

Clemson University

TigerPrints

All Theses

Theses

August 2021

Data-driven System Identification and Optimal Control Framework for Grand-Prix Style Autonomous Racing

Rongyao Wang

Clemson University, rywang5235@gmail.com

Follow this and additional works at: https://tigerprints.clemson.edu/all_theses

Recommended Citation

Wang, Rongyao, "Data-driven System Identification and Optimal Control Framework for Grand-Prix Style Autonomous Racing" (2021). *All Theses*. 3601.

https://tigerprints.clemson.edu/all_theses/3601

This Thesis is brought to you for free and open access by the Theses at TigerPrints. It has been accepted for inclusion in All Theses by an authorized administrator of TigerPrints. For more information, please contact kokeefe@clemson.edu.

DATA-DRIVEN SYSTEM IDENTIFICATION AND OPTIMAL
CONTROL FRAMEWORK FOR GRAND-PRIX STYLE
AUTONOMOUS RACING

A Thesis
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
Mechanical Engineering

by
Rongyao Wang
August 2021

Accepted by:
Dr. Yue Wang, Committee Chair
Dr. Yiqiang Han, Research Chair
Dr. Mohammad Naghnaeian

Abstract

For the past 30 years, autonomous driving has witnessed a tremendous improvements thanks to the surge of computing power. Not only did we witness the autonomous vehicle navigate itself safely in the urban area, stories about more diverse autonomous driving applications, such as off-road rally-style navigation, are also commonly mentioned. Just until recently, the exponential increase in GPU and high-performance computing technology has motivated the research on autonomous driving under extreme situations such as autonomous racing or drifting.[25] The motivation for this thesis is to offer a brief overview about the main challenge of autonomous driving control and planning in racing scenario along with the potential solutions.

The first contribution is using koopman operator and deep neural network to perform data-driven system identification. We then design optimal model-based control which is based on the learned dynamics alone. Based on our new system identification algorithm, we can approximate an accurate, explainable, and linearized system representation in a high-dimensional latent space, without any prior knowledge of the system. In this case, the learned vehicle dynamic automatically involves the information that is normally difficult to obtain, including cornering stiffness, tire slip, transmission parameters, etc. Our result shows that our koopman data-driven optimal control approach is able to deliver better tracking accuracy at high speed compared to the state-of-art vehicle controllers.

The second contribution is an iterative learning and sampling algorithm that can

perform minimum-time optimization of the global racing trajectory(aka racing line) within the limit of tire friction. This trajectory optimization algorithm is not only proven to be computationally efficient, but also safe enough for the onboard RC vehicle's test.

The research achievements we made for the last two years not only enables the F1TENTH racing team of Clemson University Mechanical Engineering Department to finish top 5 in both virtual autonomous racing hosted by IFAC and IROS congress, but also offer us the opportunity to join ICRA 2021 Autonomous racing workshop to present our work and being awarded the joint best paper. More importantly, these contributions proved to be functional and effective in the on-board testing of the real F1TENTH robot's autonomous navigation in the Flour Danial basement. Finally, this thesis will also include discussions of the potential research directions that can help improve our current method so that it can better contribute to the autonomous driving industry.

Acknowledgments

Firstly, I would love to express my gratitude to Dr. Yiqiang Han who has been my mentor and academic advisor since my senior year in Clemson. There is no doubt that he is the person who open door for me to the world of autonomous driving and robotic research. He is also the one who shaped me into a well-rounded researcher by offering me all these precious advises. My research experience is undoubtedly one of the most important part of life.

Secondly, I would also love to show my gratitude to Dr. Umesh Vaidya for offering precious academic advise in the field of data-driven system identification and control. Our collaboration since September 2020 has been the most productive period of time since I joined the graduate school in mechanical engineering. He is also the person who show me the brand new the territory of advanced control theory, which has completely reshaped my previous idea toward dynamics ans control theory.

Thirdly, I would love to thank Dr. Yue Wang who is the committee chair of my master thesis degree. I couldn't be more grateful for the professional advises and enlightening lectures from her. Also, I must thank to Dr. Ardalan for building up the my knowledge base of optimal control theory which has proven to be exceptionally helpful to my thesis research.

In addition, I must deliver my most profound and deepest gratitude to my parents in offering me both financial and emotional supports. Without their advises and encourage-

ments, I would never had the courage to step out my comfort zone to explore the world on the other side of the earth.

Lastly, I must also thank to all my colleagues in my research group, especially Alex Krolicki and Josepa Moyalan who assisted me in conducting on-board vehicle experiments.

Table of Contents

Title Page	i
Abstract	ii
Acknowledgments	iv
List of Tables	viii
List of Figures	ix
1 Introduction	1
1.1 Driving at the edge of friction limit	2
1.2 Research Contribution and Outline	6
2 Minimum-time global trajectory optimization	9
2.1 Minimum-time navigation within tire friction limit	9
2.2 Trade-off between trajectory’s curvature and distance	14
3 Vehicle control on the limit of tire friction	16
3.1 State-of-the-Art vehicle controller	16
3.2 Data-driven optimization-based vehicle controller	22
4 Results Analysis	33
4.1 Controller Specification	34
4.2 F1TENTH Simulator Experiment Result	35
4.3 Onboard Vehicle Experiment Result: 1/12th Scale Ackermann Steering Navigation Robot	41
5 Conclusions and Discussion	49
5.1 Stable and Efficient Control Performance of Deep Koopman Data-Driven MPC	49
5.2 Improvement on lap-time resulting from high tracking performance	50
5.3 Comparison with the other accomplished work on high performance vehi- cle control	51

5.4	Recommendations for Further Research	53
Appendices	55
A	Hardware configuration	56
B	Clemson Fluor Daniel EIB Basement's Onboard Experiment	57
C	Controller Specification in Simulator and RC Robot	60
D	Koopman Operator Training Specification	64
Bibliography	68

List of Tables

4.1	Minimum Time Statistic over Different Road Friction and Minimum Distance Weight	36
4.2	Tracking Performance Comparison Statistic	38
4.3	Navigation Time Comparison Statistic	44
4.4	Tracking Performance Comparison Statistic of Onboard Testing	45
5.1	Comparison of Lap-time of different Vehicle Controllers	50
5.2	Comparison of Different Optimization-based Vehicle Controllers	52
3	VESC specification for 1/12 Scale RC car	56
4	Model predictive control setting of nonlinear kinematic model in F1TENTH Simulator	60
5	Model predictive control setting of Deep Koopman Vehicle Model in F1TENTH Simulator	61
6	Model predictive control setting of nonlinear kinematic model in 1/12th Scaled RC Robot Test	62
7	Model predictive control setting of Deep Koopman Vehicle Model in 1/12th Scaled RC Robot Test	63
8	Deep Neural Network Observable Function Training Specification	64
9	Polynomial Observable Function Approximation Specification	65
10	Deep Neural Network for Augmented Observable Function Training Specification	66
11	Deep Neural Network for Augmented Observable Function Training Specification	67

List of Figures

1.1	Tire friction of F1TENTH robot. Left: free body diagram of vehicle tire force. Right: tire friction cycle constraint demonstration	3
2.1	State update demonstration of local trajectory optimization	13
2.2	Iterative Real-time Local Planning Demonstration	15
3.1	Pure Pursuit Controller Demonstration	18
3.2	Deep Koopman Data-Driven Optimal Control Framework	27
3.3	Vehicle dynamics identification based on hybrid observable function	28
3.4	Control Flowchart over Lifted Space	31
4.1	Global Trajectory Optimization Progress over a Closed Race Track	36
4.2	Tracking Error Comparison, blue bar represents the mean error value while black bar shows the range of error	38
4.3	Trajectory tracking of three different vehicle controllers. (a): Data-driven MPC (b): Kinematic NMPC (c): Pure Pursuit + PID	39
4.4	Control Output of Different Controller over a Single Lap	40
4.5	Obstacle Avoidance Test of Deep Koopman MPC in F1TENTH Simulator	41
4.6	ROS communication Setup for Onboard F1TENTH Experiment	42
4.7	Optimal Trajectory Generation through Iterative Methods	44
4.8	Onboard Tracking Record of Deep Koopman Model Predictive Control	46
4.9	Onboard Tracking Record of Nonlinear Kinematic Model Predictive Control	46
4.10	Onboard Tracking Record of Adaptive Pure Pursuit Predictive Control	47
4.11	Onboard Testing Control Comparison	47
1	Hardware configuration of the ackermann steering autonomous robot for testing	56
2	Build costmap of Clemson EIB basement	57
3	Use part of the costmap of EIB basement as racing track	58
4	RVIZ visualization of F1TENTH onboard navigation	58
5	Onboard testing of high speed navigation of 1/12 scaled ackermann steering robot	59

Chapter 1

Introduction

Despite the witness of autonomous vehicles navigate itself safely on the road, the full potential of autonomous vehicle has yet to be unleashed. For the last century, motor-sport has motivate the development of high-speed transportation system. It is reasonable to predict that today's high speed will eventually become the normal speed of tomorrow.

Once the human transportation system rush closer to L5 level, it become necessary for the vehicle to raise up the speed, which will push the vehicle to their handling limit. In this case, the autonomous vehicle's planning and control algorithm needs to be more accurate the efficient. Also, knowing the vehicle's behavior at its handling limits can have a significant improve the overall safety of the autonomous vehicle. Therefore, it is important to come up with a new vehicle design framework under racing speed.

To further explore the potential of autonomous vehicle under high speed and its handling limits, this thesis conducts detailed investigation over two major parts of autonomous vehicle: 1. Vehicle dynamics identification and control 2. Optimal navigation trajectory planning over its handling limits.

1.1 Driving at the edge of friction limit

When vehicle navigates on the road, what defines the vehicle's motion is the friction force between vehicle's tires and the road surface. In other words, whatever the other components are designed for, the final "judge" of the vehicle's behavior are the four tires that directly contact the road surface. As a result, when designing the vehicle planner and controller, it is important to setup constraints on maximum tire forces. In our design of controller and planner, as suggest in some summary works of vehicle dynamics[3][21], we incorporate maximum tire-friction cycle as major tire force constraint.

1.1.1 Vehicle dynamic model and tire friction cycle

As shown in Fig (1.1), we can assume each individual tire is under the affect of three forces: vertical force F_z , lateral force F_y and longitudinal force F_x . The forces act on each tire must satisfy the constraint as shown in Eqn.[1.1], which is equivalent to the right picture of Fig. (1.1). The vehicle's longitudinal performance(acceleration and braking) is related to the longitudinal force F_x while the vehicle's lateral performance(steering) is related to the lateral force F_y .

$$\mu F_z \geq \sqrt{F_x^2 + F_y^2} \quad (1.1)$$

In Eqn.[1.1], one important thing is that the vehicle turning force and longitudinal force are negatively related. For example, when vehicle is heavily braking or accelerating(high F_x), due to the maximum total force available being limited by the friction cycle(friction coefficient μ times vertical force F_z), the lateral forces F_y left for vehicle's turning has to be limited. Similarly, when vehicle is turning in high speed, the longitudinal force has to be limited.

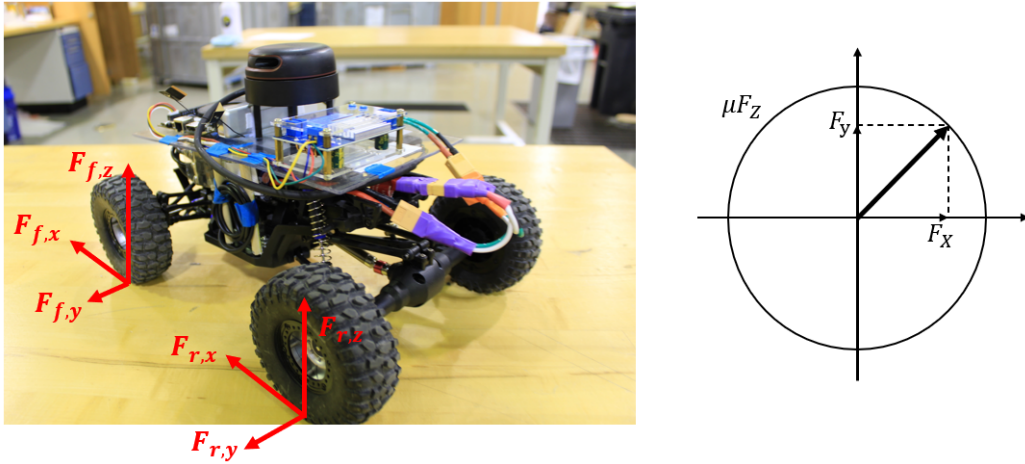


Figure 1.1: Tire friction of FITENTH robot. Left: free body diagram of vehicle tire force. Right: tire friction cycle constraint demonstration

When inspecting the tire friction cycles, it is not a difficult to tell that the friction of coefficient will decide the ultimate performance of the vehicle as it directly defines the upper limit of the tire force. The value of the coefficient of tire can range 0.2 up to 1.0 [27] depending on the weather and road condition.

Vehicle dynamical model is one of the important part in autonomous vehicle control design. The common vehicle dynamical model includes: kinematic single-track model, single-track bicycle model and multi-body model whose detail can be found from Matthias Althoff's work [3]. In our research, due the limitation of testing facilities, we use a 1/12th scale RC car to conduct the test. The roll effect of the RC car is ignored for the low center of gravity and track width, hence we will use single-track model for this research.

1.1.2 Vehicle controller

For the past decade, a couple of different types of vehicle controllers have been developed. These controller can be roughly categorized into two types: Model-free geometric controller and Model-based optimization controller.

1.1.3 Model Free Vehicle Controller

One well-known model-free geometric controller is pure pursuit [9] [35], which is proven to be effective and robust in low-speed and smooth-control scenario. The basic idea of pure pursuit controller is chasing a selected point from the desired trajectory based on kinematic single-track vehicle geometry. A detailed investigation of different improved approaches of pure pursuit will be shown in section 4.

Despite the low computational requirement and high robustness of pure pursuit controller, the disadvantage of pure pursuit is also obvious. One known issue is the performance and stability of pure pursuit can be largely compromised in high-speed due to the vehicle slip. Our research shows that pure pursuit controller's tracking accuracy can not be guaranteed in complex environment such as high speed or twisty trajectory.

Another popular model-free vehicle control method is using reinforcement learning to train a vehicle controller. Reinforcement learning is the combination of using Markov Decision Process and Deep Neural Network to find the highest probably control output given the real-time environment states and constraints.[33][22][34] This approach has attract much attention for the last few years with numerous works have been done on this topic. However, reinforcement learning controller requires high computation resource and long training time to obtain a steady controller. Also, it needs an highly accurate physics engine to obtain a applicable controller, otherwise the vehicle's performance may not be good enough. On the other hand, it is unsafe to train the reinforcement learning controller in real-world testing as the training process itself require random and noisy control inputs, which will lead to vehicle crash or dangerous move.

1.1.4 Model Based Vehicle Controller

In the light of these known issue of model-free controller, many researches have been devoted into model-based optimization approach. When it comes optimization-based approach, no other controller has been as popular as model predictive control(MPC). Shortly speaking, model predictive control, also known as receding horizon control, solving a local optimization problem aiming to minimize tracking error as well as control effort while satisfying both equality and inequality constraints. The basic setup for Model Predictive Control is shown in Eqn (1.2).

$$\begin{aligned}
 & \min_u (x_N - x_{ref,N})^T Q_f (x_N - x_{ref,N}) + \\
 & \quad \sum_{t=1}^{N-1} (x_t - x_{ref,t})^T Q (x_t - x_{ref,t}) + u_t^T R u_t \\
 & s.t. \quad x_{t+1} = f(x_t, u_t) \\
 & \quad x_{N,0} = x_0 \\
 & \quad u_{min} \leq u_t \leq u_{max} \\
 & \quad x_{min} \leq x_t \leq x_{max}, \quad t = 1, \dots, N
 \end{aligned} \tag{1.2}$$

In Eqn.(1.2), the MPC optimization setup consist of two major section: objective functions and variable constraints. Since the information used in the MPC is time-series data, the vehicle dynamics is explained by a state-space representation, which is the equality constraint $x_{t+1} = f(x_t, u_t)$. As mentioned in section 1.1.1, single-track vehicle models are used for designing MPC problem in this research. There are two type of single-track vehicle model: kinematic single-track model and dynamic single-track model, which will be discussed in detail in chapter 4.

