International Journal of Automation and Computing 04(1), January 2007, 100-106
DOI: 10.1007/s10453-004-5872-7

Experimental Tests of Autonomous Ground Vehicles with Preview

Cunjia Liu[1,*] Wen-Hua Chen[1] John Andrews[2]
[1]Department of Aeronautical and Automotive Engineering, Loughborough University, Loughborough, LE11 3TU, UK.
[2]Nottingham Transportation Engineering Centre, Faculty of Engineering, University of Nottingham, Nottingham, NG7 2RD, UK.

Abstract: This paper describes the design and experimental tests of a path planning and reference tracking algorithm for autonomous ground vehicles. The ground vehicles under consideration are equipped with forward looking sensors that provide a preview capability over a certain horizon. A two-level control framework is proposed for real-time implementation of the Model Predictive Control (MPC) algorithm, where the high-level performs on-line optimization to generat the best possible local reference respect to various constraints and the low-level commands the vehicle to follow realistic trajectories generated by the high-level controller. The proposed control scheme is implemented on an indoor testbed through networks with satisfactory performance.

Keywords: Model predictive control, Autonomous vehicle, Online optimization.

1 Introduction

There has been an increasing interest in the development of autonomous ground vehicles from both industry and academia in the past decades. Versatile tasks can be performed by autonomous vehicles to replace the human or human operated vehicles. Individual autonomous ground vehicles, for example, can perform explorations instead of humans in unknown and dangerous environments, whereas coordinated multi-vehicles can carry out more appealing tasks such as distributed sensor networks. An essential function required for an autonomous vehicle performing a task is to track reference trajectories or waypoints either pre-planned beforehand or dynamically generated by a supervisory layer. Such a function relies on the tracking control of the vehicle.

The kinematics of a ground vehicle are commonly described by a nonholonomic model, which is a highly nonlinear system, and therefore draws particular attention from control community. The related motion control problem for ground vehicles has been studied for many years, and numerous algorithms have been developed[1, 2]. More recently, Model Predictive Control (MPC) has gained more and more attentions. As all autonomous vehicles are equipped with sensors that can provide local information within a certain range, MPC (also known as receding horizon control) provides a promising, natural framework for this problem since the environment information is changed/updated as the vehicle proceeds. Both vehicle dynamics and environmental information can be considered using this approach. Therefore, it is argued that MPC may offer a number of advantages over other methods[3]. On the other hand, MPC has the property of handling with the control saturations and state constraints which naturally arise from practical problems. The application of MPC on nonholonomic mobile robots can be found in [4 − 6]. The corresponding stability issues have also been discussed in [7, 8].

In the MPC scheme, an optimization problem (OP) has to be solved within each sampling interval. This is the main obstacle of applying MPC into plants with fast dynamics such as vehicles. Although few MPC algorithms in this field have been implemented in real-time, they are based on linear settings[9, 10]. The linearized model is only valid when the vehicle is close to the reference trajectory. For the nonlinear vehicle model and more general setting, a computationally intensive nonlinear optimization problem is involved in calculating the control command[3]. Real-time implementation of nonlinear MPC on autonomous vehicles still poses a major challenge[11].

Focusing on the real-time implementation of the MPC algorithm, this paper proposes an optimization based control framework which combines MPC with traditional control techniques. Instead of attempting to implement a single nonlinear MPC, the proposed framework employs a two-level control structure. The high-level controller uses a nonlinear MPC algorithm to solve the tracking control problem with respect to the vehicle's nonholonomic constraint and other environment constraints. The low-level controller is designed based on the linearization around a reference trajectory provided by the high-level controller to stabilize the vehicle in the presence of disturbances and uncertainties. These two level controllers are operated at different time scales. The high-level MPC strategy runs at a lower sampling rate allowing enough time for performing online nonlinear optimization, while the low-level controller is executed in a much higher sampling rate to respond to external disturbances. Therefore, the proposed optimization based two-level control framework facilitates real-time implementation since it can simultaneously cope with the heavy computational burden demanded by the online optimization solver and the fast feedback as required by vehicle stabilization and control.

Both the simulation and experimental results are presented in this paper to verify the proposed approach. The time settings related to the MPC implementation are discussed. Furthermore, the hardware structure of the control

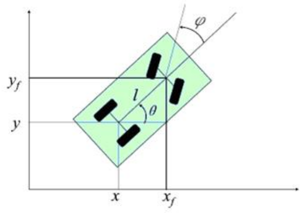
Manuscript received date; revised date
*Corresponding author. E-mail: c.liu5@lboro.ac.uk

framework is introduced, and it shows the flexibility to test the MPC algorithms on other physical plants with fast dynamics in real-time.

