MPC Based Collaborative Adaptive Cruise Control with Rear End Collision Avoidance

Presented at IEEE 2014 Intelligent Vehicles Symposium, Michigan, June, 2014

Feyyaz Emre Sancar, Baris Fidan, Jan P. Huissoon, Steven L. Waslander
Department of Mechanical and Mechatronics Engineering
University of Waterloo
fesancar@uwaterloo.ca, fidan@uwaterloo.ca, jph@uwaterloo.ca, stevenw@uwaterloo.ca

Abstract— This paper presents a model predictive control (MPC) based approach to improve a recently developed class of collaborative adaptive cruise control (CACC) schemes. The PID structure used previously is replaced with MPC, which is able to accommodate actuator limits and parameter estimation. In addition to the regular CACC functionalities, rear end collision control is also incorporated. This approach is able to avoid any rear end collisions with the following car, as long as it can still maintain the safe distance with the preceding vehicle. Simulation results are presented which demonstrate the validity of the approach.

Index Terms— MPC; CACC; Rear-End Collision Check; Optimal Control

INTRODUCTION

With the continually increasing number of cars, traffic congestion is becoming a bigger issue. According to a recent report, the daily average traffic jam length in the Netherlands can go longer the length of the country itself [1].

There are essentially two types of traffic jam: those caused by insufficiency of the road itself to physically accommodate all the cars, and ghost traffic jams. A ghost traffic jam can develop when human drivers follow each other closely in highway rush-hour traffic and a small braking action in one of the cars is magnified as following drivers react in succession, which can even cause the traffic to come to a complete stop [1]. This phenomenon is generally referred to as string instability. When a system is string stable, small disturbances are attenuated as they propagate upstream, which in turn provides smooth traffic flow with less congestion [2]. In designing controllers for platoons, string stability is one of the key points that should be taken in to consideration. Also it is essential for these controllers to operate correctly and be dependable. As will be seen in the following section, the models used are capable of operating smoothly even in the event of a small number of broken wireless communication channels.

Adaptive cruise control (ACC) is available as an option on many vehicles today. This was initially intended to improve safety and comfort, and was not designed to increase traffic efficiency [3]. Interest in this field started in 80s with the California PATH research programs, which included the development of vehicle to highway communication [4]. To increase traffic efficiency, the inter-vehicle distance should be small, while the platoon should still be string stable with that inter-vehicle distance. This is difficult to achieve in typical ACC because the platoon loses its string stability, which cooperative adaptive cruise control (CACC) could address.

The main idea behind cooperative adaptive cruise control is the introduction of wireless communication between vehicles to allow small inter-vehicular distances while maintaining string stability [3]. However, a generic CACC controller does not yet exist, and most controllers available are optimized for specific scenarios.

PROBLEM DEFINITION

In the 2011 GCDC challenge most CACC used a linear feedback controller with a feed-forward component, as shown in Figure 1.

![Figure 1 – Feedback control model](image)

PD controller gains are usually chosen experimentally and the feed-forward controller is designed to ensure the string stability of the system. This model currently works well if the vehicle is on a straight level path without any changes to surroundings.
This approach assumes a simplified car model, where the parameters have negligible effects on the feedback controller. However, in practice the car model changes based on factors such as the number of passengers in the vehicle, tire traction, road conditions and slope. The feedback controller is tuned for the ideal environment, which will often lead to poor performance under real life conditions. It also does not account for the limitations of the vehicle, such as maximum velocity, acceleration, and braking.

One objective of this work is therefore to use a controller that can be fully adaptive to changes in the car model – this is a prerequisite if the vehicles are to be used in real life scenarios, as there exists no method to finely controllers to work in every specific condition a vehicle might encounter.

Another concern with the existing class of PD control methods lies in their inability to incorporate limitations coming from the lateral controller and the slip circle. These fundamental vehicle limits must be addressed for controller deployment, as they are inherent to every vehicle on the road.

Combined with these two additional objectives, the main goal of this controller design remains to keep a safe distance with the preceding vehicle. Finally, the formulation of the CACC control using MPC easily admits a rear-end collision check, which is added to increase vehicle safety in mixed equipage environments.

LITERATURE REVIEW

Bageshwar et al. investigated MPC for ACC/CC [5], and divides longitudinal following into transitional maneuvers and steady-state operation. Transitional maneuvers occur when there is no leader to follow, so the car switches from ACC to CC. Simulation results validate the method, but also demonstrate that collisions may occurs between the lead and follower vehicle.

