Smart Regenerative Control Based on Reinforcement Learning Algorithm to Reflect Individual Driver Characteristics

Kyunghan Min*, Gyubin Sim*, Seongju Ahn*, Inseok Park, Jeamyoung Yeon**, Myoungho Sunwoo***

* Department of Automotive Engineering, Hanyang University, Seoul, Korea

** Research & Development Division, Hyundai Motor Company, Hwaseong, Korea

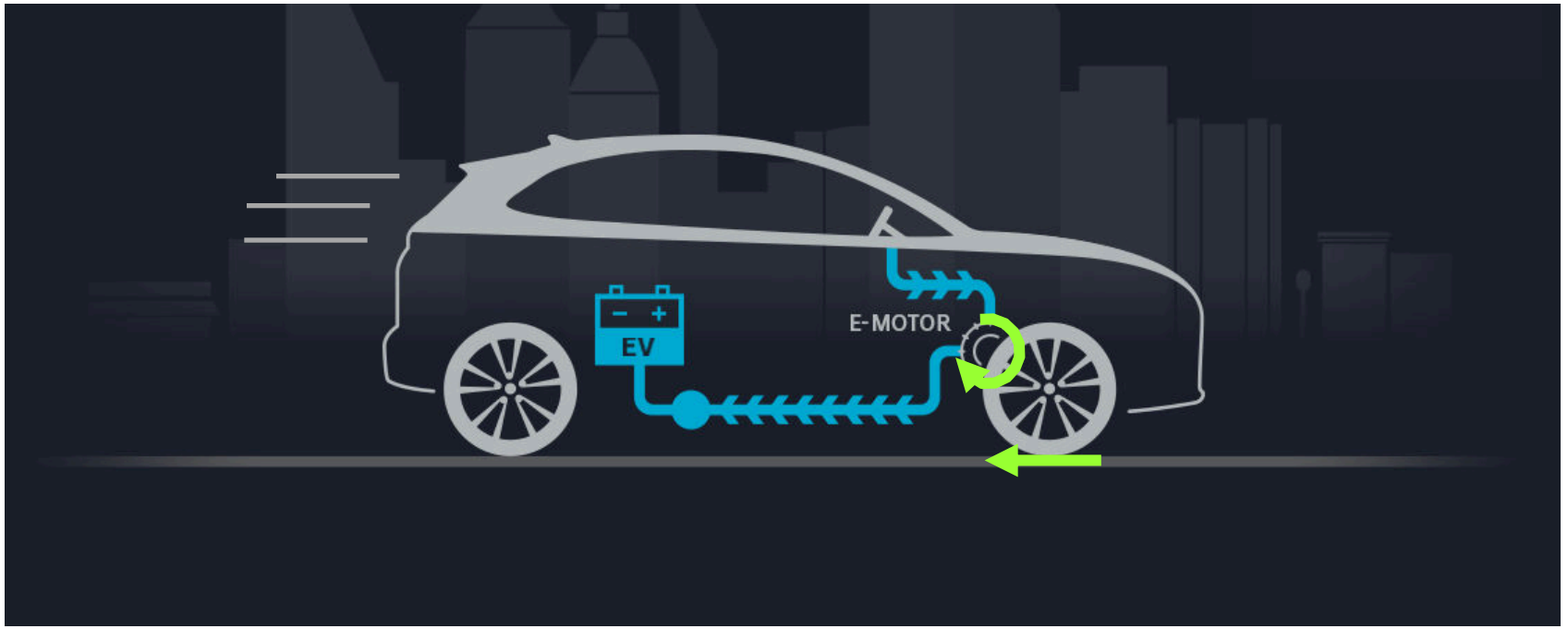
May 20, 2019

Contents

- Introduction
- Electric vehicle model
- Planning algorithm for smart regenerative control
- Validation results
- Summary

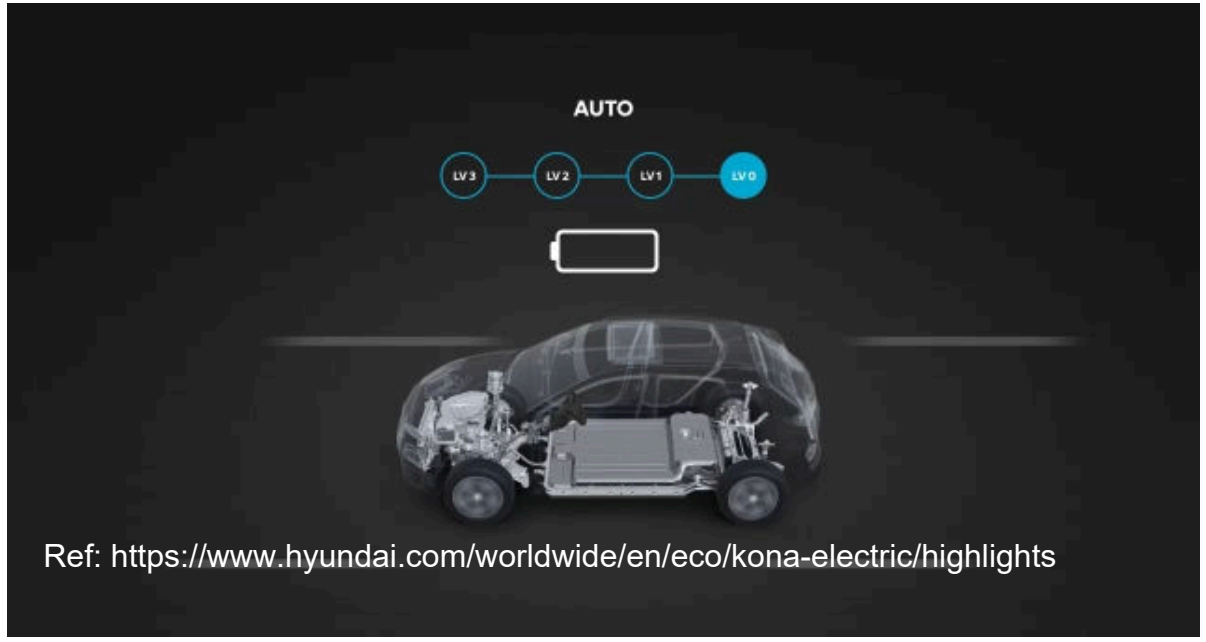
Research background (I)

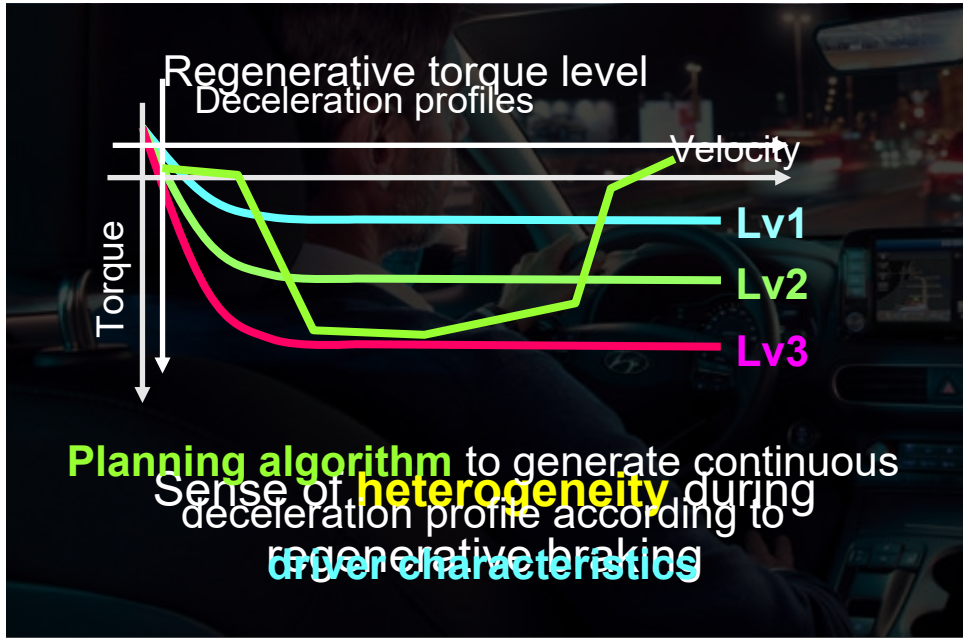
- Regenerative braking system of Electric Vehicles (EV)
  - Braking the vehicle using motor regenerative torque without hydraulic braking



Research background (II)

- Smart regenerative braking system
  - Automatically applying regenerative braking in deceleration conditions
  - Improvements in driving convenience and energy efficiency

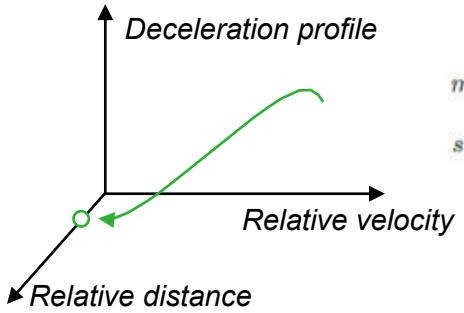




Related works on planning algorithm

Planning with optimization method*

- Generation of a continuous deceleration profile based on the **model predictive control algorithm**



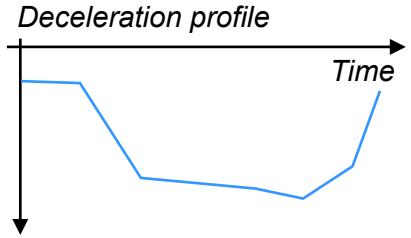
$$\begin{aligned} &\text{minimize} \quad f = \sum_{t=0}^{N} (e_t^T Q e_t + u_t^T R_1 u_t + \Delta u^T R_2 \Delta u) \\ &\text{subject to} \quad X_{t+1} = AX_t + Bu_t, \qquad t = 0, \ldots, N \\ &\qquad u_t \geq u_{min} \ \text{and} \ u_t \leq u_{max}, \qquad t = 0, \ldots, N. \end{aligned}$$

✓ Can respond to various deceleration situations with safety

✗ Cannot reflect the driver characteristics

Planning with intelligent driver model**

- Representation of **individual driver characteristics** based on mathematical equations and explicit model parameters



$$\dot{x_\alpha} = \frac{dx_\alpha}{dt} = v_\alpha$$

$$\dot{v_\alpha} = \frac{dv_\alpha}{dt} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^4 - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right)$$

✓ Can reflect the driver characteristics and deceleration specific profile

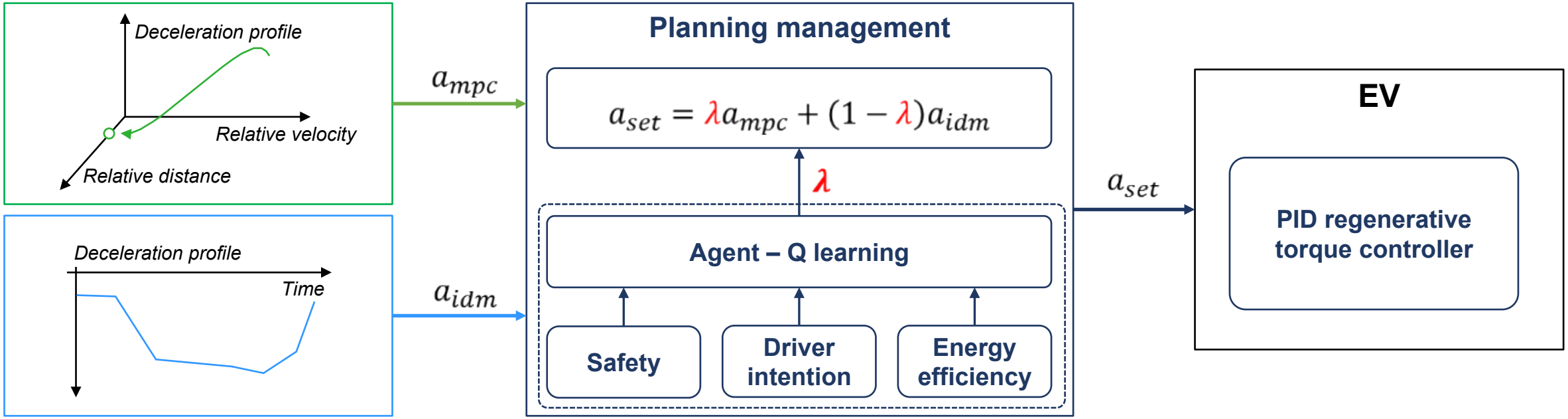
✗ Can be applied only in the modeling conditions

** "On-road Motion Planning with Roadway-based Hierarchical Searching Space" – Wonteak Lim, Myoungho Sunwoo, IEEE-ITS, 2017*

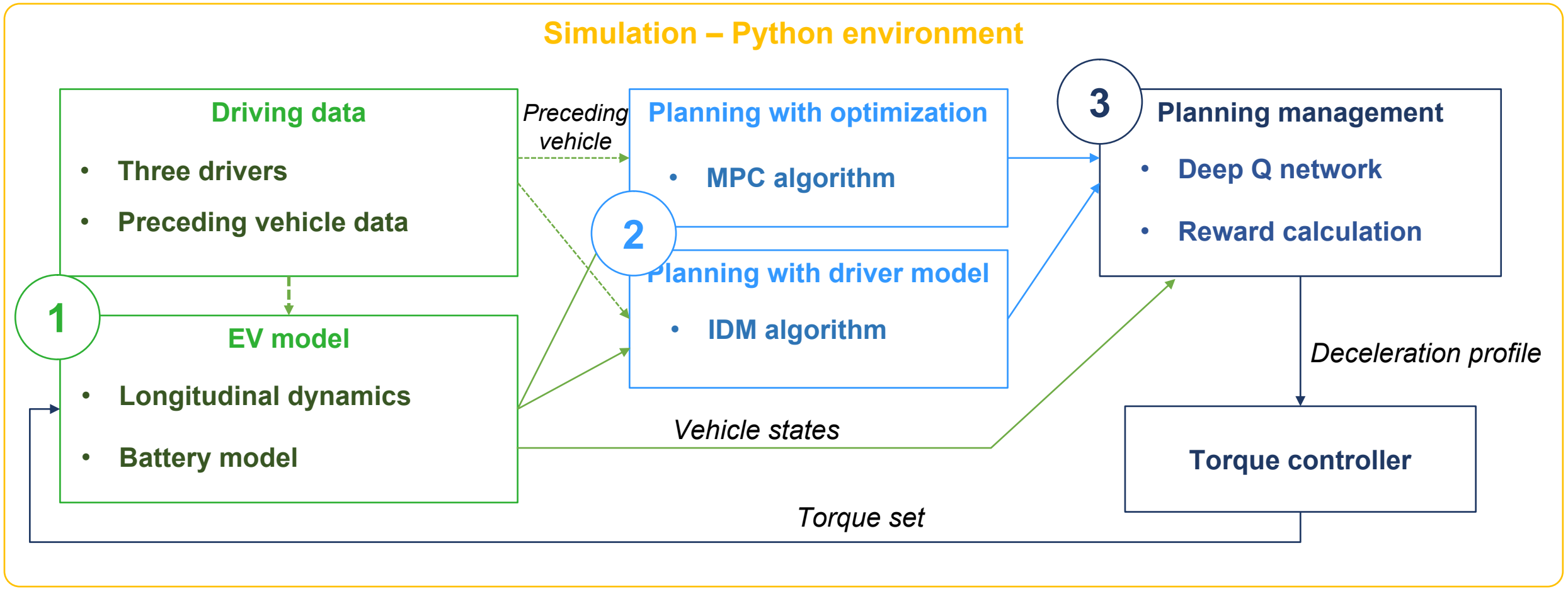
*** "Congested traffic states in empirical observations and microscopic simulations", Martin Treiber, Physical review E, 2000*

