Title

A hybrid model predictive controller for path planning and path following

ISBN

9781450334556

Authors

Zhang, Kun
Sprinkle, Jonathan
Sanfelice, Ricardo G

Publication Date

2015-04-14

DOI

10.1145/2735960.2735966

Peer reviewed

A Hybrid Model Predictive Controller for Path Planning and Path Following

Kun Zhang
University of Arizona
Tucson, AZ
dabiezu@email.arizona.edu

Jonathan Sprinkle
University of Arizona
Tucson, AZ
sprinkle@acm.org

Ricardo G. Sanfelice
University of California
Santa Cruz, CA
ricardo@ucsc.edu

ABSTRACT

The use of nonlinear model-predictive methods for path planning and following has the advantage of concurrently solving problems of obstacle avoidance, feasible trajectory selection, and trajectory following, while obeying constraints on control inputs and state values. However, such approaches are computationally intensive, and may not be guaranteed to return a result in bounded time when performing a non-convex optimization. This problem is an interesting application to cyber-physical systems due to their reliance on computation to carry out complex control. The computational burden can be addressed through model reduction, at a cost of potential (bounded) model error over the prediction horizon. In this paper we introduce a metric called uncontrollable divergence, and discuss how the selection of the model to use for the predictive controller can be addressed by evaluating this metric, which reveals the divergence between predicted and true states caused by return time and model mismatch. A map of uncontrollable divergence plotted over the state space gives the criterion to judge where reduced models can be tolerated when 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 hybrid controller that switches at runtime between predictive controllers in which respective models are deployed.

Keywords

Hybrid Control, MPC, Model Error Evaluation

1. INTRODUCTION

Model Predictive Control (MPC) produces an optimized control input sequence for a system based on performance of a predictive model according to a cost function under constraints. The optimization solution is a sequence of control inputs and state predictions over a finite horizon. The motivation of this work comes from the need to synthesize obstacle-avoiding trajectories for a path-planning ground vehicle which may be carring passengers. The vehicle (system) will need to synthesize and then follow trajectories to arrive at a final or intermediate destination while meeting a velocity and heading constraint at that location and satisfying other state constraints all along the trajectory.

MPC has some attractive features for systems such as these, where desired trajectories may be infeasible and safety constraints can be specified in terms of state and control variables: (a) it handles constraints systematically as part of the cost function; and (b) it guarantees feasibility of the trajectory through the use of a predictive model that approximates the system plant over the horizon.

There are well-known limitations of the use of MPC at runtime. The optimization process is computationally intensive for nonlinear plants (such as car-like robots) whose optimal solution cannot be expressed in closed form [3]. Further, the online optimization is not guaranteed to complete in real time [9] or may return a local optimum [16].

Approaches to mitigate these risks fall into three major categories: (i) enforce timing constraints if a solution has not yet been found; (ii) operate the system in degraded mode until a solution is found; or (iii) select a predictive model that only approximates the plant model, but which guarantees (or significantly improves the guarantee) that a solution will be obtained in time.

Approach (i), enforcing the timing constraints, can be performed in several ways. Multiple optimization engines could be executed at runtime, and the first one to return a feasible solution is selected. The most draconian approach is to cancel the optimization routine after a certain time and proceed with the best solution found so far. This approach faces the risk that no feasible solution may have been obtained when the routine is halted.

Approach (ii), operating in degraded mode, has a few alternative solutions. One is to continue to execute the control inputs from the previous solution until a new, optimal, solution is returned. The major limitation of this approach is that the solution will be for a timestep that has already passed, and any new constraint issues (e.g., to avoid undesired regions of the system state) will not have been accounted for since the last solution. This is particularly problematic if the solution to the optimization takes several timesteps to calculate. In situations such as these, the system behavior at runtime might be modified until a viable solution is returned. As an example, a vehicle might reduce its velocity until it gets a feasible solution. A limitation of this approach is that operation in degraded mode might affect nonfunctional metrics such as passenger comfort.

Approach (iii), selecting a reduced-complexity predictive model, helps to provide a significant margin for robustness to potential disturbances. When executed in some areas of the state space, this approach will execute reliably—e.g, a linearization around an operating point. When the system must be operated in a region of the state space where linearization cannot be effective, an alternative approach is to select reduced order predictive models that are known to

be simple to use when computing an optimal solution. An example of this approach is to select the kinematic (rather than the dynamic) model for problems such as path planning and following. However, as described in [5], for a car-like robot the kinematic model diverges more quickly than the dynamic model when they are used in prediction.

Contribution of this work

The contribution of this paper is a unique combination of approaches (iii) and (ii) which in turn improve guarantees such that approach (i) need not be considered. We describe the use of a hybrid controller that switches between predictive models when computing the optimal solution to the cost function. By selecting the appropriate model when performing the search for the optimal control input vector, we can reduce the time needed to compute the optimization solution (if we need to go fast), or we can reduce the error of the predictive model (if we need to be accurate).

As the astute reader realizes, a naive approach to this design is that the system may need to compute in time while being accurate, or may need to be accurate while going fast. In order to meet these two goals, the paper describes how we define the switching criteria for the hybrid model through a metric we introduce called the *uncontrollable divergence* (UD) of the system. It is calculated using the error of the predictive models to the plant model, and the state change that occurs while awaiting the optimizer's result.

In this work we take both the kinematic and dynamic models of our ground vehicle and quantify the UD (within the same MPC implementation) over the entire state space. We will show that within some state space area (e.g., high speed with low steering angle), the reduced model can have similar UD as that of high fidelity model but provides quicker return of solutions. In this region of the state space, we say the reduced model outperforms the high fidelity model.

The reason this is valid is that no model would never be used at high speed with a high steering angle, 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 *at low speed, the system can afford to wait longer for the optimization routine to return, since the vehicle is traveling slower!*

Organization

The rest of the paper is organized out as follows. We formulate the model-predictive control problem that we aim to solve, and explicitly lay out the mathematical forms of our predictive models and cost functions. We then formally state the problem and our approach to its solution. Next, we demonstrate how to calculate the uncontrollable divergence of this sytem, and how to use that metric to define the hybrid predictive controller. We then provide evidence that the designed system improves on the timing and accuracy metrics for improved behavior as described in items (i), (ii), and (iii) above. Finally, we also demonstrate that the switching conditions of our controller agree with previous research results for the application we selected, 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 Related Work

Predictive control for trajectory synthesis and path following of a car-like robot utilizes a nonlinear vehicle model with time-varying constraints. Linearized MPC (LMPC) has the advantages over nonlinear approaches with its low computational cost [14] and avoidance of the occurrence of non-convex programming which is common in NMPC [11]. As shown in [18], [9] and [13], 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).

