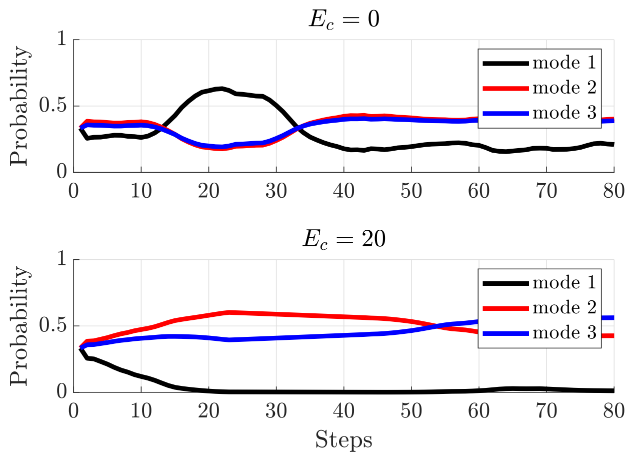
(b) Pedestrian's decision entropy



(c) Distance between car and pedestrian

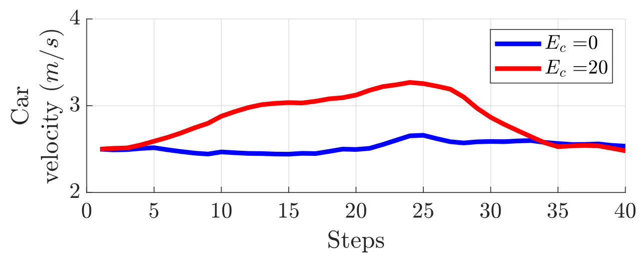


(d) Car acceleration

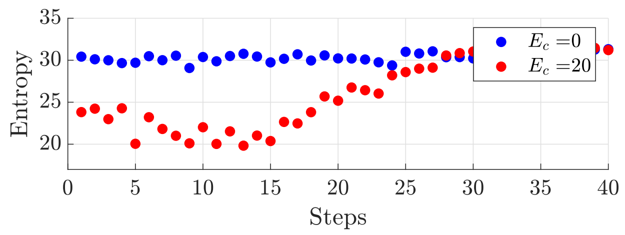


(e) PrARX mode probabilities

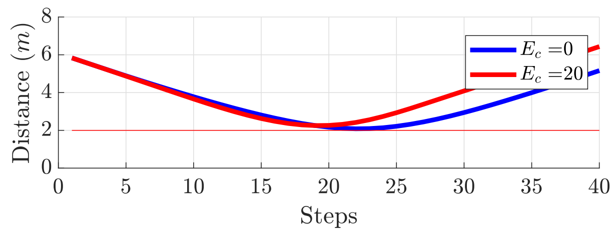
Fig. 4.12 Various plots with and without entropy at Car velocity = $2m/s$



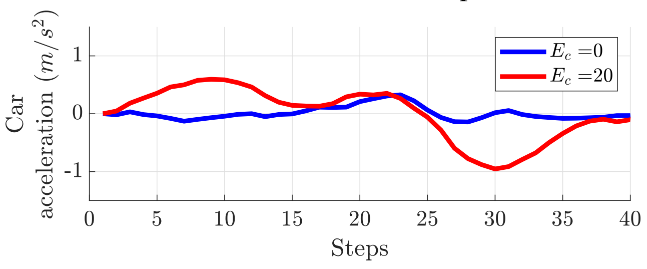
(a) Car speed profile



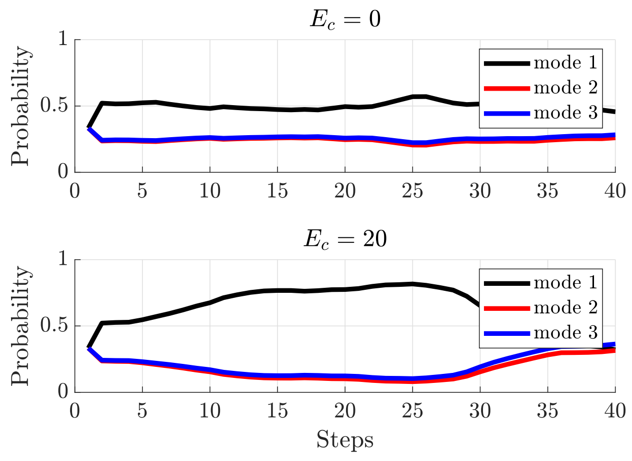
(b) Pedestrian's decision entropy



(c) Distance between car and pedestrian



(d) Car acceleration



(e) PrARX mode probabilities

Fig. 4.13 Various plots with and without entropy at Car velocity = 2.5m/s

4.9 Conclusion

This chapter has presented a randomized nonlinear model predictive controller for autonomous driving that considers the interaction between the car and pedestrians. The pedestrian's motion including the response to the vehicle's actions was identified as a PrARX model from real driving experiment data. The proposed model considers multiple modes that the pedestrian can follow at a given time with their respective probabilities.

The proposed pedestrian model being probabilistic and the MPC being able to accommodate it, this framework has guaranteed safe navigation with a considerate interaction while driving around a pedestrian in a shared road situation. This enables the anticipation of potential collision risk and to react accordingly. The framework that is proposed also allows the use of a tunable risk threshold that allows the car's risk-taking behavior to be personalized for different drivers compared to a fixed response of a conventional deterministic MPC. Considering the pedestrian's decision making entropy as a component of the cost function, this framework achieves quick motion consensus with the pedestrian, followed by considerate and human-like driving.

Despite the nonlinear nature of the vehicle model and the obstacle constraints, the proposed scheme allowed for direct consideration of nonlinear constraints by employing a randomized optimization method. The IDCT method was used to create random samples from the frequency domain for optimization. This avoids unfavorable control input oscillation, which is crucial in autonomous driving.

The proposed scheme and improvements were confirmed in simulations using MATLAB. The simulations demonstrate the effect of tunable risk threshold and weight on entropy making the driving considerate and arguably, more human-like.

This work will be extended to contain and consider multiple agents such as cyclists and pedestrians in the same framework without ignoring their interaction with one another.

Chapter 5

Combination of RMPC with Gradient Based MPC for Fail-Safe Driving at Sudden Changes in Driving Scenario

5.1 Introduction

Autonomous driving (AD) research has long been complicated by dynamic situations with frequent changes in the driving scenario.For handling the unexpected changes, most of the controllers depend on traditional control methods that consist of speed control based on PID , emergency stop and simple steering control [109, 110]. Many of these techniques are restricted to single input single output (SISO) systems due to their inability to deal with constraints. Model Predictive Control (MPC) is one of the popular control methods for multiple input multiple output (MIMO) systems that can handle a variety of state and control input limitations [20, 21, 23, 111]. As a result, MPC has a significant potential for approaching such challenges and better reflecting real-world settings. While [20] and [21] presents the basic principles behind MPC, [23] provides a survey on the theory and practice of MPC. A comparison between MPC and PID is presented in [111].

Various kinds of MPC and MPC calculation algorithms contain their own set of benefits and drawbacks. The concept of such a blending is intriguing since it allows each individual to optimize their strengths while limiting their weaknesses. The paper [112] gives an idea about the combination of different MPC methods,it also explains the advantages of merging two explicit and online MPC approaches to get beyond their respective constraints on online computation time and storage space.shifting between various cost functionals in order to enhance performance has addressed in various research studies in the context of MPC

[113, 114]. The paper [115] proposes softly switched MPC (SS-MPC) as a method for creating a smooth transition in between the two MPCs by creating several intermediate MPCs. Hard switching (a switch without any intermediary switching process) on the other hand, is likely to result in a large overshoot of state or output and a sudden change of control input. One drawback is that soft switching takes so long to fully engage the new controller, making it unsuitable for scenarios that need quick switching. Apart from that, a lot of earlier studies have focused on merging MPC with another controller, such as PID and feedback linearization [116, 117] to improve performance .Meanwhile, as far as the authors are aware, combining optimization approaches to address MPC problems is a relatively new concept, with few academic studies on the subject to date.

The adaptive cruise control (ACC) problem is used in this study to demonstrate how a combination of MPC optimization methods surpasses each individual method when there is an unexpected change in the driving situation. Since major part of our daily driving is spent following a leading car, ACC is expected to be the basic component of self driving systems of level 1 to level 5. The use of an ACC system improves the flow of traffic while ensuring safe and reliable highway driving [118, 119]. Even if basic ACC is comparatively easier control problem to solve, it has been highlighted that providing safe and optimal control under sudden changes such as a cut-in is challenging and often computationally infeasible for real-time solutions [119]. The ACC problem is treated as a nonlinear MPC problem in this study, and the controller takes into account limitations that cause nonlinearity in the system. The C/GMRES algorithm has been used in the optimization phase of the non-linear MPC. C/GMRES works effectively with high computation speed and generates smooth solution [42].On the other hand, it has the drawbacks of a considerable divergence from ideal circumstances and a high likelihood of constraint infringement when the system is suddenly changed. This is especially crucial in ACC situations where the most common cut-in disturbance is present. Randomized MPC is another prominent method of nonlinear MPC optimization (RMPC). While C/GMRES records the local optimal solution, RMPC uses input series sampling to generate a semi-global optimal solution, making it resistant to rapid changes in driving conditions. Although a bigger sample size enhances performance, it is limited by calculation time [75]. Another difficulty with this method is the random noise that appears in its solution. In RMPC-based solutions, the Inverse Discrete Cosine Transform (IDCT) [56] is used to minimise noise.The resulting smoothness, on the other hand, cannot be compared to C/GMRES.

This work makes a significant addition by introducing a novel methodology that combines two optimization approaches, C/GMRES and RMPC, in a way that they compliment each other, producing a superior overall controller for the task in hand. As a result, the suggested controller is named a combined MPC. Providing a solution that fluctuates smoothly over time, C /GMRES is used as the principal optimizer, with RMPC that supports during rapid changes in driving scenarios. A suitable switching mechanism is included in the suggested combination to provide a smooth transition between the two. The switching method also enable to switch in time before there is a failure in the main optimization method. It's also important to know the correct time to go back to the main optimizer (C/GMRES). By evaluating the results given by both methods, the proposed controller address this.

