Auton Robot (2015) 39:503–517
DOI 10.1007/s10514-015-9469-5

Computationally aware control of autonomous vehicles: a hybrid model predictive control approach

Kun Zhang[1] · **Jonathan Sprinkle**[1] · **Ricardo G. Sanfelice**[2]

Received: 11 February 2015 / Accepted: 9 July 2015 / Published online: 2 August 2015

Abstract Model predictive control (MPC) is a common approach to the control of trajectory-following systems. For nonlinear plants such as car-like robots, methods for path planning and following have the advantage of concurrently solving problems of obstacle avoidance, feasible trajectory selection, and trajectory following. A prediction function for the plant is used to simulate the trajectory with a candidate stream of inputs. Constraints on control inputs and state values, used to ensure safe trajectories and to avoid obstacles, are encoded into a cost function, and optimization routines (at runtime) compute the trajectories and their corresponding control inputs. Such approaches are computationally intensive, and in the nonlinear case the computational burden generally grows as a predictive model more closely approximates a nonlinear plant. In situations where system safety is paramount, guaranteeing model accuracy (in order to achieve more accurate behavior) comes at the cost of increased computation time, which results in increased travel time without a new solution. While the computational burden of predictive methods can be addressed through model reduction, the cost of modeling error over the prediction horizon is high and can lead to unfeasible results. In this paper, we consider the problem of controlling a ground vehicle under constraints and propose an algorithm that employs two models of the vehicle for model predictive control, one coarse and the other more accurate. We introduce a metric called *uncontrollable divergence* and, using this metric, propose a mechanism to select the model to use in the predictive controller. The novel property of the metric is that it reveals the divergence between predicted and true states caused by return time and model mismatch. More precisely, a map of uncontrollable divergence plotted over the state space gives the criterion to judge where coarse models can be tolerated when a high update rate is preferred (e.g., at high speed and small steering angles), and where high-fidelity models are required to avoid obstacles or make tighter curves (e.g., at large steering angles). With this metric, we design a controller that switches at runtime between predictive controllers in which respective models are deployed. The algorithm is a hybrid controller, which evaluates the proposed metric to select the discrete vehicle model to use for prediction and optimization. We say that the approach is computationally aware, in that the optimization time of each predictive model is dependent on the computation substrate used (chipset, machine architecture, etc.); if a different computational platform is used, then the uncontrollable divergence calculations will lead to a hybrid controller suitable to meet the computation demands for that platform. While the ideas are presented for the solution of a vehicle control problem, the approach has the potential to impact other computationally-demanding cyber-physical systems. The paper extends (Zhang et al., Proceedings of the international conference on cyber-physical system, Seattle, 2015) in a significant way, by demonstrating the calculation of uncontrollable divergence on a physical platform, by characterizing MPC return time as a function of the number of obstacles, and by simulating performance with trajectories that must navigate more obstacles.

✉ Kun Zhang
dabiezu@email.arizona.edu

Jonathan Sprinkle
sprinkle@acm.org

Ricardo G. Sanfelice
ricardo@ucsc.edu

[1] ECE Department, University of Arizona, Tucson, AZ, USA

[2] CE Department, University of California, Santa Cruz, CA, USA

Keywords Vehicle control · Model predictive control (MPC) · Hybrid control · Model error evaluation · Cyber-physical systems (CPS)

1 Introduction

Model Predictive Control (MPC) is an established approach to controlling a system based on simulated performance of a predictive model, and utilizes a cost function in order to add penalties (or reward) for constraints (goals). The resulting solution is optimal according to the cost function, and is encoded as a sequence of control inputs and state predictions over a finite horizon. For systems that must track a given trajectory under constraints, MPC has several attractive features. It systematically handles constraints, and returns feasible solutions: while the returned solutions may not match the input trajectories exactly, they are optimally close with respect to a properly chosen cost function. A specific example of such a system is an autonomous vehicle, where a typical goal is to steer a vehicle along (or near to) a desired trajectory while satisfying velocity and heading constraints, e.g., path following with a desired final heading.

As a real-time control algorithm, MPC has several limitations. The online optimization process is computationally intensive when a nonlinear plant predictive model (e.g., a car-like robot) is used. In these cases, if the solution cannot be expressed in closed form (Cannon and Kouvaritakis 2000), it complicates the prediction and optimization subroutines of MPC. Such optimization routines may not be guaranteed to terminate in finite time (Falcone et al. 2007), and prematurely halting the algorithm may result in a locally optimal solution (Parker et al. 2001). In order to mitigate these issues, approaches in the literature fall into three major categories:

(i) Enforce timing constraints if a solution has not yet been found;
(ii) Operate the system under performance limitations until a solution is found; and
(iii) Select a predictive model at design time that only approximates the plant model, but which guarantees (or significantly improves the guarantee) that a solution will be obtained in time.

Enforcing timing constraints [Approach (i)] can be handled by the use of multiple optimization engines at runtime, where the first one to return a feasible solution is selected. A heavy-handed approach cancellation of the optimization routine after a certain time, and then proceeding with the best solution found so far. Such an approach faces the risk that no feasible solution may have been obtained when the routine is halted.

Operating under performance limitations [Approach (ii)] has a few possible solutions. Continuing to execute the control inputs from the previous solution until a new, optimal, solution is returned has a limitation in that the solution would be for a timestep that has already passed. Thus, any new information (e.g., obstacles to avoid) may not have been accounted for since the previous solution. In these scenarios, the runtime system may be adapted until a viable solution is returned. As an example, a vehicle might reduce its velocity until it a feasible result is obtained. A clear limitation of this tactic is that operation in degraded mode might affect nonfunctional metrics (passenger comfort), even if stability can be shown.

Approach (iii) uses a reduced-complexity predictive model, and such an approach provides a significant margin for robustness to potential disturbances. An example for linear systems is discussed in Zeilinger et al. (2014), where a bounded disturbance can be added to the state update equation. Such a system could approximate a nonlinear plant, if it could be shown that the model error is bounded. This approach is likely to execute reliably—when executed in some areas of the state space. However, the system must be operated in a region of the state space where linearization is representative of the behavior of the system. An alternative approach utilizes a reduced order predictive model, known to be more efficient when computing an optimal solution. For example, using the kinematic (rather than the dynamic) model for problems such as path planning and following. If the model can be shown to be robust, then the approach is appropriate; regrettably, as described in Egerstedt et al. (1998), for car-like robots the kinematic model diverges more quickly than the dynamic model when used in prediction.

1.1 Contribution of this work

Considering the limitations of MPC and utilizing the approaches available in the literature, we consider a unique combination of approaches (ii) and (iii) for the control of car-like robots. We propose the problem of controlling a ground vehicle under constraints with an algorithm that employs two models of the vehicle for model predictive control, with different accuracy when compared to the plant; the two models also differ in time to calculate the optimal solution. We design a hybrid controller that switches between predictive models when computing the optimal solution to the cost function. By selecting the most appropriate model when performing the search for the optimal control input vector, we can reduce the time needed to compute the optimization solution (if the vehicle needs to go fast), or we can reduce the error of the predictive model (if the vehicle should more accurately follow a trajectory). We also demonstrate how to calculate the model error and quantify these errors by the time to calculate a solution.

Our proposed switching criteria relies on a novel metric, called the *uncontrollable divergence* (UD) of the system. The UD is calculated using the error of the predictive models with respect to the plant model, and the state change that occurs while awaiting the return of the optimization function. In this paper, we take the kinematic and dynamic models of a complex ground vehicle model and quantify the UD (within the same MPC implementation) over the entire state space. We show that within some region of the state space (e.g., high speed with low steering angle), the kinematic model can have similar error as that of a high fidelity dynamical model, but with a faster MPC optimization time. In this region of the state space, we say the kinematic (or reduced) model outperforms the high fidelity model. The validity of this assertion comes from the fact that divergence of the dynamical and kinematic models is most pronounced at high speed, high turn rate: such a large steering angle should not be used at high speeds, since this region of the state space is deemed unsafe by the system constraints. At low speed, the dynamical model will outperform the kinematic model for high steering angles, but the point is that *at low speeds, the system can afford to wait longer for the optimization routine to return, since the vehicle is traveling slower!*

This paper extends our preliminary work in Zhang et al. (2015) by additional consideration of physical plant models in addition to the simulation models for estimating and calculating the uncontrollable divergence. The process for gathering these data points is different in the physical plant, due to an inability to operate in regions of the state space where unsafe behavior might occur, and in order to obtain correct ground truth localization information. The experimental results include more complex environments through which the controllers can route the vehicle trajectory, demonstrating an ability to improve on the performance of using only one of the predictive models.

1.2 Organization

The remainder of the paper is organized as follows. We formulate the MPC problem, describe related and relevant approaches to overcoming uncertainty in prediction models, and explicitly lay out the mathematical forms of our own predictive models and cost functions in Sect. 2. We then formally state the problem and our approach to its solution in Sect. 3. In Sect. 4, we demonstrate how to calculate the uncontrollable divergence of the simulated system and the physical plant, and how to use that metric to define the hybrid predictive controller. We provide evidence in Sect. 5 that the designed system improves on the timing and accuracy metrics for improved behavior as described in items (i), (ii), and (iii) above. We also demonstrate that the switching conditions of our controller agree with previous results in which the velocity of the ground vehicle will be already be limited for reasons of safety, not for reasons of ensuring that the optimization routine returns in time.

2 Background

In this section we examine related work that follows the approaches laid out in the introduction. We then formulate the MPC problem and describe the cost functions used, the vehicle models we use as the predictive models for the MPC solution, and then describe the vehicle-specific constraints and how we define the true plant model for executing simulations.