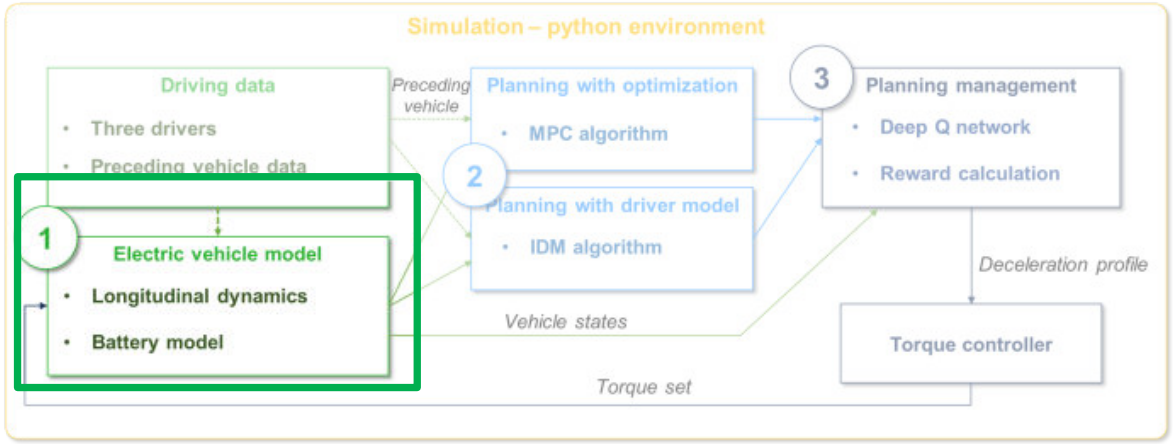
Research objective

- Design of planning management to select an optimal deceleration profile
  - Determination of a weight factor between optimization method and driver model
  - Reinforcement learning algorithm: safety, driver intention, energy efficiency



Algorithm overview

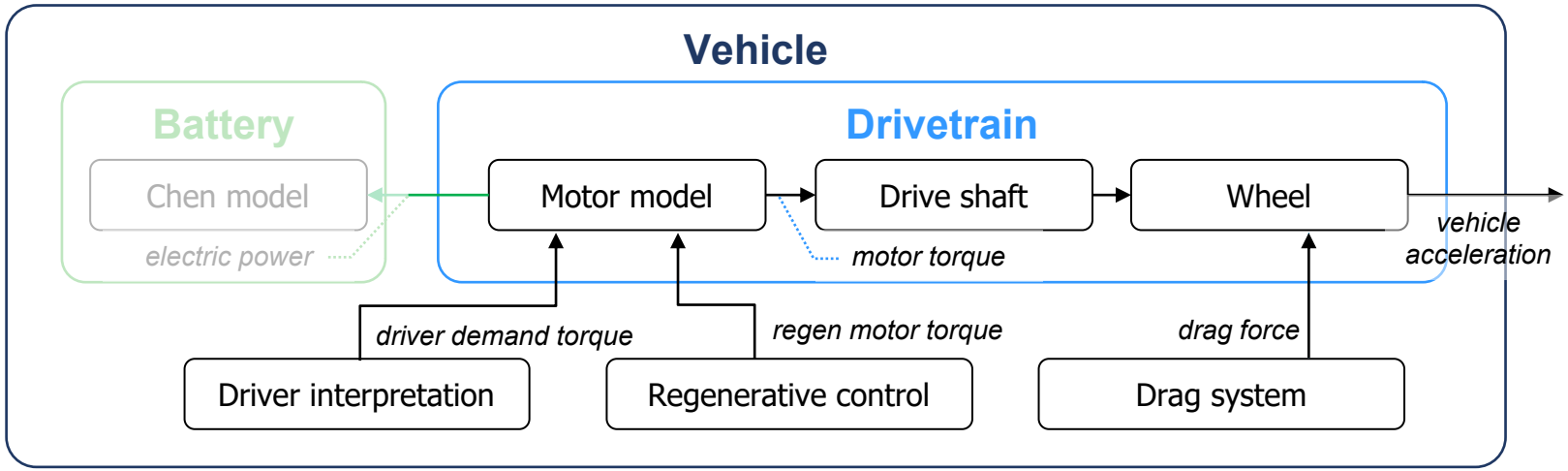




EV model

- Longitudinal dynamics model
- Battery model

Longitudinal dynamics model*



$m = mass$
$\tau = torque$
$v = velocity\ of\ the\ vehicle$
$I = inertia$
$w = wheel$
$s = shaft$
$m = motor$
$\eta_s = drivetrain\ efficiency$
$c_a = rolling\ registance\ coeff.$
$c_b = rolling\ registance\ coeff.$
$c_a = drag\ coeff.$

Driver input

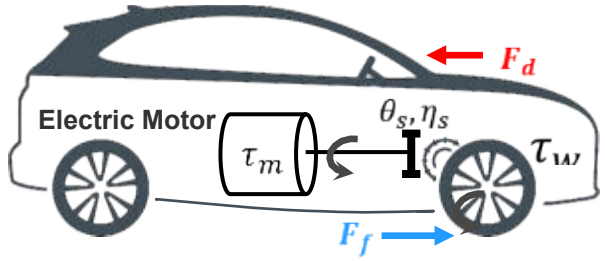
Driving state

Vehicle + Drivetrain

$$a_v = (\theta_s T_m \eta_s / r_w - F_d)/m_v$$

$$m_v = m_e + m_a + 4I_w + \theta I_m + I_s$$

$$F_d = (0.75 c_d v_v^2 + c_a + c_b v_v^2)$$

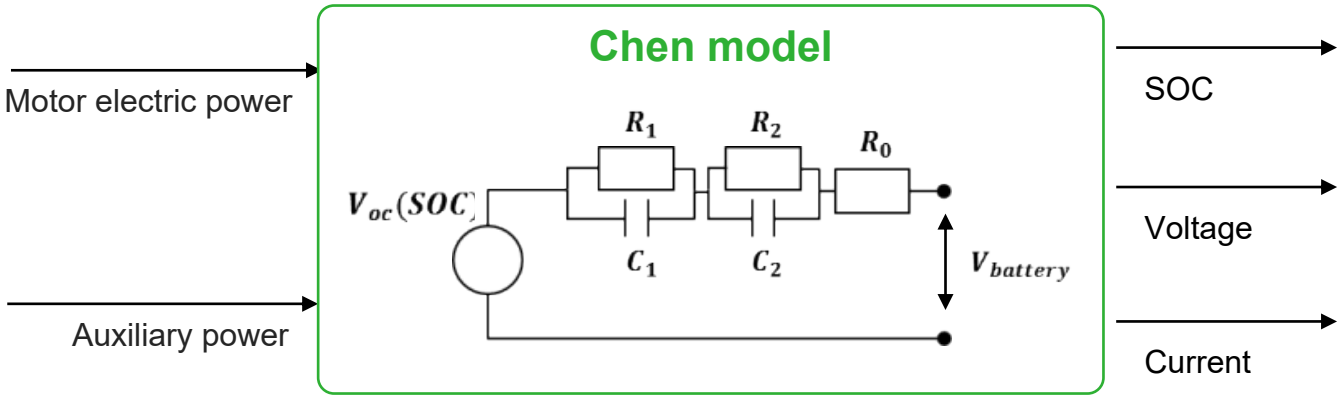


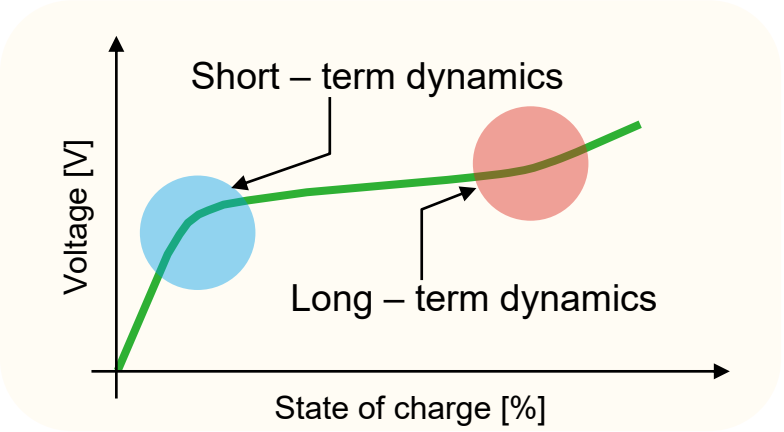
Vehicle acceleration

** "Electric and Hybrid Vehicles - Technologies, Modeling and Control : a Mechatronic Approach" – Amir Khajepour, Saber Fallah, Avesta Goodarzi*

Battery model

- Chen's two-stage battery model*





$$R_0(SOC) = R_{0a}e^{R_{0b}\cdot SOC} + R_{0c}$$

$$C_0(SOC) = C_{0a}e^{C_{0b}\cdot SOC} + C_{0c}$$

$$\vdots$$

$V_{oc} = open\ circuit\ voltage$
$R = register$
$C = capacitor$

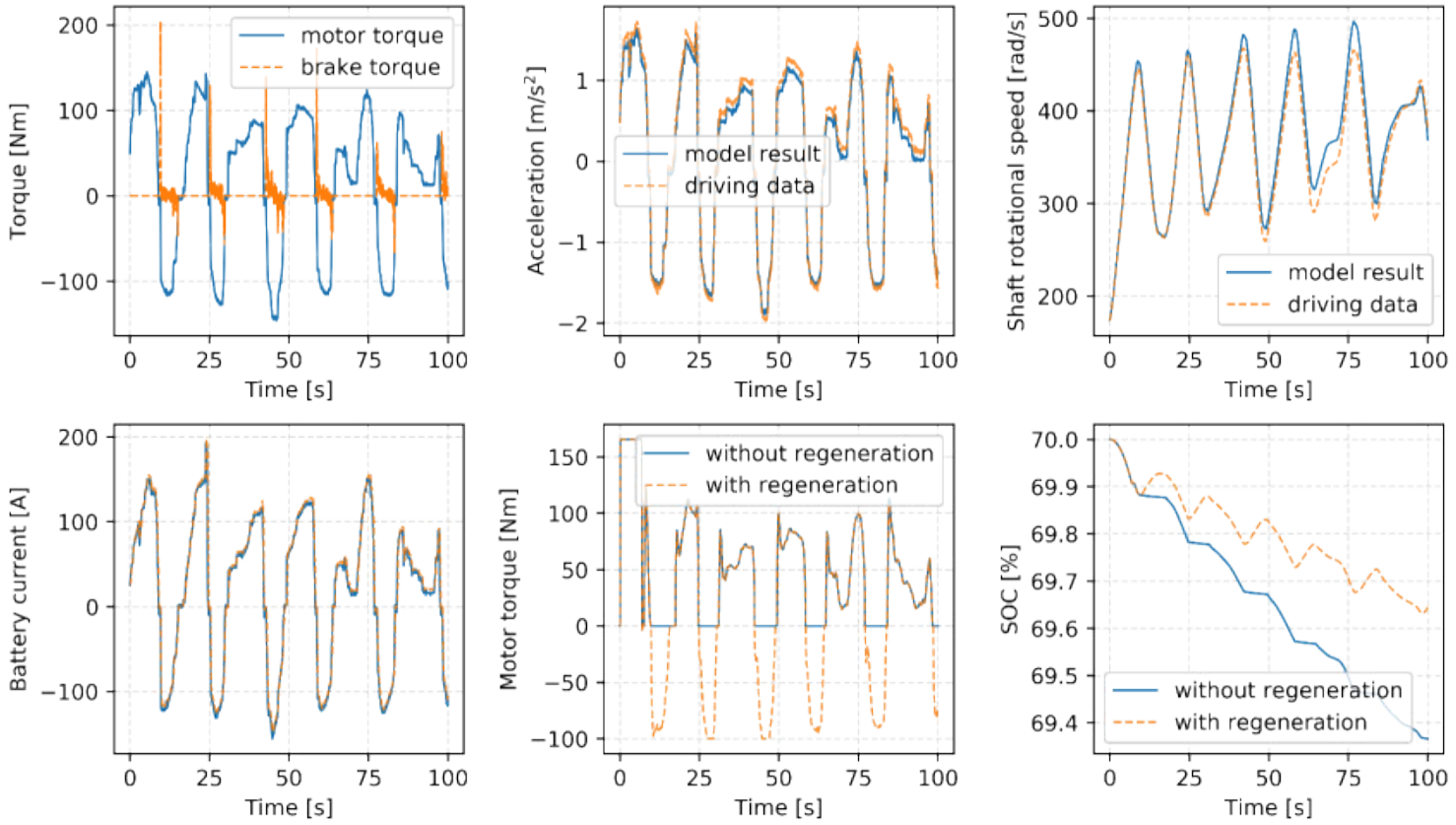
*Chen, M., & Rincon-Mora, G. A. (2006). Accurate electrical battery model capable of predicting runtime and IV performance. IEEE transactions on energy conversion