The following chapters of this paper are arranges as follows. The driving environment, state space equation for vehicle dynamics, and formulation of the finite-horizon optimization issue are all demonstrated in section 5.2. It is followed by the calculation by individual optimization methods in the section 5.3. The suggested controller's algorithm is explained in Section 5.4, which also includes switching conditions between the two optimization methods. The simulation results are then presented in the next section 5.5, demonstrating the performance gain achieved by combining C/GMRES with RMPC. The final chapter is the conclusion.

5.2 Problem setting

5.2.1 Task description

The control task in this study is the aggressive cut-in while driving on the highway. Because of the abrupt change in the status of the system during the cut-in, this task is absolutely intriguing. The target driving environment is portrayed in Figure 4.1.The following are the details of the cut-in situation discussed in this study. According to the ACC car following model, a controller-controlled ego car follows a leading car travelling at a steady speed of 80 km/h. The "two-second rule" [120] allows the ego car to keep a safe distance from the leading car. In other words, the relative distance between the two cars is equal to the ego car's velocity v[m/s] multiplied by 2[s], which is 44.4[m] in this situation. At t = 15[s], a third car cuts in and takes over as the new leading car, resulting in a 25[m] reduction in relative distance.The third car is travelling at a steady velocity of 55 [km/h], which is significantly slower than the velocity of the first car. The change in velocity of the leading vehicle during the rapid cut-in is depicted in Fig. 5.2.

The controller should be controlling the ego car in such a way that it maintains the same pace as the leading vehicle while maintaining a proper safe distance. When dealing with curves or lane changes, it is expected that the drivers will take the correct steering actions. As a result, only the vehicles' longitudinal movement is taken into account. The car

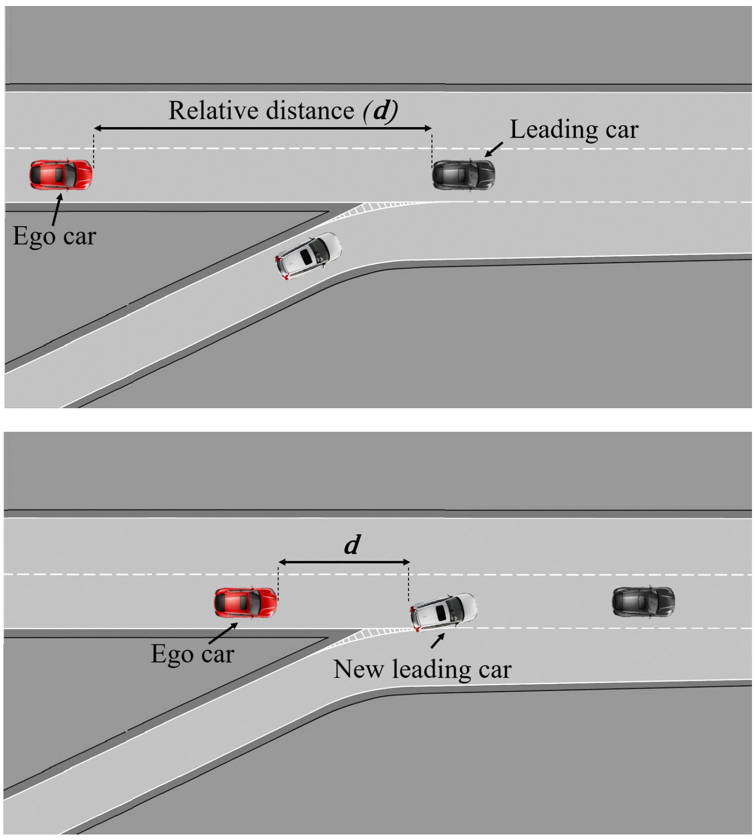


Fig. 5.1 Target environment

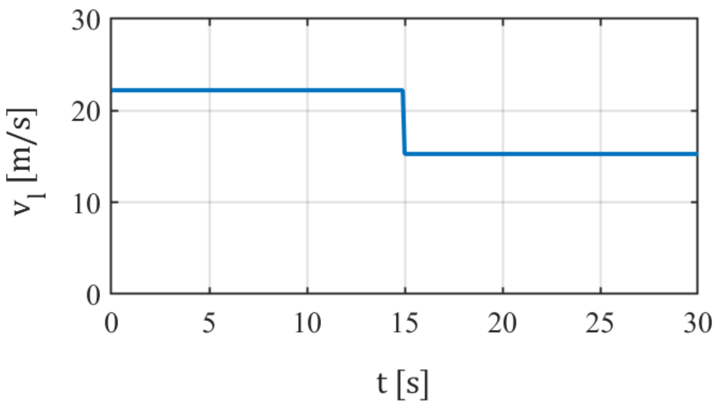


Fig. 5.2 Lead vehicle velocity during the sudden cut-in

dynamics are expressed in this study using a point-mass model.In order to assure the safety and acceptability of control input, this model utilizes some inequality constraints (see section refsec:constraints).

5.2.2 Inputs from the controller and car dynamic equations

The denotations like d, v_l, v, and a are the relative distance between the leading car and the ego car, the leading car's velocity, the ego car's velocity, and the ego car's acceleration, respectively. Acceleration is utilised as a control input unless specified:

$$u = a = \dot{v}, \tag{5.1}$$

The state equation and state vector are written as follows throughout this paper:

$$\dot{x} = f(x,u) = [v_l - v, a]^T, \tag{5.2}$$

$$x = [d, v]^T, \tag{5.3}$$

5.2.3 Constraint formulation for input bounding and safe driving

To guarantee that the input and safety are feasible, the following input and safety constraints are applied:

$$\begin{cases} 5 \leq d \leq 250 & [\mathrm{m}] \\ 0 \leq v \leq 25 & [\mathrm{m/s}] \\ -8 \leq a \leq 8 & [\mathrm{m/s^2}] \end{cases} \tag{5.4}$$

Owing to the maximum detection range, 250[m], of a long-range radar for ACC [121], it is believed that the relative distance (d) has an upper bound.

We implement a barrier function in the equation of cost of the C/GMRES and RMPC as a soft constraint to tackle represent constraints (5.4) [122, 123].

Dummy variables as part of the control input are another approach for C/GMRES to impose inequality restrictions. The inequality constraints (5.4) are turned to equality constraints using dummy variables, which C/GMRES can handle with directly [42]:

$$C(x,u) = \begin{bmatrix} (a-\bar{a})^2 + {u_{d1}}^2 + (a_{\max}-\bar{a})^2 \\ (v-\bar{v})^2 + {u_{d2}}^2 + (v_{\max}-\bar{v})^2 \\ (d-\bar{d})^2 + {u_{d3}}^2 + (d_{\max}-\bar{d})^2 \end{bmatrix} = 0, \tag{5.5}$$

Here, u_{d1},u_{d2} and u_{d3} represent dummy variables. For a given $\chi \in \{d,v,a\}$, $\bar{\chi}$ is defined as: $\bar{\chi} = 0.5(\chi_{min} + \chi_{max})$, where χ_{min} and χ_{max} indicates minimum and maximum values for corresponding variables. The variable $C(x,u)$[1] is used to calculate the Hamiltonian as shown in 5.19.

5.2.4 Formulation of the optimal control problem with a finite-horizon

In each control cycle, the finite-horizon optimum control issue for ACC can be written as below:

$$\begin{aligned}
&\textbf{given } x(t), x^{ref}(t) \quad \textbf{find } \{u(k|t)\}_{0:N-1} \\
&\textbf{that minimizes} \\
&J(\{u(k|t)\}_{0:N-1}) = \phi(N|t)^T S_f \phi(N|t) + \sum_{k=0}^{N-1} L(x(k|t),u(k|t))h, \\
&\text{with } \phi(N|t) = x(N|t) - x^{ref}(N|t), \\
&\textbf{subject to} \\
&x(0|t) = x(t), \\
&x(k+1|t) = x(t) + f(x(k|t),u(k|t))h, \ {}^{\forall}k \in \{0,\cdots,N-1\}.
\end{aligned} \tag{5.6}$$

The control input and state at the k-th step of the prediction horizon at time t are denoted by $u(k|t)$ and $x(k|t)$, respectively. $x(k|t)$ contains $d(k|t)$ and $v(k|t)$, $u(k|t)$ contains $a(k|t)$. The count of prediction steps is N, which is a constant and h represents the control interval. The terminal cost (which is connected with weight matrix S_f) is represented by the first term in the cost function J, while the sum of stage costs is the second term represented by L. The details of L can be found in 5.8 and 5.20.

Reference state corresponding to the k-th step is represented by:

$$x^{ref}(k|t) = [d^{ref}(k|t), v^{ref}(k|t)]^T, \tag{5.7}$$

[1]The variable C is also used in 3.5 and 4.9 as a state multiplier and collision indicator, respectively.

The reference velocity $v^{ref}(k|t)$ equivalent to the leading car's velocity , therefore relative velocity will be zero. Because the "two-second rule" [120] governs the recommended safety distance in driving, the reference distance is defined as $d^{ref}(k|t) = \tau v^{ref}(k|t)$, with $\tau = 2[s]$. The first element of the optimal series of input $u(t) = u^*(0|t)$ determines the system's actual control input.

5.3 Solving the optimal control problem with finite horizon using individual methods of optimization

For nonlinear optimum control problems, C/GMRES and RMPC are common optimization approaches. In C/GMRES, inequality constraints are often implemented two distinct ways: by using dummy variables to change inequality constraints to equality constraints, or by using a barrier function as a soft constraint in the cost function. The usage of a barrier function in RMPC is to handle system constraints. The procedure of Inverse Discrete Cosine Transform (IDCT) can be used to minimise noise in RMPC-based solutions.