2.1 Review of MPC

Model predictive control has been particularly effective in controlling plants whose trajectory must be controlled over a long horizon, and was pioneered in the process industry (Camacho and Bordons 1997). Early work by Garcia (reviewed in Garcia et al. 1989) was instrumental in demonstrating that by considering the future state value over a fixed horizon of discrete points, that feedback control through optimal control techniques could be used to optimize cost. A thorough review of recent approaches and results can be found in Mayne (2014), which are briefly summarized following the notation and setup of MPC, in order to draw attention to the contribution of this work when our problem statement is stated. The MPC formulation is taken from Zhang et al. (2015).

More precisely, consider the system to control be given by the nonlinear continuous-time plant

$$\dot{z} = f(z, u) \tag{1}$$

with state z and input u, where z and u are constrained to belong to the set $\mathcal{X}$ and $\mathcal{U}$, respectively. Let a discretization of this model by a sample time ΔT be given by $\hat{z}^+ = \hat{f}(\hat{z}, \hat{u})$, where $\hat{z}, \hat{u}$, and $\hat{f}$ are the discretizations of z, u, and f, respectively, and $\hat{z}^+$ denotes the value of the state after a discrete step. Let k denote discrete time. Given the state $\hat{z}$ at time k, denoted as $\hat{z}_k$, the algorithms in MPC compute the evolution of the state of the discrete-time system for N steps forward in time under the effect of input sequences of the form $\hat{U}_k = \{\hat{u}_{k,k}, \hat{u}_{k,k+1}, \ldots, \hat{u}_{k,k+N-1}\}$, where $\hat{u}_{k,k+s}$ denotes the value of the input at the s discrete step ahead of the initial state $\hat{z}_k$, so as to solve the problem

$$\mathbf{P}(\hat{z}_k) : \underset{\hat{U}_k \subset \mathcal{U}}{\operatorname{argmin}}\ J_N(\hat{z}_k, \hat{U}_k) \tag{2}$$

where $J_N(\hat{z}_k, \hat{U}_k) = \sum_{t=k}^{k+N-1} \ell(\hat{z}_{k,t}, \hat{u}_{k,t}) + \varphi(\hat{z}_{k,k+N})$ is the cost function, ℓ is the stage cost function, and φ is the

terminal cost function. Denoting an optimal sequence by $\hat{U}_k^*$ and the resulting optimal prediction state sequence by $\Xi_k^* = \{\hat{z}_{k,k+1}^*, \ldots, \hat{z}_{k,k+N-1}^*, \hat{z}_{k,k+N}^*\}$, MPC then applies to the discretized system the M first entries in $\hat{U}_k^*$ and then computes a new optimal sequence at discrete time $k + M + 1$. The parameter $M \leq N$ determines how often to update the sequence of inputs.

2.2 Predictive model uncertainty

Uncertainty in MPC accounted for with min-max models Alamo et al. (2008, 2005), but complexity of optimization sharply increases. The use of tubes in Falugi and Mayne (2014) permits unstructured uncertainty, as opposed to parameter uncertainty (Løvaas et al. 2008), to be considered when predicting a system model through MPC. When plants are uncertain with linear matrix inequality constraints, the approach in Kothare et al. (1996) can show how to quantify the performance of the predictive model.

Works such as Richards (2005), Bahadorian et al. (2012) consider the uncertainty error originating from linearization or bounded disturbance when designing robust MPCs. Such approaches specify as part of the cost function the uncertainty of the *maximum* possible model error and then *minimize* this cost function. This min-max scheme carries a high computational burden, but results in the minimum error among available linearizations. Approaches to robustness that do not require min-max schemes include (Zeilinger et al. 2014), where the approach uses a quadratically constrained quadratic program (QCQP) suitable for following piecewise constant references. For our work, the need to follow a smooth trajectory limits the ability to use these related approaches to mitigate uncertainty.

2.3 Optimization approaches and computation time

Clearly, the larger the parameters M and N, the longer the computation of $\hat{U}_k^*$ and Ξ_k^* would take. More subtly, nonlinearities in the dynamics and the cost functions, as well as the very presence of the input and state constraints, significantly affect the computation time.

The constraints in the cost function can be set such as to guarantee polynomial complexity of the optimization algorithm for linear plants, as shown in Alamo et al. (2005, 2008). That approach uses quadratic maximization through recursion, and is shown to be polynomial in the rank of the constraint/cost matrix, but is limited to linear systems. Probabilistic guarantees of completion for linear systems are explored in Alamo et al. (2009).

Predictive control for trajectory synthesis and path following of a car-like robot utilizes a nonlinear vehicle model with time-varying constraints. In several applications, linearization of the dynamics, the cost functions, and the constraints is a plausible way to reduce computation time when linearizations can be performed off-line. Linearized MPC (LMPC) has the advantages over nonlinear approaches due to its low computational cost (Künhe et al. 2005) and avoidance of the occurrence of non-convex programming which is common in NMPC (Henson 1998). As shown in Rawlings (1999), Falcone et al. (2007) and Kuhne et al. (2004), linearization is normally performed along the previous horizon of prediction. The key limitation of LMPC for nonlinear plants is that the solution either diverges significantly from the plant (due to propagation error in the linear models away from their operating region) or that the computation of the solution comes at a high computational cost (if linearizations are performed at runtime), or that the execution of the system is overly conservative (if control inputs are selected to minimize the model error during execution).

Several works address unbounded return time of the optimizer through an approach called *layering*. In Falcone et al. (2007, 2008), Bemporad and Rocchi (2011), slower dynamics are assigned to an outer loop predictive controller, which sends supervisory inputs to low-level predictive controllers that have faster dynamics. The drawback of this approach is that layering introduces complexities in design and implementation, and decouples the system design into different loops of execution, making the design harder to understand.

In this work, we assume that optimization times for a cold-start (i.e., no optimization history) of any predictive model that we could select with our hybrid controller can be bounded. If the optimization time cannot be bounded for one of the models that we use, then it would not have been used in a non-switching MPC controller.

2.4 Predictive models

Given a nonlinear system we define a family of discrete-time equivalent systems, which will be used in MPC. Given a discrete set $\mathbb{Q} := \{0, 1, \ldots, q_{\max}\}$ and, for each $q \in \mathbb{Q}$, a function $\hat{f}_q : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, let $\{\hat{f}_P^q\}_{q\in\mathbb{Q}}$ define a family of right-hand sides defining discrete-time models of (1) that are used by the predictive controller.[1] The constant $q_{\max}$ defines the number of available models for prediction. For each $q \in \mathbb{Q}$, we define $\hat{\Gamma}_q : \mathbb{R}^n \to \mathbb{R}^n$ as the model mismatch function

$$\hat{\Gamma}_q(\hat{z}) := \hat{f}_P^e(\hat{z}, \hat{u}) - \hat{f}_P^q(\hat{z}, \hat{u}) \tag{3}$$

which captures the error between the q-th model that is available for MPC ($\hat{f}_P^q$) and the discretization ($\hat{f}_P^e$) of the

[1] The dimension of the state of each resulting discrete-time model is allowed to be different. In such a case, since the number of models is finite, one can embed all of the models into the largest space of size n by adding dummy variables. The same argument applies for the dimension of the inputs.

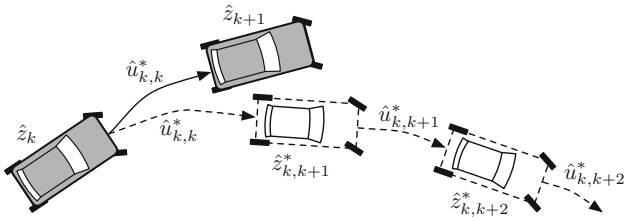


Fig. 1 The *solid* images represent the plant, and *dashed* represent predicted state at future time steps. Note that, only two predicted states, $\hat{z}^*_{k,k+1}$ and $\hat{z}^*_{k,k+2}$, are shown in the figure. The divergence between *solid* and *dashed lines* indicates model mismatch

right-hand of the system. Note that $\hat{f}^e_P$ is not necessarily equivalent to $\hat{f}$, but is sufficiently accurate to serve as a reference.

To differentiate the plant's states and MPC's predicted states (as shown in Fig. 1), let subindex t represent the discrete time over the prediction horizon $N \in \{1, 2, \ldots\}$ and write $\hat{z}_{k,t}$ to denote the predicted state at time t resulting from the initial state $\hat{z}_k$ and under the effect of the planned (given) inputs $\hat{u}_{k,k}, \hat{u}_{k,k+1}, \ldots, \hat{u}_{k,t-1}$.

By convention, $\hat{z}_{k,k}$ equivalent to $\hat{z}_k$. In this way, for a chosen $q \in \mathbb{Q}$, the discrete-time nonlinear model used by MPC for prediction is

$$\hat{z}_{k,t+1} = \hat{f}^q_P\left(\hat{z}_{k,t}, \hat{u}_{k,t}\right) \tag{4}$$

for each t over the prediction horizon, i.e.,

$$t \in \{k, k+1, \ldots, k+N-1\}$$

where $\hat{z}_k$ is fed into the prediction as the initial state.

The first input $\hat{u}^{q*}_{k,k}$ is applied to the plant, leading to the implicit control law

$$\kappa_q(\hat{z}_k) := \hat{u}^{q*}_{k,k}$$

where $\kappa_q(\hat{z}_k)$ solves the problem in (2) using predictive model q (i.e., $\mathbf{P}^q(\hat{z}_k)$) Then, at k, the state of the plant under this control law is updated via

$$\hat{z}_{k+1} = \hat{f}^e_P\left(\hat{z}_k, \kappa_q(\hat{z}_k)\right)$$

At the next step, $k+1$, the hybrid controller may select a new value for q denoted $q' \in \mathbb{Q}$, or keep it constant ($q' = q$), and the problem $\mathbf{P}^{q'}(\hat{z}_{k+1})$ is solved. The algorithm continues this process indefinitely.

In this work we must assume that the prediction models in use are suitable for the plant under control. Our approach is neither intended to stabilize systems with significant model mismatch, nor to address issues of robustness to significant structural or parameter-based error. This assumption permits us to assume that selecting a model will not drive us to a point of instability, since that model would otherwise not be considered a suitable prediction model for an MPC controller.