Works such as [19, 1] 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.

In [17] various MPC computational schemes are surveyed and commercially available vendors are described. In this paper we applied AMPL [20] and minos [10] as our optimization tools and we use MATLAB to integrate the optimization results into a cohesive simulation.

Several works address unbounded return time of the optimizer through an approach called *layering*. In [6, 8, 2], 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.

2.2 MPC Formulation

Consider the nonlinear continuous time plant:

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

where $\xi \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is the control input vector, and $f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$. Let superscript $\hat{}$ denote discretization by a sample time of ΔT seconds , and $\hat{f}$ denote the discrete time plant:

$$\xi_{k+1} = \hat{f}(\xi_k, u_k) \tag{2}$$

where $\hat{f} : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$, and $k \in \{0, 1, \cdots\}$ denotes discrete time.

To differentiate the plant's states and MPC's predicted states (as shown in Figure 1), let subindex t represent the discrete time over the prediction horizon $N \in \{1, 2, \cdots\}$,

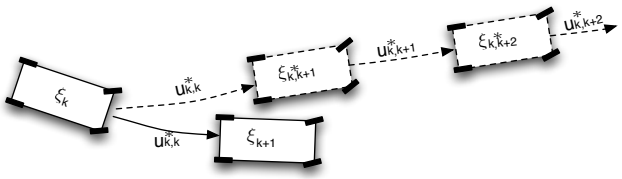


Figure 1: The solid squares represent the plant, and the dashed squares represent the prediction. Note that, only two predicted states, $\xi^*_{k,k+1}$ and $\xi^*_{k,k+2}$, are shown in the figure. The divergence between solid and dashed lines indicates model mismatch.

therefore, $\xi_{k,t}$ denotes the predicted state at time t when the initial state ξ_k and all planned inputs $u_{k,k}, \cdots, u_{k,t-1}$ up until $t-1$ are given.

For convenience, let $\xi_{k,k}$ be equivalent to ξ_k; then the nonlinear model used by MPC for prediction is

$$\begin{aligned} \xi_{k,t+1} &= \hat{f}_q(\xi_{k,t}, u_{k,t}) \\ t &\in \{k, k+1, \cdots, k+N-1\} \end{aligned} \tag{3}$$

where $\hat{f}_q : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and ξ_k is fed into the prediction as the initial state. The integer $q \in \mathbb{Q} := \{0, 1, \cdots\}$ is the index of the predictive model currently in use, and the size of $\mathbb{Q}$ indicates the number of available models. We define model mismatch $\hat{\Gamma}_q : \mathbb{R}^n \to \mathbb{R}^n$ as:

$$\hat{\Gamma}_q(\xi) := \hat{f}(\xi, u) - \hat{f}_q(\xi, u)$$

which captures the model error between the discrete time plant $\hat{f}$ and the q-th predictive model $\hat{f}_q$.

MPC solves the optimization problem $\mathbf{P}^q(\xi_k)$ at time k by using the model $\hat{f}_q$. We denote the input sequence $\{u^q_{k,k}, u^q_{k,k+1}, \cdots, u^q_{k,k+N-1}\}$ by U^q_k, and formulate the following problem:

$$\mathbf{P}^q(\xi_k) : \underset{U^q_k}{\operatorname{argmin}} \{J_N(\xi_k, U^q_k) : U^q_k \subset \mathbb{R}^m\} \tag{4}$$

$$J_N(\xi_k, U^q_k) = \sum_{t=k}^{k+N-1} \ell(\xi^q_{k,t}, u^q_{k,t}) + F(\xi^q_{k,k+N}) \tag{5}$$

where $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^+$ is the stage cost function, and $F : \mathbb{R}^n \to \mathbb{R}^+$ is the terminal cost function.

Denote the optimal solution to $\mathbf{P}^q(\xi_k)$ by:

$$U^{q*}_k = \{u^{q*}_{k,k}, u^{q*}_{k,k+1}, \cdots, u^{q*}_{k,k+N-1}\}$$

By feeding U^{q*}_k into the model $\hat{f}_q$, we obtain the optimal prediction sequence $\Xi^{q*}_k = \{\xi^{q*}_{k,k+1}, \cdots, \xi^{q*}_{k,k+N-1}, \xi^{q*}_{k,k+N}\}$. The first input $u^{q*}_{k,k}$ is applied to the plant in (2), therefore, the implicit control law is $\kappa_q(\xi_k) := u^{q*}_{k,k}$. Then, at k we obtain

$$\xi_{k+1} = \hat{f}\left(\xi_k, \kappa_q(\xi_k)\right)$$

Then the hybrid controller may select a $q' \in \mathbf{Q}$ (either $q' = q$ or $q' \neq q$) and solves problem $\mathbf{P}^{q'}(\xi_{k+1})$, and continues this process indefinitely. Note that, if we count the MPC return time in, the plant would have a displacement $\Delta\xi_k \in \mathbb{R}^n$ from the observed ξ_k during the optimization process, and the plant state at $k+1$ would actually be:

$$\xi_{k+1} = \hat{f}\left(\xi_k + \Delta\xi_k, \kappa_q(\xi_k)\right)$$

To construct a LMPC at time k, as described in [7], one can linearize the model $\hat{f}_q$ over the previous optimal solution $u^*_{k-1,t} \in U^*_{k-1}$ and prediction $\xi^*_{k-1,t} \in \Xi^*_{k-1}$:

$$\begin{aligned} \xi^q_{k,t+1} &= A_{k-1,t}(\xi^q_{k,t} - \xi^*_{k-1,t}) + B_{k-1,t}(u^q_{k,t} - u^*_{k-1,t}) \\ &\quad + \hat{f}_q(\xi^*_{k-1,t}, u^*_{k-1,t}) \end{aligned} \tag{6}$$

where $A_{k-1,t} = \left.\frac{\partial \hat{f}_q}{\partial \xi}\right|_{\xi^*_{k-1,t}, u^*_{k-1,t}}$, $B_{k-1,t} = \left.\frac{\partial \hat{f}}{\partial u}\right|_{\xi^*_{k-1,t}, u^*_{k-1,t}}$. Note that, $A_{k-1,t}$, $B_{k-1,t}$ and the last term in (6) can be obtained in advance of the current optimization computation.