5.3.1 C/GMRES method with barrier function for constraints

Stage cost in (5.6) is defined by:

$$\begin{aligned} L(x(k|t),u(k|t)) =& \phi(k|t)^T Q\phi(k|t) + u(k|t)^T Ru(k|t) \\ &+ Bar(x(k|t),u(k|t)), \end{aligned} \tag{5.8}$$

with $\phi(k|t) = x(k|t) - x^{ref}(k|t)$, $^{\forall}k \in \{0,\cdots,N-1\}$.
Q and R are weighting matrices. Barrier function for the inequality constraints is given by:

$$Bar(x,u) = B_d(x,u) + B_v(x,u) + B_a(x,u), \tag{5.9}$$

where

$$B_\chi(x,u) = \begin{cases} -\kappa \log[(\chi - \chi_{min})(\chi_{max} - \chi)] \text{ if } \chi_{min} \le \chi \le \chi_{max}, \\ \beta_1 e^{\alpha_1(\chi_{min}-\chi)} + \beta_2 e^{\alpha_2(\chi-\chi_{max})} \text{ otherwise,} \end{cases}$$

Here, $\chi \in \{d,v,a\}$, χ_{min} and χ_{max} are the minimum and the maximum of corresponding variables. κ, α_1, β_1, α_2 and β_2 are constants. We assume same weight for all three components of the barrier function since it provides satisfactory results in simulation. Note that, It is due to nonlinear barrier function, the finite-horizon optimal control issue is nonlinear. These are defined as follows:

$x(k|t) = x_k$ and $u(k|t) = u_k$. The Hamiltonian function using the co-state λ_k is specified as:

$$H(x_k, u_k, \lambda_k) = L(x_k, u_k) + \lambda_k^T f(x_k, u_k) \tag{5.10}$$

where $f(x_k, u_k)$ is the state equation. The necessary conditions for optimality, known as the Karush–Kuhn–Tucker (KKT) conditions, are given by:

$$\begin{aligned} & x_0 = x(t),\ \lambda_N = \Phi(x_N)^T, \\ & \begin{cases} x_{k+1} = x_k + f(x_k, u_k)h, \\ \lambda_k = \lambda_{k+1} + H_x(x_k, u_k, \lambda_{k+1})^T h, \\ H_u(x_k, u_k, \lambda_{k+1}) = 0, \end{cases} \quad {}^{\forall}k \in \{0, \cdots, N-1\} \end{aligned} \tag{5.11}$$

where, Φ represents the terminal condition of the co-state and H_x and H_u represents partial derivatives of the Hamiltonian with respect to x and u, respectively[2]. The vector $U_{cgmres}(t)$ is defined by:

$$U_{cgmres}(t) = [u_0^T(t)\ u_1^T(t) \cdots u_{N-1}^T(t)], \tag{5.12}$$

If $x(t)$ and $U_{cgmres}(t)$ are given, $\{x_k\}_{0:N}$ and $\{\lambda_k\}_{0:N}$ are determined. Thus, (5.11) can be regarded as an equation of U_{cgmres}:

$$F(U_{cgmres}(t), x(t)) = 0, \tag{5.13}$$

where vector $F(U_{cgmres}(t), x(t))$ presents error in optimality condition, given by:

$$F(U_{cgmres}(t), x(t)) := \begin{bmatrix} H_u(x_0, \lambda_1, u_0) \\ \vdots \\ H_u(x_{N-1}, \lambda_N, u_{N-1}) \end{bmatrix} \tag{5.14}$$

Accordingly, $||F|| = 0$ is satisfied if the solution is locally optimal, where $||F||$ is the norm of vector F.

If the initial solution $U_{cgmres}(0)$ that satisfies $F(U_{cgmres}(0), x(0)) = 0$ can be determined, then $U_{cgmres}(t)$ can be traced by the integration of $\dot{U}_{cgmres}(t)$ that satisfies the condition:

$$\dot{F}(U_{cgmres}(t), x(t)) = -\zeta F(U_{cgmres}(t), x(t)) \ \ (\zeta > 0), \tag{5.15}$$

[2]The variable H is also used to represent entropy in 4.12.

where ζ is a positive constant. Equation (5.15) is equivalent to a linear algebraic equation for $\dot{U}_{cgmres}$:

$$F_U \dot{U}_{cgmres} = -\zeta F - F_x \dot{x} \tag{5.16}$$

The linear equation (5.16) effectively solved using the generalized minimal residual method (GMRES) method [124] if F_U a non-singular matrix. $U_{cgmres}(t)$ can be updated by integrating obtained $\dot{U}_{cgmres}$, which is nothing more than the application of continuation method. It should be emphasised that when the problem changes discontinuously, C/GMRES does not promise a local optimal solution.It finds the solution in real time without iteration for convergence to the local optimal, which is employed in the typical MPC scheme, using the continuation approach.To converge to the local optimal solution, specific control periods are required. As a result, abrupt state shifts can lead to significant deviations from optimality requirements.

The normal MPC scheme, on the other hand, improves the solution until the KKT requirements are met in each control cycle. Sudden state shifts are no longer an issue when the KKT requirements are met. However, it takes a lot of time to iterate until the solution converges to an ideal value in each cycle, and it frequently violates the time-limit, or control interval. In AD applications, it ranges from 0.01[s] to 0.1[s]. As a result, C/GMRES with high computing speed is a promising method for NMPC in AD applications, even though it does not provide full optimality as described previously.

5.3.2 C/GMRES method using dummy variables for constraints

Only In this section, the input vector is defined by:

$$u = [a,\ u_{d1},\ u_{d2},\ u_{d3}], \tag{5.17}$$

and define:

$$\begin{aligned} L(x(k|t), u(k|t)) = & \phi(k|t)^T Q \phi(k|t) + u(k|t)^T R u(k|t) \\ & - g_1 u_{d1} - g_2 u_{d2} - g_3 u_{d3}, \end{aligned} \tag{5.18}$$

with $\phi(k|t) = x(k|t) - x^{ref}(k|t)$, $^{\forall}k \in \{0, \cdots, N-1\}$. In order to explain hard constraints, the dummy variables u_{d*} are called slack variables. The slack variables u_{d*} takes negative values under constraint violation, hence subtracted in the cost function.

Because of the nonlinear equality constraint shown by Eq. (5.5), the finite-horizon optimal control problem is nonlinear . g_i $(i = 1,2,3) > 0$ are weighting coefficients, and Q and R are weighting matrices. The Lagrange multiplier associated with the equality constraint is denoted as μ_k, and the costate is denoted as λ_k. The Hamiltonian currently has the following form:

$$H(x_k,u_k,\lambda_k,\mu_k) = L(x_k,u_k) + \lambda_k^T f(x_k,u_k) + \mu_k^T C(x_k,u_k). \tag{5.19}$$

The sum of stage costs is represented by L, as seen in 5.6.

Please refer [42] for the details of the necessary conditions for optimality and the definition of $F(U(t),x(t))$ in the case with equality constraints.

5.3.3 RMPC using the IDCT process

In the framework of MPC, let us define:

$$\begin{aligned} L(x(k|t),u(k|t)) =& \phi(k|t)^T Q\phi(k|t) + \Delta u(k|t)^T R\Delta u(k|t) \\ &+ Bar(x(k|t),u(k|t)), \end{aligned} \tag{5.20}$$

with $\phi(k|t) = x(k|t) - x^{ref}(k|t)$, $^\forall k \in \{0,\cdots,N-1\}$.

Eq. (5.9) the barrier function, which adds nonlinearity to the system. Δu is the time difference between control inputs, which is specified as,

$$\begin{aligned} &\Delta u(0|t) = |u(0|t) - u(t-1)|_1, \\ &\Delta u(k|t) = |u(k|t) - u(k-1|t)|_1 \ ^\forall k \in \{1,\cdots,N-1\}. \end{aligned} \tag{5.21}$$

When the required sample size is considered, the implementation of a sample-based technique has been shown to offer a semi-optimal solution that is close enough to the optimal solution [47]. The production of samples is the first step in the RMPC process. After selecting Ns sampled input series of length N from a specified distribution of random integers, the future trajectory for each of these sampled input series is determined. At this step, some filtering is done to eliminate infeasible samples. This is done after the cost of each sample has been calculated. The sample series with the lowest cost is then identified, and the first element of the matching input series is used as the ego car's input. These steps are carried out again and again for each control cycle.

The Inverse Discrete Cosine Transform (IDCT) is used to smooth out RMPC-based input because randomly generating samples causes significant noise in the control input and smooth control input is preferable in the application of vehicle control [56]. Instead of being directly sampled in the time domain, IDCT samples are created from the frequency domain. The sampled input series using IDCT, $u^i_{IDCT}(t)$ $(i \in \{1,2,\cdots,N_s\})$, are computed as described in section 3.3.

5.4 Solving the optimal control problem with finite horizon using combined MPC

In this part, a controller is introduced that uses barrier functions to handle inequality constraints and integrates RMPC with C/GMRES. In essence, it entails deciding which optimization approach is used to update control input by switching conditions. The ego car receives the acceleration command as control input by the Eq. (5.1).

Algorithm 3 Combination of C/GMRES and RMPC

```
1:  Set global variable $sw$.
2:  $sw \leftarrow 0$.
3:  **for** Every time step **do**
4:      Calculate $u_{cgmres}$ and $||F||$ by section 5.3.1.
5:      $u \leftarrow u_{cgmres}$
6:      **if** $||F|| \geq F_{th}$ **then**
7:          $sw \leftarrow 1$
8:      **end if**
9:      **if** sw=1 **then**
10:         Calculate $u_{rmpc}$ by section 5.3.3.
11:         $u \leftarrow u_{rmpc}$
12:         **if** $|u_{cgmres}|_1 \leq 8$ and $|u_{rmpc} - u_{cgmres}|_1 \leq \eta$
13:        and $||F|| < F_{th}$ **then**
14:             $sw \leftarrow 0$
15:             $u \leftarrow u_{cgmres}$
16:         **end if**
17:     **end if**
18:     Return $u$ as the next control input.
19: **end for**
```

The switching method between C/GMRES and RMPC is described by the Algorithm 3. Here, the threshold value F_{th} and the comparison between $||F||$ is to determine whether or not a change in the state of the system is "sudden". A change is "sudden" in this work, if it causes $||F||$ to be greater than F_{th}. If the leading car steadily increase or reduce its velocity, resulting in $||F||$ substantially more than zero but still less than the threshold value F_{th}, is not considered "sudden". The variable *sw* in the Algorithm 3 stands for switch. While $sw = 0$ indicates that the input is based on C/GMRES, $sw = 1$ indicates that the input is based on RMPC. The variable *sw* is first set to 0. When $||F||$ exceeds F_{th}, the value of *sw* starts to be 1 and the controller shifts from C/GMRES to RMPC (see line 6-8, Algorithm 3). Even if $||F||$ falls below F_{th} in some subsequent control intervals, *sw*, as a global variable, can stay at 1 unchanged and RMPC-based input can still be used throughout this time. Only when the if condition in lines 12 and 13 of the Algorithm 3 is satisfied, namely C/GMRES-based input satisfying the inequality constraint on control input, C/GMRES and RMPC producing solutions that are close to each other, and $||F||$ smaller than the threshold value F_{th}, does the value of *sw* return to 0, indicating that the controller switches from RMPC to the main optimizer C/GMRES. The final condition is that u_{cgmres} does not deviate too far from the optimal value. The difference between C/GMRES-based input and RMPC-based input ($|u_{rmpc} - u_{cgmres}|1 \leq \eta$) offers a smooth transition from RMPC to C/GMRES. This criterion helps to reduce the large spikes in control input that can occur due to hard switching between various optimization methods. Due to line 9, Algorithm 3, C/GRMES input is calculated in every control cycle, but RMPC-based input computation is performed only when $sw = 1$. C/GMRES uses the control input that is given to the ego car in the previous control cycle as initial guess to construct a local optimal solution, with the exception of the first control cycle, where initial guess is set to 0.

