PREDICTIVE CONTROL OF AUTONOMOUS GROUND VEHICLES WITH OBSTACLE AVOIDANCE ON SLIPPERY ROADS

Yiqi Gao
Theresa Lin
Francesco Borrelli
Department of Mechanical Engineering
University of California
Berkeley, 94720-1740, USA.
Email:{yiqigao,tlin051,fborrelli}@berkeley.edu

Eric Tseng
Davor Hrovat
Ford Research Laboratories
Dearborn, MI, USA
Email: {htseng,dhrovat}@ford.com

ABSTRACT

Two frameworks based on Model Predictive Control (MPC) for obstacle avoidance with autonomous vehicles are presented. A given trajectory represents the driver intent. An MPC has to safely avoid obstacles on the road while trying to track the desired trajectory by controlling front steering angle and differential braking. We present two different approaches to this problem. The first approach solves a single nonlinear MPC problem. The second approach uses a hierarchical scheme. At the high-level, a trajectory is computed on-line, in a receding horizon fashion, based on a simplified point-mass vehicle model in order to avoid an obstacle. At the low-level an MPC controller computes the vehicle inputs in order to best follow the high level trajectory based on a nonlinear vehicle model. This article presents the design and comparison of both approaches, the method for implementing them, and successful experimental results on icy roads.

INTRODUCTION

In Model Predictive Control (MPC) a model of the plant is used to *predict* the future evolution of the system [1]. Based on this prediction, at each time step t, a performance index is optimized under operating constraints with respect to a sequence of future input moves in order to best follow a given trajectory. The first of such optimal moves is the *control* action applied to the plant at time t. At time $t+1$, a new optimization is solved over a shifted prediction horizon.

The capability of handling constraints in a systematic way makes MPC a very attractive control technique, especially for applications where the process is required to work *in wide operating regions* and close to the boundary of the set of admissible states and inputs. Parallel advances in theory and computing systems have enlarged the range of applications where real-time MPC can be applied [2–6]. Yet, for a wide class of *fast* applications, the computational burden of Nonlinear MPC (NMPC) is still a barrier. As an example, in [7] an NMPC has been implemented on a passenger vehicle for path following via an Active Front Steering (AFS) system at 20 Hz, by using the state of the art optimization solvers and rapid prototyping systems. It is shown that the real-time execution is limited to low vehicle speeds on icy roads, because of its computational complexity. In order to decrease the computational complexity, in [8, 9] a Linear Time Varying MPC approach is presented to tackle the same problem. Experimental results [9–12] demonstrated the capability of the controller to stabilize the vehicle at higher speeds, up to 72kph, in a double lane change maneuver on slippery (snow covered) surface.

In all the aforementioned literature, obstacle avoidance is not explicitly considered in the control design. The work presented in this paper continues to investigate the application of MPC techniques in the situation where a given trajectory fails to avoid one or more pop-up obstacles. The given trajectory represents the initial driver intent and the MPC has to safely avoid the obstacle while trying to track the desired trajectory. Two different approaches to this problem will be presented. Both rely on the NMPC formulation presented in [13]. In [13], the MPC controller uses a six state four wheel vehicle model and a nonlinear semi-empirical Pacejka tire model [14]. The control inputs are the front steering angle and the left and right braking forces. The corresponding torques are distributed to the four wheels through

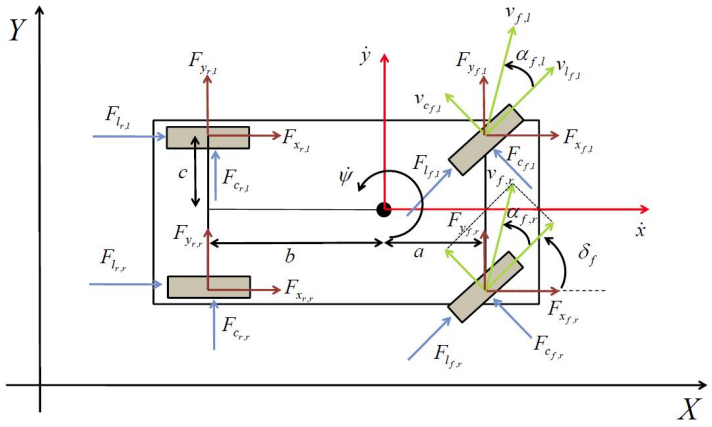


Figure 1. SIMPLIFIED FOUR WHEEL VEHICLE DYNAMICAL MODEL.

a braking logic.

In the first approach, we use the formulation in [13] and modify the cost function by adding a *distance-based* cost term which increases as the vehicle gets closer to the obstacle. In the second approach, the controller design is decomposed into two levels. The high-level uses a simplified point-mass vehicle model and replans the path based on information about the current position of the obstacle. This is done in a receding horizon fashion with an NMPC controller. The trajectory is fed to the low-level controller which is formulated as the one in [13]. We compare performance and computation time of both approaches by using simulation and experimental results.

This paper is structured as follows. First, we present the dynamics of the four wheel model and the point-mass model used in the controller, then outline the hierarchical architecture of the two approaches. We discuss about the MPC formulation of the two approaches and explain how we formulate the cost term associated with obstacle avoidance. Lastly, we close the paper by presenting simulation and experimental results and discuss the comparisons between the two approaches.

VEHICLE MODEL

In this section we introduce the vehicle models used for control design. In the section **Four Wheel Model** we present the six state nonlinear vehicle model used in [13]. The model captures the main vehicle lateral and longitudinal dynamics and uses a nonlinear Pacejka tire model to compute tire forces as a function of wheel longitudinal slips, lateral slips and road friction coefficient. In the section **Point-Mass Model**, a simple point-mass vehicle model is presented. The tire saturation is captured by a constraint on maximum lateral acceleration.

