Control theory for autonomous driving

Igarashi lab
Kyoto University, 11th Dec. 2019

Jérémy Dubut
National Institute of Informatics
Japanese-French Laboratory of Informatics

This work is supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST;
and by Grant-in-aid No. 19K20215, JSPS.

References

- S. Pruekprasert, J. Dubut, X. Zhang, C. Huang, M. Kishida. A Game Theoretic Approach to Decision Making for Multiple Vehicles at Roundabout. arXiv:1904.06224. (Submitted to ACC)

- S. Pruekprasert, X. Zhang, J. Dubut, C. Huang, M. Kishida. Decision Making for Autonomous Vehicles at Unsignalized Intersection in Presence of Malicious Vehicles. *In* ITSC 2019.

- S. Pruekprasert, T. Takisaka, C. Eberhart, A. Cetinkaya, J. Dubut. Moment Propagation of Discrete-Time Stochastic Polynomial Systems using Truncated Carleman Linearization. arXiv:1911.12683. (Submitted to IFAC)

- S. Pruekprasert, C. Eberhart, J. Dubut, K. Hashimoto. Symbolic Approach to Self-Triggered Control. (On-going)

References

- S. Pruekprasert, J. Dubut, X. Zhang, C. Huang, M. Kishida. A Game Theoretic Approach to Decision Making for Multiple Vehicles at Roundabout. arXiv:1904.06224. (Submitted to ACC)

Decision making

- S. Pruekprasert, X. Zhang, J. Dubut, C. Huang, M. Kishida. Decision Making for Autonomous Vehicles at Unsignalized Intersection in Presence of Malicious Vehicles. *In* ITSC 2019.

- S. Pruekprasert, T. Takisaka, C. Eberhart, A. Cetinkaya, J. Dubut. Moment Propagation of Discrete-Time Stochastic Polynomial Systems using Truncated Carleman Linearization. arXiv:1911.12683. (Submitted to IFAC)

- S. Pruekprasert, C. Eberhart, J. Dubut, K. Hashimoto. Symbolic Approach to Self-Triggered Control. (On-going)

References

- S. Pruekprasert, J. Dubut, X. Zhang, C. Huang, M. Kishida. A Game Theoretic Approach to Decision Making for Multiple Vehicles at Roundabout. arXiv:1904.06224. (Submitted to ACC)

Decision making

- S. Pruekprasert, X. Zhang, J. Dubut, C. Huang, M. Kishida. Decision Making for Autonomous Vehicles at Unsignalized Intersection in Presence of Malicious Vehicles. *In* ITSC 2019.

- S. Pruekprasert, T. Takisaka, C. Eberhart, A. Cetinkaya, J. Dubut. Moment Propagation of Discrete-Time Stochastic Polynomial Systems using Truncated Carleman Linearization. arXiv:1911.12683. (Submitted to IFAC)

Reachability analysis

- S. Pruekprasert, C. Eberhart, J. Dubut, K. Hashimoto. Symbolic Approach to Self-Triggered Control. (On-going)

References

- S. Pruekprasert, J. Dubut, X. Zhang, C. Huang, M. Kishida. A Game Theoretic Approach to Decision Making for Multiple Vehicles at Roundabout. arXiv:1904.06224. (Submitted to ACC)

Decision making

- S. Pruekprasert, X. Zhang, J. Dubut, C. Huang, M. Kishida. Decision Making for Autonomous Vehicles at Unsignalized Intersection in Presence of Malicious Vehicles. *In* ITSC 2019.

- S. Pruekprasert, T. Takisaka, C. Eberhart, A. Cetinkaya, J. Dubut. Moment Propagation of Discrete-Time Stochastic Polynomial Systems using Truncated Carleman Linearization. arXiv:1911.12683. (Submitted to IFAC)

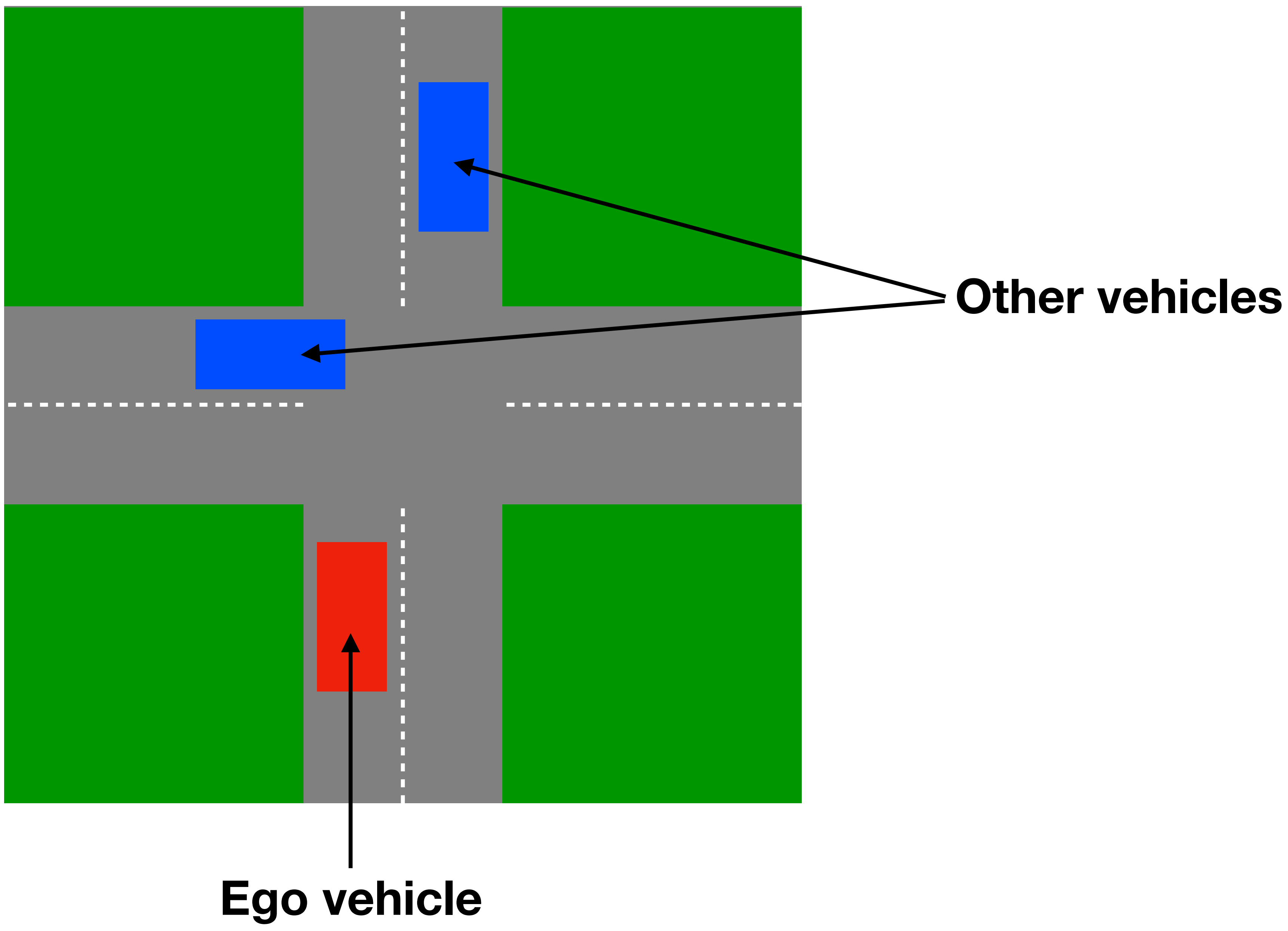
Reachability analysis

- S. Pruekprasert, C. Eberhart, J. Dubut, K. Hashimoto. Symbolic Approach to Self-Triggered Control. (On-going)

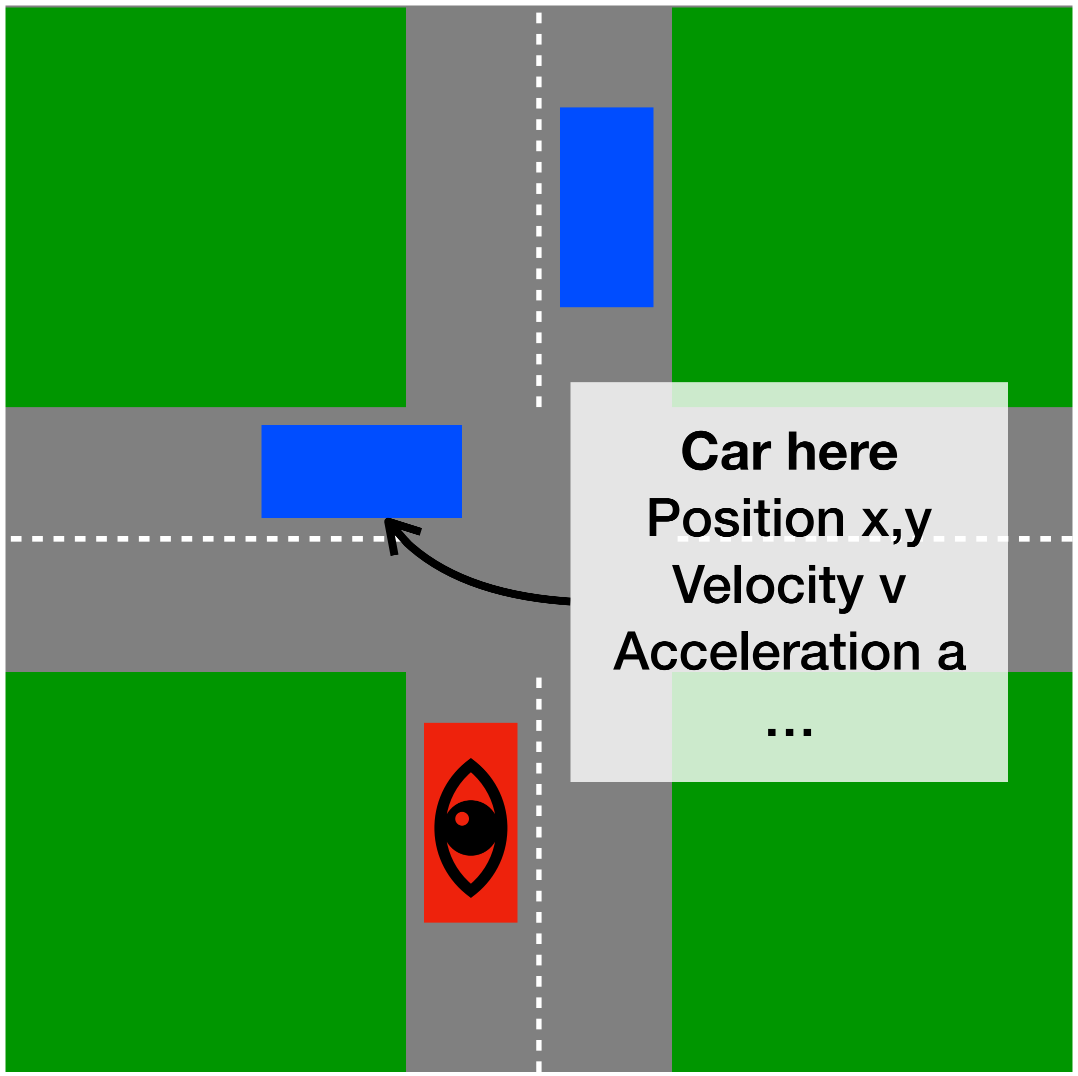
Control Synthesis

Nash equilibria for decision making of autonomous vehicles

Decision making?