Corona et al. employed MPC for their non-linear ACC control of a SMART car [6]. The car model employed included the transmission system, and one of the objectives of the MPC formulation was to minimize changing gears. Comparisons of MPC with different degrees of approximations and PI controller. Simulation results demonstrate that PI control gives the least desirable results of all configurations, even though computation time is much lower.

Naus et al. also employed explicit MPC for ACC Stop- & Go [7]. The problem is posed and solved as a multi-parametric quadratic program, which is used for offline optimization to avoid expensive computation onboard the vehicle, but restricting the approach to precomputed solutions. Seven cases were used to define an envelope of working conditions and the controller was tested for these scenarios.

Bu et al. demonstrates online MPC controller on an actual Infinity FX45 vehicles [8]. For the sake of simplicity the online MPC is combined with the existing onboard ACC controller. The output coming from the MPC controller is converted into a virtual range rate command, which is fed into the ACC controller. The two stage approach was demonstrated to be quite effective at managing intervehicle spacing.

Kianfar et al. implemented an MPC controller on a Volvo S60 [9], which included additional time domain constraints to assure that the platoon remains string stable.

DYNAMIC MODEL

The MPC controller structure used in this work is derived from the model presented by Kianfar [4].

\[
\begin{align*}
\dot{\gamma}_i(t) &= \gamma_i(t) - \omega_i(t), \quad i = 1, \ldots, N \\
\dot{\omega}_i(t) &= d_i(t) - v_i(t) - g_i(t) - \omega_i(t) - \tau_i(t), \\
d_i(t) &= \text{desired relative distance between cars}, \\
v_i(t) &= \text{velocity of the vehicle}, \\
\gamma_i(t) &= \text{relative distance to the car directly ahead and behind are}, \\
\omega_i(t) &= \text{velocity errors. These are defined as:}
\end{align*}
\]

where \( \tau_i \) represents the closed-loop bandwidth. We are assuming a platoon of \( N \) vehicles, vehicle \( i \) being the lead vehicle. Subscript \( i \) denote the number of the vehicle. The cars are assumed to have 2 lidars, one located in front and one at the back. The relative distances to the car directly ahead and behind are:

\[
\begin{align*}
\gamma_i(t) &= \gamma_{i-1}(t) - \gamma_i(t), \\
\omega_i(t) &= d_i(t) - v_i(t) - g_i(t) - \omega_i(t) - \tau_i(t).
\end{align*}
\]

The actual value of \( d_i(t) \) is not constant, but is a linear function of the spacing policy. Constant time headway spacing policy is used in our model. The formula for \( d_i(t) \) is defined as:

\[
\begin{align*}
\gamma_i(t) &= \gamma_i(t) - \omega_i(t), \\
\omega_i(t) &= d_i(t) - v_i(t) - g_i(t) - \omega_i(t) - \tau_i(t).
\end{align*}
\]

The \( h_i \) is called the desired headway time. \( c \) is the desired spacing at standstill (which is added to the actual length of the car to give a virtual vehicle length used with zero spacing policy at zero velocity). The spacing policy transfer function is then:

\[
\begin{align*}
\gamma_i(t) &= \gamma_i(t) - \omega_i(t), \\
\omega_i(t) &= d_i(t) - v_i(t) - g_i(t) - \omega_i(t) - \tau_i(t).
\end{align*}
\]
The input to this transfer block is the position of the car. Each car is also equipped with a wireless data receiver, which receives the acceleration information of both the preceding and following vehicles.

**MPC Control Model**

Instead of using a single MPC, we adopted an MPC with switching in order for the error to not accumulate when the rear car is far away. The switch is based on \( \bar{d}_i(t) \). If \( \bar{d}_{i+1}(t) \) is smaller than 10m, then the following MPC model with 6 states, and rear-end collision check, is used. Otherwise, 4 state MPC, developed by Kianfar et al. [4], with only a preceding vehicle collision-check is used. This ensures that our controller performs equivalently to most other MPC controllers in most situations, and even better in critical cases. States used for the first MPC are \( x = [d_i \ \bar{v}_i \ a_i \ v_i \ \bar{d}_{i+1} \ \bar{v}_{i+1}]^T \). State update equations are given by:

\[
\begin{align*}
\bar{v}_i &= v_i - \alpha_{i-1} - \frac{
abla x}{\Delta t} \\
\bar{v}_{i+1} &= \bar{v}_i - \Delta t \bar{a}_i \\
\bar{d}_{i+1} &= \bar{v}_{i+1} - \Delta t \bar{a}_{i+1} \\
\end{align*}
\]