1.1.5 Minimum time navigation strategy

The problem of generating the optimal time navigation line for vehicle around a closed race track has been studied over the last decade. From solving analytical solutions for a simple maneuvers using calculus of variations' theory to mimic professional driver's behavior to perform imitation learning.

Some of these methods can either be computational expensive or not close enough to optimal trajectory. For example, when using reinforcement learning for autonomous racing design, it is required to have a highly accurate vehicle dynamical models so that the vehicle is capable of driving on its limits. If using calculus of variation approach, it usually took long time to converge to a minimum value, also it depends on a highly accurate vehicle dynamical models. Therefore, an efficient and low computational cost global trajectory optimization approach remains an open research area.

1.2 Research Contribution and Outline

Section 1.1 provide a brief background of current research on autonomous racing including optimal trajectory planning as well as vehicle control. In this section, a brief summary about the contribution of this master thesis will be provided.

1.2.1 Data-driven vehicle dynamic identification

In autonomous vehicle's controller design, the challenge of obtaining highly accurate real-time vehicle dynamical model still exist, especially when essential vehicle parameters, such as cornering stiffness and yaw moment of inertia, are difficult to obtain. As mentioned in the section 1.1.4, the accuracy of vehicle dynamical model is essential to overall performance of Model Predictive Control, which emphasize the importance of

accurate system identification process in vehicle control problem.

The first contribution of our research is developing a detailed framework on learning the vehicle dynamical system in the form of state-space representation based on the recorded vehicle's data only by using koopman operator theory as well as deep neural network. Our newly developed data-driven system identification technique has proven to be able to construct a highly accuracy dynamical model in the form of state-space update. The learned dynamics is able to help improving the performance of Model Predictive Control, especially when the vehicle parameters are not fully known.

One known disadvantage of current data-driven system identification techniques is the potential high dimensions of identified model. Based on the theory of convex optimization, higher dimension and horizon can seriously compromise the computational efficiency, which may result in poor control performance in real-world application. In this paper, we incorporate deep neural network into designing koopman operator so that the approximated model's accuracy and number of lifted dimensions can achieve better balance through the training process. With our deep koopman data-driven system identification technique, we are able to learn the vehicle dynamics accurately without leading to excessive high lifted dimensions, which allow us to implement the deep koopman data-driven control on a real ackermann steering robot(1/12th scale RC car) with a satisfying control performance.

1.2.2 Fast global racing line generation: An iterative approach to optimize global trajectory

The second contribution of this thesis study is a new iterative approach to generate a optimized global trajectory aiming to achieve the minimum lap-time. This new iterative approach is able to optimize the global vehicle trajectory for a lower lap-time within less than 10 iterations. Combined with dynamic programming approach in generating optimal

velocity profile, we are able to obtain a high speed global trajectory close to the tire's performance limit, which facilitate the testing of different vehicle controller's performance under high speed and sharp steering. The detail of our global trajectory generation framework will be discussed in detail in section Chapter 2.

Chapter 2

Minimum-time global trajectory optimization

In most grand-prix style racing scenario, all vehicle consistently navigate near the tire friction limit. Therefore, a global reference trajectory that can satisfy both low lap time objective and maximum tire force constraint is needed to test the vehicle's control performance.

In this section, we will briefly introduce a global reference trajectory optimization methods. Based on our experiment shown in Chapter 2, the optimized trajectory is not only much faster than initial reference trajectory, but also contain enough driving behaviors to test the vehicle's racing control performance.

2.1 Minimum-time navigation within tire friction limit

When constructing the low lap-time racing line optimization problem, we separate the optimization into two steps: 1. Iterative approach to optimize global pose of reference trajectory aiming towards high smoothness and short distance. 2. Use dynamic pro-

gramming approach to generate optimal velocity profile based on optimized global pose trajectory.

Consider a predefined trajectory that is represented by an array of the first-order state variables such as vehicle positions and orientations in 2D plane (x_t, y_t, θ_t) , then time cost of the objective function can be approximated as shown in Eqn.(2.1).

$$t = \int_0^i \frac{ds}{U_x(s)} \quad (2.1)$$

Therefore, the variable of interest to solve minimum-time navigation problem is the velocity U_s at each pose on the given global trajectory. If the length of curve ds is sufficiently small, it can be approximated as the euclidean distance between two positions. In this case, the overall time cost can be written in discrete form as shown in Eqn.(2.2)

$$t = \sum_{i=0}^{i=t} \frac{\|x_{i+1} - x_i, y_{i+1} - y_i\|_2^2}{\bar{U}_x(s)} \quad (2.2)$$

where $\bar{U}_x(s)$ is the average velocity assigned at two adjacent poses.

2.1.1 Optimal velocity profile generation: Dynamic Programming Approach

As suggested in the work [3] by M. Althoff, since we use single-track model in this research, tire-friction cycle is selected as the maximum tire-force constraint when designing our minimum-time navigation problem. With the objective function as depicted by

Eqn.(2.2), the detail of dynamic programming optimization is shown in (2.3).

$$\begin{aligned}
\min_{u_0, \dots, u_t} \quad & \sum_{i=0}^{i=t-1} \frac{2\|x_{i+1} - x_i, y_{i+1} - y_i\|_2^2}{u_{i+1} + u_i} \\
\bar{v}_i = \quad & \frac{u_{i+1} + u_i}{2} \\
dt_i = \quad & \frac{\|x_{i+1} - x_i, y_{i+1} - y_i\|_2^2}{\bar{v}_i} \\
\bar{w}_i = \quad & \frac{|\theta_{i+1} - \theta_i|}{dt_i} \\
a_i = \quad & \frac{u_{i+1} - u_i}{dt_i} \\
\sqrt{a_i^2 + \bar{w}_i \cdot \bar{v}_i} \leq \quad & \mu
\end{aligned} \tag{2.3}$$

Notice that the dynamics programming construction above is only valid with a pre-defined trajectory where the position and orientation of the vehicle are already known. Therefore, the first step of optimization is to obtain a smooth and short trajectory, which will be introduced in the next section.

It is not difficult to conclude that the friction coefficient μ between tire and road directly affect the amount of traction force available for the vehicle to move. With a higher traction force, the vehicle can change speed and orientation at a higher rate. However, the aggressive driving behavior motivated by high traction force may also led to dangerous consequences such as crashes or flip. In our onboard scaled RC car application, due to the concern of the safety of the vehicle, we won't choose the value of μ to be higher than 0.5 to avoid the potential crash.

2.1.2 Iterative approach to optimize vehicle trajectory

In order to test the controller's performance in high-speed racing scenario, the reference global trajectory for tracking has to be as fast as possible. To generate a high-speed

and low lap-time trajectory, we apply an iterative approach to find a smooth and short global pose trajectory. At first, we define a initial global trajectory χ_0 as the reference trajectory, which can either be human driver's input or just the middle line of the track. Then we perform the racing line iteration by recording the vehicle pose based on the solution a real-time local planning optimization.

Enlightened by the work from N.Kapania in his Ph.D. thesis of Stanford University [20] and the work on learning-based MPC by Ugo.Rosolia [30][31], our local planning optimization is the real-time solution of the optimization problem as shown in Eqn. (2.6). The optimization is based on the 2D pose of robot $s_t = (x_t, y_t, \theta_t)$ where (x_t, y_t) represents the position and θ_t represents the orientation. In this case, the goal is to find the local path that satisfy the combination of two terms: 1. the shortest path between real-time vehicle pose and local goal pose selected from the racing line 2. the overall curvature of the local path.

Since the trajectory is designed for vehicle navigation, we incorporate the non-holonomic constraints for pose update as suggested in the trajectory optimization paper [2][32]. Shortly speaking , when setting up state update equations, input variable is the change of orientation $\delta\theta$ as well as the curve length δS , and state s_t 's update can be written as shown in Eqn.(2.4)

$$\begin{aligned}
 x_{t+1} &= x_t + \frac{\delta S}{\delta\theta} \cdot (\sin(\delta\theta) \cdot \cos(\theta_t) - (1 - \cos(\delta\theta)) \cdot \sin(\theta_t)) \\
 y_{t+1} &= y_t + \frac{\delta S}{\delta\theta} \cdot (\sin(\delta\theta) \cdot \sin(\theta_t) + (1 - \cos(\delta\theta)) \cdot \cos(\theta_t)) \\
 \theta_{t+1} &= \theta_t + \delta\theta
 \end{aligned} \tag{2.4}$$

A simple demonstrative diagram of state update equation h is shown in Fig.(2.1). More detail of the pose update process can be found in the work [32][29] by C.Rosmann.

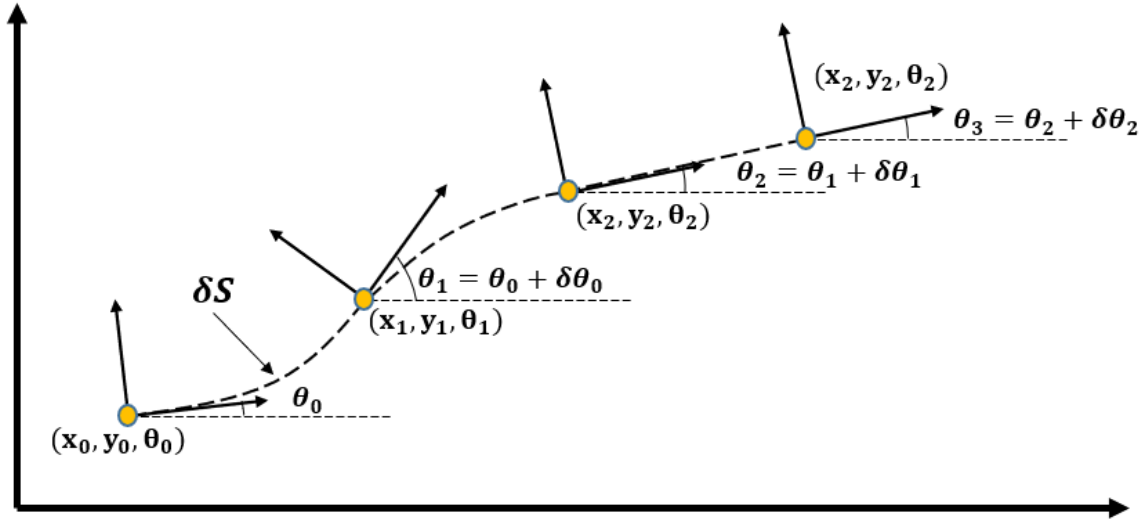


Figure 2.1: State update demonstration of local trajectory optimization

Generally speaking, with a smaller value of δS , the local trajectory can be smoother, but computational cost will be higher. Since only the first few poses from the local planner will be used to generate the control signal, therefore choosing a more sparse interval between poses is acceptable to generate a good global trajectory. On the other hand, the iterative process is recording the updated vehicle pose from local planning solution, not all solutions of the local planner.

For simplicity purpose, the consecutive pose update is written as a function h (as shown in Eqn.(2.4) that update the current pose s_t based on two control input $\delta\theta$ and δS .

$$s_{t+1} = h(s_t, \delta\theta_t, \delta S_t) \quad (2.5)$$

where $\delta\theta$ is the change of orientation and δS is the length of curve between two curves.

With the state update equation and initial vehicle pose information, the local plan-

ning problem is to minimize the cost function with two terms as suggested in (2.6).

$$\begin{aligned}
& \min_{\delta\theta_0, \delta\theta_1, \dots, \delta\theta_{t-1}} \sum_{i=0}^{i=t-1} (1 - w_d) \cdot \frac{|\delta\theta_i|}{\delta S} + w_d \cdot \|x_i - x_g, y_i - y_g\|_2^2 \\
& \mathbf{s.t.} \quad s_{i+1} = h(s_i, \delta\theta, \delta S) \\
& \quad \quad s_i \notin \chi_{obstacle} \\
& \quad \quad s_g \in \chi_j
\end{aligned} \tag{2.6}$$

where s_g is the local goal pose selected from global trajectory χ_j on j th lap, δS represent the arc length two adjacent poses. $\chi_{obstacle}$ represents the unacceptable region on the track that can either be the stationary obstacles, walls or the rival vehicles.

The minimum distance weight w_d control the balance between minimum curvature and minimum distance for the optimization. The output of the optimization problem in Eqn.(2.6) is the an array of the vehicle pose that contains the information of vehicle's position and orientation. Based on our experiment, it took around 4-5 laps for the global racing line to converge.

2.2 Trade-off between trajectory's curvature and distance

The factors that affect the minimum lap-time include more than just the speed the vehicle and distance it travels, the maximum tire force of the vehicle is also an important factor. In the light of this situation, the curvature of the global racing should also be thoroughly analyzed when generating global trajectory.

In general, a lower curvature path is equivalent to a smoother change of vehicle orientation along the path. Based on the last constraint of the optimization problem (2.3), the norm of longitudinal acceleration and angular acceleration must be lower than the road-tire friction factor μ . Therefore, with a lower change of yaw angle, the angular velocity

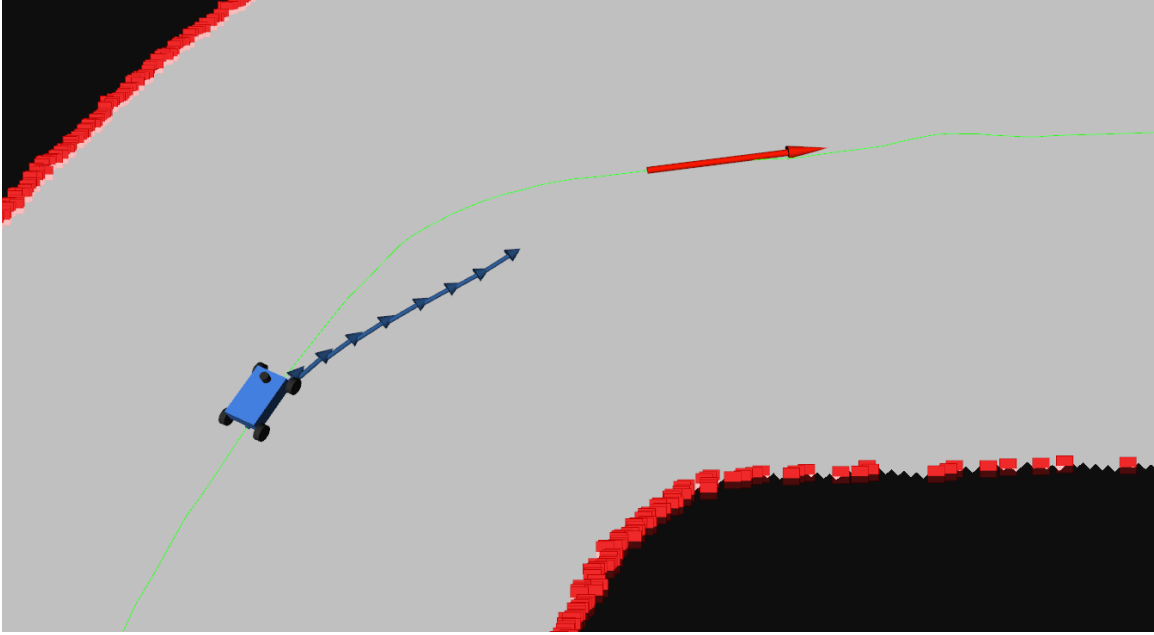


Figure 2.2: Iterative Real-time Local Planning Demonstration

w_i will be smaller, which spares more room for longitudinal velocity v_i and acceleration a_i . With this conclusion, despite the fact that lower curvature may lead to longer distance travelled, the velocity gain from low curvature path can potentially lead to a lower lap-time.

However, the optimal value of w_d in Eqn.(2.6) may vary from track to track as different tracks have completely different characters. Also, the road-tire friction factor may also affect the optimal value of w_d as the high grip road could indulge more aggressive driving behavior, which will eventually benefit the low distance navigation strategy. Therefore, when generating the optimal racing-line, both tire-road friction μ and minimum distance weight w_d should be analyzed at the same time. A detail analysis is shown in section 5.1.1.

Chapter 3

Vehicle control on the limit of tire friction

3.1 State-of-the-Art vehicle controller

In most autonomous driving control problems, the main purpose of vehicle controller is to track the optimized trajectory. In this section, we will give a brief introduction of some popular state-of-art vehicle controllers.

