Case study of the Cyber-Physical System design: *taking race car as an example*

Nan LI[1], Eric Goubault[2],
Laurent Pautet[1], Sylvie Putot[2]

[1] LTCI, Télécom Paris,
Institut Polytechnique de Paris
[2] LIX, CNRS, Ecole Polytechnique,
Institut Polytechnique de Paris

Réunion AID/CIEDS, le 12 octobre 2021

Background and motivation

- For a Cyber-Physical System, a **predictive controller** is useful for ensuring the safety of the system (in terms of the satisfaction of the constraints).
- However, similar to other sophisticated tasks, it is usually computationally **costly** and occupies non-ignorable CPU/GPU resources.
- Different tasks share the same calculation reource. We want to guarantee the **timing correctness** especially for hard deadline tasks and meanwhile maintain engouth **quality of service** (QoS).
- A **case-study** is performed on a 1:10 racecar for understanding the problem in reality.

Predictive controller

The objective in race car case:

- Achieve the best lap time.
- Ensure that no-collison happens.
- Ensure that no-violation of vehicle's physical limitation.

Predictive controller

Base-solution:

- We use a spatial Nonlinear Model Predictive Control (NMPC) method, which is proposed in [Verschueren et al., 2016], for time-optimal racing in a curvilinear coordinate system.

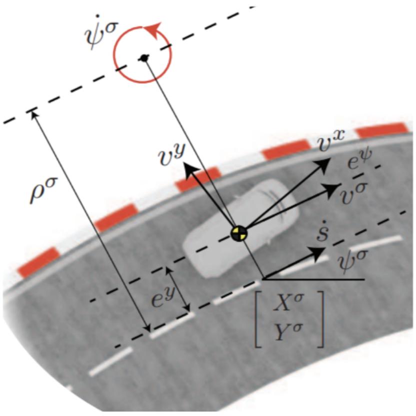


Figure: Curvilinear coordinate system. [Frasch et al., 2013]

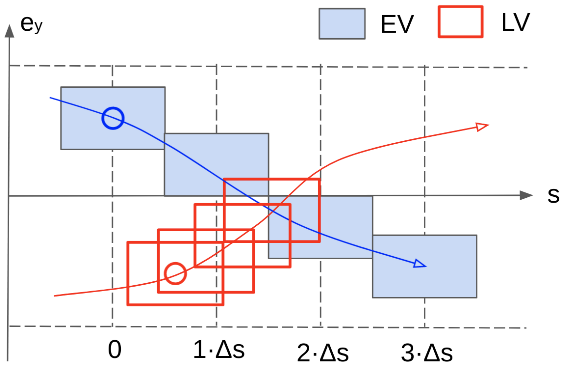


Figure: The prediction horizon in MPC.

Predictive controller

Base-solution:

- The NMPC problem formulation:

$$
\begin{aligned}
&\min_{u_i(s)} \quad t_N \\
s.t. \quad & \xi_{i+1} = f_{\text{RK4}}^{\text{integration}}(\xi_i, u_i),\ i = 0, \dots, N-1 \\
& \xi_i \in [\underline{\xi}, \overline{\xi}],\ i = 1, \dots, N \\
& u_i \in [\underline{u}, \overline{u}],\ i = 0, \dots, N-1, \\
& (\text{collision-avoidance constraint})_i,\ i = 1, \dots, N,
\end{aligned}
\tag{1}
$$

where ξ_i is the state vector, and u_i is the control vector.

Predictive controller

Base-solution:

- In our work:
  - We build an over-approximation for vehicle's shape in the curvilinear coordinate system.
  - We set up the collision-avoidance constraint in a mixed-integer form for two-vehicles head-to-head competition.

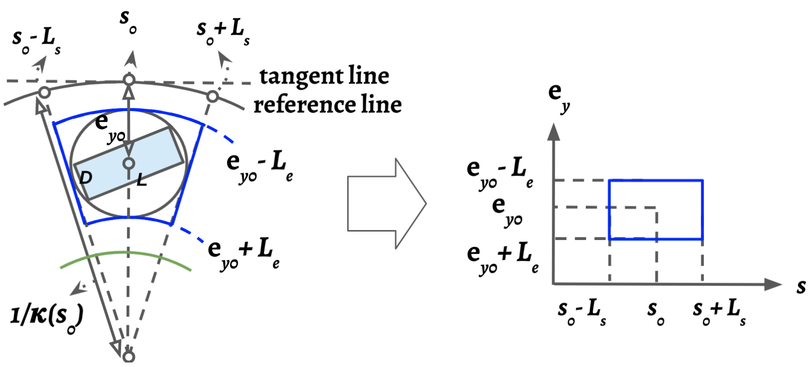


Figure: Vehicles' shape is firstly approximated as a circle and then projected as a set (blue sector) in the curvilinear coordinates system.