The use of a barrier function as a soft constraint in this research analyses the combination of RMPC and C/GMRES. It is also possible to combine RMPC with C/GMRES, which employs dummy variables to explain harsh limitations, but a suitable technique is necessary to update the value of dummy variables in the input vector when RMPC updates the acceleration component during the cut-in.

5.5 Simulation results and related discussion

In this work, the simulations are implemented using C++ and MATLAB on an Intel®Core™i5-7200U CPU.

5.5.1 Individual optimization methods

The results of simulation with C/GMRES is shown in Fig. 5.3, with related parameters shown in Table 5.1. $||F||$, the norm of the vector F, represents the deviation from optimality

Table 5.1 Parameters for simulation (C/GMRES)

Variable	G/GMRES with barrier functions	C/GMRES with dummy variables
N	20	20
h	0.1[s]	0.1[s]
ζ	5	1
Q	diag(10,10)	diag(1,1)
S_f	diag(50,1000)	diag(10,10)
R	50	diag(1,0.001,0.001,0.001)
g_1, g_2, g_3		0.1,0.1,0.1
$\kappa, \alpha_1, \beta_1, \alpha_2, \beta_2$	0.1,0.1,10,0.1,10	

Table 5.2 Parameters for simulation (RMPC)

Variable		Variable	
N	20	Q	diag(10,10)
h	0.1[s]	S_f	diag(10,100)
Ns	500	R	10
γ	1	$\kappa, \alpha_1, \beta_1, \alpha_2, \beta_2$	0.1,1,100,1,100

conditions. The value of $||F||$ equals 0 when the solution is locally optimal. The lower the value of $||F||$, the closer to optimal the solution is. As seen in the Fig. 5.3, control inputs given by C/GMRES are smooth. Nonetheless, a sudden cut-in at time t = 15[s] causes a significant jump in the $||F||$ value, this means that the solution goes far from optimal. In case of C/GMRES framework, either methods of implementation of the inequality constraints violates the constraint on acceleration ($-8 \leq a \leq 8$ [m/s^2]). It demonstrates that both approaches to handling inequality constraints are ineffective at preventing constraint violation in the face of abrupt state changes. A potential solution to address this concern involves transitioning from C/GMRES to an alternative optimization method that achieves optimal solutions more rapidly, all while maintaining adherence to system constraints. Subsequently, when conditions align favorably, reverting to C/GMRES can facilitate a smooth and efficient switching process.

The simulation results of the RMPC are displayed in Fig. 5.4 with parameter values summarized in Table 5.2. All system constraints are satisfied, as seen in Fig. 5.4. It is because the sample generation step removed the impractical sampled input series. By adjusting the cut-off frequency $F_{c/o}$, the IDCT process makes the RMPC solution smoother, but the random sample generation prevents the noise in the control input from being completely eliminated.

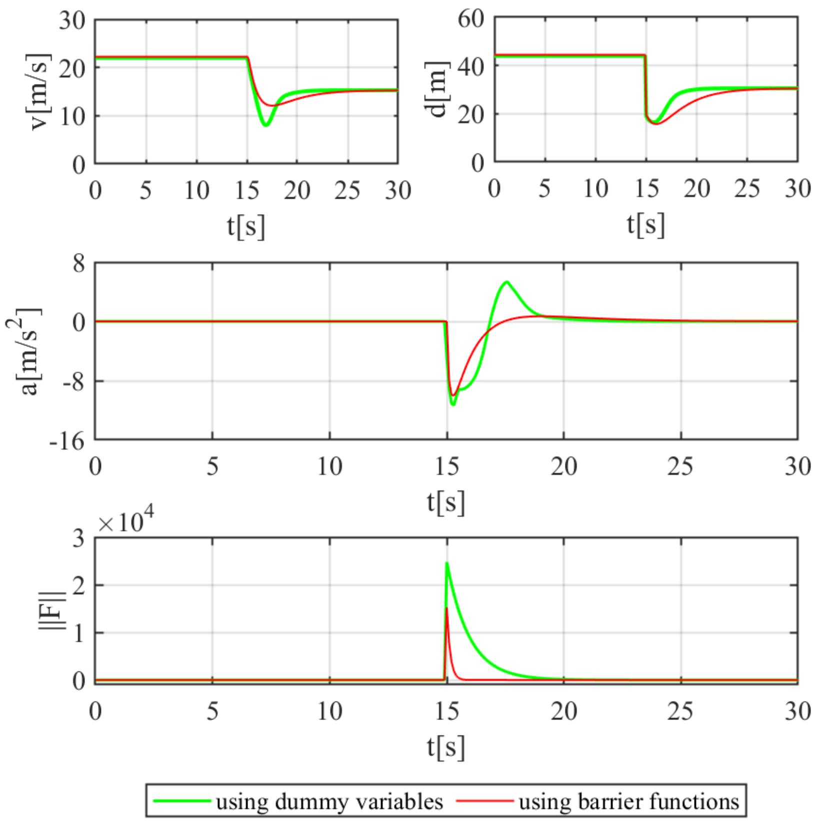


Fig. 5.3 Results of simulation with C/GMRES

5.5.2 The combined MPC

When sudden cut-in occurs at t = 15[s], the controller switches from C/GMRES to RMPC. When preset criteria are met, it switches back from RMPC to C/GMRES.

Table 5.1 and Table 5.2 contain the parameters needed to calculate C/GMRES-based input (u_{cgmres}) and RMPC-based input (u_{rmpc}). Based on the $||F||$ profile of C/GMRES shown in Fig. 5.3, the threshold value F_{th} is set at 1000. The control interval h is set as 0.1[s].

The following explanation describes how to choose a suitable η and the gap limit between RMPC-based input and C/GMRES-based input. The behaviour of u_{rmpc}, u_{cgmres}, as well as their gap in the given time, is depicted in Fig. 5.5 in the scenario when RMPC-based input is continued to be applied after the cut-in, implying that control input does not transfer from RMPC to the main optimizer, C/GMRES. Since they are determined with distinct formulas of stage cost in the cost function, u_{rmpc} and u_{cgmres} may differ. It is feasible to analyse how a spike can be created by switching from RMPC to C/GMRES with respect to a given value of η using Fig. 5.5. The input gap $|u_{rmpc} - u_{cgmres}|_1$ which is recorded once every control cycle has a local minimum of roughly 0.011 [m/s^2] at t = 17.1[s] in this

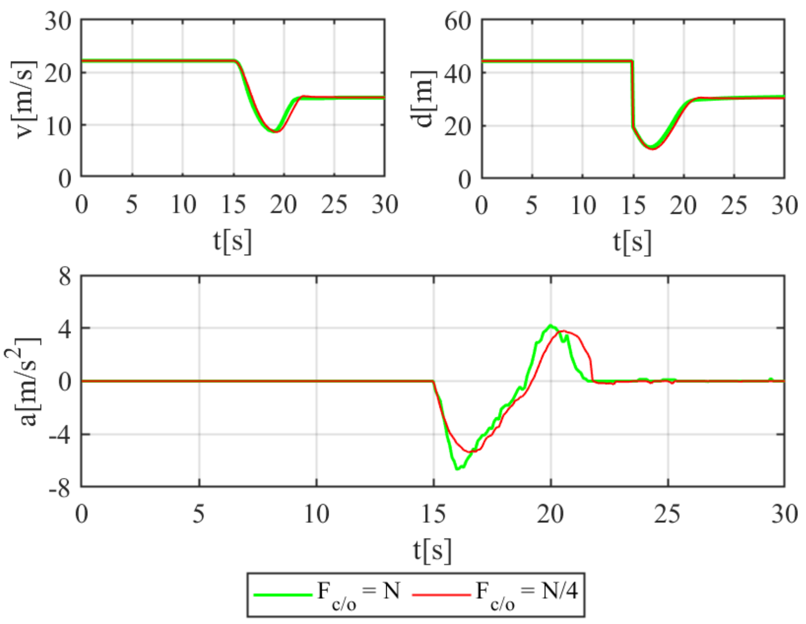


Fig. 5.4 Results of simulation with RMPC

case. To allow for an early transfer from RMPC to C/GMRES, η should be greater than 0.011. Otherwise,it prolongs the period in which RMPC-based input is applied, which is less smooth than C/GMRES-based input. In addition, Fig. 5.6 shows how different η values affect the transition from RMPC-based to C/GMRES-based input. Because a small value of η means that RMPC-based input and C/GMRES-based input are close to each other, the spike generated by switching tends to be lessened as η value decreases. For these reasons, a gap restriction of multiple 0.1 η would be a reasonable candidate. Note that this η value may not be a universal value, and η should be fine-tuned according to the task and its formulation.