Four Wheel Model

The vehicle dynamics described by the four wheel model used in this paper can be compactly written as

$$\dot{\xi}(t) = f^{4w}(\xi(t), u(t)) \tag{1}$$

where $\xi(t) \in \mathbb{R}^n$ is the state of the system and $u(t) \in \mathbb{R}^{m_r}$ is the input, $n = 6$ is the number of states and $m_r = 3$ is the number of inputs. The six states are lateral and longitudinal velocities in the body frame, the yaw angle, yaw rate, lateral and longitudinal vehicle coordinates in the inertial frame. These are denoted respectively as $\xi = [\dot{y}, \dot{x}, \psi, \dot{\psi}, Y, X]'$. The three inputs are $u = [\delta_f, F_{b_l}, F_{b_r}]'$ where δ_f is the front steering angle and F_{b_l}, F_{b_r} are the left and right braking forces.

The dynamics in (1) can be derived by using the equations of motion about the vehicles Center of Gravity (CoG) and coordinate transformations between the inertial frame and the vehicle body frame:

$$m\ddot{y} = -m\dot{x}\dot{\psi} + F_{y_{f,l}} + F_{y_{f,r}} + F_{y_{r,l}} + F_{y_{r,r}} \tag{2a}$$
$$m\ddot{x} = m\dot{y}\dot{\psi} + F_{x_{f,l}} + F_{x_{f,r}} + F_{x_{r,l}} + F_{x_{r,r}} \tag{2b}$$
$$I\ddot{\psi} = a(F_{y_{f,l}} + F_{y_{f,r}}) - b(F_{y_{r,l}} + F_{y_{r,r}}) + \\ + c(-F_{x_{f,l}} + F_{x_{f,r}} - F_{x_{r,l}} + F_{x_{r,r}}) \tag{2c}$$
$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi \tag{2d}$$
$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \tag{2e}$$

where the constant m is the vehicle's mass, I is the rotational inertial about the yaw axis, a and b are the distances from the CoG to the front and rear axles and c is the distance from the CoG to the left/right side at the wheels.

Figure 1 shows (1) the vehicle states, (2) the lateral (cornering) and longitudinal tire forces, $F_{c_{\star,\bullet}}$ and $F_{l_{\star,\bullet}}$, (3) the components of the tire forces along the lateral and longitudinal vehicle axes, $F_{y_{\star,\bullet}}$ and $F_{x_{\star,\bullet}}$ (4) the slip angle, $\alpha_{\star,\bullet}$ and (5) the steering angles, $\delta_\star$. The first subscript, $\star \in \{f, r\}$, denotes the front and rear, and the second subscript, $\bullet \in \{l, r\}$, denotes the left and right side of the vehicle.

The x and y components of lateral and longitudinal tire forces are

$$F_{y_{\star,\bullet}} = F_{l_{\star,\bullet}}\sin\delta_\star + F_{c_{\star,\bullet}}\cos\delta_\star \tag{3a}$$
$$F_{x_{f,\bullet}} + F_{x_{r,\bullet}} = F_{b_\bullet} \tag{3b}$$

where δ_f, F_{b_l} and F_{b_r} are the inputs to the system and δ_r assumed to be zero. The lateral and longitudinal tire forces, $F_{l_{\star,\bullet}}$ and $F_{c_{\star,\bullet}}$, are given by

$$F_{c_{\star,\bullet}} = f_c(\alpha_{\star,\bullet}, s_{\star,\bullet}, \mu, F_{z_{\star,\bullet}}) \tag{4a}$$
$$F_{l_{\star,\bullet}} = f_l(\alpha_{\star,\bullet}, s_{\star,\bullet}, \mu, F_{z_{\star,\bullet}}) \tag{4b}$$

where α is the slip angle of the tire, s is the slip ratio, μ is the friction coefficient and F_z is the normal force. We use a Pacejka tire model [14] to model F_c and F_l in (4) at the four tires. This complex, semi-empirical model is able to describe the tire behavior over the linear and nonlinear operating ranges of slip ratio, tire slip angle and friction coefficient.

The tire slip angle, $\alpha_{\star,\bullet}$ in (4) is defined as,

$$\alpha_{\star,\bullet} = \arctan \frac{v_{c_{\star,\bullet}}}{v_{l_{\star,\bullet}}} \tag{5}$$

where $v_{c_{\star,\bullet}}$ and $v_{l_{\star,\bullet}}$ are the lateral and longitudinal wheel velocities computed from,

$$v_{c_{\star,\bullet}} = v_{y_{\star,\bullet}} \cos\delta_\star - v_{x_{\star,\bullet}} \sin\delta_\star \tag{6a}$$
$$v_{l_{\star,\bullet}} = v_{y_{\star,\bullet}} \sin\delta_\star + v_{x_{\star,\bullet}} \cos\delta_\star \tag{6b}$$
$$v_{y_{f,\bullet}} = \dot{y} + a\dot{\psi} \qquad v_{x_{\star,l}} = \dot{x} - c\dot{\psi} \tag{6c}$$
$$v_{y_{r,\bullet}} = \dot{y} - b\dot{\psi} \qquad v_{x_{\star,r}} = \dot{x} + c\dot{\psi} \tag{6d}$$

The slip ratio is defined in (7). It depends on the wheel speeds which is not captured by the model presented here. Therefore, it is assumed to be measured at each sampling time and is kept constant until the next available update.

$$s_{\star,\bullet} = \begin{cases} \dfrac{r\omega_{\star,\bullet}}{v_{l_{\star,\bullet}}} - 1 \text{ if } v_{l_{\star,\bullet}} > r\omega_{\star,\bullet}, v \neq 0 \text{ for braking} \\ 1 - \dfrac{v_{l_{\star,\bullet}}}{r\omega_{\star,\bullet}} \text{ if } v_{l_{\star,\bullet}} < r\omega_{\star,\bullet}, \omega \neq 0 \text{ for driving} \end{cases} \tag{7}$$