3.1.1 Model-free geometric vehicle controller

As depicted in its definition, model-free geometric vehicle controller doesn't need sophisticated vehicle parameters to generate vehicle control output. Most model-free controller will dissect the vehicle control into two parts: longitudinal control and lateral control.

Assuming the controller's goal is to track the desired trajectory with 4 states: $(x_t, y_t, v_t, \theta_t)$ where x_t, y_t and θ_t represent the vehicle 2D pose, v_t represent the desired longitudinal velocity. In this case, the longitudinal control is to generate acceleration signal to

drive the vehicle's velocity close to the desired velocity, which can easily be achieved by applying simple PID control as shown in Eqn. (3.1).

$$u_t = K_p \cdot \delta_t + K_d \cdot \dot{\delta}_t + K_I \cdot \int \delta_t \quad (3.1)$$

In Eqn. (3.1), δ_t represent the error value between current state and the desired state.

For lateral control, the philosophy of most model-free methods is to eliminate lateral error between vehicle's position and reference trajectory by generating the steering angle signal. Two most popular model-free lateral controllers will be demonstrated in this section: Pure pursuit and Stanley controller.

3.1.1.1 Pure Pursuit Controller and its Modification

Pure pursuit controller uses single-track kinematic model to calculate required steering angle to reach a reference pose. The reference pose is selected based on the distance between vehicle's real-time pose and desired pose from reference trajectory. Since pure pursuit assumes zero-slip of the tire, the only vehicle parameter needed for computing steering is wheelbase of the car h . In this case, the steering angle α can be calculated as shown in Eqn.(3.1).

$$\delta = \tan^{-1}\left(\frac{2 \cdot h \cdot \sin(\alpha)}{L}\right) \quad (3.2)$$

where δ is the steering angle, h represent the wheelbase of the vehicle and L represent the distance between current vehicle state and goal state selected from the reference trajectory. A visual demonstration is shown in Fig. (3.1). Despite its high computing efficiency and robustness, the disadvantage of pure pursuit is also obvious. Firstly, since pure pursuit only tries to eliminate lateral error without considering the yaw angle tracking, steering overshoot is difficult to avoid. Secondly, pure pursuit doesn't take the dynamical effect of the

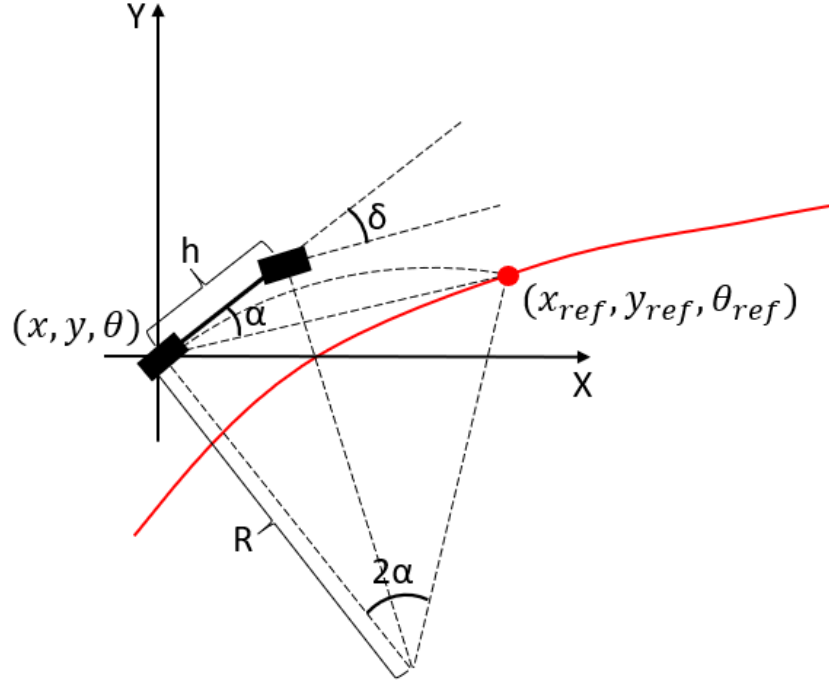


Figure 3.1: Pure Pursuit Controller Demonstration

vehicle into consideration, hence its tracking performance will be terrible when encountering sudden directional change as well as steering on a low-grip road.

To accommodate these issues, several modified versions of pure pursuit have been developed. Intuitively speaking, a negative relationship exist between look-ahead distance and settling time of the steering control, which means in simple driving scenario such as straight line or low-curvature corner, a longer look-ahead distance is preferable. Similarly, when the reference trajectory is twisty, a short look-ahead distance can help reduce tracking error. Therefore, one common modification is build a relationship between look-ahead distance and total yaw angle change of the planned path.

Another common modification is constructing a function between look-ahead distance and current vehicle velocity. This modification is intended to prevent abrupt and harsh driving behavior when vehicle is navigating at high speed, hence increase controller's safety. However, such modification may lead to poor tracking performance in hairpin cor-

ner. The analysis of these modifications will be discussed in detail in section 5.1.

3.1.1.2 Stanley Path Tracking Controller

Besides pure pursuit controller, another popular geometrical lateral controller is "Stanley" controller, which was first implemented in DARPA grand challenge by Stanford racing team in 2005.[17] Unlike pure pursuit whose steering geometry is designed based on rear axle, Stanley controller's steering geometry is designed based on front axle. The control design of Stanley controller is shown in Eqn.(3.3)

$$\delta_t = \begin{cases} \Psi_t + \arctan\left(\frac{k \cdot e_t}{v_t}\right) & |\Psi_t + \arctan\left(\frac{k \cdot e_t}{v_t}\right)| < \delta_{max} \\ \delta_{max} & \Psi_t + \arctan\left(\frac{k \cdot e_t}{v_t}\right) \geq \delta_{max} \\ -\delta_{max} & \Psi_t + \arctan\left(\frac{k \cdot e_t}{v_t}\right) \leq -\delta_{max} \end{cases} \quad (3.3)$$

where δ_t is the steering angle and δ_{max} represents the maximum steering angle of the vehicle. Ψ_t is the vehicle's heading error error with respect to the reference pose's heading direction while e_t is the cross-track error between vehicle front axle and the reference trajectory. v_t is the real-time velocity of the vehicle. Notice that the cross-track error e_t is scaled by a factor k in Eqn.(3.3). By tuning the factor k , the balance between rise-time and overshoot can be optimized in different driving scenario. The detail of Stanley controller can be found in Stanford's paper [17].

3.1.2 Model-based optimization vehicle controller

Despite its robustness, efficiency and fast update frequency, model-free controller also has obvious disadvantage. For example, poor tracking performance usually occurs for model-free controller when vehicle navigating on low-friction road or tracking twisty high-curvature corner like hairpin corner. Such disadvantage is caused by the lack of information

from the vehicle dynamics, which lead to unpredictable behavior from the controller. Thus, along with the massive improvement of the computing power, the advantage of model-based optimization vehicle controller has attracted increasing attentions.

Two most common types of vehicle model in model predictive control design are kinematic single-track model(kinematic bicycle model) and single-track model(dynamic bicycle model). As depicted in section 1.1.4, the vehicle model is commonly used as the equality constraint of the MPC convex optimization problem. In order to construct the MPC problem properly, the reference trajectory s_{ref} is needed. For different applications and scenarios, the number of state in s_{ref} may also be modified accordingly. Generally speaking, when using kinematic single track model, four states are needed when designing MPC for path tracking (x_t, y_t, v_t, ψ_t) . When using dynamic single-track model, two more states needs to be added, one is yaw rate $\dot{\psi}_t$ and the other is slip angle β_t .

3.1.3 Two Types of Single Track Model

To further elaborate how the state's information updates with certain control input, we will describe two types of single-track model in details. Based on the amazing work by Matthias Althoff on vehicle models[3], the state-space representation of kinematic bicycle model can be described as shown in Eqn.(3.4).

$$\begin{aligned}
 \dot{x} &= v \cdot \cos(\psi) \\
 \dot{y} &= v \cdot \sin(\psi) \\
 \dot{\psi} &= \frac{v}{l_{wb}} \cdot \tan(\delta) \\
 \dot{v} &= a \\
 \dot{\delta} &= v_{\delta}
 \end{aligned} \tag{3.4}$$

where (x_t, y_t, ψ_t) represent the position and orientation information of the vehicle on 2D plan while v_t represent the longitudinal velocity. Notes that there is more than one type of control input for selection when designing MPC, the most commonly used one is the combination of acceleration a and steering angle δ . In this case, the last equation of Eqn.(3.4) is no longer needed, yet a hard constraint of steering angle velocity is needed, which can be described as an inequality constraint on the rate of change of steering angle input.

Another potential control input selection is the combination of acceleration a and steering angle velocity v_δ . Then, the steering angle velocity can be written in the objective function as control effort cost rather than a hard constraint.

Given the fact that kinematic single-track model is based on the assumption of zero-slip, its performance can be compromised when tire-slip become a dominate factors. For example, when vehicle is navigating in slippery condition or close to its physically limit, understeer and oversteer are very likely to occur and these behaviors can not be described by kinematic single-track model. Therefore, dynamic single-track model is introduced and

shown in Eqn.(3.5)[3]

$$\begin{aligned}
\dot{x} &= v \cdot \cos(\beta + \Psi) \\
\dot{y} &= v \cdot \sin(\beta + \Psi) \\
\dot{v} &= a \\
\dot{\delta} &= v_\delta \\
\dot{\beta} &= \frac{\mu}{v(l_r + l_f)} \cdot (C_f(gl_f - ah_{cg})\delta - (C_r(gl_f + ah_{cg}) + C_f(gl_r - ah_{cg}))\beta) \\
&\quad + (C_r(gl_f + ah_{cg})l_r - C_f(gl_r - ah_{cg})l_f)\frac{\dot{\Psi}}{v} - \dot{\Psi} \\
\ddot{\Psi} &= \frac{\mu m}{I_z(l_r + l_f)}(l_f C_f(gl_r - ah_{cg})\delta \\
&\quad + (l_r C_r(gl_f + ah_{cg}) - l_f C_f(gl_r - ah_{cg}))\beta \\
&\quad - (l_f^2 C_f(gl_r - ah_{cg}) + l_r^2 C_r(gl_f + ah_{cg}))\frac{\dot{\Psi}}{v})
\end{aligned} \tag{3.5}$$

where two extra states are added to kinematic bicycle model: slip angle β_t and angular acceleration $\ddot{\Psi}$. Since the main focus of this thesis is not on derivation of single-track model, the detail of dynamic single-track model and kinematic single-track model can be found in [3][20][14][28].

3.2 Data-driven optimization-based vehicle controller

Based on the work from Dr.Borrelli in his research on different single-track model's performance in autonomous driving[12][23], the performance of dynamic single-track model is good enough to cope with a variety of road condition. However, dynamic single-track model requires numerous vehicle parameters to fully describe its behavior, including cornering stiffness, position of the center of gravity, moment of inertia in vertical direction. Also, it requires more states measurement compared to kinematic model. All of these can

lead to a high cost when designing autonomous vehicle controller, which become the main motivation of developing a data-driven system identification method to learn the vehicle dynamics without knowing the full parameters of the vehicle.

3.2.1 System identification of nonlinear vehicle dynamics

The main objective of system identification of the vehicle dynamics is to properly approximated a state-space model of the vehicles' state and control based on the time-series information. In this case, we incorporate koopman operator theory to fulfill system identification task.

Based on the accomplished work of koopman operator [38] [7] [24][18], the performance of system identification is based on three major factors: time series data, observable function and koopman operator \mathcal{K} . Consider a controlled dynamical system as in equation (3.6).

$$x_{t+1} = f(x_t, u_t) \quad (3.6)$$

where $x_t \in \mathbb{R}^n$. Assume we have a recorded time-series of states $X = [x_0, x_1, x_2, \dots, x_{k+1}]$ and corresponding control sequence $U = [u_0, u_1, u_2, \dots, u_k]$ that drive the evolution of X . Then the koopman operator \mathcal{K} is an infinite-dimension operator that build linear system representation of dynamical system (3.6) based on the observable space $g(x_t)$ [37] [19] [13][7] as shown in equation (3.7).

$$\mathcal{K}g(x_t) = g(f(x_t, u_t)) \quad (3.7)$$

where $g(x_t)$ lift the original space \mathbb{R}^n to a higher dimension \mathbb{R}^m . The main objective of koopman operator theory is to find \mathcal{K} that best match the relationship as shown in Eqn. (3.7) based on the recorded state and control sequence X and U .

As infinite dimensional mapping \mathcal{K} is impossible to obtained in real-applications, the koopman operator has to be approximated with a predefined number of observation functions. We define a set of basis functions $\psi_1(x_t), \psi_2(x_t), \dots, \psi_N(x_t)$. Let $z_t = [\psi_1(x_t), \psi_2(x_t), \dots, \psi_N(x_t)]^T$ with $N \gg n$, then the approximation of koopman operator \mathcal{K} is to find solution to the optimization problem (3.8).[19] [18]

$$\min_{\mathcal{K}} \sum_{t=0}^t \left\| \begin{bmatrix} z_{t+1} \\ u_t \end{bmatrix} - \mathcal{K} \begin{bmatrix} z_t \\ u_t \end{bmatrix} \right\|_2^2 \quad (3.8)$$

As depicted in the paper [24], the optimization solution \mathcal{K} can be decomposed as $\mathcal{K} = [A, B]$, which is equivalent to solving the optimization problem (3.9).

$$\min_{A,B} \sum_{t=1}^K \|z_{t+1} - (A \cdot z_t + B \cdot u_t)\|_2^2 \quad (3.9)$$

A, B is the state-space representation of the dynamical system after being transformed by nonlinear observable function ψ . To create mapping between lifted state z_t and original state x_t , the matrix C is introduced as the solution to the optimization problem (3.10).

$$\min_C \sum_{t=1}^K \|X_t - C \cdot \psi(x_t)\|_2^2 \quad (3.10)$$

The analytical solution to (3.9) and (3.10) can be calculated as shown in Eqn. (3.11)

$$\begin{aligned} \begin{bmatrix} A, B \end{bmatrix} &= z_{t+1}[z_t, U]^\dagger \\ C &= z_t x_t^\dagger \end{aligned} \quad (3.11)$$

where \dagger denotes the Moore-Penrose pseudoinverse of matrix. In practice, when the number of collected data set is significantly larger than the dimensions of observable function (N

$\ll k$), it is more desirable to solve optimization problem (3.9) using Eqn. (3.12) than Eqn.(3.11). [24]

$$\begin{bmatrix} A, B \end{bmatrix} = z_{t+1} \begin{bmatrix} z_t \\ U \end{bmatrix} \left(\begin{bmatrix} z_t \\ U \end{bmatrix}^T \begin{bmatrix} z_t \\ U \end{bmatrix} \right)^\dagger \quad (3.12)$$

3.2.2 Choosing the appropriate observable function

As shown in equation 3.8, the performance of system identification rely heavily on the choice of basis function ψ . A set of appropriate basis functions can achieve accurate system identification without pushing the number of dimensions of observable space too much, which is essential when designing optimization based controller like Model Predictive Control(MPC) as high dimensions can easily lead to poor computation efficiency in convex optimization.

In general, what affects the performance of nonlinear system identification depends on both on the dynamical system's feature as well as the sampling methods to obtain the system's data. In this section , we will focus on the choice of different basis functions. Based on the previous research on koopman operator theory, most of the basis function is based on a predefined nonlinear function including polynomial basis function, thinplate basis function and Gaussian basis function.[24][10][6] In this research, we mainly use polynomial basis function and deep neural network basis function as koopman operator.

3.2.2.1 Polynomial Basis Function

In this case, we create a polynomial-based kernel library $\Theta(X)$ and the corresponding coefficient matrix Ξ . (3.13).[8][26][7]

$$\Theta(X, U) = \begin{bmatrix} | & | & | & | \\ 1 & X_t & X_t^2 & - U \\ | & | & | & | \end{bmatrix} \quad (3.13)$$

$$\Xi = \begin{bmatrix} | & | & | & | \\ \xi_1 & \xi_2 & - & \xi_N \\ | & | & | & | \end{bmatrix}$$

where X and U are the sequence of states and controls. Each column vector ξ in Ξ represents the coefficient of the polynomials that constructed by the row vectors from the kernel-library $\Theta(X)$.

