MATLAB EXPO 2019

Develop and Test Vehicle Controllers for ADAS/Automated Driving Applications through System Simulation

Abhisek Roy
Highway Traffic Jam Assist

- It helps drivers to follow the preceding vehicle automatically with a predefined time interval in a dense traffic condition
  ... while controlling steering for keeping current lane.

- Partial/conditional automation at level 2/3
  - Speed limit < 60~65 km/h
  - Dense traffic condition in highway

Longitudinal control with ACC with stop & go

Lateral control with lane following
Challenges

- Wide variety of scenarios and difficult to gather real data
- Complex interplay between multiple sensors
- Incorporate models of right fidelity for various system components
- Costly and hazardous in-vehicle testing
Agenda

- Design model-predictive control-based vehicle controllers

- Run close-loop simulation with synthetic scenarios and test sensor fusion and control algorithms at a model level

- Improve simulation fidelity by incorporating detailed vehicle models and integrating with Unreal gaming engine
Performance Requirements: Longitudinal Control

- **Ego velocity control**: $v \leq v_{\text{set}}$
  where, $v$: ego velocity, $v_{\text{set}}$: set velocity

- **Time gap control**: $\tau \geq \tau_{\text{min}}$
  where, $\tau = \frac{c}{v}$: time gap = 1.5 .. 2.2 sec
  $\tau_{\text{min}}$: min time gap = 0.8 sec

- **Operation limits**
  - Minimum operational speed, $v_{\text{min}} = 5\text{m/s}$
  - Average automatic deceleration $\leq 3.5\text{ m/s}^2$ (average over 2s)
  - Average automatic acceleration $\leq 2.0\text{ m/s}^2$
Performance Requirements: Lateral Control

- Vehicle should follow the lane center with allowable lateral deviation.

\[ |(d_{left} + d_{right})/2| \leq e_{max} \]

where,
- \( d_{left} \): lateral offset of left lane w.r.t. ego car
- \( d_{right} \): lateral offset of right lane w.r.t. ego car
- \( e_{max} \): allowable lateral deviation

For example, \( e_{max} = (\text{LaneWidth} - \text{VehicleWidth})/2 = (3.6 - 1.8)/2 = 0.9 \text{ m} \)
What is model predictive control (MPC)?

- **Multi-variable control** strategy leveraging an internal model to predict plant behavior in the near future.

- **Optimizes** for the current timeslot while keeping future timeslots in account.

- **Suitable** for our problem statement:
  - Handles MIMO systems with coupling
  - Handles constraints
  - Has preview capabilities
How can MPC be applied to Highway Traffic Jam Assist?

- References
- Measured disturbances
- Optimizer
- Plant Model
- MPC controller
- Measured outputs
- Manipulated variables
- Ego Vehicle
How can MPC be applied to Highway Traffic Jam Assist?

**Measured outputs**
- Relative distance ($D_{\text{relative}}$)
- Ego velocity ($V_{ego}$)
- Lateral deviation ($E_{lateral}$)
- Relative yaw angle ($E_{yaw}$)

**Manipulated variables**
- Acceleration ($a$)
- Steering angle ($\delta$)

**Measured disturbances**
- MIO velocity ($V_{mio}$)
- Previewed road curvature ($\rho$)

**References**
- Ego velocity set point ($V_{set}$)
- Target lateral deviation (=0)

**Minimize:**

$$w_1|V_{ego} - V_{set}|^2 + w_2|E_{lateral}|^2$$

**Subject to:**

$$D_{\text{relative}} \geq D_{\text{safe}}$$
$$a_{\text{min}} \leq a \leq a_{\text{max}}$$
$$\delta_{\text{min}} \leq \delta \leq \delta_{\text{max}}$$
Control Algorithm
Control Algorithm

Path following control (PFC) system (mask) (link)

Keep the ego vehicle traveling along the center of a straight or curved road, track a set velocity and maintain a safe distance from a lead vehicle by adjusting the longitudinal acceleration and the front steering angle of the ego vehicle.

- **Parameters**
- **Controller**
- **Block**

**Optimization**
- ⊗ Use suboptimal solution

**Data Type**
- ⊗ double
- ○ single

**Optional Inputs**
- ⊗ Use external signal to enable or disable optimization
- ⊗ Use external control signal for bumpless transfer between PFC and other controllers

**Customization**
To customize your controller, generate an PFC subsystem from this block and modify it. The controller configuration data is exported as a structure in the MATLAB workspace.