In this paper, the friction coefficient μ is assumed to be a known constant. The normal force on the wheel, $F_{z_{\star,\bullet}}$ in Equation (4) is defined as,

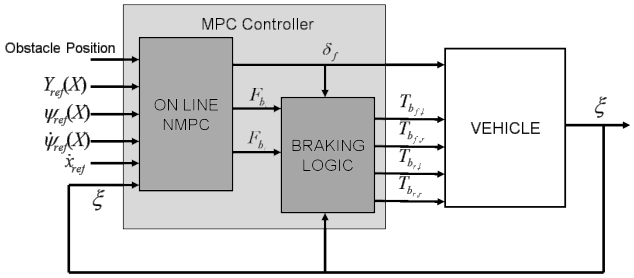
$$F_{z_{f,\bullet}} = \frac{bmg}{2(a+b)} \qquad F_{z_{r,\bullet}} = \frac{amg}{2(a+b)} \tag{8}$$

where g is the gravitational acceleration.

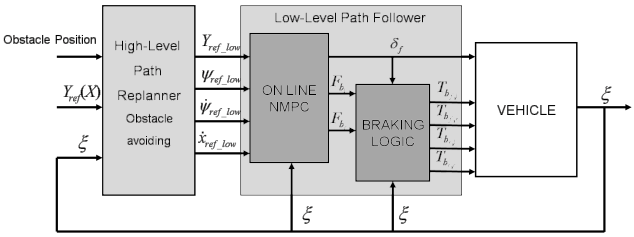
Point-mass model

The four wheel model is further simplified to reduce the computational burden of the MPC. We assume that the longitudinal velocity of the vehicle is constant and the model tire saturation with a bound μg on the lateral acceleration, a_y.

$$\ddot{y} = a_y \tag{9a}$$
$$\ddot{x} = 0 \tag{9b}$$
$$\ddot{\psi} = \frac{a_y}{\dot{x}} \tag{9c}$$
$$\dot{Y} = \dot{x}\sin\psi + \dot{y}\cos\psi \tag{9d}$$



(a) ARCHITECTURE OF ONE-LEVEL MPC. THE FOUR WHEEL VEHICLE IS USED.



(b) ARCHITECTURE OF TWO-LEVEL MPC. THE POINT-MASS VEHICLE MODEL IS USED FOR HIGH-LEVEL PATH REPLANNER. WHILE THE FOUR WHEEL VEHICLE MODEL IS USED FOR THE LOW-LEVEL PATH FOLLOWER.

Figure 2. ARCHITECTURE OF TWO DIFFERENT OBSTACLE AVOIDING MPC DESIGN

$$\dot{X} = \dot{x}\cos\psi - \dot{y}\sin\psi \tag{9e}$$
$$|a_y| < \mu g \tag{9f}$$

where ψ is defined as the direction of vehicle speed. Thus the dynamics of the point-mass model can be written as,

$$\dot{\xi}(t) = f^{pm}(\xi(t), u(t)) \tag{10a}$$
$$|u(t)| < \mu g \tag{10b}$$

where $\xi = [\dot{y},\ \dot{x},\ \psi,\ Y,\ X]'$ and $u = a_y$.

MPC CONTROL ARCHITECTURES WITH OBSTACLE AVOIDANCE

Two control architectures are considered. The first one is shown in Figure 2(a), and is referred to as *one-level MPC*. It uses the four wheel model presented in Section **Four Wheel Model**. The desired trajectory $[\dot{x}_{ref}, \psi_{ref}(X), \dot{\psi}_{ref}(X), Y_{ref}(X)]$ along with the obstacle position are fed to the control algorithm. The predictive control scheme computes the optimal inputs to safely avoid the obstacle while trying to track the desired trajectory. Note that the forces F_{b_l} and F_{b_r} computed by the MPC are distributed to the braking torques, $T_{b_{\star,\bullet}}$, on the four wheels by using a braking logic. Details about the braking logic can be found in [13].

The second approach is shown in Figure 2(b), and is referred to as *two-level MPC*. It is based on a hierarchical decomposition to the control problem. The desired trajectory $[\dot{x}_{ref}, Y_{ref}(X)]$ and obstacle position are fed to the high-level path replanner. The path replanner replans a path which avoids the obstacle and tracks the reference, using the point-mass model presented in Section **Point-Mass Model**. The replanned path $[\dot{x}_{ref}, \psi_{repl}(X), \dot{\psi}_{repl}(X), Y_{repl}(X)]$ is then passed to a low-level path follower. The path follower computes the optimal input to follow the new trajectory, using the four wheel vehicle model presented in section **Four Wheel Model**. The same braking logic as in the one-level MPC is used here. Both controllers are NMPCs and are described in detail in the next section.