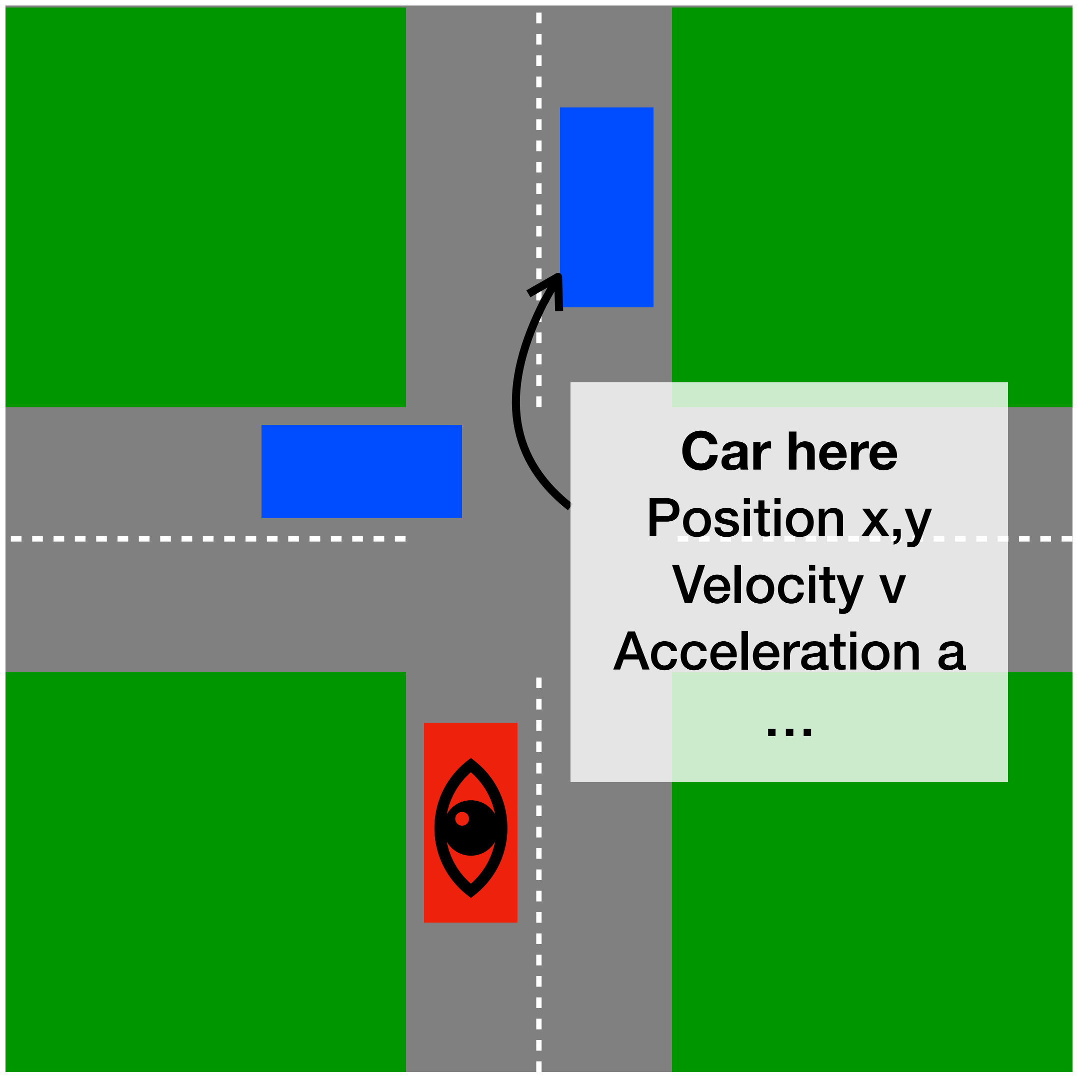


Decision making?



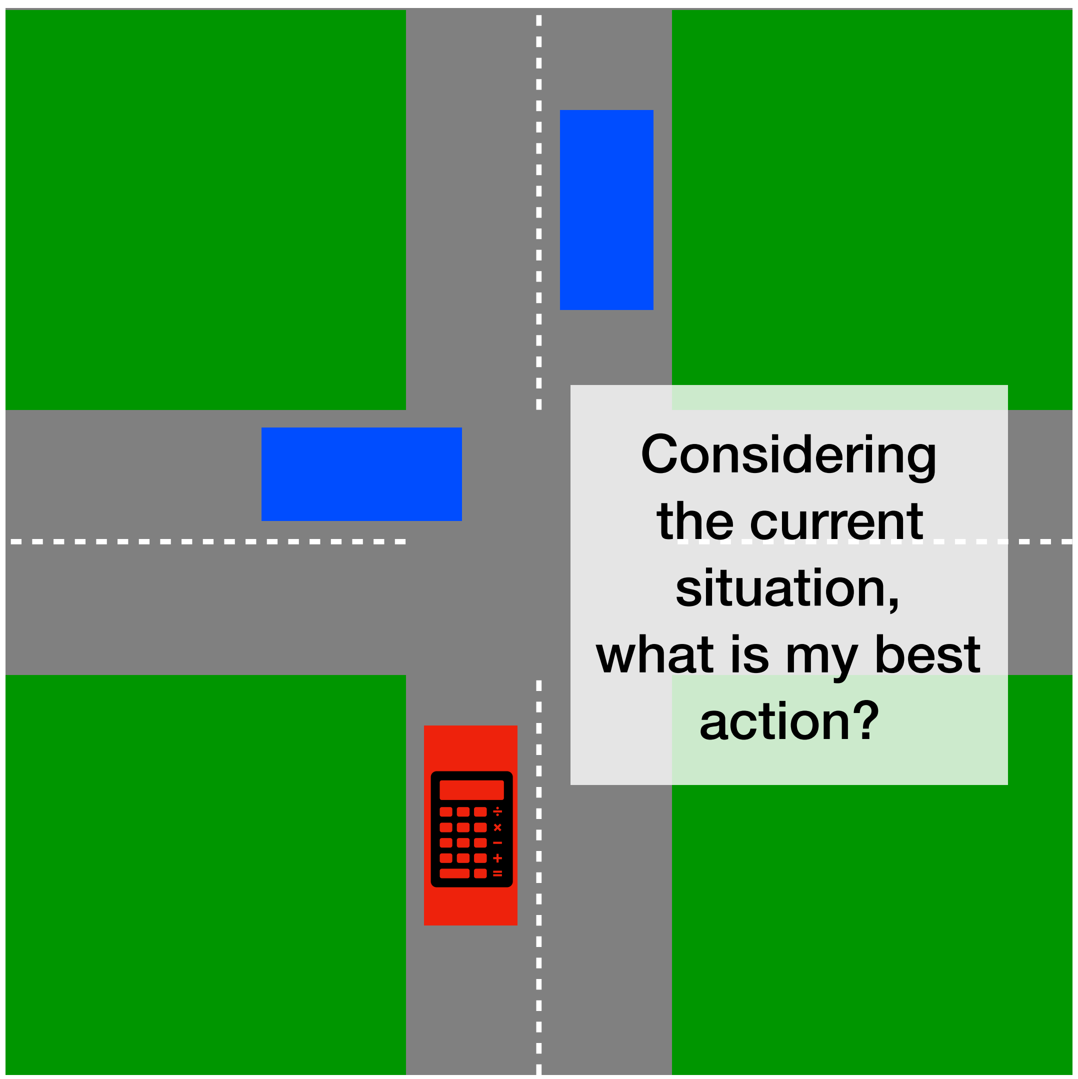
Step 1: Observation

Decision making?



Step 1: Observation

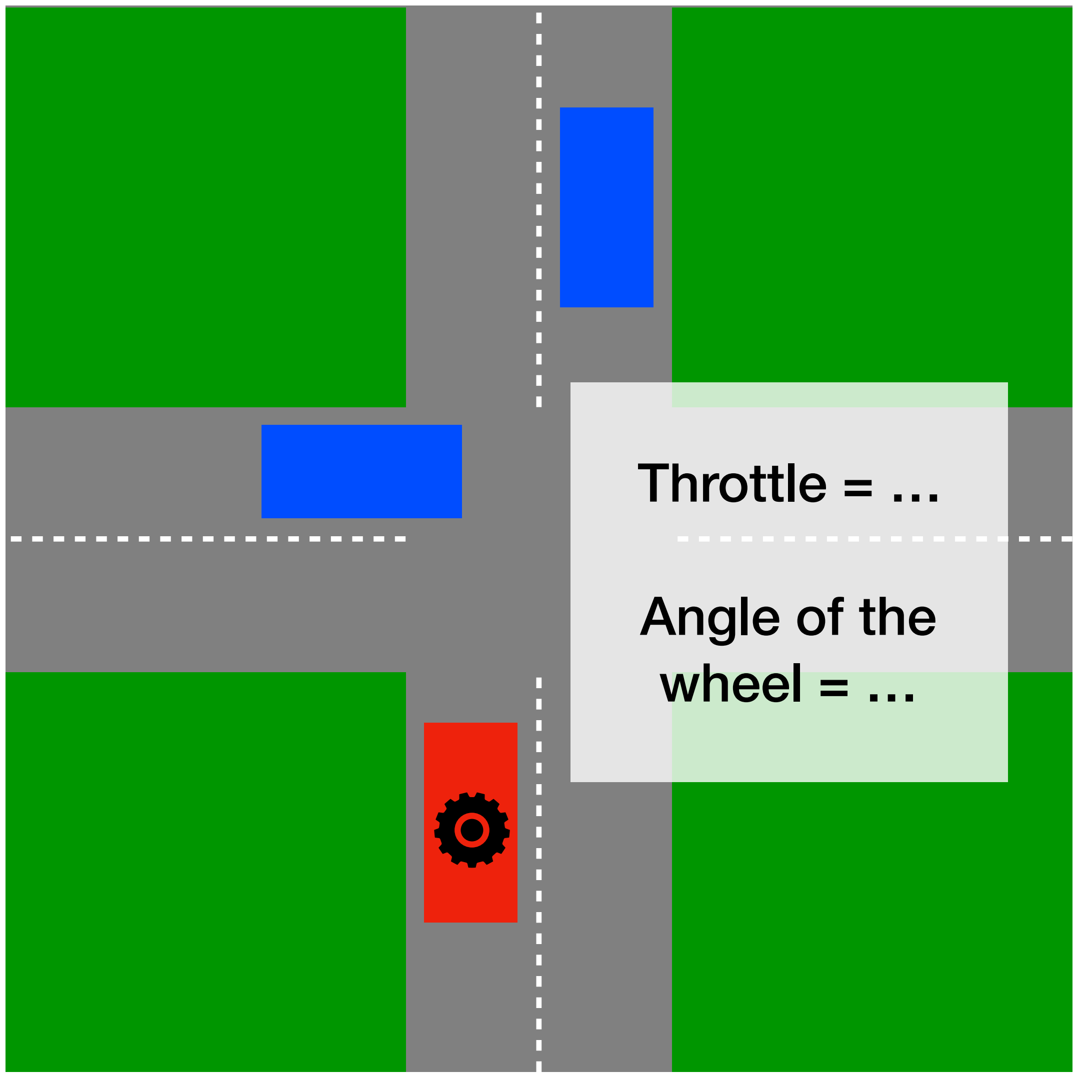
Decision making?



Step 1: Observation

Step 2: Computation

Decision making?