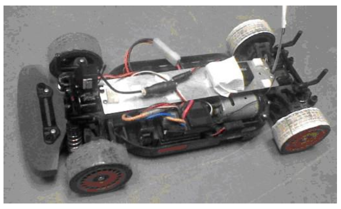
The rest of this paper is organized as follows. Section 2 describes the vehicle model. In Section 3, the high-level MPC is formulated in a discrete time setting with an emphasis on implementation, followed by simulation results. The two-level control framework including the design of the low-level controller is introduced in Section 4. Then the experiment setting is described and experimental results are presented to verify the proposed approach. Section 5 concludes this paper.

2 Vehicle model

The autonomous vehicle under consideration in this paper is a rear wheel driving ground vehicle and is equipped with forward looking sensors to provide preview for the path and the surroundings over a certain horizon. The configuration of the rear-wheel driving vehicle, as the TT01 radio controlled model car used in the Autonomous System Lab at Loughborough University, is shown in Fig. 1.



(a) Vehicle schematic



(b) Experiment vehicle

Fig. 1 Ground vehicle

The states of the model are $\mathbf{x} = [\ x \quad y \quad \theta\ ]'$, where (x, y) are the coordinates of the centre point of the rear axle, θ is the heading angle of the car body with respect to the x axis. Angle φ in Fig. 1(a) is the steering angle of the front wheels with respect to the vehicle's longitudinal line, which can be seen as a control input. Another parameter of the model is the distance between the front axle and the rear axle, which is l in the Fig. 1(a). The kinematical relationship can be described using the following mathematical model:

$$\begin{aligned} \dot{x} &= v \cdot \cos\theta \\ \dot{y} &= v \cdot \sin\theta \\ \dot{\theta} &= v \cdot \frac{\tan\varphi}{l} = v \cdot c \end{aligned} \tag{1}$$

where the control inputs for the vehicle are: φ the steering angle and v the line velocity. The curvature can be defined as:

$$c = \tan\varphi / l \tag{2}$$

For simplicity of control design, c is used as an input. That is, $\mathbf{u} = [\ c \quad v\ ]'$. If the input c is calculated, the steering angle φ can be derived from the relationship (2).

The nonholonomic constraint can be represented as follows [6]:

$$\dot{x}\sin(\theta + \varphi) - \dot{y}\cos(\theta + \varphi) - \dot{\theta}\cos\theta \cdot l = 0 \tag{3}$$

One can easily verify that the nonholonomic constraint (3) is involved in this model. In terms of the control constraints, the steering angle is usually restricted within a special range imposed by the actual mechanical saturation. The steering angle limit also imposes a minimum turning radius on the vehicle, which is:

$$R_{min} = |c_{max}|^{-1} = l / \tan\varphi_{max} \tag{4}$$

The constraints on the car velocity can also be added in the same way.

3 Tracking control using MPC

Obviously, the kinematic constraints, i.e. the nonholonomic constraint, and the input constraints prohibit the vehicle from tracking arbitrary trajectories. Therefore, a tracking controller needs to address both the local trajectory regeneration and the tracking problems. That means to provide the optimal, at least a feasible, trajectory to follow as well as the corresponding control sequence, based on a given reference trajectory or way points.

3.1 Time setting

MPC is an optimal control strategy that uses the model of the plant to obtain the optimal control sequence by minimizing an objective function which penalizes the unexpected errors. At each sampling instant, a model is used to predict the future behavior of the plant over a prediction horizon[12]. Based on these predictions, the objective function is minimized with respect to the future sequence of inputs. Thus MPC requires the solution of a constrained OP at each sampling instant. Although prediction and optimization are performed over the future time horizon, only the control inputs for the current sampling step or first few steps are eventually used to drive the system. The same procedure is repeated at the next sampling instant with updated system states for a receding horizon.

In order to exploit the MPC strategy, the vehicle tracking problem needs to be formulated into the MPC formation. Moreover, since the online optimization is performed on a digital computer, it is necessary to represent the original problem in a discrete form. Considering a time step T_d, the discrete vehicle model can be obtained, for example, by using Euler approximation as follows:

$$\begin{aligned} x(k+1) &= x(k) + T_d \cdot v(k) \cdot \cos\theta(k) \\ y(k+1) &= y(k) + T_d \cdot v(k) \cdot \sin\theta(k) \\ \theta(k+1) &= \theta(k) + T_d \cdot v(k) \cdot c(k) \end{aligned} \tag{5}$$

The compact form of (5) is $\mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k))$. The prediction of vehicle states at sampling instant k is represented as follows:

$$\mathbf{x}(k+j+1|k) = f(\mathbf{x}(k+j|k), \mathbf{u}(k+j|k)) \tag{6}$$

where, k indicates the present sampling instant, and $k+j$ indicates the j-th predicted step evolved from step k.

