Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Safe Motion Planning for Autonomous Driving using an Adversarial Road Model

Alexander Liniger
CVL, ETH Zurich, Switzerland
Email: alex.liniger@vision.ee.ethz.ch

Luc Van Gool
CVL, ETH Zurich, Switzerland
PSI, Katholieke Universiteit Leuven, Belgium
Email: vangool@vision.ee.ethz.ch

***Abstract*—This paper presents a game-theoretic path-following formulation where the opponent is an adversary road model. This formulation allows us to compute safe sets using tools from viability theory, that can be used as terminal constraints in an optimization-based motion planner. Based on the adversary road model, we first derive an analytical discriminating domain, which even allows guaranteeing safety in the case when steering rate constraints are considered. Second, we compute the discriminating kernel and show that the output of the gridding based algorithm can be accurately approximated by a fully connected neural network, which can again be used as a terminal constraint. Finally, we show that by using our proposed safe sets, an optimization-based motion planner can successfully drive on city and country roads with prediction horizons too short for other baselines to complete the task.**

I. Introduction

Over the last few decades, self-driving cars have improved dramatically, from the beginnings in the late 80's [10], to the DARPA challenges [6, 7], to the autonomous cars of today, that can autonomously navigate public roads.

The drastic advances were achieved by improving the sensors but also the algorithms within the software stack of an autonomous car. The software stack of an autonomous car can be roughly split into four layers: localization, perception, planning, and control. In this paper, we focus on the planning and control layer, and more specifically on the motion planning problem, see [24] for an overview of motion planning algorithms. The goal of this paper is to derive safe sets for motion planning that can be used as a terminal constraint in a Model Predictive Controller (MPC). MPC is a popular approach for motion planning [5, 12, 9, 20] since it allows to directly include temporal and spatial collision avoidance constraints, as well as comfort constraints. Additionally, it is straight forward to plan a motion *without* separating longitudinal and lateral dynamics, which allows for a natural motion planning formulation. However, MPC comes with several drawbacks. In this paper, we focus on the issue of recursive feasibility, where the issue is that even though constraints are considered within the prediction horizon, it is not guaranteed that the constraints can be respected when the controller is run in a receding horizon fashion since the sequence of optimization problems becomes coupled. Recursive feasibility can be achieved by imposing a terminal set constraint [22]. However, even though the theory exists for nonlinear MPC [16], it is complex to compute terminal sets for this class of problems. Therefore, in practice, one common approach to deal with this issue is to use long prediction horizons, which in terms leads to prohibitively long computation times.

In this paper, we specifically focus on the problem of following a path with safety guarantees. Formal and provable safety guarantees for planning and decision making in autonomous driving are currently studied a lot, with one good example being the Responsibility-Sensitive Safety (RRS) approach [29]. However, these approaches mainly deal with the other road users, while guaranteed following of a path is surprisingly studied less. This is mainly addressed in the subfield of autonomous racing [19, 26], where the fact that the race track is known in advance and does not change is leveraged. The same approach, however, cannot be applied directly to autonomous driving, as it would require computing the safe set for the entire road network. Therefore, there only exist a small number of papers that tackle safety and recursive feasibility for motion planning. For example in [1, 34, 23, 19, 4, 28] viability, reachability, and invariant set analysis have been used to guarantee safety. In [26] recursive feasibility is guaranteed using past data and the repetitive nature of autonomous racing. Another approach to introduce safety is by considering uncertainty and tracking errors when generating the trajectory [21, 13, 30, 31]. These methods often use tools from game theory similar to our method, but our safety guarantees are for a path (the road) which is fixed.

In this work, we solve this problem by treating the road ahead as an adversarial player. This allows us to use tools from viability theory and game theory [3, 8]. Given these tools, we design a motion planner that is guaranteed to follow a path which is not known in advance, but has a bounded curvature, all of that while guaranteeing that the car is staying within road limits and fulfilling comfort constraints.

The contributions of this paper are the following. First, we formulate the path-following problem as a game between the controller that wants to follow the road and an adversarial road, that tries to get the car off the road. Based on this game we find an analytical expression for a discriminating domain of the game, which can be used as a terminal set that proves recursive feasibility. Second, we compute the discriminating kernel using a gridding based algorithm and show that it can be accurately approximated using a fully connected neural network. This neural network can again be used as a terminal constraint for our path-following MPC.

Finally, we evaluate our proposed terminal constraints together with a path-following MPC in a simulation study and show that they outperform common baselines, and work well with short prediction horizons where the baselines fail.

This paper is organized as follows. Section II introduces the path-following MPC problem, and Section III summarizes the discriminating kernel algorithm of [8]. In Section IV the path-following task is reformulated as a game and the discriminating domain and kernel for the task are described and computed. In Section V we study the performance of the controller in simulation and compare it to several baseline terminal constraints. Finally, in Section VI we conclude the paper and give an outlook.

II. Path-Following MPC

In this section, we introduce our path-following MPC problem including all the necessary components, such as the model, constraints, and cost function. Note that our safe sets, derived in Section IV, are also based on the modeling and the constraints introduced here.

A. Model

In this paper, we use a kinematic bicycle model formulated in curvilinear coordinates. Thus the position is not formulated in global coordinates, but only the local deviations with respect to a known path are considered, see [25]. More precisely, in the curvilinear coordinate system we only consider the progress along the path s, the orthogonal distance to the path d, as well as the local heading compared to the path μ, see Figure 1 for an illustration and [27] for more details. Note that given the path, the global coordinates can be recovered, and given the global coordinates the curvilinear coordinates can be obtained, given some assumptions.

To formulate the dynamics of our MPC, we consider the kinematic bicycle conditions (see Figure 1), use the steering angle δ as an input and augment the system with a dynamic velocity state v, that is the integration of the acceleration input a. Finally, we consider the influence of the curvature κ on the evolution of the curvilinear coordinates, which results in the following dynamics,

$$\begin{aligned}
\dot{s} &= \frac{v\cos(\mu)}{1-d\kappa(s)}\,,\\
\dot{d} &= v\sin(\mu)\,,\\
\dot{\mu} &= \frac{v\tan(\delta)}{L} - \kappa(s)\frac{v\cos(\mu)}{1-d\kappa(s)}\,,\\
\dot{v} &= a\,.
\end{aligned} \tag{1}$$

Where, the state and input are given by $x = (s, d, \mu, v)$, and $u = (\delta, a)$ respectively, and $\kappa(s)$ is the curvature at the progress s. We denote the resulting continuous time dynamics (1) as $\dot{x} = f_c(x, u)$, and the discrete time version obtained by discretizing the dynamics using an Ordinary Differential Equation (ODE) integrator as $x_{k+1} = f(x_k, u_k)$. Note that the rest of the MPC ingredients are all derived in discrete time, since we use a standard discrete time MPC formulation.