Modeling results

- Model parameter identification using vehicle driving data

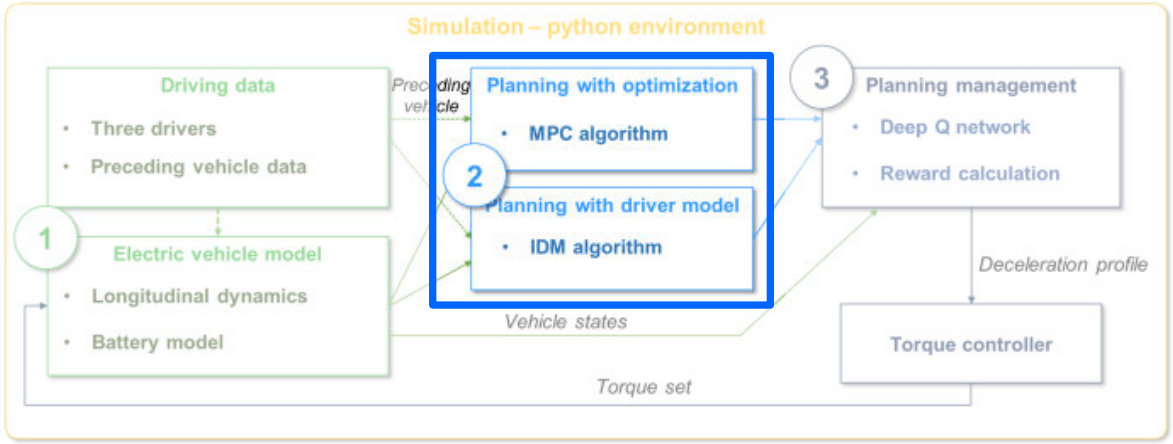
Vehicle - Hyundai KONA EV

Model	RMSE	R^2
Vehicle acceleration	0.11 [m/s²]	0.988
Battery current	13.26 [A]	0.977



Planning algorithm

- Optimization method
- Intelligent driver model



Optimization method-based planning

- Model predictive control algorithm

Model predictive control algorithm

minimize $f = \sum_{t=0}^{N_p} (e_t^T Q e_t^T + u_t^T R u_t)$

subject to $X_{t+1} = AX_t + Bu_t$

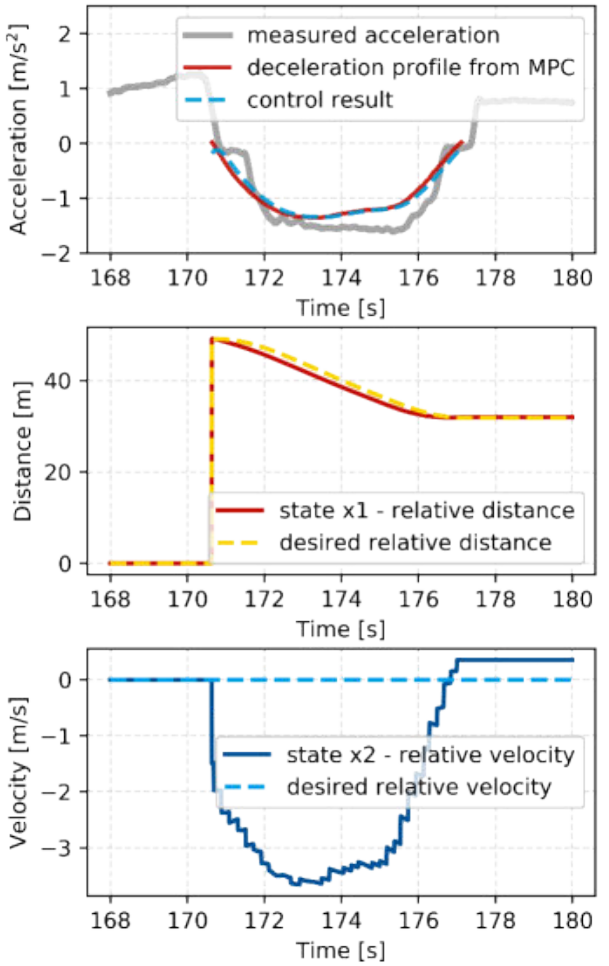
$u_t \geq u_{min}$ *and* $u_t \leq u_{max}$

- e_t: *Error* $(X_r - X)$
- u_t: *Input (Vehicle Acc)*

X	X_r	A	B
$\begin{bmatrix}\Delta d \\ \Delta v\end{bmatrix}$	$\begin{bmatrix}d_r \\ 0\end{bmatrix}$	$\begin{bmatrix}1 & \Delta t \\ 0 & 1\end{bmatrix}$	$\begin{bmatrix}-0.5\Delta t^2 \\ -\Delta t\end{bmatrix}$
State1: Relative distance State2: Relative velocity	Desired relative distance $d_r = TimeGap \times v_{ego}$	Discrete vehicle dynamics	

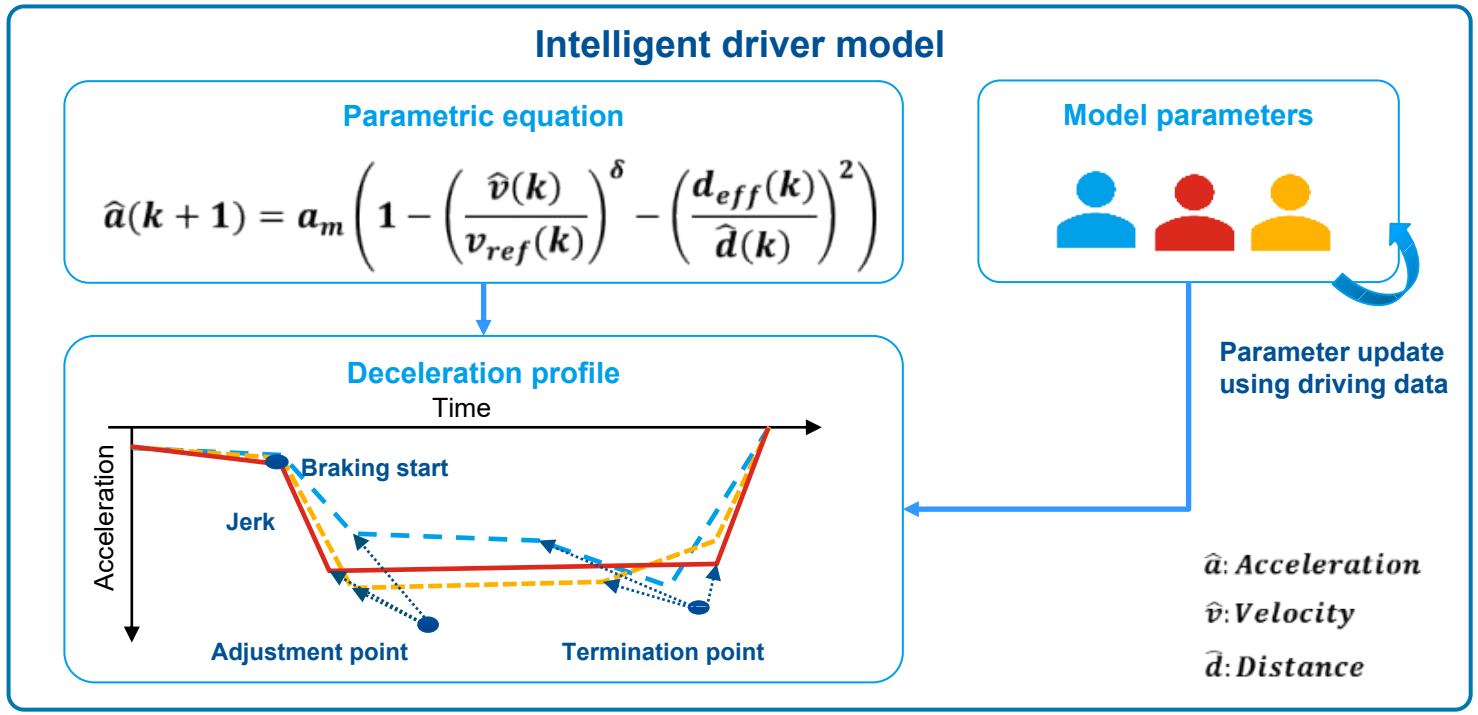
State1: Relative distance
State2: Relative velocity

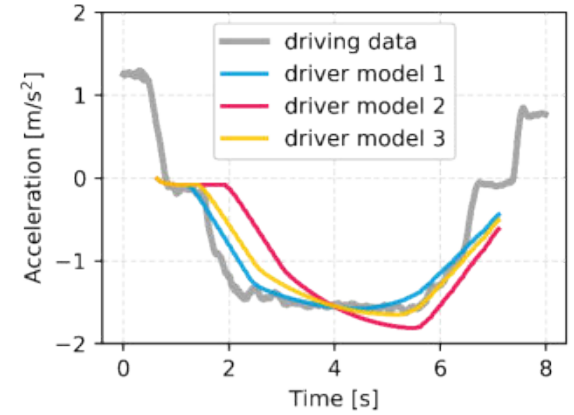
Discrete vehicle dynamics

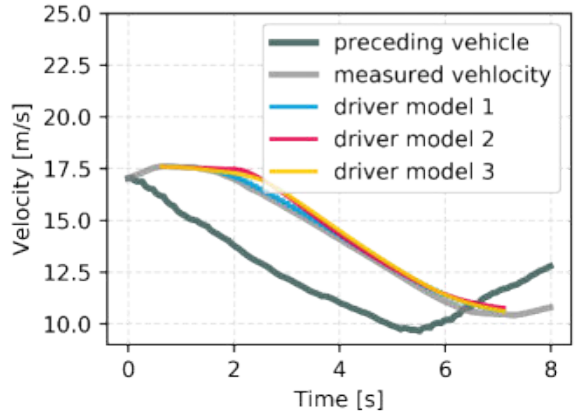


Intelligent driver model-based planning

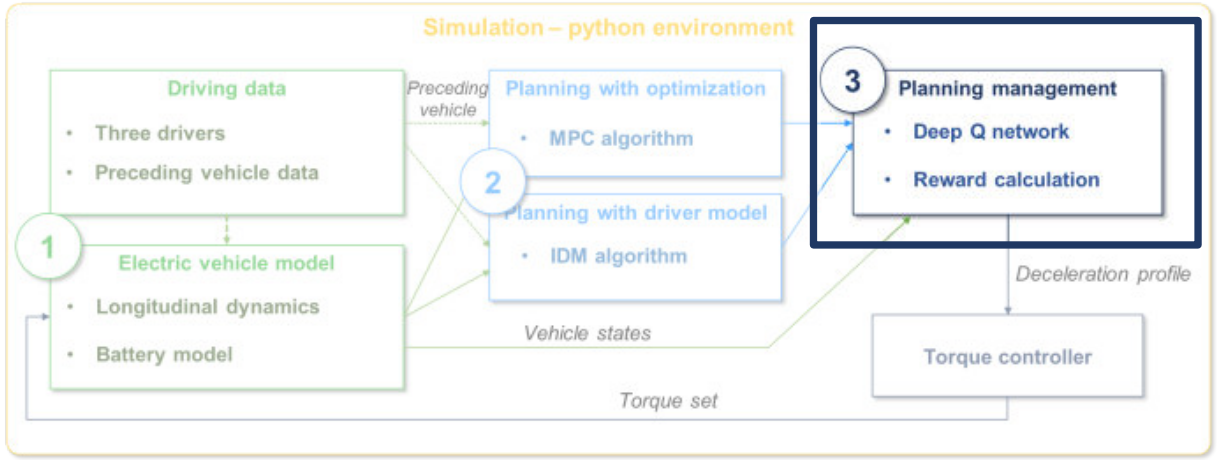
- Deceleration-specified intelligent driver model
  - Parameter update using individual drivers' driving data







Will be presented 21/May, Section B5, Gyubin Sim

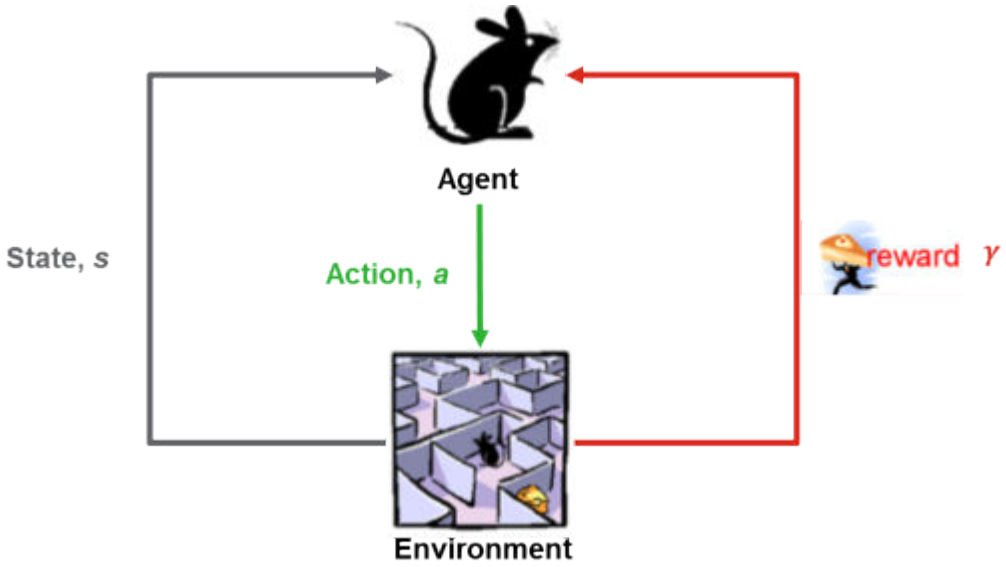


Planning management

- Introduction of reinforcement learning algorithm
- Application of Q learning algorithm

Introduction of reinforcement learning algorithm*

- Machine learning concerned with how ***agent*** in state takes ***action*** in environment so as to maximize cumulative ***reward***