2.5 Switched MPC

Multiple operating models for MPC (Zhao et al. 2003) permit scheduling of different linearization points, as the state moves to a new operating region. The use of finite automata to switch to different dynamical systems using logic (for piecewise affine systems) is discussed in Bemporad and Morari (1999), and in this paper the ability to utilize mixed-integer quadratic programming (MIQP) as the solver, demonstrates an ability to implement MPC solutions for systems whose model can be written in closed form. The survey paper by Mayne (2014) lists the various approaches to hybrid MPC for implementation, including work by the authors in Goebel et al. (2012).

Since the switching conditions play a significant role in any hybrid system's stability and performance, it is necessary to carefully craft these switching rules using tools that will enable stable behavior. The contribution of this in essence is its ability to jointly consider a model's accuracy with the time requirements that accompany that model's complexity. We next turn to the kinds of models considered in the work.

2.6 Vehicle models

Let us take a car-like robot with plant model f and its discrete-time equivalent with reference right-hand side $\hat{f}^e_P$. We use as our plant a simulation model in CarSim, configured to approximate our physical testbed. In addition, two vehicle models are selected ($\mathbb{Q} = \{0, 1\}$) as available predictive models for an MPC controller. The continuous-time models, given below in (5) and (6), are discretized to obtain $\hat{f}^0_P$ and $\hat{f}^1_P$, respectively. The kinematic model ($q = 0$) is reduced in accuracy through significant simplification from the plant; the dynamic model ($q = 1$) is relatively more accurate when compared to the plant model.

2.6.1 Complex car model

In this paper, we use the CarSim car-like robot model developed in CarSim.[2] CarSim is a complex vehicle simulation environment used in many industrial applications to establish a 50+ degree of freedom simulation of a ground vehicle. We configured our model to attempt to approximate our physical testbed, providing the same wheelbase $L = 2.51$ m and mass $m = 1700$ kg as our physical testbed, a 2008 Ford Escape Tire stiffness data gathered from CarSim are used in (7), represent relationships among lateral force, tire load and steering angle, and are summarized in Fig. 2b (image generated from

[2] Available from http://www.carsim.com.

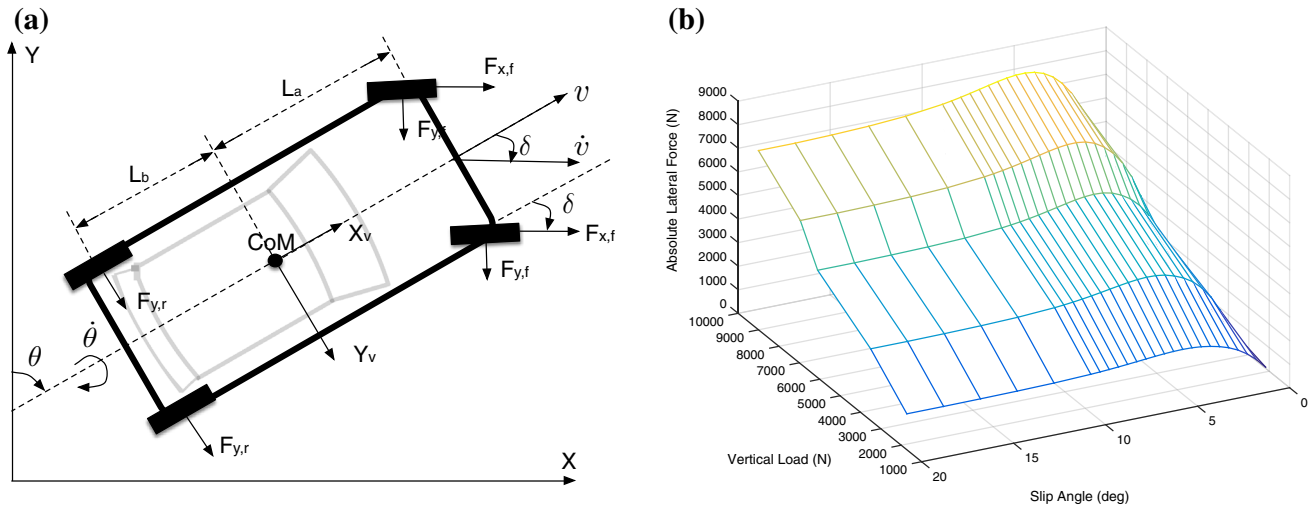


Fig. 2 **a** Schematic view of the vehicle model. $X_v \perp Y_v$ is the vehicle body-fixed coordinate. **b** Absolute lateral force corresponding to tire vertical load and slip angle (treated as steering angle in this work)

CarSim). We consider that the CarSim model represents an actual system plant, to which we compare the behavior of models used for prediction in our algorithm by applying to the CarSim model the control input generated for those simpler models. We will later compare how data gathered from the physical platform differ from data gathered from CarSim.

Regardless of which model for predictive control our hybrid controller selects, we send the input calculated from the MPC controller through the plant model.

2.6.2 Kinematic model

The simplest vehicle model considered in this paper is a kinematic vehicle model. We use the model from Walsh et al. (1994): let $z = [x, y, \theta]^T$ define state as shown in Fig. 2a, where θ is the azimuth. The model is given by

$$\dot{z} = \begin{bmatrix} v \sin\theta \\ v \cos\theta \\ \frac{v \tan\delta}{L} \end{bmatrix} \tag{5}$$

where L is a system parameter defining the length of the vehicle base, and the control inputs $u = [v, \delta]^T$ represent velocity and steering angle.

2.6.3 Dynamic model

In contrast to the kinematic model presented in the previous section, a more advanced model is given by the dynamical model in Narby (2006). We customize this model by imposing the following assumptions:

(i) Each tire shares the same parameters (vertical load, stiffness, etc.), and the lateral forces on left side and right side tires are symmetric;
(ii) Air resistance is negligible; and
(iii) The vehicle is front-wheel drive (and front-wheel steered), and the slip angle equals the steering angle.

Denoting the vehicle mass by m and the angular speed by φ, this simplified dynamical model has inputs of acceleration a and steering angular speed ω, $u = [a, \omega]^T$, as shown in Fig. 2a and given in equation form as:

$$\dot{z} = \begin{bmatrix} v \sin(\theta) \\ v \cos(\theta) \\ \cos(\delta)a - \frac{2}{m} F_{y,f} \sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma \sin(\delta) + 2F_{y,f} \cos(\delta)\right) - 2L_b F_{y,r}\right) \\ \omega \end{bmatrix} \tag{6}$$

where $z = [x, y, v, \theta, \varphi, \delta]^T$, δ is the steering angle, $F_{y,f}$ is the front tire lateral force, $F_{y,r}$ is the rear tire lateral force, L_a is the distance between the centers of the front wheels and the vehicle's center of mass, L_b is the distance between the centers of the rear wheels and the vehicle's center of mass, J is the rotational momentum. These forces can be computed from

$$F_{y,f} = C_y\left(\delta - \frac{L_a \varphi}{v}\right)$$

$$F_{y,r} = C_y\left(\frac{L_b \varphi}{v}\right)$$

where C_y is the lateral tire stiffness. Clearly (6) is more complex than (5).

3 Problem statement and approach

The problem to solve in this paper is the following:

Problem *Select a model from the family of vehicle models* $\{\hat{f}_P^q\}_{q \in \mathbb{Q}}$ *such that the error (or divergence) between the dis-*

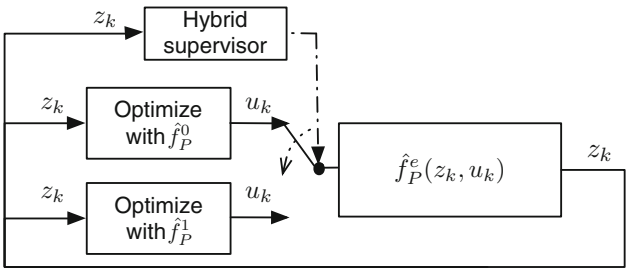


Fig. 3 Closed loop system. Hybrid MPC and CarSim plant in the loop

cretized state of the plant ($\hat{z}$) and of the model ($\hat{z}_k$) obtained with the same inputs is minimized.

To this end, we propose a hybrid controller which implements supervisory logic for the selection of the model defined by the family $\{\hat{f}_P^q\}_{q\in\mathbb{Q}}$ to use by MPC. Suppose at time $k \in \{0, 1, \ldots\}$, the vehicle state $\hat{z}_k$ is observed for an optimization problem indexed by the q-th model in use (i.e., $\mathbf{P}^q(\hat{z})$), and that two alternative predictive models are available. This setting is depicted in Fig. 3, in which the previously described models are used to synthesize two model predictive controllers: the kinematic model ($\hat{f}_P^0$) produces the Kinematic MPC (KMPC) algorithm, while the dynamic model ($\hat{f}_P^1$) produces the Dynamic MPC (DMPC) algorithm. The commercial vehicle modeling software, CarSim, is used as the true plant in our simulations.