MPC CONTROLLER FORMULATION

The MPC controller can be formulated as a general optimization problem

$$\begin{aligned}
\min_{U_t} \quad & J_N(\bar{\xi}_t, U_t, \Delta U_t) && \text{(11a)}\\
\text{subj. to} \quad & \xi_{k+1,t} = f(\xi_{k,t}, u_{k,t}) \quad k = t, ..., t+H_p-1 && \text{(11b)}\\
& \Delta u_{k+1,t} = u_{k+1,t} - u_{k,t} \quad k = t, ..., t+H_p-1 && \text{(11c)}\\
& u_{k,t} \in \mathcal{U} \quad k = t, ..., t+H_p-1 && \text{(11d)}\\
& \Delta u_{k,t} \in \Delta\mathcal{U} \quad k = t+1, ..., t+H_p-1 && \text{(11e)}\\
& \xi_{t,t} = \xi(t) && \text{(11f)}
\end{aligned}$$

where $\bar{\xi}_t = [\xi_{t,t}\ ,\ \xi_{t+1,t}\ ,...,\ \xi_{t+H_p-1,t}]$ is the sequence of states $\bar{\xi}_t \in \mathbb{R}^{nH_p}$ over the prediction horizon H_p predicted at time t, and updated according to the discretized dynamics of the vehicle model (11b), and $u_{k,t}$ and $\Delta u_{k,t} \in \mathbb{R}^{m_r}$ is the k^{th} vector of the input sequence $U_t \in \mathbb{R}^{m_r H_p}$ and $\Delta U_t \in \mathbb{R}^{m_r(H_p-1)}$ respectively,

$$U_t = [u'_{t,t}\ ,\ u'_{t+1,t}\ ,...,\ u'_{t+H_u-1,t}\ ,\ u'_{t+H_u,t}\ ,...,\ u'_{t+H_p-1,t}]' \quad \text{(12a)}$$
$$\Delta U_t = [\Delta u'_{t+1,t}, ..., \Delta u'_{t+H_u-1,t}, \Delta u'_{t+H_u,t}, ..., \Delta u'_{t+H_p-1,t}]' \quad \text{(12b)}$$

We reduce the computational complexity of the MPC problem by holding the last $H_p - H_u$ input vectors in U_t constant and equal to the vector $u_{t+H_u-1,t}$. With this assumption, only the first H_u input vectors constitutes the optimization variables. We refer to H_p as the *prediction horizon* and H_u as the *control horizon*.

At each time step t, the performance index $J_N(\bar{\xi}_t, U_t, \Delta U_t)$ is optimized under the constraints (11c)-(11e) starting from the state $\xi_{t,t} = \xi(t)$ to obtain an optimal control sequence, $U_t^* = [u_{t,t}^{*\prime}, ..., u_{t+H_p-1,t}^{*\prime}]'$. The first of such optimal moves $u_{t,t}^*$ is the *control* action applied to the vehicle at time t. At time $t+1$, a new optimization is solved over a shifted prediction horizon starting from the new measured state $\xi_{t+1,t+1} = \xi(t+1)$. The time interval between time step $t+1$ and and time step t is the sampling time T_s.

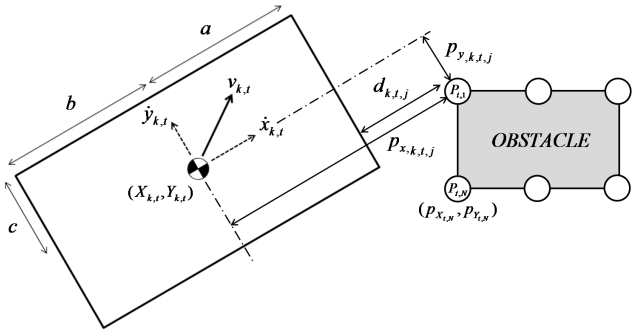


Figure 3. GRAPHICAL REPRESENTATION OF $d_{k,t,j}$, $p_{x_{t,j}}$ AND $p_{y_{t,j}}$.

Obstacle Avoidance Cost

The cost for obstacle avoidance is based on the distance between the front of the vehicle and the obstacle [15]. Assuming information on the position of the obstacle is given at time t as a collection of N discretized points, the position of the j^{th} point is denoted as $P_{t,j} = (p_{X_{t,j}}, p_{Y_{t,j}})$, $j = 1, 2, ..., N$. If these points are given in the inertial frame, they are transformed to the vehicle body frame by,

$$p_{x_{k,t,j}} = (p_{Y_{t,j}} - Y_{k,t})\sin\psi_{k,t} + (p_{X_{t,j}} - X_{k,t})\cos\psi_{k,t} \quad \text{(13a)}$$
$$p_{y_{k,t,j}} = (p_{Y_{t,j}} - Y_{k,t})\cos\psi_{k,t} - (p_{X_{t,j}} - X_{k,t})\sin\psi_{k,t} \quad \text{(13b)}$$

The cost at time k associated to the distance predicted at time t between the vehicle and an obstacle is denoted as $J_{obs_{k,t}}$:

$$J_{obs_{k,t}} = \frac{K_{obs} v_{k,t}}{d_{min_{k,t}} + \varepsilon} \quad \text{(14)}$$

where $v_{k,t}^2 = \dot{x}_{k,t}^2 + \dot{y}_{k,t}^2$ is the speed of the vehicle at time k predicted at time t, ε is a small number, K_{obs} is the collision weight and $d_{min_{k,t}} = \min_j d_{k,t,j}$ is the minimum distance to all N obstacle points, with $d_{k,t,j}$ defined as,

$$d_{k,t,j} = \begin{cases} p_{x_{k,t,j}} - a & \text{if } p_{y_{k,t,j}} \in [-c, c] \text{ and } p_{x_{k,t,j}} > a \\ 0 & \text{if } p_{y_{k,t,j}} \in [-c, c] \text{ and } p_{x_{k,t,j}} \in [-b, a] \\ M & \text{otherwise} \end{cases} \quad \forall\ j = 1, 2, ..., N \quad \text{(15)}$$

where a, b and c are the vehicle dimensions, $d_{k,t,j} = 0$ indicates a collision occurrence, and M is a constant big enough to disregard obstacles that do not lie within the vehicle's line of sight.

Note, the obstacle is discretized at a resolution which ensures that the vehicle cannot drive right through the obstacle without containing at least one point within its frame.

One-Level MPC

The one-level MPC uses the four wheel vehicle mode, i.e., equation (11b) is a discrete time version of the model $\dot{\xi}(t) = f^{4w}(\xi(t), u(t))$.