The time step T_d is a crucial parameter. Because of the Euler approximation, there is an error between the discrete model and the continuous model which monotonically increases with T_d. The MPC designed on the discrete model can stabilize the original continuous model if T_d is small enough[13]. However, this time interval is the duration in which the nonlinear optimization should be finished. To avoid this conflict, this paper introduces another important parameter T_s defined as the MPC sampling time, which decides the interval of the controller updating the current states and generating a new control sequence. T_s is usually decided by the maximum OP solution time. In the conventional MPC setting T_s usually equals T_d. Considering the intractable optimization problem within T_d, the MPC sampling time can be chosen to be greater than time step T_d. We choose the MPC sampling time as $T_s = N \cdot T_d$. Hence, the time allowed for solving the OP is enlarged. Consequently, during one sampling interval, the first N steps in the optimized control sequence need to be applied.

To clearly explain the time setting, an example is illustrated in Fig. 2. In this example N is set to 2, so the sampling interval is $T_s = 2T_d$, which is also the time allowing to solve the OP. The prediction horizon H is 4 MPC sampling intervals. At each sampling instant $K_{jN}, j = 0, 1, 2 \ldots$, the MPC obtains the new system states and calculates the new control sequence in which the first N steps are actually applied to the plant.

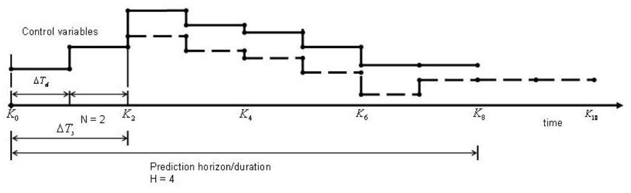


Fig. 2 Sampling time index

3.2 Controller design

Under the above time setting, the discrete MPC is employed for real-time application. By defining the reference trajectory $\mathbf{x}_r$ and the tracking error $\mathbf{x}_e = \mathbf{x}_r - \mathbf{x}$, we can formulate the following objective function to be minimized:

$$J(k) = \sum_{i=0}^{H-1} \sum_{j=0}^{N-1} L(\mathbf{x}(k+iN+j|k),\, \mathbf{u}(k+in+j|k)) + F(\mathbf{x}(k+HN|k)) \tag{7}$$

$$L(\mathbf{x}(k),\, \mathbf{x}_r(k),\, \mathbf{u}(k)) = \mathbf{x}_e(k)' Q \mathbf{x}_e(k) + \mathbf{u}(k)' R \mathbf{u}(k)$$
$$F(\mathbf{x}(k+HN),\, \mathbf{x}_r(k)) = \mathbf{x}_e(k+HN)' P \mathbf{x}_e(k+HN)$$

where, $L(\mathbf{x}(k),\, \mathbf{x}_r(k),\, \mathbf{u}(k))$is the penalty for each time step, $F(\mathbf{x}(k+HN),\, \mathbf{x}_r(k))$ is the terminal penalty, H is the prediction horizon and P, Q and R are the positive definite weighting matrices.

The nonlinear optimization problem that minimizes the objective function subjected to various constraints can be stated as:

$$\mathbf{x}_{ref},\, \mathbf{u}_{ref} = \arg\min_{\mathbf{x},\,\mathbf{u}} J(k) \tag{8}$$

subject to:

$$\begin{aligned} &\mathbf{x}(k+j+1|k) = f(\mathbf{x}(k+j|k),\, \mathbf{u}(k+j|k)) \\ &\mathbf{x}(k+j|k) \in \mathbf{X} \\ &\mathbf{u}(k+j|k) \in \mathbf{U} \\ &j = 0,\, 2,\, \cdots,\, N-1 \\ &\mathbf{x}(k|k) = \mathbf{x}_0 \end{aligned}$$

where the first term is the compact form of vehicle dynamics, and $\mathbf{X}$, $\mathbf{U}$ are control and state constraints, respectively. This optimization problem must be solved at each sampling instant, producing the local reference trajectory $\mathbf{x}_{ref}$, and the optimal control sequence $\mathbf{u}_{ref}$. The related MPC stability issue has been discussed in ref. [7], and is beyond the scope of this paper, where we will focus on the strategy of real-time implementation.