Hence, the MPC solves optimization problem $\mathbf{P}^q(\xi_k)$ in (4) subject to constraint in (6). Other constraints are introduced in the following subsections.

2.3 Vehicle Models

In this work, two vehicle models are selected ($\mathbb{Q} = \{0, 1\}$) as available predictive models for an MPC controller. The continuous-time models in (7) and (8) are discretized into $\hat{f}_0$ and $\hat{f}_1$, respectively. The kinematic model ($q = 0$) is simplified significantly from the plant, with the dynamic model ($q = 1$) being relatively more accurate when compared to the plant model.

2.3.1 Kinematic Model

The kinematic model we use is from [22]. Let $\xi = [x, y, \theta]^T$, as shown in Figure 2a, where θ is the azimuth. The model is described in (7).

$$\dot{\xi} = \begin{bmatrix} v\sin\theta \\ v\cos\theta \\ \frac{v\tan\delta}{L} \end{bmatrix} \tag{7}$$

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

2.3.2 Dynamic Model

For the dynamical predictive model we utilize the form developed in [15]. Specific customizations for this work utilize 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 that the slip angle equals the steering angle. Using vehicle mass as m, and orientational speed by φ, the simplified dynamical model, with acceleration a and steering angular speed ω as control inputs $u = [a, \omega]^T$, is shown in Figure 2a and (8), where $\xi = [x, y, v, \theta, \varphi, \delta]^T$, and steering angle is δ.

$$\dot{\xi} = \begin{bmatrix} v\sin(\theta) \\ v\cos(\theta) \\ \cos(\delta)a - \frac{2}{m}F_{y,f}\sin(\delta) \\ \varphi \\ \frac{1}{J}\left(L_a\left(ma\sin(\delta) + 2F_{y,f}\cos(\delta)\right) - 2L_bF_{y,r}\right) \\ \omega \end{bmatrix} \tag{8}$$

where 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, and J is the rotational momentum. $F_{y,f}$ is

the front tire lateral force, $F_{y,r}$ is the rear tire lateral force. 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 this model is more complex than that of (7). In order to determine whether it accurately maps to our plant, however, we must define the nonlinear plant model we will use for executing the selected control inputs.

2.4 Nonlinear Plant

Regardless of which predicted model we choose, we send the same inputs through our plant model. In this paper, we use a car-like robot model developed in CarSim [4].

CarSim is a rich vehicle simulation environment which is used in many industrial applications to establish a 50+ degree of freedom simulation of a vehicle. In this case, we configured our plant to have the same wheelbase L and mass m as a physical testbed. Since that testbed is not used in this work, we consider the CarSim model to be the actual system plant, and we will compare the behavior of our dynamical and kinematic predictor models when the control inputs we generated from those models are propagated through the CarSim plant.

We gathered tire stiffness data from CarSim to use in (9). The tire stiffness data, represented as the relationship among lateral force, tire load and steering angle, is summarized in Figure 2b, which was generated from the CarSim tool.

2.5 Vehicle-specific Constraints

Our work is motivated by the application of trajectory synthesis and path following for an obstacle-avoiding car-like robot. In this subsection we discuss the constraints which a solution must satisfy for such a vehicle, namely: speed, acceleration, and deceleration (braking), in (9):

$$\begin{aligned} v_{min} < v &< v_{max} \\ a^f &< a^f_{max} \\ \left|a^b\right| &< a^b_{max} \end{aligned} \tag{9}$$

where a^f and a^b are acceleration and deceleration, respectively. (10) constrains the steering angle and its derivative.

$$\begin{aligned} |\delta| &< \delta_{max} \\ |\omega| &< \omega_{max} \end{aligned} \tag{10}$$

To prevent unsafe behavior (such as flipping the vehicle) we consider that the steering angle is constrained by speed as in [23, 12]. We denote the upper bound of δ by $\delta = g(v)$, where $g : \mathbb{R} \to \mathbb{R}$. Thus, a linearized constraint at time k on $\delta_{k,t}$ and $v_{k,t}$ should hold $\forall t \in \{k+1, \cdots, k+N\}$:

$$\delta_{k,t} \leqslant g(v^*_{k-1,t-1}) + \left.\frac{\mathrm{d}g(v)}{\mathrm{d}v}\right|_{v^*_{k-1,t-1}} (v_{k,t} - v^*_{k-1,t-1}) \tag{11}$$

Finally, constraints on position are enforced for obstacles. Some works [2] create convex polyhedrons to depict the feasible regions. In this paper we impose a linearized distance inequality (from vehicle to obstacle) to constrain system behavior. We assume that the obstacles are detected at run time by a range-limited sensor, and not known *a priori*. If we assume obstacles are circular zones, then (x_{obs}, y_{obs}) is the obstacle's position and r_{obs} is the radius. $x_{k,t}$ and $y_{k,t}$ are predicted positions in $\xi_{k,t}$. We denote the distance from the point $(x_{k,t}, y_{k,t})$ to the obstacle center by:

$$D(x_{k,t}, y_{k,t}) := \sqrt{(x_{k,t} - x_{obs})^2 + (y_{k,t} - y_{obs})^2} \geqslant r_{obs}$$

and we can obtain (12) by linearizing the above inequality.

$$\begin{aligned} D_A x_{k,t} + D_B y_{k,t} + D_C &\geqslant r_{obs} \\ \forall t \in \{k+1, \cdots, k+N\} & \end{aligned} \tag{12}$$

where

$$D_A = \left.\frac{\partial D}{\partial x}\right|_{\xi^*_{k-1,t-1}}$$
$$D_B = \left.\frac{\partial D}{\partial y}\right|_{\xi^*_{k-1,t-1}}$$
$$\begin{aligned} D_C = D(x^*_{k-1,t-1}, y^*_{k-1,t-1}) - \\ D_A x^*_{k-1,t-1} - D_B y^*_{k-1,t-1} \end{aligned}$$

With these constraints on the state and control inputs, plus the various predictive and plant models, we are ready to state the problem to be solved.

3. PROBLEM STATEMENT & APPROACH

We propose a hybrid controller which implements supervisory logic for a predictive controller, as shown in Figure 4b. Suppose at time $k \in \{0, 1, \cdots\}$, the vehicle state ξ_k is observed for an optimization problem indexed by the predictive model in use (i.e., $\mathbf{P}^q(s_k)$), and that two alternative predictive models are available (shown in Figure 3).

Problem: select the predictive model q such that the divergence of the state at ξ_k from the plant's state with the same inputs is minimized.