Predictive controller

Experimental configuration for simulation:

- NMPC problem is solved by Sequential Quadratic Programs (SQP) framework generated by ACADO Code Generation Tool.
- A wrapper is written to call GUROBI solver for solving MIQP.
- The simulation runs on a Ubuntu 18.04 laptop featured with Intel i7 CPU and 32 GB of RAM.

An typical example in simulation is shown in the next slide.

Predictive controller

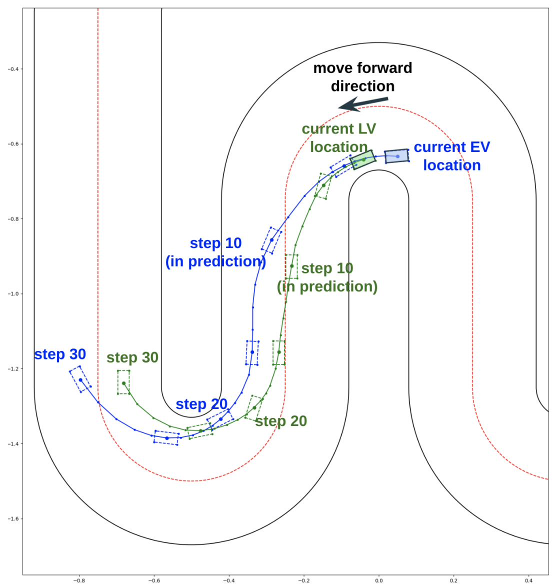


Figure: The trajectory of EV and LV in a typical scenario

In an example of simulation scenario, EV is planning to:

1. follow LV from step 1 to 10
2. overtake LV at the right from step 11 to 19
3. be completely ahead of LV at step 20
4. keep this advantage until step 26, to keep at the left of LV at the last 4 steps

Predictive controller

Simulation result:

Track	Predeiction horizon length	# of collisions (in H2H*)	Mean lap time [s]		Mean calc. time [ms/step]	
			H2H*	Single*	H2H*	Single*
1	15	3/24	**4.942**	4.852	**247**	137
	30	0/24	**4.899**	4.773	**905**	245
2	15	0/45	**10.278**	10.189	**244**	118
	30	0/45	**10.148**	10.064	**832**	205

*H2H = Head-to-head mode, Single = Single car racing mode

- There might be collisions when horizon length is short.
- By increasing horizon length, we obtain better lap time while the calculation time increases too.

Identified problem 1: computational delay

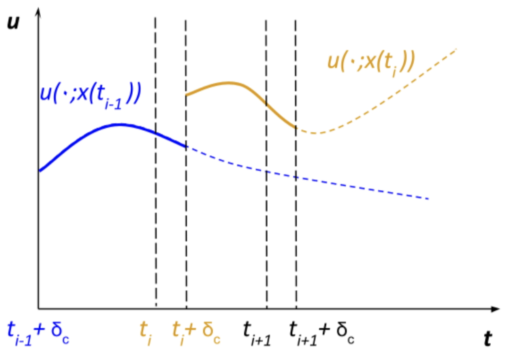
- The computational time is non-ignorable.
- We should take it account into our NMPC model (or into the system design)!

Identified problem 1: computational delay

We define the time model using some notations in [Findeisen and Allgöwer, 2004]:

- The time between two calculation point is defined as $\pi \in [\underline{\pi}, \bar{\pi}]$.
- The maximum calculation time is defined as δ^c.
- The prediction horizon of MPC is defined as T_p.

Identified problem 1: computational delay



We define the time model using some notations in [Findeisen and Allgöwer, 2004]:

- with the following assumptions
  - (A1) $\delta^c < \underline{\pi}$, i.e.: enough calculation time
  - (A2) $T_p > \bar{\pi}$, i.e.: the calculation period is always covered by optimal control planning
  - (A3) the control between time instant t_i and $t_i + \delta^c$ is known and applied: $\bar{u}(\tau; x(t_{i-1})), \tau \in [t_i, t_i + \delta^c)$

Identified problem 1: computational delay

Possible solution:

