A New Nonlinear Model Predictive Control Algorithm for Vehicle Path Tracking

Georg Schildbach (Elektronische Fahrwerksysteme GmbH)
E-mail: gschildbach@gmail.com

Model Predictive Control (MPC) is a powerful technique for the control of nonlinear, multi-input multi-output systems with input and state constraints. It has been considered for path tracking control of automated vehicles in many projects. However, MPC faces several challenges in practice, mainly regarding computation times and implementation difficulties. This paper presents a new MPC algorithm to address these challenges specifically for vehicle path tracking. The algorithm comes with a code generator for open-source, library-free C-code. Thus it can be easily deployed to embedded platforms. The algorithm consists of a single controller block, which only requires the specification of vehicle and tuning parameters. Simulation and experimental results show the effectiveness of the code, with computation times ranging within a few milliseconds.

Topics: automated driving and collision avoidance; active safety and driver assistance systems

1. INTRODUCTION

Control design for the automation of vehicles typically comprise, at a minimum, a module for path planning and path tracking [1]. Path planning designs a reference path (or trajectory) that avoids obstacles and ensures passenger comfort. Path tracking controls the vehicle, by commanding the steering and throttle/brake, with the goal of tracking the given reference path.

For a good overall performance, the reference path should be compatible with the vehicle dynamics and respect the actuation constraints, so a conventional controller is able to track it closely. If these requirements are not satisfied, the controller may be difficult to tune, produce large tracking errors, or even become unstable.

The computation of reference paths which satisfy these requirements is generally difficult. So the problem is often simplified. Common approaches are to neglect the dynamic component (i.e., computing paths instead of trajectories), to decouple the lateral and longitudinal vehicle motion (i.e., separating steering and acceleration control), to sacrifice dynamic feasibility (i.e., drivability of the path), or to ignore the actuator constraints. Other approaches choose a heuristic to simplify the solution, by limiting the search to a small (often finite) set of possible paths, for example by random sampling.

MPC is a holistic method for path planning and control. It is highly effective for nonlinear, multi-input multi-output systems with input and state constraints. As such, it is naturally suited for problems in vehicle control [1]. Since it is generally not able to find the globally optimal solution, it still has to rely on a high-level path planner for a reference input. However, it can significantly lower the requirements for the path planner and deliver an excellent control performance.

In contrast to conventional control methods, the presented MPC algorithm requires only a piecewise linear path and a desired velocity, both of which may be generated independently. The path does not have to be dynamically feasible or feasible with respect to actuator or safety constraints. The MPC algorithm will automatically satisfy these constraints and follow the closest feasible and drivable path. Here ‘close’ is defined by a tunable cost function, which allows to trade-off the accuracy of path tracking with the smoothness / input usage. Unsafe driving areas (e.g., obstacles or lane boundaries) are specified explicitly as safety constraints, which the MPC will keep at all cost.

2. NEW CONTRIBUTION

The main drawbacks of MPC remain its computational complexity and difficult implementation procedures. This work aims at addressing these challenges.

Many excellent, general purpose optimization packages exist for solving the underlying finite-horizon optimal control problem (FHOCP). However, these solvers are not available on embedded platforms. Moreover, recent research [2,3,4] has shown that significant speed-ups (of factor 10 and above) are possible, by exploiting the specific properties of the FHOCP.

However, most dedicated MPC solvers are designed for linear systems. They may be used also for nonlinear systems, e.g., by Sequential Quadratic Programming (SQP), but tend to lose some of their efficiency.

In this work, a new Active-Set Method (ASM) is presented, which is known to be very efficient in the context of linear MPC [2]. Here it is tailored specifically to nonlinear systems and for the control problem of vehicle path tracking. It is easy to use also for non-experts, and it achieves a significant speed-up over what is generally thought possible for this application.

The algorithm exploits the same sparsity structure as a state-of-the-art Interior-Point Method (IPM) [3], which is generally favored for large-scale nonlinear optimization. The advantages of the ASM over the IPM, however, are its capabilities for warm-starting and maintaining a feasible solution at all times.

The algorithm comes with an open-source code generator for library-free C-code, which is easily deployed to embedded platforms. It consists of a single function block that can also be used in Simulink (s-function). Besides dynamic measurements, the function block only requires specification of vehicle and tuning parameters. The code package can be downloaded from: https://app.box.com/s/lmmkpakkx177362wz9b0t5ycs9opxe7h

3. PROBLEM DESCRIPTION

3.1 Vehicle Model

An arbitrary dynamic model can be used with the code. In this paper, it is a kinematic bicycle model [1], as shown in Figure 1. Its equations of motion are given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\xi\\ \eta\\ \varphi\\ v\\ \delta\end{bmatrix}=\begin{bmatrix}v\cos(\varphi+\beta)\\ v\sin(\varphi+\beta)\\ v\cos\beta/(l_f+l_r)\tan\delta\\ a\\ \dot{\delta}\end{bmatrix},$$

where

$$\beta=\tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan\delta\right).$$

The model has $n=5$ states $x=[\xi,\eta,\varphi,v,\delta]$ and $m=2$ control inputs $u=[a,\dot{\delta}]$. The Center of Gravity (CoG) serves as the coordinate reference point. Two parameters, the distance from the CoG to the rear and front axle (l_r and l_f), are used to customize the model.