To provide a solution to the stated problem, we assume that the return time to solve $\mathbf{P}^q(\hat{z})$ for each predictive model q can be estimated that the quantity $\Delta t_q(\hat{z})$ provides such estimation. After time $\Delta t_q(\hat{z})$ the system state will be $\hat{z}' \approx \hat{z} + \hat{f}(\hat{z}, \hat{u}^*_{k-1,k})\Delta t_q(\hat{z})$ when the control is received. Thus, the vehicle state at $k+1$ is given by

$$\begin{aligned}\hat{z}_{k+1} &= \hat{f}_P\left(\hat{z}'_k, \kappa_q(\hat{z})\right) = \hat{f}_P^q\left(\hat{z}'_k, \kappa_q(\hat{z})\right) + \hat{\Gamma}_q(\hat{z}'_k)\\ &\approx \hat{f}_P^q\left(\hat{z}, \kappa_q(\hat{z})\right) + \hat{\Gamma}_q(\hat{z})\\ &+ \left(\frac{\partial \hat{f}_P^q\left(\hat{z}, \kappa_q(\hat{z})\right)}{\partial \hat{z}} + \frac{\partial \hat{\Gamma}_q\left(\hat{z}, \kappa_q(\hat{z})\right)}{\partial \hat{z}}\right)\\ &\times \hat{f}\left(\hat{z}, \kappa_q(\hat{z})\right)\Delta t_q(\hat{z}) \end{aligned} \tag{7}$$

Since $\hat{z}^q_{k,k+1} = \hat{f}_P^q\left(\hat{z}, \kappa_q(\hat{z})\right)$, we obtain the following from (7)[3]:

$$\hat{z}_{k+1} - \hat{z}^q_{k,k+1} \approx \hat{\Gamma}_q(\hat{z}) + \frac{\partial \hat{f}_P\left(\hat{z}, \kappa_q(\hat{z})\right)}{\partial \hat{z}} \hat{f}\left(\hat{z}, \kappa_q(\hat{z})\right)\Delta t_q(\hat{z}) \tag{8}$$

[3] For clarity of notation, we use $\hat{z}^q_{k,k+1}$, rather than $\hat{z}^{q*}_{k,k+1}$, even though $\hat{f}_P^q\left(\hat{z}, \kappa_q(\hat{z})\right)$ produces state from the initial optimal input $\hat{u}^{q*}_{k,k} = \kappa_q(\hat{z}_k)$, as the notation could also indicate selection of optimal $q \in \mathbb{Q}$.

The logic to select q when solving $\mathbf{P}^q(\hat{z}_k)$ should minimize the error between prediction and the actual state at $k+1$. Then, the selection of q is given by the law

$$q = \underset{q\in\mathbb{Q}}{\operatorname{argmin}} \left\| \hat{z}_{k+1} - \hat{z}^q_{k,k+1} \right\| \tag{9}$$

We refer to the value $\left\| \hat{z}_{k+1} - \hat{z}^q_{k,k+1} \right\|$ as the uncontrollable divergence (UD).

We will demonstrate how to calculate $\hat{\Gamma}_q$ as a function of v and δ. Thus, prior to running the hybrid controller, the upper bound value of $\left\| \hat{z}_{k+1} - \hat{z}^q_{k,k+1} \right\|, \forall q \in \mathbb{Q}$ over finite samplings of bounded v and δ can be obtained. In this way, the hybrid controller can take advantage of the switching logic in (9).

In the next section we go through the demonstration and calculation of the uncontrollable divergence, and describe a way to calculate the estimated return times of the MPC controllers to solve each $\mathbf{P}^q(\hat{z}_k)$. If improved estimates to determine these numbers can be obtained, then those approaches can be substituted.

4 Divergence and return time for plant models

The evolving state of the vehicle during the solution of $\mathbf{P}^q(\hat{z}_k)$ is a chief metric for characterizing uncontrollable divergence. Since the vehicle is typically moving during this time, the state evolves for that time duration. As motivated in the introduction, we consider that in some regions of the state space the model divergence between the two predictive models is low, so a faster return time may be preferred, even if that model has higher error, depending on the divergence value.

4.1 Model divergence calculation approach

Using the concepts from Schubert et al. (2008) to calculate the model divergence, we begin to quantify the uncontrollable divergence by first considering model divergence. The predictive models will receive the same inputs as the plant model, with a constant tire angle (δ) and velocity (v). Differences between the models are then compared throughout the range of possible (v, δ) for the system. Figure 4a shows that the model error is defined in the vehicle's body-fixed coordinates ($X_v \perp Y_v$) as the correcting vector $[e_x\ e_y\ e_\theta]^T$ pointing from model to plant. With the predictive model (indexed by q) and the plant, each starting from the same initial state $\hat{z}$ with the same constant inputs $\hat{u}$, each diverging into model state $\hat{f}_P^q(\hat{z}, \hat{u})$ and vehicle state $\hat{f}_P^e(\hat{z}, \hat{u})$ after ΔT, then we have model mismatch $\hat{\Gamma}_q(\hat{z}) = \hat{f}_P^e(\hat{z}, \hat{u}) - \hat{f}_P^q(\hat{z}, \hat{u})$. Since tire angle and velocity match in the predictive model and plant, the model mismatch value is obtained by:

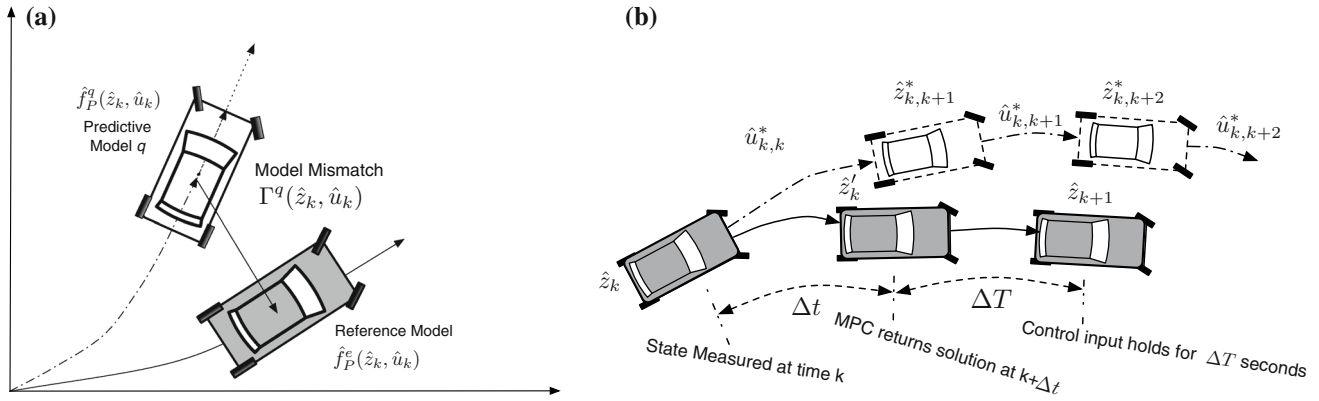


Fig. 4 **a** The *dashed* and *solid squares* represent the model and the plant, respectively. The *bold arrow*, pointing from the model to the vehicle, is the vector of model mismatch accumulated during ΔT. The scenario of control divergence. The controller minimizes the difference between $\hat{z}_{k+1}$ and $\hat{z}^*_{k,k+1}$. A high-fidelity model with large return time Δt can cause large divergence during the timespan Δt. A reduced accuracy model could have smaller Δt, but still produce a large divergence during ΔT if the model mismatch is large

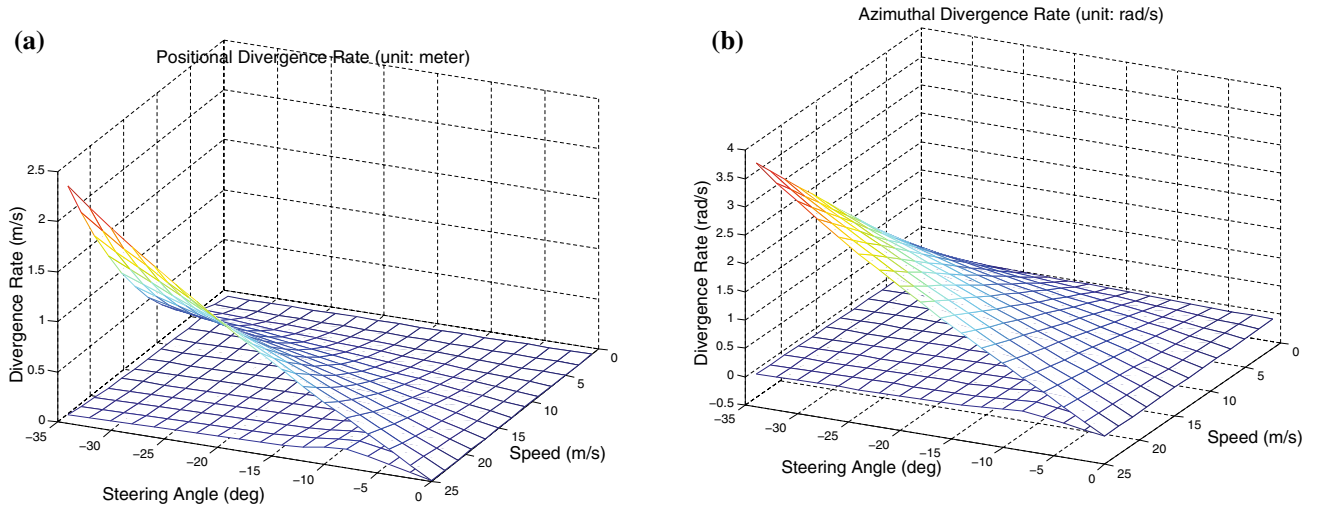


Fig. 5 **a** Comparison of positional divergence rate ($\left\|[e_x\ e_y]^T\right\| / \Delta T$) of the dynamic model (the *bottom* surface) and the kinematic model (the *upper* surface) to the plant model of CarSim. **b** Comparison of azimuthal divergence rate ($\|e_\theta\| / \Delta T$) of the dynamic model (the *bottom* surface) and the kinematic model (the *upper* surface) to the plant model of CarSim

$$\left\|\hat{\Gamma}_q(\hat{z})\right\| = \left\|\left[e_x\ e_y\ e_\theta\right]^T\right\|$$

4.1.1 Divergence in CarSim

Since the plant can be readily simulated in CarSim, we can execute detailed experiments in batch mode to gather data points throughout the state space in order to calculate the uncontrollable divergence. Further, we are not limited in state estimation by noisy or time-varying data measurement units such as GPS and inertial navigation systems. Figure 5a, b reveal example positional and azimuthal divergence rates.