State dynamics equation given by:

\[
x'(t) = Ax(t) + Bu(t) + B\bar{u}(t),
\]

where

\[
A = \begin{bmatrix} 0 & 1 & -h_i & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1/\tau_i & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},
\]

\[
B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & -h_i \\ 0 & 0 \\ 0 & 0 \end{bmatrix},
\]

\[
C = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}
\]

The state dynamics equation will be discretized before using it in the MPC. This ensures that the platoon leader is followed without any delays.

The cost function for the MPC controller is defined as following:

\[
\begin{align*}
\text{cost function} &= \sum_{i=1}^{n} [y(k+i+1) - r(k+i)]^T Q [y(k+i+1) - r(k+i+1)] + \\
&\Delta u(k+i)^T R_u \Delta u(k+i) + u(k+i)^T R_u u(k+i),
\end{align*}
\]

where \( k \) represents the current time instance and \( p \) is the controller prediction horizon [10]. This optimization problem is bounded by constraints [10]:

\[
\begin{align*}
\Delta u_{i_{\text{min}}} - \varepsilon_{u_{i_{\text{max}}} - 0} &= u_{i_{\text{max}}} - u_{i_{\text{min}}} \\
\Delta u_{i_{\text{min}}} - \varepsilon_{u_{i_{\text{max}}} - 0} &= \Delta u_{i_{\text{max}}} - \varepsilon_{u_{i_{\text{max}}} - 0} \\
y_{i_{\text{min}}} - \varepsilon_{y_{i_{\text{max}}} - 0} &= y_{i_{\text{max}}} - \varepsilon_{y_{i_{\text{max}}} - 0},
\end{align*}
\]

Here \( \varepsilon \) represents the panic variable, which is set to 1 if the controller cannot get a solution within the bounds. The vector \( V \) defines which constraints are hard and which are soft, and their level of softness.

The second MPC controller has nearly the same properties as the first, the only differences being the states and state update equations. The states used for the second MPC are:

\[
x = [d_i \ \bar{v}_i \ a_i \ v_i]^T. \quad \text{State update equations are given by:

\[
\begin{align*}
\bar{v}_i &= v_i - \alpha_{i-1} - \frac{
abla x}{\Delta t} \\
\frac{\bar{d}_{i+1}}{\Delta t} &= \bar{v}_i - \Delta t \bar{a}_i \\
\end{align*}
\]

The state dynamics equation is:

\[
x'(t) = Ax(t) + Bu(t) + B\bar{u}(t),
\]

where

\[
A = \begin{bmatrix} 0 & 1 & -h_i & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},
\]

\[
B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},
\]

\[
C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]
CONTRAINTS ON THE SYSTEM

A critical aspect of using an MPC is choosing the right constraints and right level of softness for each one of them. \(d_i\) and \(d_{i+1}\) are the distance error between vehicles. Ideally, this should be zero with both the preceding and the following vehicle, but the minimum constraint for both of these are set as \(-h_i \frac{\bar{v}_i}{4}\). This allows \(u_i\) to get closer to preceding vehicle to avoid collision with the follower, while still maintaining a safe distance of \(\frac{3h_i \bar{v}_i}{4}\). A maximum value is defined for the preceding vehicle while there is no upper limit for the follower vehicle.

\(\bar{v}_i\) and \(\bar{v}_{i+1}\) are the velocity error and should be kept within a certain limit to ensure performance. There is a lower and upper limit for the preceding vehicle, while for the follower vehicle there is only a lower limit.

\(u\) is the inputs to the actuator and its limits are defined by the limits of the car itself.

\(v_i\) is the velocity of our vehicle. It also has its own limitations, both coming from the actuator and the lateral controller, to keep the stability of the vehicle.

To assure that the system is string stable we have to make sure that the acceleration of the vehicle is always less than the preceding vehicle if the acceleration is positive. If the acceleration is negative then it becomes the lower limit. Proof of string stability can be seen in Kianfar et al.’s paper [9]

All constraints regarding the 2\(^{nd}\) follower, position and velocity errors, are defined as semi soft constraints. The priority of the controller is still to keep a safe distance with the preceding vehicle, rear end collision check is just and additional functionality.