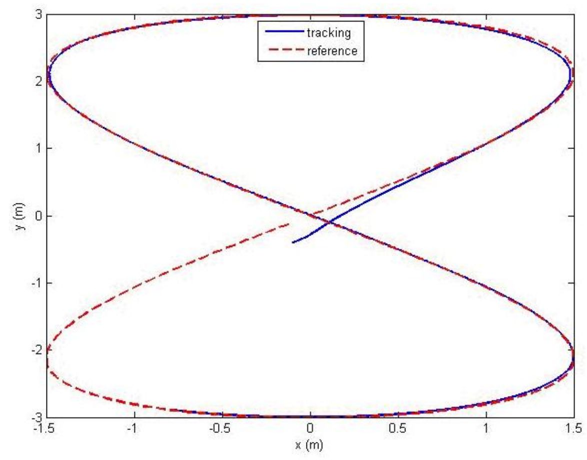
3.3 Numerical simulation

To show the performance of the MPC strategy, the simulation was firstly carried out based on parameters for the Subaru TT01 model car in our lab. After various driving tests in the lab, the model of the vehicle has been built. Simulations for a number of tasks have been carried out. A classical task is to track an eight shape trajectory defined as follows:

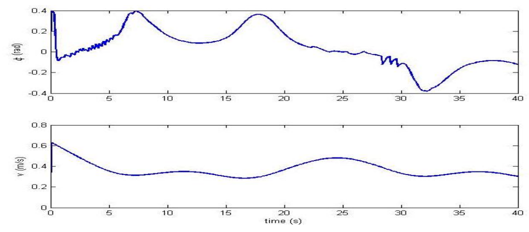
$$\begin{aligned} x_r(t) &= 1.5 \cdot \sin(t/4) \\ y_r(t) &= 3 \cdot \sin(t/8) \\ \theta_r(t) &= \arctan 2(\dot{x_r}(t),\, \dot{y_r}(t)) \end{aligned} \tag{9}$$

where the function arctan2 is a four-quadrant inverse tangent function.

In the simulation, the matrices Q and R in the cost function of MPC are chosen as $diag(1,\, 1,\, 0.5)$ and $diag(0.2,\, 0.2)$ respectively. The terminal penalty weighting P is chosen as $diag(10,\, 10,\, 5)$. The discrete sampling time of model T_d is 0.1 s and the MPC sampling time T_s is set to 0.5 s, which means each control sequence is applied for the first 5 steps. The prediction horizon H is set to 4 steps, which implies the overall prediction duration is 2 s. In terms of input constraints, the steering angle φ is limited from -0.4 rad to 0.4 rad and the speed from 0.15 m/s to 0.8 m/s.



(a) Tracking result

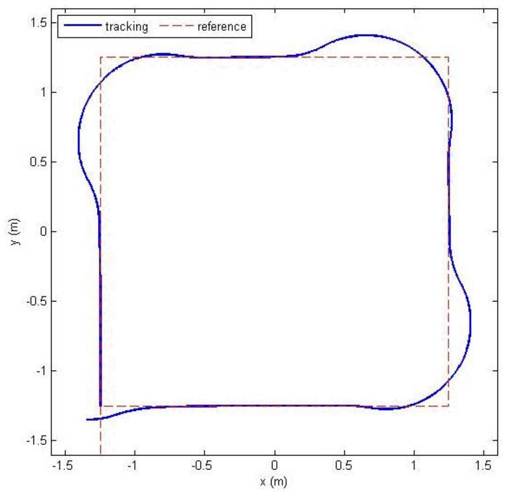


(b) Control inputs

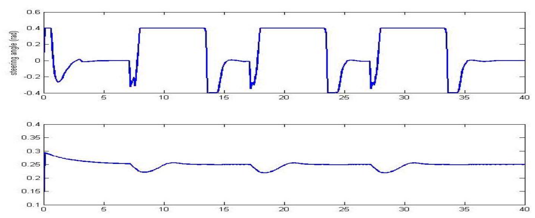
Fig. 3 Eight-shape tracking

The simulation results are shown in Fig. 3. It can be seen that although the initial position of the vehicle is not on the reference trajectory, the controller drives the vehicle back to the reference track. The average OP solving time is less than 0.05 second using a computer with 2.3GHz CPU in a Matlab environment, however the maximum time can reach to 0.4 seconds when the measured error states are far away from the reference at the beginning.

Another task in the simulations is to force the car to follow a square trajectory which by no means can be tracked by a car-like vehicle accurately. However, the MPC based controller can provide an optimized smooth local trajectory to follow.