4.1.2 Divergence for the physical plant

In order to gather additional data for divergence for the physical plant, we must describe our experiment setup for our Ford Escape Robotic Car. In order to make safe calculations of model divergence, all sampling was done in human-driving mode, where inputs to the vehicle are made by a human driver but all data are recorded (including inputs) in order to estimate the divergence of a reduced complexity plant model with the same inputs. The ground truth for pose is based on the same system used for vehicle state estimation when following trajectories in a local coordinate system, so as to prevent errors in GPS from biasing accuracy based on the time of day that the experiments were carried out.

In Fig. 6 the datapoints gathered to build the surface functions that describe model mismatch are displayed. Experiments were conducted at sampled vehicle state (sampled steering angle and sampled speed) of 20 Hz while under human control. At varying speeds, the vehicle was driven in a controlled track to attempt to exercise as much of the

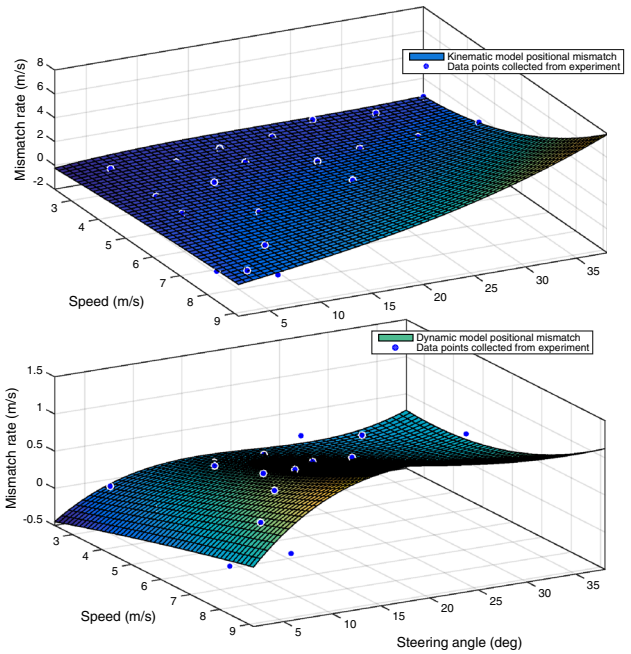


Fig. 6 Experiment datapoints for building the physical plant model

safe state space as possible, while measuring a ground truth trajectory.

With these results, the two charts in Fig. 7 can be generated to demonstrate the uncontrollable divergence, according to the approach discussed in Sect. 4.1. When compared to Fig. 5, the figures for our physical platform demonstrate a steady-state error when compared to the data gathered for the dynamic simulation in CarSim. Such a deviation from the platform is expected for any model, and may be corrected in future work with improved system identification. For purposes of this paper, it influences how the uncontrollable divergence is observed.

4.2 Return time of MPC

Simulations were conducted to measure the return time of KMPC and DMPC. In Qin and Badgwell (2003) various MPC computational schemes are surveyed and available vendors are described. In this paper, we applied AMPL (Richards and How 2002) and minos (Gay and Kernighan 2002) as our optimization tools and we use MATLAB to integrate the optimization results into a cohesive simulation. For the convenience of comparison, KMPC and DMPC are simulated under the same circumstances (the same initial state, obstacles and target state). We treat the optimization return time as a black box, where the most important parameter besides the prediction model is the number of obstacles to consider. This is because each obstacle may show up as N constraints into the optimization (with a horizon of N). The maximum number of obstacles in our simulations is 20.

Results of elapsed time are shown in Fig. 8, and the expected values of return time are selected as 0.02 s (KMPC) and 0.05 s for DMPC. It is important to note that even though KMPC is expected to return more than twice as fast as DMPC in these simulations (and generally in our experience), we assume that both algorithms will return in bounded time—else they would not have been deemed suitable predictive models. We are careful not to claim that our approach will rectify a design with potentially unbounded return times: rather, we instead declare that if the optimization routine may not return for a particular model, that it is unsuitable for use in our framework.

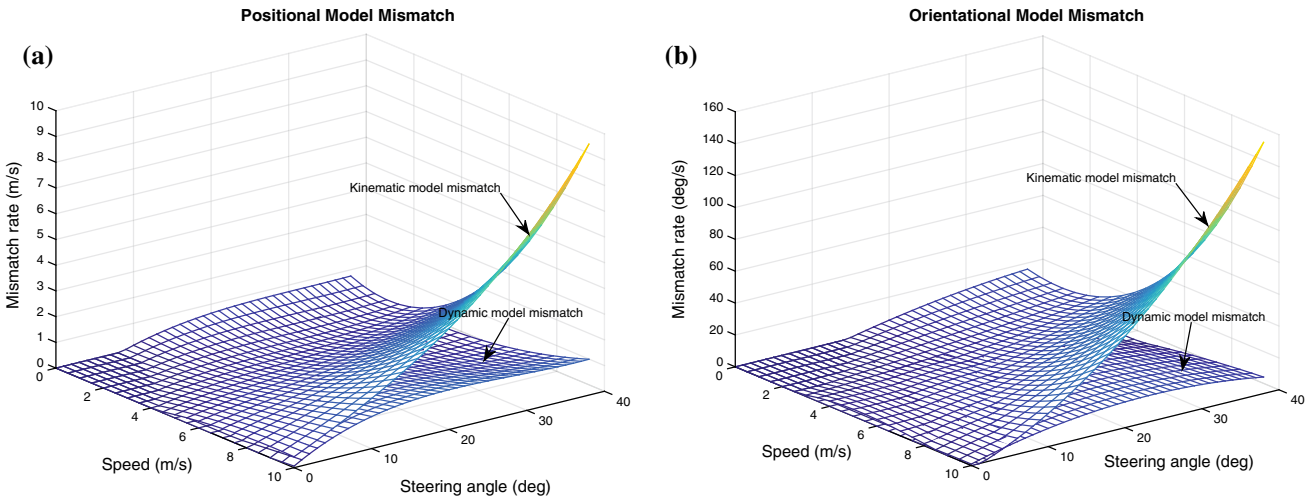


Fig. 7 **a** Comparison of positional divergence rate ($\|[e_x\ e_y]^T\| / \Delta T$) of the dynamic model (the *bottom* surface) and the kinematic model (the *upper* surface) to the plant model of our physical platform. **b** Comparison of azimuthal divergence rate ($\|e_\theta\| / \Delta T$) of the dynamic model (the *bottom* surface) and the kinematic model (the *upper* surface) to the plant model of our physical platform

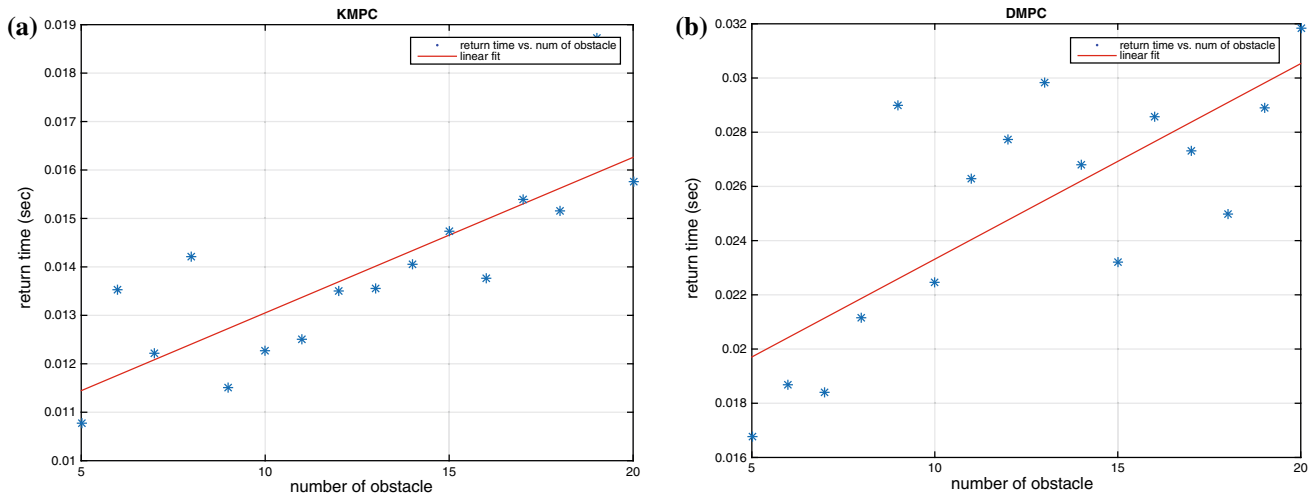


Fig. 8 Number of obstacles versus MPC return time. For each given obstacle number, random plant state and random obstacles are generated for simulation. Samples represent the average return time. **a** Kinematic MPC return time. **b** Dynamic MPC return time

4.3 Influence of uncontrollable divergence

To study the influences of UDs on control performance, the CarSim model is employed as the plant. Since the original CarSim model we selected accepts open-loop throttle, brake cylinder pressure and steering torque as control inputs, we tuned a simple PID controller as the interface for converting target speed and steering angle (elements in $\hat{z}^*_{k,k+1}$) to controls on throttle, cylinder pressure and steering torque. Zero initial state step responses of the interface controller are shown in Fig. 9 to demonstrate its ability to stabilize itself to the target state within the required time. As the plant

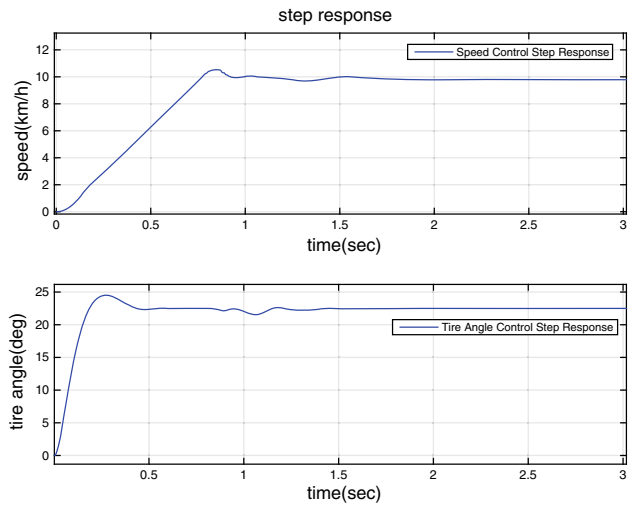


Fig. 9 Step response of controllers that convert MPC signal into CarSim-compatible input. The maximum acceleration is $3\,\mathrm{m/s^2}$, which means the maximum increasing speed difference of 10 km/h may happen within 1 s. The *upper* figure reveals the controller's ability to track target speed. Similarly, the response in the *bottom* figure satisfies the maximum steering speed of 22.5° within 1 s

is nonlinear, however, the controllers may fail to meet timing requirements at some portions of the state space. Additional design work can mitigate this shortcoming, should it influence our simulation results significantly.