Approach: assume that the return time to solve $\mathbf{P}^q(\xi_k)$ for each predictive model q can be estimated as Δt_q. After time Δt_q the system state will be $\xi'_k \approx \xi_k + f(\xi_k, u^*_{k-1,k})\Delta t$ when the control is received (the vehicle executes the remaining control sequences generated from $\mathbf{P}^q(\xi_{k-1})$). Thus, the vehicle state at $k+1$ is:

$$\begin{aligned} \xi_{k+1} &= \hat{f}\left(\xi'_k, \kappa_q(\xi_k)\right) = \hat{f}_q\left(\xi'_k, \kappa_q(\xi_k)\right) + \hat{\Gamma}_q(\xi'_k) \\ &\approx \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right) + \hat{\Gamma}_q(\xi_k) + \left(\frac{\partial \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi}\right. \\ &\quad \left. + \frac{\partial \hat{\Gamma}_q\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi}\right) f\left(\xi_k, \kappa_q(\xi_k)\right) \Delta t_q(\xi_k) \end{aligned} \tag{13}$$

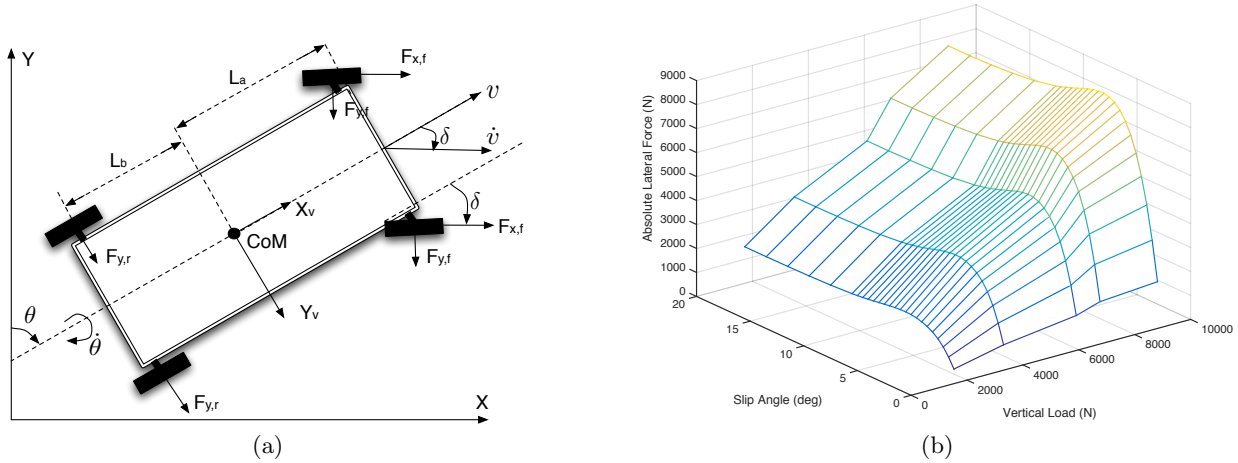
Since $\xi^{q*}_{k,k+1} = \hat{f}_q\left(\xi_k, \kappa_q(\xi_k)\right)$, we obtain the following from (13):

$$\xi_{k+1} - \xi^{q*}_{k,k+1} \approx \hat{\Gamma}_q(\xi_k) + \frac{\partial \hat{f}\left(\xi_k, \kappa_q(\xi_k)\right)}{\partial \xi} f\left(\xi_k, \kappa_q(\xi_k)\right) \Delta t_q(\xi_k) \tag{14}$$

The logic of q selection when solving $\mathbf{P}^q(\xi_k)$ should minimize the prediction and actual state at $k+1$:

$$q = \underset{q}{\operatorname{argmin}} \left\|\xi_{k+1} - \xi^{q*}_{k,k+1}\right\| \tag{15}$$

We name the value of $\left\|\xi_{k+1} - \xi^{q*}_{k,k+1}\right\|$ as uncontrollable divergence (UD). To simplify the equation (14) further for the



(a) (b)

Figure 2: (a) Schematic view of the vehicle model. $X_v \perp Y_v$ is the vehicle body-fixed coordinate. (b) Absolute best (treated as steering angle in this work).

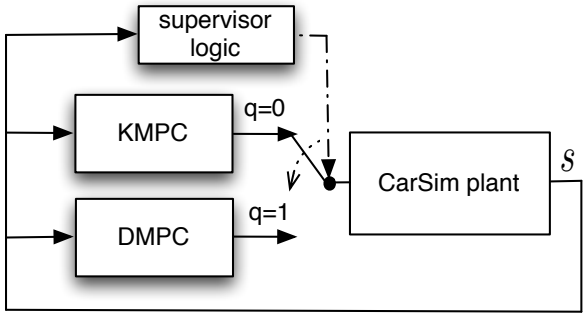


Figure 3: Closed-loop system. Hybrid MPC and CarSim plant are in the loop.

specific vehicle system, we take the kinematic model (7) to estimate the values of $\frac{\partial \hat{f}(\xi_k, \kappa_q(\xi_k))}{\partial \xi}$ and $f(\xi_k, \kappa_q(\xi_k))$, and use the upper bound of $\left\| \xi_{k+1} - \xi^{q*}_{k,k+1} \right\|$ to find q:

$$\begin{aligned} &\left\| \xi_{k+1} - \xi^{q*}_{k,k+1} \right\| \leqslant \left\| \hat{\Gamma}_q(\xi_k) \right\| \\ &+ \left\| I + \frac{\partial f(\xi_k, \kappa_q(\xi_k))}{\partial \xi} \Delta T \right\| \left\| f(\xi_k, \kappa_q(\xi_k)) \right\| \Delta t_q(\xi_k) \\ &\approx \left\| \hat{\Gamma}_q(\xi_k) \right\| + v \Delta t_q(\xi_k) \sqrt{1 + \left(\frac{\tan\delta}{L} v \Delta T \right)^2} \end{aligned} \tag{16}$$

where I is an identity matrix.

We will show that $\hat{\Gamma}_q$ is a function of $v \in \xi$ and $\delta \in \xi$ in the following section. Thus, prior to running the hybrid controller, the upper bound value of $\left\| \xi_{k+1} - \xi^{q*}_{k,k+1} \right\|, \forall q \in \mathbf{Q}$ over finite samplings of bounded v and δ can be obtained. Then the hybrid controller can take advantage of the switching logic in (15).