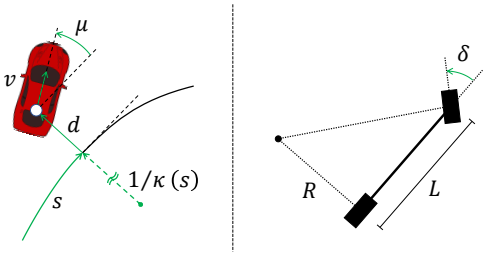

Fig. 1. Right: Visualization of the curvilinear coordinates. Left: Visualization of the kinematic bicycle conditions.

Note that the progress state s, is only linked to the other states due to the s-dependent curvature. However, s could also be used to implement collision constraints.

B. Constraints

In our MPC formulation, we consider two types of constraints: acceleration constraints and road constraints. The acceleration constraint is fundamental for two reasons: first, it guarantees comfort for the people inside the car, and second, the kinematic model (1) does not consider slip in the tires and would overestimate the velocity in corners in the absence of the acceleration constraint.

The acceleration constraint is formulated by limiting the combined acceleration in longitudinal and lateral direction to be smaller than $a_{\max}$. For the model in (1), the longitudinal acceleration is given by $a_{\text{long}} = a$, and the lateral acceleration is $a_{\text{lat}} = v^2\tan(\delta)/L$. The lateral acceleration is given by the centripetal acceleration of the car, combined with the fact that by using a kinematic car model we know that the car is driving a circle with a radius of $R = L/\tan(\delta)$, see Figure 1.

Thus, the acceleration constraint can be formulated as a velocity-dependent input constraint,

$$\mathcal{U}(v) = \{\delta \in \mathcal{D}, a \in \mathcal{A} \,|\, (v^2\tan(\delta)/L)^2 + a^2 \leq a_{\max}^2\}, \tag{2}$$

where $\mathcal{D} = [-\bar{\delta}, \bar{\delta}]$ and $\mathcal{A} = [\underline{a}, \bar{a}]$ are the physical limits of the inputs.

The road constraints force the car to remain within the lane. By using curvilinear coordinates, the road constraints can be implemented by limiting d; however, to guarantee that the whole car remains within the road, the local heading of the car has to be considered. To formulate the constraint, let L_c and W_c be the length and the width of the car respectively, and l_r the distance from the center of the rear axle to the geometric center of the car, see Figure 2 for an illustration. Then, the car stays within the road if,

$$\begin{aligned}
C_w &= W_c/2\cos(\mu) + L_c/2\sin(|\mu|)\,,\\
d + l_r\sin(\mu) + C_w - W_{r,l} &\leq 0\,,\\
-d - l_r\sin(\mu) + C_w + W_{r,r} &\leq 0\,,
\end{aligned} \tag{3}$$

where C_w is the width of the car relative to the lane (considering the local heading), $W_{r,l}$ the left road width and $W_{r,r}$ the right road width, both relative to the reference path (in our experiments we use the center line of the lane). Note that

W_c is as such not differentiable; however, it can be exactly reformulated as a smooth constraint, by using four instead of two constraints. To simplify the notation, in Section IV we denote the road constraint (3) as $h_r(d, \mu) \leq 0$.

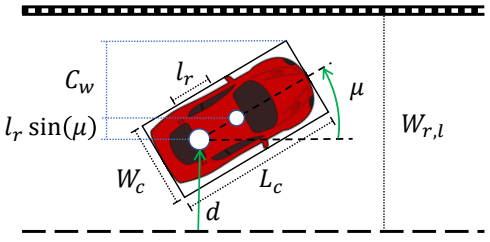

Fig. 2. Visualization of the road constraints.

Finally, we also constrain the relative heading to be within reasonable bounds $\underline{\mu} \leq \mu \leq \bar{\mu}$, and the velocity to be between zero and the speed limit $v_{\max}(s)$ at the current progress s. The combination of the road constraints as well as the heading and velocity constraints is denoted as $x_k \in \mathcal{X}$

C. Terminal Constraint

As discussed in the introduction, for an MPC to have safety guarantees, terminal constraints are necessary. In Section IV a solution is presented for how to tackle this problem for the here presented path-following MPC. However, for now, we assume that we have an abstract terminal constraint of the form $x_{N+1} \in \mathcal{X}_T$, where $N+1$ is the prediction horizon.

D. Cost

The cost of the MPC is the combination of path-following and comfort costs. More precisely, the path-following cost consists of two parts: j_{pf} a quadratic cost on d and μ that penalizes deviations from the path, and j_p a maximum progress cost,

$$\begin{aligned} j_{pf}(x_k) &= q_d d_k^2 + q_\mu \mu_k^2 \,, \\ j_p(x_k) &= -q_p s_{N+1} \,. \end{aligned} \tag{4}$$

Where q_d, q_μ, and q_p are positive weights. Note that instead of a maximum progress cost it is also possible to penalize the deviation from the speed limit. However, we use the maximum progress cost since speed limits are often not appropriate on curvy and urban roads.

The comfort cost consists of two terms, an acceleration cost j_a, that penalizes lateral and longitudinal acceleration, and an input rate cost j_r that penalizes changes in the inputs or in other words jerky motions,

$$\begin{aligned} j_a(x_k, u_k) &= q_{\text{lat}}(v_k^2 \tan(\delta_k)/L)^2 + q_{\text{long}} a_k^2 \,, \\ j_r(u_k, u_{k-1}) &= q_{\Delta\delta} \Delta\delta_k^2 + q_{\Delta a} \Delta a_k^2 \,. \end{aligned} \tag{5}$$

Where q_{lat}, q_{long}, $q_{\Delta\delta}$, and $q_{\Delta a}$ are positive weights and the input rates are defined as follows: $\Delta\delta_k := \delta_k - \delta_{k-1}$ and $\Delta a_k := a_k - a_{k-1}$.

