Optimal Control and Robot Learning on Agile Safety-Critical Systems

Haoru Xue

CMU-RI-TR-24-21

May 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

To my supportive advisor and hardworking collaborators.

Abstract

We present a pipeline of optimal control methods to learn an optimal control policy and locally accurate dynamics models for agile and safety-critical robots using autonomous racing as an application example. We introduce Spline-Opt, a fast offline/online optimization and planning method that can produce a reasonably good initial optimal trajectory given very little dynamics data. We then introduce EL-MPC, a residual learning MPC that relies on prior data to estimate dynamics models and learn the optimal control policy bounded by safe set constraints. All together, the data-driven pipeline is a road map going from zero understanding of a robot's dynamics, to the mastery of its handling limit and optimal performance.

Acknowledgments

The experience in M.S. Robotics at Carnegie Mellon University has been unparalleled. I first and foremost appreciate my advisor Professor John M. Dolan for his unwavering support in building a research community at CMU around autonomous racing. I also give many thanks to Professor Guanya Shi for being an invaluable member on my thesis committee and guiding my current research work. I've also had a wonderful time working with my lab-mates Dvij Kalaria, Wenli Xiao, Juan Alvarez-Padilla, Simin Liu, and Yiwei Lyu.

Across the coast, I must also give recognition to my teammates and collaborators at University of California Berkeley. Mr. Gary Passon, Dr. Allen Y. Yang, and Professor S. Shankar Sastry, three great principal investigators there, have been able to foster the growth of AI Racing Tech, a cross-university collaborative research program with over 30 researchers and 5 labs across CMU, UC Berkeley, and UC San Diego, with millions of funding. I found that my research took off by being an integrated part of the team. I especially recognize the effort of the great engineers and researchers who keep the vehicles running and pushes the limit of autonomous robotic systems: Siddharth Saha, C.K. Wolfe, Chris Lai, Eric Berndt, Raymond Song, and Tianlun Zhang.

I have had the privilege to work with Professor Francesco Borrelli and his PhD student Edward Z. Liu during summer 2023, who are remarkable researchers that have guided my early research with much patience and dedication.

Lastly, I want to thank my lover Vyvyen for always supporting me in facing life and career challenges, and encouraging me to reach out to opportunities far beyond my imagination. As I continue on my PhD study at UC Berkeley, I will continue to feel extremely grateful to all these amazing people who have crossed paths with me during my brief time at CMU.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Robot Learning at the Limit

Robot learning studies techniques that enable robots to explore the environment, acquire new skills, and adapt to changing conditions. Rooted in classic adaptive control and early machine learning research, robot learning has taken off with recent advancements in data-driven model predictive control (MPC), deep reinforcement learning (RL), large language models (LLM) and vision-language models (VLM). These technologies allow robot learning to be applied on complex, agile, hybrid robots, and largely expand robots' ability to interpret, reason, and plan tasks over longer horizons. They have also improved robots' ability to generalize and adapt to unseen, uncertain, time-varying tasks and conditions.

We are specifically interested in studying robot learning "at the limit". This refers to scenarios where the robot is learning to explore the extreme limit of its dynamics, and obtain an optimal control policy that fully exploits such a limit. Racing (of drones and cars) is a good example of such a task, where robots (or their human expert counterparts) must take the vehicles to the edge of their handling limits, and delicately explore control strategies at the limits.

Why is on-the-limit learning an interesting, relevant, and challenging research area? First, we are presented with a task that cannot be adequately addressed by existing robot learning paradigms. Traditional data-driven MPCs using parametric mathematical system models often fail to factor in the unmodeled properties of the robot and the environment. On the other hand, RL-based methods are often not data-efficient enough to be applied on a safety-critical system, where the cost of failure can be unboundedly high. When the system is already at the limit, any uninformed exploration may immediately send the robot into an unstable and uncontrollable state, and there is simply no reset button after crashing a million-dollar robot in the real world. We will explain these drawbacks more in detail in the background section.

Second, we find that existing studies often actively avoid on-the-limit when discussing robot learning. If a researcher wants to let the robot learn how to manipulate an object on the table, their first instinct is to place the object comfortably in the middle of the task space, and let the robot move as slowly as it can. If a humanoid team is demonstrating robot walking, they often teach the robot to bend its knees at all times to avoid dealing with singularities when the legs are fully stretched, which is inefficient but easy to implement. We aim to contribute to the ongoing discussion about limit-reasoning in robot learning techniques, and present a set of algorithms that are explainable and data-efficient.

Third, we emphasize that even finding the limit itself is a challenging problem. A limit is not drawn empirically by human experts, but is rather a result of safe exploration carried out by the robot to find the system dynamics in various conditions and understand the extreme at which it is still controllable. The limit is also often not a clear-cut line and includes uncertainties and time-varying properties. Learning the limit should also be incorporated into the robot learning problem itself.

1.2 Autonomous Racing

1.2.1 Introduction to the Racing Challenge

Autonomous racing is a growing research field in automotive technology and robotics. Over the years, researchers have scaled up from miniature RC cars to full-size, high-performance race cars traveling nearly as fast as human professional drivers; algorithms that were previously only evaluated in simulations are now deployed onto real-world autonomous race cars. For example, a vehicle in the Indy Autonomous Challenge (IAC) [1], which is the target platform of this work, is capable of reaching 320 km/h (200 MPH). The platform uses the same vehicle as the IndyLights racing series, and is equipped with multiple LiDARs, cameras, radars, and RTK GNSS units, enabling high-speed, head-to-head competitions. The platform represents the current state of the art in the field.

Many characteristics can be used to define autonomous racing as a complex robotics problem. First, an autonomous race car is an agile system with complex, uncertain, and time-varying dynamics. With sophisticated tire dynamics and aerodynamics, the accurate modeling of such a system is difficult and costly. This introduces two challenges for model-based approaches such as MPC and RL: a large sim-to-real gap is inevitable; and direct online evaluation of such a model, such as in a nonlinear model predictive controller (NMPC), is computationally prohibitive. Adding to these challenges, the dynamics of a race car also change over time due to, for example, track surface temperature and tire degradation, requiring a control system that can identify and adapt to these changing environment conditions and system parameters.

Second, real-world autonomous racing is a safety-critical task. Outside of rigorous engineering efforts to design real-time software and hardware systems, algorithm research must also steer towards robustness, explainability, and safety assurances. With no room for error, there is no such thing as resetting and respawning in real-world autonomous racing. Since we identify and exploit the full dynamical capability of the vehicle, this creates a unique challenge compared to typical robotic systems operating far within the safety limit.

With these characteristics and challenges, autonomous racing presents an opportunity to develop agile robotic systems in safety-critical tasks.

1.2.2 Racing and Robot Learning

What are the robots learning in a racing problem?

Learn a model: Commonly seen in data-driven MPC and model-based reinforcement learning (MBRL), a robot can leverage online and offline data to refine its dynamics model. Table 1.1 compares different classes of model learning methods that will be covered in the background section. We specifically highlight that autonomous racing requires real-time, online and transparent learning approaches, which is why traditional approaches like system ID and Gaussian processes are favored in much previous literature, working with MPC or reactive controllers (RC).

Method	Learning Objective	When	Where	Related Work
System ID	specific model parameters (mass, inertia, etc.)	Online / Offline	MPC and RC	[33]
Gaussian process	Gaussian process model.	Online / Offline	MPC and RC	[53]
Full model	A complex nonlinear system model.	Offline	MBRL	[34]
Local linear model	The local linearization of a nonlinear model.	Online	LMPC	[41]
Error model	Residual between a prior model and data.	Online	EL-MPC	Our Work

Table 1.1: A comparison of model learning methods: objectives, applications, and related works.

Learn a policy: Whereas RL policies are less structured functions constructed from generic neural network layers, MPC policy components are structurally defined into model, cost, constraint, and optimizer. In both cases, reasoning about the handling limits via such methods as control barrier function (CBF) [4, 34] or reachability constraints [40] can be fused into the policy to encourage limit-exploring behaviors while considering the safety of the system.

1.2.3 Introduction to The Experiment Platform

In our real-world experiments, we will target two autonomous racing platforms: F1TENTH [39], a 1/10-th scale vehicle platform; and AV-21, the primary full-size racing vehicle by the IAC. For a list of autonomous racing vehicles in research for comparison, we refer the reader to [5].

The F1TENTH [39] platform developed by the University of Pennsylvania uses an off-the-shelf RC car chassis which has been heavily modified. Figure 1.1 visualizes some of its major components. The main sources of localization and perception are the 2D LiDAR scanner and depth camera mounted on the front of the vehicle. System ID is conducted on the electronic speed controller system to command the correct current amount (A) in response to an acceleration command in m/s^2. We also thoroughly measure and calibrate the steering system dynamics to identify the maximum steering rate as well as the steering angle at the wheel.

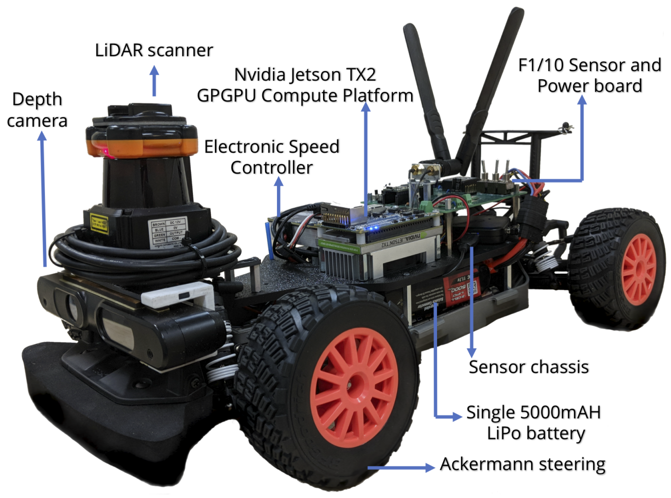
LiDAR scanner
Depth camera
Nvidia Jetson TX2 GPGPU Compute Platform
Electronic Speed Controller
F1/10 Sensor and Power board
Sensor chassis
Single 5000mAH LiPo battery
Ackermann steering

Figure 1.1: Components onboard an F1TENTH vehicle [39].

The AV-21 is modified from the Dallara IL-15, a human-driven race car chassis for the IndyLights series. A drive-by-wire system has been engineered to allow onboard autonomous operation. The cockpit hosts a x86-64 server with a 24-core Intel Xeon D-2166NT CPU at 3 GHz, and an NVIDIA Quadro RTX 8000 GPU. Perception is handled by 3 Luminar Hydra LiDARs and 6 cameras, complemented by mmWave radars. This perception package can be visualized in Figure 1.2. In addition, loccalization ground-truth is provided by two NovAtel RTK GNSS sensors.

In Table 1.2, we compare the specifications and peak performance of the two platforms. We highlight that although the F1TENTH is only a fraction of the size of an AV-21, it can achieve a similar level of peak longitudinal and lateral acceleration on a much smaller track, making it an ideal platform for controlled experiments and rapid prototyping.

1.2.4 Contribution of this Work

We present two novel optimal control frameworks for autonomous racing: spline-based minimum-curvature trajectory optimization (Spline-Opt), and Error-Learning Model Predictive Controller (EL-MPC). The two frameworks are connected in that

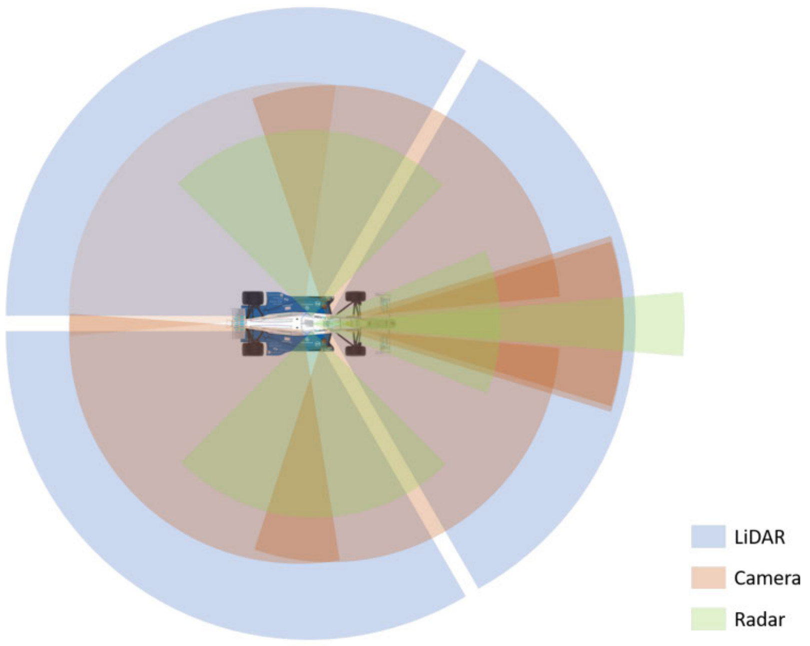

Figure 1.2: Visualization of the perception ranges on AV-21 [44].

Spline-Opt provides an offline optimization prior, which can be leveraged to initialize the EL-MPC in its online learning of optimal dynamics model and control policy.

In Spline-Opt, we consider the unavailability of sophisticated race car and race track dynamics in early-stage autonomous motorsports development and derive trajectory optimization techniques that work with limited dynamics data and additional conservative constraints. We formulate a minimum-curvature optimization problem with only the spline control points as optimization variables. We then compare the current state-of-the-art method with our optimization result, which achieves a similar level of optimality with a 90% reduction on the decision variable dimension, and in addition offers a mathematical smoothness guarantee and flexible manipulation options. We concurrently reduce the problem computation time from seconds to milliseconds for a long race track, enabling future online adaptation of the previously offline technique.

In EL-MPC, we present a novel Learning Model Predictive Control (LMPC) strategy for autonomous racing at the handling limit that can iteratively explore and learn unknown dynamics in high-speed operational domains. We start from existing LMPC formulations and modify the system dynamics learning method. In

Specification	**F1TENTH**	**AV-21**
Mass	2.2 kg	800 kg
Wheelbase	0.26 m	4 m
Top Speed	20 km/h	320 km/h
Top Lateral Acceleration	10-15 m/s^2	15-20 m/s^2
Top Longitudinal Acceleration	10-15 m/s^2	10-15 m/s^2
Top Braking	10-15 m/s^2	20-30 m/s^2
Max Engine Power	100 W	340 kW

Table 1.2: Vehicle specifications of F1TENTH and IAC platforms used in this work.

particular, our approach uses a nominal, global, nonlinear, physics-based model with a local, linear, data-driven learning of the error dynamics. We conducted experiments in simulation and on 1/10th-scale hardware, and deployed the proposed LMPC on a full-scale autonomous race car used in the Indy Autonomous Challenge (IAC) with closed loop experiments at the Putnam Park Road Course in Indiana, USA. The results show that the proposed control policy exhibits improved robustness to parameter tuning and data scarcity. Incremental and safety-aware exploration toward the limit of handling and iterative learning of the vehicle dynamics in high-speed domains is observed both in simulations and experiments.

The rest of this work is organized as follows. Chapter 2 discusses some previous literature on Learning Model Predictive Controller (LMPC) and spline-based trajectory optimizations; Chapter 3 introduces Spline-Opt; Chapter 4 introduces EL-MPC; and finally Chapter 5 gives conclusion remarks.