For computations, the model in continuous time t has to be discretized with a selected sampling time t_s. The discrete-time model is written in vector form as

$$x_{i+1}=f_{t_\mathrm{s}}(x_i,u_i)\,,\quad \text{for } i=0,1,2,\dots\,.$$

where x_i, u_i denote the state, input at time $t=it_\mathrm{s}$.

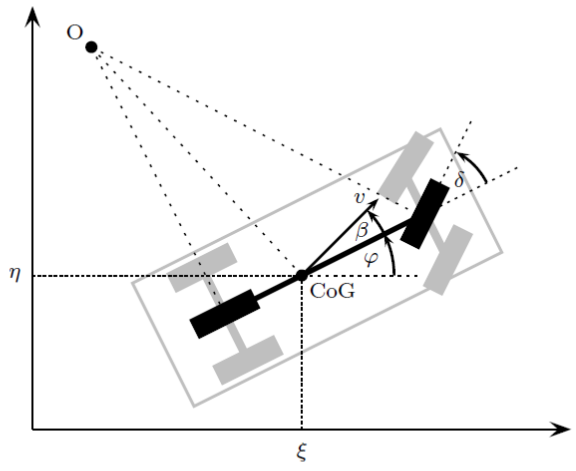

Fig. 1: Kinematic bicycle model (black) of a car (gray).

3.2 Reference Path

The reference path, to be tracked by the MPC, does not come in a specific functional form (e.g., arc, clothoid, sigmoid). It is represented by a polygon with the following information for each linear element $j=1,\dots,N_\mathrm{path}$:

- the coordinates of the starting point ξ_j,η_j;
- the element's orientation φ_j;
- the reference speed v_j and steering angle δ_j;
- the element's length l_j.

3.3 Cost Function

Based on the localization of the vehicle along the reference path, a list of reference states is generated (indicated by the subscript 'ref'). The objective is then to minimize the deviation of the predicted vehicle trajectory over a prediction horizon N. The deviation at each time step $i=1,\dots,N$ is defined by a stage cost

$$\begin{aligned}\ell(u_{i-1},x_i)=r_a\big(a_{i-1}-a_{\mathrm{ref},i-1}\big)^2+r_{\dot\delta}\big(\dot{\delta}_{i-1}\big)^2\\ +q_\xi\big(\xi_i-\xi_{\mathrm{ref},i}\big)^2+q_\eta\big(\eta_i-\eta_{\mathrm{ref},i}\big)^2\\ +q_\varphi\big(\varphi_i-\varphi_{\mathrm{ref},i}\big)^2+q_v\big(v_i-v_{\mathrm{ref},i}\big)^2\\ +q_\delta\big(\delta_i-\delta_{\mathrm{ref},i}\big)^2\,,\end{aligned}$$

where $r_a,r_{\dot\delta}$ and $q_\xi,q_\eta,q_\varphi,q_v,q_\delta$ denote the (positive) input and state weights, selected for tuning.

3.4 Input Constraints

The sum of stage costs over $i=1,\dots,N$ is minimized, subject to bounds and rate constraints on the inputs:

$$u_\mathrm{min}\le u_{i-1}\le u_\mathrm{max}\,,$$
$$\Delta u_\mathrm{min}\le u_i-u_{i-1}\le \Delta u_\mathrm{max}\,,$$

where the inequalities are defined element-wise.

3.5 State Constraints

State constraints on the CoG coordinates ξ_i, η_i are used to model areas where it is unsafe for the vehicle to drive (e.g., due to obstacles or road/lane boundaries). These constraints are modeled as soft constraints, to avoid infeasibility of the FHOCP. The penalty functions are designed to keep the cost function continuously differentiable. The penalty parameters can be specified for tuning.

4. PRELIMINARY RESULTS

Results from various simulations and experiments will be available in the full paper. Preliminary results show an excellent performance of the new algorithm, in terms of path tracking and obstacle avoidance. Exemplary computation times for various horizon lengths N are given in Table 1.

Table 1. Computation times on a MacBook Pro (2.7 GHz).

Computation time	N = 20	N = 30	N = 40
Average [ms]	0.53	0.68	0.95
Maximum [ms]	2.8	3.2	4.5

REFERENCES

[1] Kong, J. et al., "Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design", IEEE Intelligent Vehicles Symposium, Seoul, Korea, 2015.
[2] Houska, B. et al., "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization", Optimal Control Applications and Methods, vol. 32(3), 2011.
[3] Domahidi, A., "Methods and Tools for Embedded Optimization and Control", Ph.D. Thesis, ETH Zurich, 2013.
[4] Richter, S., "Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control", Ph.D. Thesis, ETH Zurich, 2012.