Therefore, the objective of polynomial approximation is to find the solution Ξ to the optimization problem (3.14).

$$\underset{\Xi}{\operatorname{argmin}} \|X_{t+1} - \Xi\Theta^T(X_t, U_t)\|_2 \quad (3.14)$$

The optimal solution to Eqn. (3.14) can be calculated as shown in Eqn. (3.15).

$$\Xi = X_{t+1} \cdot (\Theta^T(X_t, U_t))^\dagger \quad (3.15)$$

In this case, the approximated model is still on the original state space since there is no observable function has been utilized when solving Eqn. (3.14). For the polynomial basis function, if the dynamical system is not highly nonlinear, using polynomial basis function alone will be sufficient for optimal control application. However, when the dynamical

system is highly nonlinear, using polynomial basis function alone may not be able to deliver stable and efficient control. In this case, a more nonlinear observable function is needed to perform state-space transformation.

3.2.2.2 Artificial Neural Network Basis Function

If the predefined basis functions fail to satisfy the accuracy and efficiency requirement for vehicle dynamical system identification, we can also use artificial neural network to learn the best observable function.[16][15] Similar as using predefined observable func-

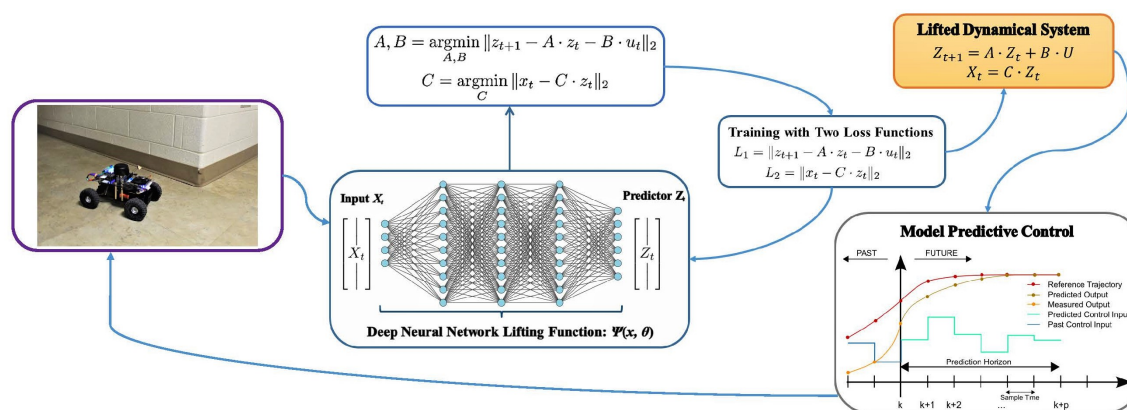


Figure 3.2: Deep Koopman Data-Driven Optimal Control Framework

tion, the main objective of using observable function is attempting to find a linear time invariant system that can accurately represent the system’s dynamics over the vector space of choice. Therefore, when designing the deep neural network as the observable function, the system loss is the same as the optimization problem as shown in 3.9. To guarantee the robustness of the training procedure, we replace the L2 norm loss with the combination of pseudo-huber loss and L1 loss as the final loss function for training the deep neural network observable function [5]. The final loss function is shown in equation [3.16] where $\Psi(x_t)$ is

the state output of deep neural network observable function.

$$L_a = \frac{1}{L-1} \sum_n^{L-1} |\psi(x_{t+1}) - (A \cdot \psi(x_t) + B \cdot u_t)| \quad (3.16)$$

$$L(\theta) = \delta^2 \left(\sqrt{1 + \left(\frac{L_a}{\delta}\right)^2} - 1 \right)$$

3.2.2.3 Augmented(Hybrid) basis function

Even though using DNN as lifting function can deliver accurate approximate with a low-dimension system, its performance is sensitive to the initial condition of the neural network. On the other hand, using DNN as observable function will blur the information from original states, which compromises the explainability of the lifted dynamical system hence adding extra difficulties when designing cost functions and constraints in MPC.

In this section, we proposed a hybrid lifting function that combines the neural network and polynomial basis function as a single lifting function. In this case, we are able to design the cost function and constraints based on the zero-order and first-order term while using DNN's output states as the optimizer to increase tracking accuracy. The training diagram is shown in Fig. 3.3.

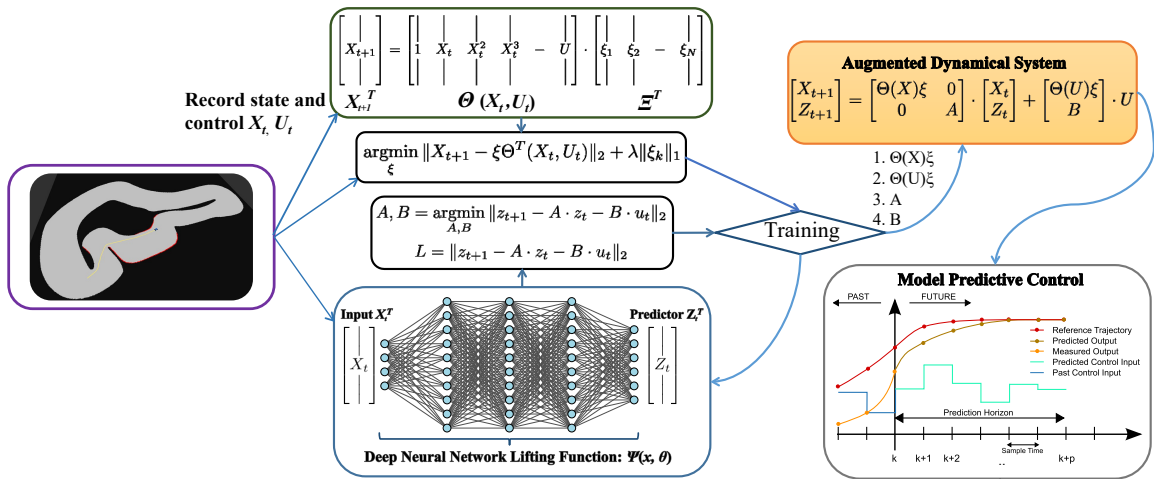


Figure 3.3: Vehicle dynamics identification based on hybrid observable function

Compared to deep koopman representation of control, the major difference of augmented basis function is to incorporate polynomial basis function into the training process of deep neural network observable functions. Therefore, the dimension of the neural network's part of the observable function no longer has to be as high as using DNN alone.

Algorithm 1 Training Algorithm of Artificial Neural Network Observable Function

1: **Input**

Recorded state X_t and the corresponding control U_t , Kernel Function Set $\Theta(X, U)$, Initialized Neural Network $\Psi(x, \theta)$

2: **Initialize state-space representation [A, B] from DNN basis function**

$$z_t = \psi(x_t, \theta), z_{t+1} = \psi(x_{t+1}, \theta)$$

$$[A, B] = z_{t+1} \begin{bmatrix} z_t \\ U \end{bmatrix} \left(\begin{bmatrix} z_t \\ U \end{bmatrix}^T \begin{bmatrix} z_t \\ U \end{bmatrix} \right)^\dagger$$

3: **Calculate initial system loss**

$$L_a = \frac{1}{L-1} \sum_n^{L-1} |z_{t+1} - A \cdot z_t - B \cdot u_t|$$

$$L(\theta) = \delta^2 \left(\sqrt{1 + \left(\frac{L_a}{\delta}\right)^2} - 1 \right)$$

4: **Set termination criterion ϵ and learning rate ρ for neural network training**

5: **while** $L_\theta > \epsilon$ **do**

6: Update Deep Neural Network based on L_θ : $\psi \leftarrow \rho \frac{\partial L_\theta}{\partial \omega}$

7: Recalculate the updated state-space represent $[A, B]$: $z_t = \psi(x_t, \theta)$, $z_{t+1} = \psi(x_{t+1}, \theta)$

$$[A, B] = z_{t+1} \begin{bmatrix} z_t \\ U \end{bmatrix} \left(\begin{bmatrix} z_t \\ U \end{bmatrix}^T \begin{bmatrix} z_t \\ U \end{bmatrix} \right)^\dagger$$

8: Calculate new system loss $L(\theta)$

$$L_a = \frac{1}{L-1} \sum_n^{L-1} |z_{t+1} - A \cdot z_t - B \cdot u_t|$$

$$L(\theta) = \delta^2 \left(\sqrt{1 + \left(\frac{L_a}{\delta}\right)^2} - 1 \right)$$

9: **end while**

10: **Return** ψ, A, B

Unlike DNN basis function, polynomial basis function preserve the information from original state space to the observable state space, which can facilitate the design of more complex control behavior like obstacle avoidance and emergency brake. However, the approximated model from using polynomial basis function alone is far less accurate than using deep neural network alone due the limitation of predefined basis function. Furthermore, when the navigation task doesn't contain emergency behavior, using a local planner

and a deep koopman MPC is sufficient to satisfy most of the navigation task.

3.2.3 Optimization-based control over observable-space

The augmented system's state is the combination of first-order state X_t and the DNN lifted states Z_t . Since the model predictive control design is based on the augmented dynamical system, the quadratic cost on the states has to be augmented as well. In this case, we write the augmented state as $\Omega_t = [X_t, Z_t]^T$, then the model predictive control on the augmented system can be formed as shown in Eqn. (3.17)

$$\begin{aligned}
& \min_u (\Omega_N - \Omega_{ref,N})^T \begin{bmatrix} Q_{X,f} & 0 \\ 0 & C^T Q_{Z,f} C \end{bmatrix} (\Omega_N - \Omega_{ref,N}) + \\
& \sum_{t=1}^{N-1} (\Omega_t - \Omega_{ref,t})^T \begin{bmatrix} Q_X & 0 \\ 0 & C^T Q_Z C \end{bmatrix} (\Omega_t - \Omega_{ref,t}) \\
[h] \quad & + u_t^T \begin{bmatrix} R_X & 0 \\ 0 & R_Z \end{bmatrix} u_t \\
& s.t. \Omega_{t+1} = \begin{bmatrix} \Theta(X)\xi & 0 \\ 0 & A \end{bmatrix} \Omega_t + \begin{bmatrix} \Theta(U)\xi \\ B \end{bmatrix} u_t \\
& \Omega_{N,0} = \Omega_0 \\
& u_{min} \leq u_t \leq u_{max} \\
& X_{min} \leq X_t \leq X_{max}
\end{aligned} \tag{3.17}$$

The relative weights among $Q_{X,f}$, $Q_{Z,f}$, Q_X and Q_Z directly affect the tracking performance of the controller. In general, the value of $Q_{X,f}$ and Q_X are designed to be larger than $Q_{Z,f}$ and Q_Z as the polynomial basis function maintains the original states after approximation which facilitate the design of cost function and constraint in MPC. Conse-

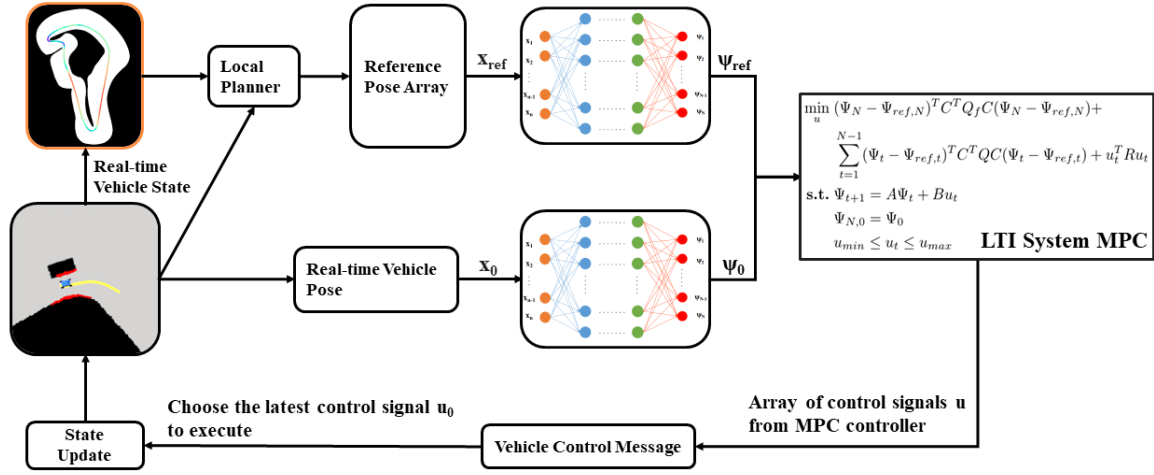


Figure 3.4: Control Flowchart over Lifted Space

quently, the tracking cost over DNN lifted states serve as the optimizer for the tracking cost of polynomial-function lifted state. In our case, the control cost term R_X and R_Z are set to be the same while both of them are smaller than state costs Q .

When the control task is not highly complicated, the information from the original state-space may not be necessary. As demonstrated in the work by Yiqi, Theresa and Dr.Borrelli [1], using two-step optimization can potentially deliver better control performance in low friction environment. In this case, using the combination of local planner and deep koopman MPC alone is sufficient enough for navigation tasks. The control diagram of using Deep Koopman MPC is shown in Fig.(3.4).

In this case, the MPC control solution are completely based on the lifted state-space, so that both reference pose and real-time have to be lifted into observable state-space through deep neural network first. In this case, the reference poses from local planner is the solution of the optimization problem 2.6 with collision-free constraints, which can be considered as the first step MPC solution. The second step MPC is to calculate control signal that satisfy the tracking error and control effort objectives. The execution of the control signal is the same as most of the MPC applications, only the first control signal will

be selected as control output and being published at a given frequency. The control output frequency specification can be found in Appendix C.

Chapter 4

Results Analysis

The experiment to analyze the performance of the data-driven vehicle controller as well as global trajectory optimization is consisted of two parts. The first part is conducted within F1tenth simulation environment which is built based on dynamic single track model [3.5] with state time delay interval of 0.01 seconds.

The second part is a real-vehicle test that include a scaled RC car as test robot and a closed area as the testing environment. In our case, the testing robot is 1/12 scaled with maximum speed of 7 m/s, the testing track is the basement of Clemson Fluor Daniel Engineering Innovation Building.

For both experiments, we measured the tracking performance of three different vehicle controllers for comparison: 1. adaptive pure pursuit 2. nonlinear model predictive control based on kinematic single-track model 3. Linear model predictive control based on deep koopman data-driven vehicle model.

4.1 Controller Specification

In this section, we will introduce some key features and modifications of all three controllers. These modifications are intended to extract the best performance of each controller for fair measurement.

Unlike the existing work on improving pure pursuit's performance[35][9], we attempted to build relationship between local planner's curvature change with look-ahead distance. Intuitively speaking, a shorter look-ahead distance can lead to higher tracking accuracy at the cost of control stability and vice versa. In this case, it is reasonable to build a negative relationship between path's curvature and look-ahead distance.

Due to safety concern of the high-speed navigation, low look-ahead distance not recommended in autonomous racing scenario. Therefore, we implement a nonlinear function that build relationship between look-ahead distance and vehicle speed as shown in Eqn.(4.1)

$$L_d = L_{max} - (L_{max} - L_{min}) \cdot \frac{2}{\pi} \cdot \arctan((\sum \delta\theta)^2) \quad (4.1)$$

where L_{max} and L_{min} refer to the longest and shortest look-ahead distance. $\delta\theta$ represents the sum of yaw angle changes of the local planner's path. For the sake of the overall stability of steering control outputs, we use inverse tangent function rather than linear function. In that case, only when vehicle navigate at highly twisty part of track will the controller generate harsh steering control.

For nonlinear kinematic MPC, the major concern is how to setup the single-track model's parameters. The first parameter of concern is the time discretization interval dt . Based on the research work [23] from Jason Kong and Dr. Borrelli, even dynamic single-track model has better performance compared to kinematic single-track model at low dt , yet the difference narrow down significantly at higher dt . Also, since computational efficiency is a key factor for real-vehicle controller design, it is preferable to use kinematic bicycle

model with high dt for this MPC design as it has lower number of states.

4.2 F1TENTH Simulator Experiment Result

To validate the robustness and efficiency of deep koopman data-driven MPC controller without damaging the experiment facility, we first conduct the test within the F1tenth simulation environment. Firstly, a high-speed global trajectory, also known as racing line, has to be generated as the control reference. Then we conduct the vehicle control comparison between three different controllers: pure pursuit, nonlinear MPC based on kinematic single track model, linear MPC based on deep koopman approximated vehicle model.