The cost function considers both the deviations of the tracking state $\eta_{k,t} = [\dot{x}_{k,t}, \psi_{k,t}, \dot{\psi}_{k,t}, Y_{k,t}]'$ from the state reference, $\eta_{ref_{k,t}} = [\dot{x}_{ref_{k,t}}, \psi_{ref_{k,t}}, \dot{\psi}_{ref_{k,t}}, Y_{ref_{k,t}}]'$, and the cost term $J_{obs_{k,t}}$ associated with obstacle avoidance:

$$J_N(\bar{\xi}_t, U_t, \Delta U_t) = \sum_{k=t}^{t+H_{tr}-1} \|\eta_{k,t} - \eta_{ref_{k,t}}\|_Q^2 + \|u_{k,t}\|_R^2 + \|\Delta u_{k,t}\|_S^2 + \\ + \sum_{k=t}^{t+H_p-1} J_{obs_{k,t}} \tag{16}$$

where Q, R and S are weighting matrices of appropriate dimensions. Notice that predicted state trajectories are penalized over the interval $k \in [t,\ t+H_{tr}-1]$, where H_{tr} is the *tracking horizon*, $H_{tr} < H_p$. The obstacle avoidance cost term is penalized over the entire prediction horizon. In addition we use two different sampling times: T_{s1} for $k \in [t,\ t+H_{tr}-1]$ and T_{s2} for $k \in [t+H_{tr},\ t+H_p-1]$ with $T_{s1} < T_{s2}$. Both constructions are fundamental for obtaining a scheme that is real-time implementable on the vehicle hardware.

The inputs vector $u_{k,t} = [\delta_f, F_{b_l}, F_{b_r}]'$ consists of the steering angle δ_f, left braking force F_{r_l} and right braking force F_{b_r}. The braking logic in [13] is used to distribute the corresponding torques at the four wheels.

Two-Level MPC

Two MPC problems are formulated for the high-level path replanning and for the low-level path following. The low-level MPC is similar to the one-level MPC: the vehicle model, the inputs and their constraints are the same. The cost function neglects the obstacle avoidance term and the tracking reference is the replanned path, $\eta_{repl_{k,t}} = [\dot{x}_{repl_{k,t}}, \psi_{repl_{k,t}}, \dot{\psi}_{repl_{k,t}}, Y_{repl_{k,t}}]'$:

$$J_N(\bar{\xi}_t, U_t, \Delta U_t) = \sum_{k=t}^{t+H_{p,ll}-1} \|\eta_{k,t} - \eta_{repl_{k,t}}\|_Q^2 + \|u_{k,t}\|_R^2 + \|\Delta u_{k,t}\|_S^2 \tag{17}$$

The sampling time and prediction horizon is denoted as $T_{s,ll}$ and $H_{p,ll}$ respectively.

The high-level of MPC, uses the point-mass model (10), equation (11b) is a discrete time version of the model $\dot{\xi}(t) = f^{pm}(\xi(t), u(t))$.

The cost function weights the deviations from the lateral reference $Y_{ref_{k,t}}$, the lateral acceleration input $a_{y_{k,t}}$ and the obstacle avoidance term:

$$J_N(\bar{\xi}_t, U_t) = \sum_{k=t}^{t+H_{p,hl}-1} \|Y_{k,t} - Y_{ref_{k,t}}\|_Q^2 + \|a_{y_{k,t}}\|_R^2 + J_{obs_{k,t}} \tag{18}$$

The sampling time and prediction horizon is denoted as $T_{s,hl}$ and $H_{p,hl}$ respectively.

SIMULATION AND EXPERIMENTAL RESULTS

To test the performance of the two approaches, a series of double lane change maneuvers at different entry speeds have been simulated and experimentally tested. The goal is to follow this maneuver as closely as possible while avoiding obstacles.

The desired path is described in terms of lateral position Y_{ref} and yaw angle ψ_{ref} as a function of the longitudinal position X. Details of this trajectory generation can be found in [9].

Simulation Setup Description

We use Matlab® to simulate the close loop systems. The MPC optimization problem has been implemented as a C-coded s-Function. The commercial NPSOL software package [16] is used for solving the nonlinear programming problem (11). The first element of the optimized control sequence is passed to another external block which uses a four wheel model and Pacejka tire model to simulate the dynamics of the vehicle, and feeds the current state of the vehicle back to the controller.We simulate results using this procedure to compare the computation loads for the two different approaches.

Experimental Setup Description

The two approaches presented before have been tested through simulations and experiments on slippery surfaces. The experiments have been performed at a test center equipped with icy and snowy handling tracks. The MPC controllers have been tested on a passenger car, with a mass of 2050 Kg and an inertia of 3344 $\mathrm{Kg/m^2}$. The controllers were run in a dSPACE Autobox system, equipped with a DS1005 processor board and a DS2210 I/O board.

We used an Oxford Technical Solution (OTS) RT3002 sensing system to measure the position and the orientation of the vehicle in the inertial frame and the vehicle velocities in the vehicle body frame. The OTS RT3002, is housed in a small package that contains a differential GPS receiver, Inertial Measurement Unit (IMU), and a DSP. It is equipped with a single antenna to receive GPS information. The IMU includes three accelerometers and three angular rate sensors. The DSP receives both the measurements from the IMU and the GPS, utilizes a Kalman filter for sensor fusion, and calculate the position, orientation and other states of the vehicle such as longitudinal and lateral velocities.

The car was equipped with an Active Front Steering (AFS) and Differential Braking system which utilizes an electric drive motor to change the relation between the hand steering wheel and road wheel angles. This is done independently from the hand wheel position, thus the front road wheel angle is obtained by summing the driver hand wheel position and the actuator angular movement. Both the hand wheel position and the angular relation between hand and road wheels are measured. The sensor, the dSPACE Autobox and the actuators communicate through a CAN bus.