Key Elements of RL

Q learning – Estimation of Q value

Q: Action value function

- *Expected **cumulative reward** (from state **s**, taking action **a**)*

$$Q(S,A) = E(\sum r \,|S_t = s, A_t = a)$$

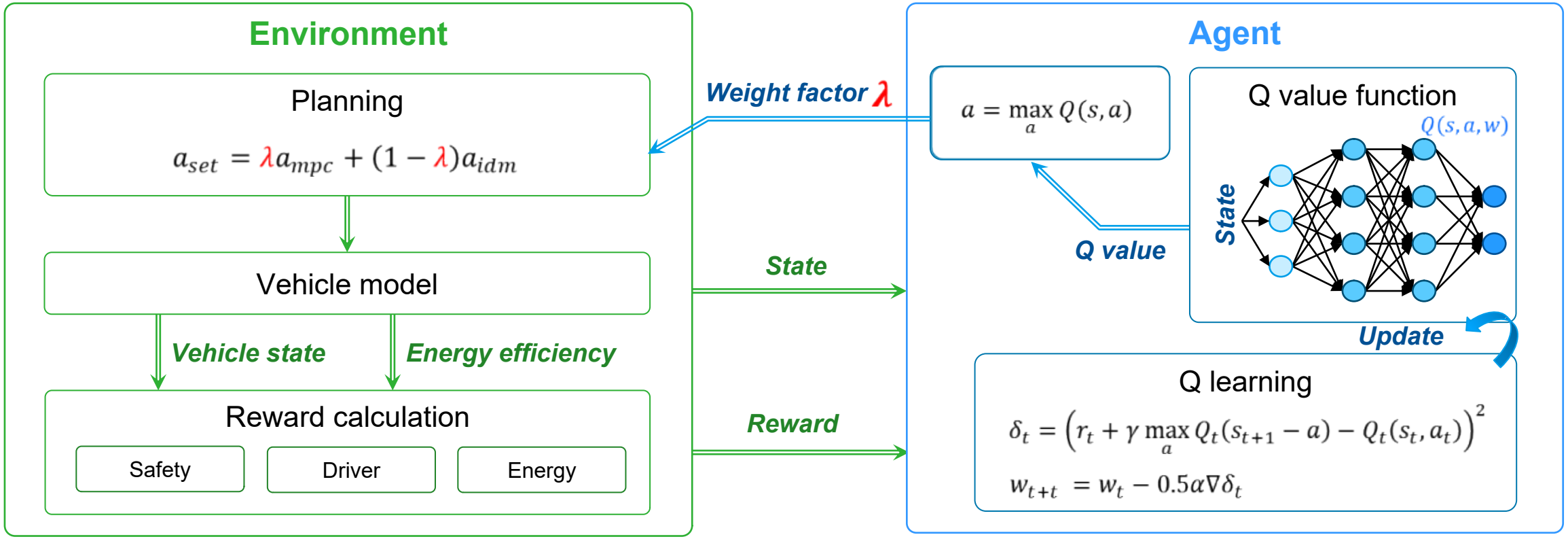
a

r

*RS Sutton, AG Barto. (1998). Introduction to reinforcement learning

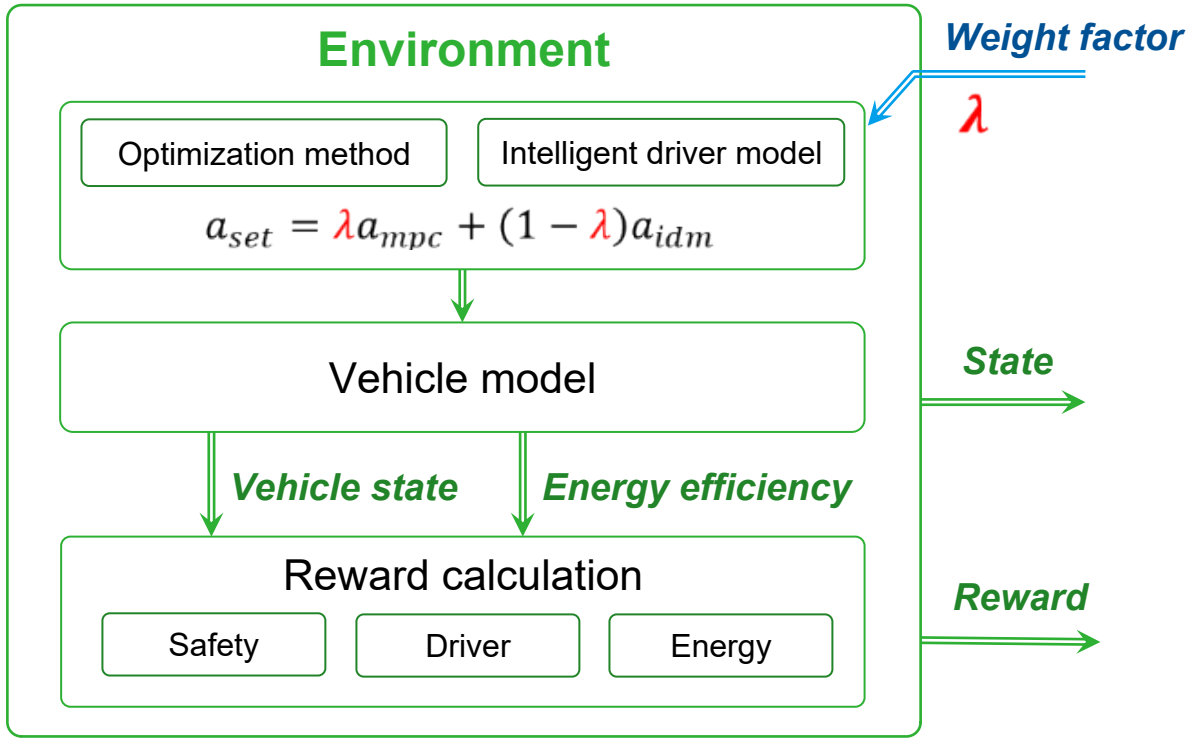
Application of Q learning algorithm to planning management

- Determination of weight factor as action of agent



Environment of Q learning algorithm

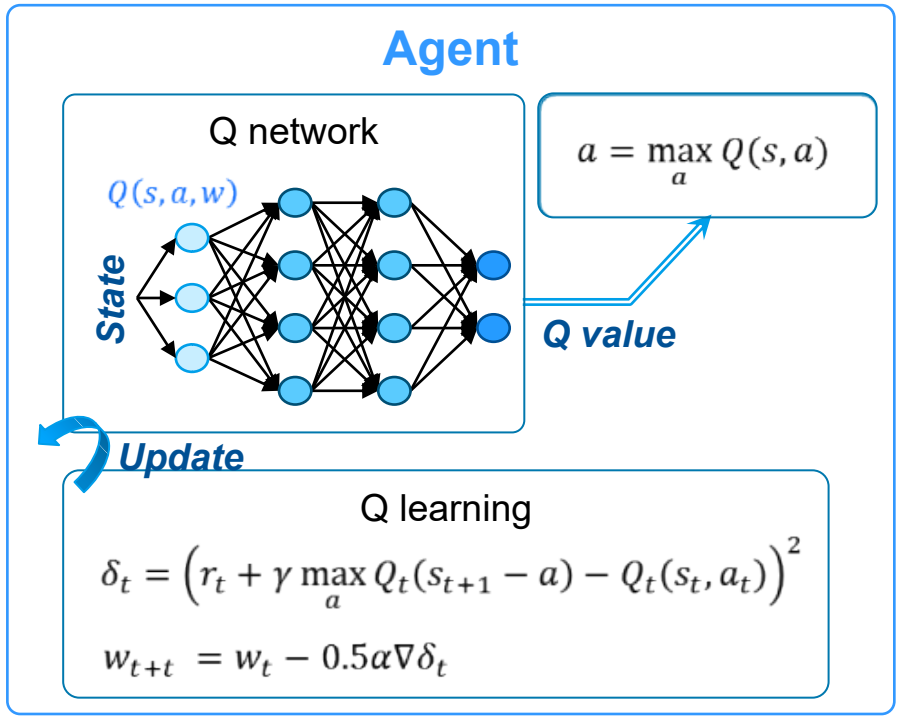
- Regenerative control according to the weight factor determined from agent
- Determination of state and reward by simulation



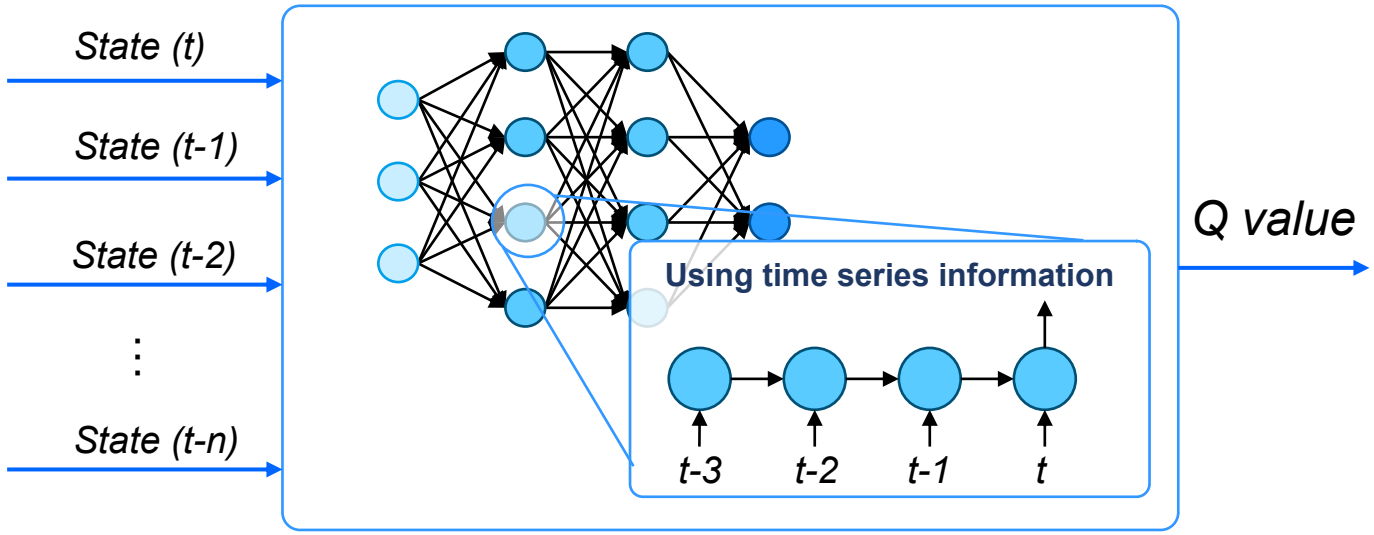
1. Safety reward : Penalty ($rel_{dis} \leq rel_{dis,cri}$)
2. Driver reward: Similarity with driving data
3. Energy reward: Increase in Battery SOC

Agent of Q learning algorithm

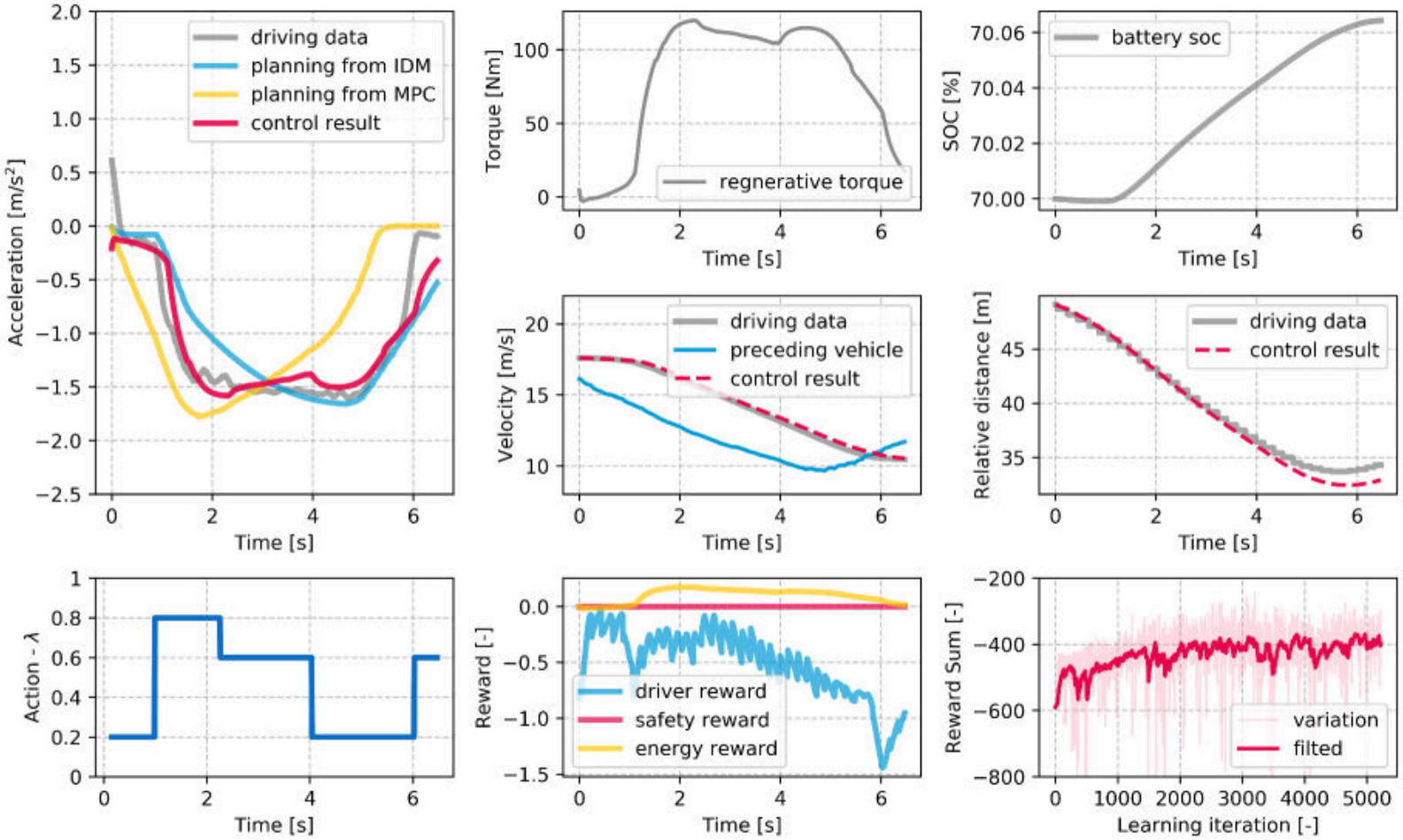
- Q network: Estimation of Q value, based on a ***sequential neural network***
- Learning: Parameter update of Q network using reward from environment