4.2.1 Global trajectory optimization result

Based on the approach introduced in chapter 2, the trajectory optimization started with an initial global trajectory from human operator's input. Then update the global pose trajectory based on the real-time solution of optimization 2.6 in section 2.2. The update progress can be visualized in Fig.(4.1). Finally, with the global pose trajectory converged, we compute the optimal velocity profile using dynamic programming technique. As shown in the right picture of Fig.(4.1), we compare the optimal velocity profile of initial global trajectory and the optimized global trajectory. The optimized global trajectory's velocity profile is not only much smoother compared to the initial trajectory, its velocity is also consistently higher, which lead to an immense lap-time improvement.

As explained in section 2.2, the best lap-time can be not achieved with minimum curvature or minimum distance alone, it need both of them. The balance between these two terms can direct impact the lap-time. In this case, we perform a detail analysis on lap-time based on different weights of minimum distance and minimum curvature. The result we obtain from the experiment in F1TENTH simulator is shown in table 4.1.

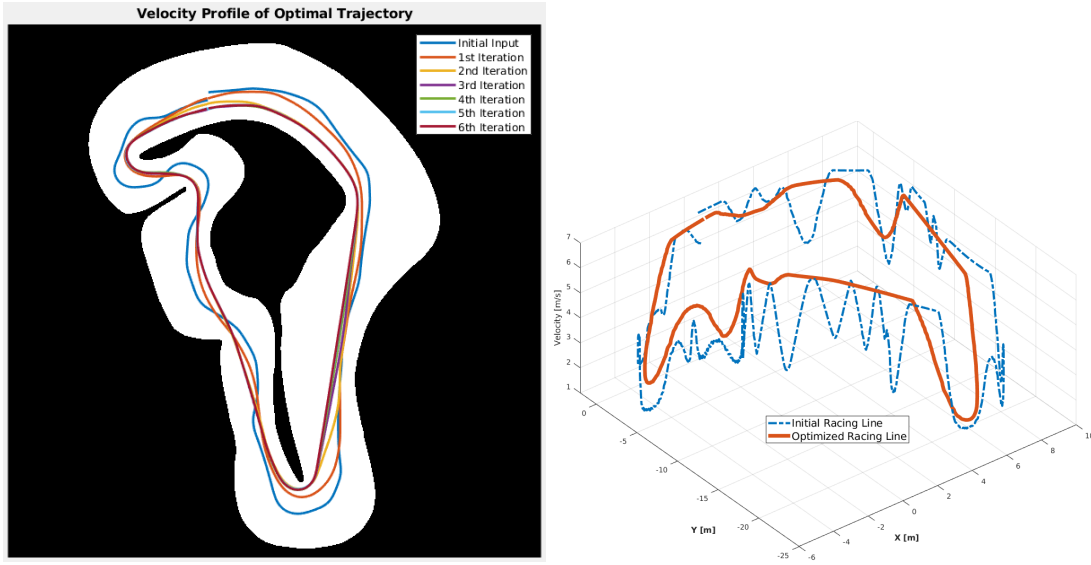


Figure 4.1: Global Trajectory Optimization Progress over a Closed Race Track

Table 4.1: Minimum Time Statistic over Different Road Friction and Minimum Distance Weight

	$\mu = 0.2$	$\mu = 0.3$	$\mu = 0.4$	$\mu = 0.5$	$\mu = 0.6$	$\mu = 0.7$	$\mu = 0.8$	$\mu = 0.9$
$w_d = 0.05$	19.18	15.6134	13.4987	12.0556	10.994	10.1829	9.5495	9.1014
$w_d = 0.2$	19.14	15.5798	13.4639	12.026	10.9656	10.1492	9.5318	9.0766
$w_d = 0.35$	19.19	15.605	13.49	12.0517	10.99	10.17	9.543	9.094
$w_d = 0.50$	19.1873	15.6342	13.4983	12.0573	10.9985	10.181	9.5533	9.1054
$w_d = 0.65$	19.147	15.5715	13.46	12.03	10.962	10.15	9.527	9.0956
$w_d = 0.8$	19.2392	15.6635	13.5218	12.0779	11.0146	10.2	9.5764	9.1387
$w_d = 0.95$	19.3256	15.7311	13.602	12.156	11.08	10.2555	9.6281	9.2004
$w_d = 1.0$	19.883	16.1719	13.9724	12.4789	11.3818	10.5328	9.8861	9.4607

Despite the effect of different tire-road friction of coefficient on the optimized lap-time, the best weight factor of minimum distance is around 0.2 and 0.65. Note that different tracks have different characteristic, which may lead to different optimal balance between minimum distance and minimum curvature. In our experiment, since the simulator's friction of coefficient is 0.7, we choose $w_d = 0.2$ as our minimum distance weight factor.

4.2.2 Path Tracking Error Analysis

The scenario we applied our data-driven MPC is the optimal tracking problem. The baseline models are nonlinear MPC and adaptive pure pursuit algorithms, which have been utilized in various autonomous driving community applications and the autonomous racing control for the F1TENTH environment. However, depending on the identification of the linearized model and the choice of ahead distance of the algorithm, those methods need a much more accurate model for the MPC control, and the control performance depends on the tuning of the model parameters. With the data-driven Koopman-based approach, the model can be identified in the high dimensional lifted space and hence provide more parameter robustness in the model identification.

More importantly, when the parameters of the vehicle dynamical system can not be explicitly obtained (center of gravity, cornering stiffness, racing track condition, etc.), data-driven system identification can capture the nonlinear feature of the system without direct knowledge of all parameters of the dynamical system. In our case, the nonlinear behavior of the vehicle dynamics can be properly captured through the identification process, which has proven to deliver much better control performances compared to pure pursuit or nonlinear kinematic MPC.

Since the assumption has been made that the vehicle dynamics are only partially known, parameters related to vehicle dynamics such as cornering stiffness, friction coefficient, vehicle mass are unavailable for designing the controller. The only information we can obtain from the vehicle is related to vehicle kinematics, such as wheelbase and track width. As shown in Fig. 4.2, due to the lack of information on vehicle dynamics parameters, the tracking performance of nonlinear kinematic MPC and adaptive pure pursuit is not as good as the data-driven MPC.

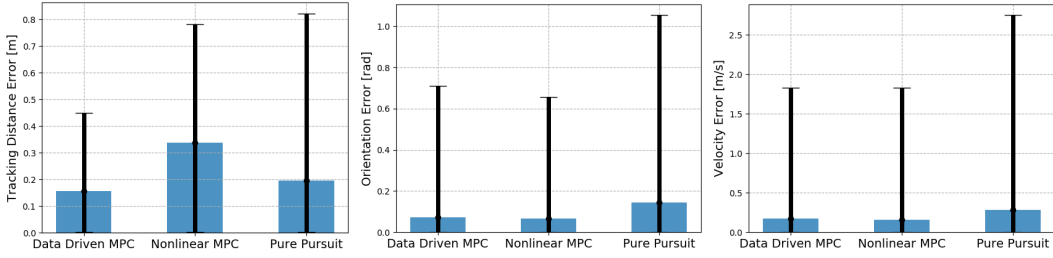


Figure 4.2: Tracking Error Comparison, blue bar represents the mean error value while black bar shows the range of error

Table 4.2: Tracking Performance Comparison Statistic

	Position Error [m]	Yaw Error [rad]	Speed Error [m/s]
Data-driven MPC	0.155	0.074	0.177
Nonlinear MPC	0.338	0.066	0.161
Pure Pursuit + PID	0.196	0.143	0.279

In Table (4.2) and Figure (4.3), the error represents the state difference between real-time vehicle pose and its closet reference pose. Data-driven MPC controller has the best performance compared to the other two controllers. In general, using geometric controller like pure pursuit achieve better position tracking performance, yet this method suffers from the unstable control output and large tracking error in sharp trajectory change. The Kinematic NMPC controller has better performance in tracking trajectory orientation and velocity since yaw angle and velocity has assigned the same cost as position states x and y . As mentioned before, data-driven MPC uses the approximated model based on recorded state and data. Consequently, the vehicle dynamics was capture in the approximation process, which gave the data-driven MPC higher position tracking accuracy than pure pursuit while maintaining the high yaw angle and velocity tracking performance from optimization-based control method.

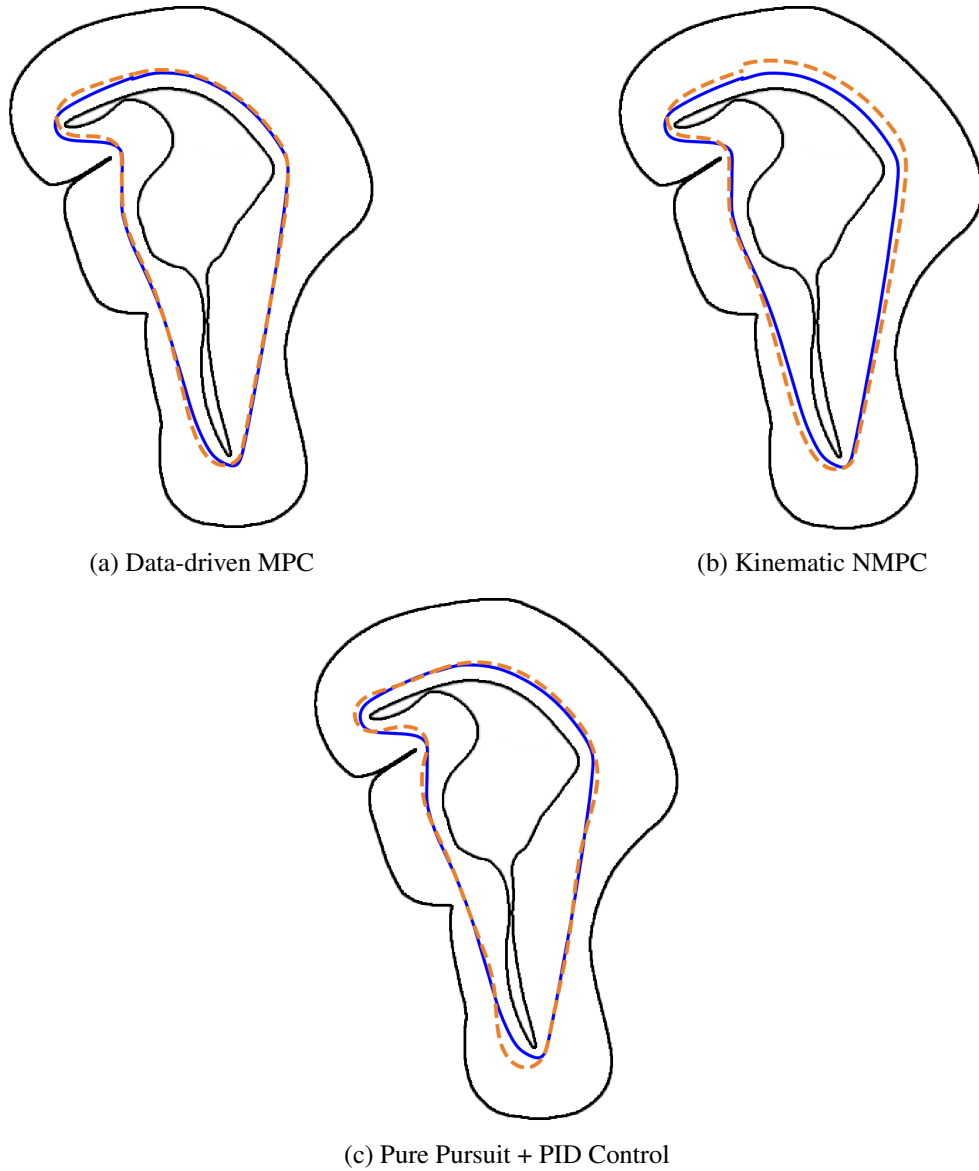


Figure 4.3: Trajectory tracking of three different vehicle controllers. (a): Data-driven MPC (b): Kinematic NMPC (c): Pure Pursuit + PID

4.2.3 Control Output Analysis

The control output record of these three different controllers are plotted in the same graph as shown in figure 4.4. The controllers' parameters detail are listed in the Appendix C.

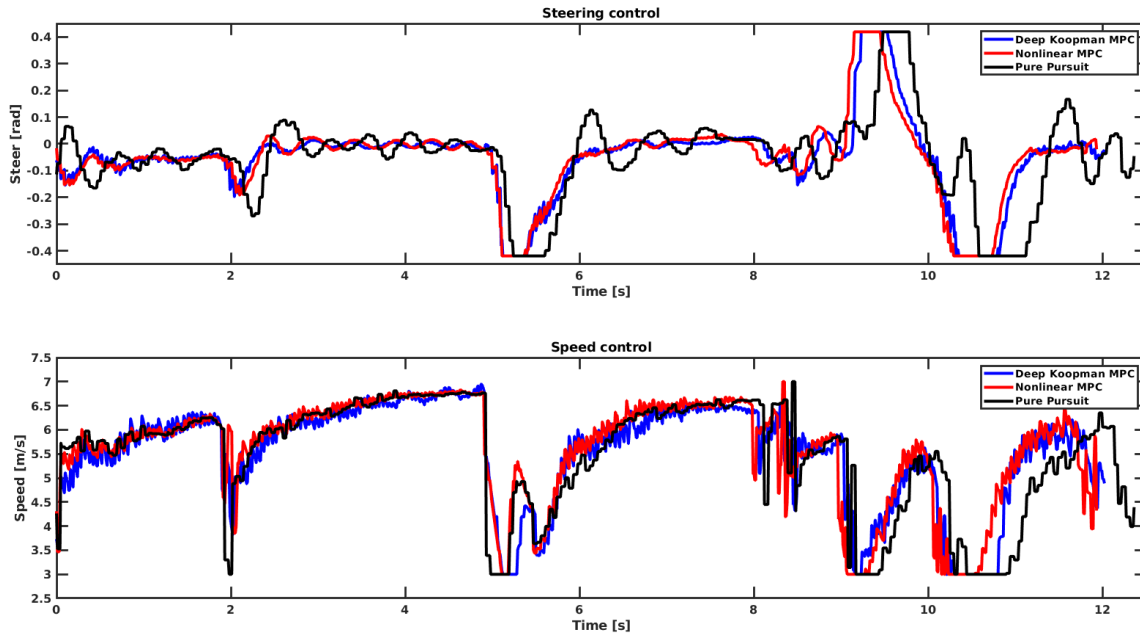


Figure 4.4: Control Output of Different Controller over a Single Lap

Due to the general issue of geometrical controller, the pure pursuit controller's settling time and control stability can't match model-based optimization controller's performance. For the optimization-based control, the result from kinematic single-track model and koopman learned data-driven model are somewhat similar, except that deep koopman data-driven MPC controller tends to generate more small correction steering controls on the straight line compared to nonlinear kinematic MPC controller. As a result, this small correction may give deep koopman MPC a better position tracking accuracy than nonlinear kinematic MPC as shown in Fig.(4.3).

4.2.4 Obstacle avoidance test

As one of the most important aspect of robot navigation, obstacle avoidance has become a standard test for vehicle control design. In our case, we perform the obstacle avoidance test in fltenth simulator environment as shown in Fig.(4.5).