(a) Tracking result



(b) Control inputs

Fig. 4 Square tracking

The simulation results can be found in Fig. 4. The tracking trajectory is smooth and the constraints for both steering angle and speed are satisfied. Parameter settings are the same as the previous simulation except that the prediction horizon is enlarged to 6 steps (i.e. 3 seconds). It can be seen that at the corner of the trajectory, the speed of the car is reduced to get a better turning as a human driver does. The big error at each corner is caused by the minimum turning radius of the vehicle. For the vehicle with $l = 0.25$ m and maximum steering angle 0.4 rad the minimum turning radius based on (4) is about 0.6 m. The computation load of this optimisation problem is heavier than the previous one. This is because there are more control variables to be optimised due to the increased prediction horizon, and there is no input reference during the turning, thereby the initial guess of the control variables are much different from their optimal values. The maximum solving time during the simulation is approaching 0.5 seconds.

4 Real-time implementation using two-level framewwork

4.1 Control framework

The premise of using MPC in vehicle control is that the formulated optimization problem can be solved within one sampling interval. If the computation time is too long, i.e. the sampling time T_s is too large, the MPC scheme would become a piecewise open-loop optimal control. Unfortunately, due to the mismatch between the mathematical model and the real plant, the noises and disturbances in the process, this kind of optimal control will not perform well as being designed.

In order to avoid these difficulties in the implementation, a two-level structure was adopted in the control framework (see Fig. 5). The high-level controller is the original MPC, which can provide an optimized reference trajectory and the corresponding control inputs, whereas the low-level controller is a traditional controller which tracks the reference trajectory and provides the stability around the reference trajectory in the presence of disturbances and uncertainties. The high-level controller runs at a lower sampling rate to adapt to the calculation time required to solve the nonlinear optimization problem. In contrast, the low-level controller works at a higher sampling rate to respond to the latest vehicle states and the external environment.

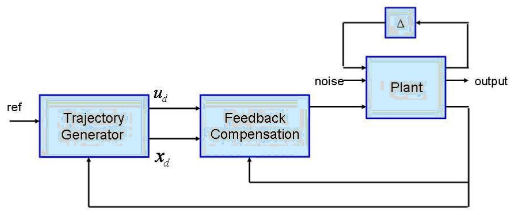


Fig. 5 Two-level control framework

4.2 Low-level controller design

The low level control has to be designed to accommodate the characteristic of the experimental vehicle. Intuitively, two PID controllers can be employed in the low-level control. One is the steering local controller, and another is the local velocity controller used to regulate the car's speed.

In the implementation, the high-level MPC provides the optimal state reference and also the corresponding optimal control inputs. The low-level controller measures the current state and then compares it with the high-level state reference. The error signals are used to generate compensation control efforts through the local controller. The overall control inputs applied to the vehicle consist of two parts: the reference control inputs and the compensation control generated by the local controller. The strategy is shown in Fig. 6.

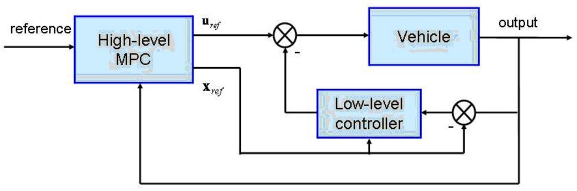


Fig. 6 Two-level control configuration

In a systematic way, the low-level controller can be designed based on perturbation models around the reference state produced by the high-level MPC. Since the low-level control works in a much higher sampling rate, the controller design can be performed in the continuous time domain. The vehicle model (1) can be linearized around the nominal trajectory and nominal inputs ($\mathbf{x}_{ref}$, $\mathbf{u}_{ref}$) as following:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \approx f(\mathbf{x}_{ref}, \mathbf{u}_{ref}) + \left.\frac{\partial f}{\partial x}\right|_{\mathbf{x}_{ref}} (\mathbf{x} - \mathbf{x}_{ref}) + \left.\frac{\partial f}{\partial u}\right|_{\mathbf{u}_{ref}} (\mathbf{u} - \mathbf{u}_{ref}) \quad (10)$$