Create PFC subsystem
Agenda

- Design model-predictive control-based vehicle controllers

- Run close-loop simulation with synthetic scenarios and test sensor fusion and control algorithms at a model level

- Improve simulation fidelity with gaming engine integration, vehicle dynamics modelling, and automated scenario creation from recorded data
Architecture for Traffic Jam Assist Controller

Control Algorithm

- Radar Detection
- Vision Detection
- Lane Detection

Sensor Fusion and Tracking → Find Lead Car → Model Predictive Control (MPC) → Preview Curvature → Estimate Lane Center

Vehicle and Environment

- Ego Vehicle Dynamics
- Radar, Vision, Lane Detection Generator

Driving Scenario

- Acceleration (Longitudinal)
- Steering (Lateral)
- Ego Longitudinal Velocity
Develop and Test Vehicle Controller
Traffic Jam Assist

Control Algorithm

- Radar Detection
- Vision Detection
- Lane Detection

- Sensor Fusion and Tracking
- Find Lead Car
- Model Predictive Control (MPC)
- Acceleration (Longitudinal)
- Steering (Lateral)

Vehicle and Environment

- Ego Vehicle Dynamics
- Ego Longitudinal Velocity
- Radar, Vision, Lane Detection Generator

Driving Scenario

- Estimate Lane Center
- Preview Curvature

- Lane Detection
- Vision Detection

Ego

Model Predictive Control (MPC)
Incorporate Ego Vehicle Dynamics

- Implement a single track 3DOF rigid vehicle body to calculate longitudinal, lateral, and yaw motion.
- Block calculates only lateral forces using the tire slip angles and linear cornering stiffness.
Develop and Test Vehicle Controller
Traffic Jam Assist

Control Algorithm

- Radar Detection
- Vision Detection
- Lane Detection

- Sensor Fusion and Tracking
- Find Lead Car

- Estimate Lane Center
- Preview Curvature

- Model Predictive Control (MPC)

Vehicle and Environment

- Acceleration (Longitudinal)
- Steering (Lateral)

Driving Scenario

- Ego Vehicle Dynamics
- Ego Longitudinal Velocity

- Radar, Vision, Lane Detection Generator
Create Test Scenario using Driving Scenario Designer

Test Description
Lead car cut in and out in curved highway (curvature of road = 1/500 m)

Host car
initial velocity = 20.6m/s
HWT(Headway Time) to lead car = 4sec
HW(Headway) to lead car = ~80m
\(v_{set}\) (set velocity for ego car) = 21.5m/s

Lead Car
Initially, fast moving car (orange) at 19.4m/s
Passing car (yellow) at 19.6m/s cut in the ego path with HWT=2.3s, then cut out

Third Car
Slow moving car (purple) at 11.1m/s in the 2nd lane
Add sensors to test scenario
Control Algorithm

Radar Detection → Sensor Fusion and Tracking → Find Lead Car → Model Predictive Control (MPC) → Estimate Lane Center → Preview Curvature → Vehicle and Environment

Vehicle and Environment

Accelerate (Longitudinal)

Steering (Lateral)

Driving Scenario

Ego Vehicle Dynamics

Ego Longitudinal Velocity

Radar, Vision, Lane Detection Generator

Develop and Test Vehicle Controller
Traffic Jam Assist
Simulation with Simulink Model for Traffic Jam Assist

Test Description
Lead car cut in and out in curved highway (curvature of road = 1/500 m)

Host car
- initial velocity = 20.6m/s
- HWT (Headway Time) to lead car = 4 sec
- HW (Headway) to lead car = ~80m
- v_set (set velocity for ego car) = 21.5m/s

Lead Car
- Initially, fast moving car (orange) at 19.4 m/s
- Passing car (yellow) at 19.6 m/s cut in the ego path with HWT = 2.3s, then cut out

Third Car
- Slow moving car (purple) at 11.1 m/s in the 2nd lane
Simulation with Simulink Model for Traffic Jam Assist

**Test Description**
Lead car cut in and out in curved highway (curvature of road = 1/500 m)

**Host car**
- Initial velocity = 20.6 m/s
- HWT (Headway Time) to lead car = 4 sec
- HW (Headway) to lead car = ~80 m
- \( v_{set} \) (set velocity for ego car) = 21.5 m/s