The path-following cost (4) and the comfort cost (5) are combined to the following stage cost

$$\begin{aligned} \ell(x_k, u_k, u_{k-1}) =& j_{pf}(x_k) + j_p(x_k) \\ &+ j_a(x_k, u_k) + j_r(u_k, u_{k-1}) \,, \end{aligned}$$

where u_{-1} is the known last applied input. Additionally, we also introduce a terminal cost which is a modified path-following cost (4),

$$\ell_T(x_{N+1}) = q_{d,T} d_{N+1}^2 + q_{\mu,T} \mu_{N+1}^2 \,,$$

where the weights $q_{d,T}$ and $q_{\mu,T}$ are increased compared to the normal weights.

E. MPC Problem

Combining all of the previous ingredients, we can formulate the path-following MPC as the following Nonlinear Optimization Problem (NLP),

$$\begin{aligned} \min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N} \ell(x_k, u_k, u_{k-1}) + \ell_T(x_{N+1}) \\ \text{s.t.} \quad & x_0 = x(0), \; u_{-1} = u(-1), \\ & x_{k+1} = f(x_k, u_k), \\ & x_k \in \mathcal{X}, \quad x_{N+1} \in \mathcal{X}_T, \\ & u_k \in \mathcal{U}(v_k), \\ & k = 0, \dots, N \,. \end{aligned} \tag{6}$$

Where $\mathbf{x} := [x_0, x_1, \dots, x_{N+1}]$ is the collection of all states and $\mathbf{u} := [u_0, u_1, \dots, u_N]$ is the collection of all inputs. Additionally, $x(0)$ is the measured state and $u(-1)$ is the previously applied input.

III. Viability Theory

In this section, we will briefly discuss discrete-time viability theory [8], which we will later use to establish safe terminal constraints for our path-following MPC. More precisely we are interested in the game-theoretic version of viability theory. There the goal is to establish a *victory domain* for which there exists an input policy that is able to keep the state of the system within a constraint set forever, for all adversarial actions [8].

Therefore, let us consider a discrete-time system with two inputs $z_{k+1} = g(z_k, u_k, \nu_k)$, where $z \in \mathbb{R}^n$ is the state, $u \in \mathcal{U} \subset \mathbb{R}^m$ is the control input, $\nu \in \mathcal{V} \subset \mathbb{R}^d$ is the adversarial input and $g : \mathbb{R}^n \times \mathcal{U} \times \mathcal{V} \to \mathbb{R}^n$ is a continuous function. Furthermore, the constraint set is given as $K \subset \mathbb{R}^n$ and is assumed to be a closed subset of $\mathbb{R}^n$. Furthermore, the policies have the following information pattern when computing the *victory domain*: the control player has access to both the state z_k as well as the adversarial input ν_k thus $u_k = \pi_u(z_k, \nu_k)$, whereas the adversarial player only has access to the state $\nu_k = \pi_\nu(z_k)$. Finally, the goal of the control player is to stay within the constraint set K forever, whereas the adversarial player wants to reach the open set $\mathbb{R}^n \setminus K$.

Given these preliminaries let us now state the standard results from viability theory for this setting.

A. Discriminating Domain and Kernel

The difference equation $z_{k+1} = g(z_k, u_k, \nu_k)$ can be reformulated as a difference inclusion with one input,

$$z_{k+1} \in G(z_k, \nu_k) \quad \text{with } G(z,\nu) = \{g(z,u,\nu) | u \in U\}\,. \quad (7)$$

Thus, the difference inclusion predicts all the states for all allowed actions, given an adversarial input. Given the set-valued map $G(z,\nu)$ we can define a discriminating domain and the discriminating kernel.

Definition 1 ([8]): A set $Q \subset \mathbb{R}^n$ is a discrete discriminating domain of G, if for all $z \in Q$, it holds that $G(z,\nu) \cap Q \neq \emptyset$ for all $\nu \in \mathcal{V}$. The *discrete discriminating kernel* of a set $K \subset \mathbb{R}^n$ under G, denoted by $\text{Disc}_G(K)$, is the largest closed discrete discriminating domain contained in K.

One can show that under the given assumptions, namely that K is a closed subset of $\mathbb{R}^n$ and $G : \mathbb{R}^n \times \mathcal{V} \rightsquigarrow \mathbb{R}^n$ is an upper-semicontinuous set-valued map with compact values, the discriminating Kernel $\text{Disc}_G(K)$ exists.

B. Discriminating Kernel Algorithm

The discriminating kernel $\text{Disc}_G(K)$ can conceptually be computed using the discriminating kernel algorithm, which generates a sequence of nested closed sets,

$$\begin{aligned} K^0 &= K\,, \\ K^{n+1} &= \{z \in K^n | \forall \nu \in \mathcal{V}, G(z,\nu) \cap K^n \neq \emptyset\}\,. \end{aligned} \quad (8)$$

Proposition 1 ([8]): Let G be an upper-semicontinuous set-valued map with compact values and let K be a closed subset of $\mathbb{R}^n$, then, $\bigcap_{n=0}^{\infty} = \text{Disc}_G(K)$

However, since the discriminating kernel algorithm has to operate on arbitrary sets, it is as such not implementable. One popular method to solve this problem is to discretize the state space as well as the input space of the adversarial player. If the state space is appropriately discretized, [8] showed that the discrete space discriminating kernel converges to the discriminating kernel if the space discretization goes to zero.

IV. Safe Set for Path-Following

The main issue when trying to establish a safe set for the path-following task is that the MPC only knows the road until the end of the prediction horizon. Thus, one popular approach to solve the issue is to force the car to stop at the end of the horizon [18]. However, this requires a long prediction horizon, such that the terminal constraint does not negatively influence the current decisions. In this paper we propose to treat the unknown road ahead as an adversarial player, which allows us to compute the discriminating kernel, and in turn guarantees infinite horizon constraint satisfaction.