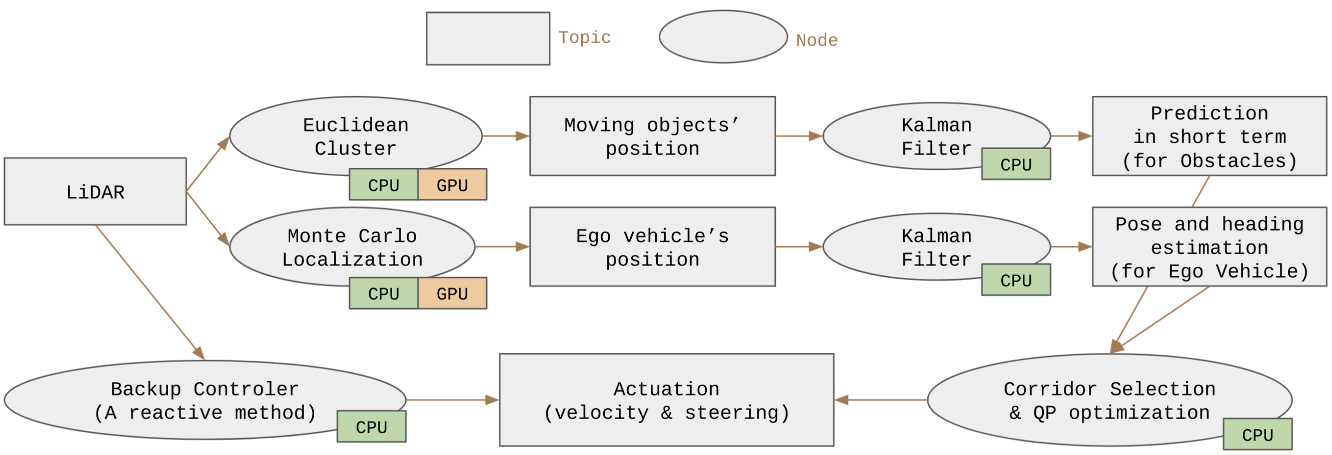
- At time instant t_i, we get the measurement of the system's state $x(t_i)$.
- We estimate the state evaluation at $t_i + \delta^c$ by simulating (integration): $x(t_i) \xrightarrow[\text{duration:}\delta^c]{\text{control:}\bar{u}(\tau;x(t_{i-1})),\tau\in[t_i,t_i+\delta^c)} x(t_i + \delta^c)$.
- We solve the NMPC problem that starts from time instant $t_i + \delta^c$.
- We finally obtains the optimal control sequence: $u^*(\tau; x(t_i)), \tau \in [t_i + \delta^c, t_i + \delta^c + T_p)$.

Identified problem 1: computational delay

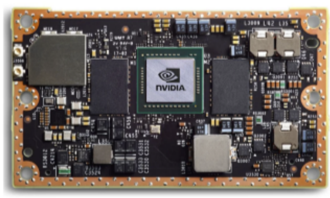
Further work:

- The tasks other than the controller can also build a timing model for their computational delay.
- We can consider all these timing models to make an efficient scheduling policy.

A minimalist architecture for autonomous race car system



Implementation on Jetson TX2



- Jetson TX2 is featuring 2 multi-cores CPUs (Dual-Core NVIDIA Denver 2 64-Bit CPU and Quad-Core ARM Cortex-A57) + 1 GPU (256-core NVIDIA Pascal architecture).
- The controller and all other algorithms will run within Robot Operation System (ROS) under Ubuntu 18.04.

Identified problem 2: scheduling problem

- Different tasks have different criticality levels.
- Without explicit scheduling policy, tasks at high-criticality level may be delayed by low-criticality ones or by other tasks running in the background of Ubuntu.

Identified problem 2: scheduling problem

Possible solution (on going):

- Use the real-time patch for Linux (PREEMPT_RT) to make the base OS support preemptive scheduling.
- Explicitly assign tasks in control loop to high priority.
- Explicitly assign high criticality task to dedicated core.
- Implement a proper scheduling algorithm.

Conclusions

- The NMPC controller provides the precise predictive control under explicit constraints while its computational cost is high. We need to model this computational delay to make the system works properly.
- Scheduling algorithm is needed to ensure the timing correctness of the system especially on a multi-core platform.

Thanks for your listening!

Findeisen, R. and Allgöwer, F. (2004).
Computational delay in nonlinear model predictive control.
IFAC Proceedings Volumes, 37(1):427–432.

Frasch, J. V., Gray, A., Zanon, M., Ferreau, H. J., Sager, S., Borrelli, F., and Diehl, M. (2013).
An auto-generated nonlinear mpc algorithm for real-time obstacle avoidance of ground vehicles.
In *2013 ECC*, pages 4136–4141. IEEE.

Verschueren, R., Zanon, M., Quirynen, R., and Diehl, M. (2016).
Time-optimal race car driving using an online exact hessian based nonlinear mpc algorithm.
In *2016 ECC*, pages 141–147. IEEE.