The test is initiated by the driver with a button. When the button is pushed, the inertial frame in Figure 1 is initialized as

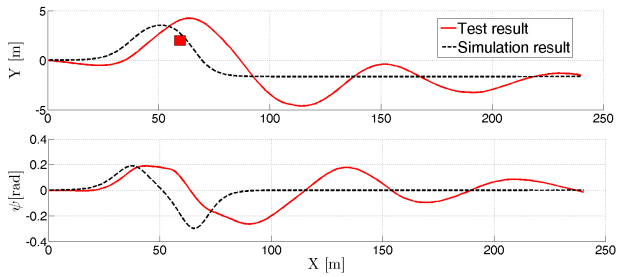


Figure 4. TEST RESULTS OF THE ONE-LEVEL MPC AT VEHICLE SPEED 40KPH. THE RED BOX REPRESENTS THE OBSTACLE.

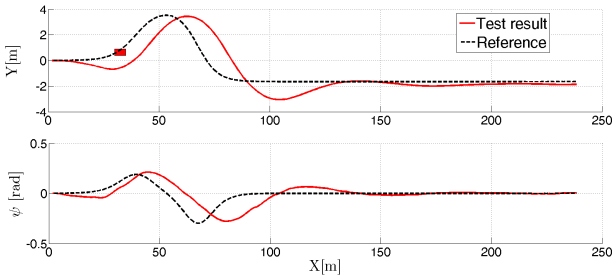


Figure 5. TEST RESULTS OF THE TWO-LEVEL MPC AT VEHICLE SPEED 45KPH. THE RED BOX REPRESENTS THE OBSTACLE.

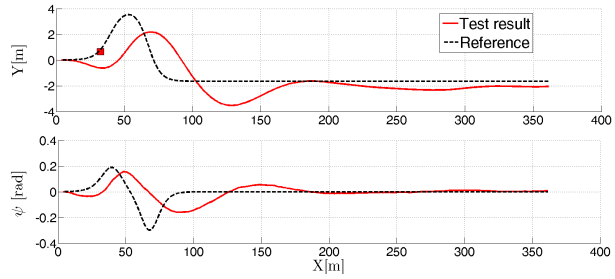


Figure 6. TEST RESULTS OF THE TWO-LEVEL MPC AT VEHICLE SPEED 55KPH. THE RED BOX REPRESENTS THE OBSTACLE.

follows: the origin is the current vehicle position, the axes X and Y are directed as the current longitudinal and lateral vehicle axes, respectively. Such inertial frame becomes also the desired path coordinate system. Once the initialization procedure is concluded, the vehicle executes the double lane change maneuver.

Note that, noise may affect the yaw angle measurement due to the single antenna sensor setup. Compared to a dual antenna setup, a single antenna system has to learn the vehicle orientation and/or coordinate during vehicle motion. When the vehicle stands still the yaw angle is computed by integrating the yaw rate measurement from the IMU. This might cause the presence of a small offset in the orientation measurement, while traveling at low speed or being still.

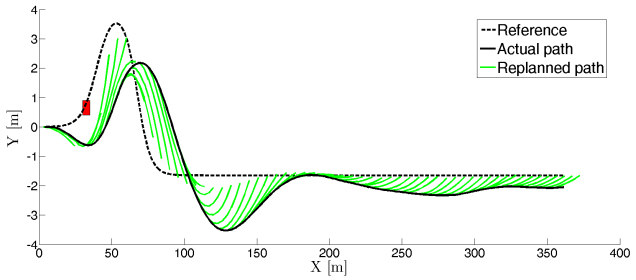


Figure 7. REPLANNED PATHS FROM THE HIGH-LEVEL PATH REPLANNER IN TWO-LEVEL MPC. VEHICLE SPEED IS 55KPH.

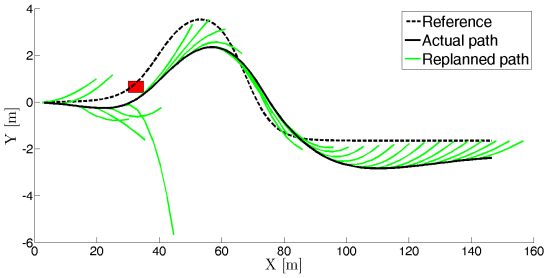


Figure 8. REPLANNED PATHS FROM HIGH-LEVEL WHEN THE COMMUNICATION FROM HIGH-LEVEL TO LOW-LEVEL IS CUT OFF. VEHICLE SPEED IS 55KPH.

Experimental Results

In the next section, we will compare the two approaches presented before, the one-level and the two-level MPC controller. Shown in Figure 4, 5 and 6 are the experimental results obtained with the vehicle travelling on icy roads ($\mu \simeq 0.3$) at different entry speeds. The controllers parameters are reported next.

Tuning 1A *One-level MPC:* Nonlinear MPC (11) with model (1) and cost (16) and with the following parameters

- $T_{s1} = 0.1\text{s}$, $T_{s2} = 0.3\text{s}$, $H_p = 10$, $H_u = 2$, $H_{tr} = 5$
- $\delta_f \in [-10^\circ, 10^\circ]$, $\Delta\delta_f \in [-17^\circ, 17^\circ] \times T_{s1}$
- $F_{b_\bullet} \in [-1500, 0]$, $\Delta F_{b_\bullet} \in [-1000, 1000] \times T_{s1}$
- $Q = \text{diag}(10^{-2}, 1, 1, 30)$, $\quad R = \text{diag}(10, 1, 1)$
  $S = \text{diag}(10^{-2}, 1, 1)$, $\quad K_{obs} = 1$

Tuning 2HL *Hierarchical MPC High-level:* Nonlinear MPC (11) with model (10) and cost (18) and with the following parameters.