Suggested by the work from [1], using two step MPC to achieve obstacle avoidance

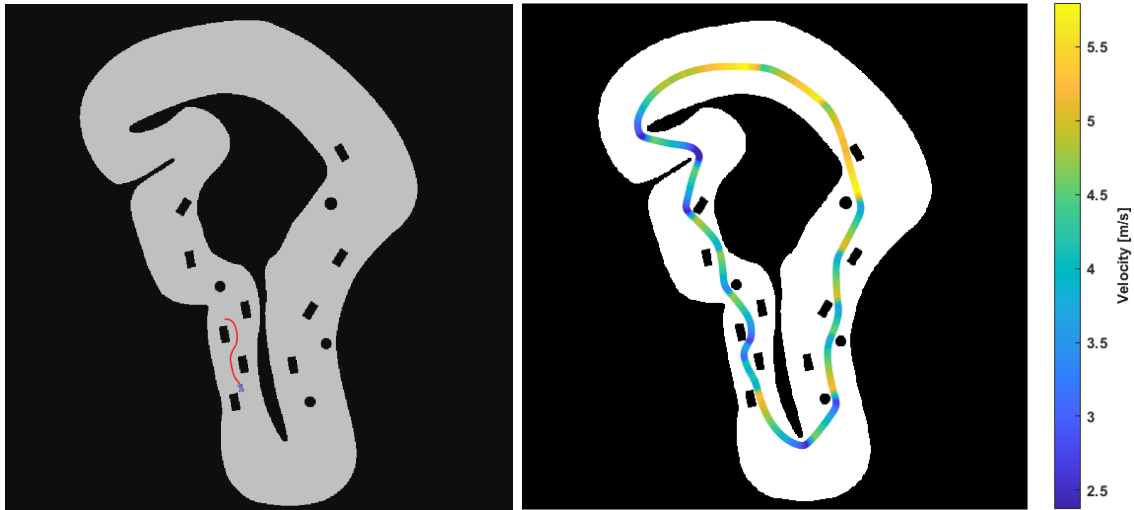


Figure 4.5: Obstacle Avoidance Test of Deep Koopman MPC in F1TENTH Simulator

task tend to generate smoother and safe controls compared to using one step only. Therefore, in our experiment, the obstacle avoidance task was also dissected into two steps as well. Similar to global trajectory optimization progress, the first step was to find a collision free path from the real-time vehicle pose to the desire reference pose selected from global trajectory. Then we compute the optimal velocity profile of this collision-free local path based on the optimization process as shown in section 2.1.1.

As shown in the Fig.(4.5), the deep koopman data-driven MPC is able to fulfill collision free navigation in a obstacle clustered environment with a highly twisted trajectory, which further validate the safety of our control approach.

4.3 Onboard Vehicle Experiment Result: 1/12th Scale Ackermann Steering Navigation Robot

As the deep koopman data-driven model predictive control's performance has been verified in simulation environment, it is equally important to validate it's performance in a real vehicle. In this case, we use a 1/12th scale autonomous driving robot as our test

platform.

The robot system hardware configuration is shown in figure 1 from the Appendix A section. For this particular application, we use particle filter as our localization algorithm. Therefore, the sensors used for localization is an IMU as well as an LiDAR. For low-level controller, we use a low-cost VESC(Vedders electronic speed controllers) to transform the control signals from the high level control into the voltage and frequency inputs to the brushless-motor and servo.

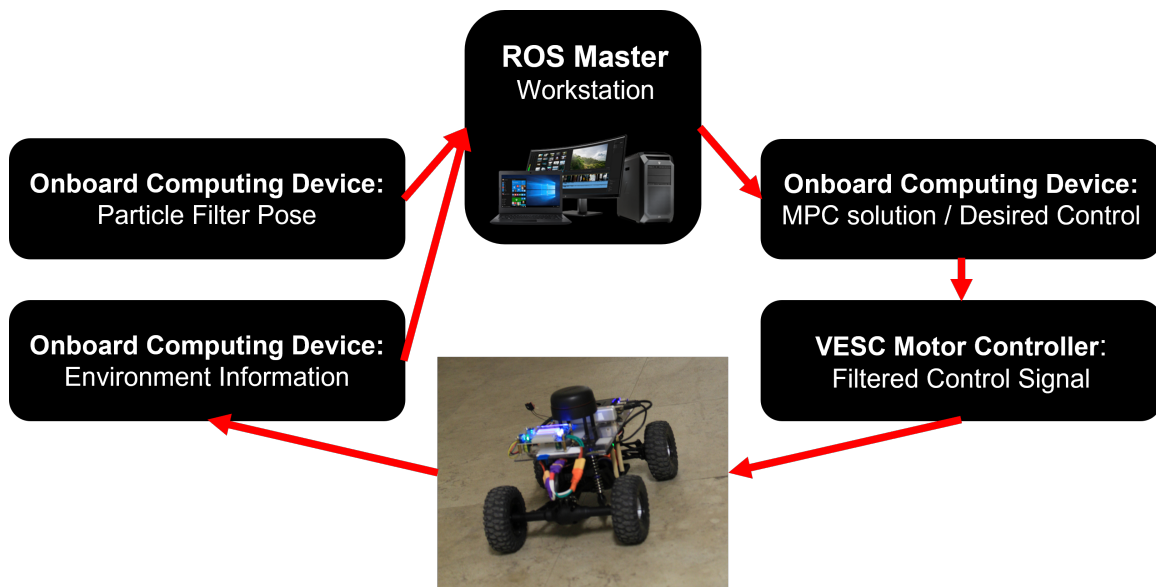


Figure 4.6: ROS communication Setup for Onboard F1TENTH Experiment

The high-level computing unit, which serves the purpose of implementing different planning and control algorithms, was initially selected to be a Jetson Nano. However, since Jetson Nano is built under ARM architecture, it doesn't allow us to use necessary optimization tools such as CasADI[4] and CVXPY to design complicated MPC controllers. Therefore, a computing device with x86/64 architecture is needed.

On the other hand, the efficiency of solving convex optimization will directly affect the control performance, and depending on only one low-power onboard computing device may not suffice. In this case, we use a workstation-level computing device to perform deep

koopman lifting as well as MPC solving task, then sending the control signal to the vehicle's onboard computing device(LattePanda, a x86 architecture computing device). Before executing the control signal from the ROS master, we applied hard constraints on the rate of change of control signals before sending them into the VESC controller so that we can guarantee the smoothness of the control outputs and prevent abrupt motion of the vehicle.

4.3.1 Trajectory Optimization in EIB basement

Before implementing deep koopman data-driven control, we need to obtain a closed-track for navigation. In this case, we use slam-gmapping from ROS repository to build the costmap of Clemson Fluor Danial Engineering Innovation Building, the map construction process can be visualized in Appendix A.

Once the complete costmap is built, we can use this map as the testing track for scaled RC car's navigation. In this case, an open area that contains both straight line and corners is selected (shown in the figure 2 from Appendix B) to test both the speeding and steering performance of deep koopman MPC controller.

During the initial part of the test, we first conduct the system identification of the vehicle dynamics. In this process, we ask a human to operate the vehicle within the selected area. When controlling the vehicle, it is preferable that the control input to be noisy so that more vehicle behaviors can be explored.

Once the system identification is complete, we use the iterative global trajectory generation methods to provide a fast racing line to test the controller's performance in high speed. As suggested in the Ph.D thesis by N. Kapania from Stanford [20], the value of minimum distance weight w_d in optimization in Eqn. (2.6) can effect the time consumed for navigation. Based on our experiment result, we choose w_d around 0.65 as the final minimum distance weight as it delivered the shortest navigation time on the track we selected

from the map. As for the choice of tire-road friction coefficient, we set μ to be 0.5 to avoid extreme control behavior, hence lower the risk of dangerous crash.

Table 4.3: Navigation Time Comparison Statistic

	Initial Path	Lap 1	Lap 2	Lap 3
Time [sec]	11.946	9.737	9.652	9.228
Mean speed [m/s]	3.0534	3.457	3.441	3.595

The statistics of trajectory evolution is shown in Table (4.3) and Fig.(4.7), the navigation time converges to a minimum value of 9.228 seconds within only 4 laps of iteration and average speed raise from 3.05 m/s to 3.595 m/s.



Figure 4.7: Optimal Trajectory Generation through Iterative Methods

4.3.2 Onboard Navigation Performance

To measure the control performance deep koopman data-driven MPC, we compare its tracking performance to nonlinear MPC based on kinematic single track model as well as adaptive pure pursuit controller. With the assistance of a deep neural network as the lifting function, the number of dimension on the lifted state-space is only 7 to 9, which can

be even lower than the required number of state from dynamic single track model as shown in 3.5. As a result, the update frequency of the MPC solver can reach nearly 100 hz, which is sufficient for high speed navigation task for 1/12th scaled RC car.

Table 4.4: Tracking Performance Comparison Statistic of Onboard Testing

	Position Error	Yaw Error	Speed Error
Nonlinear Kinematic MPC	0.111 m	0.0597 rad	0.5477 m/s
Deep Koopman MPC	0.0506 m	0.0501 rad	0.4746 m/s
Pure Pursuit + PID	0.1022 m	0.1076 rad	0.4854 m/s

The navigation statistic of onboard vehicle test is shown in table 4.3. Similar to the navigation performance in the simulator, pure pursuit controller’s performance is far behind the optimization based controller. Deep Koopman Data-Driven MPC has the lowest tracking error in all three tracking error measurements thanks to the accurate vehicle dynamical model. Nonlinear MPC based on kinematic single track model can match the yaw angle tracking performance yet the translation error is twice higher than the deep koopman MPC.

When observing the trajectory of all three controllers as shown in Fig.(4.8), Fig.(4.8) and Fig.(4.9), it is clear that pure pursuit and deep koopman MPC move towards the reference trajectory in almost the same rate, and both of them are faster than nonlinear kinematic MPC.(The state cost Q and control cost R are set to be the same for deep koopman mpc and nonlinear kinematic mpc, detail can be found in Appendix C) However, pure pursuit controller has higher overshoot compared to optimization-based controllers. Nonlinear kinematic MPC doesn’t have the high overshoot problem as pure pursuit, yet it suffers from position steady-state error as shown in Fig.(4.9). Therefore, in terms of overall performance, deep koopman MPC is better than the other two popular controllers.

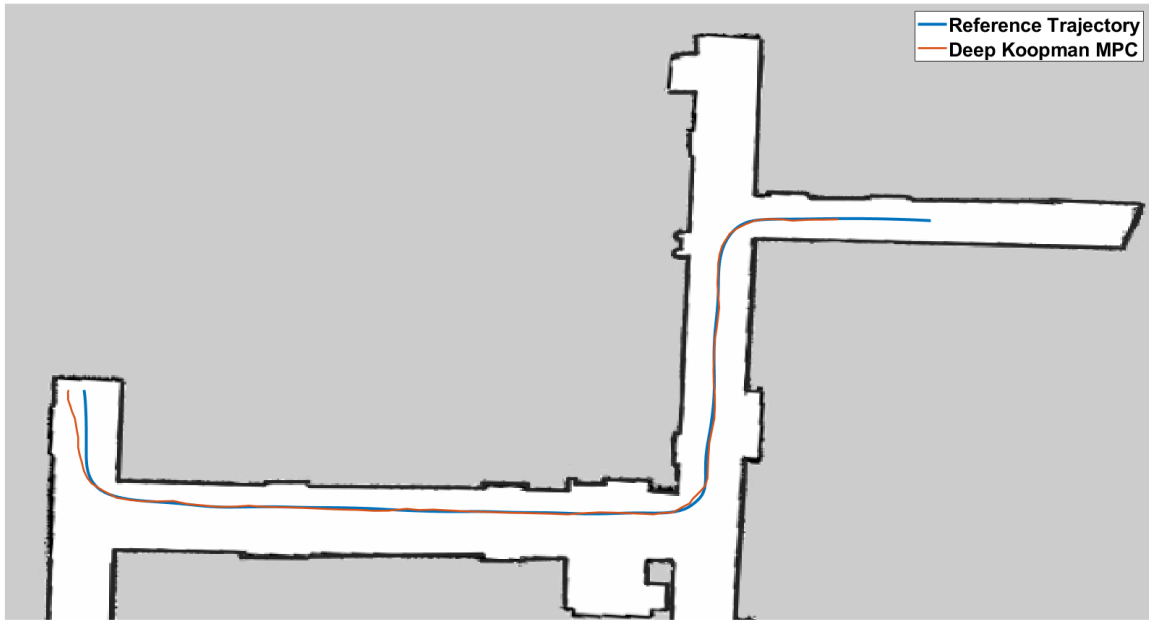


Figure 4.8: Onboard Tracking Record of Deep Koopman Model Predictive Control

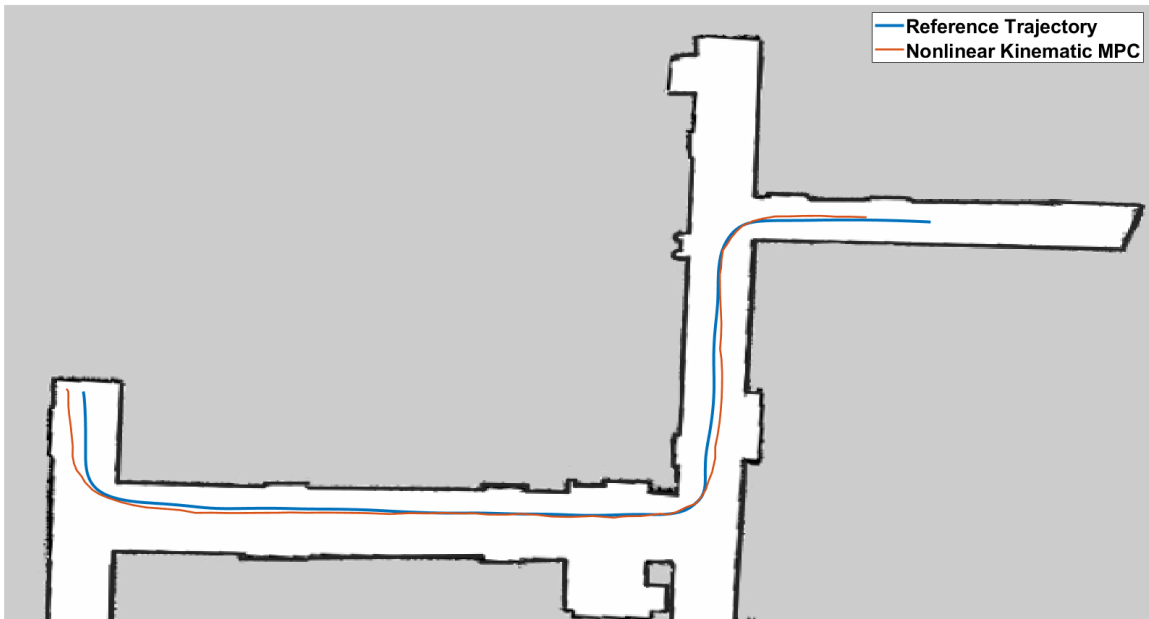


Figure 4.9: Onboard Tracking Record of Nonlinear Kinematic Model Predictive Control

One obvious difference between deep koopman MPC and nonlinear kinematic MPC in onboard test is the steering control when tracking low curvature reference path. As shown in the steering comparison Fig.(4.11), Deep koopman MPC tend to generate more

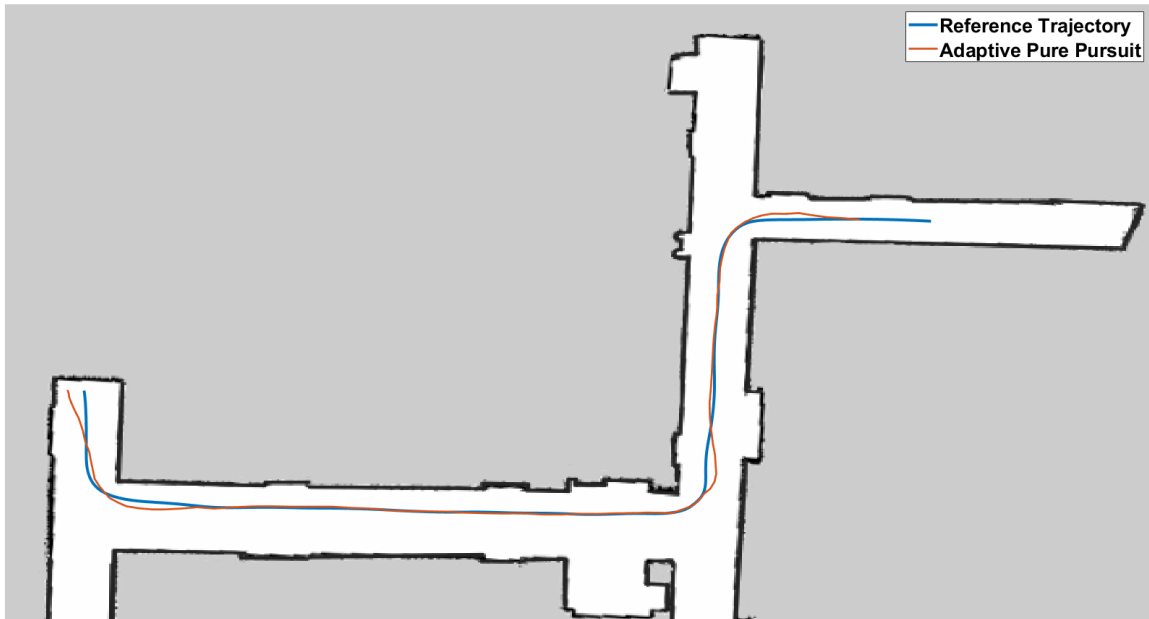


Figure 4.10: Onboard Tracking Record of Adaptive Pure Pursuit Predictive Control

steering corrections on the low curvature path such as straight line compared to the other two controllers. To an extent, these small corrections help to reduce the translation tracking error compared to nonlinear kinematic MPC. Also, despite the steering control being a bit noisy with low curvature reference path, the scale of the steering control noise is small enough not to affect the vehicle navigation safety as it will not led to dangerous behavior such as sudden change of steering.