Assume two models are applied to synthesize two MPCs: the kinematic model ($\hat{f}_0(\xi, u)$) produces the Kinematic MPC (KMPC), and the dynamic model ($\hat{f}_1(\xi, u)$) produces the Dynamic MPC (DMPC). The commercial vehicle modeling software, CarSim, is used as the true plant in our simulations.

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

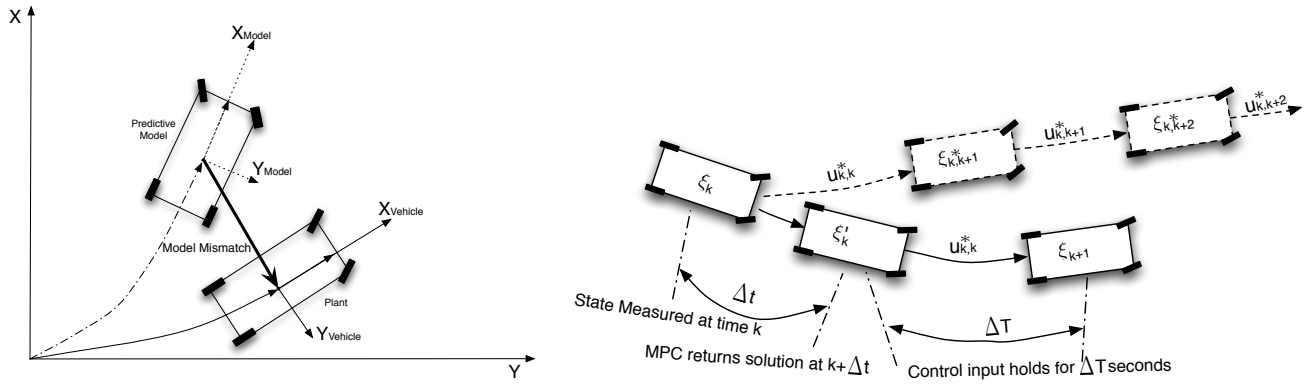
4. DIVERGENCE & RETURN TIME

The principal characteristics of the uncontrollable divergence revolve around the evolving state of the vehicle while calculating the solution to $\mathbf{P}^q(s_k)$. In our particular example, the vehicle is moving during this time, so the state evolves for that time duration. However, we discuss how to consider that in some regions of the state space, the model divergence between the two predictive models is low: so if the return time is faster, a model with higher error (but fast control input results) may actually be preferred to an accurate model.

4.1 Model Divergence

We follow the divergence calculation concepts from [21]. Predictive models 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. As shown in Figure 4a, we define the model error in the vehicle's body-fixed coordinates ($X_M \perp Y_M$) as the correcting vector $[x' \; y' \; \theta']^T$ (T represents matrix transpose) pointing from model to plant. With the predictive model (indexed by q) and the plant, each starting from the same initial state ξ with the same constant input u, each diverging into model state $\hat{f}_q(\xi, u)$ and vehicle state $\hat{f}(\xi, u)$ after the period ΔT ($\Delta T = 0.1sec$ in simulation), then we have model mismatch $\Gamma_q(\xi) = \hat{f}(\xi, u) - \hat{f}_q(\xi, u)$. Since the predictive model and the plant share the same tire angle δ and speed v, the model mismatch value can be obtained by the following:

$$\left\| \hat{\Gamma}_q(\xi) \right\| = \left\| \begin{bmatrix} x' & y' & \theta' \end{bmatrix}^T \right\|$$



(a) a (b) b

Figure 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. A good controller should minimize the difference between ξ_{k+1} and $\xi^*_{k,k+1}$. A high-fidelity model with unbounded return time Δt can cause unbounded divergence. A reduced model with smaller Δt would produce larger divergence during ΔT.

Figure 5a and Figure 5b reveal example positional and azimuthal divergence rates.

4.2 Return Time of MPC

Simulations were conducted to measure the return time of KMPC and DMPC. For the convenience of comparison, KMPC and DMPC are simulated under the same circumstances (the same initial state, obstacles and target state). Results of elapsed time are shown in Figure 6, and the expected values of return time are selected as $0.02sec$ (KMPC) and $0.05sec$ 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), there is still no guarantee for either approach that the algorithms will return in bounded time. Therefore, we are careful not to claim that our approach will always return a result: rather, we instead use a knowledge of the expected return time to select the model whose return time and error divergence agree with our constraints on accuracy.

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 controller as the interface for converting target speed and steering angle (elements in $\xi^*_{k,k+1}$) to controls on throttle, cylinder pressure and steering torque. Step responses of the interface controller are shown in Figure 7 to demonstrate its ability to stabilize itself to the target state within the required time.

When the CarSim plant is applied, vehicle trajectory diverges from the predicted path in an obvious manner, as shown in Figure 8a and Figure 8b. Such divergences originate from model mismatch. For example, the predictive model would have a larger curvature than that of the plant, when given a specific steering angle and speed. Thus, this steering angle value that the MPC thinks is adequate to reach a certain curvature would probably produce an inadequate curvature in the plant, and the vehicle can rush into the obstacle and then normally will bounce out (or return infeasible solution of $\mathbf{P}^*(\xi_k)$). As a consequence, the predicted path will oscillate, and the actual trajectory does not fit the obstacle boundary well.

5. HYBRID PREDICTIVE CONTROLLER

By taking the $\left\|\hat{\Gamma}_q\right\|, \forall q \in \mathbf{Q} = \{0, 1\}$ and $\Delta t_q, \forall q \in \mathbf{Q}$ obtained above into (16), we plot out the upper bound of UDs in Figure 9. Recall the switching logic in (15), to speed up at run time, we plot out the switching boundary in Figure 10a based on Figure 9 and let the supervisory logic inquiry the figure instead of computing the $argmin$ for q.