By defining the error state $\Delta\mathbf{x} = \mathbf{x} - \mathbf{x}_{ref}$ and control compensation $\Delta\mathbf{u} = \mathbf{u} - \mathbf{u}_{ref}$. The error system can be stated as:

$$\Delta\dot{\mathbf{x}} = \left.\frac{\partial f}{\partial x}\right|_{\mathbf{x}_{ref}} \Delta\mathbf{x} + \left.\frac{\partial f}{\partial u}\right|_{\mathbf{u}_{ref}} \Delta\mathbf{u} = A\Delta\mathbf{x} + B\Delta\mathbf{u} \quad (11)$$

where,

$$A = \begin{bmatrix} 0 & 0 & -v_{ref} \cdot \sin\theta_{ref} \\ 0 & 0 & v_{ref} \cdot \cos\theta_{ref} \\ 0 & 0 & 0 \end{bmatrix} \quad (12)$$

and

$$B = \begin{bmatrix} 0 & \cos\theta_{ref} \\ 0 & \sin\theta_{ref} \\ v_{ref} & c_{ref} \end{bmatrix} \quad (13)$$

In terms of the vehicle tracking control, a novel time varying feedback control law is proposed as in (14).

$$K = \begin{bmatrix} -k_1 \dfrac{\sin\theta_{ref}}{v_{ref}^2} & k_1 \dfrac{\cos\theta_{ref}}{v_{ref}^2} & k_2 \dfrac{1}{v_{ref}} \\ k_3 \cos\theta_{ref} & k_3 \sin\theta_{ref} & 0 \end{bmatrix} \quad (14)$$

where k_1, k_2 and k_3 are control parameters to be tuned, the other parameters depend on the high-level reference. Then the state transition matrix of the closed-loop error system $\Delta\dot{\mathbf{x}} = (A - BK)\Delta\mathbf{x}$ is derived in (15):

$$\begin{bmatrix} -k_3 \cos^2\theta & -k_3 \cos\theta \sin\theta & -v \sin\theta \\ -k_3 \cos\theta \sin\theta & -k_3 \sin^2\theta & v \cos\theta \\ k_1 \dfrac{\sin\theta}{v} - k_3 c \cos\theta & -k_1 \dfrac{\cos\theta}{v} - k_3 c \sin\theta & -k_2 \end{bmatrix} \quad (15)$$

For the sake of simplicity, the subscripts for the reference states are omitted. The resulting close-loop system is a linear time-varying system. However, the closed-loop system has a very nice property. That is, its eigenvalues are always constant under the proposed control law. It can be shown from the characteristic equation of the closed-loop system given by (16).

$$\det(sI - (A - BK)) = (s^2 + k_2 s + k_1)(s + k_3) \quad (16)$$

where s is the Laplace operator.

Remark. There is no term depending on the reference trajectory in the characteristic equation (16). Therefore, constant eigenvalues are guaranteed by the proposed local control strategy for any feasible trajectory as long as the control parameters are chosen as positive. In terms a slowly varying system, this condition indicates the stability. On the other hand, the tracking performance can be easily tuned by assigning the poles to the required locations through properly choosing control parameters k_1, k_2 and k_3. This greatly simplifies the control parameter tuning process, as compared to other local controllers[10, 14].

4.3 Real-time implementation

The implementation of the two-level control framework in real-time is achieved by integrating Matlab and its xPC target real-time environment. The low-level controller is located on xPC target, and the high-level controller is executed on another computer in the Matlab environment using the SQP algorithm to solve the optimization problem online. The information exchange between them relies on the local area network (LAN) with UDP protocol (Fig. 7). The synchronization between the two computers is guaranteed by the real-time xPC target application calling Matlab program based on its own timer. The xPC target also integrates interfaces to the sensors and radio controller which measure the vehicle states and send control signals, respectively.

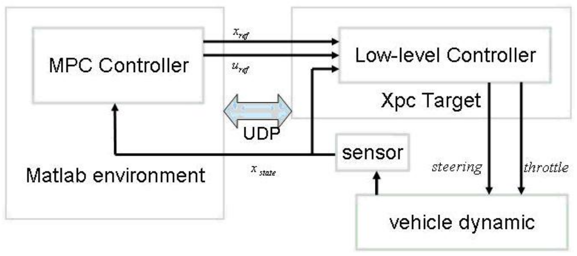


Fig. 7 Two-level control configuration

The experiment we considered is the square tracking since it is more computationally intensive. Due to the intractable optimization problem, the sampling time T_s for high-level controller is set to 0.5 seconds, which means that in every 0.5 seconds the MPC program fetches the current state of the vehicle, compares with the target trajectory and solves the formulated optimization problem, consequently producing the optimized state reference and control inputs. These outcomes are transferred to the low-level controller as references. The low-level controller running on the xPC target takes charge of tracking this reference trajectory. The sampling time of the low-level controller T_d is 0.02 seconds (50Hz).