- ***Q network***



Simulation results – Learning results for one deceleration case

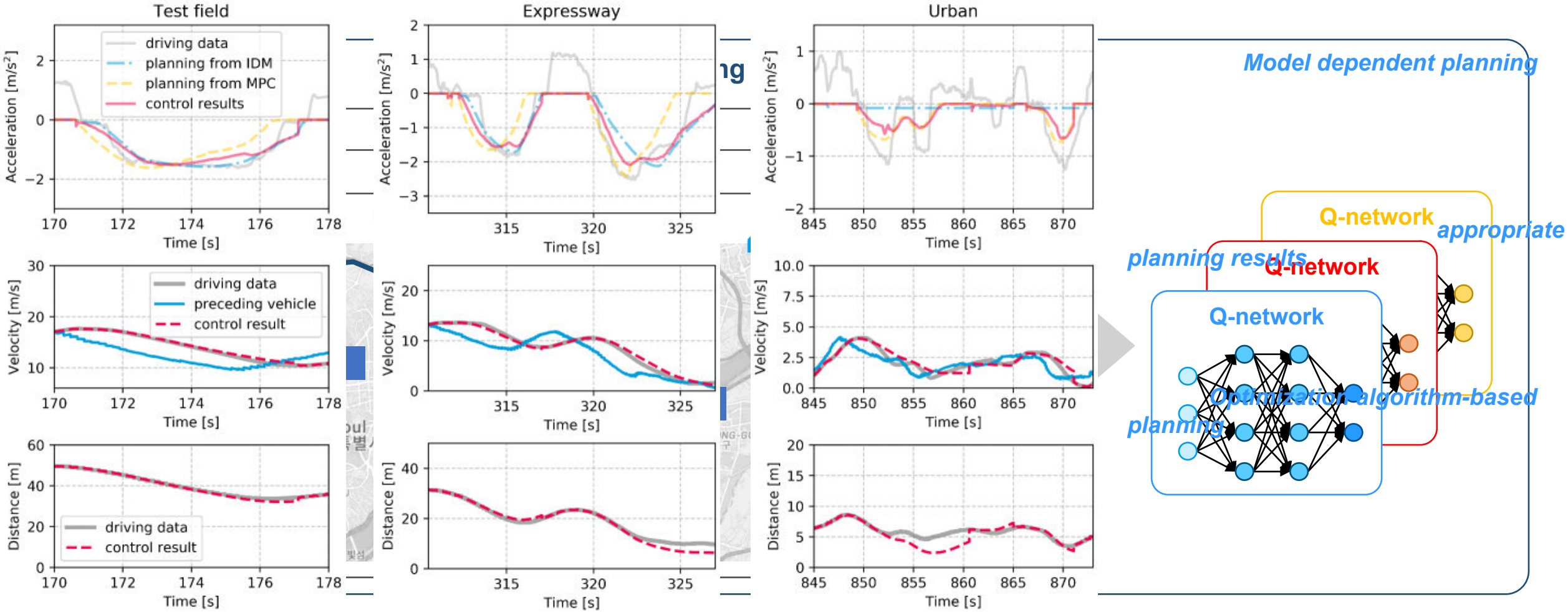


- ***A deceleration profile is determined by the proposed planning algorithm***

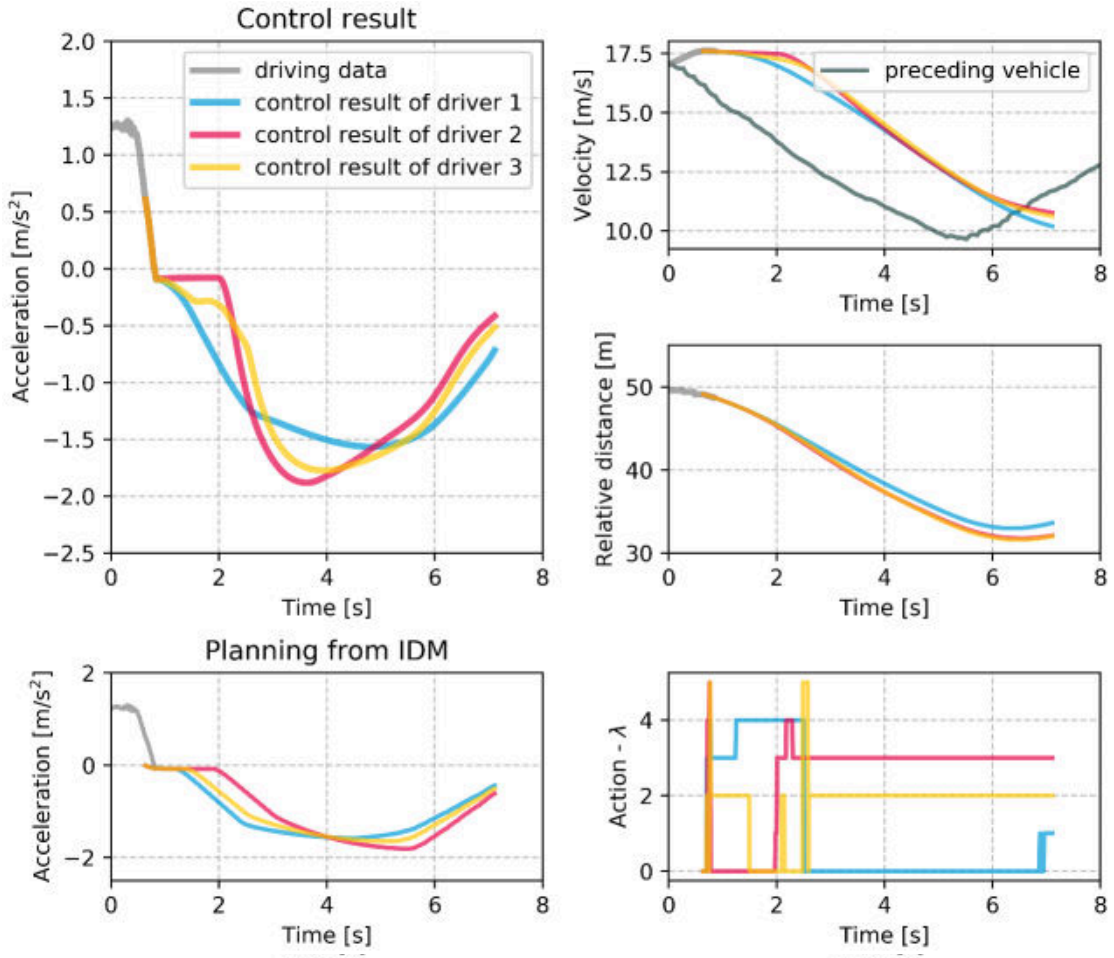
$$a_{set} = \lambda a_{mpc} + (1 - \lambda) a_{idm}$$

- ***Driver reward***
- ***Energy reward***
- ***Control result satisfies the safety criterion***

Simulation results – Learning using various driving data

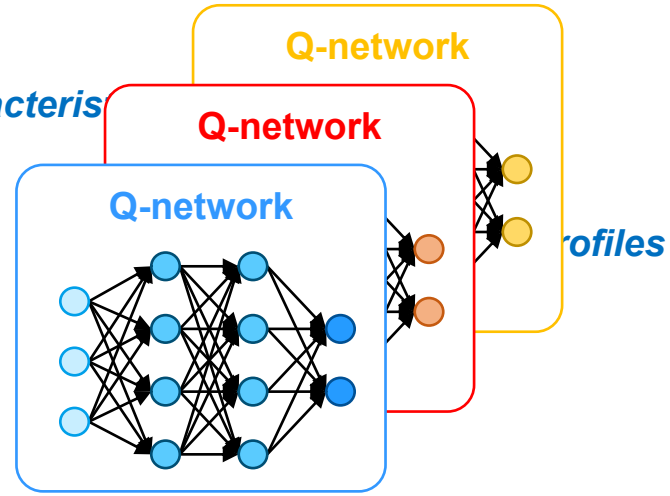


Simulation results – Driver characteristics



- ***Learned agents*** ***same start***
  conditions for deceleration

characteris ***each driver***



rofiles

Summary

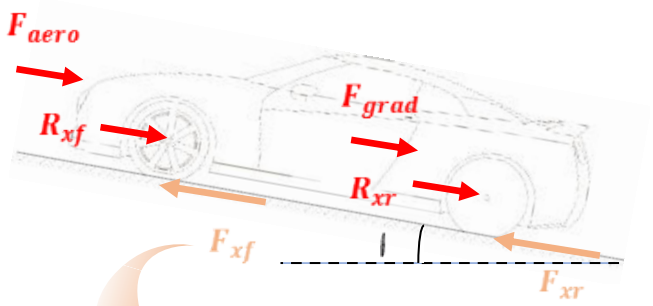
- In this paper, we proposed a regenerative control system which reflects individual driver characteristics based on the reinforcement learning algorithm
  - Electric vehicle model: Vehicle longitudinal dynamic model, Battery model
  - Planning algorithm: Optimization method, Intelligent driver model
  - Planning management: Determination of weight factor with Q learning algorithm
- The validation results show that the generated deceleration profile maximizes the reward while representing driver characteristics under various deceleration conditions

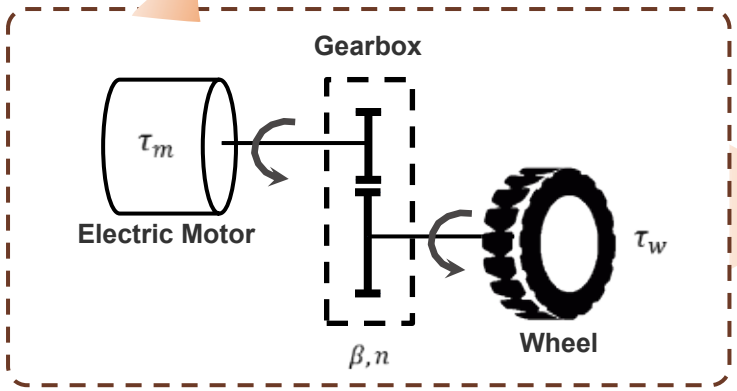
Thank you for your attention!!

Extra slide

Vehicle dynamics

Vehicle configuration





Dynamic equations

Aerodynamic resistance force

$$F_{aero} = \frac{1}{2}\rho_a C_d A_f v^2$$

Rolling resistance force

$$R_x = fmg$$

Gradient resistance force

$$F_{grad} = mgsin\theta$$

Longitudinal tire traction force

$$\tau_{wheel} = \beta n \tau_{motor}$$
$$F_x = \tau_{wheel}/r_{wheel}$$

Total longitudinal motion equation

$$m\ddot{x} = F_{xf} + F_{xr} - F_{aero} - R_{xf} - R_{xr} - F_{grad}$$

$m = mass$
$g = gravitational\ acceleration$
$v = velocity\ of\ the\ vehicle$
$x = moving\ distance\ of\ the\ vehicle$
$\theta = the\ road\ slope\ angle$
$f = rolling\ resistance\ coefficient$
$\beta = gear\ ratio$
$n = efficiency\ of\ whole\ powertrain$
$\rho_a = air\ density$
$C_d = drag\ coefficient$
$A_f = vehicle\ front\ projection\ area$

Reference : "Electric and Hybrid Vehicles - Technologies, Modeling and Control : a Mechatronic Approach" – Amir Khajepour, Saber Fallah, Avesta Goodarzi

Electric vehicle model

Construction

τ_{motor}
$v_{vehicle}$

EV Model

$a_{vehicle}$

* Assumptions : No wind, no slope

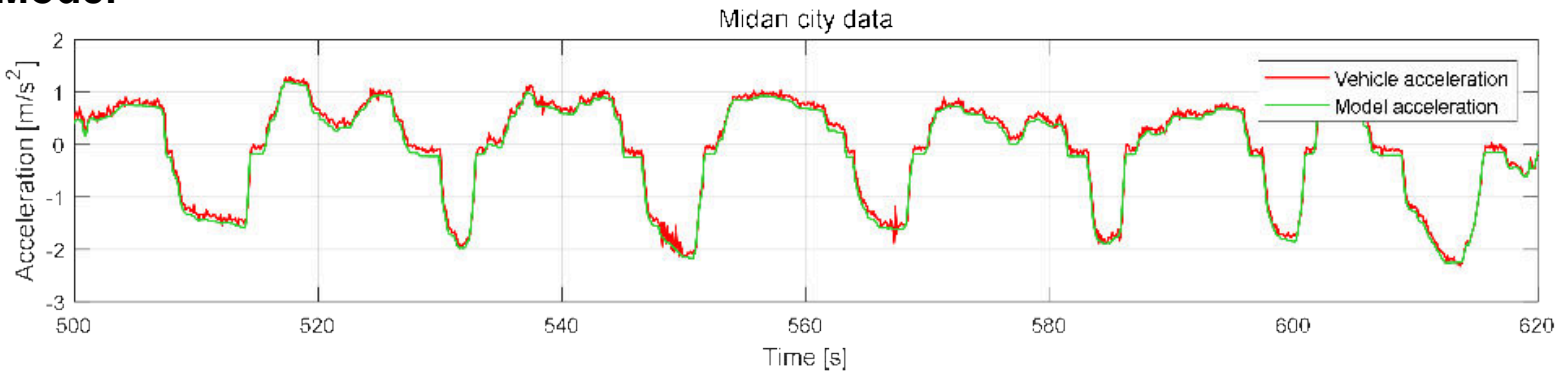
Parameter identification

```
x = lsqcurvefit(fun,x0,xdata,ydata)
```

Find **coefficients x** to best fit the nonlinear function $f(x, xdata)$ to the data $ydata$