Chapter 2

Background

2.1 Deep Learning in Adaptive Control

An adaptive control system is designed to modify control policies online based on some discrepancies between a prior model's prediction and real-world observation. An illustration of such a process is presented in figure 2.1, which shows a reactive controller with adaptation capabilities against model deviations.

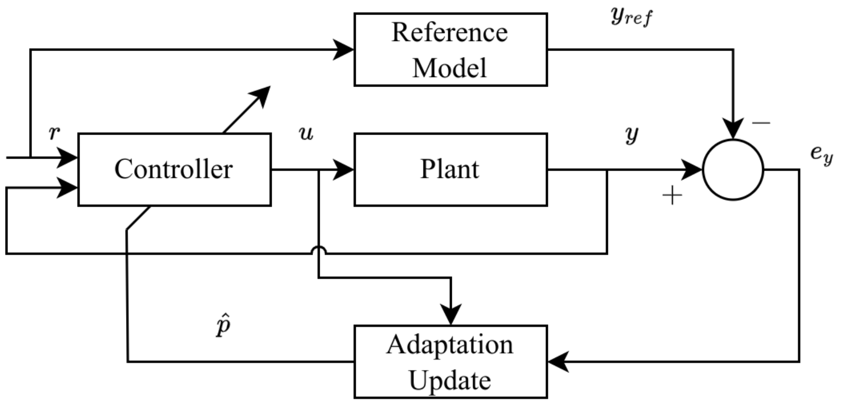

Figure 2.1: A classic adaptive control system that updates some controller parameter $\hat{p}$ based on the difference between some observation y and model prediction y_{ref}.

These adaptive control systems have been widely used, for example, in the control of unmanned aerial vehicles (UAVs) under variable and unpredictable environmental conditions, such as strong winds [21, 24, 50]. These methods, while effective in steady, nondynamic wind conditions, fall short in adaptability and precision when faced with changing aerodynamics.

There has been a growing interest in leveraging deep learning for aerodynamic modeling, aiming to capture the complex, nonlinear interactions between UAVs and their operating environments [45, 46, 47]. *Neural-Fly* [38] introduced a deep learning-based trajectory tracking controller that rapidly adapts to changing wind conditions. By employing a Domain Adversarially Invariant Meta-Learning (DAIML) algorithm, *Neural-Fly* learns a wind condition-independent representation of aerodynamics, which is then adapted online through linear coefficients for precise control in various wind conditions. This method not only improves upon the data efficiency of previous models but also ensures robustness and generalizability across different UAVs and wind conditions.

Furthermore, the integration of meta-learning techniques in *Neural-Fly* aligns with current research trends in robotic control, where learning to quickly adapt to new tasks or environments using efficient data models has shown great promise [3, 23, 28, 36, 49]. This approach, emphasizing the separation of a common representation from environment-specific adaptations, enhances real-time adaptability and performance in dynamic conditions.

2.2 Learning Model Predictive Controller

For controller architectures responsible for trajectory tracking at the edge of vehicle dynamics handling, a variety of MPC methods continue to be the state of the art in full-size race car systems. These MPC algorithm variations address a finite-horizon optimal control problem (FHOCP) by calculating optimal actuation instructions for a vehicle represented by kinematic or dynamic models, subject to specific track boundary and state constraints [6, 25, 31, 52]. However, when it comes to high-speed racing, the performance of traditional non-adaptive MPC techniques is ultimately limited by the accuracy of the vehicle dynamics model. Faulty vehicle models can not only lead to less optimal behaviors like weaving, hunting, and steady-state tracking error, but can also become hazardous at the handling limit, particularly when the model overestimates this limit. This presents a paradox for autonomous racing research at the handling limit: to safely reach higher speed levels, more vehicle dynamics data is needed; but to gather high-speed data, the controller must first attain higher speed levels. Hence, the aim of this work is to resolve the aforementioned circular problem by conducting incremental and safety-conscious exploration towards the handling limit and repetitively learning the vehicle dynamics in high-speed environments.

Learning unknown or changing system dynamics has been studied within some classic control frameworks such as adaptive control [14, 43]. System identification is usually part of the controller design, which reasons about some modeling error or noise in the system and adapts the system model to account for them. In autonomous racing, previous works have proposed dynamics learning strategies using Gaussian Processes [27, 30, 37] or implicitly with Gaussian noise assumption [32]. The method is further extended to model multi-vehicle interactions and solve for the optimal overtaking maneuver [13, 53].

Another branch of works exploits the repetitive nature of autonomous racing in a single-vehicle scenario, and improves controller performance without any prior assumption about the distribution of the modeling error or noise. Data-driven methods, such as iterative learning control, can optimize the tracking of a repetitive trajectory given data from previous iterations by solving for control variable corrections [11]. More recently, the learning model predictive control (LMPC) framework has been proposed with a direct regression of local affine dynamics [12, 40, 42]. This combination of LMPC and regressive dynamics learning can iteratively learn the unknown system dynamics and reduce lap-time without an explicit prior reference trajectory or a nominal dynamics model. The approach has also been demonstrated on scaled vehicle hardware platforms in [40].

2.3 Offline Trajectory Optimization in Autonomous Racing

Recent advancements in high-speed autonomous racing present new challenges and opportunities for evaluating Offline Trajectory Optimization (OTO) algorithms. Lack of prior race car and race track data is a significant challenge for university-level research and racing development. For example, although autonomous race cars in the Indy Autonomous Challenge (IAC) have reached a top speed of over 320 km/h, the teams still have limited access to critical data such as tire model parameters and load transfer characteristics, especially in the earlier stages of development, when estimation of these parameters is not viable with the limited data gathered. Therefore, a simple trajectory optimization algorithm should account for this data-scarce use case and support early development efforts.

OTO is often used to generate a reference safety set in the development phase of an autonomous race car, which is often desired when the handling limit of the vehicle is yet to be determined. Instead of directly applying an experimental tire model as the optimization limit, it is often desired to work with more conservative handling constraints. The traction circle (ellipse, or diamond) is an intuitive and effective alternative to the raw tire parameters. By controlling the maximum acceptable lateral and longitudinal acceleration in each direction as hyper-parameters, an OTO approach can generate different trajectories subjected to additional dynamics constraints. In addition, the track geometry constraints can be altered to impose extra track limits, and generate trajectories that pass through the non-optimal part of the race track, which provides a planning and control reference when the vehicle is forced into these regions.

Prior to autonomous racing, the generation of an optimal velocity profile given a fixed trajectory was first studied in the motorsports domain. Quasi-steady state (QSS) approaches have been developed since the 1980s [9, 10, 15, 48]. The full trajectory is broken down into small segments, through which the race car is assumed to have steady-state behavior. The algorithm starts with segments corresponding to peak curvature on the trajectory, which are considered the "bottlenecks". The algorithm then proceeds to generate a full velocity profile entering and exiting these bottlenecks, considering only the neighboring vehicle states subject to the dynamic limits, until the velocity profiles of these bottlenecks meet each other [18]. This method is known for its robustness and fast run time, but it fails to capture transient effects such as load transfer characteristics and damper dynamics [18]. We adopt a similar method in our work, whose algorithm will be formally proposed in a later section.

The optimization of the trajectory geometry in autonomous racing has been studied in Braghin et al. and Kapania et al. with the "minimum curvature" heuristic, which states that an optimal racing trajectory minimizes the sum of curvatures around the track to minimize lap time [8, 35]. Heilmeier et al. extend the idea to a quadratic programming (QP) formulation with improvements to the curvature calculation. They also apply spline interpolation to the noisy raw data and the final output to obtain a smooth trajectory [26]. Their work was extensively used by the TUM team in recent high-speed autonomous racing events such as Roborace and IAC. However, to guarantee that the continuity of the trajectory during and after optimization is at least C^2 (continuous position, velocity, and acceleration), it is insufficient to optimize with respect to discrete samples along the trajectory (3.0 m interval in [26]), although spline interpolation could be applied in post-processing.

Chapter 3

Spline-Based Minimum Curvature Trajectory Optimization

3.1 Background

A single B-spline of order n is a parameterized curve, denoted as $B_{i,n}(x)$. It can be uniquely constructed from a series of non-descending knot points $t_0, t_1, \ldots, t_N$, subjected to

$$\sum_{i=1}^{N-n} B_{i,n}(x) = 1 \tag{3.1}$$

A B-spline has non-zero values only in the range of knot vectors $t_i < x \leq t_{i+n}$. Higher-order B-splines can be recursively defined.

$$\begin{gathered} B_{i,n+1}(x) = w_{i,n}(x)B_{i,n}(x) + (1 - w_{i+1,n}(x))B_{i+1,n}(x) \\ \text{where } w_{i,k}(x) = \begin{cases} \frac{x-t_i}{t_{i+k}-t_i}, & t_{i+k} \neq t_i \\ 0, & \text{otherwise} \end{cases} \end{gathered} \tag{3.2}$$

The resulting basis functions are C^{n-2} continuous and overlap throughout the knot sequence, which is visualized in Figure 3.1.

These basis functions allow us to define a spline on t_0, t_N that is a linear combina-

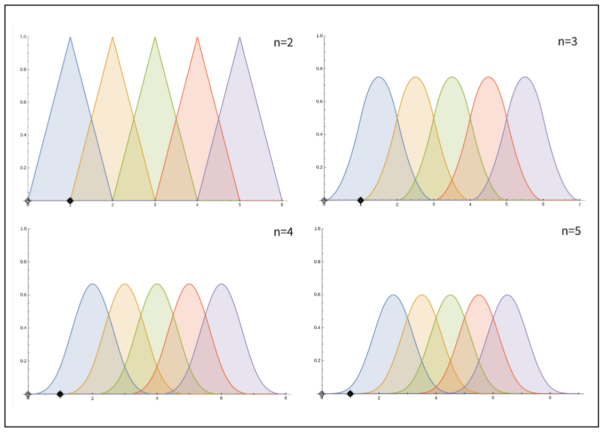

Figure 3.1: Visualization of basis functions $B_{i,n}(x)$ of order 2 to 5 [29]

tion of the basis functions:

$$T_n(x) = \sum_{i=1}^{N-n} \alpha_i B_{i,n}(x) \tag{3.3}$$

The weights $\alpha_0 \ldots \alpha_{N-n}$ are also known as control points, which can be visualized in Figure 3.2. We use $S = N - n$ to denote the number of control points. The shape of the curve can be manipulated by moving the control points while keeping the basis functions constant. The movement of the control points is the main subject of interest in this work, and we aim to derive an optimization formulation that optimizes their placement to form a curvature-optimal trajectory for autonomous racing.

To extend the 1D B-spline to handle a trajectory in the 2D plane, we take two sets of control points to parameterize the x and y coordinates separately on the same basis functions. That is, given a trajectory $T(t) : \mathbb{R} \rightarrow \mathbb{R}^2$ and a sequence of control points $\mathbf{z} = [\alpha_1, \ldots, \alpha_S, \beta_1, \ldots, \beta_S]^T$

$$T(t, \mathbf{z}) = (\sum_{i=1}^{S} \alpha_i B_{i,n}(t), \sum_{i=1}^{S} \beta_i B_{i,n}(t)) \tag{3.4}$$

where α_i, β_i respectively denote the x and y coordinates of the control point. Conventionally, we use $t \in [0.0, 1.0]$ to parameterize the trajectory and denote progress along the track. We can also denote the two resulting 1D B-splines as $T_x(t.\mathbf{z}), T_y(t, \mathbf{z})$.

Since we will be optimizing with respect to the control points, it is useful to take the derivative of a spline with respect to the control points, which is simply the corresponding basis function.

$$\frac{\partial T(t, \mathbf{z})}{\partial \alpha_i} = \frac{\partial T(t, \mathbf{z})}{\partial \beta_i} = B_{i,n}(t) \tag{3.5}$$

The curvature of a B-spline trajectory, as of any parametric curve equation, can be calculated as [26]

$$k(t) = \frac{T_x'(t)T_y''(t) - T_y'(t)T_x''(t)}{(T_x'(t)^2 + T_y'(t)^2)^{\frac{3}{2}}} \tag{3.6}$$

B-Splines have certain advantageous properties for trajectory optimization problems with respect to the control point movements. First, each control point can manipulate a piece of segment holistically. Intuitively, given the same knots $t_0 \ldots t_n$, a control point has a longer influence range along the curve as the degree of the B-spline increases. Visually in Figure 3.1, the basis function corresponding to that control point spans more intervals of knots. Second, since the basis function is evaluated to zero outside of its intervals according to (3.2), we can perform partial optimizations to a specific section of the trajectory without affecting the others. Lastly, the resulting trajectory from the control point movements is guaranteed to have the same order of continuity as the original trajectory since it is still a B-spline of the same order.

These features of B-splines make them suitable for autonomous racing applications. For a vehicle to have continuous velocity and acceleration profiles, the trajectory should be at least C^2 continuous, subject to additional curvature constraints since vehicle kinematics is not omnidirectional. To optimize the vehicle's trajectory through a specific segment of the race track, we can control the scope of the optimization by controlling the number of control points to be included in the optimization problem, and obtain results that are perhaps more locally optimized for a particular turn, or more globally optimized through a combination of turns.

3.2 Problem Formulation

3.2.1 Generating and Evaluating a Spline Trajectory

As an example, in this work we will use the Monza Circuit, which is a 5.8 km (3.6 mile) long race track used in the Indy Autonomous Challenge. It has a combination of long straights, high-speed turns, and chicanes. We will also show experiments on an other race track in a later section.

We obtain the race track geometries from satellite images and geographical surveys. We represent the track using a reference center line, with left and right offsets to denote the distances to the track boundaries at every waypoint. We then draw a cubic B-spline interpolation of the reference center line using the least square periodic spline interpolation algorithm discussed in [19], which forms a closed-loop spline on $t \in [0.0, 1.0]$.

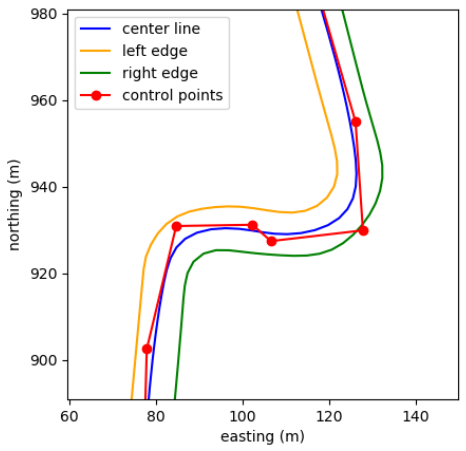

Figure 3.2: Turn 1 and 2 of Monza Circuit after interpolation

We then discretize the trajectory by taking waypoints at a constant 3-meter interval. This is done by performing a numerical integration to calculate the length of the trajectory:

$$L(t_{\min}, t_{\max}) = \int_{t_{\min}}^{t_{\max}} T'_x(t)^2 + T'_y(t)^2 dt \tag{3.7}$$

and solving a subsequent root-finding problem to find t_i such that the trajectory advances by 3 m.

$$t_i = \underset{t_i}{\operatorname{argmin}} \quad L(t_{i-1}, t_i) - 3 \tag{3.8}$$

We note that the optimization is done on the continuous spline, and these discretization points are simply used for sampling the curvature throughout the trajectory. The discrete trajectory is also useful in the evaluation process to be described below.