A. Road Curvature as an Adversary

In our curvilinear dynamics (1), the road ahead is captured by the curvature. Thus, modeling the road ahead as an adversarial player is equivalent to modeling the curvature as an input and limiting the input to stay within an upper and lower bound. This makes the curvature independent of the progress, and therefore we can neglect the progress dynamics for the computation of the discriminating kernel, as it is not linked to any of the other states or the constraints. Thus, we can introduce a reduced state $z = (d, \mu, v)$. The input remains the same $u = (\delta, a)$, but the system now has an adversarial input $\nu = \kappa$, and the system dynamics are given by,

$$\begin{aligned} \dot{d} &= v \sin(\mu)\,, \\ \dot{\mu} &= \frac{v \tan(\delta)}{L} - \kappa \frac{v \cos(\mu)}{1 - d\kappa}\,, \\ \dot{v} &= a\,. \end{aligned} \quad (9)$$

Similar to (1), we denote the continuous dynamics as $\dot{z} = g_c(z,u,\nu)$, and the discretized dynamics as $z_{k+1} = g(z_k, u_k, \nu_k)$. Similarly the input constraint set is given by (2), and the state constraint set is given by,

$$\begin{aligned} K = \{z \in \mathbb{R}^3 \mid -\underline{\mu} \leq \mu \leq \bar{\mu},\, , 0 \leq v \leq \bar{v}, \\ h_r(d,\mu) \leq 0\,, \end{aligned}$$

where $\bar{v}$ is the maximum speed limit. Finally, the action of the adversarial player is limited to $\mathcal{V} = [-\kappa_{\max}, \kappa_{\max}]$, where $\kappa_{\max}$ is the maximal curvature of the road.

Thus, we can now characterize the discriminating kernel for our path-following system.

B. Analytical Discriminating Domain

First, we formulate an analytical discriminating domain, which by definition is a subset of the discriminating kernel. Even though it is only a subset of the discriminating kernel, the advantage is that the set is defined analytically and therefore no expensive computations are necessary. We additionally have to make the following mild assumptions.

Assumption 1: The set K has a relative interior and there exists a d such that $\mu = 0$ is feasible.

Assumption 2: The road constraints (3) and $\kappa_{\max}$ are such the $d\kappa < 1$ is always fulfilled.

Assumption 3: For the ODE integrator that discretizes the continuous time dynamics (9), it holds that $z_{k+1} = z_k$ if $g_c(z,u,\nu) = 0$

The first assumption 1, informally guarantees that the road is wide enough. Assumption 2 guarantees that the curvilinear dynamics are well defined. Finally, Assumption 3 guarantees that the ODE integrator is well behaved in steady-state, which is fulfilled for the integrators used in this paper.

Finally, for the following results, we need $d_{\max}$ which is the "largest" allowed deviation from the path. For the special case when $\mu = 0$, this can simply be found by considering the road constraints (3), thus $d_{\max} = \max(|W_{r,l} + W_c/2|, |W_{r,r} - W_c/2|)$. Note that $d_{\max}$ is also the deviation d for which the disturbance κ has the largest influence on the system (9).

Thus we can now state our main result which is the analytical discriminating domain.

Theorem 1: Given that Assumptions 1, 2, and 3 hold, and

that $\kappa_{\max} \geq \tan(\bar{\delta})/(L + d_{\max}\tan(\bar{\delta}))$, then

$$\mathcal{Z}_{DD} = \begin{cases} W_{r,l} + W_c/2 \leq d \leq W_{r,r} - W_c/2 \\ \mu = 0 \\ 0 \leq v \leq \min\left(\bar{v}, \sqrt{\frac{a_{\max}(1-|d\kappa_{\max}|)}{\kappa_{\max}}}\right) \end{cases}, \quad (10)$$

is a discriminating domain.

Proof: Since, $\mathcal{Z}_{DD} \subset K$, it is sufficient that there exists an input in $u \in U(v)$ such that $z_{k+1} = z_k$ for all $\nu \in \mathcal{V}$, for $\mathcal{Z}_{DD}$ to be a *discriminating domain*, since in this case $G(z,\nu) \cap \mathcal{Z}_{DD} \neq \emptyset$ holds. Furthermore, $z_{k+1} = z_k$ holds, if the continuous time dynamics (9) are stationary, $g_c(z,u,\nu) = 0$. Which can be achieved by designing the following policy,

$$\pi_u(z,\nu) = \begin{cases} \delta = \tan^{-1}\left(\frac{\kappa L}{1-d\kappa}\right) \\ a = 0 \end{cases}. \quad (11)$$

The policy (11) fulfills the input constraints if $0 \leq v \leq \sqrt{a_{\max}(1-d\kappa_{\max})/\kappa_{\max}}$ and if $\kappa_{\max} \geq \tan(\bar{\delta})/(L + d_{\max}\tan(\bar{\delta}))$. The curvature limitation is needed for the physical limit of the steering input, whereas the velocity limitation comes form the lateral acceleration limits.

First, using $a = 0$ guarantees that the input constraints are fulfilled and allows to simplify the input constraints for δ to $-\tan^{-1}(a_{\max}L/v^2) \leq \delta \leq \tan^{-1}(a_{\max}L/v^2)$. Plugging in the policy, and only considering the upper bound due to the symmetry of $\mathcal{U}(v)$ and $\mathcal{V}$, results in $\tan^{-1}(\kappa L/(1-d\kappa)) \leq \tan^{-1}(a_{\max}L/v^2)$, which holds true for all $\kappa \in [-\kappa_{\max}, \kappa_{\max}]$ if $0 \leq v \leq \sqrt{a_{\max}(1-|d\kappa_{\max}|)/\kappa_{\max}}$

If $\kappa_{\max} \geq \tan(\bar{\delta})/(L + d_{\max}\tan(\bar{\delta}))$ then it holds that $\delta = \tan^{-1}(\kappa L/(1-d\kappa)) \leq \bar{\delta}$ for all $z \in \mathcal{Z}_{DD}$. This is simple to verify by reformulating, $\tan^{-1}(\kappa_{\max}L/(1-d_{\max}\kappa_{\max})) \leq \bar{\delta}$, as this guarantees that the biggest steering angle of the policy is within the bounds. ∎

Due to the analytical nature of the discriminating domain established in Theorem 1, it is possible to generalize Theorem 1 to the interesting case of steering rate constraints. This can be done if the road curvature changes slowly enough compared to the steering rate constraint.