$f(x, xdata)$: **EV model**
$ydata$: vehicle test data in Midan city

Model



R^2	0.9599	0.9599
$RMSE$	0.0011	
R^2	0.9599	0.0011
$RMSE$	0.0011	

Valid Model
(Vlaue of R^2 is near 1,
Value of RMSE is near 0)

Model parameter identification

- Using vehicle experiment data

	Description	Value [unit]		Description	Value [unit]
	Vehicle acceleration	[m/s^2]		Inertia of wheel	0.14 [khm^2]
	Vehicle velocity	[m/s]		Inertia of motor	0.028 [khm^2]
	Motor torque	[Nm]		Inertia of shaft	0.75 [khm^2]
	Drag force	[N]		Air drag coefficient	0.171 [Ns^2/m^2]
	Wheel radius	0.318 [m]		Rolling coefficient	143 [N]
	Gear ratio of shaft	7.98 [-]		Rolling coefficient	0.389 [Ns^2/m^2]
	Efficiency of shaft	0.99 [-]		Additional mass	100 [kg]
	Empty vehicle mass	1685 [kg]			

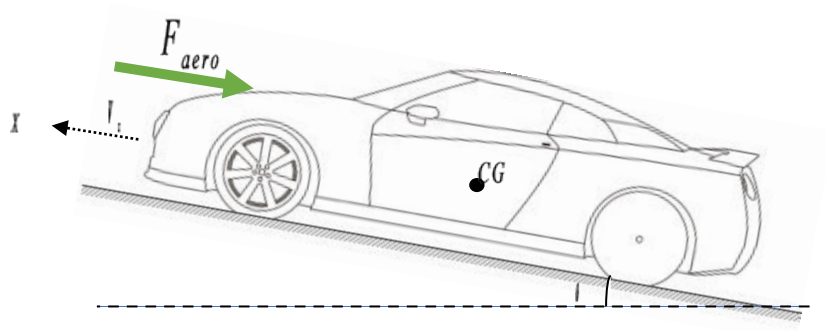
	Description	Value [unit]		Description	Value [unit]
	Series register	[Ohm]		Short capacitor param a	-649
	Short register param a	76.52		Short capacitor param b	-64.3
	Short register param b	-7.95		Short capacitor param c	12692
	Short register param c	23.83		Long capacitor param a	-78409
	Long register param a	5.21		Long capacitor param b	-0.013
	Long register param b	-35.23		Long capacitor param c	30802
	Long register param c	124.9		Open circuit voltage	356 [V]

Aerodynamic drag force

- Equation of Aerodynamic drag force

$$F_{aero} = \frac{1}{2}\rho C_d A_F (V_x + V_{wind})^2$$

- ρ : mass density of air
- C_d : aerodynamic drag coefficient
- A_F : frontal area of the vehicle
- V_{wind} : wind velocity

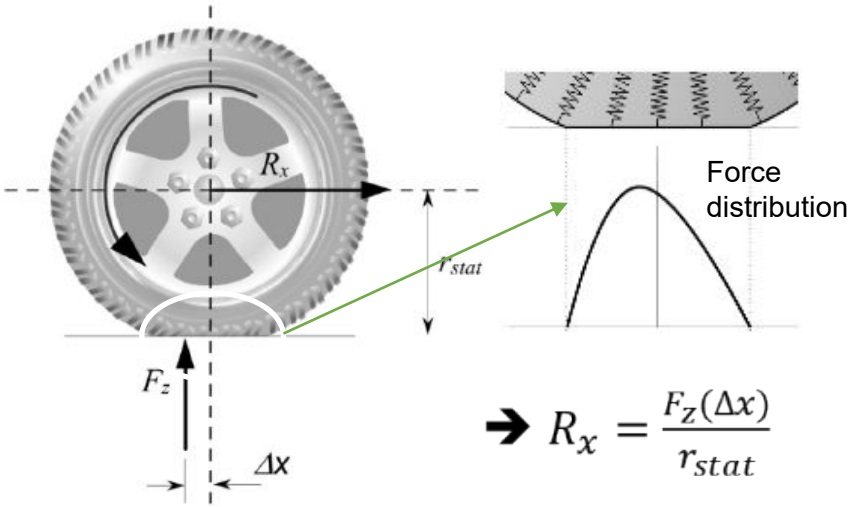


Rolling resistance force

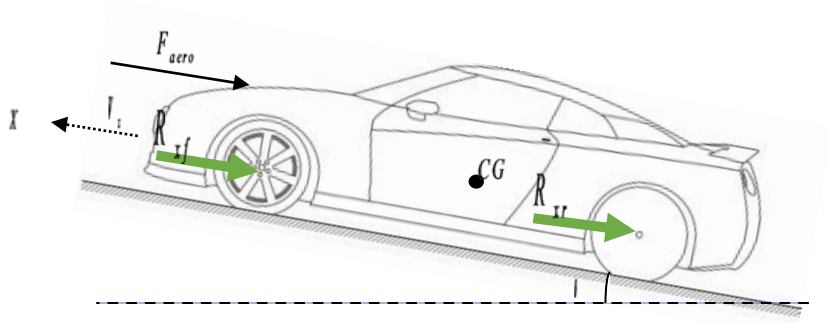
- Force from energy loss of tire material

$$R_{xf} + R_{xr} = f\ (F_{zf} + F_{zr})$$

- R_x : rolling resistance force
- f : rolling resistance coefficient
- F_z : distribution normal load

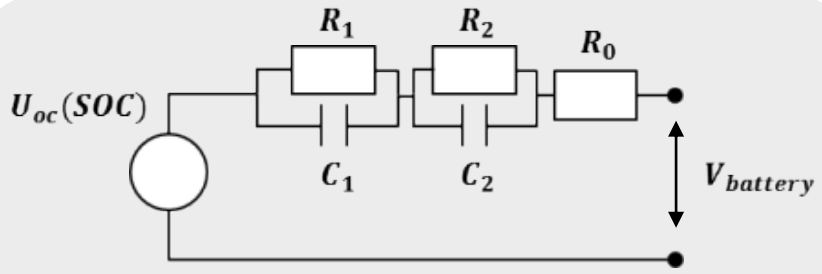


$$\rightarrow R_x = \frac{F_Z(\Delta x)}{r_{stat}}$$



Model parameter optimization

- Pareto optimal solution
  - Multi-objective optimization



Parameter	Initial value
R0	0.0012
R1_a	1.604
R1_b	-0.3
R1_c	2.4
…	…
C2_c	84270

Reference data : Real driving data, GA: Genetic algorithm, V_{oc} : open – curcuit voltage
$Power_{aux}$: Auxiliary electric power consumption, LUT : Look – up table

Objective function

J_1 = RMSE of SOC, J_2 = RMSE of voltage, J_3 = RMSE of current

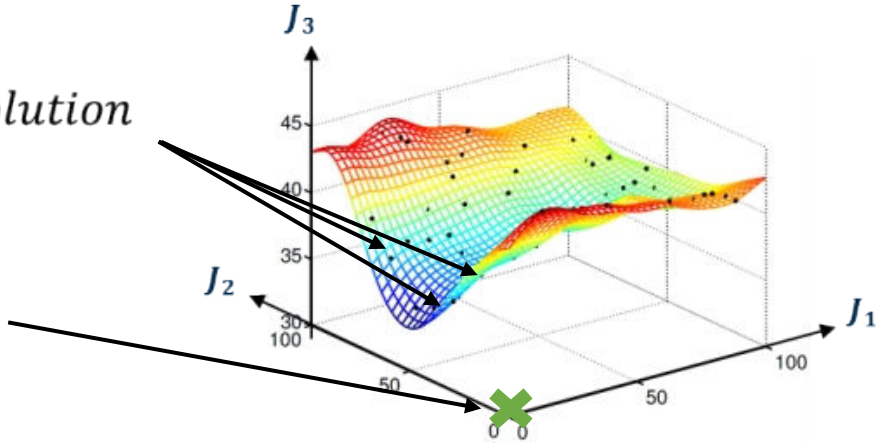
Optimal problem : *Minimize* J_1, J_2, J_3

Input : $R_0, R_{1_a}, R_{2_a}, \dots, C_{2_c}, Power_{acc}, V_{oc}, V_{oc}$ *vs SOC LUT* (29 params)

