Brief Introduction to CommonRoad-io

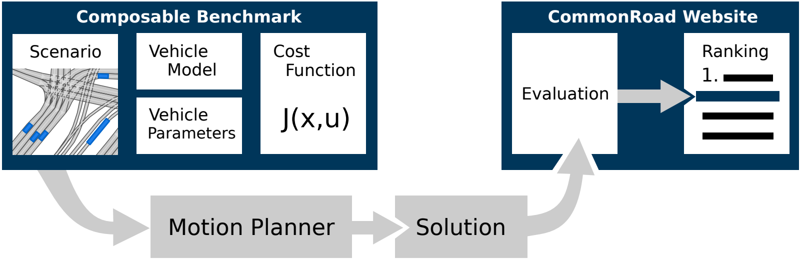
Edmond Irani Liu

Cyber-Physical Systems Group
Technische Universität München

April 9, 2020

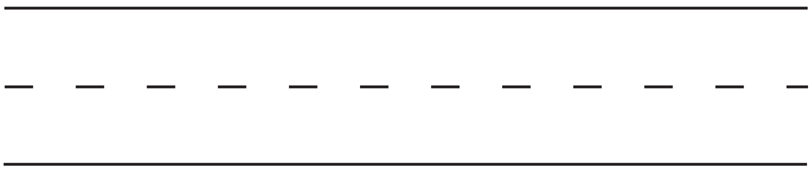
What is CommonRoad?

Composable Benchmarks for **Mo**tion Planning **on Roads**



Website: **https://commonroad.in.tum.de**

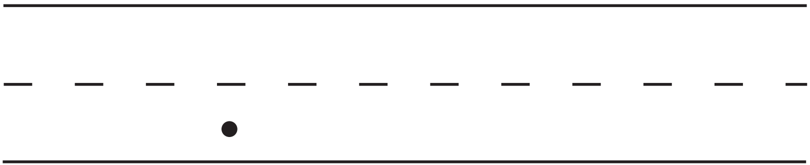
Motion Planning With CommonRoad



Scenario (S)

Road network

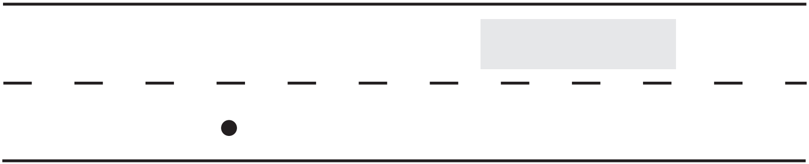
Motion Planning With CommonRoad



Scenario (S)

Road network, initial state x_0

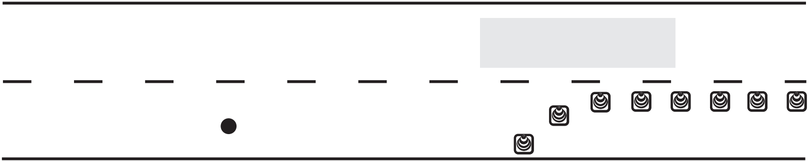
Motion Planning With CommonRoad



Scenario (S)

Road network, initial state x_0, goal region $\mathcal{G}$

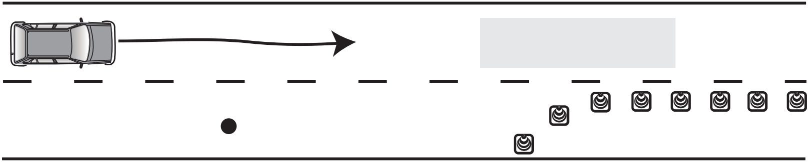
Motion Planning With CommonRoad



Scenario (S)

Road network, initial state x_0, goal region $\mathcal{G}$, static obstacles

Motion Planning With CommonRoad



Scenario (S)

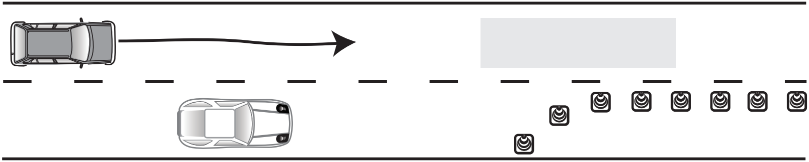
Road network, initial state x_0, goal region $\mathcal{G}$, static obstacles, dynamic obstacles (including movement over time)

Motion Planning With CommonRoad

Vehicle model (M)

$\dot{x}(t) = f(x(t), u(t))$

x: state, u: input



Scenario (S)

Road network, initial state x_0, goal region $\mathcal{G}$, static obstacles, dynamic obstacles (including movement over time)

Motion Planning With CommonRoad

Vehicle model (M)

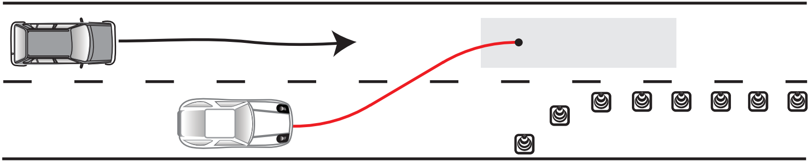
$$\dot{x}(t) = f(x(t), u(t))$$

x: state, u: input

Cost function (C)

$$J_C = \Phi_C(x(t_0), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} L_C(x(t), u(t), t)\,\mathrm{d}t$$