Notice that although the high-level controller works at the sampling rate of 0.5 seconds, the actual step time of the discrete model used in the prediction, T_d, is 0.1 seconds to guarantee the prediction accuracy. Since the prediction horizon H is set to 6 steps, the prediction and the corresponding control sequence cover a duration of $N \times T_s = 3$ seconds. The number of optimization variables is calculated as $N \times T_s/T_d$, which is 30 sets of control variables. In total, there are 60 variables to be optimized.

Generally, an MPC scheme assumes instantaneous acquisition of the control sequence after the state updated. However, due to the existence of the computational delay, the previous control sequence is executed until the new optimization result is available. The related problem has been investigated in ref. [15], in this paper we follow the relationship represented in Fig. 8. At time K_i, the low-level controller sends the current states back to the high-level controller, expecting the new control sequence as a reference. Then the high-level controller takes T_c to solve the optimization problem and send results back. The control actions being applied in one control sequence may cover a period of T_s+T_c, indicated by the 'applied horizon' in Fig. 8. T_c in most cases is a small number, however when the OP is difficult to solve the time increases significantly. During the design process, we need to choose T_s greater than the maximum T_c. In the worst case, if the computational time is too long, the optimization should be terminated and output a feasible result. Within this calculation period, the low-level plays an important role, whereas the high-level controller does not have a chance to respond to the external disturbances.

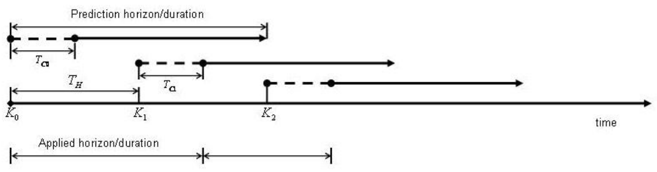


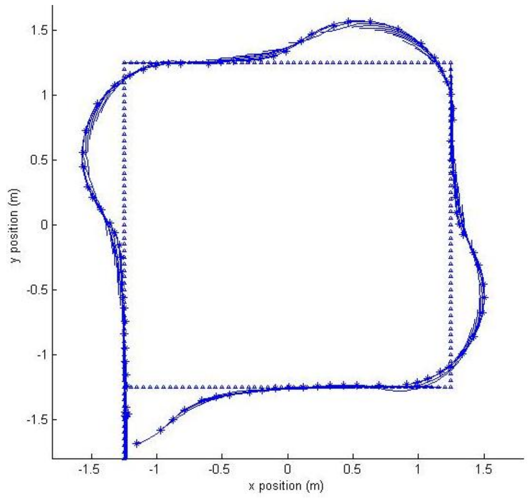
Fig. 8 Execution time index

The experiment tracking results are shown in Fig. 9. The star point in the Fig. 9(a) is the vehicle position at the moment when the low-level controller sends the vehicle state back to the high-level controller. The solid line evolved from each star point is the regenerated local trajectory for the vehicle to track. So the connection of all these star points is the actual tracking result, and the triangles represent for the original square reference trajectory. The corresponding control inputs are shown in Fig. 9(b).

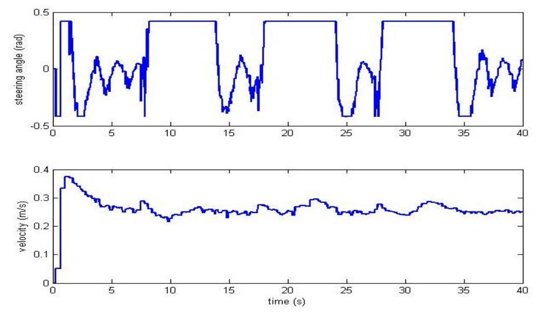
5 Conclusion

This paper describes a two-level control framework using the MPC strategy for autonomous vehicle trajectory tracking. The two-level framework overcomes the barriers of implementing computationally intensive MPC on vehicles with fast dynamics, making the MPC strategy more affordable and reliable. The design of the discrete nonliner MPC and low-level tacking control are presented. The hardware and software environment for the implementation are also described. This facility provides a convenient platform for further research on MPC algorithms and their applications in autonomous vehicles.

Based on our tests, the MPC based control framework has the capability to implement online optimization and provide stability to a nonlinear system. This two-level control framework has the potential to be applied on more complicated plants such as unmanned helicopters. In terms of the vehicle motion control, it is very convenient to put obstacle avoidance algorithms into the high-level MPC, and that can be further expanded by including a vehicle dynamics model to improve the performance.