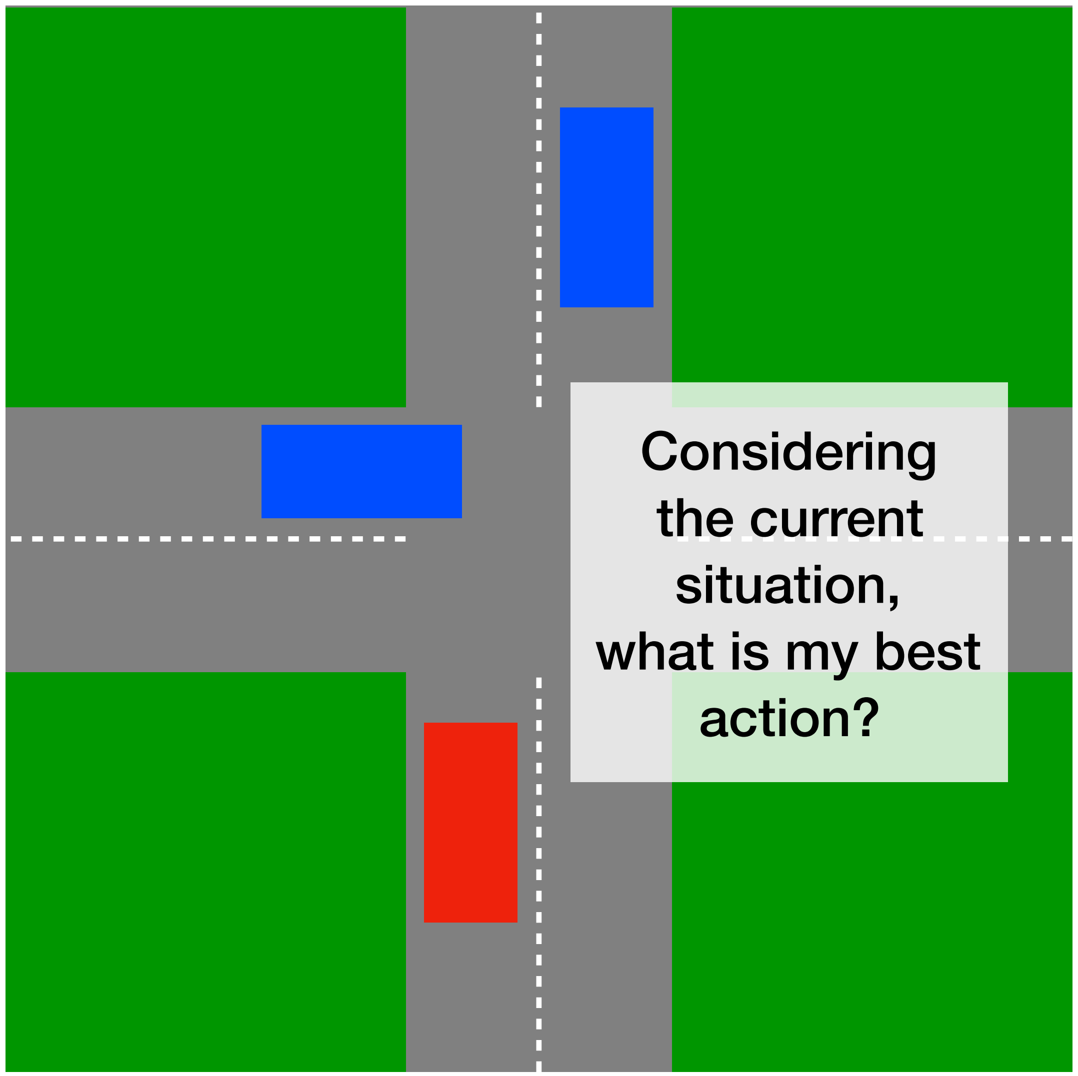


Step 1: Observation

Step 2: Computation

Step 3: Actuation

Decision making?

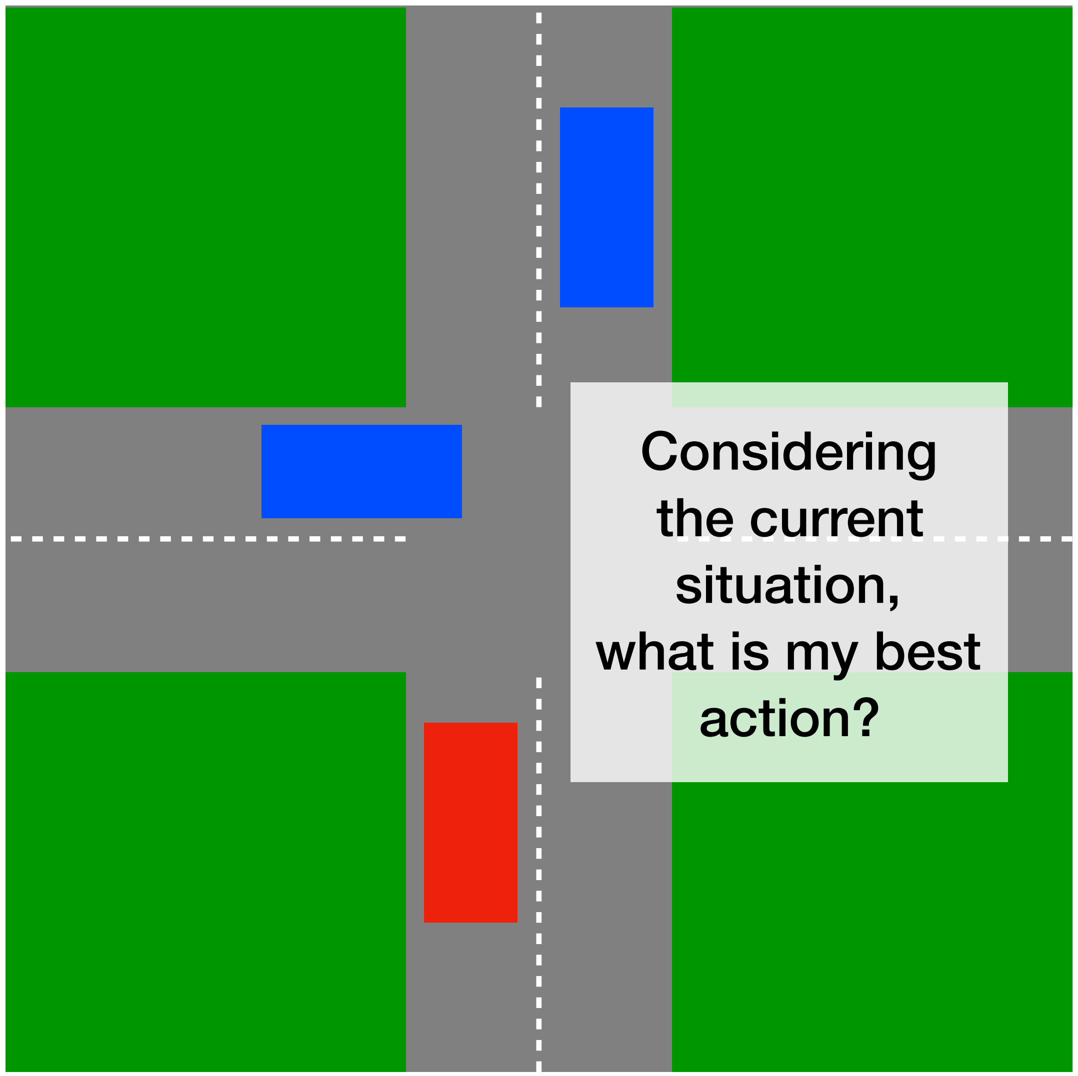


Step 1: Observation

Step 2: Computation

Step 3: Actuation

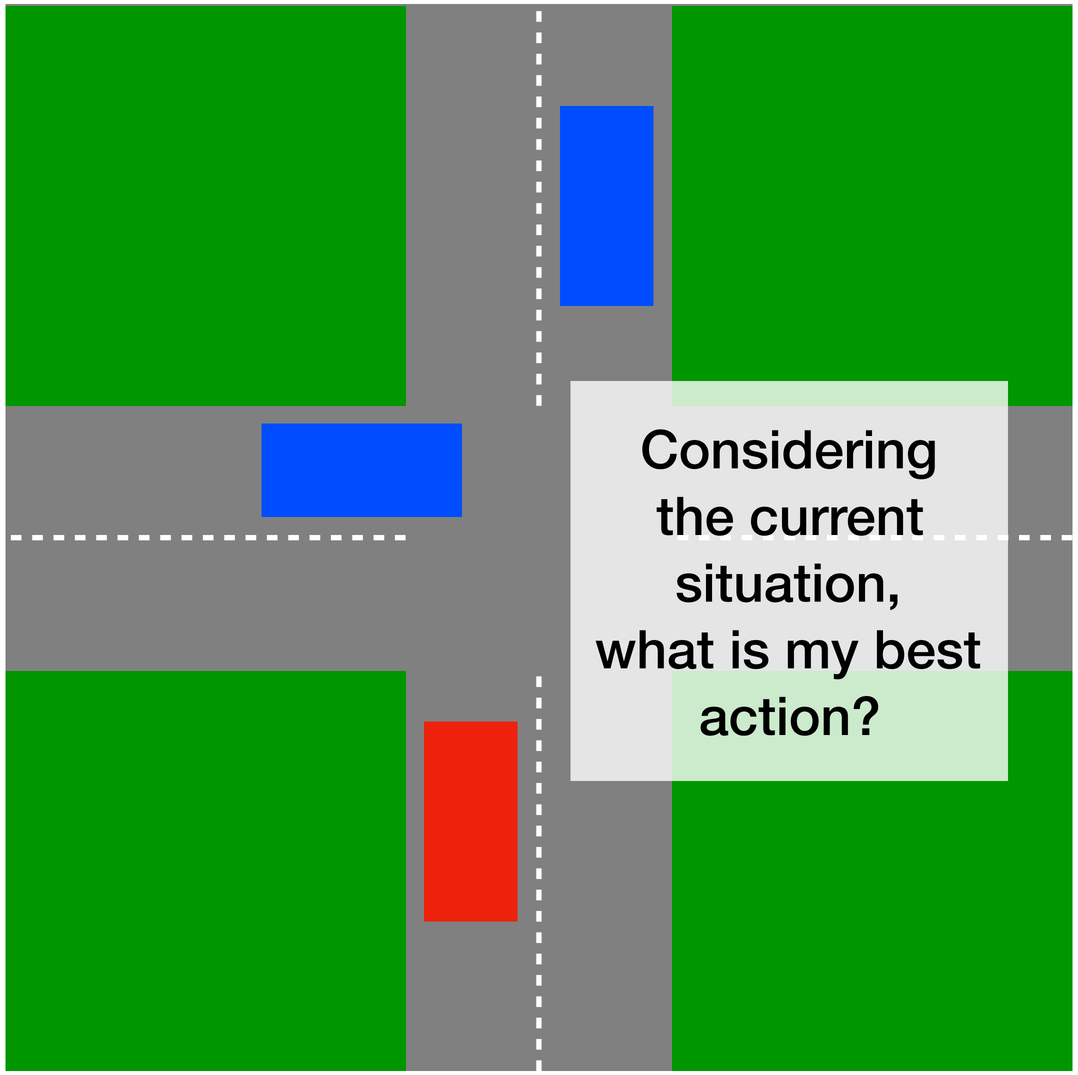
Two main methods



Two main methods:

- Learning methods
- Game theoretic methods

Two main methods



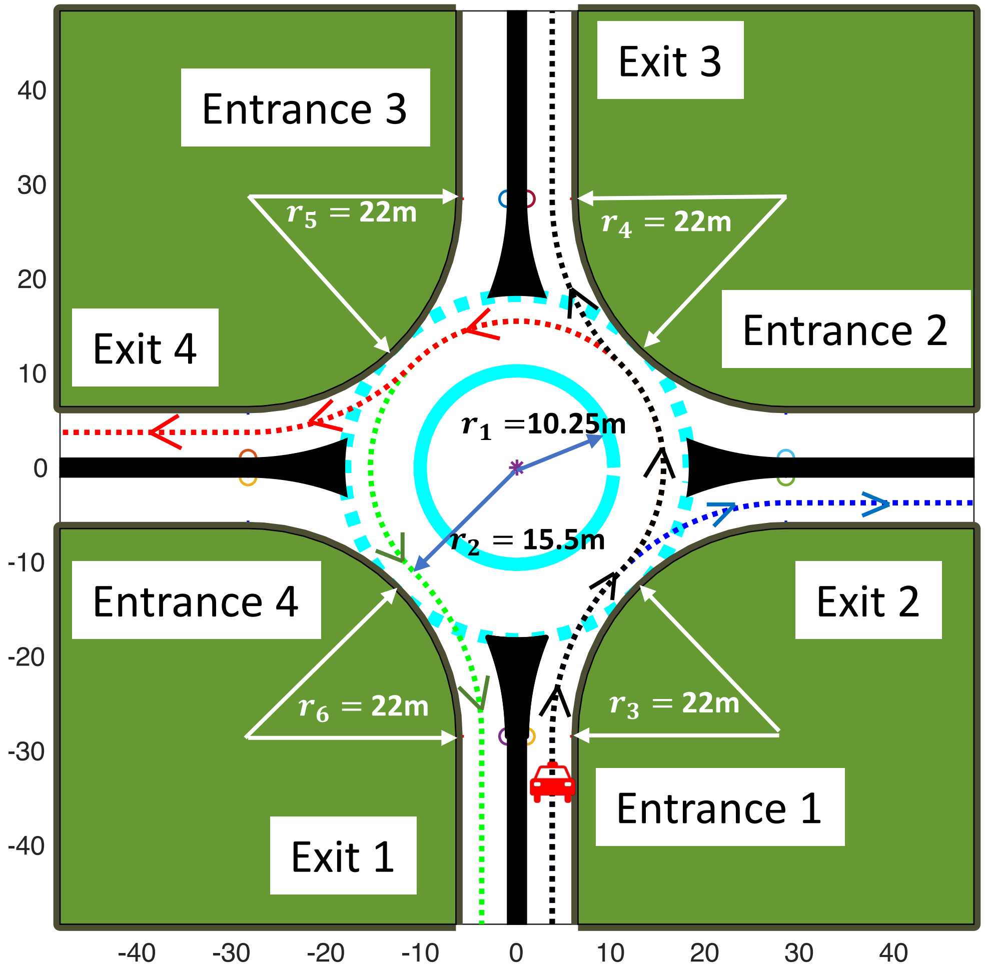
Two main methods:

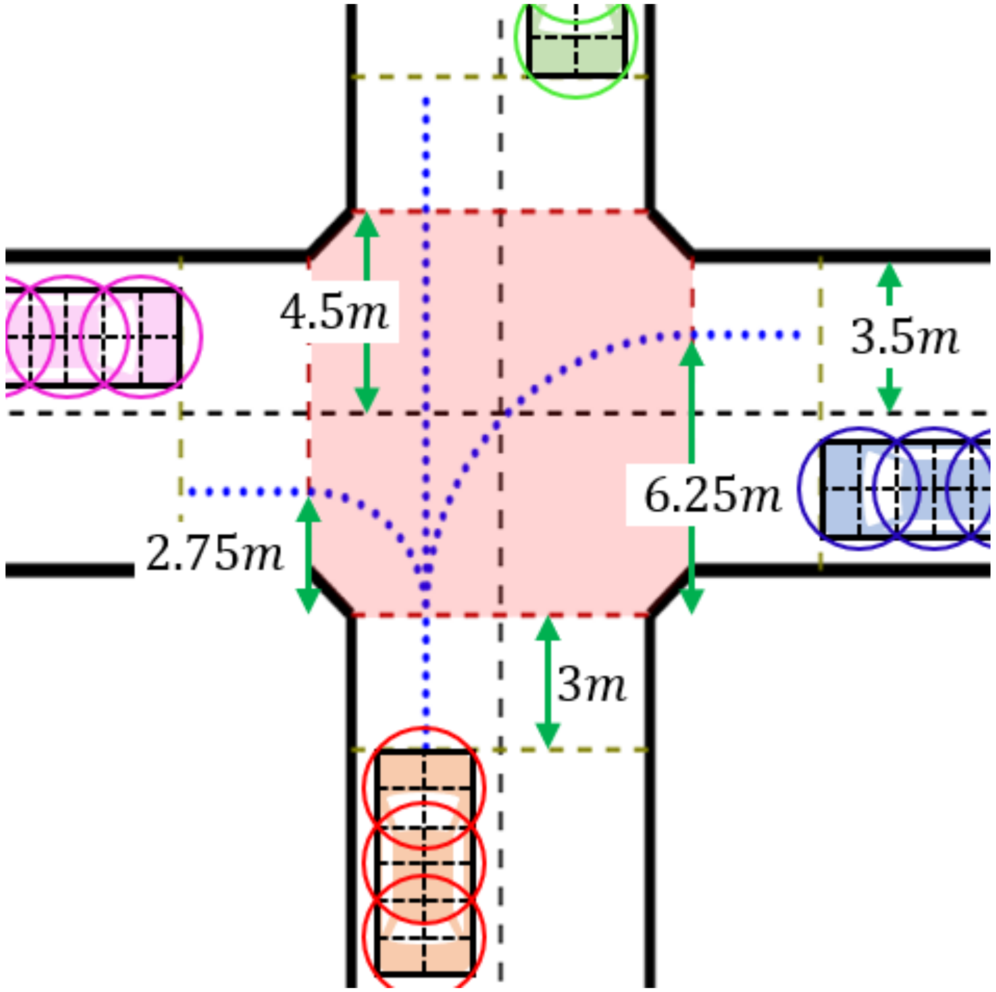
- Learning methods
- Game theoretic methods

Li et al., "Game Theoretic Modeling of Vehicle Interactions at Unsignalized Intersections and Application to Autonomous Vehicle Control" IEEE-ACC2018.

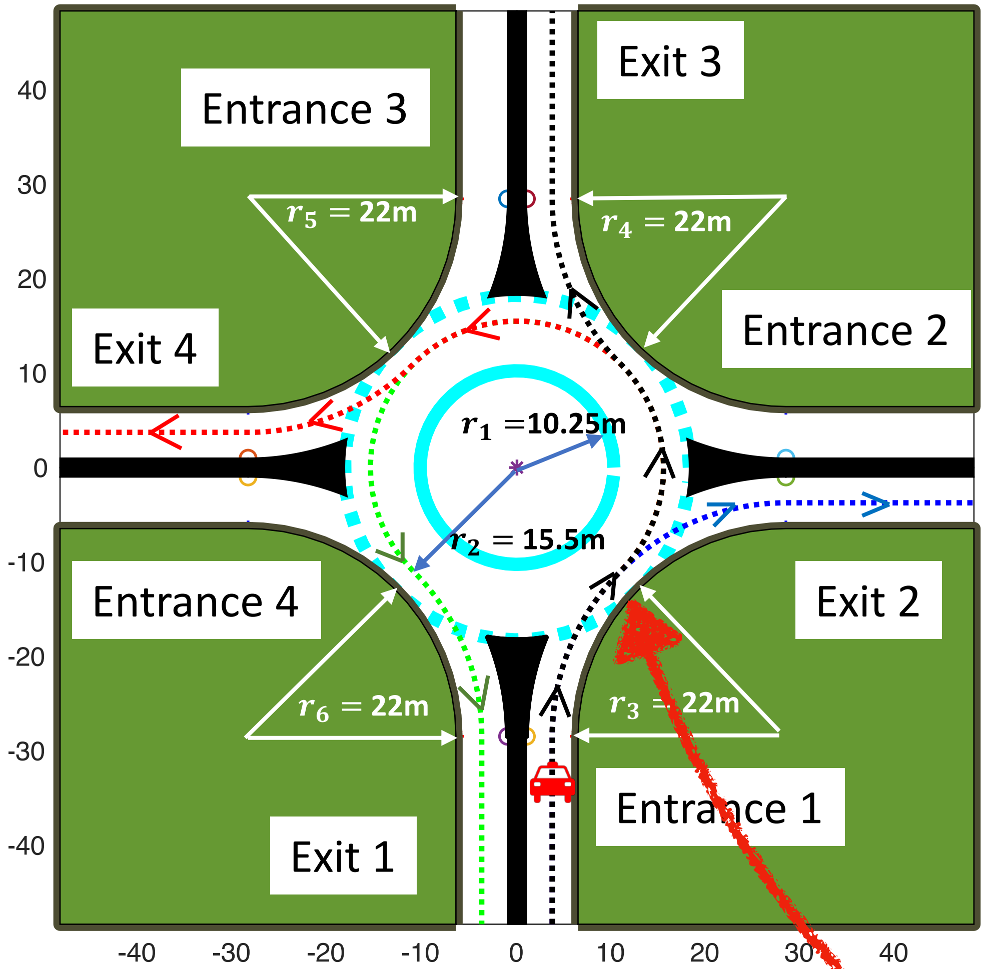
Tian et al., "Adaptive Game-Theoretic Decision Making for Autonomous Vehicle Control at Roundabouts" IEEE-CDC2018.

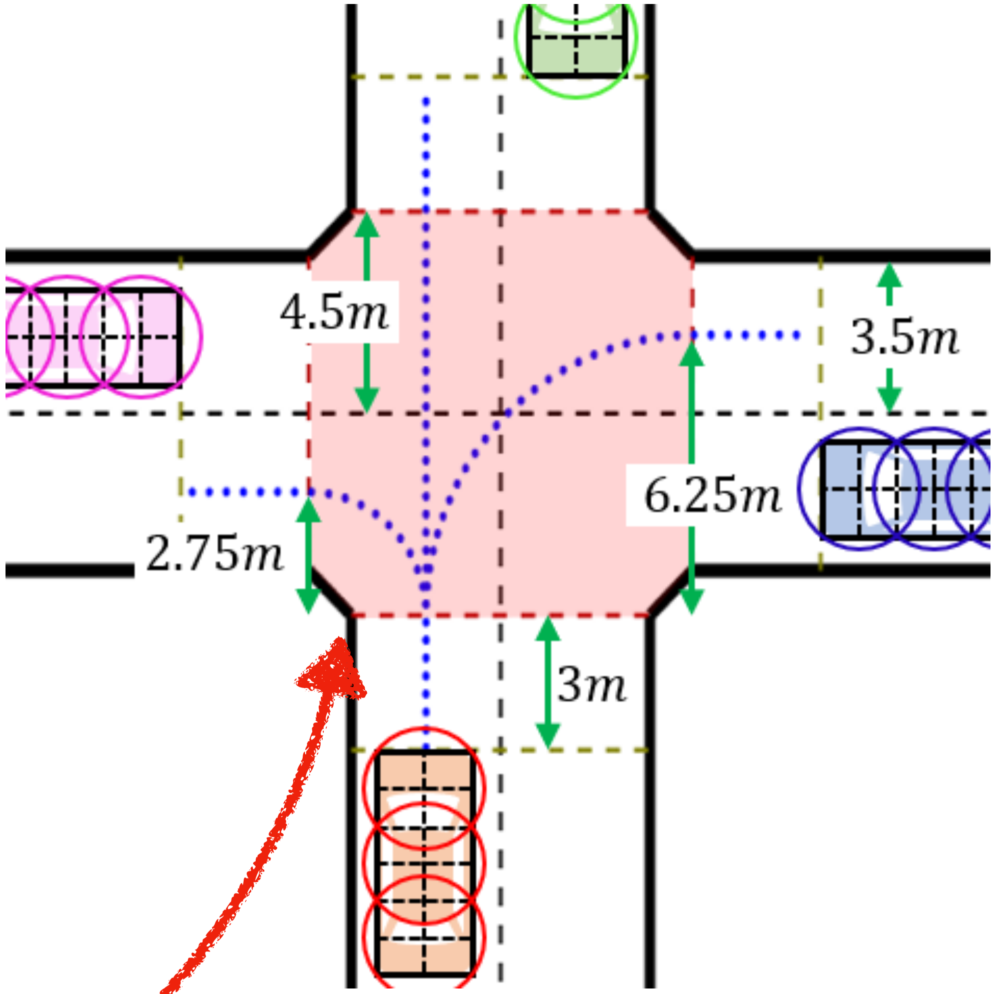
Which scenarii?