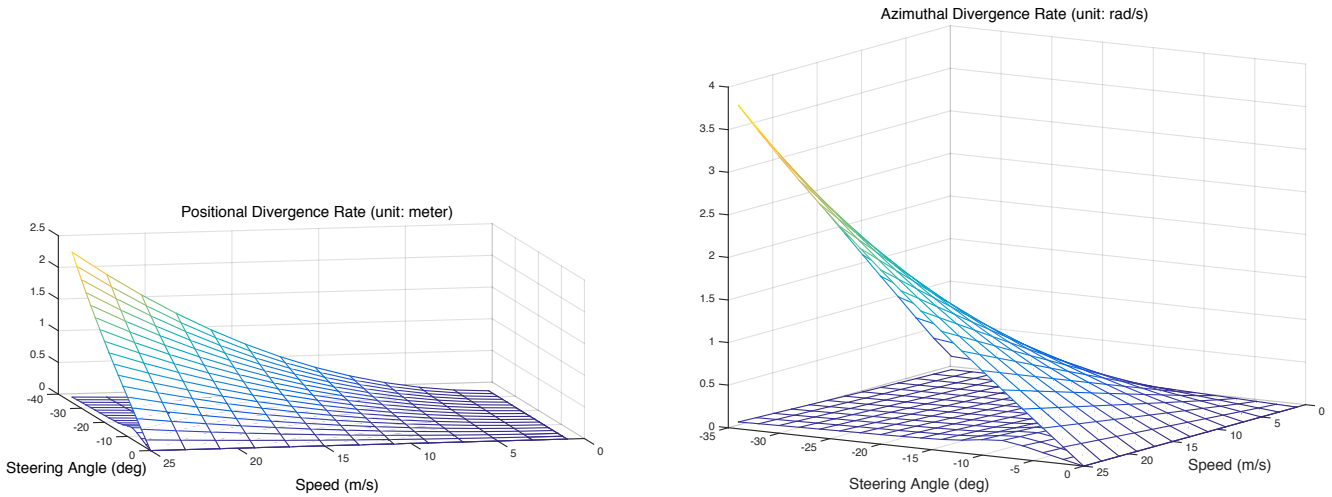
The abscissa and ordinate of Figure 10a are velocity v and steering angle δ. The asterisk points region (upside) is where kinematic model outperforms. The other region belongs to the dynamic model. We denote the switching boundary (the solid line in Figure 10a) between KMPC ($q = 0$) and DMPC ($q = 1$) by $\Omega : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and fit the boundary by tuning a constant $c > 0$:

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

The mutual constraint between speed and steering angle described in (11) is also plotted in the figure as a dashed curve. Let $q \in \{0, 1\}$ for the MPC control law $\kappa_q(\xi)$,where 0 indicates KMPC, and 1 indicates DMPC. Let $\varepsilon := \begin{bmatrix} q & \xi^T \end{bmatrix}^T \in \{0, 1\} \times \mathbb{R}^n$, we describe the closed-loop system, as shown in Figure 3, in (17).

$$\mathfrak{H} = \begin{cases} \varepsilon \in (\{0\} \times \mathbb{C}_0) \cup (\{1\} \times \mathbb{C}_1), \varepsilon^+ = \begin{bmatrix} q \\ \hat{f}\left(\xi, \kappa_q(\xi)\right) \end{bmatrix} \\ \varepsilon \in (\{0\} \times \mathbb{D}_0) \cup (\{1\} \times \mathbb{D}_1), \varepsilon^+ = \begin{bmatrix} 1 - q \\ \xi \end{bmatrix} \end{cases} \quad (17)$$

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



(a) (b)

Figure 5: (a). Comparison of positional divergence rate ($\left\|[x'\ y']^T\right\|/\Delta T$) of the dynamic model (the bottom surface) and the kinematic model(the upper surface). (b). Comparison of azimuthal divergence rate ($\|\theta'\|/\Delta T$) of the dynamic model (the bottom surface) and the kinematic model (the upper surface).

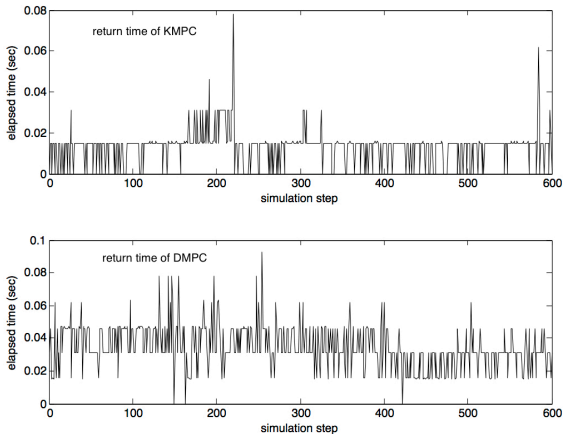


Figure 6: Comparison of expected return time of KMPC $\Delta t_q\ |_{q=0}= 0.02$ and DMPC $\Delta t_q\ |_{q=1}= 0.05$.

defined in (18).

$$\begin{aligned}\mathbb{C}_0 &= \{\xi \in \mathbb{R}^n : \Omega(v,\delta) - \varrho < 0\}\\ \mathbb{D}_0 &= \{\xi \in \mathbb{R}^n : \Omega(v,\delta) - \varrho \geqslant 0\}\\ \mathbb{C}_1 &= \{\xi \in \mathbb{R}^n : \Omega(v,\delta) + \varrho > 0\}\\ \mathbb{D}_1 &= \{\xi \in \mathbb{R}^n : \Omega(v,\delta) + \varrho \leqslant 0\}\end{aligned} \tag{18}$$

The CarSim trajectory controlled by the hybrid MPC is shown in Figure 10b. We plot out the trajectories generated by the hybrid controller, KMPC and DMPC in Figure 11. The DMPC has similar performance as the hybrid controller, and they both outperform the KMPC. Figure 12 shows the deviations between trajectories and predictions. The KMPC deviation oscillates more frequently than the DMPC, due to its low control accuracy and frequent re-adjustment. The hybrid controller and the DMPC have the similar deviations.

Figure 13 shows the return time of these 3 controllers. The hybrid MPC reduces the return frequency; KMPC is