Actuator and velocity error are defined as hard constraints, while the acceleration is semi-hard. When the following car gets closer to the vehicle and preceding vehicle is moving at constant speed, preset constraint limits the car to have zero acceleration. In this case acceleration constraint will be broken.

Velocity error with the lead vehicle is also defined as soft constraint.

Minimum position error with the preceding vehicle is defined as a hard constraint, while the maximum is semi-hard.

SIMULATIONS

Two cases will be shown to demonstrate the working of the proposed controller. All the simulations are done in MATLAB Simulink using the Model Predictive Control Toolbox.

Velocity profiles of the first and third vehicles are set manually, only the second vehicle is equipped with MPC controller. In each of the plots V1 represents the lead vehicle, while V3 represents the last vehicle.

The car model used belongs to a Volvo S60 [9]. For the car, \(K\) is set to 1 and \(\bar{v}_i\) to 0.4. Time headway \(h\) is defined as 1 sec. The weights of the cost function are like the following: \(R_{Du} = 0\), \(R_u = 0\) and diagonals of matrix \(Q\) are 10, 8, 3, 0, 6, 8.

When the 3\(^{rd}\) vehicle is within 10 meters of the 2\(^{nd}\) vehicle, the weights given previously is used with the 6-state Otherwise the state vector of MPC is modified to be \(x = [E_{PLATF}ONT \ E_{V_i}ONT \ a_i \ v_i]^T\) and new diagonals for \(Q\) will be 10, 8, 3, and 0. When the follower vehicle goes out of bound, then the regular MPC takes on the controller.

Constraints on the system are:

\ [-\frac{x \ v_i}{3} \ v_i \ a \ -2 \ -h_i \ \bar{v}_i/4 \ -2.5 \ -4.5 \ a \ 0 \ -h_i \bar{v}_i/4 \ -2.5 \ \bar{v}_{i+1} \ -2.5.]

(14)

The level of hardness for each one of these constraints can be seen by looking at the \(V_{V_{i min}}^V\) and \(V_{V_{i max}}^V\), which were defined in equation 10. Zero exponent hard constraint, while one is a soft constraint. Smaller the number gets, constraint gets harder. Constraints coming from the limitations of the vehicle itself had to be set as hard. Also the spacing error with the preceding vehicle \(\bar{d}_i\) has a hard constraint too. For the 6-state MPC the hardness values are:

\[
V_{V_{i min}}^V = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}
\]

(15)

For the 4-state MPC hardness values are:

\[
V_{V_{i min}}^V = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}
\]

(16)

In the first case, first and third vehicle start accelerating with the same acceleration and keep constant speed after 30\(^{th}\) second. But the third vehicle has some small sinusoidal disturbances in its velocity profiles. Figure 3 shows the velocity profiles of all three vehicles, and how the second vehicle adjusts its velocity to keep a safe distance to both vehicles.
As specified in the constraints, the position error with the first vehicle never goes below $-\frac{h}{4}$, but since the constraints for the second vehicle were soft it can go below this if necessary, even though it’s not desired. Position error ($\tilde{d}_{i+1}$) going below zero does not mean collision; it indicates that the desired spacing policy is violated. This is mainly because the priority of this controller is still the first vehicle. Figure 4 shows the position error graphs for both vehicles.

Case 2 shows the need of a MPC switching by temporarily disabling the 4-state MPC. In the case that third vehicles slows down or stops, MPC still keeps adding that error to the cost function and tries to minimize it by increasing the spacing with the first vehicle. This is an undesired effect because it is important for us to minimize the total platoon length. Figures 5 and 6 shows what would happen if no switching is used and figures 7 and 8 show the actual controller with switching.
CONCLUSION & FUTURE WORKS

In this paper, we investigated the use of a modified CACC controller that also tries to avoid rear end collisions of the following vehicle, by breaking the spacing policy with the preceding vehicle just enough to avoid collision, while making sure that no real danger is caused to the preceding vehicle.

The results from the simulations were satisfactory, but the control still needs some improvements. The car and communication delays are planned to be considered in our consequent work, along with the addition of parameter estimators to get accurate plant models. Our group is currently working on a platform to test all the CACC algorithms on scaled car models. Once they are ready, experimental results will also be submitted.

Our future works include testing of the effects of this controller on passengers comfort and formal stability analysis of switching controller between regular CACC and CACC with rear end collision control.

ACKNOWLEDGMENTS

The authors would like to thank to Nuvation for their valuable inputs and helps.

REFERENCES