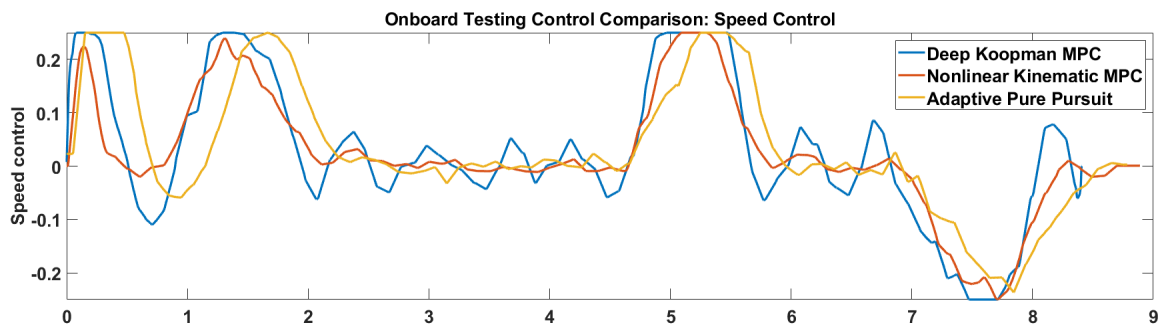


Figure 4.11: Onboard Testing Control Comparison

In the onboard testing, instead of using acceleration as control input to change the

vehicle speed, we use a desired speed input and feed the desired speed input directly into the low-level VESC controller. Since the VESC unit has a built-in PID controller that can directly control the motor speed based on the desired speed input, it is unnecessary to add an extra step into the longitudinal speed tracking task. Also the desired velocity has already satisfied the acceleration and maximum-tire-force constraint in the global trajectory optimization phase.

Chapter 5

Conclusions and Discussion

5.1 Stable and Efficient Control Performance of Deep Koopman Data-Driven MPC

Based on our experiment result, despite the vehicle's parameters are only partially known, using deep neural network and polynomial basis function as koopman operator to approximate the vehicle dynamics is good enough to capture the nonlinear behavior of the vehicles.

Not only did the simulation experiment result suggests that data-driven Koopman MPC is able to achieve better tracking performance than vehicle controllers such as pure pursuit and kinematic nonlinear MPC, the onboard testing also validate the computation efficiency of deep koopman data-driven controller.

One important reason of the satisfying real-world performance of deep koopman data-driven controller is that the koopman operator is optimized through a neural network training process rather than using predefined basis function such thinplate radial basis function or Gaussian radial basis function [19][10][24]. In this case, the dimension of the lifted

state-space can be set to a reasonably low value for the sake of efficiency of solving MPC problem on the lifted state-space. As for the accuracy of the model approximation, since we incorporate deep neural network and we use the state-update error as loss, the accuracy of the model depends on the training process of DNN, which means the DNN can eventually converge to the best koopman operator with a given output dimension.

With the balance between solver efficiency and model accuracy been optimized by applying DNN to koopman operator, our deep Koopman model predictive control can safely control the vehicle to track a high-speed optimized racing line without knowing the detailed vehicle information like cornering stiffness or center of gravity. This can potential save production and development cost of the autonomous vehicle since the facilities that originally used for measuring vehicle parameters are no longer needed or maintained. On the other hand, this vehicle model learning process can be implemented both online and offline. This give the vehicle the ability to learn the dynamics of itself within a very short time. In this case, we can adjust the vehicle dynamical model consistently to guarantee the vehicle control performance in any conditions.

5.2 Improvement on lap-time resulting from high tracking performance

Based on the statistics we collected from the simulator, the benefit of high tracking performance includes not only lower tracking error, but also lower lap-time.

Table 5.1: Comparison of Lap-time of different Vehicle Controllers

	Benchmark Lap-time	Deep Koopman MPC	Nonlinear Kinematic MPC	Adaptive Pursuit	Pure Pursuit
Lap-Time	10.15 sec	11.17 sec	11.24 sec	12.22 sec	

In table 5.1, the benchmark lap time is the optimized global trajectory time from

table 4.1. It is clear that Deep koopman has the closest lap time to benchmark lap-time compared to the other two controllers. More importantly, since we are racing in a grand-prix style closed track, the advantage of lap-time advantage will accumulate lap after lap. In this case, a small lap-time advantage is sufficient to decide the race winner.

5.3 Comparison with the other accomplished work on high performance vehicle control




Started from 2005 Grand Challenge Competition where Stanley from Stanford won a 132 miles Rally-style endurance racing, many well-known research groups have been devoted into autonomous racing and some of them have achieved great result. So it is necessary to compare our deep koopman data-driven MPC with the other well-accomplished vehicle controllers.

As mentioned above, the Dynamic Design Lab of Stanford University has successfully accomplished a full-scale vehicle autonomous racing project back in 2012. In their project, they conducted a detailed vehicle modeling based on a highly delicate vehicle parameters measurements to design their nonlinear MPC controller.[21] However, such method can be very expensive including the sensors needed for autonomous system design as well as the facilities to obtain detail vehicle parameters.

The Auto Rally Lab from Georgia Tech also successfully accomplished their autonomous racing project in 2017. Unlike the work from Stanford, they used a deep neural network to learn the vehicle dynamics. Given the difficulty of obtaining real vehicle's dynamics in real life, using learning-based approach can potentially deliver a better vehicle model compared to conventional methods.

As Georgia-Tech's work shown in their MPPI paper [11][36], the learned vehi-

Table 5.2: Comparison of Different Optimization-based Vehicle Controllers

	 Clemson	 Georgia Tech	 Stanford
Type of Vehicle Dynamical System	Learned	Learned	Modeled
Dynamical Model Representation	Neural Network + Linear State-space	Neural Network Only	Nonlinear State-space Only
Type of Optimization Method	Linear MPC / iLQR	Sample-Based MPC	Nonlinear MPC
Computational Efficiency	High	Low	Moderate
Vehicle Parameters Requirement	Low	Low	High
Overall production cost	Low	Moderate	High
Cost function Design Complexity	Low	High	Low

cle dynamics is only represented as a Deep Neural Network, which forced them to use a sampled-based MPC as well as a CNN learned cost function to design their controller. Such method can be highly computational expansive both in cost function as well as optimization design. Consequently, in order to satisfy the efficiency of high-speed navigation, the Auto Rally project has to rely on GPU acceleration and parallel computing to perform sampled-based MPC, which further increase the overall production cost.

To a extent, our Deep koopman MPC combines the advantage of these two approaches. Neural network is applied in Deep koopman MPC method, but its purpose is to transform the nonlinear vehicle dynamics from state-space into the observable space, rather than directly create mapping between state-update and control as in Auto Rally project. In this way, the state-update with respect to certain control inputs is still constructed as a linear system, which allow us to use convex optimization to find the optimal control solution.

Thus, not only did deep koopman operator use neural network to learn the vehicle dynamics, but it also facilitate the design of convex optimization on the space of observable.

5.4 Recommendations for Further Research

Despite the numerous advantages of the deep koopman data-driven MPC controller, its disadvantages are also obvious. The first disadvantage is the high requirement of training data selection.

Based on our research, the performance of approximated vehicle dynamical model depends on the number of time-series data and the amount of noises added to the control input. More data and more noisy control input can lead to a better model, yet highly noisy control input will lead to dangerous driving behavior in real world. In this case, finding the optimal noise level of vehicle control input when collecting time-series data for system identification is absolutely necessary.

Another problem is the robustness of deep neural network observable function's training process. During our training process as shown in chapter 4, we observed that the training performance rely heavily on the initial condition of the deep neural network. In other word, it can't not be guaranteed that every training process can deliver a highly efficient and accurate model. Therefore, another area for improvement is to find a better training strategy to increase the training robustness.

Lastly, we didn't include the vehicle model in the trajectory optimization process. Yet in the autonomous racing work from Stanford[[20]], they incorporated their vehicle dynamical model in the global path planning and optimization. Therefore, one potential new research direction is to use data-driven model in the path planning and optimization. Our model identification is easier to do, but the states on the observable-space contain very little physical meanings that can be used in the path planning design. In this case, the

challenge is how to properly setup the mapping from observable-space to the original state-space, and how to setup minimum-time optimization problem with data-driven model.

Appendices

Appendix A Hardware configuration

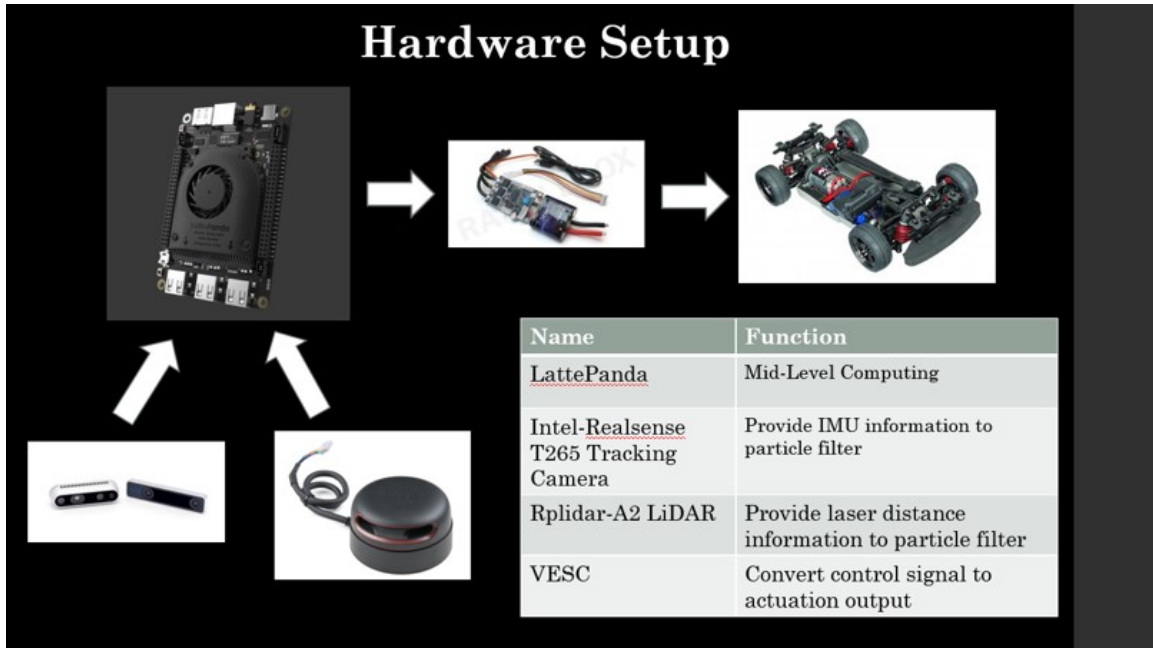


Figure 1: Hardware configuration of the ackermann steering autonomous robot for testing

Table 3: VESC specification for 1/12 Scale RC car

VESC parameter	Value
Speed to erpm gain	4150
Speed to erpm offset	0.0
Min Motor Speed	-30000
Max Motor Speed	3250
Maximum Current	30 A
Servo Position Range	[0 0.75]
Maximum Servo Speed	10.0 rad/s
Steering Angle to Servo Gain	-1.16
Steering Angle to Servo Offset	0.405