3.2.2 Configuring the Vehicle Parameters

As the main constraint on vehicle dynamics, we use a traction ellipse, which considers the longitudinal and lateral accelerations of the vehicle when accelerating, braking and cornering, or a combination of them. To characterize the shape of the ellipse, four parameters are used, which correspond to the maximum longitudinal acceleration, longitudinal deceleration, and left and right lateral accelerations. For the purpose of autonomous racing development, these parameters are easily adjustable to impose additional safety constraints within tire limits. For race cars with asymmetrical setups, such as those racing on an oval racing circuit, the left and right lateral accelerations can be adjusted separately. In our example, shown in Figure 3.3, we impose a maximum acceleration of 10 $\mathrm{m\,s^{-2}}$, deceleration of -20 $\mathrm{m\,s^{-2}}$, and symmetric maximum cornering load of $\pm$15 $\mathrm{m\,s^{-2}}$.

3.2.3 Simulating a Spline Trajectory

We perform a simulation for the best lap time on the spline trajectory to obtain the velocity profile, which will be used in the evaluation. The QSS algorithm starts by examining the curvature profile of the discretized trajectory and using the minimum-curvature points as constraints for this simulation. The relationship between the vehicle's tangential velocity v and curvature k satisfies

$$v = \sqrt{a_{lat}/k} \tag{3.9}$$

where a_{lat} is the lateral acceleration. Therefore, to maximize vehicle velocity through the bottleneck, the vehicle should have zero longitudinal acceleration to maximize

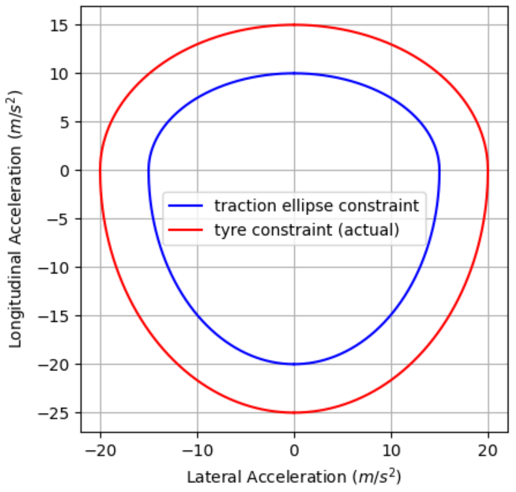

Figure 3.3: Geometry of traction ellipse used as constraint (blue), and actual tire constraints (red, simulated)

lateral acceleration according to the traction ellipse.

After obtaining this initial condition, the algorithm proceeds to generate an entry and exit velocity profile around the bottleneck points, until the two profiles of two bottlenecks converge. The trajectory is then further adjusted to ensure a smooth acceleration and velocity transition at the meeting points. Figure 3.4 shows a baseline simulation done on the reference center line trajectory, with a heatmap indicating the velocity levels that the vehicle can achieve in various sections of the track.

3.2.4 Spline-Based Minimum Curvature Problem

Building on [8] and [26], we consider the minimum-curvature optimization problem with respect to all control point coordinates $\mathbf{z}$.

$$\min_{\mathbf{z}} \quad \sum_{j=1}^{M} k_j^2(t) \tag{3.10}$$

where $k_1 \ldots k_M$ are the curvatures of the discretization points within the span of the corresponding basis function of $\mathbf{z}$. The difference from the previous work in the formulation is the optimization variable, which is the lateral movements of individual discretization points in the previous work, but is replaced with the control point

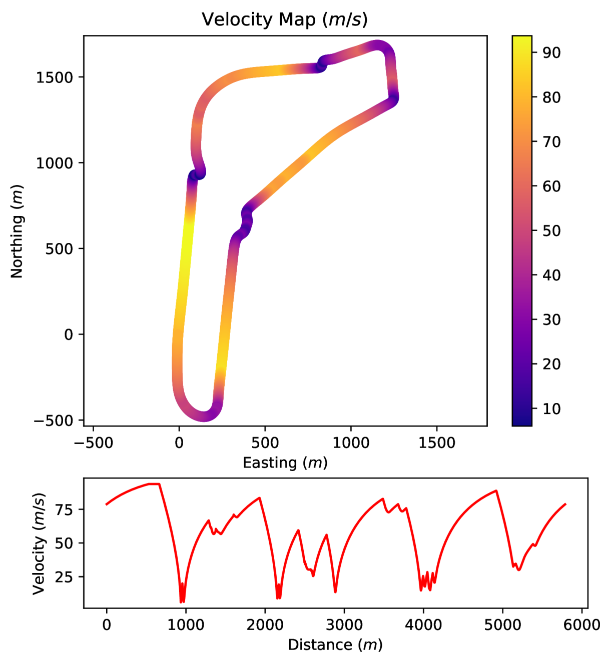

Figure 3.4: Example simulation on the center line trajectory. The heatmap shows the velocity levels (m/s) at individual track sections.

placements in our work. This reduces the dimension of the decision variable from M, the number of discretization points, to $2S$, twice the number of control points. For Monza Circuit with 3-meter interval discretization, the dimension reduces from 1932 to 204.

We then substitute (3.6) into (3.10) and omit the constant terms in the problem. Discarding the $(t, \mathbf{z})$ notation in $T(t, \mathbf{z})$ for simplicity, we arrive at a similar QP formulation to that in Heilmeier et al. [26]:

$$\min_{z} \quad T_x''^T P_{xx} T_x'' + T_y''^T P_{xy} T_x'' + T_y''^T P_{yy} T_y'' \tag{3.11}$$

where

$$P_{xx} = \begin{bmatrix} \frac{(T_{y_1})'^2 v_1}{((T_{x_1})'^2 + (T_{y_1})'^2)^3} & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(T_{y_M})'^2 v_M}{((T_{x_M})'^2 + (T_{y_M})'^2)^3} \end{bmatrix}$$

$$P_{xy} = \begin{bmatrix} \frac{-2(T_{x_1})'(T_{y_1})' v_1}{((T_{x_1})'^2 + (T_{y_1})'^2)^3} & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{-2(T_{x_M})'(T_{y_M})' v_M}{((T_{x_M})'^2 + (T_{y_M})'^2)^3} \end{bmatrix}$$

$$P_{yy} = \begin{bmatrix} \frac{(T_{x_1})'^2 v_1}{((T_{x_1})'^2 + (T_{y_1})'^2)^3} & 0 & \cdots & 0 \\ 0 & \ddots & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{(T_{x_M})'^2 v_M}{((T_{x_M})'^2 + (T_{y_M})'^2)^3} \end{bmatrix}$$

We now need to correlate T_x'' and T_y'' with the location of the j-th control point $z_j = \begin{bmatrix} \alpha_j & \beta_j \end{bmatrix}^T$ through the second-order derivative of the underlying B-spline basis. These lower-order bases also have a recursive closed-form solution and can be treated as constants [17].

$$
\begin{aligned}
T_x'' &= \sum_{i=1}^{j-1} \frac{d^2 B_{i,n}(t)}{dt^2} x_i + \frac{d^2 B_{j,n}(t)}{dt^2} x_j + \sum_{i=j+1}^{S} \frac{d^2 B_{i,n}(t)}{dt^2} x_i \\
&:= F_x + \frac{d^2 B_{j,n}(t)}{dt^2} x_j \qquad (3.12a) \\
T_y'' &= \sum_{i=1}^{j-1} \frac{d^2 B_{i,n}(t)}{dt^2} y_i + \frac{d^2 B_{j,n}(t)}{dt^2} y_j + \sum_{i=j+1}^{S} \frac{d^2 B_{i,n}(t)}{dt^2} y_i \\
&:= F_y + \frac{d^2 B_{j,n}(t)}{dt^2} y_j \qquad (3.12b)
\end{aligned}
$$

This shows that the relation between T_x'' and x_j, or T_y'' and y_j, is affine. If optimization is done with respect to all the control points at once, then F_x and F_y reduces to zero, since (3.12a) and (3.12b) now become

$$
\begin{aligned}
T_x'' &= \sum_{i=1}^{S} \frac{d^2 B_{i,n}(t)}{dt^2} x_i \qquad (3.13a) \\
T_y'' &= \sum_{i=1}^{S} \frac{d^2 B_{i,n}(t)}{dt^2} y_i \qquad (3.13b)
\end{aligned}
$$

In which $\{x_0, \ldots, x_S\}$ and $\{y_0, \ldots, y_S\}$ are all decision variables.

Substituting (3.13a) and (3.13b) into (3.11), we can formulate this problem as a standard QP.

$$
\min_{\mathbf{z}} \quad \frac{1}{2}\mathbf{z}^T H z + g^T \mathbf{z} \qquad (3.14a)
$$

where

$$
\begin{aligned}
H &= B_x^T P_{xx} B_x + B_y^T P_{xy} B_x + B_y^T P_{yy} B_y \qquad (3.14b) \\
g &= (F_x^T P_{xx} B_x + F_y^T P_{xy} B_y + F_y^T P_{yy} B_y) + \\
&\quad (B_x^T P_{xx} F_x + B_y^T P_{xy} F_x + B_y^T P_{yy} F_y)^T \qquad (3.14c)
\end{aligned}
$$

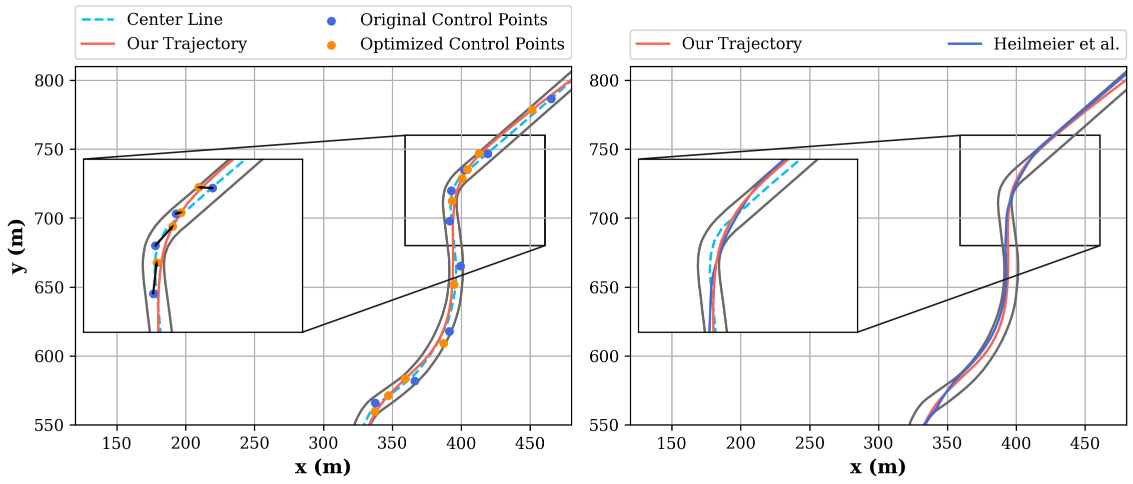

Figure 3.5: Optimization visualization at Monza Turn 8-10. The figure on the left visualizes the movements of the control points which shift the center line spline into a minimum-curvature spline. The figure on the right compares the optimization results of ours and [26].

and where

$$B_x = \begin{bmatrix} \mathbf{B}_2 & \mathbf{0}^{M\times S} \end{bmatrix}^T \tag{3.14d}$$

$$B_y = \begin{bmatrix} \mathbf{0}^{M\times S} & \mathbf{B}_2 \end{bmatrix}^T \tag{3.14e}$$

$$\mathbf{B}_2 = \begin{bmatrix} \frac{d^2 B_{1,n}}{dt^2}(t_1) & \cdots & \frac{d^2 B_{S,n}}{dt^2}(t_1) \\ \vdots & \ddots & \vdots \\ \frac{d^2 B_{1,n}}{dt^2}(t_M) & \cdots & \frac{d^2 B_{S,n}}{dt^2}(t_M) \end{bmatrix} \tag{3.14f}$$

Note that (3.14c) is evaluated to zero when considering all the control points since $F_x = F_y = \mathbf{0}$.

Finally, we obtain the constraints for the optimization, which is the distance to the left and right boundaries. For the i-th discretization point, we use $\mathbf{l} = \begin{bmatrix} l_0 & \cdots & l_M \end{bmatrix}^T$ and $\mathbf{r} = \begin{bmatrix} r_0 & \cdots & r_M \end{bmatrix}^T$ to denote the center line's distances to the left and right boundaries. Then we use a local frenet frame approximation to find the distance between the optimized discretization point and the original center line. This is done by applying a rigid body transformation to the boundary point from the fixed map frame into the center line's local frame.

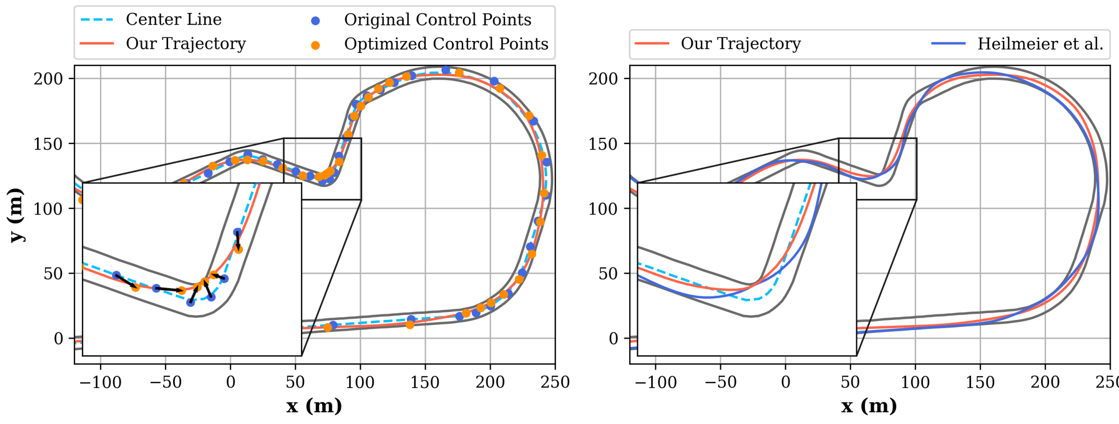

Figure 3.6: Optimization visualization at Putnam Road Course, Indiana, USA. The left figure shows the movement of the control points and the optimal trajectory shape as a result. The right figure compares ours and Heilmeier et al.'s trajectory through the chicane.

$$A^{2M\times 2S} = \begin{bmatrix} \mathbf{B} & \mathbf{0^{M\times S}} \\ \mathbf{0^{M\times S}} & \mathbf{B} \end{bmatrix}$$

$$R^{M\times 2M} = \begin{bmatrix} \cos\theta_1 & 0 & 0 & \sin\theta_1 & 0 & 0 \\ 0 & \ddots & 0 & 0 & \ddots & 0 \\ 0 & 0 & \cos\theta_M & 0 & 0 & \sin\theta_M \end{bmatrix}$$

where

$$\mathbf{B} = \begin{bmatrix} B_{1,n}(t_1) & \cdots & B_{S,n}(t_1) \\ \vdots & \ddots & \vdots \\ B_{1,n}(t_M) & \cdots & B_{S,n}(t_M) \end{bmatrix}$$

and where $\theta_i \quad \forall i \in \{1, \dots, M\}$ denotes the orientation of the center line at discretization point p_i. The linear constraint is therefore

$$\mathbf{r} \leq R A \mathbf{z} \leq \mathbf{l}$$

Breaking down this equation, $A\mathbf{z}$ computes the discretization coordinates from the control points, which is rotated by R to compute its projection on the left or right side of the center line.