The solutions provided by C/GMRES, RMPC, and the combined MPC are compared in Fig. 5.7 .Here, The combined MPC η is set to = 0.3, and the $F_{c/o}$ of the IDCT process is set to 5. The profile of $||F||$ of the combined MPC is depicted in Fig. 5.8. The designed switching mechanism updates control input using the RMPC method rather than the C/GMRES approach when $||F||$ exceeds F_{th} . As a result, the C/GMRES problem is addressed using a combination of the two different optimization strategies. As RMPC-based input is only used for a short period of time, C/GMRES is the major optimization approach.

5.5.3 Discussion

The proposed combination of RMPC with C/GMRES is observed to outperform the individual methods. Since the framework uses C/GMRES as the major optimizer, the smoothness of control input is often increased compared to when RMPC is used completely. On the other

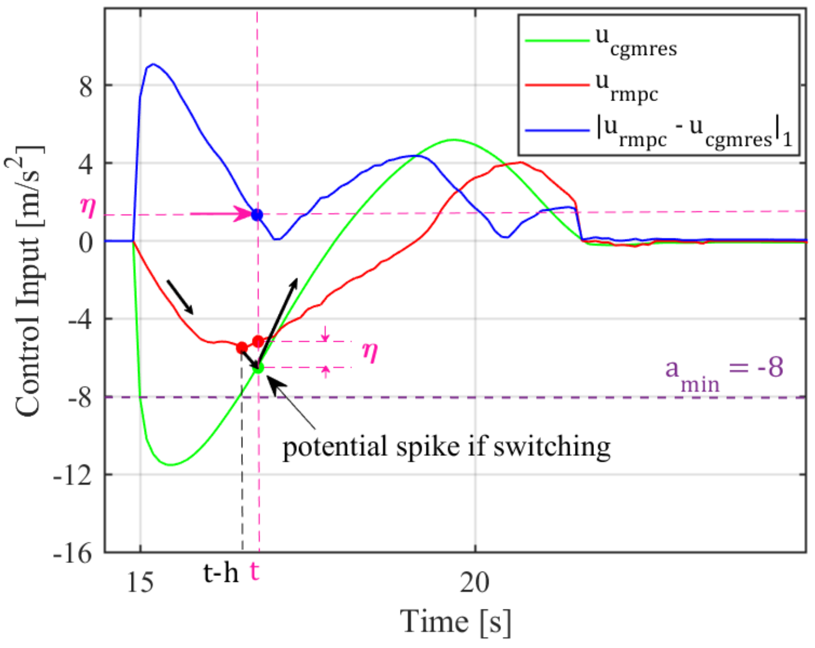


Fig. 5.5 Effect of η investigation on potential spike's intensity

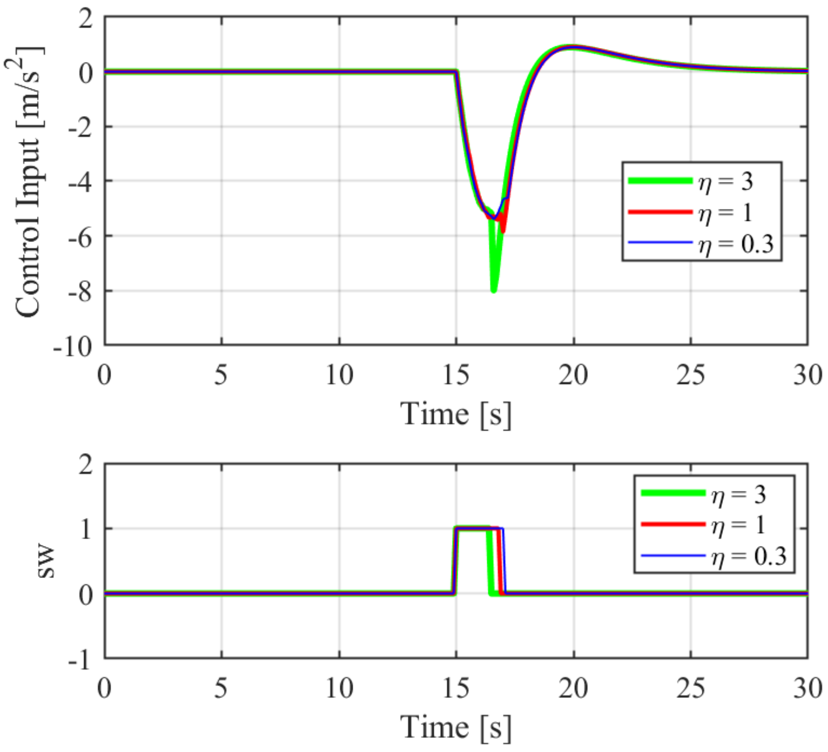


Fig. 5.6 Impact of various values of η on the transition between RMPC and C/GMRES-based input

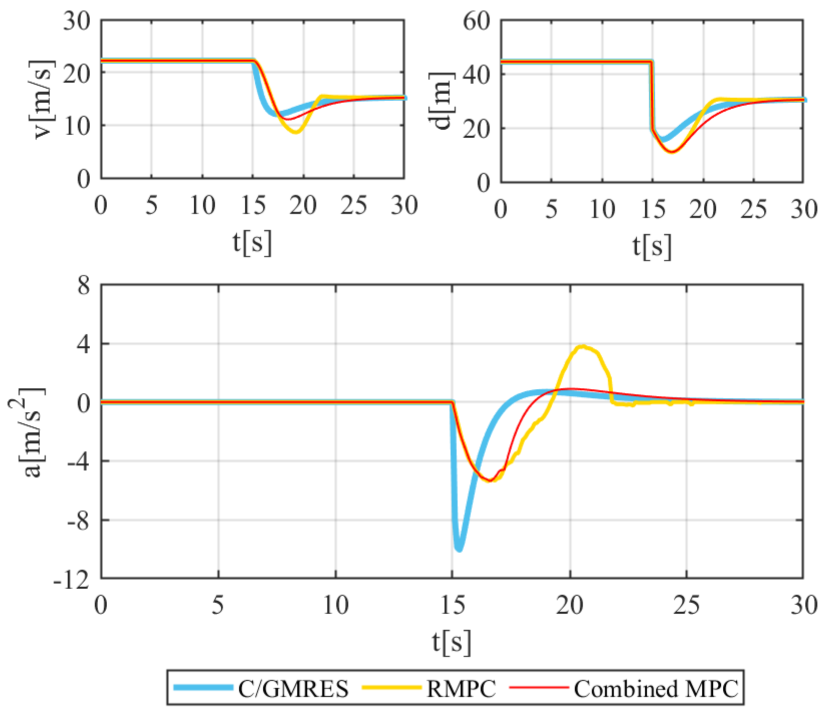


Fig. 5.7 Comparison of Individual optimization methods with the combined MPC

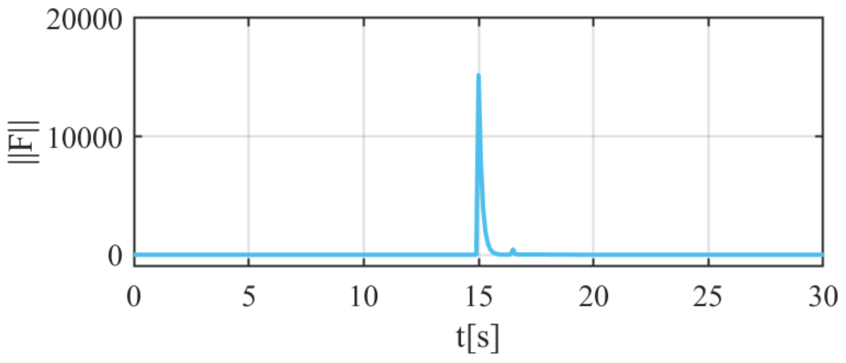


Fig. 5.8 Combined MPC's $||F||$ profile

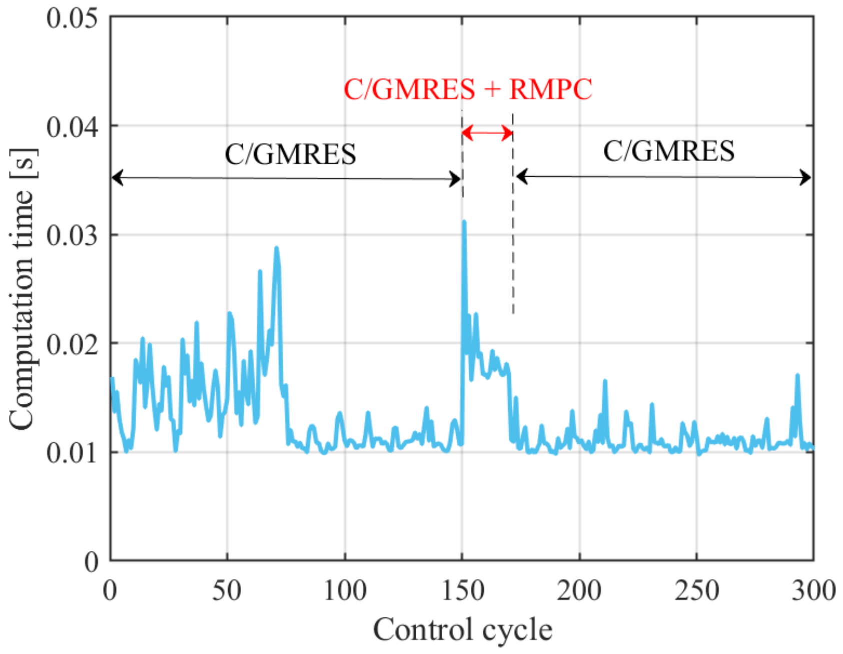


Fig. 5.9 Computation time consumed for input calculation

hand, in the sudden change scenarios where C/GMRES is found to violate constraints, the combined framework stays within constraint limits by switching to RMPC. Such a temporary switch to RMPC also guides the C/GMRES to the optimal solution and thereby stabilizes it faster. As a result, the combined MPC can produce reliable performance without compromising smoothness. Considering the fact that the safety guarantee and smooth driving are significant factors for successful self-driving, the authors believe that the proposed framework is of great interest.

Moreover, in practical implementation of the combined MPC, computational load is a significant concern. Fig.5.9 illustrates the computation time required for input calculation per control cycle. Notably, C/GMRES-based input calculation occurs in every control cycle. However, the introduction of RMPC for additional control input calculation results in a temporary increase in computation time, particularly following sudden cut-in events. Fortunately, the calculation time remains within manageable limits, ensuring that the problem can be solved within the control interval h. This underscores the practical feasibility of the combined MPC approach in real-world applications. Future endeavors will focus on further enhancements in this aspect, aiming to optimize computational efficiency.