Φ_C: terminal costs,
L_C: running costs



Scenario (S)

Road network, initial state x_0, goal region $\mathcal{G}$, static obstacles, dynamic obstacles (including movement over time)

Motion Planning With CommonRoad

Vehicle model (M)

$\dot{x}(t) = f(x(t), u(t))$

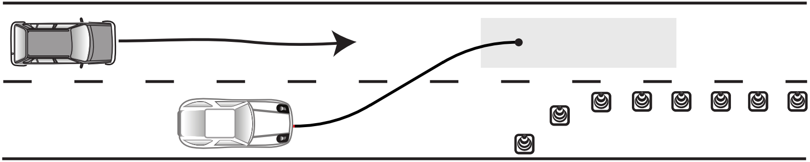
x: state, u: input

Cost function (C)

$$J_C = \Phi_C(x(t_0), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} L_C(x(t), u(t), t)\,\mathrm{d}t$$

Φ_C: terminal costs,

L_C: running costs

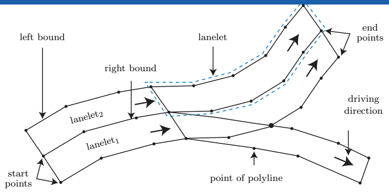


Individual ID: M:C:S

Scenario (S)

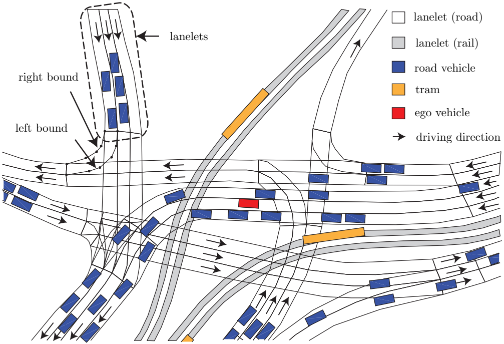
Road network, initial state x_0, goal region $\mathcal{G}$, static obstacles, dynamic obstacles (including movement over time)

Scenarios: Road Network

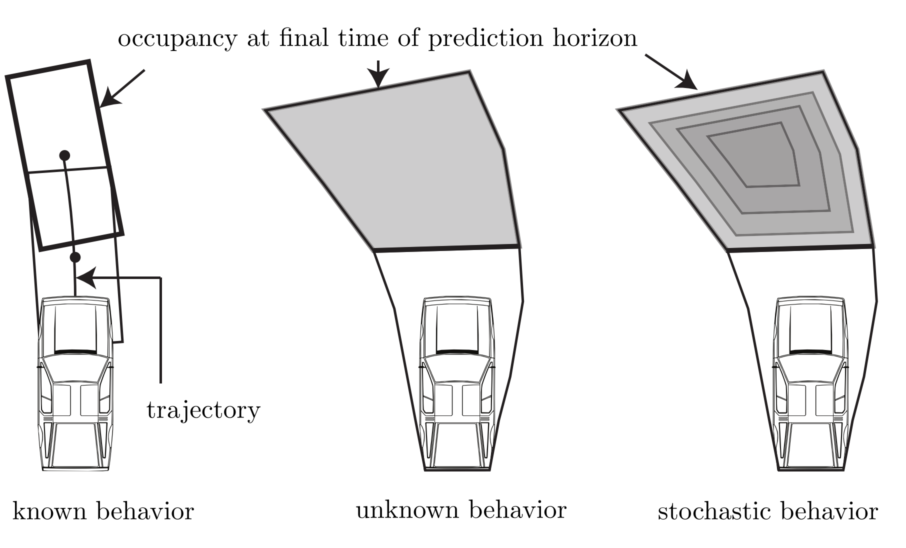


P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

Example of a complicated crossing in Munich (Stachus):

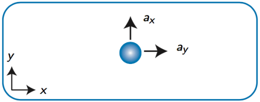


Scenarios: Obstacles



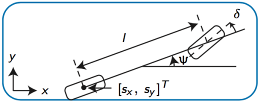
Kinematic Models

Models



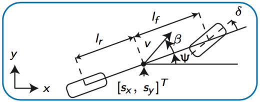
Point-mass model (PM)

- Holonomic system
- $\ddot{x} = a_x, \quad \ddot{x} = a_y$



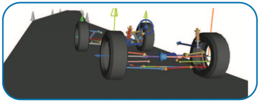
Kinematic single-track model (KS)

- Nonholonomic system
- Considers minimum turning radius
- No tire slip



Single-track model (ST)

- Considers tire slip
- Can explain understeer and oversteer
- No individual tire loads



Multi-body model (MB)

- Individual tire loads
- Effects from yaw, pitch, and roll
- Detailed suspension model

