Royal Society Short Industry Fellowship

Dr Qian Lu

Ranked No.13 UK University
Guardian University Guide 2019

University of the Year for Student Experience
The Times and Sunday Times Good University Guide 2019

2nd in UK for Teaching Excellence (TEF)
Times Higher Education metrics ranking 2017 - Gold winner

Queen's Award for Enterprise
International Trade 2015

Content

- Progress Update From 14/01/23 to: 03/02/23
- Next Steps
- Other activities

A summary of the research
Project Plan

	M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12	...	M24	Visit Plan (24 visits budget)
WP1 Requirement and Test Plan															4
1.1 Requirement defination															
1.1.1 Requirement elicitation															
1.1.2 Vehicle test observation															
1.1.3 Requirement specification															
1.2 Test plan															
1.2.1 Simulation testing scenarios															
1.2.2 Real world testing scenarios															
WP2: Control algorithm design															4
2.1 MPC design															
2.2 EMPC design															
WP3: Simulation Study															4
3.1 Simulation Test Setup (Carmaker?)															
3.2 Algorithm implementation (in ASLAN)															
3.2.1 upgrade to ROS2															
3.2.2 MPC implementation															
3.2.3 EMPC implementation															
3.3 Validation in Simulation															
WP4: Vehicle study															9
4.1 Risk assessment for proving ground testing															
4.2 Streetdrone vehicle integration															
4.2.1 Upgrade to ROS2															
4.2.2 MPC integration															
4.2.3 EMPC integration															
4.4 PG testing at ASSURED CAV test track															
WP5: Road Trial															2
4.1 Risk assessment for public road testing															
4.2 Public road trial															
WP6															2
5.1 Dissemination activities (write paper, seminars)															
5.2 Accumulated impact report															
															Appoxi: 2 visits/month
Note: items highlighted in red need support from SD															

Link to the project plan document

Advantages of using MPC for motion control of CAV

- Model predictive control (MPC) involves solving an optimization problem on demand and using the initial parts of the generated solution to plan or control the system for the next few time steps.
- This technique is good for determining control inputs as it is aware of the dynamics of the modelled system, and thus can choose a control action that is optimal for not just the immediate state, but to also set up future states for success.

Advantages of using MPC for motion control of CAV

- Model Predictive Control (MPC) has several advantages over pure pursuit when it comes to controlling autonomous vehicles. These include:

1. Handling of constraints: MPC can handle constraints on the vehicle's states, such as its speed and acceleration, as well as constraints on the control inputs, such as steering and throttle. Pure pursuit, on the other hand, does not have the ability to handle constraints.
2. Handling of dynamic environments: MPC can handle dynamic environments, such as other vehicles or obstacles, by predicting their future states and adjusting the control inputs accordingly. Pure pursuit, on the other hand, assumes that the environment is static and does not account for potential changes in the future.
3. Flexibility in choosing the reference trajectory: MPC allows for flexibility in choosing the reference trajectory for the vehicle to follow. Pure pursuit, on the other hand, requires that the reference trajectory be a point or a set of points.
4. Handling of nonlinear systems: MPC can handle nonlinear systems, such as those that describe the dynamics of an autonomous vehicle, by using a linearized model of the system in the optimization problem. Pure pursuit is typically limited to linear systems.
5. Handling of multi-objective: MPC can handle multiple objectives, such as minimizing fuel consumption while ensuring safety and comfort. Pure pursuit, on the other hand, typically has a single objective, such as minimizing the distance to a reference point.

Limitations of Online MPC

- Solving optimization problems in real-time

- Lack of reliable real-time optimisation solvers meeting safety requirements of Level 4 autonomous vehicle.