5.6 Limitations

Nonetheless, there are significant limitations to this study that will need to be addressed in future research. The nature of soft constraints results in one defect. The combined MPC's C/GMRES-based input value and RMPC-based input value are both the consequence of using the barrier function as a soft constraint in the cost function. Soft constraints, on the other hand, can be solved by definition, despite the fact that they guarantee the existence of a solution. This is demonstrated in section 5.5.1,where the C/GMRES employing barrier function violates the acceleration limitation. It's also a major issue because the soft constraint may create violations of the safe relative distance condition, resulting in collisions. This occurs when the cut-in becomes considerably more aggressive, such as when the third car cuts in at a significantly low velocity than the initial leading car, and/or when the relative space between the cars decreases as a result of the quick cut-in. Even though RMPC can eliminate all infeasible sample series during sample generation, it cannot guarantee that a solution exists. In future work, the problem of the existence of a practicable solution will be better investigated, with more ACC trials demonstrating the limits of the proposed approach by altering the velocity of the two leading vehicles and the drop in relative distance due to cut-in. Furthermore, how to choose values of parameters such as η and F_{th} with respect to diverse driving environments is meant to be included in future work.

In conclusion, the application of the proposed MPC controller extends beyond addressing sudden cut-in scenarios. It proves valuable for a wide range of nonlinear optimal control problems, including situations involving abrupt shifts in the system's state. Here, the synergistic combination of C/GMRES and RMPC proves to be more effective than relying on each optimization method independently.

5.7 Conclusion

In this chapter, we introduce a model predictive controller that integrates C/GMRES and RMPC methodologies to design a fail-safe controller capable of responding effectively to abrupt changes in driving scenarios. To illustrate the efficacy of this controller, we examine its performance in managing the adaptive cruise control system during sudden cut-in scenarios. While C/GMRES excels in continuous driving situations, it struggles to handle unexpected cut-in events, resulting in significant deviations from optimal conditions and breaches of input constraints. Conversely, RMPC demonstrates resilience in the face of sudden changes, albeit with some residual noise in its solutions, despite efforts to enhance smoothness using IDCT. Given the strengths and weaknesses of both methods, our proposed controller operates on the principle of selecting the most suitable optimization method for updating control inputs over time. During continuous driving, it leverages C/GMRES-based inputs. However, when a sudden cut-in event occurs, it seamlessly transitions to RMPC-based inputs and reverts to C/GMRES when specific predefined conditions are met. These conditions ensure a smooth and efficient shift between RMPC and C/GMRES methods. Furthermore, our analysis confirms that the computational time required to update control inputs using the proposed controller remains sufficiently low to enable real-time control implementation. Consequently, the proposed controller stands as a practical solution that can be effectively deployed in real-world automotive experiments. Additionally, we acknowledge the need for addressing issues related to the availability of feasible solutions, fine-tuning parameters to suit varying driving conditions, and exploring further applications for this innovative controller, all of which constitute avenues for our future research efforts.

Chapter 6

Conclusion and future work

6.1 Conclusion

This research has realized higher safety levels in challenging AD tasks using randomized model predictive control (RMPC). RMPC was demonstrated as a powerful tool for achieving nonlinear real-time control targets in autonomous driving (AD) at higher levels of safety. This work started with the identification of major bottlenecks in the application of RMPC to address challenging control problems in AD. In chapter 3, all such bottlenecks were addressed. This includes the following :

- The input signals tend to be noisy since they are originating from the random sampling. This has been addressed with a smoother frequency domain sampling.
- RMPC was not useful at higher sample counts because of the computational burden. This was addressed by a parallel implementation running on GPU.
- RMPC adds complexity to normal MPC in terms of extra variables such as the number of samples. This work proposes an efficient method for parameter selection for RMPC.

It was also demonstrated that RMPC can safely perform high-speed obstacle avoidance driving with an RC car. Chapter 3 is also expected to work as a reference for the readers while deciding whether RMPC is a good controller choice for their problem at hand.

In chapter 4, an interaction problem between a crossing pedestrian and a self-driving car is presented. The pedestrian being modelled by a novel PrARX model, this framework enabled human-like considerate driving. This was achieved using the proposed method of representing stochastic constraints in an RMPC framework. Considering the entropy of pedestrian's decision-making in the cost function, this frame work achieve early motion consensus and hence enable safer interaction. While being human-like in comparison to traditional control solutions, the behavior of the proposed RMPC is also tuneable to suit the driving style of different drivers.

Even though RMPC can handle most of the non-linear control problems well, there are certain cases where a common gradient-based optimisation provides a stable and smooth solution. In chapter 5, a combination of such a commonly used nonlinear MPC framework with RMPC is proposed. The base controller being gradient-based, violates safety constraints under discontinuity. The presented method of supplementing such discontinuous cases with RMPC guarantees safety in all conditions. The sudden switching to RMPC at discontinuity and the switch back once the situation is stable are both presented.

The authors believe that the proposed tools and methods will help to expand the usage of RMPC as a powerful solution for nonlinear optimum control problems.

6.2 Future work

Due to limited time, many different applications and real-world experiments based on RMPC have been left for our future work. Future works will contain certain theoretical investigations and implementation techniques to tackle different control challenges. Experiments with real cars are also planned to ensure the usability of these techniques in real-world self-driving.

The authors share a strong interest in exploring better sampling strategies to cover the solution space of interest with fewer samples. In the RMPC implementations in this paper, the sampling was performed by random numbers generated uniformly in the respective solution spaces. This leads to equally dense sampling in the complete space. This can be replaced with some heuristics on how to sample. Such heuristics can be created based on the nature of the particular control problem, for example, while driving at top speed, acceleration samples can be avoided for speed control of a car. There is also a possibility of sampling at varying densities in different parts of the solution space. This is particularly relevant while processing multiple input systems, where different inputs can be sampled with different strategies and then combined to form better coverage. Such a sampling ensure deep coverage at solution spaces of interest without compromising the inclusion of parts of the solution space.

The authors think that the flexibility in choosing sample sizes in RMPC will enable the use of MPC in computing platforms that are considered too slow to run a real-time nonlinear MPC. This idea will be explored on some common low-cost processors that do not support floating-point calculations. This work is also motivated by the intention to explore the possibility running MPC in a typical automotive ECU. Even though MPC in a low-cost micro-controller is often considered impossible, the authors expect RMPC to change this belief.

Even though RC car experiments with RMPC have been demonstrated in this work, real car experiments haven't been carried out. This will be a fascinating aspect of future work. It will include real car experiments for the problem statements in chapters 3-5. The intention is to compare RMPC implementation with current state of the art for the respective tasks. This could establish a comparison in performance and computational resource requirements at the same time.

Acknowledgements

I would like to express my deepest gratitude to my supervisor Prof.Dr.Tatsuya Suzuki and Dr.Hiroyuki Okuda for their invaluable advice and encouragement during my PhD study. Your continued support and encouragement are appreciated more than you know. I would like to thank my friends, Suzuki lab members and colleagues for a cherished time spent together. I also could not have undertaken this journey without my defense committee, who generously provided knowledge and expertise to refine my work.

I would like to extend my sincere thanks to my coauthor Dr.Anh-Tuan Tran for the interesting work that motivated me towards chapter 4. Also to Ms.Ha Bui and Ms.Srishti Sinha for the contributions towards chapter 5.

Additionally, this endeavor would not have been possible without the support from the Ministry of Education, Culture, Sports, Science and Technology (MEXT). MEXT has supported me with a generous scholarship during my time at Nagoya University. Thanks should also go to Toyota Motor Corporation and Kawasaki Heavy Industries for the interesting collaborative projects where I could apply my research as a research assistant. I am also grateful to Rapyuta Robotics, Onclouds Inc. and Robert Bosch Japan. Internship opportunities with these companies helped me greatly to develop my engineering skills.

Lastly, I would be remiss in not mentioning my family, especially my sister who has helped me write a well-worded thesis. Their belief in me has kept my spirits and motivation high during this process.

References

[1] coalitionforfuturemobility. Benefits of self-driving vehicles. https://www2.deloitte.com/content/dam/Deloitte/us/Documents/manufacturing/us-global-automotive-consumer-study-2019.pdf, 2022.

[2] Synopsis. What's next for autonomous vehicles? https://www.synopsys.com/automotive/autonomous-driving-levels.html, 2022.

[3] McKinsey&Company. What's next for autonomous vehicles? https://www.mckinsey.com/features/mckinsey-center-for-future-mobility/our-insights/whats-next-for-autonomous-vehicles, 2021.

[4] David Silver. How control works for self-driving cars. https://www.linkedin.com/pulse/how-control-works-self-driving-cars-david-silver/, 2018.

[5] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[6] H. Okuda, N. Ikami, T. Suzuki, Y. Tazaki, and K. Takeda. Modeling and analysis of driving behavior based on a probability-weighted ARX model. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):98–112, 2013.

[7] WHO. Road traffic injuries. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, 2022.

[8] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, 2015.

[9] Travis J Crayton and Benjamin Mason Meier. Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy. *Journal of Transport & Health*, 6:245–252, 2017.

[10] Marc Scribner. Self-driving regulation. *Report, Competitive Enterprise Institute*, 2014.

[11] tomorrowsworldtoday. History of autonomous cars. https://www.tomorrowsworldtoday.com/2021/08/09/history-of-autonomous-cars/, 2021.

[12] Peter Stanchev and John Geske. Autonomous cars. history. state of art. research problems. In *International Conference on Distributed Computer and Communication Networks*, pages 1–10. Springer, 2015.

[13] Todd Jochem, Dean Pomerleau, Bala Kumar, and Jeremy Armstrong. Pans: A portable navigation platform. In *Proceedings of the Intelligent Vehicles' 95. Symposium*, pages 107–112. IEEE, 1995.

[14] BBC. Uber sells self-driving cars to focus on profits. https://www.bbc.com/news/business-55224462, 2020.