When the CarSim plant is used, the vehicle trajectory diverges from the predicted path in an obvious manner (see the simple scenario shown in Fig. 10a, b). Such a divergence originates from model mismatch between the kinematic and dynamical models. Say that a predictive model has a larger curvature than that of the plant at a given specific steering angle and speed. At this steering angle value, the MPC predicts a certain curvature that is infeasible for the plant; thus the vehicle could enter the obstacle zone near its edge. When this is detected as the vehicle nears the obstacle, it will attempt to correct its trajectory (or it could return an infeasible solution of $\mathbf{P}^q(\hat{z})$). As a consequence, the predicted path could oscillate, and the actual trajectory may not fit the obstacle boundary well. Such errors are typical of kinematic models where the curvature is overestimated, as the nearer the vehicle gets to the obstacle, the less opportunity it has to increase its steering angle sufficiently to avoid the obstacle.

In Fig. 11b, the uncontrollable divergences of the physical platform from its two predictive models are shown. In contrast to the CarSim model shown in Fig. 11a, the dynamical model in this example does not significantly diverge at high speeds. However, at low speeds and tire angles it is still superior to the kinematic model, just as the dynamical model was for the CarSim plant.

The uncontrollable divergence for the physical plant is clearly more significant than the CarSim plant; one mitigation to this observation is that the region of the model where mismatch is highest is one in which it was not possible to gather data—since this region of the model is one in which the vehicle cannot safely operate (Whitsitt and Sprinkle 2012).

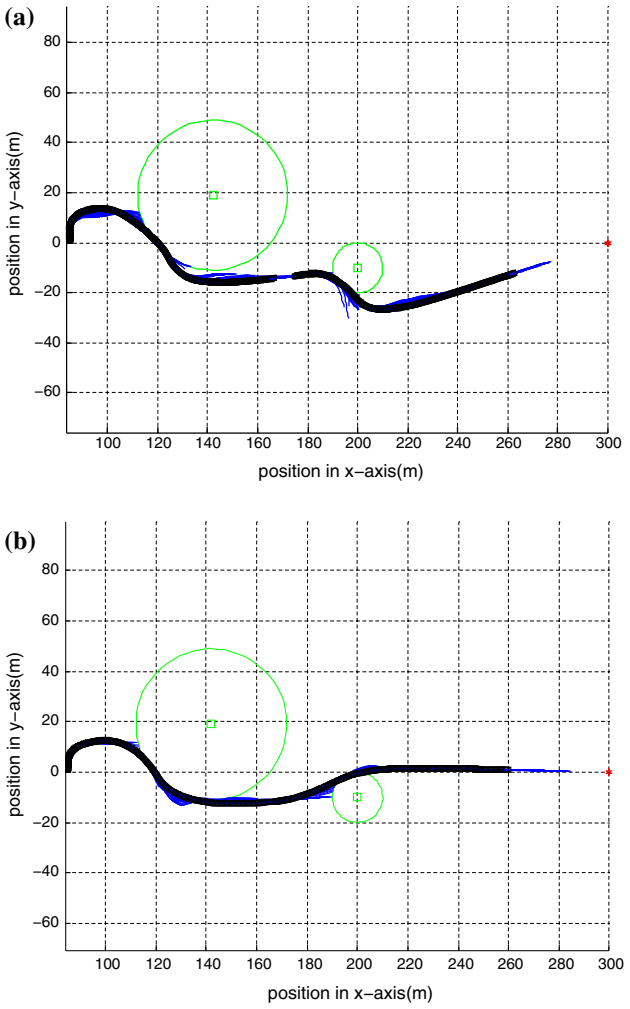


Fig. 10 **a** Kinematic MPC ($q = 0$) only; there is significant divergence of the predicted from the plant model, which results in a path that is unable to navigate between the two obstacles. **b** Dynamic MPC ($q = 1$) only; here the selected trajectory and its tracking are much more aggressive

Thus, this region of the divergence calculations is based on extrapolated data from Fig. 7.

Regardless of mitigating factors, it is important to note that the approach we present in this work will not resolve such fundamental issues of model mismatch. If a selected $\hat{f}_P^q$ is sufficiently divergent from the plant model, then the system could be unstable. Our approach is limited in its scope to selecting the predictive model so that error is reduced, but this error cannot be eliminated without accurate prediction models.

5 Hybrid predictive controller

By replacing $\left\|\hat{\Gamma}_q\right\|$ and Δt_q for each $q \in \mathbb{Q} = \{0, 1\}$ obtained above, we can now plot the upper bound on UDs for our simulation model in CarSim in Fig. 11a. Recall that the proposed switching logic in (9) is to solve $argmin$ for q. To improve performance at run time, we produce a lookup table at the switching boundary in Fig. 12 based on Fig. 11a. Thus, the switching logic is governed by query to this table.

The abscissa and ordinate of Fig. 12 are velocity v and steering angle δ. The region near the top of the figure (low steering angle, and low speeds, or high speed and very low steering angle) is where the kinematic model outperforms. The rest of the region belongs to the dynamic model. We denote the switching boundary (the solid line in Fig. 12) between KMPC ($q = 0$) and DMPC ($q = 1$) by $\Omega : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and fit the boundary by tuning a constant $c > 0$. This procedure leads to a switching boundary defined by the surface

$$\Omega(v, \delta) = v - c\left|\frac{1}{\delta}\right| = 0$$

The mutual constraint between speed and steering angle described in Whitsitt and Sprinkle (2012) is also plotted in the figure as a dashed curve. Let $q \in \mathbb{Q} = \{0, 1\}$ and suppose the control law obtained from each MPC algorithm is represented (with some abuse of notation due to the control law being recalculated at every iteration) $\hat{z} \mapsto \kappa_q(\hat{z})$, where $q = 0$ indicates KMPC and $q = 1$ indicates DMPC. Let $\varepsilon := \left[q\ \hat{z}^T\right]^T \in \mathbb{Q} \times \mathbb{R}^n$. The mechanism that selects the proper value of q, and hence the MPC controller to use, is modeled as a hybrid controller which leads to a closed-loop system with hybrid dynamics. The resulting hybrid system is given by

$$\mathfrak{H} = \begin{cases} \varepsilon \in (\{0\} \times \mathbb{C}_0) \cup (\{1\} \times \mathbb{C}_1) & \varepsilon^+ = \begin{bmatrix} q \\ \hat{f}\left(\hat{z}, \kappa_q(\hat{z})\right) \end{bmatrix} \\ \varepsilon \in (\{0\} \times \mathbb{D}_0) \cup (\{1\} \times \mathbb{D}_1) & \varepsilon^+ = \begin{bmatrix} 1-q \\ \hat{z} \end{bmatrix} \end{cases} \tag{10}$$

where $\varrho \in \mathbb{R}^+$ is a tuned value to prevent the possibility of several instantaneous switches between controllers. The sets $\mathbb{C}_q$ and $\mathbb{D}_q$ are given by

$$\begin{aligned} \mathbb{C}_0 &= \{\hat{z} \in \mathbb{R}^n : \Omega(\kappa_0(\hat{z})) - \varrho < 0\} \\ \mathbb{D}_0 &= \{\hat{z} \in \mathbb{R}^n : \Omega(\kappa_0(\hat{z})) - \varrho \geqslant 0\} \\ \mathbb{C}_1 &= \{\hat{z} \in \mathbb{R}^n : \Omega(\kappa_1(\hat{z})) + \varrho > 0\} \\ \mathbb{D}_1 &= \{\hat{z} \in \mathbb{R}^n : \Omega(\kappa_1(\hat{z})) + \varrho \leqslant 0\} \end{aligned} \tag{11}$$

where as described in Sect. 2.4 we have embedded the discrete state $\hat{z}$ of each model such that they are all in $\mathbb{R}^n$. Also, recall that $\kappa_q(\cdot)$ defines the implicit control law to supply control inputs for the next timestep. In particular, the set