- $T_{s,hl} = 0.1\text{s}$, $H_{p,hl} = 15$; $H_u = 2$
- $a_y \in [-\mu g, \mu g]$
- $Q = 10$, $\quad R = 12$, $\quad K_{obs} = 1$

Tuning 2LL *Hierarchical MPC Low-level:* Nonlinear MPC (11) with model (1) and cost (17) and with the following parameters

- $T_{s,ll} = 0.05\text{s}$, $H_{p,ll} = 15$, $H_u = 2$

- $\delta_f \in [-10^\circ, 10^\circ]$, $\Delta\delta_f \in [-17^\circ, 17^\circ] \times T_{s,ll}$
- $F_{b_\bullet} \in [-1500, 0]$, $\Delta F_{b_\bullet} \in [-1000, 1000] \times T_{s,ll}$
- $Q = \mathrm{diag}(0, 10, 1, 30)$, $R = \mathrm{diag}(1, 10, 10)$
  $S = \mathrm{diag}(1, 4, 4)$

The road friction coefficient μ in (4) and (9) is set to be a constant 0.3 even if the road conditions were changing during each experiment depending on the weather conditions (ice, snow, fresh snow on top of packed snow). Also, during the experiments, the maximum iteration number in NPSOL has been limited in order to guarantee real-time computation.

For the one-level MPC, successful tests were obtained only at 40kph (Figure 4). The two-level MPC was able to run at higher speeds of 45 and 55kph (Figure 5 and 6). In Figure 4 to 6, the dotted black line represents the double-lane reference trajectory for Y_{ref} and ψ_{ref} and the solid red line represents the test results. There is also a red rectangular box drawn to illustrate the position of the obstacle. In Figure 7 the replanned trajectories at the high-level are shown along with the actual vehicle trajectory.

In order to better understand the role of the path replanner, in Figure 8 we present a test where the communication from the high-level to the low-level controller is cut off. Therefore the low-level controller tries to follow the desired path regardless of the obstacle position and the replanned trajectory. We observe that before passing the obstacle, the high-level replans paths that avoid the obstacle by turning left at beginning and then by turning right when driver insists staying to the right. After passing the obstacle, the path replanner plans paths that converge to the reference.

Computation Time

The average and maximum computation times for the one-level and the two-level MPC are summarized in Table 1. In order to test the one-level MPC for longer horizons (not implementable in real-time on the experimental platform), the results have been obtained in simulation with a 3.0 GHz Intel® Core(TM)2Duo desktop running Matlab 7.5, with the vehicle speed set to 40kph, and obstacle position as shown in Figure 6.

The computation load is measured in terms of Floating Point Operations (FLOP), N_{comp}, NPSOL takes at each iteration. Using the conversion, $T_{comp} = N_{comp}/3 \times 10^6$, the results summarized in Table 1 are the actual running times on the given processor.

For the two-level MPC, the computation time in Table 1 is obtained with **Tuning 2HL** and **Tuning 2LL**. For the one-level MPC, **Tuning 1A** and **Tuning 1B** is used, where Tuning 1B is similar to Tuning 1A with these parameters changed: $T_{s1} = 0.05s, T_{s2} = 0.1s, H_p = 23, H_u = 2, H_{tr} = 16$.

Notice that, with the one-level MPC set at Tuning 1B, the prediction horizon is now comparable to the two-level MPC. The tracking horizon is, $H_{p,ll}T_{s,ll} \simeq H_{tr}T_{s1} = 0.8s$ and the prediction horizon with $J_{obs_{k,t}}$ is, $H_{p,hl}T_{s,hl} = H_{tr}T_{s1} + (H_p - H_{tr})T_{s2} = 1.5s$.

With Tuning 1B, the one-level MPC is not able to run in real-time on the experimental platform. As shown in Table 1, the computation time of two-level MPC is 39% less than that of one-level MPC at (Tuning 1A). The one-level MPC suffers from the heavy computational burden and is only able to run in real time with compromised horizon and sampling time, which limits the performance (Figure 4). The two-level MPC, on the other hand, reduces the computation complexity by a hierarchical decomposition to the problem and is able to obtain good performance (Figure 5 and 6).

Table 1. COMPUTATION TIME OF ONE-LEVEL AND TWO-LEVEL MPC AT VEHICLE SPEED OF 40 KPH.

	One-lvl MPC Comp.time [ms]		Two-lvl MPC Comp.time [ms]	
	Tuning 1A	Tuning 1B	High-lvl	Low-lvl
Mean	10 ± 1	23 ± 3	0.5 ± 0.2	3.6 ± 1.4
Max	15	36	1.8	8.2

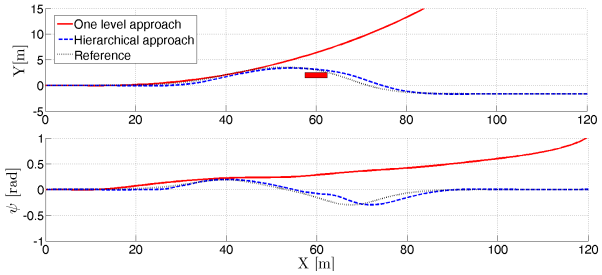


Figure 9. SIMULATION RESULTS AT SPEED OF 70KPH SHOWS THE ONE-LEVEL MPC BECOMES UNSTABLE EVEN IF GIVEN ENOUGH COMPUTATION TIME. WHILE THE TWO-LEVEL MPC IS ABLE TO STABILIZE THE VEHICLE IN THE SAME SITUATION

Stability and Obstacle Avoidance

Consider the one-level MPC. From Figure 9, we observe that once the vehicle deviates too far from the reference, the controller can no longer pull the system back to the reference trajectory and the system becomes uncontrollable. This means that the vehicle state is outside the region of attraction of the equilibrium trajectory associated to the desired reference. When the vehicle is travelling at high speeds, this region of attraction becomes smaller, and the reference trajectory becomes harder to follow. In addition, when obstacle avoidance is combined with trajectory following, the controller will force the vehicle to deviate from the reference in order to avoid the obstacle. This often causes the system to become unstable for the one-level controller running at high speeds. We remark that this type of behavior is induced by tire saturation and cannot be observed with a simple point-mass model.