Now we have all the components for our constrained QP problem, which we can solve with qpOASES [22].

$$\begin{aligned} \min_{\mathbf{z}} \quad & \frac{1}{2}\mathbf{z}^T H\mathbf{z} + g^T\mathbf{z} \\ \text{s.t.} \quad & \mathbf{r} \leq RA\mathbf{z} \leq \mathbf{l} \end{aligned} \tag{3.15}$$

3.3 Experiments and Results

Before presenting the results, it is beneficial to discuss the evaluation metrics for OTO methods. Since this is an open-loop offline method, it is insufficient to show the end trajectory achieved by a vehicle driving on the track, since there are other processes down the pipeline such as online dynamic planning and optimal control calculations. These modules also have a significant impact on the outcome of the experiment. Therefore, we will directly compare the OTO result with the current state-of-the-art method in minimum-curvature optimization in [26] to see if our method has achieved our objective, which is to offer an alternative minimum-curvature optimization formulation for autonomous racing that uses significantly fewer decision variables, guarantees continuity and still offers a trajectory comparable to previous work.

To this end, we will perform minimum-curvature optimization using our formulation and the Heilmeier et al. formulation [26]. The discretization intervals are the same in both experiments. We then simulate lap times with the QSS simulation method described in III.A to show that we have reached comparable results with the previous work.

We first compare the overall lap time performance of our method with the reference center line and Heilmeier et al. [26] in Table 3.1. The optimization algorithms produce 7.65% (ours) and 8.94% (Heilmeier et al.) lap time reduction, respectively, offering comparable maximum velocity and acceleration profiles. The percentage difference of the metrics is shown in the right column, which suggests that we achieved a level of optimality similar to [26]. However, our result produces a C^2 continuous B-spline curve that can be easily re-discretized and even artificially manipulated afterwards by dragging the control points, which is a very useful feature in the developmental stage of autonomous racing, whereas [26] only produces a discretized trajectory with no guarantee of continuity level due to its discrete decision variables. We also see a significant reduction in the dimension of the decision variable. Our QP problem has 204 decision variables (102 control points), down from 1932 using [26] with a 3 m discretization interval. The resulting QP computation time is 3.8 ms, down from 8.225 s using [26]. This shows that our method not only gives a more compact formulation that captures the essence of the minimum-curvature problem, but also enables the possibility of adopting such a method in online planning on a very long track.

Looking turn by turn, Figure 3.5 zooms in on the optimization at turns 8-10. The control points of the center-line spline are shifted to form a minimum-curvature trajectory. For turn 7 in the zoomed-in section, we are able to optimize the full trajectory with only four control points.

Table 3.1: Simulation Results on Monza

	Center Line	**Heilmeier et al.**	**Ours**	**Delta (%)**
Lap Time (s)	131.33	119.59	121.28	-1.4
Ave Speed (m/s)	43.72	48.60	47.73	1.8
Max Speed (m/s)	93.69	95.12	94.33	0.8
Min Speed (m/s)	6.04	10.51	9.44	11.3
Max Lat G (m/s^{-2})	13.96	13.78	14.48	-4.8
Max Throttling (m/s^{-2})	9.74	9.62	9.68	-0.6
Max Braking (m/s^{-2})	-19.87	-19.27	-19.85	-2.9

We then move on to a different race track and apply the same OTO method to see how our method transfers to a new track. Figure 3.6 visualizes the optimization results done on Putnam Road Course, Indiana, USA, which is a test track of the Indy Autonomous Challenge. The zoomed-in portion of the graph visualizes the movement of control points which leads to a curvature-optimal trajectory. Table 3.2 shows metrics similar to 3.1, in which comparable lap time reduction of 13.9% and 15.67% are respectively achieved by [26] and our work. The difference in the

Table 3.2: Simulation Results on Putnam Road Course

	Center Line	**Heilmeier et al.**	**Ours**	**Delta (%)**
Lap Time (s)	79.63	68.55	66.91	2.4
Ave Speed (m/s)	36.36	41.57	42.75	-2.7
Max Speed (m/s)	73.93	86.54	88.07	-1.7
Min Speed (m/s)	8.02	13.83	12.0	15.2
Max Lat G (m/s^{-2})	15.0	15.0	15.0	0.0
Max Throttling (m/s^{-2})	9.75	9.18	9.52	-3.6
Max Braking (m/s^{-2})	-16.95	-17.84	-18.22	-2.1

simulation metrics between ours and [26] remains close, suggesting that the outcome of the optimization is very comparable.

3.4 Summary

In this chapter we present a B-spline OTO method for autonomous racing which solves a minimum-curvature optimization problem. Compared to previous works, which only output a discretized trajectory, this work outputs a fully parameterized trajectory of C^2 continuity, which ensures a smooth control profile for high-speed vehicle handling. The algorithm also considers the data scarcity of early-stage autonomous motorsports development and requires minimal vehicle dynamics data. Compared to previous work [26], the dimension of the problem is significantly reduced from thousands of discretization points down to a few dozens of control knot points. The problem computation time is also drastically reduced as a result. This enables future work to explore the online application of minimum-curvature OTO for autonomous racing.

The work from Spline-Opt is also directly applicable to the Error-Learning MPC paradigm to be introduced in the next chapter. The curvature-optimal reference could serve as a reasonably good starting point for further online learning of the robot's true dynamics limit.

Chapter 4

Error-Learning MPC

4.1 Background

4.1.1 Sampled Safe Set

Learning from previous data often requires that the robot's state is not too out of distribution from the dataset. Previous work in Learning Model Predictive Control (LMPC) formalizes this idea into sampled Safe Set, which is a control invariant set [40]. Given a robot at its j-th iteration of completing a task, its sampled Safe Set is defined as

$$\mathcal{SS}^j = \left\{ \bigcup_{i \in M^j} \bigcup_{t=0}^{\infty} x_t^i \right\} \tag{4.1}$$

which represents the set of all robot states for all iterations indexed $i \in M^j$. Here M^j is the set of indices $k \leq j$ that successfully completed the iterative task before the current iteration:

$$M^j = \left\{ k \in [0, j] : \lim_{t \leftarrow \infty} x_t^k = x_F \right\} \tag{4.2}$$

where x_F represents some final state. In racing, for example, x_F can be easily thought of as all states of the robot past the finish line, which we will formalize in the next section.

A sampled Safe Set is interesting in real-world robot learning, because it offers invariance and reachability. If the environment has not changed, then we know that as long as the robot can reach a state in the sampled Safe Set, it is guaranteed to complete the task, because the previous iterations of data have proven that this trajectory will lead to the final state x_F. This property motivates us to impose a Safe Set constraint in optimal control, such that any exploration of a better trajectory always ends in the safe set:

$$\min_{u_0,\dots,u_\infty} \sum_{k=0}^{\infty} h(x_k, u_k) \tag{4.3a}$$

$$\text{s.t.} \quad x_{k+1} = f(x_k, u_k), \forall k \geq 0 \tag{4.3b}$$

$$x_0 = x_{\text{ic}} \tag{4.3c}$$

$$x_k \in \mathcal{X}, u_k \in \mathcal{U}, \forall k \geq 0 \tag{4.3d}$$

$$x_\infty \in \mathcal{SS}^j \tag{4.3e}$$

where $h(x_k, u_k)$ represents some stage cost, $f(x_k, u_k)$ represents the discrete robot dynamics, $\mathcal{X}$ and $\mathcal{U}$ represent the primal bound, and (4.3e) constrains the robot's terminal state to be within the Safe Set.

4.1.2 Convex Safe Set and Linear System

One drawback of the sampled Safe Set is that it is comprised of discrete robot states rather than a subspace which can continuously represent constraints for a robot's state. Without some relaxation, using the sampled Safe Set in optimial control directly would result in a mixed integer programming problem that is computationally prohibitive. Therefore, we investigate some techniques to relax this constraint.

Let's consider $f(x_k, u_k)$ in (4.3b) to be linear time-invariant (LTI), i.e.

$$x_{k+1} = Ax_k + Bu_k \tag{4.4}$$

Then, if the primal bounds $\mathcal{X}$ and $\mathcal{U}$ in (4.3d) are convex, which is often the case with simple maximum and minimum limits, then the convex combination of the states in $\mathcal{SS}^j$ would also have some control sequence that leads the robot into the final state

x_F. This is defined as the convex Safe Set $\mathcal{CS}^j$[40]

$$\mathcal{CS}^j = \left\{ \sum_{i=1}^{|\mathcal{SS}^j|} \alpha_i z_i : \alpha_i \geq 0, \sum_{i=1}^{|\mathcal{SS}^j|} \alpha_i = 1, z_i \in \mathcal{SS}^j \right\} \tag{4.5}$$

where $|\mathcal{SS}^j|$ is the cardinality of $\mathcal{SS}^j$. $\mathcal{CS}^j$ is also a control invariant set [7]. $\alpha_0, \alpha_1, \ldots,$ which are the convex combination coefficients, become decision variables in the optimal control problem. This relaxed version of the problem states that, for an LTI system, if the robot ends in the convex hull formed by states in $\mathcal{SS}^j$, then the terminal state x_F is reachable via some sequence of control commands.

However, this formulation leaves some important questions unanswered:

- How to apply the convex hull relaxation to a nonlinear system?
- When $\mathcal{SS}^j$ contains a large number of states, how do we choose its subset to allow efficient computation of the convex safe set?

We will explore these questions in the next section.

4.2 Problem Formulation

4.2.1 State-Space Representation of Vehicle Dynamics

A prior, closely-matched vehicle dynamics model would significantly improve the base MPC performance and satisfy algorithm transparency, explainability, and safety-fallback requirements for deploying model predictive controllers on a full-scale vehicle. We also theorize that, compared to previous work in [42] in which dynamics learning is performed without a model *a priori*, incorporating the vehicle model prior is key to reduce parameter tuning sensitivity and modeling error exploitation in EL-MPC.

First, we consider the state and control variables for the modeling. We can represent the vehicle's 2D dynamics with 6 state variables corresponding to the vehicle's position in a local Cartesian coordinate system.

$$x_c = [v_x, v_y, \omega_z, \psi, p_x, p_y]^\top \tag{4.6}$$

where p_x, p_y, ψ represents the vehicle's 2D pose, and v_x, v_y, ω_z represents the vehicle's

2D body velocity. To enable a simple expression of the road boundary constraint, and to assist with the linearization and normalization of dynamics models later on, we will mainly use a Frenet frame system [21] to express the 2D pose. This is illustrated in figure 4.1. A curvilinear Frenet frame works by denoting s, e_y, and e_ψ as the pose of the vehicle expressed w.r.t. a parametric path $\tau : [0, L] \mapsto \mathbb{R}^2$, which is assumed to be continuously twice differentiable and of total length L. In particular, when given an arclength $s \in [0, L]$, τ returns the global x-y position of the path. In the case where τ forms a closed circuit, i.e., $\tau(0) = \tau(L)$, $\tau'(0) = \tau'(L)$, $\tau''(0) = \tau''(L)$, where τ' and τ'' are the element-wise first and second derivatives of τ, we define the circuit as $\bar{\tau}(s) = \tau(\text{mod}(s, L))$ for $s \geq 0$, where mod is the modulo operator. Given a path τ and a vehicle with global position $p = (x, y)$ and heading ψ, the Frenet frame pose is defined as the vehicle's path progress $s(p) = \arg\min_s \|\tau(s) - p\|_2$, its lateral deviation from the path $e_y(p) = \min_s \|\tau(s) - p\|_2$, and its heading deviation from the path tangent $e_\psi(p, \psi) = \psi - \arctan(\tau'(s(p)))$.

$$x_f = [v_x, v_y, \omega_z, e_\psi, s, e_y]^\top \tag{4.7}$$

We will omit the subscript and write the vehicle state x_f as x in the rest of this chapter.

We further note that any dynamics represented in a global Cartesian coordinate system $(\dot{p_x}, \dot{p_y}, \dot{\psi})$ can be easily transformed into the Frenet frame $(\dot{s}, \dot{e_y}, \dot{e_\psi})$, without affecting the body velocity dynamics $(\dot{v_x}, \dot{v_y}, \dot{\omega_z})$ [16]. This can be achieved given the Frenet frame curvature k at the vehicle's location:

$$\dot{s} = \frac{v_x \cos\psi - v_y \sin\psi}{1 - e_y k} \tag{4.8a}$$

$$\dot{e_y} = v_x \sin\psi + v_y \cos\psi \tag{4.8b}$$

$$\dot{e_\psi} = \omega_z - k\frac{v_x \cos\psi - v_y \sin\psi}{1 - e_y k} \tag{4.8c}$$

This property of the Frenet frame allows the general expression of any robot dynamics.

The inputs to the model are the longitudinal acceleration a and the front steering

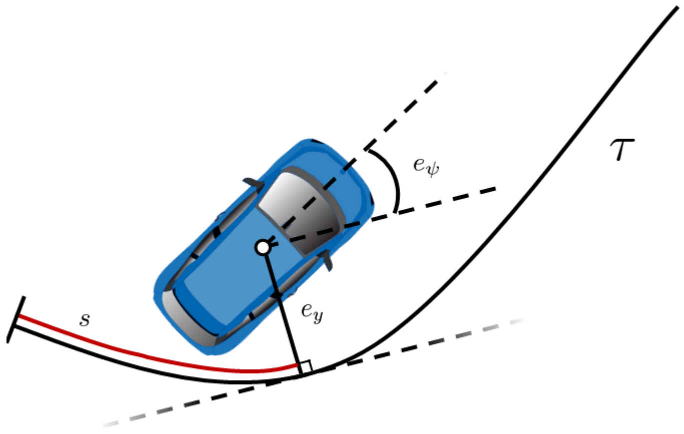

Figure 4.1: Illustration of the Frenet frame pose states.

angle δ.

$$u = [a, \delta]^\top \tag{4.9}$$

4.2.2 Prior Vehicle Dynamics Model

We employ a single-track vehicle dynamics model that is specifically designed to consider race car characteristics that have significantly more impact on vehicle performance than on passenger vehicles. This includes aerodynamic down-force and non-linear tire model. [16]

We start by describing the fundamental tire forces, considering a rear-wheel-drive, four-wheel-brake race car. The force calculated is per wheel and is decomposed into longitudinal and lateral tire forces. The front and rear longitudinal tire forces are

$$F_{x,f} = \frac{1}{2}(k_b m a_b - f_r m g \frac{l_r}{l}) \tag{4.10a}$$

$$F_{x,r} = \frac{1}{2}(m a_d + (1 - k_b) m a_b - f_r m g \frac{l_f}{l}) \tag{4.10b}$$

where k_b denotes the front brake bias (0-100%); f_r denotes rolling resistance; m denotes the vehicle mass; l denotes wheelbase; l_f, l_r denotes the distance between center-of-gravity and the two axes; a_d, a_b are derived from control variable a to enable the modeling of different longitudinal dynamics under driving and braking with an activation function:

$$a_d = a(0.5\tanh(a) + 0.5) \tag{4.11a}$$

$$a_b = a(0.5\tanh(-a) + 0.5) \tag{4.11b}$$

Then we describe normal tire loads considering longitudinal load transfer.

$$F_{z,f} = \frac{1}{2}(mg\frac{l_r}{l} - \frac{h_{\text{cog}}}{l}ma_x + f_D c_{l,f}) \tag{4.12a}$$

$$F_{z,r} = \frac{1}{2}(mg\frac{l_r}{l} + \frac{h_{\text{cog}}}{l}ma_x + f_D c_{l,r}) \tag{4.12b}$$

$$\text{where} \quad f_D = \frac{1}{2}C_D A v_x^2 \tag{4.12c}$$

$$a_x = a - \frac{f_D}{m} - f_r g \tag{4.12d}$$

h_{cog} denotes the center-of-gravity height; $c_{l,f}c_{l,r}$ denotes the lift coefficients at the axles; f_D denotes the air resistance; C_D denotes the drag coefficient; A denotes the frontal area of the car; a_x denotes the overall longitudinal acceleration.