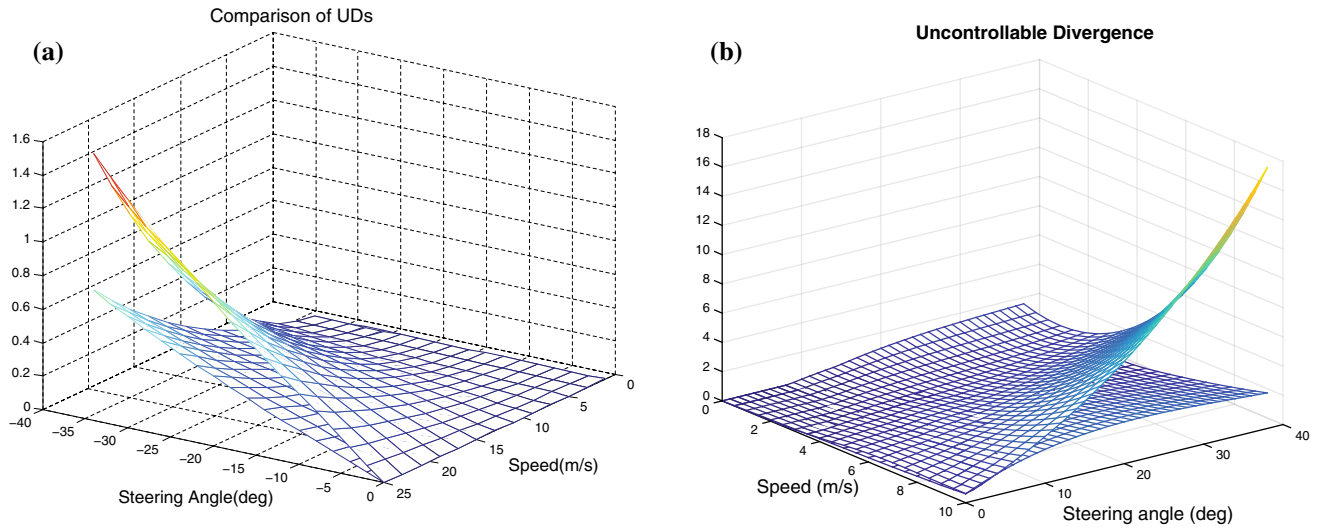


Fig. 11 Comparison of uncontrollable divergences (UDs) for **a** the CarSim plant, and **b** the physical plant. In each figure, the *upper* surface belongs to KMPC, and the *bottom* belongs to DMPC. These two surfaces are close at some region, and diverge when v and δ increase

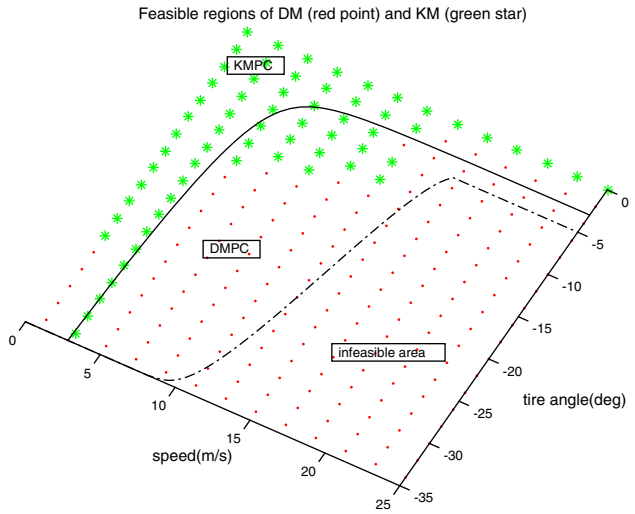


Fig. 12 The region of star points and the region of dot points are where the kinematic model and the dynamic model outperform, respectively. The solid curve is the switching boundary between KMPC and DMPC. The dashed line is the constraint on speed and steering angle, which means any point locating below the *dashed line* is an infeasible solution

$\mathbb{D}_q$ defines the conditions on $\hat{z}$ that trigger a change of q according to the law $1-q$, which flips the value of q at each switch. The set $\mathbb{C}_q$ defines the set of points where continuous evolution of the system is allowed. For more details about this approach to hybrid systems, see Goebel et al. (2012).

The construction of the hybrid algorithm leading to the hybrid closed-loop system (10) permits the following assertions regarding properties of its data and its dynamics:

(1) The sets $\mathbb{C}_0$, $\mathbb{C}_1$, $\mathbb{D}_0$, and $\mathbb{D}_1$ are closed;

(2) For each closed-loop trajectory, there exists a constant $\lambda > 0$ such that the time elapsed between every consecutive jump is lower bounded by λ.

The first property is key since, when for each $q \in \mathbb{Q}$ the function $\hat{z} \mapsto \hat{f}(\hat{z}, \kappa_q(\hat{z}))$ is continuous, the closed-loop system is a well-posed hybrid system as defined in (Goebel et al. 2012, Definition 6.2). Well-posedness implies that, over finite horizons, the closed-loop system is robust to small perturbations and if continuous feedback laws κ_q induce an asymptotic stability property, this property is robust to small perturbations (over the infinite horizon). The second property is crucial for any switching system as it guarantees that switches do not occur arbitrarily fast. Given that the right-hand side defining the continuous dynamics of our hybrid system is bounded, and the hysteresis-type switching mechanism implemented by our construction, this guarantees that there is a dwell time property in our closed-loop system. In order to increase the time λ beyond what is provided by the dynamics of the system, the value ϱ may be increased. In the simulations shown in Fig. 13a, $\lambda = 1$ s.

The CarSim trajectory controlled by the hybrid MPC is shown in Fig. 13a. Generally when the vehicle is turning or ensuring it has sufficient accuracy to avoid an obstacle, it utilizes the DMPC predictor; when it can go straight, it utilizes the KMPC predictor. Intuitively this matches the reality that the vehicle is more likely to go fast when going straight, and that straight trajectories for a car-like robot are not that different from Euler kinematic models. The trajectories generated by the hybrid controller, KMPC, and DMPC are shown in Fig. 13b, without additional annotations that demonstrate the mode switches. The DMPC performs similar to the hybrid controller, and each of them outperform the KMPC. The

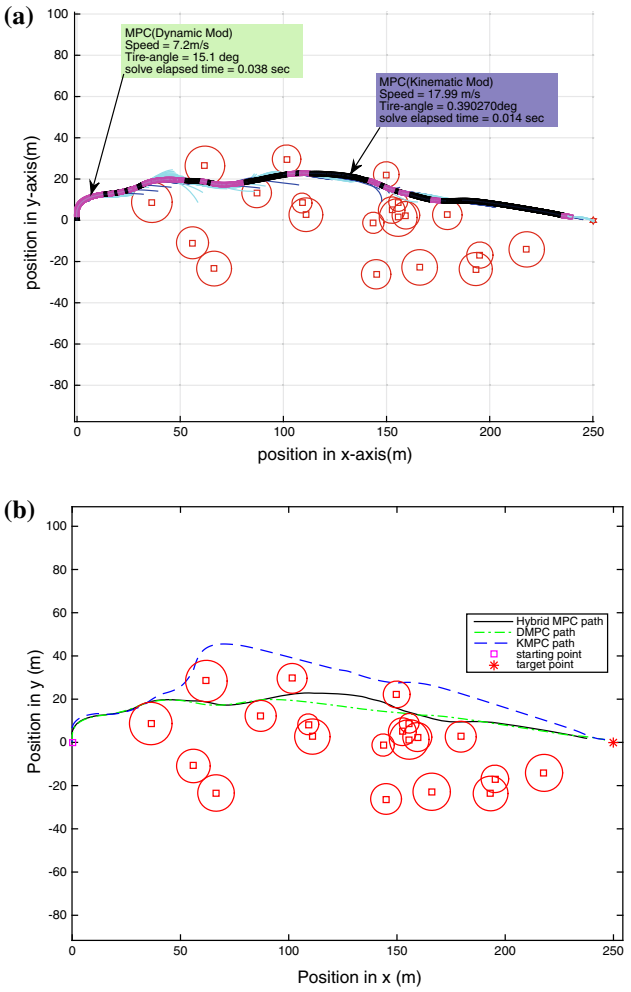


Fig. 13 **a** Trajectory generated by the hybrid controller when CarSim is utilized as the plant. The *magenta squares* are where DMPC is in use, while the *black squares* represent the use of KMPC. *Cyan* and *blue lines* are predicted path produced by DMPC and KMPC, respectively. Two samples are selected to indicate the vehicle state during the simulation. **b** Comparison of the plant trajectories between the KMPC, DMPC, and hybrid controller (Color figure online)

advantage to the hybrid controller over the DMPC is not just in its average return time, but its ability to drive at higher speeds since it will be able to leverage the KMPC controller at those speeds.

All simulations are run in real time using the CarSim plant. The return time per planning period is shown in Fig. 14, for each of the 3 controllers. The hybrid MPC dramatically improves the return period over the DMPC, though KMPC still has the fastest return time. However, as shown in the figures, the faster return time is not sufficient if the predicted error is so large that the vehicle cannot recover from a control input that drives it into an obstacle avoidance zone. This happens to the KMPC trajectory (the blue line) on several occasions, and serves as a reminder that faster return time is not always a substitute for a more accurate model.

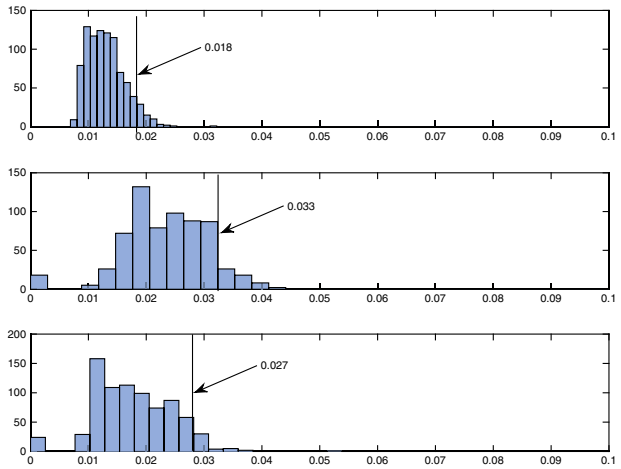


Fig. 14 Comparisons of return time. These three figures, from *top* to *bottom*, belongs to KMPC, DMPC and the hybrid controller, respectively. 700–900 samples of return time are collected for each controller. The abscissa is return time, and the ordinate is the number of the specific return time. The *solid lines* are where more than 90 % of return time samples fall bellow

6 Conclusion

This work proposes a metric called uncontrollable divergence, which accounts for tradeoffs of the divergence of model and plant caused by model mismatch by normalizing the diverging models according to the effort required to optimize with these models in a predictive controller. Based on this metric, a mechanism for switching between multiple predictive controllers is developed in order to lower the controller's return time while maintaining predictive accuracy.