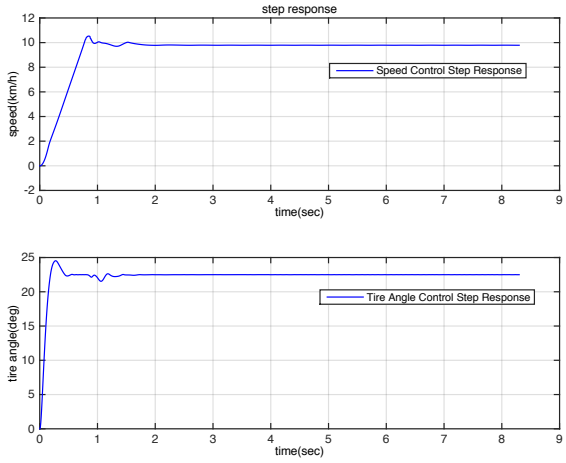
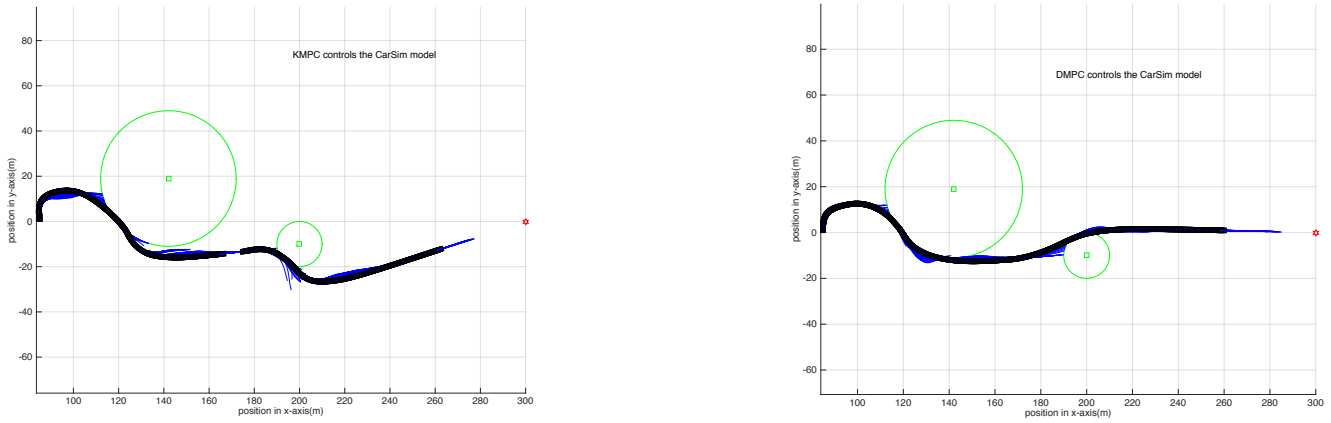


Figure 7: Step response of controllers that convert MPC signal into CarSim-compatible input. The maximum acceleration is $3m/s^2$, which means the maximum increasing speed difference of $10km/h$ may happen within $1sec$. The upper figure reveals the controller's ability to track target speed. Similarly, the response in the bottom figure satisfies the maximum steering speed of 22.5° within $1sec$.



(a) (b)

Figure 8: (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.

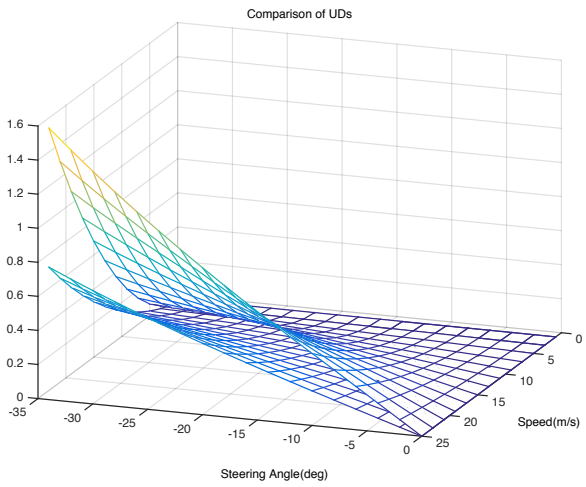


Figure 9: Comparison of UDs. The upper surface belongs to KMPC, and the bottom belongs to DMPC. These two surfaces are close at some region, and diverge when v and δ increase.

still fastest, but with higher error (see Figure 12).

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.

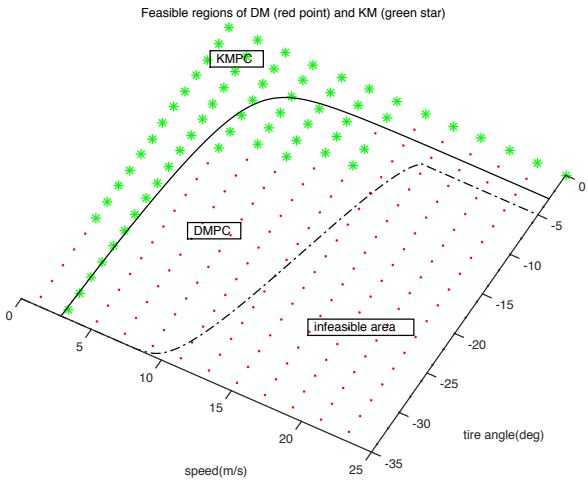
The work is relevant to many kinds of problems in cyber-physical systems where the runtime of a computer-in-the-loop affects the performance of the system: if the models used by the computer can impact this runtime in various ways, then the technique proposed in this paper may be used as a metric to determine whether switching between models poses a significant runtime benefit.

Future work involves investigation of the stability and robustness of the controllers. We are also interested in switching controllers where more than two predictive models may be used. 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.

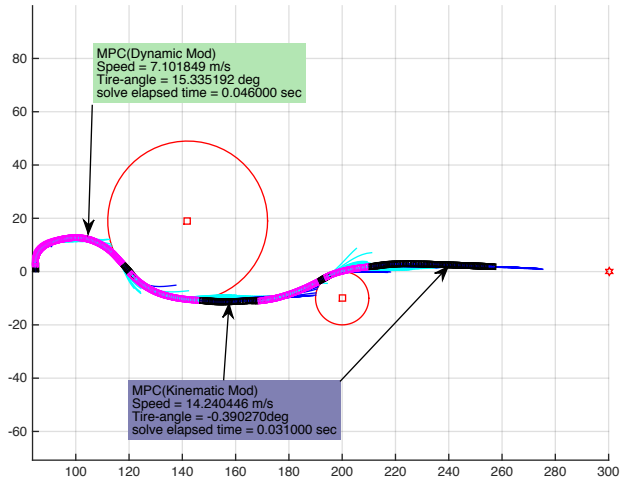
7. REFERENCES

[1] M. Bahadorian, B. Savkovic, R. Eaton, and T. Hesketh. Robust model predictive control for automated trajectory tracking of an unmanned ground vehicle. In *American Control Conference (ACC), 2012*, pages 4251–4256. IEEE, 2012.

[2] A. Bemporad and C. Rocchi. Decentralized linear time-varying model predictive control of a formation of unmanned aerial vehicles. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 7488–7493. IEEE, 2011.



(a)



(b)

Figure 10: (a) The green-dot region and the red-dot region 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 (high speed and high tire angle region) is an infeasible solution. (b) Trajectory generated by the hybrid controller when the CarSim is utilized as 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.