We can now compute the lateral tire forces F_y via normal tire forces F_z using a pure lateral slip Pacejka tire model [2]. We take note that the pure lateral slip assumption will not hold when the race car is employing a "trail braking" technique and carrying longitudinal acceleration when cornering, but it is part of the EL-MPC's goal to learn such fine-grained control from data.

$$\alpha_f = \delta - \arctan((l_f\omega_z + v_y)/v_x) \tag{4.13a}$$

$$\alpha_r = \arctan((l_r\omega_z - v_y)/v_x) \tag{4.13b}$$

$$F_{y,f} = \mu F_{z,f}\sin(C_f \arctan(B_f a_f)) \tag{4.13c}$$

$$F_{y,r} = \mu F_{z,r}\sin(C_r \arctan(B_r a_r)) \tag{4.13d}$$

where α_f, α_r denotes tire side slip angles; μ denotes tire-road friction coefficient; C, B are Pacejka tire parameters.

Finally, these equations describe the evolution of state variables as a function of state variables (4.7) and control variables (4.9).

$$\dot{v}_x = \frac{1}{m}(2F_{x,r} + 2F_{x,f}\cos\delta - 2F_{y,f}\sin\delta - F_D) + \omega_z v_y \tag{4.14a}$$

$$\dot{v}_y = \frac{1}{m}(2F_{y,r} + 2F_{y,f}\cos\delta + 2F_{x,f}\sin\delta) - \omega_z v_y \tag{4.14b}$$

$$\dot{\omega}_z = \frac{1}{J_{zz}}((2F_{y,f}\cos\delta + 2F_{x,f}\sin\delta)l_f - 2F_{y,l}l_r) \tag{4.14c}$$

$$\dot{e}_\psi = \omega_z - k\frac{v_x\cos e_\psi - v_y\sin e_\psi}{1 - e_y k} \tag{4.14d}$$

$$\dot{s} = \frac{v_x\cos e_\psi - v_y\sin e_\psi}{1 - e_y k} \tag{4.14e}$$

$$\dot{e}_y = v_x\sin e_\psi + v_y\cos e_\psi \tag{4.14f}$$

where J_{zz} denotes the vehicle's yaw moment of inertia.

We note that despite the level of complication, the vehicle dynamics described by these parameters and equations are not accurate enough for safe high-speed handling at the limit. This is the aforementioned chicken-and-egg problem at play: Some parameters such as tire models and brake characteristics are not available before the controller has reached the high-speed region in the first place and, therefore, need to be roughly estimated. Bridging this gap would require the dynamics learning process to be formulated below. We also note that the kinematic part of the state evolution, namely $\dot{e}_\psi, \dot{s}, \dot{e}_y$, depends purely on the geometry of the vehicle and not on dynamics parameters such as weight or moment of inertia. Since the geometry of the vehicle can be measured rather accurately, there is very little practical motivation to employ dynamics learning on these state variables.

4.2.3 Learning MPC Problem

We write the discretized nonlinear dynamics as:

$$x_{k+1} = f(x_k, u_k) \tag{4.15}$$

The vehicle's state and input are subject to the constraints $\mathcal{X} = \{x \mid -W/2 \leq e_y \leq W/2\}$ and $\mathcal{U} = \{u \mid u_l \leq u \leq u_u\}$, which describe the boundary constraints for a track of constant width W and the limits on achievable acceleration and steering angle.

The objective of this work is to design a control policy which solves the infinite-horizon, minimium lap time, optimal control problem:

$$\begin{aligned} T = \min_{u_0,u_1,\ldots} \quad & \sum_{k=0}^{\infty} \mathbb{1}_{\mathcal{F}}(x_k) && \text{(4.16a)} \\ \text{subject to} \quad & x_0 = \bar{x}, && \text{(4.16b)} \\ & x_{k+1} = f(x_k, u_k), && \text{(4.16c)} \\ & x_k \in \mathcal{X},\ u_k \in \mathcal{U}, && \text{(4.16d)} \end{aligned}$$

where $\bar{x} \in \mathcal{S} = \{x \mid s = 0\}$ is a state at the start of the track and $\mathcal{F} = \{x \mid s \geq L\}$ is the set of states beyond the finish line of the circuit. If for some time step k, $x_k \in \mathcal{F}$ for the first time, then the vehicle has finished the lap and has transitioned onto the next lap. $\mathbb{1}_{\mathcal{F}}$ is the indicator function for the set $\mathcal{F}$, which is defined as

$$\mathbb{1}_{\mathcal{F}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{F} \\ 1 & \text{otherwise} \end{cases}$$

Due to the infinite horizon cost in (4.16a), it should be clear that the optimal control problem posed in (4.16) is not amenable to online implementation. Moreover, it is likely that there is mismatch between the vehicle model in (4.16c) and the true vehicle dynamics, the effects of which would be especially apparent at high speeds, where the vehicle is operating at the limit of handling. To address these challenges, previous work on LMPC [12, 40] proposed a finite-horizon convex approximation of (4.16), which uses state and input trajectories from prior laps to synthesize a convex terminal cost-to-go function and target set. The same data are additionally used to identify affine time-varying (ATV) models which approximate the true dynamics of the vehicle. In this work, we carry over the same approach for terminal cost-to-go function and target set synthesis, but propose a modification to the system identification procedure which results in an LMPC control policy that achieves similar performance to [12, 40] w.r.t. minimum lap time, while exhibiting a significant improvement in robustness to tuning parameters.

4.2.4 Local Error-Learning MPC

In this section, we briefly describe the procedure for synthesizing EL-MPC policies over iterations (or laps) of the minimum time control task. For further details, we refer the reader to [12, 40].

The main idea of EL-MPC is to use the data from iterations 1 to $j-1$ to synthesize a finite-horizon local convex approximation of (4.16) for iteration j, which is then solved in a receding horizon manner. Let us first denote the available dataset at iteration j as $\mathcal{D}^j = (\mathbf{x}^j, \mathbf{u}^j) \cup \mathcal{D}^{j-1}$, where $\mathbf{x}^j = \{x_0^j, x_1^j, \dots, x_{T^j}^j\}$ and $\mathbf{u}^j = \{u_0^j, u_1^j, \dots, u_{T^j}^j\}$ are the closed-loop state and input sequence for iteration j and T^j denotes the time step at which the lap was completed for iteration j, i.e., the first time step where $\mathbb{1}_{\mathcal{F}}(x_k) = 0$. Note that on the first iteration, the dataset $\mathcal{D}^0$ may be initialized using human-driven laps or closed-loop trajectories from a simple low-speed center-line tracking controller. In addition, we assume that the trajectories stored in $\mathcal{D}^j$ are feasible w.r.t. the constraints (4.16d) and reach the finish line in a finite amount of time.

Local Convex Target Set

As the closed-loop trajectories stored in $\mathcal{D}^j$ are from the actual vehicle, it is straightforward to see that if we can control the vehicle to any state in $\mathcal{D}^j$, then there exists a known feasible control sequence which can be used to complete the lap from that state. EL-MPC uses this fact to construct a safe set as a discrete collection of the states in $\mathcal{D}^j$, which is then used as the terminal set for a FHOCP [40]. In this work, we construct convex terminal sets which are local about a given state x. These terminal sets sacrifice the theoretical guarantee of persistent feasibility for the FHOCP in favor of reduced computational complexity for fast online control. In particular, for a given x, we take the K-nearest neighbors from each of the previous P laps by evaluating the weighted Euclidean distance over the relevant states:

$$\begin{aligned}(x_k^i - x)^\top D(x_k^i - x),\ \forall i \in \{j, j-1, \dots, j-(P-1)\},& \\ k \in \{0, 1, \dots, T^i\}& \end{aligned} \tag{4.17}$$

for some given P and $D \succeq 0$. Letting $\mathbf{X}^j(x;\mathcal{D}^j) \in \mathbb{R}^{n \times KP}$ denote the matrix formed by the KP states closest to x, we then define the target set as the convex hull of these states: $\mathcal{X}_N^j(x;\mathcal{D}^j) = \{\bar{x} \in \mathbb{R}^n \mid \exists \lambda \in \mathbb{R}^{KP},\ 0 \leq \lambda \leq 1,\ \mathbf{1}^\top \lambda = 1, \mathbf{X}^j(x;\mathcal{D}^j)\lambda = \bar{x}\}$.

Local Terminal Cost-to-go Function

By assumption, the closed-loop state trajectories stored in $\mathcal{D}^j$ reach the finish line in a finite amount of time. We can therefore compute the cost-to-go for any state x_k^i in $\mathcal{D}^j$ as $T^i - k$, which represents the time remaining to finish the i-th lap for $i \in \{0, \ldots, j\}$. Let $J_N^j(x;\mathcal{D}^j) \in \mathbb{R}^{KP}$ denote the vector of cost-to-go values calculated for the states in $\mathbf{X}^j(x;\mathcal{D}^j)$. We can therefore define the minimum cost-to-go function for any $\bar{x} \in \mathcal{X}_N^j(x;\mathcal{D}^j)$ as:

$$\begin{aligned} & Q_N^j(\bar{x}, x; \mathcal{D}^j) \\ & \quad = \min_{\lambda \in \mathbb{R}^{KP}} \ J_N^j(x;\mathcal{D}^j)^\top \lambda \\ & \quad \text{subject to} \ \ 0 \leq \lambda \leq 1,\ \mathbf{1}^\top \lambda = 1,\ \mathbf{X}^j(x;\mathcal{D}^j)\lambda = \bar{x} \end{aligned}$$

This function can be thought of as a local convex approximation to the optimal cost-to-go (from solving (4.16)) at iteration j, which when incorporated as the terminal cost-to-go function, allows an FHOCP to reason about infinite-horizon behavior of the vehicle, to the extent that it is captured in the dataset.

Local EL-MPC

Using the terminal target set and cost-to-go function synthesized from the dataset $\mathcal{D}^{j-1}$, we may now construct the FHOCP at time step k of iteration j for our local

LMPC policy as follows:

$$
\begin{aligned}
& J_k^j(x_k, u_{k-1}, \bar{\mathbf{z}}_k; \mathcal{D}^{j-1}) = \\
& \min_{\mathbf{x},\mathbf{u},\lambda} \quad \sum_{t=0}^{N-1} \mathbb{1}_{\mathcal{F}}(x_t) + c_u \|u_t\|_2^2 + c_{\Delta u}\|u_t - u_{t-1}\|_2^2 \\
& \qquad\qquad + J_N^{j-1}(\bar{x}_{k+N}; \mathcal{D}^{j-1})^\top \lambda && \text{(4.18a)} \\
& \text{subject to} \quad x_0 = x_k, \;\; u_{-1} = u_{k-1} && \text{(4.18b)} \\
& \qquad x_{t+1} = A(\bar{z}_{k+t}; \mathcal{D}^{j-1}) x_t + B(\bar{z}_{k+t}; \mathcal{D}^{j-1}) u_t \\
& \qquad\quad + C(\bar{z}_{k+t}; \mathcal{D}^{j-1}), \;\; t \in \{0, \dots, N-1\}, && \text{(4.18c)} \\
& \qquad x_t \in \mathcal{X}, \;\; u_t \in \mathcal{U}, \;\; t \in \{0, \dots, N\} && \text{(4.18d)} \\
& \qquad \mathbf{X}^{j-1}(\bar{x}_{k+N}; \mathcal{D}^{j-1})\lambda = x_N, && \text{(4.18e)} \\
& \qquad 0 \le \lambda \le 1, \;\; \mathbf{1}^\top \lambda = 1 && \text{(4.18f)}
\end{aligned}
$$

where $\bar{z}_k = (\bar{x}_k, \bar{u}_k)$ and $\bar{\mathbf{z}}_k = \{\bar{z}_k, \dots, \bar{z}_{k+N}\}$ is a state and input sequence which is used to form the local convex approximation. In practice, this is chosen as the solution to (4.18) from the previous time-step. Construction of the A, B, and C matrices in the ATV dynamics (4.18c) is the main contribution of this work and will be discussed in the next section. Note that (4.18) is a convex program which can be solved efficiently using existing solvers.

Finally, let $\mathbf{u}^\star = \{u_0^\star, \dots, u_{N-1}^\star\}$ be the optimal control sequence which solves (4.18) at time step k. The input $u_k = u_0^\star$ is applied to the vehicle and the FHOCP (4.18) is repeated at time step $k+1$ for the measured state x_{k+1} until the vehicle reaches the finish line, at which point $\mathcal{D}^j$ will be constructed using the closed-loop trajectory for lap j.