We demonstrate how to calculate the uncontrollable divergence for two different plant models, one of which is a simulator, and the other of which is a physical vehicle plant. In the case of the physical plant, experiments are carried out to sample the data for the state space, but the entire state space cannot be sampled due to safety considerations: thus, a curve fitting approach is taken to estimate the uncontrollable divergence across the same region as in the simulator.

The results demonstrate the efficacy for this hybrid control approach, where the switching conditions are determined by the uncontrollable divergence of the simulator plant to the two predictor models. Simulations provide evidence that even in a complex environment ($n = 20$ obstacles) that the switching controller can outperform either single model predictor function.

The work is relevant to many kinds of problems in cyber-physical systems where the runtime of a computer-in-the-loop affects the performance of the system: if the models used by the computer can impact this runtime in various ways,

then the technique proposed in this paper may be used as a metric to determine whether switching between models poses a significant runtime benefit.

Future work involves investigation of the stability and robustness of the controllers. Our formalism required the construction of a hysteresis function to ensure that the system does not switch rapidly between models (potentially leading to instability). The investigation of conditions for stability that would not require such a construction as an explicit part of the model is key in our future work. We are also interested in switching controllers where more than two predictive models may be used, and issues of whether switching between models must be constrained to some portions of the state space.

Acknowledgments The work by K. Zhang and J. Sprinkle is supported in part by the National Science Foundation under Award CNS-1253334. Research by R. G. Sanfelice has been partially supported by the National Science Foundation under Grant No. ECS-1150306 and by AFOSR under Grant No. FA9550-12-1-0366.

References

Alamo, T., de la Pena, D. M., Limon, D., & Camacho, E. F. (2005). Constrained min-max predictive control: Modifications of the objective function leading to polynomial complexity. *IEEE Transactions on Automatic Control*, *50*(5), 710–714.

Alamo, T., de la Pea, D. M., & Camacho, E. F. (2008). An efficient maximization algorithm with implications in min-max predictive control. *IEEE Transactions on Automatic Control*, *53*(9), 2192–2197.

Alamo, T., de la Peña, D. M., & Camacho, E. F. (2008). Min-max MPC based on a network problem. *Systems & Control Letters*, *57*(2), 184–192.

Alamo, T., Tempo, R., & Camacho, E. F. (2009). Randomized strategies for probabilistic solutions of uncertain feasibility and optimization problems. *IEEE Transactions on Automatic Control*, *54*(11), 2545–2559.

Bahadorian, M., Savkovic, B., Eaton, R., & Hesketh, T. (2012). Robust model predictive control for automated trajectory tracking of an unmanned ground vehicle. In: *Proceedings of the American control conference (ACC), 2012* (pp. 4251–4256). IEEE.

Bemporad, A., & Rocchi, C. (2011). Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles. In: *Proceedings of the 50th IEEE conference on decision and control and european control conference (CDC-ECC), 2011* (pp. 7488–7493). IEEE.

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, *35*(3), 407–427.

Camacho, E. F., & Bordons, C. A. (1997). *Model predictive control in the process industry*. New York: Springer

Cannon, M., & Kouvaritakis, B. (2000). Continuous-time predictive control of constrained nonlinear systems. In *Nonlinear model predictive control* (pp. 205–215). New York: Springer.

Egerstedt, M., Hu, X., & Stotsky, A. (1998). Control of a car-like robot using a dynamic model. In: *Proceedings of the IEEE international conference on robotics and automation ICRA* (pp. 3273–3278). Citeseer.

Falcone, P., Tufo, M., Borrelli, F., Asgari, J., & Tsengz, H. (2007). A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. In: *Proceedings of the 46th IEEE conference on decision and control, 2007* (pp. 2980–2985). IEEE.

Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., & Hrovat, D. (2007). Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, *15*(3), 566–580.

Falcone, P., Borrelli, F., Tsengz, H., Asgari, J., & Hrovat, D. (2008). A hierarchical model predictive control framework for autonomous ground vehicles. In: *Proceedings of the american control conference, 2008* (pp. 3719–3724). IEEE.

Falugi, P., & Mayne, D. Q. (2014). Getting robustness against unstructured uncertainty: a tube-based MPC approach. *IEEE Transactions on Automatic Control*, *59*(5), 1290–1295.

Garcia, C. E., Prett, D. M., & Morari, M. (1989). Model predictive control: Theory and practice—a survey. *Automatica*, *25*(3), 335–348.

Gay, D. M., & Kernighan, B. (2002). Ampl: A modeling language for mathematical programming (vol. 2). Pacific Grove, CA: Duxbury Press/Brooks/Cole

Goebel, R., Sanfelice, R. G., & Teel, A. R. (2012). *Hybrid dynamical systems: Modeling, stability, and robustness*. Princeton, NJ: Princeton University Press.

Henson, M. A. (1998). Nonlinear model predictive control: Current status and future directions. *Computers and Chemical Engineering*, *23*(2), 187–202.

Kothare, M. V., Balakrishnan, V., & Morari, M. (1996). Robust constrained model predictive control using linear matrix inequalities. *Automatica*, *32*(10), 1361–1379.

Kuhne, F., Lages, W. F., & da Silva Jr, J. M. G. (2004). Model predictive control of a mobile robot using linearization. In: *Proceedings of mechatronics and robotics* (pp. 525–530).

Künhe, F., Gomes, J., & Fetter, W. (2005). Mobile robot trajectory tracking using model predictive control. In: *Proceedings of the II IEEE Latin-American robotics symposium*. IEEE

Løvaas, C., Seron, M i a M, & Goodwin, G . C. (2008). Robust output-feedback model predictive control for systems with unstructured uncertainty. *Automatica*, *44*(8), 1933–1943.

Mayne, D. Q. (2014). Model predictive control: Recent developments and future promise. *Automatica*, *50*(12), 2967–2986.

Narby, E. (2006). Modeling and estimation of dynamic tire properties. Master's Thesis, Linkpings Universitet, Linkpings.

Parker, R. S., Gatzke, E. P., Mahadevan, R., Meadows, E. S., & Doyle, F. (2001). Nonlinear model predictive control: Issues and applications. *IEE Control Engineering Series* (pp. 33–58).

Qin, S. J., & Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, *11*(7), 733–764.

Rawlings, J. B. (1999). Tutorial: Model predictive control technology. In: *Proceedings of the american control conference, 1999* (vol. 1, pp. 662–676). IEEE.

Richards, A. (2005). Robust model predictive control for time-varying systems. In: *Proceedings of the 44th IEEE conference on decision and control, 2005 European Control Conference CDC-ECC'05* (pp. 3747–3752). IEEE.

Richards, A., & How, J. P. (2002). Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In: *Proceedings of the american control conference, 2002* (vol. 3, pp. 1936–1941). IEEE.

Schubert, R., Richter, E., & Wanielik, G. (2008). Comparison and evaluation of advanced motion models for vehicle tracking. In: *Procedings of the 11th international conference on information fusion, 2008* (pp. 1–6). IEEE.

Walsh, G., Tilbury, D., Sastry, S., Murray, R., & Laumond, J.-P. (1994). Stabilization of trajectories for systems with nonholonomic constraints. *IEEE Transactions on Automatic Control*, *39*(1), 216–222.

Whitsitt, S., & Sprinkle, J. (2012). A passenger comfort controller for an autonomous ground vehicle. In: *Proceedings of the 51st IEEE conference on decision and control* (pp. 3380–3385). IEEE

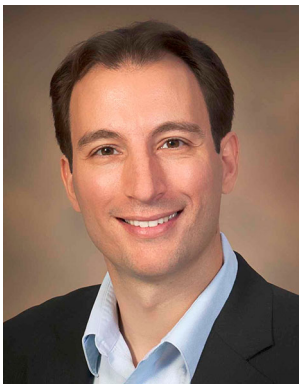
Zeilinger, M. N., Raimondo, D. M., Domahidi, A., Morari, M., & Jones, C. N. (2014). On real-time robust model predictive control. *Automatica*, *50*(3), 683–694.

Zhang, K., Sprinkle, J., & Sanfelice, R. G. (2015). A hybrid model predictive controller for path planning and path following. In: *Proceedings of the international conference on cyber-physical systems (ICCPS), Seattle, WA* (pp. 139–148). New York: ACM.

Zhao, Z., Xia, X., Wang, J., Gu, J., & Jin, Y. (2003). Nonlinear dynamic matrix control based on multiple operating models. *Journal of Process Control*, *13*(1), 41–56.

Kun Zhang is a Ph.D. student of Electrical and Computer Engineering at the University of Arizona. His research interests are domain-specific modeling, cyber-physical systems and software engineering.

Jonathan Sprinkle is an Associate Professor of Electrical and Computer Engineering at the University of Arizona. In 2013 he received the NSF CAREER award, and in 2009, he received the UA's Ed and Joan Biggers Faculty Support Grant for work in autonomous systems. His work has an emphasis for industry impact, and he was recognized with the UA "Catapult Award" by Tech Launch Arizona in 2014, and in 2012 his team won the NSF I-Corps Best Team award. His research interests and experience are in systems control and engineering, and he teaches courses ranging from systems modeling and control to mobile application development and software engineering.



Ricardo G. Sanfelice is an Associate Professor at the Department of Computer Engineering, University of California at Santa Cruz. He received the B.S. degree in Electronics Engineering from the Universidad Nacional de Mar del Plata, Buenos Aires, Argentina, in 2001. He joined the Center for Control, Dynamical Systems, and Computation at the University of California, Santa Barbara in 2002, where he received his M.S. and Ph.D. degrees in 2004 and 2007, respectively. During 2007 and 2008, he was a Postdoctoral Associate at the Laboratory for Information and Decision Systems at the Massachusetts Institute of Technology. He visited the Centre Automatique et Systemes at the Ecole de Mines de Paris for 4 months. From 2009 to 2014, he was Assistant Professor in the Aerospace and Mechanical Engineering at the University of Arizona, where he was also affiliated with the Department of Electrical and Computer Engineering and the Program in Applied Mathematics.