Proposition 2: If the rate of change of the steering input is limited by $|(\delta_k - \delta_{k+1})| \leq \overline{\Delta\delta}$ and the rate of change of the curvature is limited by $|(\kappa_k - \kappa_{k+1})| \leq \tan(\overline{\Delta\delta})(1 - d_{\max}\kappa_{\max})/L$ then $\mathcal{Z}_{DD}$ is a discriminating domain.

Proof: From the proof of Theorem 1, we know that the policy (11) renders $\mathcal{Z}_{DD}$ invariant, while not violating $U(v)$.

To investigate the case where the rate of change is limited, we can describe the curvature at time $k+1$ as $\kappa_{k+1} = \kappa_k + \Delta\kappa$, with $|\Delta\kappa| \leq \tan(\overline{\Delta\delta})(1 - d_{\max}\kappa_{\max})/L$. Thus the steering angle is given by $\delta_{k+1} = \tan^{-1}((\kappa_k + \Delta\kappa)L/(1-d\kappa))$. Since we are only interested in the absolute value of the difference we can take the absolute value of the expression, which allows us to make a series of simplifications,

$$\begin{aligned} |\delta_{k+1}| &= |\tan^{-1}((\kappa_k + \Delta\kappa)L/(1-d\kappa))| \\ &\leq |\tan^{-1}(\kappa_k L/(1-d\kappa))| + |\tan^{-1}(\Delta\kappa L/(1-d\kappa))| \\ &\leq |\delta_k| + |\tan^{-1}(\Delta\kappa L/(1-d\kappa))| \\ &\leq |\delta_k| + |\tan^{-1}(\Delta\kappa L/(1-d_{\max}\kappa_{\max}))| \end{aligned}$$

Which can be reformulated as, $|\delta_{k+1}| - |\delta_k| \leq |\tan^{-1}(\Delta\kappa L/(1 - d_{\max}\kappa_{\max}))|$. Thus, the change in the steering angle is bounded by $|\tan^{-1}(\Delta\kappa L/(1-d_{\max}\kappa_{\max}))|$, which is at most $\overline{\Delta\delta}$ if $|\Delta\kappa| \leq \tan(\overline{\Delta\delta})(1 - d_{\max}\kappa_{\max})/L$. ∎

C. Numerical Discriminating Kernel

The discriminating domain $\mathcal{Z}_{DD}$ is only a subset of the discriminating kernel $\text{Disc}_G(K)$. However, to numerically compute the discriminating kernel using the discriminating kernel algorithm, it is necessary to fix the model and constraint set K. Therefore, in Table I the model and road parameters that are used for the rest of our paper are introduced. Note that the parameters are partially taken from [25] and from a spec sheet of a 2005 BMW 320i.

TABLE I
MODEL AND ROAD PARAMETERS

$L = 2.68$m	$l_r = 1.34$m
$L_c = 4.52$m	$W_c = 1.817$m
$a_{\max} = 1.6\text{ms}^{-2}$	$\mathcal{D} = [-0.6, 0.6]$rad
$\mathcal{A} = [-1.6, 1.6]\text{ms}^{-2}$	$\bar{\mu} = -\underline{\mu} = 0.2$rad
$W_{r,l} = -1.5$m	$W_{r,r} = 1.5$m

Additionally, it is necessary to grid the state and adversarial input space. In our implementation we also grid the input space, since this makes the implementation considerably simpler. The parameters used for the gridding can be found in Table II, where $\bar{\delta}(v) = \min(\tan^{-1}(a_{\max}L/v^2), \bar{\delta})$. Note that we use the results of Theorem 1 as the upper bound of the velocity state. We also experimented with denser grids for the inputs, and found that they did not change the resulting discriminating kernel noticeably, if at all. We assume that this is the case since the extreme inputs are the ones which are most important.

TABLE II
GRID PARAMETERS

state	range	#grid	input	range	#grid
d	$[-0.3415, 0.3415]$	101	ν	$[-\kappa_{\max}, \kappa_{\max}]$	5
μ	$[-0.2, 0.2]$	81	δ	$[-\bar{\delta}(v), \bar{\delta}(v)]$	9
v	$[0, \sqrt{a_{\max}/\kappa_{\max}}]$	135	a	$[-a_{\max}, a_{\max}]$	9

Given all the parameters, the discriminating kernel algorithm (8) is implemented in C++ and the discriminating kernel is computed for different $\kappa_{\max}$. We use a 4th order Runge-Kutta integrator to discretize the dynamics with a sampling time of $T_s = 0.2$s. The sampling time is chosen to match our state discretization, which is important for discriminating kernel algorithms [8]. Given the state discretization from Table II

TABLE III
ACTIVATION FUNCTIONS, 3 LAYERS AND 16 NEURONS, (TEST/VAL)

activation	acc [%]	fn[%]	fp[%]
ELU	**99.33/99.19**	**0.61/0.70**	**0.05/0.11**
ReLU	98.82/99.02	0.84/0.88	0.34/0.11
Softplus	99.23/99.15	0.71/0.74	0.05/0.11
Tanh	99.30/99.22	0.65/0.66	0.04/0.11

TABLE IV
NUMBER OF NEURONS, ELU ACTIVATION AND 3 LAYERS, (TEST/VAL)

#neurons	acc [%]	fn[%]	fp[%]
8	98.96/98.88	0.94/0.96	0.09/0.15
16	**99.33/99.19**	**0.61/0.70**	**0.05/0.11**
32	99.42/99.36	0.53/0.53	0.05/0.1
128	99.52/99.53	0.42/0.39	0.05/0.09

TABLE V
NUMBER OF LAYERS, ELU AND 16 NEURONS, (TEST/VAL)

#layers	acc [%]	fn[%]	fp[%]
2	98.89/98.73	1.0/1.14	0.11/0.12
3	**99.33/99.19**	**0.61/0.70**	**0.05/0.11**
6	99.41/99.40	0.51/0.52	0.08/0.08

combined with our C++ implementation the computation times were between 15.3 and 135.9s per maximal curvature. We ran 13 different maximum curvatures $\kappa_{\max,\text{ all}} = \{0.1, 0.05, 0.04, 0.03, 0.02, 0.01, 0.005, 0.004, 0.003, 0.002, 0.0015, 0.00125, 0.001\}\text{m}^{-1}$, which corresponds to roads with maximum radius between 10m and 1000m, thus capturing everything from highways to roundabouts.