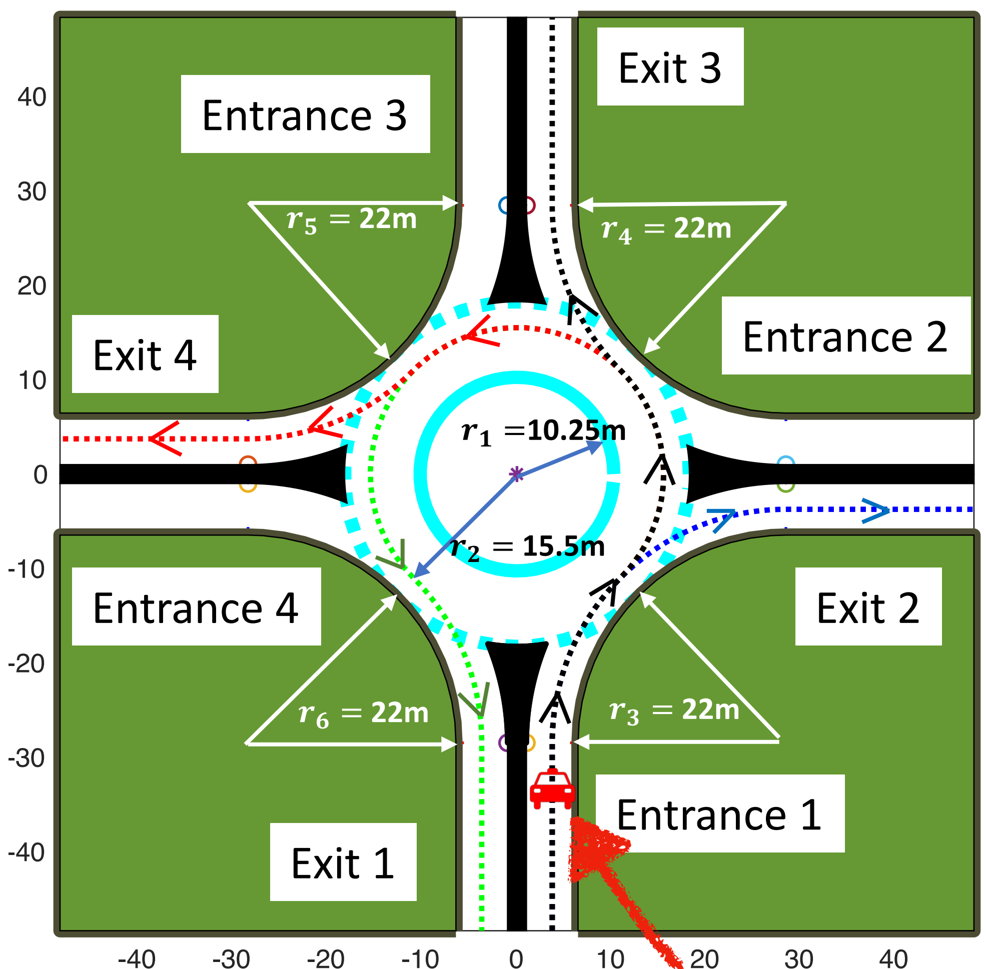
Which scenarii?

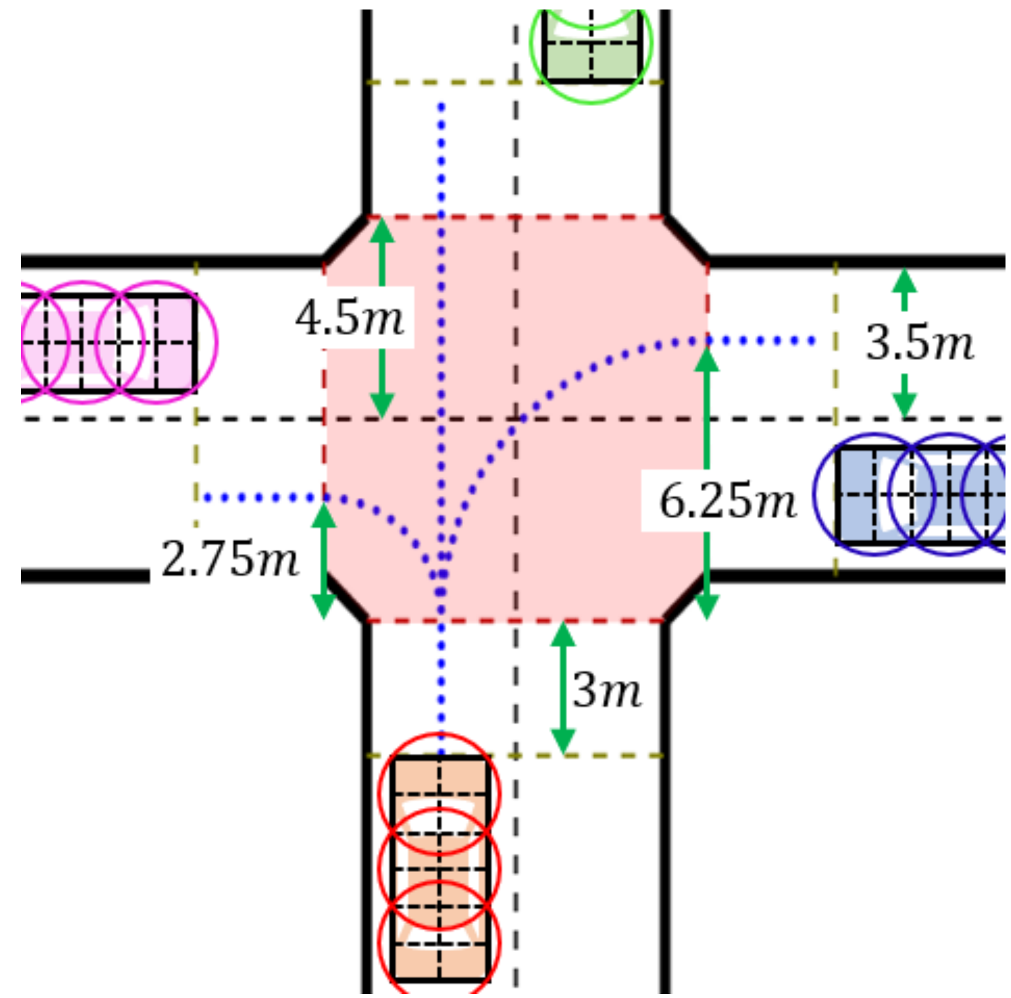




Fixed intersection-like road situations

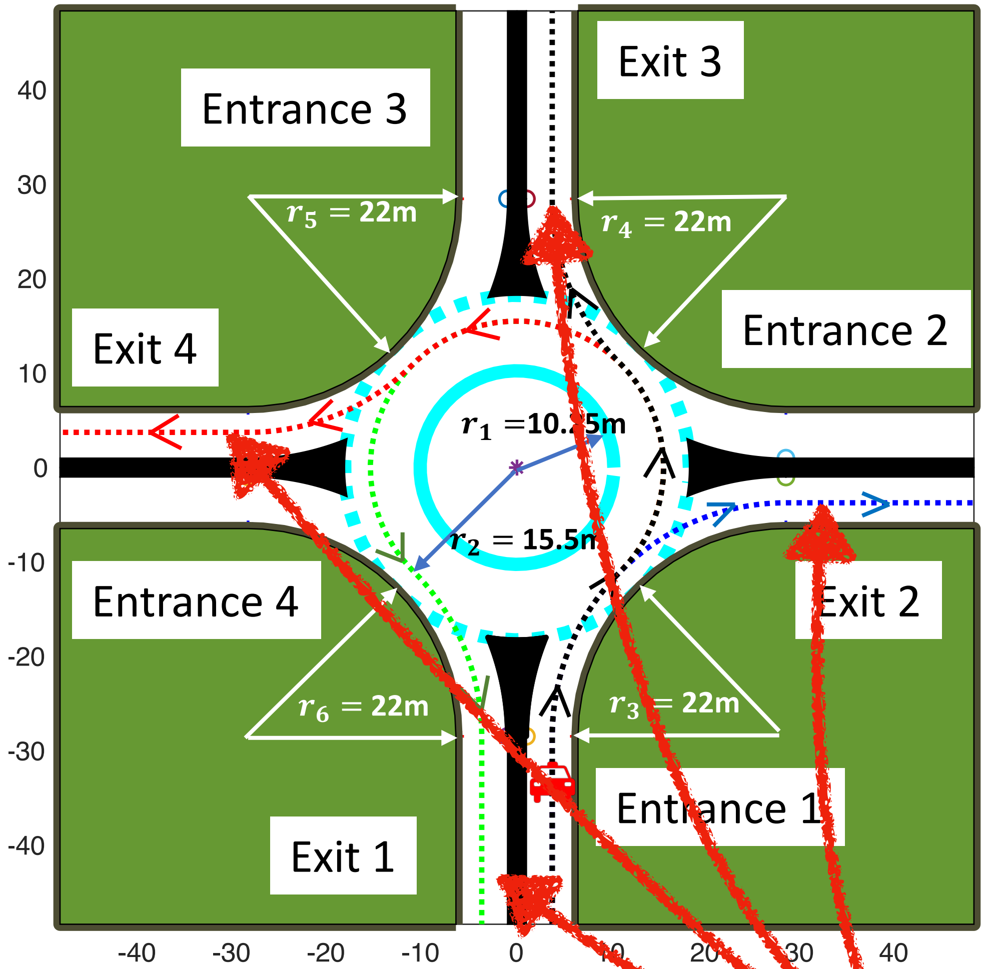
Which scenarii?