**Lead Car**
- Initially, fast moving car (orange) at 19.4 m/s
- Passing car (yellow) at 19.6 m/s cut in the ego path with HWT=2.3 s, then cut out

**Third Car**
- Slow moving car (purple) at 11.1 m/s in the 2nd lane

---

**Graphs**
- Ego Velocity
- Time Gap
- Ego Acceleration
- Lateral Deviation
Agenda

- Design model-predictive control-based vehicle controllers
- Run close-loop simulation with synthetic scenarios and test sensor fusion and control algorithms at a model level
- Improve simulation fidelity with gaming engine integration, vehicle dynamics modelling, and automated scenario creation from recorded data
Improve simulation fidelity: Include detailed vehicle dynamics

Vehicle Dynamics Blockset
**Improve simulation fidelity:**
Include detailed vehicle dynamics

<table>
<thead>
<tr>
<th>Vehicle Model</th>
<th>Description</th>
<th>Vehicle Body Degrees-of-Freedom (DOFs)</th>
<th>Wheel DOFs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passenger</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14DOF Vehicle</td>
<td>• Vehicle with four wheels</td>
<td>Six Translational</td>
<td>Rotational</td>
</tr>
<tr>
<td></td>
<td>• Available as model variant in the maneuver reference applications</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>7DOF Vehicle</td>
<td>• Vehicle with four wheels</td>
<td>Three Translational</td>
<td>Rotational</td>
</tr>
<tr>
<td></td>
<td>• Available as model variant in the maneuver reference applications</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3DOF Vehicle</td>
<td>• Vehicle with ideal tire</td>
<td>Three Translational</td>
<td>Rotational</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

**Wheel DOFs**
- Two per wheel - eight total
  - Translational
    - Vertical ✓
  - Rotational
    - Rolling ✓

- One per wheel - four total
  - Rotational
    - Rolling ✓

- None
Improve simulation fidelity: Co-simulate with Unreal Engine
Game Engine Co-Simulation

**Simulink**
- Physics of vehicle
- Initialization of game engine camera

**Unreal Engine**
- Rendering / lighting
- Physics of non-Simulink objects
- Collision detection

---

Vehicle / camera location

Camera image, ground height, ...

Vehicle / camera location

Camera image, ground height, ...

Lane Change Reference Generator

Predictive Driver

Environment

Controllers

Passenger Vehicle

Lane

Car
Develop and Test Vehicle Controller
Traffic Jam Assist: Key takeways

**Design Traffic Jam Assist Controller**

- Create driving scenario
- Synthesize sensor detection
- Include Vehicle Dynamics
- Design sensor fusion algorithm
- Design controller using MPC
Develop and Test Vehicle Controller
Traffic Jam Assist: Next Steps

**Design Traffic Jam Assist Controller**
- Create driving scenario
- Synthesize sensor detection
- Include Vehicle Dynamics
- Design sensor fusion algorithm
- Design controller using MPC

Reference examples to get started:

1. [Lane Following Using Nonlinear Model Predictive Control](#)
2. [Lane Following Control with Sensor Fusion and Lane Detection](#)
3. [Testing a Lane-Following Controller with Simulink Test](#)
Hitachi develops model-predictive controller for adaptive cruise control in traffic jam

Model Predictive Control Approach to Design Practical Adaptive Cruise Control for Traffic Jam

Taku TAKAHAMA 1)  Daisuke AKASAKA 2)

1) Hitachi Automotive Systems, Ltd.
4-7-1 Omia, Atsugi, Kanagawa, 243-8510, Japan (E-mail: tabu.takahama.ts@hitachi-automotive.co.jp)
2) The MathWorks GK
4-15-1 Akasaka, Minato-ku, Tokyo, 107-0052, Japan

The MPC controller was implemented in an embedded microprocessor (Renesas SH-4A, 32-bit processor), we confirmed the processing time of the MPC. The measurement result is shown in Fig. 5, the average time of the ACC function was 1.1 ms. The C-code is automatically generated from a Simulink model using Embedded Coder®.

Hitachi paper published with SAE, Japan 2017
Hitachi also presented at 2017 MathWorks Expo, Japan
Call to action

- Visit the booth!

- Attend the session:
  - Simplifying Requirements-Based Verification with Model-Based Design

- MATLAB Tech Talk:
  - Understanding Model Predictive Control