1) Neural Network Approximation: As such the discriminating kernel is not usable as a terminal constraint in an MPC since the set is only known in the discretized space, and for discrete values of $\kappa_{\max}$. Therefore, we propose to learn a neural network classifier that approximates the discriminating kernel in continuous space and for continuous values of $\kappa_{\max}$. We therefore train a neural network $h_{nn} : \mathbb{R}^3 \times \mathbb{R} \to [0, 1]$, that maps states and curvatures to one output. The constraint is then implemented as $h_{nn}(x, \kappa_{\max}) \geq c$, where $c \in (0, 1)$.

We use Fully Connected Neural Networks (FCNN) since they can easily handle the massive amount of data that the gridding based discriminating kernel algorithm generates. For our 13 different $\kappa_{\max}$, we already have about 14.5 million data points, while the amount of data points grows exponentially when finer grids are used. Furthermore, FCNN can be included in an MPC formulation, since they are at least once continuously differentiable nonlinear functions if appropriate activation functions are used.

We perform a small ablation study to compare different network architectures, where we mainly compare activation functions as well as the number of neurons and layers of the network. Our study is set up as a sensitivity analysis, where we compare everything to our base case, which is a three-layer network with 16 neurons per layer, using ELU activation functions. For all our networks we use a sigmoid output layer to map the output to $[0, 1]$, and use a binary cross-entropy as a training loss. The training is consistent for all the different networks, and we train the networks for 9 epochs using ADAM as an optimizer, with an initial learning rate of 0.01, which is reduced every 3 epochs by one order. Finally, we use a batch size of 1500 which allows us to train our networks within a few minutes on an i7 desktop processor.

As training dataset we use the union of the 13 computed discriminating kernels. We label points inside the discriminating kernel as 1 (safe) and points outside as 0 (unsafe). Finally, we normalize the dataset and randomly split it between a training (95%) and a validation set (5%). Our test set is designed to test how well the neural network generalizes to unseen $\kappa_{\max}$. Therefore, we use two additional discriminating kernels with curvatures $\kappa_{\max,\text{ test}} = \{0.015, 0.0035\}\text{m}^{-1}$. The test kernels use a finer grid with twice the grid points in each direction.

To compare the results we use the following metrics: accuracy (acc), false-positive (fp), and false-negative (fn) rates. We on purpose choose the cut-off c of the neural network output to result in low false-positive rates, as otherwise the states close to the top forward velocity are often deemed unsafe; specifically we chose $c = 0.25$. Note that the network that we use in the rest of the paper is always highlighted bold in the ablation study tables.

Our sensitivity analysis mainly reassures common knowledge: deeper or wider networks get better scores, see Table V and IV, which indirectly shows that over-fitting seems not to be an issue for this task. The only somewhat unexpected outcome is that ReLU activation functions perform worst for the given task, which can be seen in Table III. We did not investigate this behavior further since ReLUs are not suited to be included in an MPC problem as they are not continuously differentiable. It is also interesting that in general the networks performed better on the test data, than on the validation data. This could be related to the fact that the test data only contains two unseen curvatures. However, the test set still shows that the neural networks learned to successfully interpolate in the curvature dimension.

Our final decision to go with the selected network, was based on computational considerations, as wider and deeper networks increase the computational burden during of the MPC, and Table V and IV show that three layers and 16 neurons strike a good balance between accuracy and simplicity[1]. Furthermore, the ELU activation functions had accuracy scores only rivaled by $\tanh$ activation functions, with the approximation having visually less artifacts[2]. For a visualization of the resulting neural network based discriminating kernel and the comparison to the result of the discriminating kernel algorithm see Figure 3, which shows the test case with $\kappa_{\max} = 0.0035\text{m}^{-1}$.

[1]Code for the discriminating kernel algorithm and FCNN learning available at https://github.com/alexliniger/AdversarialRoadModel

[2]We tested different random seeds for the ELU network and got accuracy scores between 99.29 - 99.34% on the testset. We report scores for one of the intermediate runs with 99.33% accuracy.

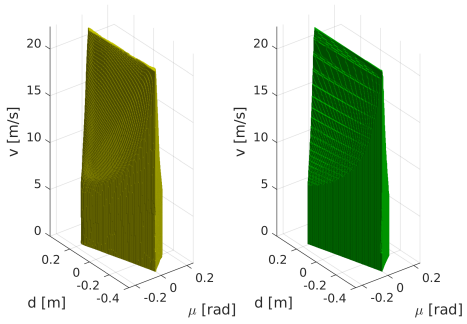

Fig. 3. Visualization of the discriminating kernel for $\kappa_{\max} = 0.0035\text{m}^{-1}$, on the left the neural network approximation and on the right the ground truth from the discriminating kernel algorithm.

V. Simulation

To test the proposed terminal constraints, we performed two tests, first driving along a curvy country road, where the first part is inside a town with a 50km/h speed limit and the second part has an 80km/h speed limit, but a tight turn. The second test should show the performance in a city, and consists of two 90° curves, followed by a 45° curve. The curvature for the two test cases can be seen in Figure 4.

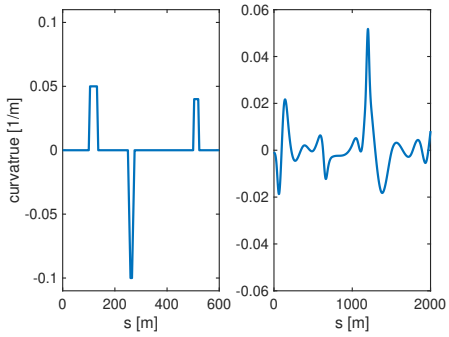

Fig. 4. Test road curvature, left the city road and right the country road.

A. Setup

To make our simulation study more realistic we use a bicycle model with linear tire forces for simulation [25]. To deal with low velocities where the model has a singularity, we use the method proposed in [15]. The parameters for the simulation model are taken from [25] and [2].