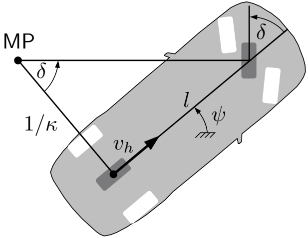
Point-Mass Model (PM)

$$\begin{aligned}\dot{x} &= v_x\\ \dot{y} &= v_y\\ \dot{v}_x &= a_x\\ \dot{v}_y &= a_y\end{aligned}$$

- Point mass with state space $\mathcal{X}$ and admissible controls $\mathcal{U}$
- Control variables $u_1 = a_x$ and $u_2 = a_y$
- Constrained by Kamm's circle: $\sqrt{a_x^2 + a_y^2} \leq a_{\max}$
- Disadvantage: ignores minimum turning radius

Kinematic Single-Track Model (KS)

$$\dot{x} = v_h \cos(\psi)$$
$$\dot{y} = v_h \sin(\psi)$$
$$\dot{\psi} = \frac{v_h}{l} \tan(\delta)$$
$$\dot{v}_h = a_{\mathrm{long}}$$
$$\dot{\delta} = v_\delta$$



- Two wheels connected by rigid link
- Considers differential constraints (nonholonomic constraint)
- Disregards tire slip
- Control variables $u_1 = a_{\mathrm{long}}$ and $u_2 = v_\delta$

Cost Functions

Just like other components of the benchmark, the cost functions are also interchangeable:

$$J_C(x(t), u(t), t_0, t_f) = \sum_{i \in \mathcal{I}} w_i \, J_i(x(t), u(t), t_0, t_f),$$

where $\mathcal{I}$ contains the IDs of partial cost functions and $w_i \in \mathbb{R}^+$ are weights. Examples:

- **Time**: $J_T = t_f$ (see Bobrow et al., 1988).
- **Acceleration**: $J_A = \int_{t_0}^{t_f} a(t)^2 \, \mathrm{d}t$ (see Ziegler et al., 2014b).
- **Jerk**: $J_J = \int_{t_0}^{t_f} \dot{a}(t)^2 \, \mathrm{d}t$ (see Werling et al., 2010).
- **Steering angle**: $J_{SA} = \int_{t_0}^{t_f} \delta(t)^2 \, \mathrm{d}t$ (see Magdici et al., 2016).
- etc.

A cost-function ID (e.g. $JB1$, $SA1$, and $WX1$) uniquely specifies a set of weights for the partial costs.

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.
- **Composability:** All components (vehicle models, cost functions, and scenarios) are interchangeable.

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.
- **Composability:** All components (vehicle models, cost functions, and scenarios) are interchangeable.
- **Representativeness:** Real traffic and hand-crafted problems (most recorded traffic situations are not critical).

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.
- **Composability:** All components (vehicle models, cost functions, and scenarios) are interchangeable.
- **Representativeness:** Real traffic and hand-crafted problems (most recorded traffic situations are not critical).
- **Portability:** XML for scenarios (platform-independent); Vehicle models in MATLAB and Python (both platform-independent).

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.
- **Composability:** All components (vehicle models, cost functions, and scenarios) are interchangeable.
- **Representativeness:** Real traffic and hand-crafted problems (most recorded traffic situations are not critical).
- **Portability:** XML for scenarios (platform-independent); Vehicle models in MATLAB and Python (both platform-independent).
- **Scalability:** From simple static to complex scenarios with many dynamic obstacles.

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.
- **Composability:** All components (vehicle models, cost functions, and scenarios) are interchangeable.
- **Representativeness:** Real traffic and hand-crafted problems (most recorded traffic situations are not critical).
- **Portability:** XML for scenarios (platform-independent); Vehicle models in MATLAB and Python (both platform-independent).
- **Scalability:** From simple static to complex scenarios with many dynamic obstacles.
- **Openness:** All benchmarks downloadable from our website & possibility to suggest new ones.

Key Features

- **Reproducibility/unambiguity:** Unambiguous information representation & manuals on our website.
- **Composability:** All components (vehicle models, cost functions, and scenarios) are interchangeable.
- **Representativeness:** Real traffic and hand-crafted problems (most recorded traffic situations are not critical).
- **Portability:** XML for scenarios (platform-independent); Vehicle models in MATLAB and Python (both platform-independent).
- **Scalability:** From simple static to complex scenarios with many dynamic obstacles.
- **Openness:** All benchmarks downloadable from our website & possibility to suggest new ones.
- **Independence:** Our benchmarks are independent from planning libraries.

Installation

1. Download and install Anaconda (**https://www.anaconda.com/**).
2. Create a new Anaconda environment for Python 3.7 (here called **cr37**). Run in your Terminal window:

```
$ conda create -n cr37 python=3.7
```

3. Activate your environment with

```
$ source activate cr37 , or
$ conda  activate cr37
```

4. Install CommonRoad-io with the command:

```
$ pip install commonroad-io
```

5. Install Jupyter Notebook with the command:

```
$ conda install jupyter
```

Tutorial

1. Open Terminal window at the root directory.
2. Activate your environment with

```
$ source activate cr37 , or
$ conda  activate cr37
```

3. Open Jupyter Notebook with the command:

```
$ jupyter notebook
```

4. Navigate to the directory of the iPython-Notebook "tutorial_commonroad-io.ipynb", open it and follow the instructions.