How to resolve reliable issue of online MPC

- The real-time issue in online MPC can be resolved in several ways:

1. Using a fast optimization algorithm: Online MPC requires solving a optimization problem in real-time, which can be computationally demanding. To overcome this, companies may use fast optimization algorithms, such as Interior Point Method (IPM) or Sequential Quadratic Programming (SQP), which can solve the optimization problem quickly.
2. Using a simplified model of the system: Online MPC requires a model of the system that describes its dynamics. To reduce the computational burden, companies may use a simplified model that captures the essential dynamics of the system but is computationally less demanding.
3. Using parallel computation: To reduce the computation time, companies may use parallel computation, such as Graphic Processing Units (GPUs) or Field-Programmable Gate Array (FPGA) to speed up the computation process.
4. Model Predictive Control with receding horizon: Using receding horizon MPC, it allows the optimization problem to be solved over a shorter time horizon, which reduces the computation time and makes the control system more responsive.
5. **Pre-computing and storing the solutions of the optimization problem**: A company could pre-compute the solutions of the optimization problem for a range of possible initial conditions, and store them in a lookup table. This allows the control system to quickly look up the solution for the current initial condition, rather than having to solve the optimization problem in real-time.

Other potential solutions for resolving the reliability issue of online MPC

- There are several optimization solvers that are capable of solving the Model Predictive Control (MPC) problem in real-time and can meet the safety requirements of level 4 autonomous vehicles. These solvers are based on different optimization algorithms and can be divided into three main categories:

1. Interior Point Methods (IPM): these are optimization algorithms that are based on the theory of convex optimization and are known for their fast convergence and robustness to ill-conditioned problems. Interior Point Methods are suitable for problems with a large number of variables and constraints and are widely used in MPC for autonomous vehicles.
2. Sequential Quadratic Programming (SQP): these are optimization algorithms that are based on the theory of nonlinear optimization and are known for their ability to handle nonlinear constraints and objectives. SQP methods are suitable for problems with a moderate number of variables and constraints and are widely used in MPC for autonomous vehicles.
3. Real-time Iterative Linear Quadratic Regulator (iLQR): these are optimization algorithms that are based on the theory of optimal control and are known for their ability to handle nonlinear dynamics and constraints. iLQR methods are suitable for problems with a moderate number of variables and constraints and are widely used in MPC for autonomous vehicles.

Requirements of vehicle motion control

- Requirement relating to safe navigation at low speed (<=30mph) and high speed(>30mph)
  - The controller's responsibility for safe maneuvering then boils down to the following use case:
    - Given a set of inputs, the controller must be able to achieve the specified sequence of dynamic states with a bounded level of error
    - More specifically, given that a reference trajectory is a sequence of time-indexed dynamic states (e.g. poses, velocities, accelerations), the controller must be able to generate a series of actuation commands such that at each time step, the vehicle's actual dynamic state is within some error bound with respect to the reference dynamic state.
    - The dynamic state is assumed to have at least the following properties:
      - Position (at least x, y)
      - Velocity (at least longitudinal and lateral)

Runtime evaluation

- Method 1:

start_time = clock;

my_computation;

runtime = etime(clock, start_time)

The time is different every time, sometime online is higher than explicit, sometime not. Need to evaluate based on fixed allocated resources.

- Method 2:

https://uk.mathworks.com/matlabcentral/answers/1796810-mpc-toolbox-computation-time

Reduce number of regions

- A number of factors influence the number of regions
  - Increasing the sampling time can reduce the regions
  - Reduce the prediction horizon can reduce the regions
  - Low number of constraints results in low number of regions
  - …

Start with designing a lateral control for path tracking

- Lateral control only generates steering angle command to regulate y and psi
- Explore Autoware.auto implementation of this

Autoware.Auto MPC implementation for lateral control

- Three different models
  - Kinematic with no delay
  - Kinematic model with 1-st order delay
  - Dynamics model
  - Reference for the models

  https://www.ri.cmu.edu/pub_files/2009/2/Automatic_Steering_Methods_for_Autonomous_Automobile_Path_Tracking.pdf

Autoware.Auto MPC implementation for lateral control Kinematic with no delay

- Kinematic model with no delay (linearised around δr)
  - States: ey, $e\theta$, input: δ
  - $A = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix}$
  - $B = \begin{bmatrix} 0 \\ \frac{v}{l * \cos^2(\delta r)} \end{bmatrix}$
  - $C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
  - $W = \begin{bmatrix} 0 \\ -\frac{v}{l * \cos^2(\delta r)} \delta r \end{bmatrix}$