[15] Damon Lavrinc. This is how bad self-driving cars suck in the rain. https://jalopnik.com/this-is-how-bad-self-driving-cars-suck-in-the-rain-1666268433, 2014.

[16] Alex Davies. Google's self-driving car caused its first crash. https://www.wired.com/2016/02/googles-self-driving-car-may-caused-first-crash/, 2016.

[17] Matt McFarland. Who's responsible when an autonomous car crashes? https://money.cnn.com/2016/07/07/technology/tesla-liability-risk/index.html, 2016.

[18] Deloitte. Deloitte global automotive consumer study — advanced vehicle technologies and multimodal transportation, global focus countries. https://coalitionforfuturemobility.com/benefits-of-self-driving-vehicles/, May 2019.

[19] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[20] M Ohshima and M Ogawa. Model predictive control-i-basic principle: history & present status. *SYSTEMS CONTROL AND INFORMATION*, 46(5):286–293, 2002.

[21] Manabu Kano and M Oshima. Model predictive control-ii: Linear model predictive control. *Trans. on the Institute of Sys., Cont. and Info. Eng.*, 46(7), 2002.

[22] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on control systems technology*, 15(3):566–580, 2007.

[23] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[24] J Rault, A Richalet, JL Testud, and J Papon. Model predictive heuristic control: application to industrial processes. *Automatica*, 14(5):413–428, 1978.

[25] Charles R Cutler and Brian L Ramaker. Dynamic matrix control?? a computer control algorithm. In *joint automatic control conference*, number 17, page 72, 1980.

[26] David M Prett and RD Gillette. Optimization and constrained multivariable control of a catalytic cracking unit. In *Joint automatic control conference*, number 17, page 73, 1980.

[27] Carlos E Garcia and Manfred Morari. Internal model control. a unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323, 1982.

[28] David W Clarke, Coorous Mohtadi, and P Simon Tuffs. Generalized predictive control—part i. the basic algorithm. *Automatica*, 23(2):137–148, 1987.

[29] David W Clarke, Coorous Mohtadi, and P Simon Tuffs. Generalized predictive control—part ii extensions and interpretations. *Automatica*, 23(2):149–160, 1987.

[30] Jay H Lee, Manfred Morari, and Carlos E Garcia. State-space interpretation of model predictive control. *Automatica*, 30(4):707–717, 1994.

[31] Alberto Bemporad and Manfred Morari. Robust model predictive control: A survey. In *Robustness in identification and control*, pages 207–226. Springer, 1999.

[32] Peter J Campo and Manfred Morari. -norm formulation of model predictive control problems. In *1986 American Control Conference*, pages 339–343. IEEE, 1986.

[33] Peter J Campo and Manfred Morari. Robust model predictive control. In *1987 American control conference*, pages 1021–1026. IEEE, 1987.

[34] Daniel L Laughlin, Daniel E Rivera, and Manfred Morari. Smith predictor design for robust performance. *International Journal of Control*, 46(2):477–504, 1987.

[35] Alex Zheng and Manfred Morari. Stability of model predictive control with mixed constraints. *IEEE Transactions on automatic control*, 40(10):1818–1823, 1995.

[36] Christopher V Rao, James B Rawlings, and Jay H Lee. Constrained linear state estimation—a moving horizon approach. *Automatica*, 37(10):1619–1628, 2001.

[37] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit solution of model predictive control via multiparametric quadratic programming. In *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, volume 2, pages 872–876. IEEE, 2000.

[38] 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.

[39] Mina Kamel, Michael Burri, and Roland Siegwart. Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles. *IFAC-PapersOnLine*, 50(1):3463–3469, 2017. 20th IFAC World Congress.

[40] Wikipedia. Model predictive control. https://en.wikipedia.org/wiki/Model_predictive_control, 2022.

[41] Rolf Findeisen and Frank Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux meeting on systems and control*, volume 11, pages 119–141. Citeseer, 2002.

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

[43] Míriam R García, Carlos Vilas, Lino O Santos, and Antonio A Alonso. A robust multi-model predictive controller for distributed parameter systems. *Journal of Process Control*, 22(1):60–71, 2012.

[44] Reza Kamyar and Ehsan Taheri. Aircraft optimal terrain/threat-based trajectory planning and control. *Journal of Guidance, Control, and Dynamics*, 37(2):466–483, 2014.

[45] Jorge L Piovesan and Herbert G Tanner. Randomized model predictive control for robot navigation. In *2009 IEEE International Conference on Robotics and Automation*, pages 94–99. IEEE, 2009.

[46] Georg Schildbach, Giuseppe C Calafiore, Lorenzo Fagiano, and Manfred Morari. Randomized model predictive control for stochastic linear systems. In *2012 American Control Conference (ACC)*, pages 417–422. IEEE, 2012.

[47] Mathukumalli Vidyasagar. Randomized algorithms for robust controller synthesis using statistical learning theory. *Automatica*, 37(10):1515–1528, 2001.

[48] Damion D Dunlap, Emmanuel G Collins Jr, and Charmane V Caldwell. Sampling based model predictive control with application to autonomous vehicle guidance. In *Florida Conference on Recent Advances in Robotics*, 2008.

[49] Xiaojing Zhang, Sergio Grammatico, Georg Schildbach, Paul Goulart, and John Lygeros. On the sample size of random convex programs with structured dependence on the uncertainty. *Automatica*, 60:182–188, 2015.

[50] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[51] J-P Laumond, Paul E Jacobs, Michel Taix, and Richard M Murray. A motion planner for nonholonomic mobile robots. *IEEE Transactions on robotics and automation*, 10(5):577–593, 1994.

[52] Long Han, Hironari Yashiro, Hossein Tehrani Nik Nejad, Quoc Huy Do, and Seiichi Mita. Bezier curve based path planning for autonomous vehicle in urban environment. In *2010 IEEE intelligent vehicles symposium*, pages 1036–1042. IEEE, 2010.

[53] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[54] Shai A Arogeti and Nadav Berman. Path following of autonomous vehicles in the presence of sliding effects. *IEEE Transactions on Vehicular Technology*, 61(4):1481–1492, 2012.

[55] Ayame Koga, Hiroyuki Okuda, Yuichi Tazaki, Tatsuya Suzuki, Blaine Levedahl, Kentaro Haraguchi, and Zibo Kang. Autonomous lane tracking reflecting skilled/un-skilled driving characteristics. In *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pages 003175–003180. IEEE, 2015.

[56] Hiroyuki Okuda, Yingqi Liang, and Tatsuya Suzuki. Sampling based predictive control with frequency domain input sampling for smooth collision avoidance. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2426–2431. IEEE, 2018.

[57] Hiroyuki Okuda, Nobuto Sugie, and Tatsuya Suzuki. Realtime collision avoidance control based on continuation method for nonlinear model predictive control with safety constraint. In *2017 11th Asian Control Conference (ASCC)*, pages 1086–1091. IEEE, 2017.

[58] Christian Gotte, Martin Keller, Carsten Hass, Karl-Heinz Glander, Alois Seewald, and Torsten Bertram. A model predictive combined planning and control approach for guidance of automated vehicles. In *2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 69–74. IEEE, 2015.

[59] Yiqi Gao, Andrew Gray, H Eric Tseng, and Francesco Borrelli. A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles. *Vehicle System Dynamics*, 52(6):802–823, 2014.

[60] Michael Garland, Scott Le Grand, John Nickolls, Joshua Anderson, Jim Hardwick, Scott Morton, Everett Phillips, Yao Zhang, and Vasily Volkov. Parallel computing experiences with cuda. *IEEE micro*, 28(4):13–27, 2008.

[61] Alex Brooks, Tobias Kaupp, and Alexei Makarenko. Randomised mpc-based motion-planning for mobile robot obstacle avoidance. In *2009 IEEE International Conference on Robotics and Automation*, pages 3962–3967. IEEE, 2009.

[62] Jesús Velasco Carrau, Alexander Liniger, Xiaojing Zhang, and John Lygeros. Efficient implementation of randomized mpc for miniature race cars. In *2016 European Control Conference (ECC)*, pages 957–962. IEEE, 2016.

[63] Ajay Kumar Sampathirao, Pantelis Sopasakis, Alberto Bemporad, and Panagiotis Panos Patrinos. Gpu-accelerated stochastic predictive control of drinking water networks. *IEEE Transactions on Control Systems Technology*, 26(2):551–562, 2017.

[64] Duc-Kien Phung, Bruno Hérissé, Julien Marzat, and Sylvain Bertrand. Model predictive control for autonomous navigation using embedded graphics processing unit. *IFAC-PapersOnLine*, 50(1):11883–11888, 2017.

[65] Yang Gang and Liu Mingguang. Acceleration of mpc using graphic processing unit. In *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, pages 1001–1004. IEEE, 2012.

[66] Shimpei Ohyama and Hisashi Date. Parallelized nonlinear model predictive control on gpu. In *2017 11th Asian Control Conference (ASCC)*, pages 1620–1625. IEEE, 2017.

[67] Karam M Abughalieh and Shadi G Alawneh. A survey of parallel implementations for model predictive control. *IEEE Access*, 7:34348–34360, 2019.

[68] Arun Muraleedharan, Hiroyuki Okuda, and Tatsuya Suzuki. Path tracking control using model predictive control with on gpu implementation for autonomous driving. *Journal of Arid Land Studies*, 28(S):163–167, 2018.

[69] Arun Muraleedharan, Hiroyuki Okuda, and Tatsuya Suzuki. Improvement of control performance of sampling based model predictive control using gpu. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1999–2004. IEEE, 2019.

[70] Masato Abe. *Vehicle handling dynamics: theory and application*. Butterworth-Heinemann, 2015.

[71] C++. C++ numerics library, 2019.

[72] NVIDIA. Nvidia cuda c++ programming guide, 2019.

[73] NVIDIA. Nvidia cuda curand library, 2019.

[74] Sean Brennan and Andrew Alleyne. Using a scale testbed: Controller design and evaluation. *IEEE Control Systems Magazine*, 21(3):15–26, 2001.

[75] Arun Muraleedharan, Hiroyuki Okuda, and Tatsuya Suzuki. Real-time implementation of randomized model predictive control for autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 7(1):11–20, 2021.