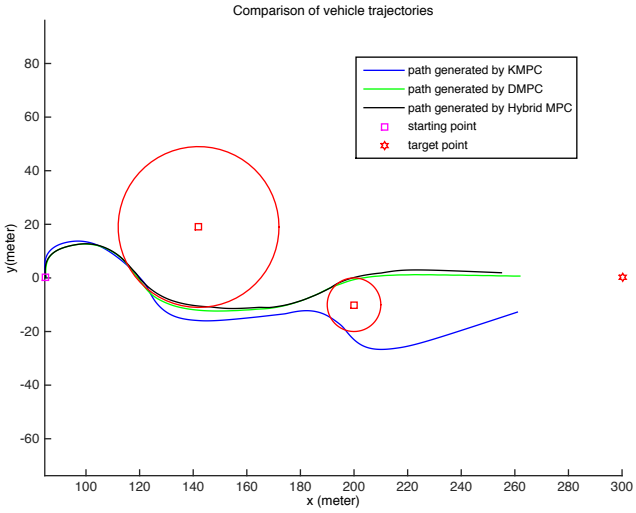


Figure 11: Comparison of the plant trajectories.

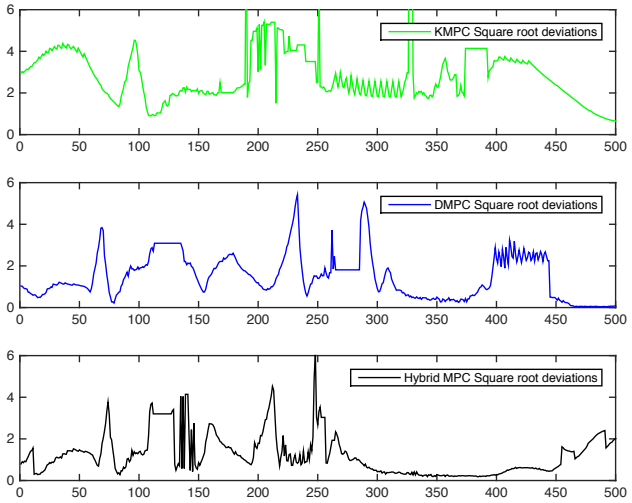


Figure 12: Square root deviations between plant trajectories and predictions. The last several samples are truncated for the reason that predictions always run ahead of the current state.

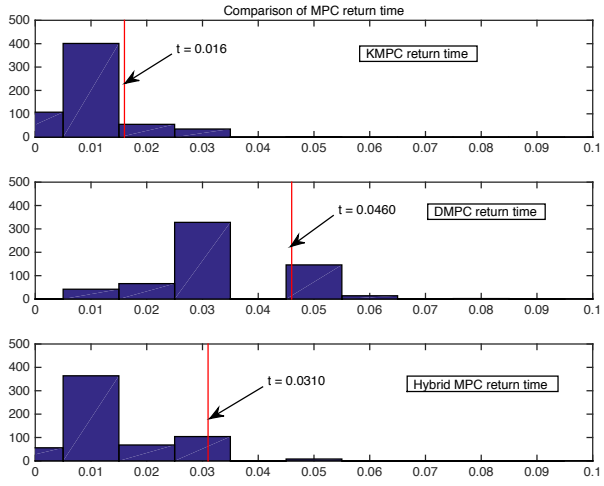


Figure 13: Comparisons of return time. These three figures, from top to bottom, belongs to KMPC, DMPC and the hybrid controller, respectively. 600 **samples of return time are collected for each controller. The abscissa is return time, and the ordinate is the number of the specific return time. The vertical lines on the plot are where** 90% **of return time samples fall bellow.**

[3] M. Cannon and B. Kouvaritakis. Continuous-time predictive control of constrained nonlinear systems. In *Nonlinear Model Predictive Control*, pages 205–215. Springer, 2000.

[4] CarSim. Available from http://www.carsim.com.

[5] M. Egerstedt, X. Hu, and A. Stotsky. Control of a car-like robot using a dynamic model. In *ICRA*, pages 3273–3278. Citeseer, 1998.

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

[7] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat. Linear time-varying model predictive control and its application to active steering systems: Stability analysis and experimental validation. *International journal of robust and nonlinear control*, 18(8):862–875, 2008.

[8] P. Falcone, F. Borrelli, H. Tsengz, J. Asgari, and D. Hrovat. A hierarchical model predictive control framework for autonomous ground vehicles. In *American Control Conference, 2008*, pages 3719–3724. IEEE, 2008.

[9] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. Tsengz. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. In *Decision and Control, 2007 46th IEEE Conference on*, pages 2980–2985. IEEE, 2007.

[10] D. M. Gay and B. Kernighan. Ampl: A modeling language for mathematical programming. *Duxbury Press/Brooks/Cole*, 2, 2002.

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

[12] D. Kim, J. Kang, and K. Yi. Control strategy for high-speed autonomous driving in structured road. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 186–191. IEEE, 2011.

[13] F. Kuhne, W. F. Lages, and J. M. G. da Silva Jr. Model predictive control of a mobile robot using linearization. In *Proceedings of Mechatronics and Robotics*, pages 525–530, 2004.

[14] F. Künhe, J. Gomes, and W. Fetter. Mobile robot trajectory tracking using model predictive control. In *II IEEE latin-american robotics symposium*, 2005.

[15] E. Narby. Modeling and estimation of dynamic tire properties. 2006.

[16] R. S. Parker, E. P. Gatzke, R. Mahadevan, E. S. Meadows, and F. Doyle. Nonlinear model predictive control: issues and applications. *IEE CONTROL ENGINEERING SERIES*, pages 33–58, 2001.

[17] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control engineering practice*, 11(7):733–764, 2003.

[18] J. B. Rawlings. Tutorial: Model predictive control technology. In *American Control Conference, 1999. Proceedings of the 1999*, volume 1, pages 662–676. IEEE, 1999.

[19] A. Richards. Robust model predictive control for time-varying systems. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC'05. 44th IEEE Conference on*, pages 3747–3752. IEEE, 2005.

[20] A. Richards and J. P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *American Control Conference, 2002. Proceedings of the 2002*, volume 3, pages 1936–1941. IEEE, 2002.

[21] R. Schubert, E. Richter, and G. Wanielik. Comparison and evaluation of advanced motion models for vehicle tracking. In *Information Fusion, 2008 11th International Conference on*, pages 1–6. IEEE, 2008.

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

[23] S. Whitsitt and J. Sprinkle. A passenger comfort controller for an autonomous ground vehicle. In *51st IEEE Conference on Decision and Control*, pages 3380–3385, 2012.