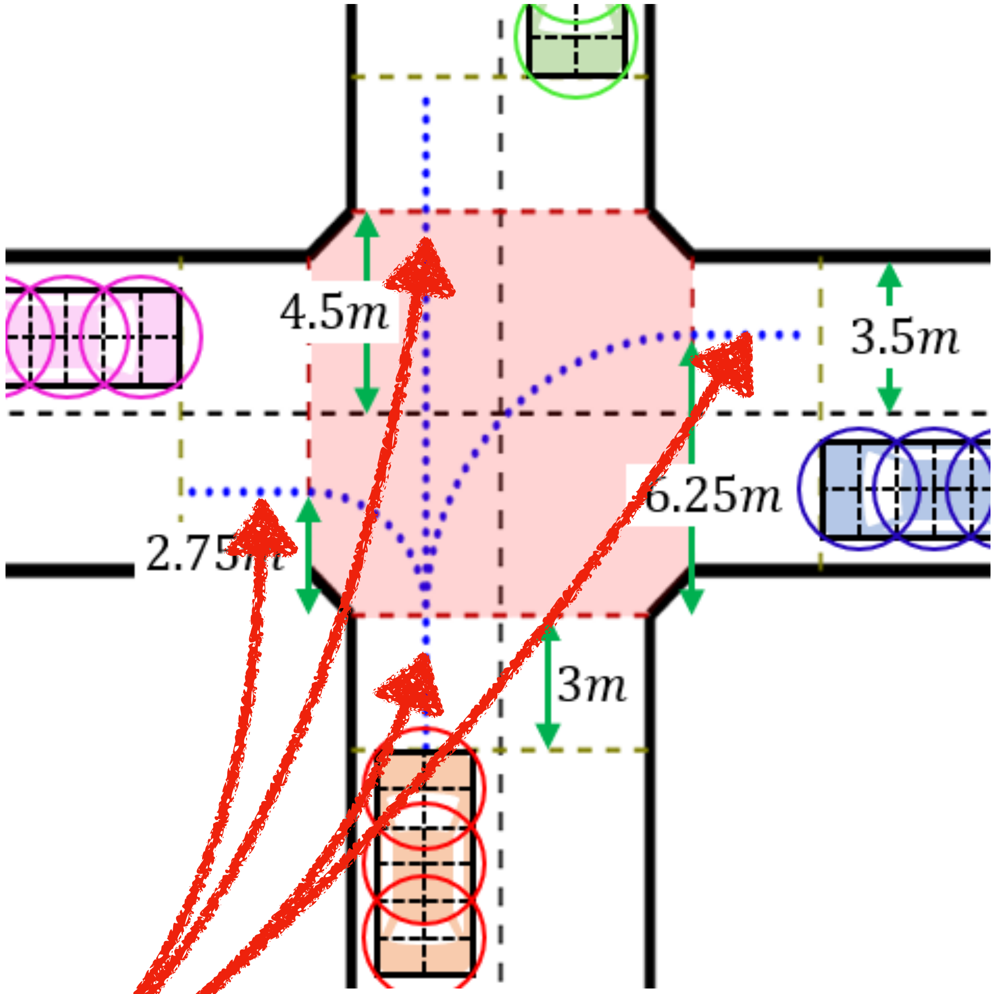




Autonomous vehicles « playing » with each other or with others types of vehicles

Which scenarii?





Their paths are fixed

$\textbf{NextStep}_i : \textbf{conf} * a \longrightarrow \textbf{conf}$

Best move?

The ego vehicle must choose its acceleration in such a way that:

- it follows its path,
- it optimises its time in the intersection,
- it does not collide with other vehicles,
- it respects the law,
- …

Since we have several things to optimise, we have to think about trade-off.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$

$$\text{with } \alpha_i + \beta_i = 1$$

Cost function

$$\boxed{\textbf{Cost}^i}(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$
$$\text{with } \alpha_i + \beta_i = 1$$

The cost of a situation (either observed or predicted)
The lower, the better.

Cost function

$$\mathbf{Cost}^i(\boxed{\mathbf{conf}_1}, \ldots, \boxed{\mathbf{conf}_n}) = \alpha_i \,.\, \mathbf{Cost}^i_{\mathbf{velo}} + \beta_i \,.\, \mathbf{Cost}^i_{\mathbf{safe}}$$

with $\alpha_i + \beta_i = 1$

Configurations of the vehicle $1, \ldots, n$ given by:

- Their positions (cartesian, polar coordinates)
- Their velocities
- Their situations in the intersection (entering, inside, exiting)

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \boxed{\textbf{Cost}^i_{\textbf{velo}}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$
$$\text{with } \alpha_i + \beta_i = 1$$

A cost function that describes how close to the limit velocity the vehicle is.
The lower, the closer.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \boxed{\textbf{Cost}^i_{\textbf{safe}}}$$

$$\text{with } \alpha_i + \beta_i = 1$$

A cost function that describes how safe the vehicle is
Roughly speaking: how closed to the other vehicle it is
The lower, the safer.

Cost function

$$\textbf{Cost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n) = \alpha_i \,.\, \textbf{Cost}^i_{\textbf{velo}} + \beta_i \,.\, \textbf{Cost}^i_{\textbf{safe}}$$

$$\text{with } \alpha_i + \beta_i = 1$$

Coefficient that describes how much the vehicle cares about the optimising its velocity compared to its safety.
Roughly speaking: its an "aggressiveness" coefficient

Receding horizon cost

$$\textbf{HCost}^i(\textbf{conf}_1, \ldots, \textbf{conf}_n, (a^i_{j,s})_{1 \leq j \leq n, 0 \leq s \leq h}) = \sum_{s=0}^{h} \delta^h \,.\, \textbf{Cost}^i(\textbf{conf}_{1,s}, \ldots, \textbf{conf}_{n,s})$$

with:

$$\textbf{conf}_{j,0} = \textbf{conf}_j$$

$$\textbf{conf}_{j,s+1} = \textbf{NextStep}^i_j(\textbf{conf}_{j,s}, a^i_{j,s})$$

Receding horizon cost

$$\boxed{\mathbf{HCost}^i(\mathbf{conf}_1, \ldots, \mathbf{conf}_n, (a^i_{j,s})_{1 \le j \le n, 0 \le s \le h})} = \sum_{s=0}^{h} \delta^h \,.\, \mathbf{Cost}^i(\mathbf{conf}_{1,s}, \ldots, \mathbf{conf}_{n,s})$$

with:

$$\mathbf{conf}_{j,0} = \mathbf{conf}_j$$

$$\mathbf{conf}_{j,s+1} = \mathbf{NextStep}^i_j(\mathbf{conf}_{j,s}, a^i_{j,s})$$

Accumulated cost of the all upcoming situations when the initial situation is

$\mathbf{conf}_1, \ldots, \mathbf{conf}_n$

and if the vehicle i predicts the accelerations of all vehicles are given by

$(a^i_{j,s})_{1 \le j \le n, 0 \le s \le h}$

and that the vehicle j follows the path given by

$\mathbf{NextStep}^i_j$

Best global move = Nash equilibrium

A **game**:

- A set of **players** $P = \{1,\ldots,n\}$
  - Ex: the vehicles
- Each player i has a set of **possible moves** Γ_i
  - Ex: an acceleration profile $(a_s)_{0\leq s\leq h}$
- Each player has a cost function it wants to minimise, of type:

$$H_i : \Gamma_1 \times \ldots \times \Gamma_n \to \mathbb{R}$$

  - Ex: the accumulated costs

What does it mean for the players to conjunctly optimise their cost?

$\Rightarrow$ best possible response: a move m_i for every player such that for any other move m_i':

$$H_i(m_1, \ldots, m_n) \leq H_i(m_1, \ldots, m_i', \ldots, m_n)$$

Nash equilibrium

Priority order and backward induction

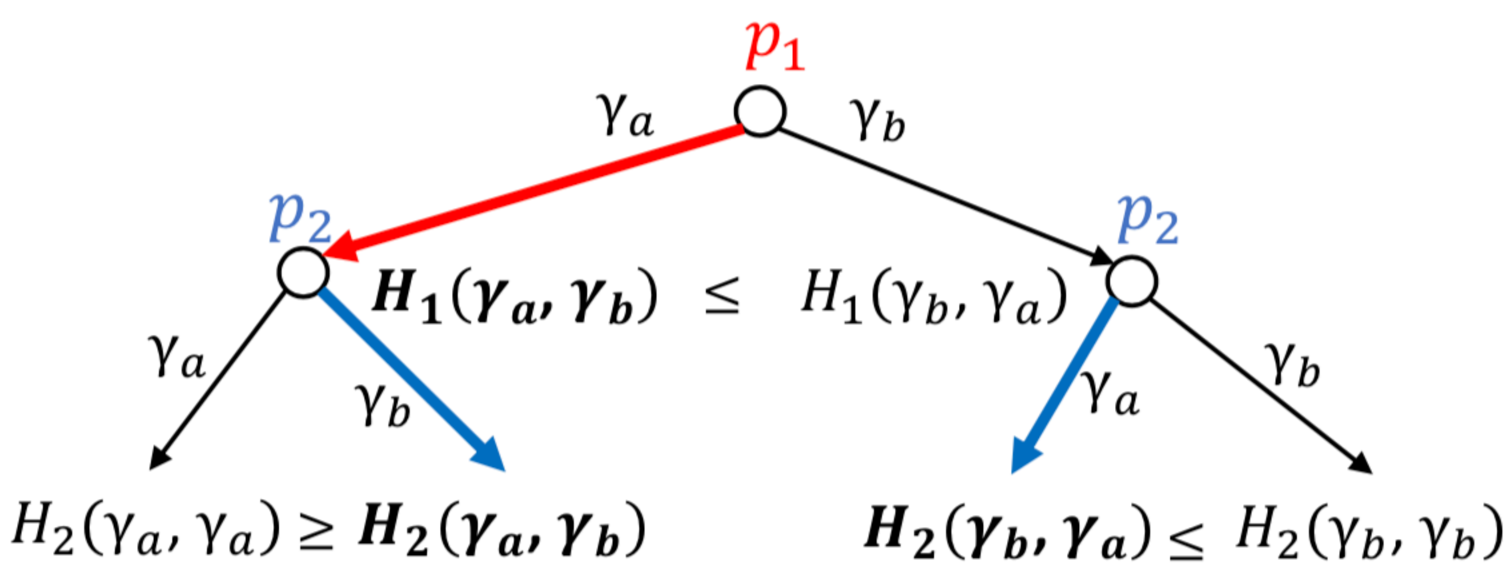
How to enforce the existence of a Nash equilibrium and compute it?

Idea: order the players, the smallest one chooses first, the second smallest chooses second, …

Assume: a total order $\preceq$ on P

Ex: $i \preceq j$ if i is more aggressive than j, or if the law tells that i has priority over j

Do backward induction:



Decision making procedure

Algorithm Decision making of the ego vehicle

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
  $X_j :=$ observe(j, t);
  NextStep$_j :=$ initial_path(j);
  HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order;
while I am still in the intersection **do**
  $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j$, $\preceq$);
  return $a_{\text{ego},0}$ as control input;
  $t := t +$ time_step;
  $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
  $X_j :=$ observe(j, t);
  if some $\widehat{X}_j$ are not close to X_j **then**
    for all $j \in N$ **do**
      NextStep$_j :=$ update_path(j);
      HCost$_j :=$ update_cost(j);
    end for;
    $\preceq :=$ update_order;
  end if
  $N :=$ update_neighbors;
end while

Decision making procedure

Algorithm Decision making of the ego vehicle

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
  $X_j :=$ observe(j, t);
  NextStep$_j :=$ initial_path(j);
  HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order;
while I am still in the intersection **do**
  $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
  return $a_{\mathrm{ego},0}$ as control input;
  $t := t +$ time_step;
  $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
  $X_j :=$ observe(j, t);
  if some $\widehat{X}_j$ are not close to X_j **then**
    for all $j \in N$ **do**
      NextStep$_j :=$ update_path(j);
      HCost$_j :=$ update_cost(j);
    end for;
    $\preceq :=$ update_order;
  end if
  $N :=$ update_neighbors;
end while

Initialization phase

Restrict the set of vehicles you are considering

Observe their current configuration

Guess their path

Guess their cost function

Initialise the order of the vehicles

Decision making procedure

Algorithm Decision making of the ego vehicle

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
  $X_j :=$ observe(j, t);
  $\mathrm{NextStep}_j :=$ initial_path(j);
  $\mathrm{HCost}_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order;
while I am still in the intersection **do**
  $(a_{j,s})_{j,s} :=$ nash_equilibrium$(\mathrm{HCost}_j, \preceq)$;
  return $a_{\mathrm{ego},0}$ as control input;
  $t := t +$ time_step;
  $\widehat{X}_j := \mathrm{NextStep}_j(X_j, a_{j,0})$;
  $X_j :=$ observe(j, t);
  if some $\widehat{X}_j$ are not close to X_j **then**
    for all $j \in N$ **do**
      $\mathrm{NextStep}_j :=$ update_path(j);
      $\mathrm{HCost}_j :=$ update_cost(j);
    end for;
    $\preceq :=$ update_order;
  end if
  $N :=$ update_neighbors;
end while

Initialization phase

Initialize the order of the vehicles

Order the vehicles using the right-of-way:

- Vehicles already in the intersection have more priority.
- Vehicles on your left-hand-side have more priority.
- …

Decision making procedure

Algorithm Decision making of the ego vehicle

```
t := 0;
N := initial_neighbors;
for all j ∈ N do
    X_j := observe(j, t);
    NextStep_j := initial_path(j);
    HCost_j := initial_cost(j);
end for;
≼ := initial_order;
while I am still in the intersection do
    (a_{j,s})_{j,s} := nash_equilibrium(HCost_j, ≼);
    return a_{ego,0} as control input;
    t := t + time_step;
    X̂_j := NextStep_j(X_j, a_{j,0});
    X_j := observe(j, t);
    if some X̂_j are not close to X_j then
        for all j ∈ N do
            NextStep_j := update_path(j);
            HCost_j := update_cost(j);
        end for;
        ≼ := update_order;
    end if
    N := update_neighbors;
end while
```

Main loop

Compute the best responses from your guesses

Your best response is your new control input

Compute your guess of the next configurations

Observe the real next configurations

In case your guesses are far from reality, make some updates

Decision making procedure

Algorithm Decision making of the ego vehicle

```
t := 0;
N := initial_neighbors;
for all j ∈ N do
    X_j := observe(j, t);
    NextStep_j := initial_path(j);
    HCost_j := initial_cost(j);
end for;
≼ := initial_order;
while I am still in the intersection do
    (a_{j,s})_{j,s} := nash_equilibrium(HCost_j, ≼);
    return a_{ego,0} as control input;
    t := t + time_step;
    X̂_j := NextStep_j(X_j, a_{j,0});
    X_j := observe(j, t);
    if some X̂_j are not close to X_j then
        for all j ∈ N do
            NextStep_j := update_path(j);
            HCost_j := update_cost(j);
        end for;
        ≼ := update_order;
    end if
    N := update_neighbors;
end while
```

Main loop

In case your guesses are far from reality, make some updates

Compute the order that fit the situation the most:

- For every possible order, compute:

$$(a_{j,s}) = \textbf{nash\_equilibrium}(\textbf{HCost}_j, \preceq)\,.$$

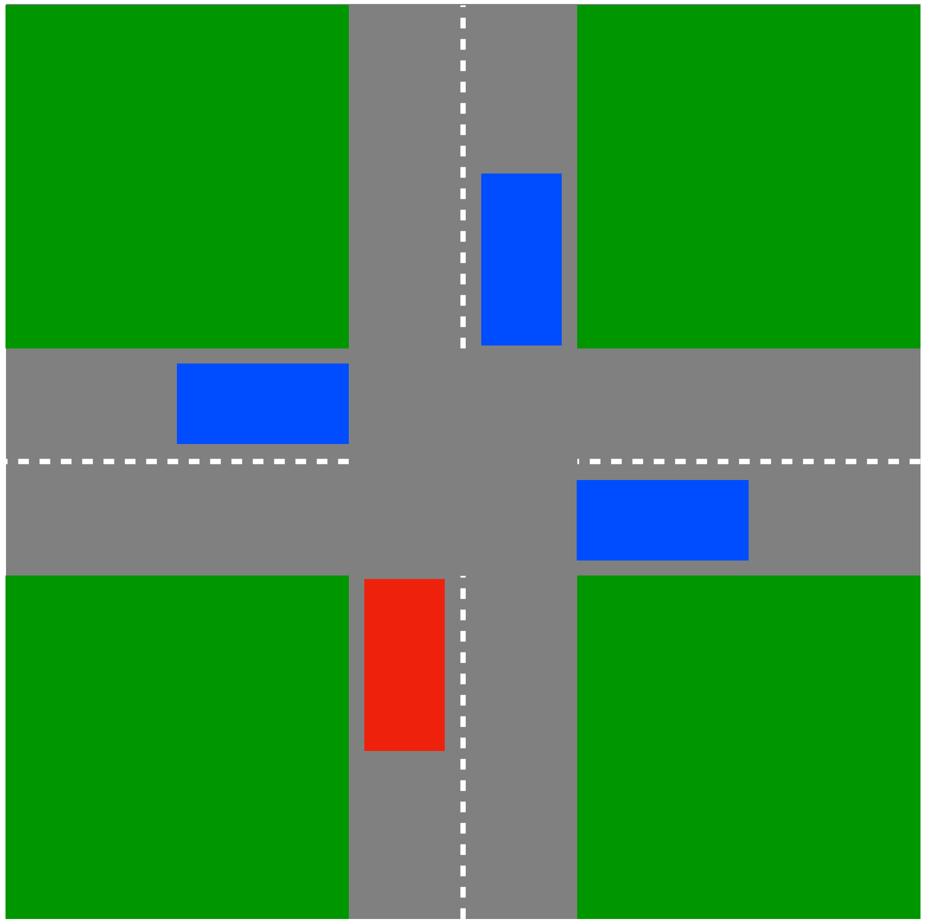
- Then compute the corresponding predicted configuration:

$$\widehat{X}_j(\preceq) = \textbf{NextStep}_j(X_j, a_{j,0})\,.$$

- Choose $\preceq$ whose predictions are the closest to the observation X_j.

Dealing with deadlocks

A **dead-lock**: a situation where all the vehicles are waiting for others to take a decision.



In our case: symmetric situations where every vehicles are stopped at the entrance of the intersection.

How to solve it?

⇒ theoretically insolvable with deterministic systems

⇒ add probabilities: when a deadlock is detected, take a decision with some probability

Behavior of the adversaries

Algorithm Decision making

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
  $X_j :=$ observe(j, t);
  NextStep$_j :=$ initial_path(j);
  HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order; → **Right of way**
while I am still in the intersection **do**
  $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
  return $a_{\text{ego},0}$ as control input;
  $t := t +$ time_step;
  $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
  $X_j :=$ observe(j, t);
  if some $\widehat{X}_j$ are not close to X_j **then**
    for all $j \in N$ **do**
      NextStep$_j :=$ update_path(j);
      HCost$_j :=$ update_cost(j);
    end for;
    $\preceq :=$ update_order; → **Fitting**
  end if
  $N :=$ update_neighbors;
end while

ego/angelic

Behavior of the adversaries

Demonic

Algorithm Decision making

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
    $X_j :=$ observe(j, t);
    NextStep$_j :=$ initial_path(j);
    HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order; → **I have priority**
while I am still in the intersection **do**
    $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j$, $\preceq$);
    return $a_{\text{ego},0}$ as control input;
    $t := t +$ time_step;
    $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
    $X_j :=$ observe(j, t);
    if some $\widehat{X}_j$ are not close to X_j **then**
        for all $j \in N$ **do**
            NextStep$_j :=$ update_path(j);
            HCost$_j :=$ update_cost(j);
        end for;
        $\preceq :=$ update_order; → **I do not care**
    end if
    $N :=$ update_neighbors;
end while

Behavior of the adversaries

Algorithm Decision making

$t := 0$;
$N :=$ initial_neighbors;
for all $j \in N$ **do**
  $X_j :=$ observe(j, t);
  NextStep$_j :=$ initial_path(j);
  HCost$_j :=$ initial_cost(j);
end for;
$\preceq :=$ initial_order;
while I am still in the intersection **do**
  $(a_{j,s})_{j,s} :=$ nash_equilibrium(HCost$_j, \preceq$);
  return $a_{\mathrm{ego},0}$ as control input;
  $t := t +$ time_step;
  $\widehat{X}_j :=$ NextStep$_j(X_j, a_{j,0})$;
  $X_j :=$ observe(j, t);
  if some $\widehat{X}_j$ are not close to X_j **then**
    for all $j \in N$ **do**
      NextStep$_j :=$ update_path(j);
      HCost$_j :=$ update_cost(j);
    end for;
    $\preceq :=$ update_order;
  end if
  $N :=$ update_neighbors;
end while

Intermediate

I have priority

Fitting

Behavior of the adversaries

Algorithm Random decision making

1: **while** I am still in the intersection **do**
2:     **choose randomly** an acceleration a
3:     **return** a as control input;
4: **end while**

Irrational

Simulation results

Roundabout

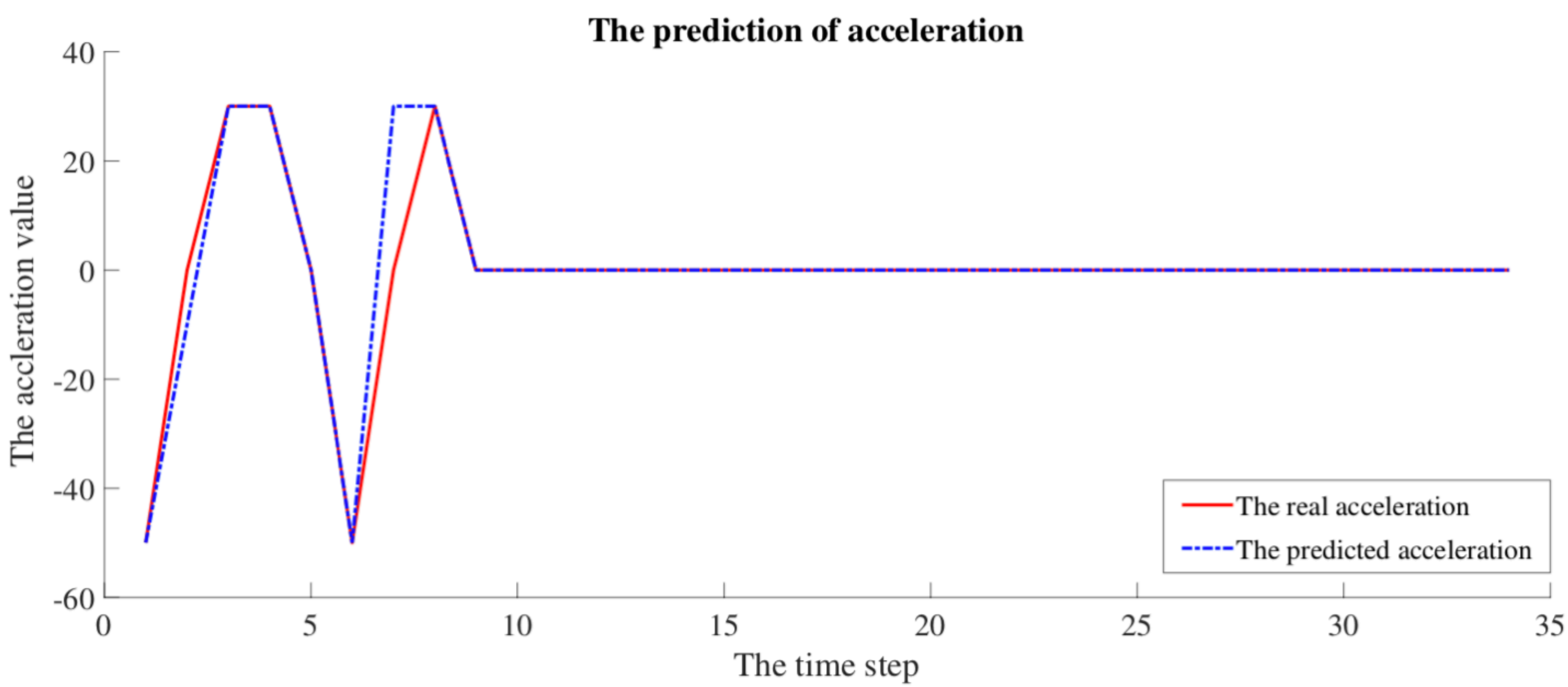
Case	Collision rate(%)	Min dist.(m)	Avg. Total time(s)
4	0	14.49	10.4
5	0	9.81	12.0
6	0	8.94	13.3
7	0	8.90	14.4
8	0	8.93	15.1