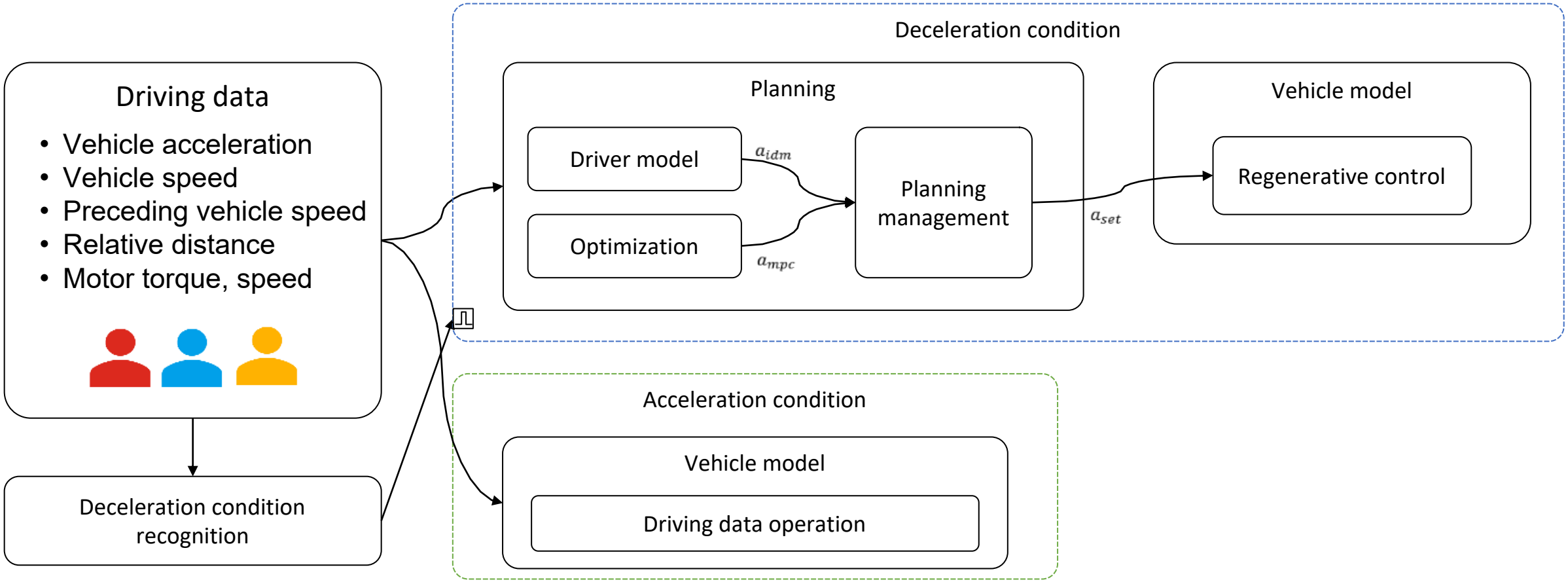
Optimizer : Multi objcective GA

Pareto optimal solution

Optimal solution



Application of Driving data



Driving data of divergence environment

MPC configuration

minimize $$f = \sum_{t=0}^{N_p} (e_t^T Q e_t^T + u_t^T R u_t)$$

- e_t: *Error* $(X_r - X)$
- u_t: *Input (Vehicle Acc)*

subject to $$X_{t+1} = AX_t + Bu_t$$

$$u_t \geq u_{min} \text{ and } u_t \leq u_{max}$$

$$Q = \begin{bmatrix} Q1 & 0 \\ 0 & Q2 \end{bmatrix} \qquad N_p$$

N_p	10 ~ 20
Q_1	4
Q_2	0.1

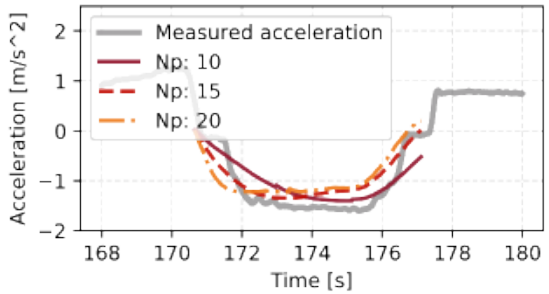
10 ~ 20

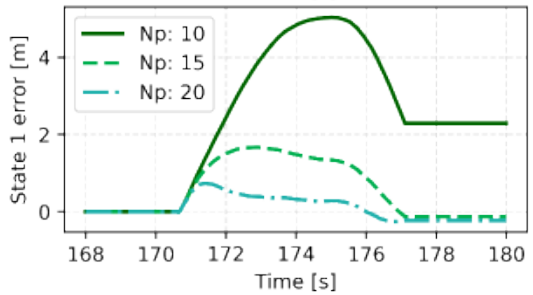
N_p	10 ~ 20
Q_1	4
Q_2	0.1

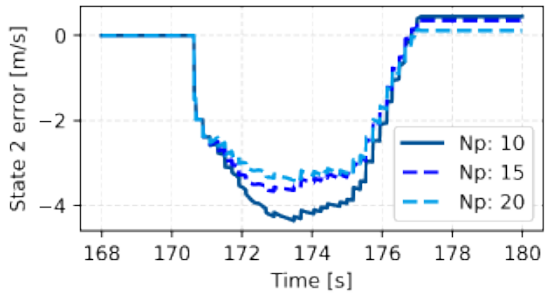
4

N_p	10 ~ 20
Q_1	4
Q_2	0.1

0.1







N_p	15
Q_1	1 ~ 8
Q_2	0.1

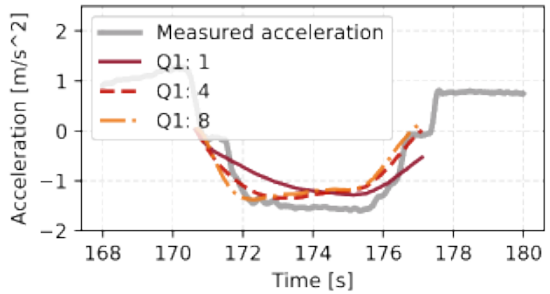
15

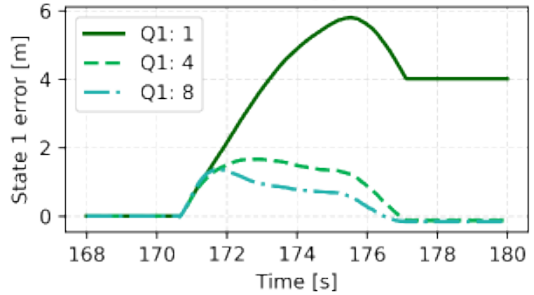
N_p	15
Q_1	1 ~ 8
Q_2	0.1

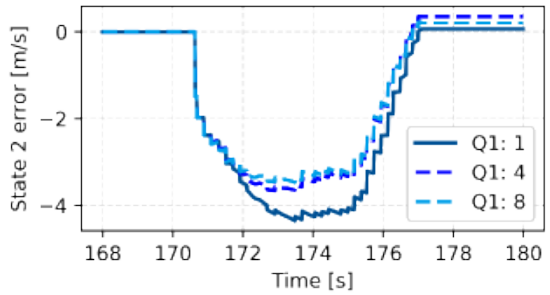
1 ~ 8

N_p	15
Q_1	1 ~ 8
Q_2	0.1

0.1







N_p	15
Q_1	4
Q_2	0.02 ~ 0.5

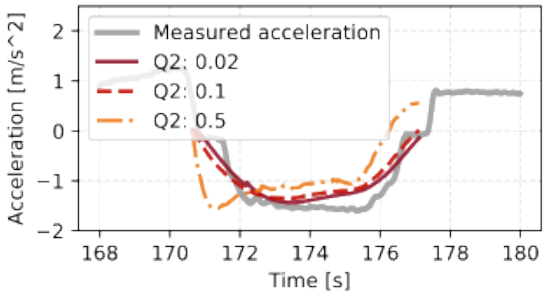
15

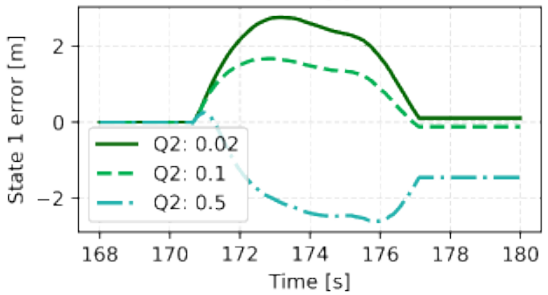
N_p	15
Q_1	4
Q_2	0.02 ~ 0.5

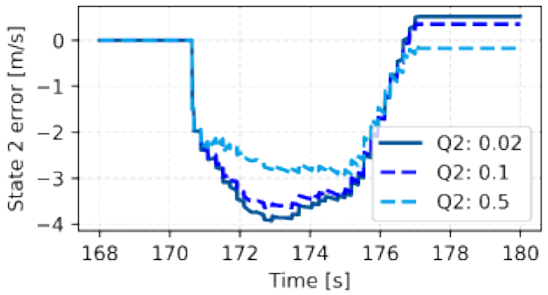
4

N_p	15
Q_1	4
Q_2	0.02 ~ 0.5

0.02 ~ 0.5

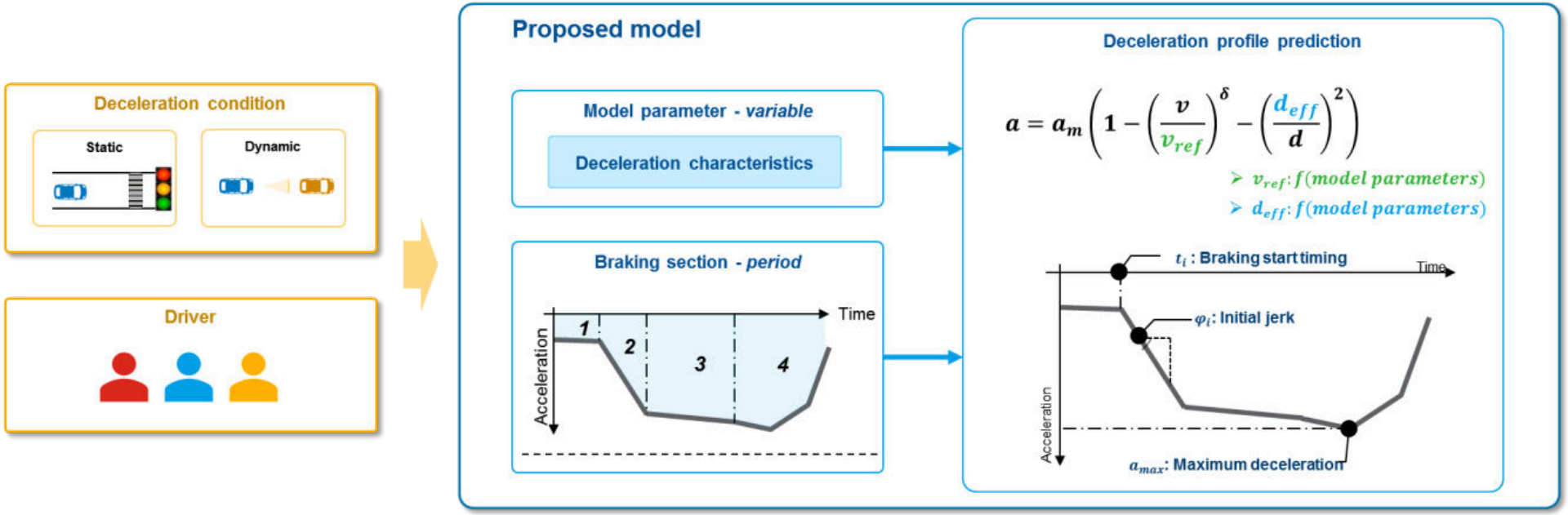






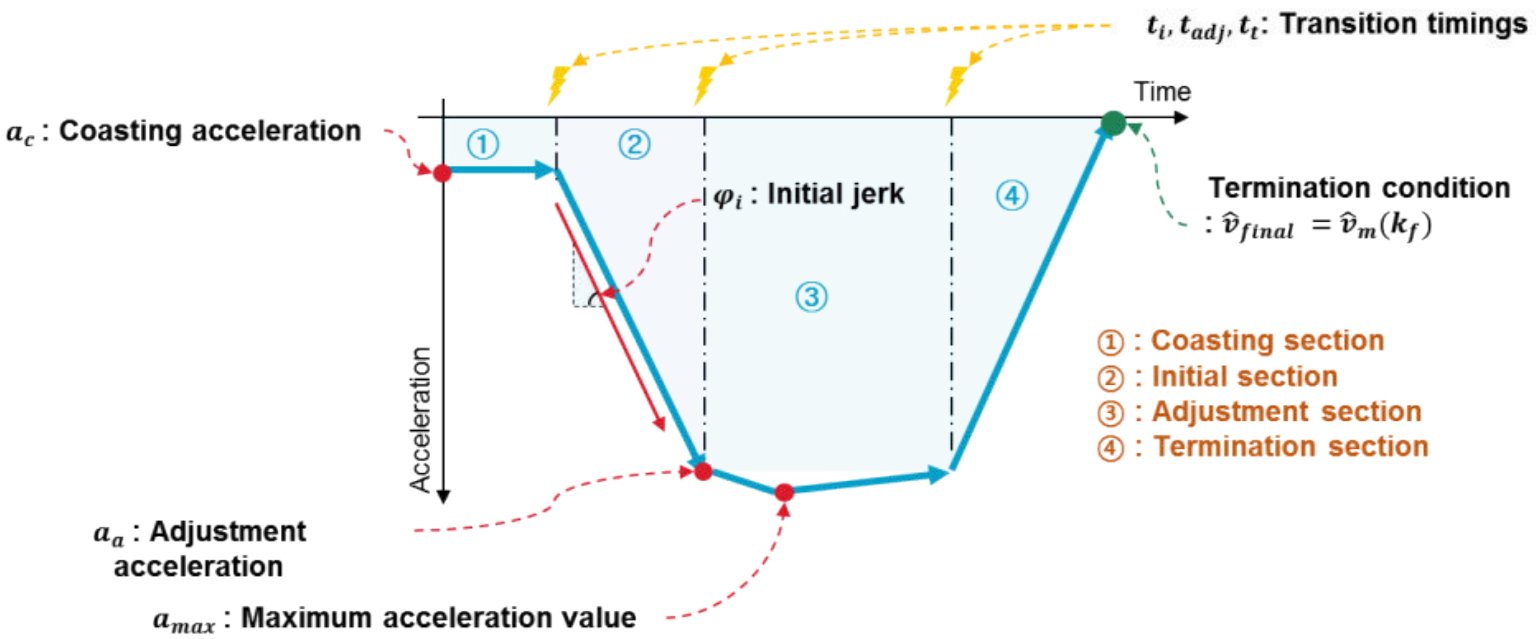
Proposed model - Modified intelligent driver model for SRB

- Deceleration profile prediction based on parametric equations
  - Deceleration condition recognition when release timing of acceleration pedal
  - Activation of model parameters according to deceleration condition and driver
  - Deceleration prediction using different parametric equations depending on braking section



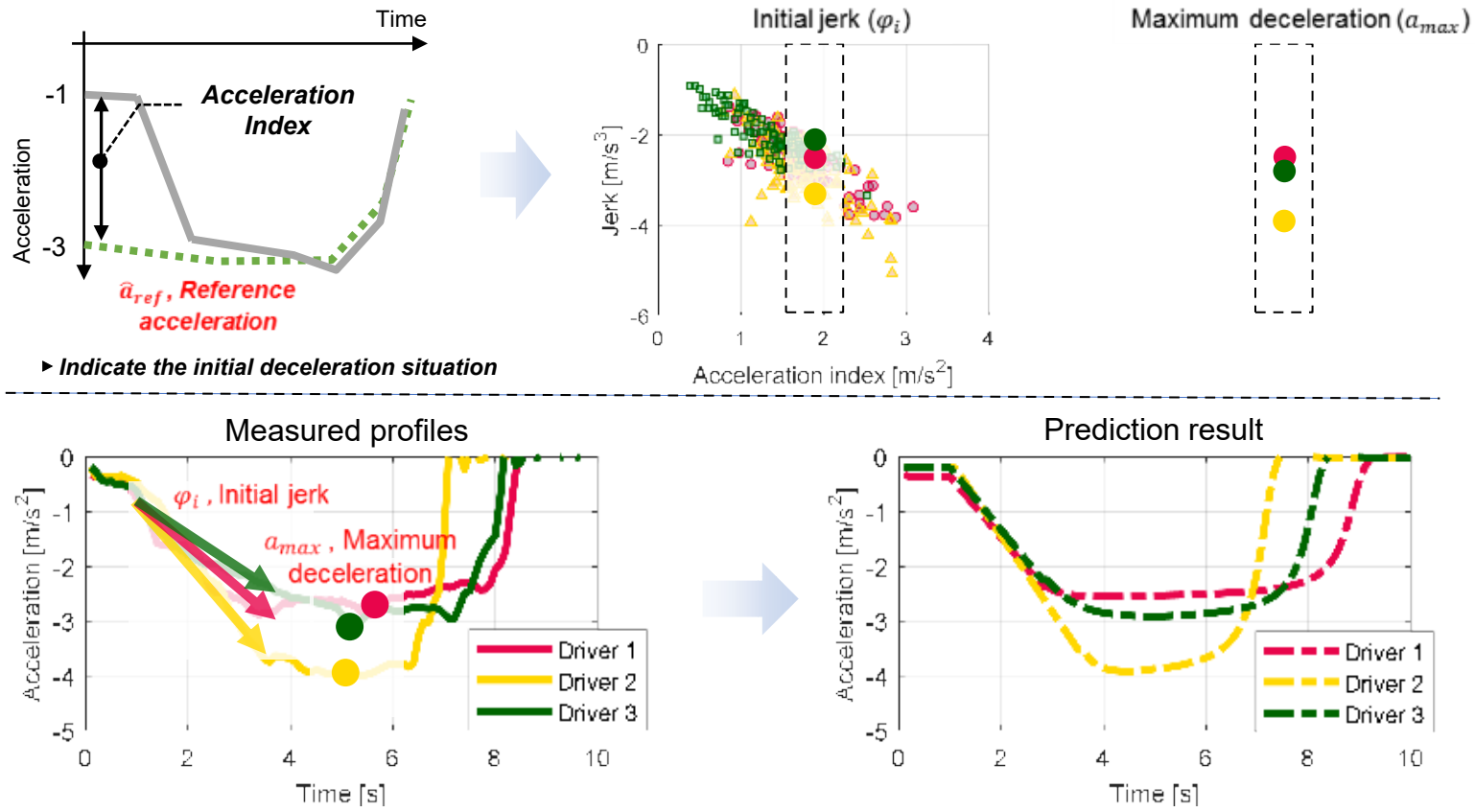
Model parameters about deceleration

- Representation of individual driver characteristics
  - Transition timings of braking sections
  - Acceleration and jerk
  - Relative distance to objective
  - Termination condition



Driver characteristics analysis

- Analysis of the driver characteristics based on model parameters



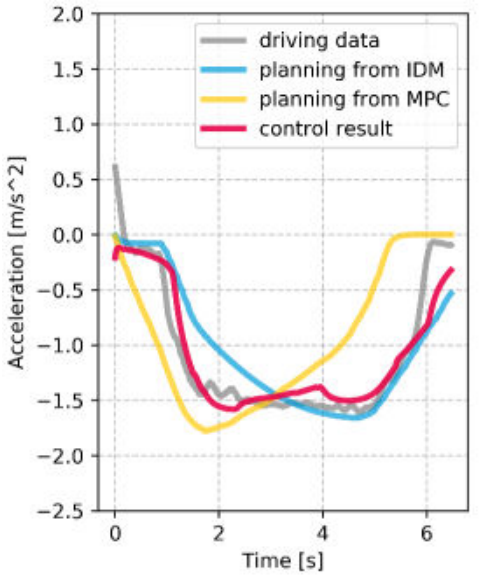
Details for reward calculation

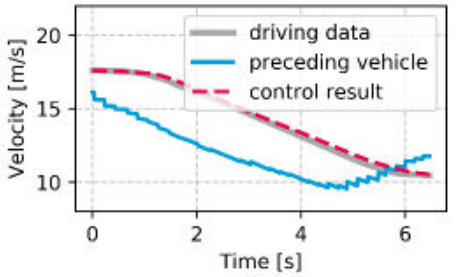
$$r_{drv} = c_1 * |(a_{drv} - a_{ctl})| + c_2 * |(vel_{drv} - vel_{ctl})| + c_3 * |(dis_{drv} - dis_{ctl})|$$
$$c_1 = 1, \quad c_2 = 0.5, \quad c_3 = 0.5$$