[76] Axel Niehaus and Robert F Stengel. An expert system for automated highway driving. *IEEE Control Systems Magazine*, 11(3):53–61, 1991.

[77] Benoit Vanholme, Dominique Gruyer, Benoit Lusetti, Sébastien Glaser, and Saïd Mammar. Highly automated driving on highways based on legal safety. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):333–347, 2012.

[78] Anh Tuan Tran, Masato Kawaguchi, Hiroyuki Okuda, and Tatsuya Suzuki. A model predictive control-based lane merging strategy for autonomous vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 594–599, 2019.

[79] Yinan Zheng, Thomas Chase, Lily Elefteriadou, Bastian Schroeder, and Virginia P Sisiopiku. Modeling vehicle–pedestrian interactions outside of crosswalks. *Simulation Modelling Practice and Theory*, 59:89–101, 2015.

[80] Ankit Kathuria and Perumal Vedagiri. Evaluating pedestrian vehicle interaction dynamics at un-signalized intersections: a proactive approach for safety analysis. *Accident Analysis & Prevention*, 134:105316, 2020.

[81] Su Yang. Driver behavior impact on pedestrians' crossing experience in the conditionally autonomous driving context, 2017.

[82] Thibault Kruse, Amit Kumar Pandey, Rachid Alami, and Alexandra Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

[83] Hiroyuki Okuda, Tatsuya Suzuki, Kota Harada, Shintaro Saigo, and Satoshi Inoue. Quantitative driver acceptance modeling for merging car at highway junction and its application to the design of merging behavior control. *IEEE Transactions on Intelligent Transportation Systems*, 22(1):329–340, 2021.

[84] Anh Tuan Tran, Noboru Sakamoto, and Tatsuya Suzuki. A combination of analytical and model predictive optimal methods for adaptive cruise control problem. In *2018 Australian & New Zealand Control Conference (ANZCC)*, pages 119–124, 2018.

[85] Kai Eckoldt, Marc Hassenzahl, Matthias Laschke, Thies Schneider, Josef Schumann, and Stefan Könsgen. The Gentleman. A prosocial assistance system to promote considerate driving. In *Proceedings on the 10th Conference on Design and Emotion*, pages 307–314, 2016.

[86] Hiroyuki Okuda. Realization of considerate driving based on behavioral prediction and inducement of other traffic participants. *Traffic sciences*, 52(1):18–23, 2021.

[87] C. Daniel Batson and Adam A. Powell. *Altruism and Prosocial Behavior*, chapter 19, pages 463–484. John Wiley & Sons, Ltd, 2003.

[88] Paul B Harris, John M Houston, Jose A Vazquez, Janan A Smither, Amanda Harms, Jeffrey A Dahlke, and Daniel A Sachau. The prosocial and aggressive driving inventory (PADI): A self-report measure of safe and unsafe driving behaviors. *Accident Analysis & Prevention*, 72:1–8, 2014.

[89] Julian Francisco Pieter Kooij, Nicolas Schneider, Fabian Flohr, and Dariu M Gavrila. Context-based pedestrian path prediction. In *European Conference on Computer Vision*, pages 618–633. Springer, 2014.

[90] Matthias Luber, Luciano Spinello, Jens Silva, and Kai O Arras. Socially-aware robot navigation: A learning approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 902–907. IEEE, 2012.

[91] Jos Elfring, René Van De Molengraft, and Maarten Steinbuch. Learning intentions for improved human motion prediction. *Robotics and Autonomous Systems*, 62(4):591–602, 2014.

[92] Gonzalo Ferrer and Alberto Sanfeliu. Behavior estimation for a complete framework for human motion prediction in crowded environments. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5940–5945. IEEE, 2014.

[93] Anirudh Vemula, Katharina Muelling, and Jean Oh. Modeling cooperative navigation in dense human crowds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1685–1692. IEEE, 2017.

[94] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016.

[95] Arun Muraleedharan, Anh-Tuan Tran, Hiroyuki Okuda, and Tatsuya Suzuki. Scenario-based model predictive speed controller considering probabilistic constraint for driving scene with pedestrian. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2020.

[96] Anh-Tuan Tran, Arun Muraleedharan, Hiroyuki Okuda, and Tatsuya Suzuki. Scenario-based stochastic MPC for vehicle speed control considering the interaction with pedestrians. In *20th World Congress of the International Federation of Automatic Control*, 2020.

[97] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[98] Kai Chen, Xiao Song, and Xiaoxiang Ren. Modeling social interaction and intention for pedestrian trajectory prediction. *Physica A: Statistical Mechanics and its Applications*, 570:125790, 2021.

[99] Julian Bock., Jens Kotte., Till Beemelmanns., and Markus Klösges. Self-learning trajectory prediction with recurrent neural networks at intelligent intersections. In *Proceedings of the 3rd International Conference on Vehicle Technology and Intelligent Transport Systems - VEHITS,*, pages 346–351. INSTICC, SciTePress, 2017.

[100] Fanta Camara, Nicola Bellotto, Serhan Cosar, Florian Weber, Dimitris Nathanael, Matthias Althoff, Jingyuan Wu, Johannes Ruenz, André Dietrich, Gustav Markkula, et al. Pedestrian models for autonomous driving part ii: high-level models of human behavior. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5453–5472, 2020.

[101] Nico Kaempchen, Kristian Weiss, Michael Schaefer, and Klaus CJ Dietmayer. IMM object tracking for high dynamic driving maneuvers. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 825–830. IEEE, 2004.

[102] Nicolas Schneider and Dariu M Gavrila. Pedestrian path prediction with recursive bayesian filters: A comparative study. In *German Conference on Pattern Recognition*, pages 174–183. Springer, 2013.

[103] Andreas T Schulz and Rainer Stiefelhagen. A controlled interactive multiple model filter for combined pedestrian intention recognition and path prediction. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 173–178. IEEE, 2015.

[104] Dongfang Yang, Linhui Li, Keith Redmill, and Ümit Özgüner. Top-view trajectories: A pedestrian dataset of vehicle-crowd interaction from controlled experiments and crowded campus. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 899–904. IEEE, 2019.

[105] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[106] Transport Ministry of Land Infrastructure and Tourism. Road structure ordinance, article 5. https://www.mlit.go.jp/road/road_e/q4_standard.html, May 2022.

[107] Dongfang Yang. Vehicle-Crowd Intraction (VCI) - CITR Dataset. https://github.com/dongfang-steven-yang/vci-dataset-citr, 2021.

[108] P.S. Bokare and A.K. Maurya. Acceleration-deceleration behaviour of various vehicle types. *Transportation Research Procedia*, 25:4733–4749, 2017. World Conference on Transport Research - WCTR 2016 Shanghai. 10-15 July 2016.

[109] Yun Li, Kiam Heong Ang, and Gregory CY Chong. Pid control system analysis and design. *IEEE Control Systems Magazine*, 26(1):32–41, 2006.

[110] Annika FL Larsson, Katja Kircher, and Jonas Andersson Hultgren. Learning from experience: Familiarity with acc and responding to a cut-in situation in automated driving. *Transportation research part F: traffic psychology and behaviour*, 27:229–237, 2014.

[111] V Balaji and DL Rajaji. Comparative study of pid and mpc controller using labview. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2(11):5545–5550, 2013.

[112] Melanie Nicole Zeilinger, Colin Neil Jones, and Manfred Morari. Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization. *IEEE Transactions on Automatic Control*, 56(7):1524–1534, 2011.

[113] Matthias A Müller and Frank Allgöwer. Improving performance in model predictive control: Switching cost functionals under average dwell-time. *Automatica*, 48(2):402–409, 2012.

[114] Andong Liu, Li Yu, and Wen-an Zhang. Switched model predictive control for networked control systems with time delays and packet disordering. *IFAC Proceedings Volumes*, 47(3):3764–3769, 2014.

[115] Jingsong Wang, Michal Grochowski, and Mietek A Brdys. Analysis and design of softly switched model predictive control. *IFAC Proceedings Volumes*, 38(1):76–81, 2005.

[116] Xing Qian, Kejin Huang, Shengkun Jia, Haisheng Chen, Yang Yuan, Liang Zhang, and Shaofeng Wang. Composition/temperature cascade control for a kaibel dividing-wall distillation column by combining pi controllers and model predictive control integrated with soft sensor. *Computers & Chemical Engineering*, 126:292–303, 2019.

[117] Weihua Zhao and Tiauw Hiong Go. Quadcopter formation flight control combining mpc and robust feedback linearization. *Journal of the Franklin Institute*, 351(3):1335–1355, 2014.

[118] Junmin Wang and Rajesh Rajamani. Adaptive cruise control system design and its impact on highway traffic flow. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*, volume 5, pages 3690–3695. IEEE, 2002.

[119] Yinglong He, Biagio Ciuffo, Quan Zhou, Michail Makridis, Konstantinos Mattas, Ji Li, Ziyang Li, Fuwu Yan, and Hongming Xu. Adaptive cruise control strategies implemented on experimental vehicles: A review. *IFAC-PapersOnLine*, 52(5):21–27, 2019.

[120] Edwin F Meyer III. Multiple-car pileups and the two-second rule. *Physics Teacher*, 32(8):496–97, 1994.

[121] Jürgen Hasch, Eray Topak, Raik Schnabel, Thomas Zwick, Robert Weigel, and Christian Waldschmidt. Millimeter-wave technology for automotive radar sensors in the 77 ghz frequency band. *IEEE transactions on microwave theory and techniques*, 60(3):845–860, 2012.

[122] Yuto Owaki, Tsuyoshi Yuno, and Taketoshi Kawabe. Nonlinear model predictive control for path following of simple small electric vehicle using c/gmres. *IFAC-PapersOnLine*, 51(20):253–258, 2018.

[123] Hiroyuki Okuda, Nobuto Sugie, Tatsuya Suzuki, Kentaro Haraguchi, and Zibo Kang. Simultaneous realization of decision, planning and control for lane-changing behavior using nonlinear model predictive control. *IEICE TRANSACTIONS on Information and Systems*, 103(12):2632–2642, 2020.

[124] Richard Barrett, Michael Berry, Tony F Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van der Vorst. *Templates for the solution of linear systems: building blocks for iterative methods*. SIAM, 1994.