4.2.5 Learned Vehicle Dynamics Model

We now introduce the main contribution of our work, which is a learning strategy on the dynamics states v_x, v_y, ω_z based on regressive error. The learning process takes as inputs the nominal dynamics model and a set of neighboring states, and outputs a learned local affine approximation of the true vehicle dynamics. We assume that the true dynamics of the system can be written as follows

$$x^+ = f(x,u) + e(x,u), \tag{4.19}$$

where f are the nominal dynamics from (4.15) and e denotes some unknown modeling error which we would like to identify using data. For a given state and input x and u, let us denote the prediction of the nominal model as $\hat{x}^+ = f(x,u)$. We may now linearize the error dynamics about a reference state and input $\bar{z}$ as follows:

$$x^+ - \hat{x}^+ = A^e x + B^e u + C^e, \tag{4.20}$$

where A^e and B^e are the Jacobians of e w.r.t. x and u evaluated at $\bar{z}$ and $C^e = e(\bar{x},\bar{u}) - A^e\bar{x} - B^e\bar{u}$. It can be clearly seen from (4.20) that the linearization of the unknown error term is related to the system state and input x and u, the actual state evolution x^+, and the prediction of the nominal model $\hat{x}^+$, which are all quantities that can be easily obtained from the dataset $\mathcal{D}^{j-1}$ at iteration j. As such, we query $\mathcal{D}^{j-1}$ to find the M nearest neighbors $\mathbf{z} = \{z_1, \dots, z_M\}$ of $\bar{z}$ and their corresponding state evolutions $\mathbf{x}^+ = \{x_1^+, \dots, x_M^+\}$ (using a technique similar to (4.17)). We then compute the prediction of the nominal model at each of the state and input pairs in $\mathbf{z}$ to obtain $\hat{\mathbf{x}}^+ = \{\hat{x}_1^+, \dots, \hat{x}_M^+\}$. As mentioned before, we are interested in learning only the error dynamics associated with the velocity components of the state x, as the kinematic components of the state are well-understood. Therefore, using the datasets $\mathbf{z}$, $\mathbf{x}^+$, and $\hat{\mathbf{x}}^+$, we define the residuals for each velocity state as follows:

$$\begin{aligned}
A^e[11:13]x_m[1:3] + B^e[11]a_m + C^e[1] &= v_{x,m}^+ - \hat{v}_{x,m}^+, \\
A^e[21:23]x_m[1:3] + B^e[22]\delta_m + C^e[2] &= v_{y,m}^+ - \hat{v}_{y,m}^+, \\
A^e[31:33]x_m[1:3] + B^e[32]\delta_m + C^e[3] &= \omega_{z,m}^+ - \hat{\omega}_{z,m}^+,
\end{aligned} \tag{4.21}$$

where $m \in \{1, \dots, M\}$ and we use the notation $A[11:13]$ to index the first three elements of the first row of matrix A. Note that we have injected additional structure into the individual residuals by making explicit the dependence of each velocity state on a certain subset of the input components. From these three expressions, we define the following regressor vectors:

$$\Gamma_{v_x} = \begin{bmatrix} A^e[11{:}13] \\ B^e[11] \\ C^e[1] \end{bmatrix}, \Gamma_{v_y} = \begin{bmatrix} A^e[21{:}23] \\ B^e[22] \\ C^e[2] \end{bmatrix}, \Gamma_{\omega_z} = \begin{bmatrix} A^e[31{:}33] \\ B^e[32] \\ C^e[3] \end{bmatrix}.$$

We solve the regression problem in a manner similar to [42], where we weigh the importance of data point m using the Epanechnikov kernel function [20] with bandwidth h:

$$K(u) = \begin{cases} \frac{3}{4}(1 - u^2/h^2), & |u| < h \\ 0, & \text{otherwise} \end{cases}.$$

such that larger penalties are assigned to the residuals for data points that are close to the linearization point $\bar{x}$. For $l = \{v_x, v_y, \omega_z\}$, the weighted least squares problem is defined as follows:

$$\Gamma_l^\star = \underset{\Gamma_l}{\operatorname{argmin}} \sum_{m=1}^{M} K\left(||\bar{z} - z_m||_Q^2\right) y_m^l(\Gamma_l) + \epsilon\|\Gamma_l\|_2^2, \tag{4.22}$$

where $Q \succeq 0$ denotes the relative scaling between the variables, $y_m^l(\Gamma_l)$ are the l^2-norms of the residuals in (4.21), and $\epsilon > 0$ is a regularization parameter.

Finally, we can construct the matrices A^e, B^e, and C^e from $\Gamma_l^\star$ as follows:

$$A^e = \begin{bmatrix} \Gamma_{v_x}^\star[1:3] & \\ \Gamma_{v_y}^\star[1:3] & \mathbf{0}_{6\times3} \\ \Gamma_{\omega_z}^\star[1:3] & \\ \mathbf{0}_{3\times3} & \end{bmatrix}, \quad B^e = \begin{bmatrix} \Gamma_{v_x}^\star[4] & 0 \\ 0 & \Gamma_{v_y}^\star[4] \\ 0 & \Gamma_{\omega_z}^\star[4] \\ \multicolumn{2}{c}{\mathbf{0}_{3\times2}} \end{bmatrix},$$
$$C^e = \begin{bmatrix} \Gamma_{v_x}^\star[5] & \Gamma_{v_y}^\star[5] & \Gamma_{\omega_z}^\star[5] & \mathbf{0}_{1\times3} \end{bmatrix}^\top.$$

These are then added to the nominal ATV dynamics to obtain the complete ATV model for (4.18c): $A(\bar{z}; \mathcal{D}^{j-1}) = A^f + A^e$, $B(\bar{z}; \mathcal{D}^{j-1}) = B^f + B^e$, and $C(\bar{z}; \mathcal{D}^{j-1}) = C^f + C^e$, where A^f and B^f are the Jacobians of f w.r.t. x and u evaluated at $\bar{z}$ and $C^f = f(\bar{x}, \bar{u}) - A^f\bar{x} - B^f\bar{u}$. We note that the key difference between this procedure and that in [42] is that the regressive learning is performed on the *error* between the prediction of the nominal model and the actual state evolution data instead of on the entire model in (4.19).

To understand why our approach to model learning is more robust to scarce data, consider the case where all $\|\bar{z} - z_m\|_Q$ in (4.22) are close to or greater than the kernel bandwidth h (i.e., when the data z_m are far from the linearization point $\bar{z}$ w.r.t. h). When this occurs, $K(\|\bar{z} - z_m\|_Q^2) \approx 0$, which means that the regularization term in (4.22) dominates the least squares objective and we can therefore conclude that the optimal solution $\Gamma_l^\star \approx 0$. When regressing the nominal dynamics matrices directly, as in [42], this would result in setting elements of those matrices to approximately zero, which would be disastrous in terms of model accuracy. On the other hand, when regressing the error matrices, setting elements to zero would simply correspond to *not* applying any data-based correction to the nominal linearized dynamics. This allows for a natural trade-off between the nominal and data-corrected models based on the quality of the dataset. Of course, this could be mitigated by the selection of a large enough bandwidth h. However, as this is a tuning parameter which is meant to act as a threshold on a weighted distance metric in high dimensions, it may not be immediately apparent what constitutes a good setting. Therefore, we want the EL-MPC policy to be robust to values of h to allow for safe tuning of the controller.

4.3 Experiments and Results

We now present a set of experiments in simulation and hardware to demonstrate the effectiveness of EL-MPC's error dynamics regression in autonomous racing. The key metrics throughout these experiments are

- 20th-iteration lap time (ILT-20): the average lap time after running the LMPC and EL-MPC for 20 iterations.
- Iterations to fail (ITF): the average number of iterations before an experiment fails due to loss of control or track boundary violation.

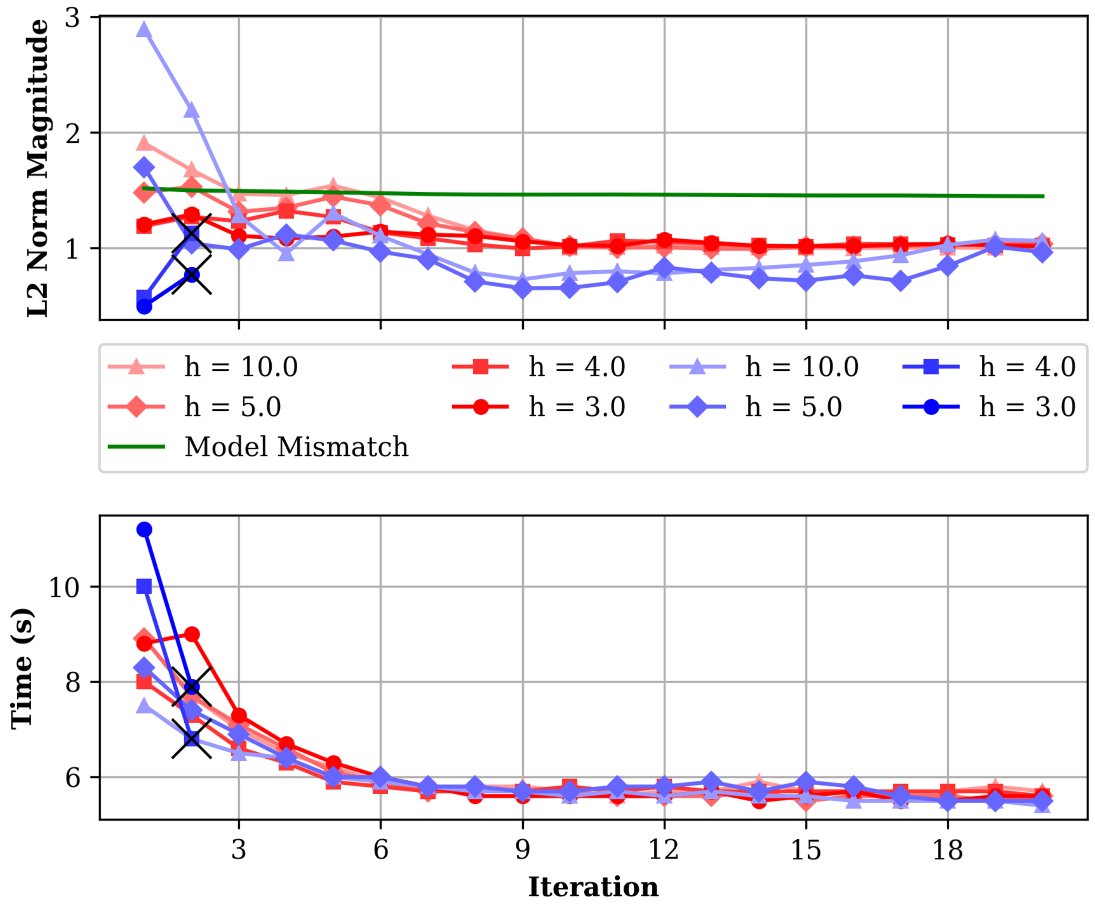

Figure 4.2: Average error of the learned model (Top) and lap times (bottom) over iterations for various settings of the bandwidth h. Red and blue lines correspond to the error regression and full regression cases, respectively. A cross is placed where failure occurs due to the vehicle leaving the track.

4.3.1 Robustness Study on Learning Parameters

This experiment aims to study how LMPC and EL-MPC parameter tunings affect the safety of the learning process. The motivation is that there are not many trial-and-error opportunities when tuning parameters on safety-critical hardware such as a race car, and a robust control setup should be able to handle a broad range of parameter settings without destabilizing the system. We set up simulation experiments to compare our approach against the LMPC implementation in [42] in terms of their robustness to the change in the bandwidth parameter h and the control-rate cost (CRC) ($c_{\Delta u}$ in (4.18a)). Intuitively, the bandwidth h governs a tradeoff between the quality and quantity of the data used for model regression. Whereas a high bandwidth allows for potentially more data points to be selected, they can be far (in the sense of the norm $\|\cdot\|_Q$) from the linearization point. On the other hand, a low bandwidth requires that the data points be close to the linearization point, but could result in few data points being selected when data is sparse. As for the CRC, it is analogous to the "learning rate" for EL-MPC. A higher CRC will penalize drastic changes in the control input and keep the vehicle closer to the safe set over the MPC horizon. Both h and the CRC are therefore key parameters when tuning the behavior of the EL-MPC policy. For all simulation studies, we use the L-shaped track shown in Figure 4.5 and model the rigid body with the dynamic bicycle model. The tire forces are modeled with the Pacejka tire model [2] with tire-road friction coefficient $\mu = 0.9$.

In our first simulation experiment, we set CRC $= 0.1$ and compare the effect of the bandwidth h on the accuracy of the learned model for our error regression EL-MPC and the full regression LMPC from [42]. In the error regression case, we use the kinematic bicycle model as the nominal model in (4.15), which induces mismatch with the simulation model. To measure accuracy of the learned model, we compute the Frobenius norm of the difference between the learned dynamics matrix $A(\bar{z}, \mathcal{D}^{j-1})$ and the linearized dynamics of the simulation model at each time step and record the average over each iteration. The results for $h = \{3, 4, 5, 10\}$ are shown in Figure 4.2, where we may immediately observe that for high settings of h, both the error and full regression cases perform similarly in both the learned model accuracy and lap times. Importantly, we see that the error regression case can correct for the mismatch between the nominal and simulation models, which is shown by the green line. For low settings of h, it can be seen that while [42] fails after the second iteration, the error regression EL-MPC is able to remain stable and converge to a similar performance to that of the high-bandwidth cases. This is due to the reasons discussed in Section 4.2.5, where sparsity w.r.t. h can result in significant model inaccuracies in [42], but would naturally induce a fallback to the nominal model in the error regression case. Sparsity in the regression data is especially acute in the initial laps when the LMPC policy is rapidly improving vehicle performance, as there can be significant differences in the data between successive laps. We note that the behavior of the error regression case is largely dependent on the accuracy of the nominal model when data is sparse and stability cannot be guaranteed for arbitrary realizations of the nominal model. However, especially in car racing, it is not unreasonable to assume access to a fairly accurate nominal model, which can be obtained through standard system identification approaches. Future work will examine the effect of distance between the nominal and true system on the stability of the EL-MPC.

In our second simulation experiment, we fixed the bandwidth to $h = 5$, ran both LMPC and EL-MPC implementations with CRC $= \{1.0, 0.5, 0.1, 0.05, 0.01\}$, and recorded the lap time at every iteration. Like the previous study, we purposely introduce mismatch in the nominal model, though for this study, we do so through the tire-road friction coefficient. Whereas a value of 0.9 is used in the simulation model, a value of 1.2 is chosen for the EL-MPC nominal dynamics model in (4.15), which is typically only seen on full-scale race cars with racing slick tires. This would have induced dangerous over-confidence in the model about the surface friction, as shown in the following hardware experiments, if no dynamics learning is used. Table 4.1 presents the key results of this experiment, where we observe that with CRC penalties of 1.0, 0.5, and 0.1, both [42] and our approach show similar performance at the end of 20 laps, with [42] achieving slightly faster lap time reduction, as evidenced in Figure 4.3. However, once we decrease the CRC below 0.1, we see that [42] fails to complete the 20 laps, whereas our method exhibits greater robustness to different settings of the CRC and can still converge to high-performing lap times within 20 iterations.

Overall, our simulation experiments suggest that EL-MPC with error dynamics regression exhibits greater stability and robustness to key parameter settings and works well even when scarce dynamics data are available, compared to LMPC. These properties could potentially make the system more suitable for hardware deployment in safety-critical applications.

4.3.2 Hardware Experiment Comparing Learning Performance

In this section, we present results from physical experiments with the Berkeley Autonomous Race Car (BARC) platform, which closely mimics the original F1TENTH vehicle system presented in Chapter 2. EL-MPC computation is done on a laptop

Control Rate Cost	ILT-20		ITF	
	[42]	ours	[42]	ours
1.0	**6.4**	6.5	20+	20+
0.5	**6.0**	6.2	20+	20+
0.1	**5.5**	5.6	20+	20+
0.05	-	**5.2**	6	**20+**
0.01	-	**5.0**	4	**20+**

Table 4.1: Results of the simulation robustness study. A hyphen in the 20th-Iteration Lap Time (ILT-20) means that the algorithm fails before the 20th iteration. An Iteration to Fail (ITF) of 20+ means that the algorithm did not fail in the 20-iteration experiment.

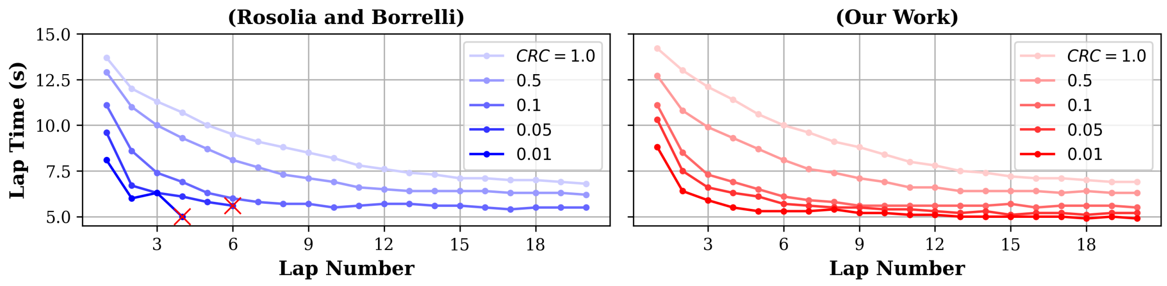

Figure 4.3: Results from the CRC robustness study. The left and right plots show the iteration lap times over different CRC tunings for [42] and our EL-MPC respectively. A red cross is placed where failure occurs due to the vehicle leaving the track.

computer with a 2.6 GHz 9th-Gen Intel Core i7 CPU, localization is provided by a motion capture system, and communication with the vehicle is handled with ROS2. We conducted two sets of experiments to demonstrate the effect of the bandwidth h on the stability and learning performance of [42] and our error regression EL-MPC. Each set of experiments consists of 10 runs of at most 20 laps with CRC $= 0.1$. We record the lap times and record any failure cases before the 20-lap threshold.