However, for the two-level controller, this issue is signifi-

cantly resolved, since the path replanner always replans a path starting from the current state of the vehicle, and therefore, ensures that the low-level reference is close to current state. This explains why the performance of the two-level approach is better than the one-level approach.

Figure 9 shows the comparison between simulations for the two approaches at 70kph. Both controllers tried to avoid the obstacle by steering to the right, forcing the vehicle away from the reference trajectory. The one-level MPC becomes unstable while the two-level MPC avoids the obstacle and is still able to return to the reference trajectory. We can improve the performance of the one-level MPC by extending the prediction horizon and control horizon, however, this is done at the expenses of computation time, and generates runtime error on the described experimental platform.

If the one-level MPC is properly designed (for example, by using an invariant set as a terminal constraint [1]), then persistent feasibility is guaranteed for all initial feasible states (note that the obstacle position becomes a state of the system). For the current two-level MPC design, even if the high-level path replanner computes a path that avoids the obstacle, collision may still occur due to errors in the path following at the low-level. By adding a safety distance to the obstacle (i.e. enlarge the obstacle) and a constraint to the maximum tracking error in the low-level controller, obstacle avoidance can be guaranteed (for all initial conditions for which the low-level controller is persistently feasible).

CONCLUSIONS

We have presented two MPC approaches for trajectory following for autonomous ground vehicles with obstacle avoidance. The one-level approach combines both requirements into one large finite horizon optimization problem, while the two-level approach breaks this problem into a high-level path replanning and low-level trajectory following. We have shown experimental results up to 55kph on icy surfaces. The computation time of the one-level approach increases at higher speeds and therefore real-time implementation presents an issue. The two-level approach shows promise in this aspect with its much lower computational burden.

REFERENCES

[1] Mayne, D., Rawlings, J., Rao, C., and Scokaert, P., 2000. "Constrained model predictive control: Stability and optimality". *Automatica,* **36**(6), June, pp. 789–814.

[2] Borrelli, F., Bemporad, A., Fodor, M., and Hrovat, D., 2006. "An MPC/hybrid system approach to traction control". *IEEE Trans. Control Systems Technology,* **14**(3), May, pp. 541–552.

[3] Borrelli, F., Keviczky, T., Balas, G. J., Stewart, G., Fregene, K., and Godbole, D., 2005. "Hybrid decentralized control of large scale systems". In Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, Springer Verlag.

[4] Keviczky, T., and Balas, G. J., 2005. "Flight test of a receding horizon controller for autonomous uav guidance". In Proc. American Contr. Conf.

[5] Ferrau, H. J., Bock, H. G., and Diehl, M., 2006. "An online active set strategy for fast parametric quadtratic programming in mpc applications". *IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, plenary talk*.

[6] Zavala, V. M., Laird, C. D., and Biegler, L. T., 2006. "Fast solvers and rigorous models: Can both be accomodated in nmpc?". *IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, plenary talk*.

[7] Borrelli, F., Falcone, P., Keviczky, T., Asgari, J., and Hrovat, D., 2005. "MPC-based approach to active steering for autonomous vehicle systems". *Int. J. Vehicle Autonomous Systems,* **3**(2/3/4), pp. 265–291.

[8] Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., and Hrovat, D., 2006. "A real-time model predictive control approach for autonomous active steering". In Nonlinear Model Predictive Control for Fast Systems, Grenoble, France.

[9] Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., and Hrovat, D., 2007. "Predictive active steering control for autonomous vehicle systems". *IEEE Trans. on Control System Technology,* **15**(3).

[10] Falcone, P., Tufo, F. B. M., Tseng, H. E., and Asgari, J., 2007. "Predictive autonomous vehicles: A linear time varying model predictive control approach". *46th Conference on Decision and Control*.

[11] Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., and Hrovat, D., 2007. "A model predictive control approach for combined braking and steering in autonomous vehicles". *Submitted to the 15th Mediterranean Conference on Control and Automation (*available at http://www.grace.ing.unisannio.it/home/pfalcone*)*.

[12] Falcone, P., Borrelli, F., Tseng, H. E., Asgari, J., and Hrovat, D., 2007. "Integrated braking and steering model predictive control approach in autonomous vehicles". *Fift IFAC Symposium on Advances of Automotive Control*.

[13] Falcone, P., Borrelli, F., Asgari, J., Tseng, H. E., and Hrovat, D., 2008. "Low complexity mpc schemes for integrated vehicle dynamics control problems". *9th International Symposium on Advanced Vehicle Control*.

[14] Bakker, E., Nyborg, L., and Pacejka, H. B., 1987. "Tyre modeling for use in vehicle dynamics studies". *SAE paper # 870421*.

[15] Yoon, Y., Shin, J., Kim, H. J., Park, Y., and Sastry, S., 2009. "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles". *Control Engineering Practice,* **17**(7), pp. 741–750.

[16] Gill, P., Murray, W., Saunders, M., and Wright, M., 1998. NPSOL – Nonlinear Programming Software. Stanford Business Software, Inc., Mountain View, CA.