- The original nonlinear model in vehicle coordinate

$$\dot{y} = v * \sin(\theta)$$

$$\dot{\theta} = \frac{v}{l} * \tan(\delta)$$

- Uses Eular forward discretisation method to convert it into discrete model
- Different from the model in Maurovic 2011 which controlling both longitudinal and lateral and linearisation is conducted around wr instead of δr

Autoware.Auto MPC implementation for lateral control Kinematic with with 1st order delay

- Kinematic model with no delay (linearised around δr)
  - States: ey, $e\theta$, $\dot{e\theta}$, input: δ
  - $A = \begin{bmatrix} 0 & v & 0 \\ 0 & 0 & \frac{v}{l * \cos^2(\delta r)} \\ 0 & 0 & -\frac{1}{\tau} \end{bmatrix}$
  - $B = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{\tau} \end{bmatrix}$
  - $C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
  - $W = \begin{bmatrix} 0 \\ -v * \mathrm{K} + \frac{v}{l} * (\tan(\delta r) - \frac{\delta r}{\cos^2(\delta r)}) \\ 0 \end{bmatrix}$

- Uses bilinear discretisation approach to convert it into a discrete model

Autoware.Auto MPC implementation for lateral control Dynamic model

- States: ey, $\dot{ey}$, $e\theta$, $\dot{e\theta}$, input: δ
  - $A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{cf+cr}{m*v} & \frac{cf+cr}{m} & -\frac{lf*cf-lr*cr}{m*v} \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{lf*cf-lr*cr}{Iz*v} & \frac{lf*cf-lr*cr}{Iz} & -\frac{lf^2*cf+lr^2*cr}{Iz*v} \end{bmatrix}$
  - $B = \begin{bmatrix} 0 \\ \frac{cf}{m} \\ 0 \\ \frac{lf*cf}{Iz} \end{bmatrix}$
  - $C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
  - $W = \begin{bmatrix} 0 \\ -\frac{lf*cf-lr*cr}{m*v} - v \\ 0 \\ -\frac{lf^2*cf+lr^2*cr}{Iz*v} \end{bmatrix} * v * \mathrm{K}$

- Uses bilinear discretisation approach to convert it into a discrete model
- This model is same as paper Lee 2018

Autoware.Auto MPC implementation for lateral control Constraints

- Option1: No constraints
- Option 2: Constraints on steering <+/-35degree

Autoware.Auto MPC implementation for lateral control QP Solver

- Unconstraint-fast

- QSQP for constraints
  - Link for QSQP library; https://osqp.org/

Validate above control scheme 3 in MATLAB/Simulink

- Control Scheme 3 based on the dynamic model
- Three controllers
  - For speed = 10m/s, Ts = 0.1s, Prediction Horizon = 4, other design parameters are same for all controllers
    - Controller 1: An Online MPC using QSPQ solver
    - Controller 2: An Explicit MPC solved using MPT3 toolbox
    - Controller 3: An LQR controller designed using MATLAB

  - Performance are identical between Controller 1 & 2，very similar between Controller 1 & 3
  - Number of regions:
    - Constraints on all states: 441
    - Constraints on lateral error and heading error only: 283
  - Runtime (for 441 regions)
    - Controller 1: average 0.15s for 100 iterations simulation
    - Controller 2: average 0.07s for 100 iteration simulation

Next Steps

- MPC & EMPC implementation in ASLAN(open)
- Evaluate the computing time and storage of EMPC and MPC after implementation in ASLAN
- Possibly explore methods to reduce the number of region of EMPC (to reduce compute time and storage)
- Explore the approach for developing an EMPC for a range of vehicle speed
    - EMPC for LPV system: https://yalmip.github.io/example/explicitlpvampc/

Other Activities


Coventry
University
Thank you