(a) Tracking result



(b) Control inputs

Fig. 9 Square tracking experiment

References

[1] I. Kolmanovsky and N. H. McClamroch. Developments in nonholonomic control problems. *Control Systems Magazine, IEEE*, 15(6):20–36, 1995. ID: 1.

[2] P. Morin and C. Samson. Trajectory tracking for nonholonomic vehicles: overview and case study. pages 139–153, 2004. ID: 1.

[3] Yongsoon Yoon, Jongho Shin, H. Jin Kim, Yongwoon Park, and Shankar Sastry. Model-predictive active steering and obstacle avoidance for autonomous ground vehicles. *Control Engineering Practice*, 17(7):741–750, 7 2009.

[4] A. Ollero and O. Amidi. Predictive path tracking of mobile robots. application to the cmu navlab. pages 1081–1086 vol.2, 1991. ID: 1.

[5] Dongbing Gu and Huosheng Hu. Neural predictive control for a car-like mobile robot. *Robotics and Autonomous Systems*, 39(2):73–86, 5/31 2002.

[6] W. Xi and J. S. Baras. Mpc based motion control of car-like vehicle swarms. In *Control and Automation, 2007. MED'07. Mediterranean Conference on*, pages 1–6, 2007.

[7] D. Gu and H. Hu. Receding horizon tracking control of wheeled mobile robots. *IEEE Transactions on Control Systems Technology*, 14(4):743, 2006.

[8] Y. Zhu and U. Ozguner. Constrained model predictive control for nonholonomic vehicle regulation problem. In *Proceedings of the 17th IFAC World Congress*, 2008.

[9] 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*, 2004.

[10] Gregor Klanar and Igor krjanc. Tracking-error model-based predictive control for mobile robots in real time. *Robotics and Autonomous Systems*, 55(6):460–469, 6/30 2007.

[11] K. Kanjanawanishkul and A. Zell. Path following for an omnidirectional mobile robot based on model predictive control. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3341–3346, May 2009.

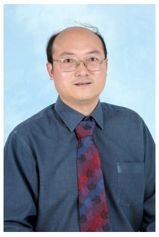
[12] D. Q. [Reference to Mayne], J. B. [Reference to Rawlings], C. V. [Reference to Rao], and P. O. M. [Reference to Scokaert]. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000.

[13] Eva Gyurkovics and Ahmed M. Elaiw. Stabilization of sampled-data nonlinear systems by receding horizon control via discrete-time approximations. *Automatica*, 40(12):2017 – 2028, 2004.

[14] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for a non-holonomic mobile robot. pages 1236–1241 vol.3, 1991. ID: 1.

[15] W.-H. Chen, D.J. Ballance, and J. O'Reilly. Model predictive control of nonlinear systems: computational burden and stability. *Control Theory and Applications, IEE Proceedings -*, 147(4):387–394, Jul 2000.

Cunjia Liu received a B. Eng in Detection, Guidance and Control technology (2005), a M. Eng in Guidance, Navigation and Control (2008) from Beijing University of Aeronautics and Astronautics, Beijing, China. He currently is a PhD student in the Department of Aeronautical and Automotive Engineering at Loughborough University, UK. His research interests include the optimization based control, flight control and path planning of Unmanned Aerial Vehicles.



Wen-Hua Chen received his M. Sc and Ph. D. degrees from Department of Automatic Control at Northeast University, China, in 1989 and 1991, respectively. From 1991 to 1996, he was a lecturer in Department of Automatic Control at Nanjing University of Aeronautics and Astronautics, China. He held a research position and then a lectureship in control engineering in Center for Systems and Control at University of Glasgow, UK, from 1997 to 2000. He currently holds a senior lectureship in flight control systems in Department of Aeronautical and Automotive Engineering at Loughborough University, UK.

He has published one book and more than 100 papers on journals and conferences. His research interests include the development of advanced control strategies and their applications in aerospace engineering.

John Andrews is the Royal Academy of Engineering Professor of Infrastructure Asset Management at the Nottingham Transportation Engineering Centre (NTEC) at University of Nottingham. Prior to this he spent 20 years at loughborough University. The prime focus of his research has been on methods for predicting system reliability in terms of the component failure probabilities and a representation of the system structure. Much of this work has concentrated on the Fault Tree technique and the use of the Binary Decision Diagrams (BDDs) as an efficient and accurate solution method.

He is the author of around 200 research papers on this topic. John is Founding Editor of the Journal of Risk and Reliability (part O of the IMechE Proceedings). He is also a member of the Editorial Boards for Reliability Engineering and System Safety, and Quality and Reliability Engineering International.