Appendix B Clemson Fluor Daniel EIB Basement's On-board Experiment

B.1 Costmap Built using Slam-gmapping

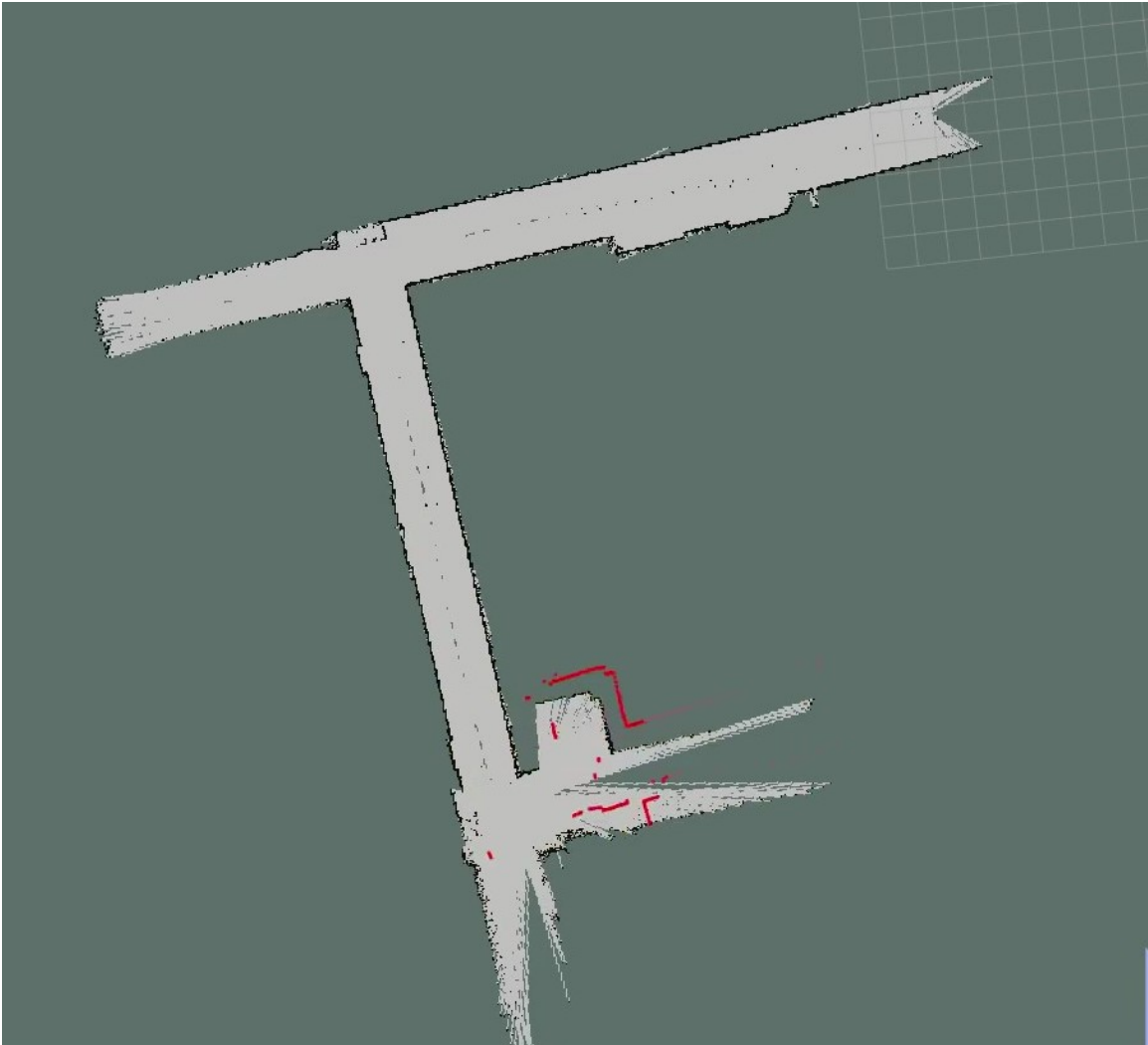


Figure 2: Build costmap of Clemson EIB basement

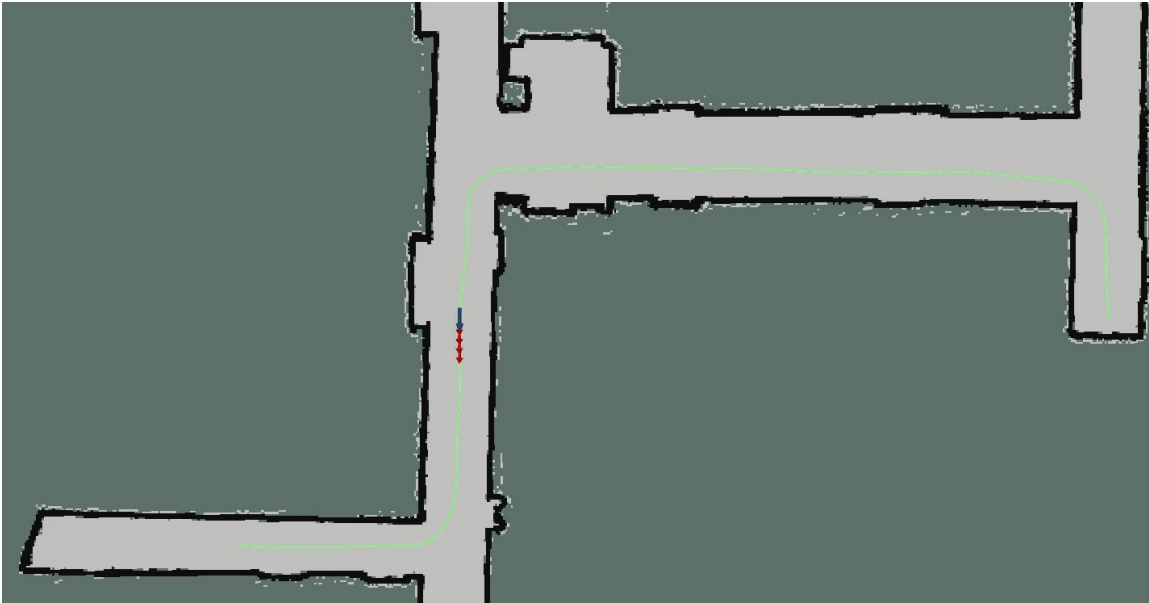


Figure 3: Use part of the costmap of EIB basement as racing track

B.2 Camera Record of RC car Navigation

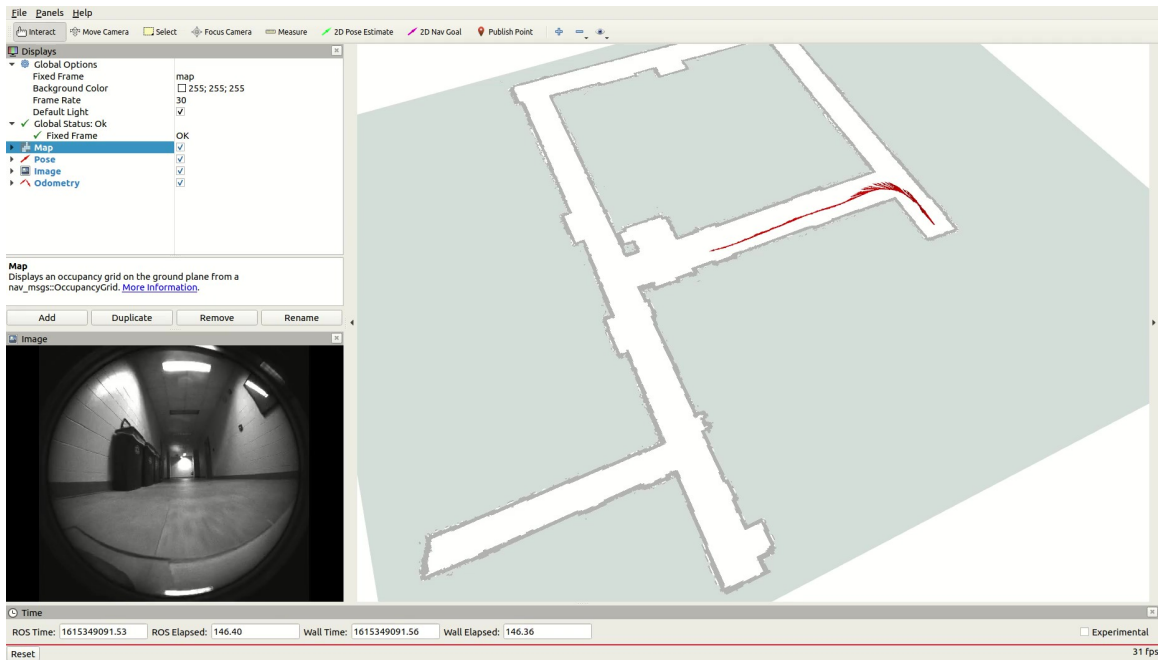


Figure 4: RVIZ visualization of F1TENTH onboard navigation



Figure 5: Onboard testing of high speed navigation of 1/12 scaled ackermann steering robot

Appendix C Controller Specification in Simulator and RC Robot

C.1 MPC parameters in F1TENTH Simulator

Table 4: Model predictive control setting of nonlinear kinematic model in F1TENTH Simulator

MPC parameter	Value
Horizon Length	5
Tracking Cost Q	$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$
Control Cost R	$\begin{pmatrix} 0.01 & 0.0 \\ 0.0 & 0.001 \end{pmatrix}$
Time interval dt	0.2s
Reference Pose Interval	0.2 m
Optimization solver	CasADi: ipopt
Acceleration Constraint	$[-8.0m/s^2 \quad 7.0m/s^2]$
Steering Rate Constraint	$[-5.0rad/s \quad 5.0rad/s]$
Maximum Steering Angle	0.4189 rad
Maximum Longitudinal Velocity	7 m/s
Control Output Frequency	1000 Hz

Table 5: Model predictive control setting of Deep Koopman Vehicle Model in F1TENTH Simulator

MPC parameter	Value
Horizon Length	5
Tracking Cost Q	$C^T \cdot \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \cdot C$
Control Cost R	$\begin{pmatrix} 0.001 & 0.0 \\ 0.0 & 0.001 \end{pmatrix}$
Reference Pose Interval	0.2 m
Optimization solver	CVXPY: OSQP
Acceleration Constraint	$[-8.0m/s^2 \quad 7.0m/s^2]$
Steering Rate Constraint	$[-5.0rad/s \quad 5.0rad/s]$
Maximum Steering Angle	0.4189 rad
Maximum Longitudinal Velocity	7 m/s
Control Output Frequency	1000 Hz

C.2 MPC Parameters for 1/12th Scaled RC Car

Table 6: Model predictive control setting of nonlinear kinematic model in 1/12th Scaled RC Robot Test

MPC parameter	Value
Horizon Length	4
Tracking Cost Q	$\begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix}$
Control Cost R	$\begin{pmatrix} 0.001 & 0.0 \\ 0.0 & 0.01 \end{pmatrix}$
Time interval dt	0.2s
Reference Pose Interval	0.25 m
Optimization solver	CasADi: ipopt
Acceleration Constraint	$[-4.0m/s^2 \quad 4.0m/s^2]$
Steering Rate Constraint	$[-8.0rad/s \quad 8.0rad/s]$
Maximum Steering Angle	0.25 rad
Maximum Longitudinal Velocity	5 m/s
Control Output Frequency	1000 Hz

Table 7: Model predictive control setting of Deep Koopman Vehicle Model in 1/12th Scaled RC Robot Test

MPC parameter	Value
Horizon Length	5
Tracking Cost Q	$C^T \cdot \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \cdot C$
Control Cost R	$\begin{pmatrix} 0.01 & 0.0 \\ 0.0 & 0.01 \end{pmatrix}$
Reference Pose Interval	0.15 m
Optimization solver	CVXPY: OSQP
Acceleration Constraint	$[-4.0m/s^2 \quad 4.0m/s^2]$
Steering Rate Constraint	$[-8.0rad/s \quad 8.0rad/s]$
Maximum Steering Angle	0.25 rad
Maximum Longitudinal Velocity	5 m/s
Control Output Frequency	1000 Hz

Appendix D Koopman Operator Training Specification

D.1 Deep Koopman in F1tenth Simulator

Table 8: Deep Neural Network Observable Function Training Specification

Neural Network Parameters	Value
Output Dimension	12
Input Dimension	4
Number of Hidden Layer	2
Dimension of Hidden Layer	512
Optimizer	Adam(Pytorch)
Learning Rate	10^{-4}
Number of Training Data	2508
Time delay for data collection	0.15 seconds

D.2 Augmented Koopman in F1tenth Simulator

Table 9: Polynomial Observable Function Approximation Specification

Polynomial Basis Function Parameters	Value																												
Polynomial Kernel Candidate	<table border="1"> <tr> <td>1</td> <td>X</td> <td>Y</td> <td>V</td> <td>Θ</td> <td>a</td> <td>δ</td> </tr> </table>	1	X	Y	V	Θ	a	δ																					
1	X	Y	V	Θ	a	δ																							
Kernel Coefficient matrix	<table border="1"> <tr> <td>0.0</td> <td>0.0</td> <td>0.0</td> <td>0.0</td> </tr> <tr> <td>1.028</td> <td>0.0</td> <td>0.0</td> <td>0.0</td> </tr> <tr> <td>0.0</td> <td>1.232</td> <td>0.0</td> <td>0.0</td> </tr> <tr> <td>0.0</td> <td>0.0</td> <td>1.0</td> <td>0.0</td> </tr> <tr> <td>0.0</td> <td>0.0</td> <td>0.0</td> <td>0.998</td> </tr> <tr> <td>0.0</td> <td>0.0</td> <td>0.1</td> <td>0.0</td> </tr> <tr> <td>0.034</td> <td>0.4114</td> <td>0.0</td> <td>1.2</td> </tr> </table>	0.0	0.0	0.0	0.0	1.028	0.0	0.0	0.0	0.0	1.232	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.998	0.0	0.0	0.1	0.0	0.034	0.4114	0.0	1.2
0.0	0.0	0.0	0.0																										
1.028	0.0	0.0	0.0																										
0.0	1.232	0.0	0.0																										
0.0	0.0	1.0	0.0																										
0.0	0.0	0.0	0.998																										
0.0	0.0	0.1	0.0																										
0.034	0.4114	0.0	1.2																										

Table 10: Deep Neural Network for Augmented Observable Function Training Specification

Neural Network Parameters	Value
Output Dimension	7
Input Dimension	4
Number of Hidden Layer	2
Dimension of Hidden Layer	512
Optimizer	Adam(Pytorch)
Learning Rate	10^{-4}
Number of Training Data	8802
Time delay for data collection	0.1 seconds

D.3 Deep Koopman in 1/12 Scale RC Car

Table 11: Deep Neural Network for Augmented Observable Function Training Specification

Neural Network Parameters	Value
Output Dimension	9
Input Dimension	4
Number of Hidden Layer	2
Dimension of Hidden Layer	512
Optimizer	Adam(Pytorch)
Learning Rate	10^{-4}
Number of Training Data	8802
Time delay for data collection	0.2 seconds

Bibliography

- [1] *Predictive Control of Autonomous Ground Vehicles With Obstacle Avoidance on Slippery Roads*, volume ASME 2010 Dynamic Systems and Control Conference, Volume 1 of *Dynamic Systems and Control Conference*, 09 2010.
- [2] Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88:142–153, 2017.
- [3] Matthias Althoff, Markus Koschi, and Stefanie Manzingler. Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726, 2017.
- [4] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.
- [5] Jonathan T. Barron. A general and adaptive robust loss function, 2019.
- [6] Alexander Broad, Todd Murphey, and Brenna Argall. Learning models for shared control of human-machine systems with unknown dynamics. *arXiv preprint arXiv:1808.08268*, 2018.
- [7] Steven L. Brunton, Bingni W. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS ONE*, 11(2):e0150171, Feb 2016.
- [8] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc), 2016.
- [9] Y. Chen, Y. Shan, L. Chen, K. Huang, and D. Cao. Optimization of pure pursuit controller based on pid controller and low-pass filter. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3294–3299, 2018.
- [10] V. Cibulka, T. Haniš, and M. Hromčík. Data-driven identification of vehicle dynamics using koopman operator. In *2019 22nd International Conference on Process Control (PC19)*, pages 167–172, 2019.

- [11] Paul Drews, Grady Williams, Brian Goldfain, Evangelos A. Theodorou, and James M. Rehg. Aggressive deep driving: Model predictive control with a cnn cost model, 2017.
- [12] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007.
- [13] Keisuke Fujii, Yuki Inaba, and Yoshinobu Kawahara. Koopman spectral kernels for comparing complex dynamics: Application to multiagent sport plays. In Yasemin Altun, Kamalika Das, Taneli Mielikäinen, Donato Malerba, Jerzy Stefanowski, Jesse Read, Marinka Žitnik, Michelangelo Ceci, and Sašo Džeroski, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 127–139, Cham, 2017. Springer International Publishing.
- [14] J. h. Jeon, S. Karaman, and E. Frazzoli. Anytime computation of time-optimal off-road vehicle maneuvers using the rrt*. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 3276–3282, 2011.
- [15] Yiqiang Han, Wenjian Hao, and Umesh Vaidya. Deep learning of koopman representation for control. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 1890–1895. IEEE, 2020.
- [16] Wenjian Hao and Yiqiang Han. Data driven control with learned dynamics: Model-based versus model-free approach, 2020.
- [17] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In *2007 American Control Conference*, pages 2296–2301, 2007.
- [18] Bowen Huang, Xu Ma, and Umesh Vaidya. Feedback stabilization using koopman operator. In *2018 IEEE Conference on Decision and Control*, pages 6434–6439. IEEE, 2018.
- [19] Bowen Huang and Umesh Vaidya. Data-driven approximation of transfer operators: Naturally structured dynamic mode decomposition. In *2018 American Control Conference*, pages 5659–5664. IEEE, 2018.
- [20] Nitin R. Kapania. *Trajectory Planning and Control for an autonomous race vehicle*. PhD thesis, Stanford University, 2016.
- [21] Nitin R. Kapania and J. C. Gerdes. An autonomous lanekeeping system for vehicle path tracking and stability at the limits of handling. 2014.
- [22] Hilbert J Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *AIP conference proceedings*, volume 887, pages 149–181. American Institute of Physics, 2007.

- [23] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, 2015.
- [24] Milan Korda and Igor Mezić. Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control. *Automatica*, 93:149–160, Jul 2018.
- [25] Ruipeng Li and Yousef Saad. Gpu-accelerated preconditioned iterative linear solvers. *The Journal of Supercomputing*, 63(2):443–466, 2013.
- [26] Niall M. Mangan, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Inferring biological networks by sparse identification of nonlinear dynamics, 2016.
- [27] S. Muller, M. Uchanski, and K.Hedrick. Estimation of the maximum tire-road friction coefficient. *Journal of Dynamic Systems Measurement and Control-transactions of The Asme*, 125:607–617, 2003.
- [28] Brian Paden, Michal Cap, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles, 2016.
- [29] Christoph Rösmann, F. Hoffmann, and T. Bertram. Kinodynamic trajectory optimization and control for car-like robots. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5681–5686, 2017.
- [30] U. Rosolia and F. Borrelli. Learning how to autonomously race a car: A predictive control approach. *IEEE Transactions on Control Systems Technology*, 28(6):2713–2719, 2020.
- [31] U. Rosolia, A. Carvalho, and F. Borrelli. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, pages 5115–5120, 2017.
- [32] Christoph Rösmann, Wendelin Feiten, Thomas Wösch, Frank Hoffmann, and Torsten Bertram. Efficient trajectory optimization using a sparse model. In *2013 European Conference on Mobile Robots*, pages 138–143, 2013.
- [33] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [34] Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

- [35] W. Wang, T. Hsu, and T. Wu. The improved pure pursuit algorithm for autonomous driving advanced system. In *2017 IEEE 10th International Workshop on Computational Intelligence and Applications (IWCIA)*, pages 33–38, 2017.
- [36] Grady Williams, Brian Goldfain, Paul Drews, James M. Rehg, and Evangelos A. Theodorou. Autonomous racing with autorally vehicles and differential games, 2017.
- [37] Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, 2015.
- [38] Matthew O. Williams, Ioannis G. Kevrekidis, and Clarence W. Rowley. A data–driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346, Jun 2015.