To validate our proposed terminal constraints, based on the Discriminating Domain (DD), and the Neural Network Discriminating Kernel (NNDK), we test five different terminal constraints. Two baselines: no terminal constraint, and a zero terminal velocity constraint, as well as three based on our results: first, DD with fixed $\kappa_{\max}$, second, DD with an adaptive $\kappa_{\max}$ based on an algorithm introduced in Section V-B, and finally NNDK with an adaptive $\kappa_{\max}$. We also tested long horizons where the car can stop within the horizon (9 - 14s), and short horizons of 2s where only our proposed safe sets with the adaptive $\kappa_{\max}$ approaches produce useful results.

Our experiments are split in two parts: first, we compare to the baselines, using long horizons, since the baselines only work with sufficiently long horizons, and in a second step we compare the DD terminal constraint with the NNDK terminal constraint using short horizons.

B. Adaptive Worst Case Curvature Computation

To implement our proposed terminal constraints, we need to know the worst curvature $\kappa_{\max}$ of the road ahead. We propose to compute $\kappa_{\max}$ only for a look ahead that is long enough. A long enough look ahead is one that allows us to react to whatever the curvature in the not yet seen part of the road may be. Therefore, we first compute how long it would take to decelerate to zero from the end of the current horizon, and then compute $\kappa_{\max}$ for 1.5 times the distance it would take to stop. Finally, we smooth $\kappa_{\max}$ such that the controller is not disturbed, see Algorithm 1 for the full method.

Algorithm 1: Adaptive $\kappa_{\max}$ computation

at time step t, given $\kappa_{\max,t-1}$;
get v_T, and s_T from previous solution;
compute:
  $t_{\text{stop}} = v_T / a_{\max}$;
  $s_{\text{stop}} = 0.5 a_{\max} t_{\text{stop}}^2 + v_T t_{\text{stop}}$;
  $\kappa_{\max} = \max_{s \in [s_T, s_T + 1.5 s_{\text{stop}}]} |\kappa(s)|$;
result: $\kappa_{\max,t} = (1-\lambda)\kappa_{\max,t-1} + \lambda \kappa_{\max}$;

C. Implementation Details

We implemented the MPC in Julia with the optimization framework JuMP [11] and use IPOPT to solve the resulting NLP [32]. Note that this implementation is not optimized for real-time use and the computation times are only reported to show relative differences. We refer to [17] for a discussion on real-time nonlinear MPC methods. Finally, we discretized the continuous dynamics (1) using a trapezoidal integrator, and used a sampling time of 0.05s.

D. Results

1) Comparison to Baselines: For our country road test we used a look ahead of 14s which corresponds to a horizon of 280 time steps, while for the city road test we used a look ahead of 9s which corresponds to a horizon of 180 time steps. Both are chosen such that the car can stop from the allowed top speed within the prediction horizon. We compare the long prediction horizon variants to our proposed approach that uses the DD-based terminal constraints and the adaptive $\kappa_{\max}$ algorithm, with a look ahead of only 2s which corresponds to a horizon of 40 time steps.

Figure 5, shows that if a long horizon is used, only the MPC with a zero terminal velocity is significantly different, as the allowed speed limit is not reached. The MPC with the short horizon and the DD-based terminal constraint with adaptive $\kappa_{\max}$, has again a different behavior and starts braking earlier

than the other motion planners. However, the distance to the center line is very similar for all five variants, see lower part of Figure 5.

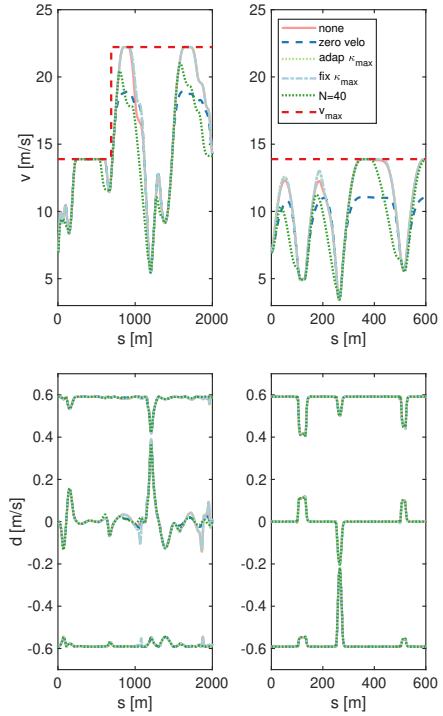

Fig. 5. Top: Speed profile for the comparison with the baselines. Bottom: d profile for the comparison with the baselines, including a visualization of the heading dependent road constraints (3).

Similar patterns can also be seen in Table VI, where the mean combined acceleration is the lowest for the MPC with the zero terminal velocity and the short horizon approach which is a consequence of the slower driving. For the computation time it is visible that the the horizon length has the largest impact, but for a fixed horizon length there is no clear pattern visible. The computation times are recorded on a Desktop PC with an AMD 3900X processor running Ubuntu 18.04.

TABLE VI
QUANTITATIVE RESULTS, TOP COUNTRY ROAD, BOTTOM CITY ROAD

	none	0-velo	adap $\kappa_{\max}$	fix $\kappa_{\max}$	short
mean time	0.4273	0.4301	0.4251	0.4284	0.0428
com acc	1.1696	1.0640	1.1783	1.1774	0.9834
mean time	0.3183	0.2242	0.3040	0.2659	0.0368
comb acc	1.0729	0.7850	1.1105	1.1066	0.8815

2) Discriminating Domain vs Discriminating Kernel: Finally, we compare our path-following controller with the DD and the NNDK terminal constraint. Since we use a short horizon, we use the adaptive $\kappa_{\max}$ approach to not be too conservative. Note that the MPC with no terminal constraint was not able to perform this task, and that imposing a zero terminal velocity or a non adaptive $\kappa_{\max}$ results in the car driving very slowly. For example, the theoretical top speed for the zero terminal velocity approach would be 3.2m/s, and around 8m/s for the fixed $\kappa_{\max}$ approach. As these results are of no practical interest we excluded them.

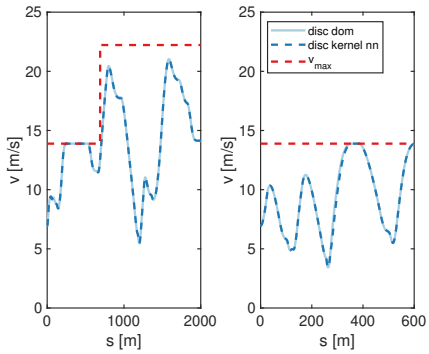

Fig. 6. Speed profile for the comparison between discriminating domain and discriminating kernel.