In the first experiment, we run [42] and our EL-MPC with $h = 10$. This corresponds to the high-bandwidth case in the simulation study, where both policies achieved similar performance and were able to successfully complete 20 laps. The results of this experiment are shown in the top row of plots in Figure 4.4 with [42] on the left and ours on the right, where we again see that the two approaches show similar performance in lap time reduction and are able to remain stable over 20

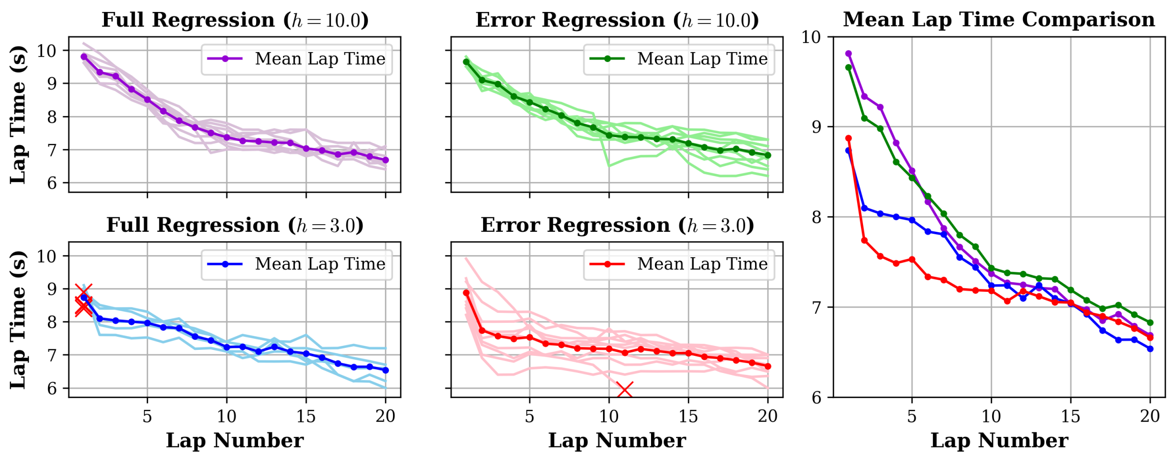

Figure 4.4: Per-iteration results of the BARC hardware experiment. The four smaller plots show the lap-time reduction of the 10 trials. The purple and blue plots on the left are with [42], whereas the green and red plots on the right are with EL-MPC. The combined plot on the right compares the average lap times achieved. A red cross is placed where failure occurs due to the vehicle leaving the track. Note that the blue plot has five overlapping failure marks.

Controller Configuration	**Avg. ILT-20**	**Avg. ITF**
Full Regression ([42], $h = 10.0$)	6.69	20+
Error Regression (ours, $h = 10.0$)	6.82	20+
Full Regression ([42], $h = 3.0$)	6.53	10.5
Error Regression (ours, $h = 3.0$)	6.66	19.1

Table 4.2: Results of the BARC hardware experiment.

laps. This corroborates our simulation study and shows that with proper tuning, our EL-MPC is essentially equivalent to [42]. In the second experiment, we run [42] and our EL-MPC with $h = 3$. This corresponds to the low-bandwidth case in the simulation study, where data sparsity has a destabilizing effect on [42]. The results of our experiment are shown in the bottom row of plots in Figure 4.4, which also supports our observations from the simulation study. In particular, we observe that when a low-bandwidth setting is used, [42] failed in 5 of the 10 trials, whereas our EL-MPC only failed in a single trial.

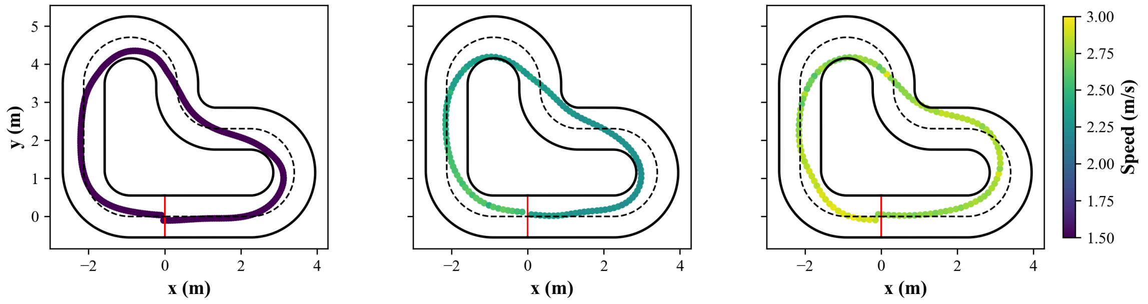

Figure 4.5: Visualization of the 1st, 5th and 20th iteration's trajectory of EL-MPC with error dynamics regression on the BARC platform.

4.3.3 Hardware Demonstration on a Full-Size Race Car

In this demonstration, we deployed our EL-MPC and error dynamics regression strategy on the AV-21 race car [51], which is the primary competition vehicle for Indy Autonomous Challenge as stated in Chapter 1. The race track used for this demonstration is the Putnam Park Road Course in Indiana, USA. We construct the initial dataset $\mathcal{D}^0$ for the vehicle by running a tracking MPC on the center line of the track at 10 m/s for 3 laps. We note that this step could also be done with a more rudimentary controller. We then ran EL-MPC with error dynamics regression for 10 iterations. Figure 4.6 visualizes the trajectory driven by the vehicle, with a zoomed-in view to highlight how EL-MPC optimizes the shape of the trajectory for a hairpin turn. It can be clearly seen in Figure 4.6 that the vehicle starts at lower speeds closer to the center line and incrementally increases its speed through the corners and approaches the track boundary constraint.

4.4 Summary

In this chapter, we proposes a new error dynamics learning approach under the EL-MPC framework to iteratively and safely explore the boundary of vehicle handling limit in autonomous racing. We based the learning on a nominal dynamics model, which provides explanability and safety fallback for the learning process. We then derived the regression method on top to learn a local approximation of the nonlinear model mismatch. Through simulation and hardware experiments, we showed that our method exhibits greater robustness against parameter tuning and data scarcity

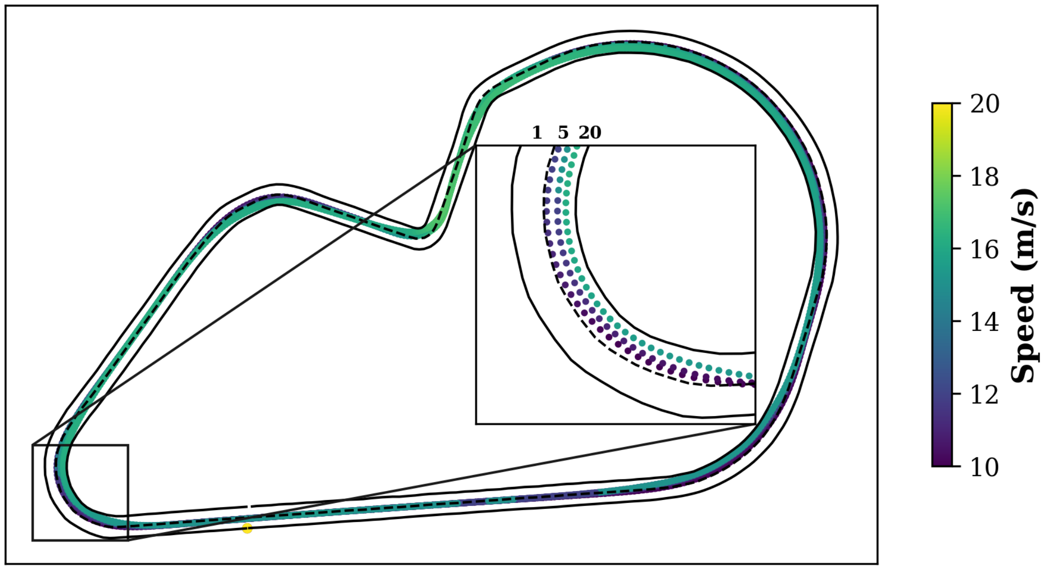

Figure 4.6: Overlay of the IAC car's trajectory running EL-MPC with error dynamics regression. The data are collected at the Putnam Park Road Course in Indiana, USA. The zoomed-in image of the hairpin turn shows iterative improvement towards optimal cornering in the 1st, 5th, and 20th lap.

compared with previous LMPC works, and offers new potential solutions to address the challenge of vehicle dynamics data acquisition in high-speed autonomous racing.

We highlight the application of Spline-Opt in this chapter. In the experiment section, in order to capture the entire dynamics and control policy learning process from an unbiased prior, we have chosen to use the centerline of the race tracks as the tracking reference to form the initial dataset $\mathcal{D}^0$. But we could also use the result of Spline-Opt as the initial reference, which hypothetically would lead to faster convergence of the learning algorithm, and higher likelihood in finding the globally optimal trajectory.

Chapter 5

Conclusions

This thesis introduces a pipeline of robot learning frameworks driven by optimal control in autonomous racing. They are synthesized in Figure 5.1. During the offline process, we use Spline-Opt to convert an un-optimized reference, such as the centerline of a trajectory, into a curvature-optimal, fully-parameterized, and twice-differentiable trajectory. This process relies on minimum dynamics data available to the user, and produces a reasonably good initial guess for the true time-optimal trajectory. We then run the EL-MPC online, which iteratively learns the locally approximated error dynamics, and stores them as a dataset for querying. This allows the Learning MPC policy to explore a convex safe set around the current-iteration trajectory, and improve the vehicle lap time while considering the safety of the handling.

Future work would look into converting Spline-Opt into a tangible online trajectory planning method, thanks to its superiority in optimization speed compared to discrete trajectory optimization. In addition, the current EL-MPC formulation, like its predecessor LMPC, fails to reason about changing dynamic conditions, as the dataset of previous iterations cannot rapidly cope with sudden changes in the environment. Deep adaptation based on [45] could be explored to let EL-MPC policy continue to perform under time-varying dynamic conditions.

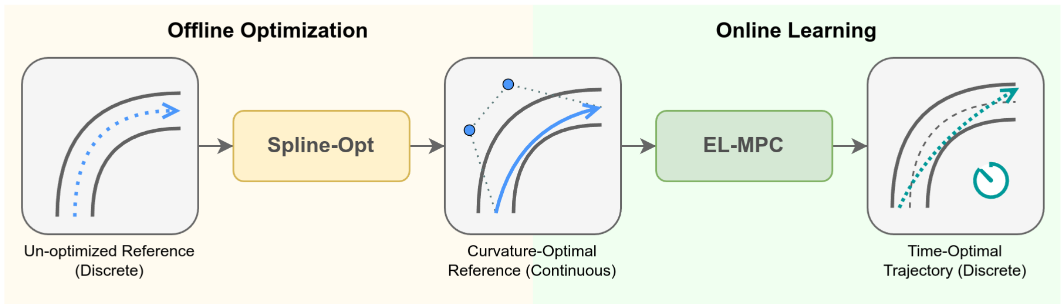
Offline Optimization
Online Learning
Spline-Opt
EL-MPC
Un-optimized Reference (Discrete)
Curvature-Optimal Reference (Continuous)
Time-Optimal Trajectory (Discrete)

Figure 5.1: A pipeline combining Spline-Opt with EL-MPC for offline trajectory optimization and online learning of dynamics and control policy.

Bibliography

[1] Indy Autonomous Challenge - Official Website, February 2023. URL https://www.indyautonomouschallenge.com. 1.2.1

[2] Egbert Bakker, Lars Nyborg, and Hans B. Pacejka. Tyre Modelling for Use in Vehicle Dynamics Studies. *SAE Transactions*, 96:190–204, 1987. ISSN 0096-736X. URL https://www.jstor.org/stable/44470677. Publisher: SAE International. 4.2.2, 4.3.1

[3] Suneel Belkhale, Rachel Li, Gregory Kahn, Rowan McAllister, Roberto Calandra, and Sergey Levine. Model-Based Meta-Reinforcement Learning for Flight With Suspended Payloads. *IEEE Robotics and Automation Letters*, 6(2):1471–1478, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2021.3057046. URL https://ieeexplore.ieee.org/document/9345959. Conference Name: IEEE Robotics and Automation Letters. 2.1

[4] Luigi Berducci, Shuo Yang, Rahul Mangharam, and Radu Grosu. Learning Adaptive Safety for Multi-Agent Systems, October 2023. URL http://arxiv.org/abs/2309.10657. arXiv:2309.10657 [cs, eess]. 1.2.2

[5] Johannes Betz, Hongrui Zheng, Alexander Liniger, Ugo Rosolia, Phillip Karle, Madhur Behl, Venkat Krovi, and Rahul Mangharam. Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing. *IEEE Open Journal of Intelligent Transportation Systems*, 3:458–488, 2022. ISSN 2687-7813. doi: 10.1109/OJITS.2022.3181510. URL https://ieeexplore.ieee.org/document/9790832/. 1.2.3

[6] Johannes Betz, Tobias Betz, Felix Fent, Maximilian Geisslinger, Alexander Heilmeier, Leonhard Hermansdorfer, Thomas Herrmann, Sebastian Huch, Phillip Karle, Markus Lienkamp, Boris Lohmann, Felix Nobis, Levent Ögretmen, Matthias Rowold, Florian Sauerbeck, Tim Stahl, Rainer Trauth, Frederik Werner, and Alexander Wischnewski. TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge. *Journal of Field Robotics*, 40(4):783–809, 2023. ISSN 1556-4967. doi: 10.1002/rob.22153. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.22153. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.22153. 2.2

[7] Francesco Borrelli. *Constrained Optimal Control of Linear and Hybrid Systems*, volume 290 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, Heidelberg, 2003. ISBN 978-3-540-00257-4. doi: 10.1007/3-540-36225-8. URL `http://link.springer.com/10.1007/3-540-36225-8`. 4.1.2

[8] F. Braghin, F. Cheli, S. Melzi, and E. Sabbioni. Race driver model. *Computers & Structures*, 86(13):1503–1516, July 2008. ISSN 0045-7949. doi: 10.1016/j.compstruc.2007.04.028. URL `https://www.sciencedirect.com/science/article/pii/S0045794908000163`. 2.3, 3.2.4

[9] D L Brayshaw and M F Harrison. A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 219(6):725–739, June 2005. ISSN 0954-4070. doi: 10.1243/095440705X11211. URL `https://doi.org/10.1243/095440705X11211`. Publisher: IMECHE. 2.3