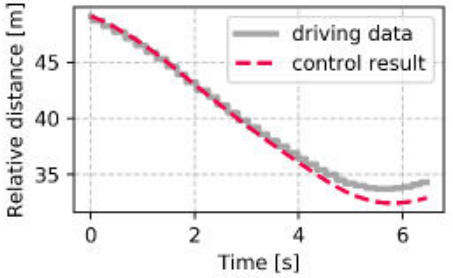
$$r_{eng} = c_e * \dot{soc}, \quad c_e = 10$$

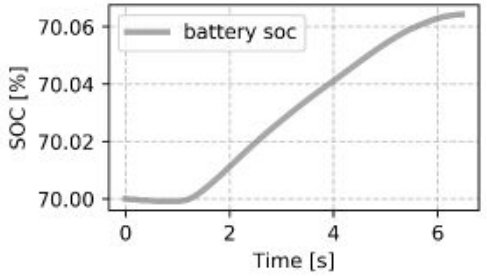
$$r_{saf} = \begin{cases} c_s & dis \le dis_{cri} \\ c_c & dis \le 0 \\ 0 & else \end{cases}, \quad c_s = 10, \quad c_c = 100, \quad dis_{cri} = 3\,m$$

$$r_{sum} = r_{drv} + r_{eng} + r_{saf}$$









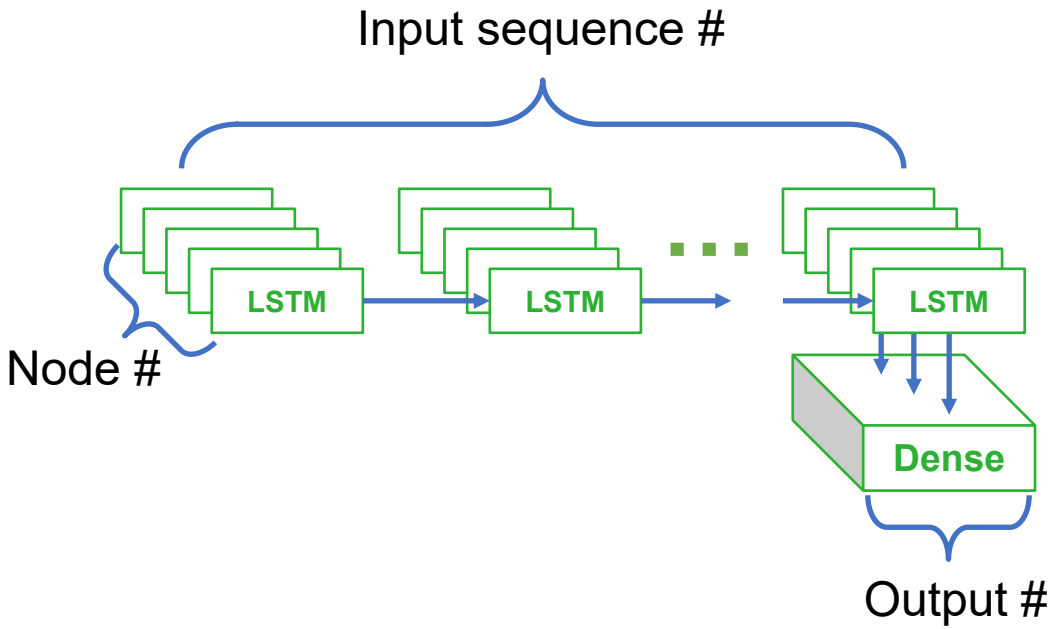
Details of Q network model

Model summary

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 265)	290440
dense_1 (Dense)	(None, 6)	1596

Total params: 292,036
Trainable params: 292,036
Non-trainable params: 0

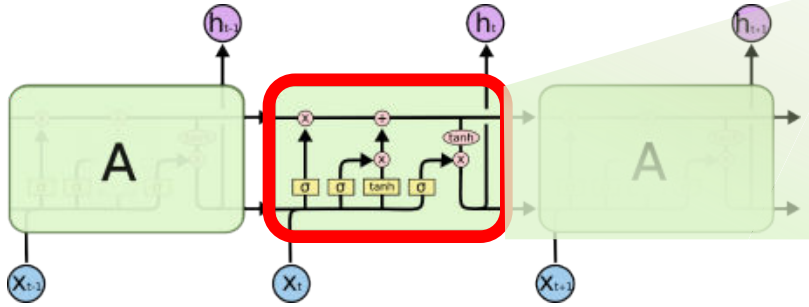
Layer	Two layers (Sequential + output)
Sequential network cell	Long short term memory
Node #	265
Input sequence #	8

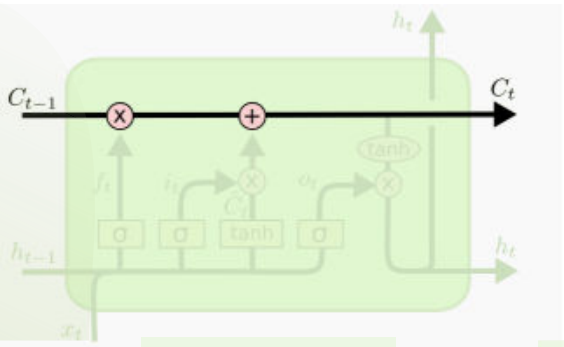


Long short term memory networks (I)

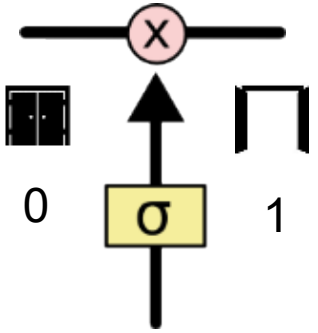
- The core idea behind LSTM (Long Short Term Memory network)
  - To relieve the vanishing gradient problem, LSTM uses cell state and gate(sigmoid function)
  - Cell state is a conveyor belt carrying information.
  - Gates(sigmoid function) determine to remove or add information to the cell state
    - Output of sigmoid function is from 0 to 1. It is used as gate (0: let nothing through, 1: let everything through)

LSTM





Cell state

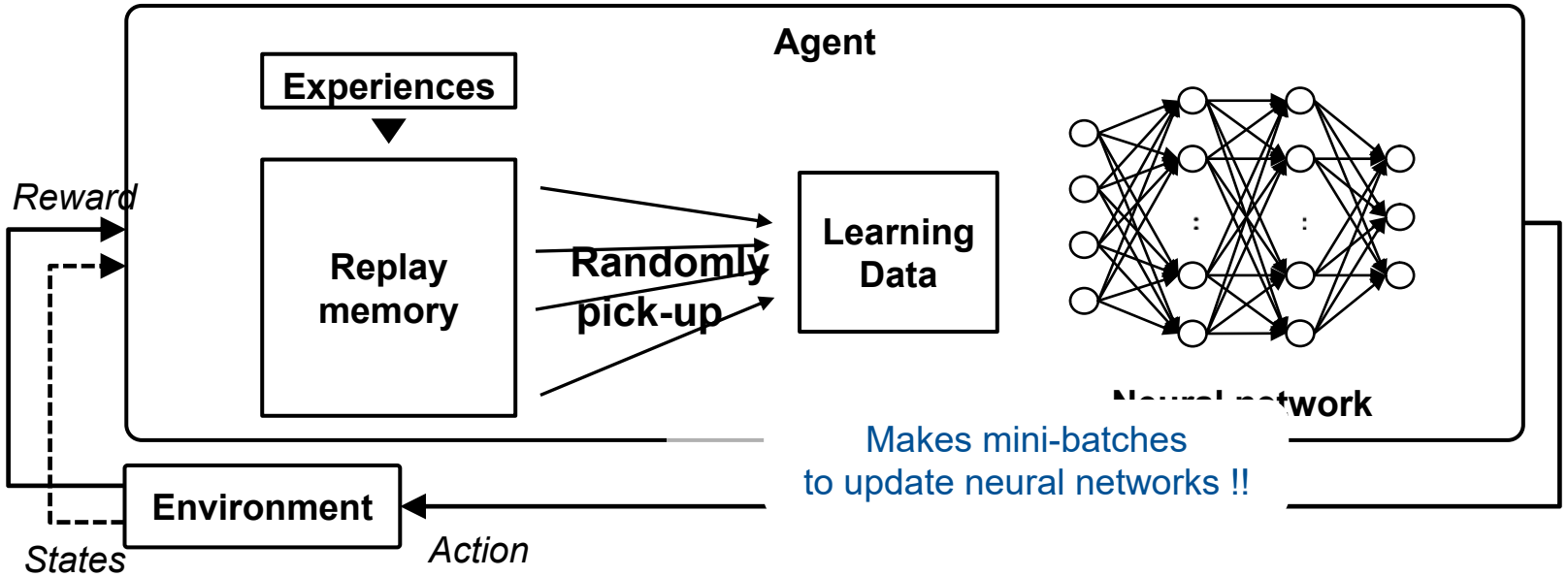


Sigmoid function (σ)

Deep Q - learning*

- "Deep Q-learning from Demonstrations" , *arXiv (*Open access, *Cornell University Library), November, 2017, Google DeepMind Team (Todd Hester, etc)*

- Experience replay
  - Saving experiences (states, action, reward, next states) in a replay memory
  - Training neural network with randomly selected experiences from the replay memory
  - Overwriting the oldest self-generated experiences if over capacity



Q-learning

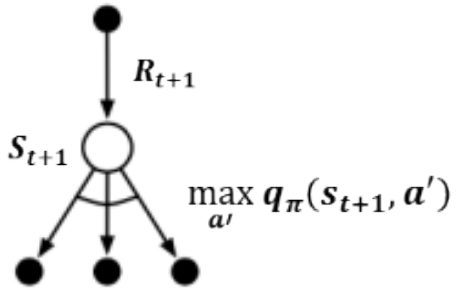
- Policy evaluation
  - Value function is updated by the *current value (t)*, *reward (t+1)*, and *expected value (t+1)* with the policy

Example)

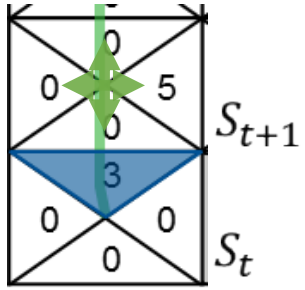
Updated value

$$q_\pi(s_t, a_t) \leftarrow q_\pi(s_t, a_t) + \alpha(\cancel{q_t} - q_\pi(s_t, a_t))$$

$$(\mathbf{R}_{t+1} + \max_{a'} q_\pi(s_{t+1}, a'))$$



Trajectory



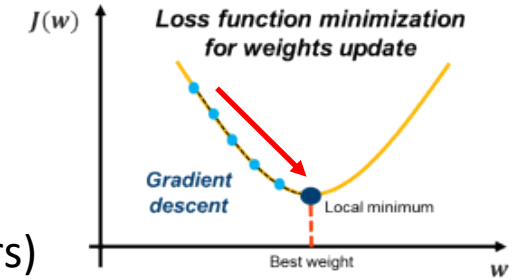
$$q_\pi(s_{t+1}, a'_u) = 0$$
$$q_\pi(s_{t+1}, a'_d) = 0$$
$$q_\pi(s_{t+1}, a'_l) = 0$$
$$q_\pi(s_{t+1}, a'_r) = 5$$

$$\max_{a'} q_\pi(s_{t+1}, a') = 5$$

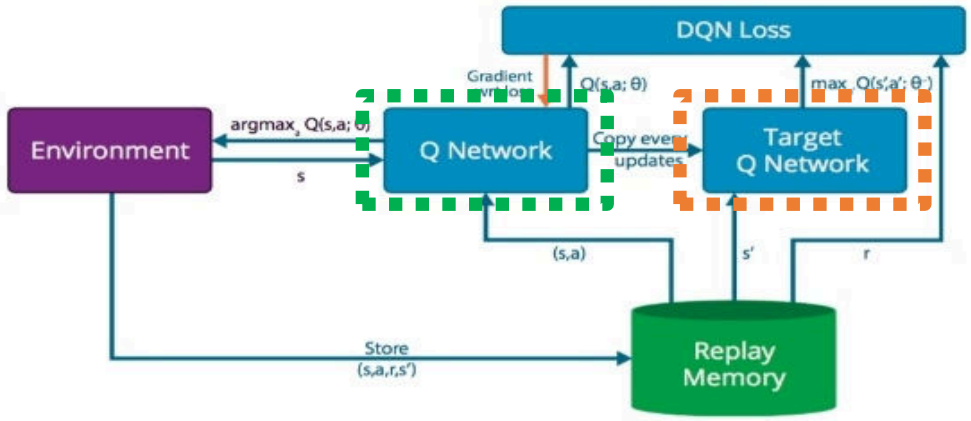
Value update in Q-learning

$$\rightarrow q_\pi(s_1, a_u) \leftarrow 3 + (0.2)((r_2 + 5) - 3)$$

Deep Q - network (DQN)*



- Network update
  - Cloning the Q network (Design with same network structure and update same parameters)
  - Approximating target values / updating parameters at every episode (Q network updates at every step)



Nair, et. al "Massively parallel methods for deep reinforcement learning." In ICML Deep learning Workshop. 2015.

◆ Learning policy

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max Q(S_{t+1}, a') - Q(S_t\right.$$

Learning rate · Discount factor · Old value

Updated value · Old value

Reward · Estimation of future value

Learned value

◆ Loss function

$$J_{DQ}(\theta) = (answer - prediction$$

$$= \left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a'; \theta^-) - Q(S_t, A_t; \theta)\right)^2$$

Network parameter

Answer (Target network)

Prediction (Q network)

- "Deep Q-learning from Demonstrations" , *arXiv (*Open access, *Cornell University Library), November, 2017, Google DeepMind Team (Todd Hester, etc)*

Result – driver 1 (all driving cases)