When comparing the velocity profiles in Figure 6 it is visible that there is basically no difference between the two approaches. The bigger difference can be seen in computation times that increase from 0.0428 to 0.0506s for the country road case and from 0.0368 to 0.0545s in the city road case.

Therefore, we conclude that for the here studied path-following MPC problem, using the DD-based terminal constraint is clearly a better solution. Although the NNDK-based terminal constraint performs the same in our simulation study, it has three main disadvantages: higher computation time, the loss of formal guarantees, and the loss in flexibility. However, if the problem would be changed, for example by using a learned car model [14, 33], the NNDK approach would still be applicable, but the DD approach not.

VI. Conclusion and Future Work

In this paper we introduce a safe optimization-based path-following controller for autonomous driving. The problem uses a kinematic model in curvilinear coordinates and has road constraints that include the orientation of the car as well as comfort constraints. Based on this path-following controller we derived a game theoretic version, where the road ahead is the adversary. Using this model we used tools from viability theory to establish safe sets for the original controller. The first is an analytical discriminating domain, that can be shown to also work if steering rate constraints are considered. The second approach computes the discriminating kernel using the gridding based discriminating kernel algorithm. We then approximate this discriminating kernel with a neural network, which allows us to find an approximate safe set that can be used in an MPC. Finally, we tested the safe terminal sets in simulation, and showed that if an adaptive $\kappa_{\max}$ algorithm is used, our terminal sets allow to run short horizons, with only slightly reduced performance.

In future work, we want to include obstacle avoidance and interactions with other cars by using ideas similar to the ones proposed in [29].

References

[1] Matthias Althoff, Olaf Stursberg, and Martin Buss. Model-based probabilistic collision detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 10(2):299–310, 2009.

[2] Matthias Althoff, Markus Koschi, and Stefanie Manzinger. Commonroad: Composable benchmarks for motion planning on roads. In *Intelligent Vehicles Symposium (IV)*, 2017.

[3] Jean-Pierre Aubin. *Viability Theory*. Springer, 2009.

[4] K. Berntorp, A. Weiss, C. Danielson, I. V. Kolmanovsky, and S. Di Cairano. Automated driving: Safe motion planning using positively invariant sets. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[5] Francesco Borrelli, Paolo Falcone, Tamas Keviczky, and Jahan Asgari. MPC-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3(2):265–291, 2005.

[6] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The 2005 DARPA grand challenge: the great robot race*. Springer, 2007.

[7] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: Autonomous vehicles in city traffic*. Springer, 2009.

[8] Pierre Cardaliaguet, Marc Quincampoix, and Patrick Saint-Pierre. Set-valued numerical analysis for optimal control and differential games. *Stochastic and Differential Games*, 4:177–247, 1999.

[9] Stefano Di Cairano and H Eric Tseng. Driver-assist steering by active front steering and differential braking: design, implementation and experimental evaluation of a switched model predictive control approach. In *Conference on Decision and Control (CDC)*, 2010.

[10] Ernst D Dickmanns and Alfred Zapp. Autonomous high speed road vehicle guidance by computer vision. *IFAC Proceedings Volumes*, 20(5):221–226, 1987.

[11] Iain Dunning, Joey Huchette, and Miles Lubin. Jump: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

[12] Yiqi Gao, Theresa Lin, Francesco Borrelli, Eric Tseng, and Davor Hrovat. Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads. In *Dynamic Systems and Control Conference*, 2010.

[13] Sylvia L Herbert, Mo Chen, SooJean Han, Somil Bansal, Jaime F Fisac, and Claire J Tomlin. Fastrack: A modular framework for fast and guaranteed safe motion planning. In *Conference on Decision and Control (CDC)*, 2017.

[14] 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, 2019.

[15] Juraj Kabzan, Miguel de la Iglesia Valls, Victor Reijgwart, Hubertus Franciscus Cornelis Hendrikx, Claas Ehmke, Manish Prajapat, Andreas Bühler, Nikhil Gosala, Mehak Gupta, Ramya Sivanesan, et al. Amz driverless: The full autonomous racing system. *arXiv preprint arXiv:1905.05150*, 2019.

[16] E Kerrigan and M Maciejowski, J. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *Conference on Decision and Control (CDC)*, 2000.

[17] Dimitris Kouzoupis, Gianluca Frison, Andrea Zanelli, and Moritz Diehl. Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, 46(4):863–882, 2018.

[18] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[19] Alexander Liniger and John Lygeros. Real-time control for autonomous racing based on viability theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, 2017.

[20] Alexander Liniger, Alexander Domahidi, and Manfred Morari. Optimization-based autonomous racing of 1:43 scale rc cars. *Optimal Control Applications and Methods*, 36:628–647, 2015.

[21] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[22] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36:789 – 814, 2000.

[23] Jonas Nilsson, Jonas Fredriksson, and Anders CE Odblom. Verification of collision avoidance systems using reachability analysis. In *IFAC World Congress*, 2014.

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

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

[26] Ugo Rosolia, Ashwin Carvalho, and Francesco Borrelli. Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)*, 2017.

[27] Alessandro Rucco, Giuseppe Notarstefano, and John Hauser. An efficient minimum-time trajectory generation strategy for two-track car vehicles. *IEEE Transactions on Control Systems Technology*, 23(4):1505–1519, 2015.

[28] Tom Schouwenaars, Jonathan How, and Eric Feron. Receding horizon path planning with implicit safety guarantees. In *American Control Conference (ACC)*, 2004.

[29] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[30] Sumeet Singh, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust online motion planning via contraction theory and convex optimization. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[31] Stanley W Smith, He Yin, and Murat Arcak. Continuous abstraction of nonlinear systems using sum-of-squares programming. *arXiv preprint arXiv:1909.06468*, 2019.

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

[33] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M Rehg, Byron Boots, and Evangelos A Theodorou. Information theoretic mpc for model-based reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2017.

[34] Tony A. Wood, Peyman Mohajerin Esfahani, and John Lygeros. Hybrid modelling and reachability on autonomous rc-cars. In *Analysis and Design of Hybrid Systems*, 2012.