[10] D L Brayshaw and M F Harrison. Use of numerical optimization to determine the effect of the roll stiffness distribution on race car performance. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 219(10):1141–1151, October 2005. ISSN 0954-4070. doi: 10.1243/095440705X34900. URL `https://doi.org/10.1243/095440705X34900`. Publisher: IMECHE. 2.3

[11] D.A. Bristow, M. Tharayil, and A.G. Alleyne. A survey of iterative learning control. *IEEE Control Systems Magazine*, 26(3):96–114, June 2006. ISSN 1941-000X. doi: 10.1109/MCS.2006.1636313. URL `https://ieeexplore.ieee.org/document/1636313`. Conference Name: IEEE Control Systems Magazine. 2.2

[12] Maximilian Brunner, Ugo Rosolia, Jon Gonzales, and Francesco Borrelli. Repetitive learning model predictive control: An autonomous racing example. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2545–2550, December 2017. doi: 10.1109/CDC.2017.8264027. URL `https://ieeexplore.ieee.org/document/8264027`. 2.2, 4.2.3, 4.2.4

[13] Tim Brüdigam, Alexandre Capone, Sandra Hirche, Dirk Wollherr, and Marion Leibold. Gaussian Process-based Stochastic Model Predictive Control for Overtaking in Autonomous Racing, May 2021. URL `http://arxiv.org/abs/2105.12236`. arXiv:2105.12236 [cs]. 2.2

[14] Monimoy Bujarbaruah, Xiaojing Zhang, Ugo Rosolia, and Francesco Borrelli. Adaptive MPC for Iterative Tasks. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 6322–6327, December 2018. doi: 10.1109/CDC.2018.8618694. URL `https://ieeexplore.ieee.org/document/8618694`. ISSN: 2576-2370. 2.2

[15] Danilo Cambiaghi, Marco Gadola, Luca Manzo, and David Vetturi. A Tool for Lap Time Simulation. doi: 10.4271/962529. 2.3

[16] Fabian Christ, Alexander Wischnewski, Alexander Heilmeier, and Boris Lohmann. Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients. *Vehicle System Dynamics*, 59(4):588–612, April 2021. ISSN 0042-3114. doi: 10.1080/00423114.2019.1704804. URL `https://doi.org/10.1080/00423114.2019.1704804`. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00423114.2019.1704804. 4.2.1, 4.2.2

[17] Carl d. Boor. *A Practical Guide to Splines*. Springer Verlag, New York, 1978. 3.2.4

[18] Nicola Dal Bianco, Enrico Bertolazzi, Francesco Biral, and Matteo Massaro. Comparison of direct and indirect methods for minimum lap time optimal control problems. *Vehicle System Dynamics*, 57(5):665–696, May 2019. ISSN 0042-3114, 1744-5159. doi: 10.1080/00423114.2018.1480048. URL `https://www.tandfonline.com/doi/full/10.1080/00423114.2018.1480048`. 2.3

[19] P Dierckx. Algorithms for smoothing data with periodic and parametric splines. *Computer Graphics and Image Processing*, 20(2):171–184, October 1982. ISSN 0146-664X. doi: 10.1016/0146-664X(82)90043-0. URL `https://www.sciencedirect.com/science/article/pii/0146664X82900430`. 3.2.1

[20] V. A. Epanechnikov. Non-Parametric Estimation of a Multivariate Probability Density. *Theory of Probability & Its Applications*, 14(1):153–158, January 1969. ISSN 0040-585X. doi: 10.1137/1114019. URL `https://epubs.siam.org/doi/abs/10.1137/1114019`. Publisher: Society for Industrial and Applied Mathematics. 4.2.5

[21] Matthias Faessler, Antonio Franchi, and Davide Scaramuzza. Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories. *IEEE Robotics and Automation Letters*, 3(2):620–626, April 2018. ISSN 2377-3766. doi: 10.1109/LRA.2017.2776353. URL `https://ieeexplore.ieee.org/document/8118153`. Conference Name: IEEE Robotics and Automation Letters. 2.1, 4.2.1

[22] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, December 2014. ISSN 1867-2957. doi: 10.1007/s12532-014-0071-1. URL `https://doi.org/10.1007/s12532-014-0071-1`. 3.2.4

[23] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1126–1135,

Sydney, NSW, Australia, August 2017. JMLR.org. 2.1

[24] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, July 2021. doi: 10.1126/scirobotics.abh1221. URL `https://www.science.org/doi/10.1126/scirobotics.abh1221`. Publisher: American Association for the Advancement of Science. 2.1

[25] Joseph Funke, Paul Theodosis, Rami Hindiyeh, Ganymed Stanek, Krisada Kritatakirana, Chris Gerdes, Dirk Langer, Marcial Hernandez, Bernhard Müller-Bessler, and Burkhard Huhnke. Up to the limits: Autonomous Audi TTS. In *2012 IEEE Intelligent Vehicles Symposium*, pages 541–547, June 2012. doi: 10.1109/IVS.2012.6232212. ISSN: 1931-0587. 2.2

[26] Alexander Heilmeier, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp, and Boris Lohmann. Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 58(10):1497–1527, October 2020. ISSN 0042-3114, 1744-5159. doi: 10.1080/00423114.2019.1631455. URL `https://www.tandfonline.com/doi/full/10.1080/00423114.2019.1631455`. (document), 2.3, 3.1, 3.2.4, 3.2.4, 3.5, 3.3, 3.3, 3.4

[27] Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars. In *2018 European Control Conference (ECC)*, pages 1341–1348, June 2018. doi: 10.23919/ECC.2018.8550162. 2.2

[28] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(09):5149–5169, September 2022. ISSN 0162-8828. doi: 10.1109/TPAMI.2021.3079209. URL `https://www.computer.org/csdl/journal/tp/2022/09/09428530/1twaJR3AcJW`. Publisher: IEEE Computer Society. 2.1

[29] Tammo Jan Dijkema. B-Spline Basis Function, March 2011. URL `http://demonstrations.wolfram.com/BSplineBasisFunctions/`. (document), 3.1

[30] Juraj Kabzan, Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Learning-Based Model Predictive Control for Autonomous Racing. *IEEE Robotics and Automation Letters*, 4(4):3363–3370, October 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2926677. URL `https://ieeexplore.ieee.org/document/8754713`. Conference Name: IEEE Robotics and Automation Letters. 2.2

[31] Juraj Kabzan, Miguel I. Valls, Victor J. F. Reijgwart, Hubertus F. C. Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, Ankit Dhall, Eugenio Chisari, Napat Karnchanachari,

Sonja Brits, Manuel Dangel, Inkyu Sa, Renaud Dubé, Abel Gawel, Mark Pfeiffer, Alexander Liniger, John Lygeros, and Roland Siegwart. AMZ Driverless: The full autonomous racing system. *Journal of Field Robotics*, 37(7):1267–1294, 2020. ISSN 1556-4967. doi: 10.1002/rob.21977. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21977. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21977. 2.2

[32] Dvij Kalaria, Qin Lin, and John M. Dolan. Delay-aware Robust Control for Safe Autonomous Driving. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 1565–1571, June 2022. doi: 10.1109/IV51971.2022.9827111. URL https://ieeexplore.ieee.org/document/9827111. 2.2

[33] Dvij Kalaria, Qin Lin, and John M. Dolan. Adaptive Planning and Control with Time-Varying Tire Models for Autonomous Racing Using Extreme Learning Machine, March 2023. URL http://arxiv.org/abs/2303.08235. arXiv:2303.08235 [cs]. ??

[34] Dvij Kalaria, Qin Lin, and John M. Dolan. Towards Optimal Head-to-head Autonomous Racing with Curriculum Reinforcement Learning, August 2023. URL http://arxiv.org/abs/2308.13491. arXiv:2308.13491 [cs]. ??, 1.2.2

[35] Nitin R. Kapania, John Subosits, and J. Christian Gerdes. A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories. *Journal of Dynamic Systems, Measurement, and Control*, 138(9), June 2016. ISSN 0022-0434. doi: 10.1115/1.4033311. URL https://doi.org/10.1115/1.4033311. 2.3

[36] Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to Adapt in Dynamic, Real-World Environments Through Meta-Reinforcement Learning, February 2019. URL http://arxiv.org/abs/1803.11347. arXiv:1803.11347 [cs, stat]. 2.1

[37] Jingyun Ning and Madhur Behl. Vehicle Dynamics Modeling for Autonomous Racing Using Gaussian Processes, June 2023. URL http://arxiv.org/abs/2306.03405. arXiv:2306.03405 [cs]. 2.2

[38] Michael O'Connell, Guanya Shi, Xichen Shi, Kamyar Azizzadenesheli, Anima Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural-Fly enables rapid learning for agile flight in strong winds. *Science Robotics*, 7(66):eabm6597, May 2022. doi: 10.1126/scirobotics.abm6597. URL https://www.science.org/doi/10.1126/scirobotics.abm6597. Publisher: American Association for the Advancement of Science. 2.1

[39] Matthew O'Kelly, Hongrui Zheng, Dhruv Karthik, and Rahul Mangharam. F1TENTH: An Open-source Evaluation Environment for Continuous Control and Reinforcement Learning. In *Proceedings of the NeurIPS 2019 Compe-*

tition and Demonstration Track, pages 77–89. PMLR, August 2020. URL https://proceedings.mlr.press/v123/o-kelly20a.html. ISSN: 2640-3498. (document), 1.2.3, 1.1

[40] Ugo Rosolia and Francesco Borrelli. Learning Model Predictive Control for Iterative Tasks: A Computationally Efficient Approach for Linear System. *IFAC-PapersOnLine*, 50(1):3142–3147, July 2017. ISSN 2405-8963. doi: 10.1016/j.ifacol.2017.08.324. URL https://www.sciencedirect.com/science/article/pii/S2405896317306523. 1.2.2, 2.2, 4.1.1, 4.1.2, 4.2.3, 4.2.4, 4.2.4

[41] Ugo Rosolia and Francesco Borrelli. Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, July 2018. ISSN 1558-2523. doi: 10.1109/TAC.2017.2753460. Conference Name: IEEE Transactions on Automatic Control. **??**

[42] Ugo Rosolia and Francesco Borrelli. Learning How to Autonomously Race a Car: A Predictive Control Approach. *IEEE Transactions on Control Systems Technology*, 28(6):2713–2719, November 2020. ISSN 1558-0865. doi: 10.1109/TCST.2019.2948135. Conference Name: IEEE Transactions on Control Systems Technology. (document), 2.2, 4.2.1, 4.2.5, 4.2.5, 4.3.1, **??**, **??**, 4.3, 4.3.2, 4.4, **??**, **??**

[43] András Sasfi, Melanie N. Zeilinger, and Johannes Köhler. Robust adaptive MPC using control contraction metrics. *Automatica*, 155:111169, September 2023. ISSN 0005-1098. doi: 10.1016/j.automatica.2023.111169. URL https://www.sciencedirect.com/science/article/pii/S0005109823003308. 2.2

[44] Florian Sauerbeck, Sebastian Huch, Felix Fent, Phillip Karle, Dominik Kulmer, and Johannes Betz. Learn to See Fast: Lessons Learned From Autonomous Racing on How to Develop Perception Systems. *IEEE Access*, 11:44034–44050, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3272750. URL https://ieeexplore.ieee.org/document/10114934. Conference Name: IEEE Access. (document), 1.2

[45] Guanya Shi, Xichen Shi, Michael O'Connell, Rose Yu, Kamyar Azizzadenesheli, Animashree Anandkumar, Yisong Yue, and Soon-Jo Chung. Neural Lander: Stable Drone Landing Control Using Learned Dynamics. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9784–9790, May 2019. doi: 10.1109/ICRA.2019.8794351. URL https://ieeexplore.ieee.org/document/8794351. ISSN: 2577-087X. 2.1, 5

[46] Guanya Shi, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Neural-Swarm: Decentralized Close-Proximity Multirotor Control Using Learned Interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247, May 2020. doi: 10.1109/ICRA40945.2020.9196800. URL

https://ieeexplore.ieee.org/document/9196800. ISSN: 2577-087X. 2.1

[47] Guanya Shi, Wolfgang Hönig, Xichen Shi, Yisong Yue, and Soon-Jo Chung. Neural-Swarm2: Planning and Control of Heterogeneous Multirotor Swarms Using Learned Interactions. *IEEE Transactions on Robotics*, 38(2):1063–1079, April 2022. ISSN 1941-0468. doi: 10.1109/TRO.2021.3098436. URL https://ieeexplore.ieee.org/document/9508420. Conference Name: IEEE Transactions on Robotics. 2.1

[48] Blake Siegler, Andrew Deakin, and David Crolla. Lap Time Simulation: Comparison of Steady State, Quasi- Static and Transient Racing Car Cornering Strategies. *Proceedings of the 2000 SAE Motorsports Engineering Conference & Exposition*, page 361, November 2000. doi: 10.4271/2000-01-3563. 2.3

[49] Xingyou Song, Yuxiang Yang, Krzysztof Choromanski, Ken Caluwaerts, Wenbo Gao, Chelsea Finn, and Jie Tan. Rapidly Adaptable Legged Robots via Evolutionary Meta-Learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776, October 2020. doi: 10.1109/IROS45743.2020.9341571. URL https://ieeexplore.ieee.org/document/9341571. ISSN: 2153-0866. 2.1

[50] Patricia Ventura Diaz and Steven Yoon. High-Fidelity Computational Aerodynamics of Multi-Rotor Unmanned Aerial Vehicles. In *2018 AIAA Aerospace Sciences Meeting*, AIAA SciTech Forum. American Institute of Aeronautics and Astronautics, January 2018. doi: 10.2514/6.2018-1266. URL https://arc.aiaa.org/doi/10.2514/6.2018-1266. 2.1

[51] Alexander Wischnewski, Maximilian Geisslinger, Johannes Betz, Tobias Betz, Felix Fent, Alexander Heilmeier, Leonhard Hermansdorfer, Thomas Herrmann, Sebastian Huch, Phillip Karle, Felix Nobis, Levent Ögretmen, Matthias Rowold, Florian Sauerbeck, Tim Stahl, Rainer Trauth, Markus Lienkamp, and Boris Lohmann. Indy Autonomous Challenge - Autonomous Race Cars at the Handling Limits. In Peter Pfeffer, editor, *12th International Munich Chassis Symposium 2021*, pages 163–182. Springer Berlin Heidelberg, Berlin, Heidelberg, 2022. ISBN 978-3-662-64549-9 978-3-662-64550-5. doi: 10.1007/978-3-662-64550-5_10. URL https://link.springer.com/10.1007/978-3-662-64550-5_10. Series Title: Proceedings. 4.3.3

[52] Alexander Wischnewski, Thomas Herrmann, Frederik Werner, and Boris Lohmann. A Tube-MPC Approach to Autonomous Multi-Vehicle Racing on High-Speed Ovals. *IEEE Transactions on Intelligent Vehicles*, 8(1):368–378, January 2023. ISSN 2379-8904. doi: 10.1109/TIV.2022.3169986. Conference Name: IEEE Transactions on Intelligent Vehicles. 2.2

[53] Edward L. Zhu, Finn Lukas Busch, Jake Johnson, and Francesco Borrelli. A Gaussian Process Model for Opponent Prediction in Autonomous Racing, March

2023. URL http://arxiv.org/abs/2204.12533. arXiv:2204.12533 [cs, eess]. ??, 2.2