Unsignalized intersection

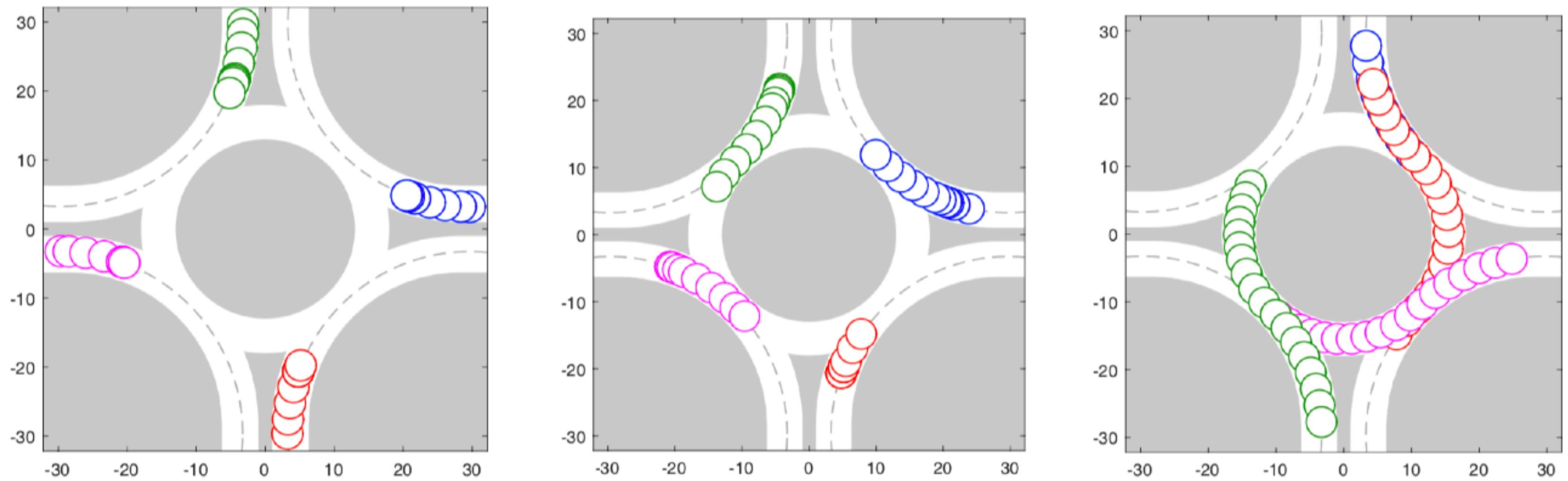
Case	Collision rate(%)	Congestion rate(%)	Avg. Total time steps
1	0	0	56.87 (5.687s)
2	0	0.2	53.98 (5.398s)
3	0	0	59.09 (5.909s)
4	0.4	4.0	91.88 (9.988s)
1’	0	0.5	55.43 (5.543s)
2’	0	1.4	50.58 (5.058s)
3’	0	9.4	55.81 (5.581s)
4’	1.1	14.3	75.82 (7.582s)

1: four angelic
2: three angelic + one demonic
3: four intermediate
4: three intermediate + one irrational

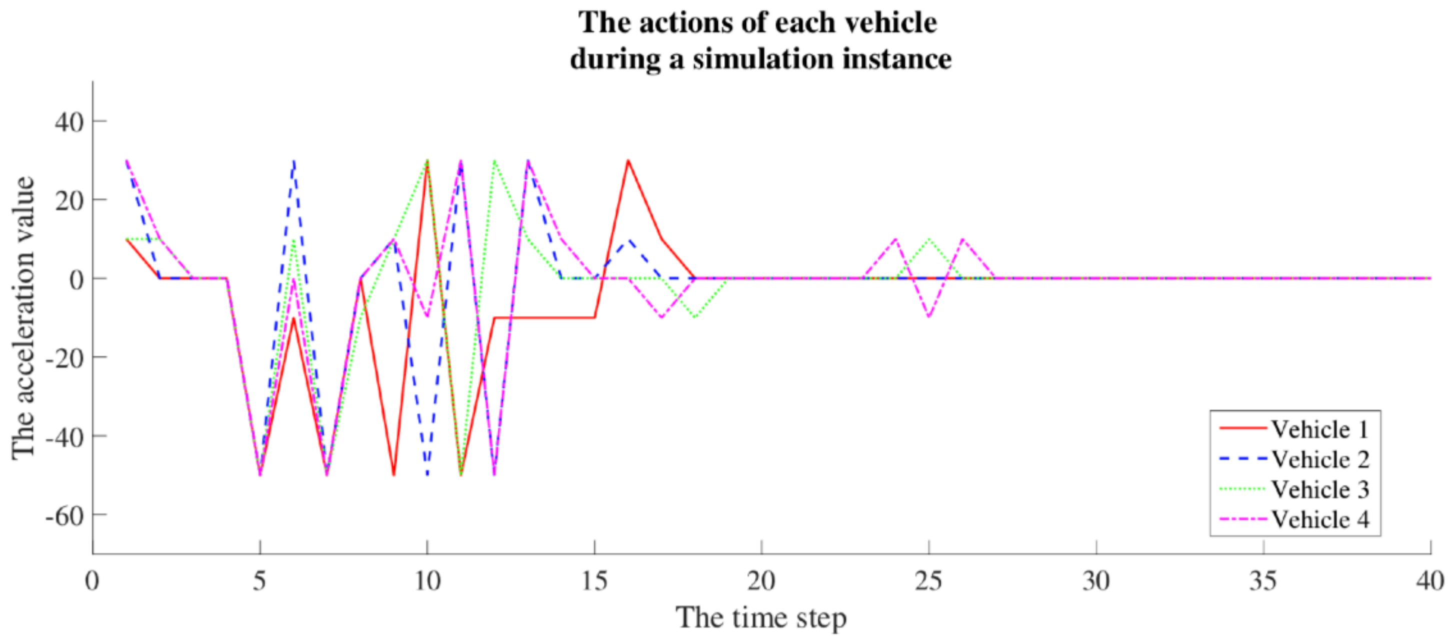
Simulation results



Simulation results



(a) time step ≤ 6 (b) $7 \leq$ time step ≤ 15 (c) time step ≥ 16



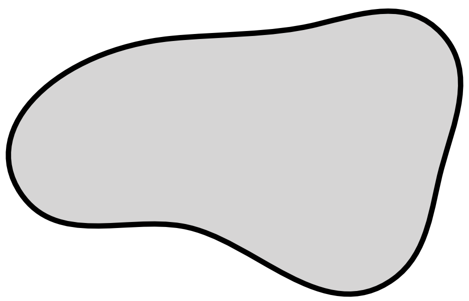
Future work

- Having a baseline experiment with human drivers?
- Going to Bayesian games?
- Using learning methods?
- Proving some guarantees?

Reachability analysis for stochastic systems

Reachability analysis?

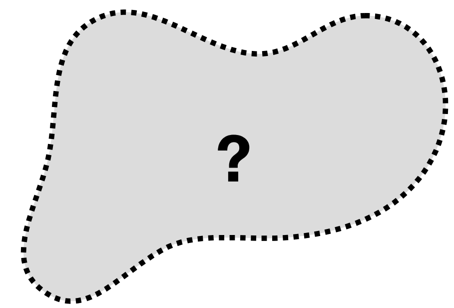
Reachability analysis?



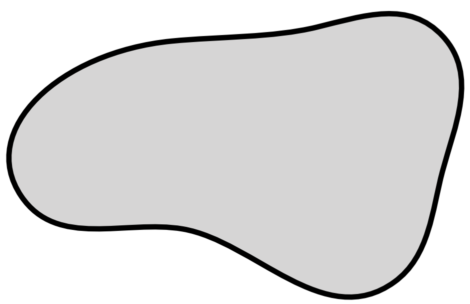
Assume your car is in this zone
and its dynamics is given by

$$\dot{X} = f(X)$$

Reachability analysis?



Compute a zone where the car is guaranteed to be in Δt time



Assume your car is in this zone and its dynamics is given by

$$\dot{X} = f(X)$$

One state-of-the-art's method

M. Althoff, J. M. Dolan, "Online Verification of Automated Road Vehicles Using Reachability Analysis", IEEE Transactions on Robotics, 2014.

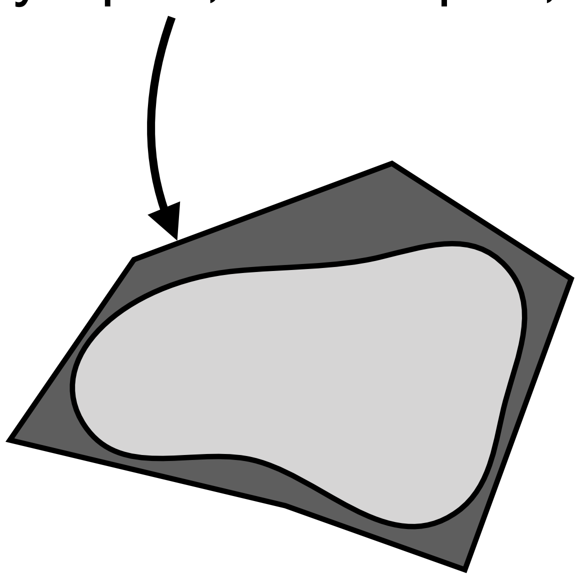
Assume your car is in this zone and its dynamics is given by

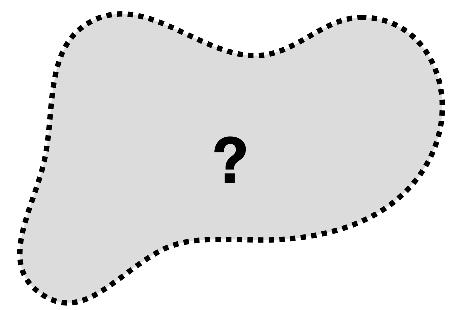
$$\dot{X} = f(X)$$

One state-of-the-art's method

M. Althoff, J. M. Dolan, "Online Verification of Automated Road Vehicles Using Reachability Analysis", IEEE Transactions on Robotics, 2014.

Over-approximate the initial zone
with polytopes, zonotopes, …



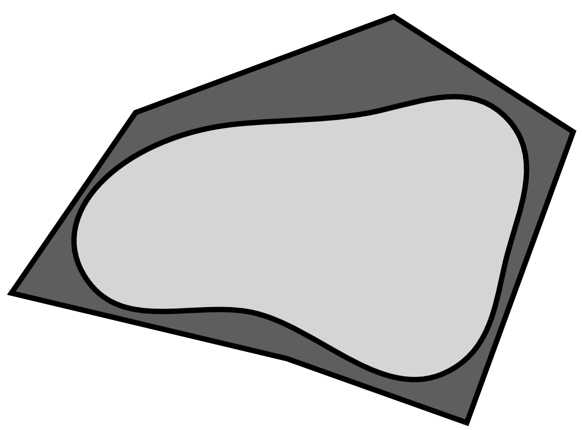


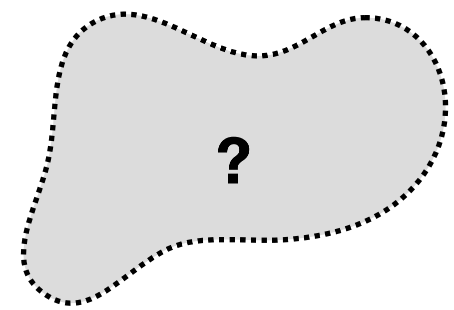
Assume your car is in this zone
and its dynamics is given by

$$\dot{X} = f(X)$$

One state-of-the-art's method

M. Althoff, J. M. Dolan, "Online Verification of Automated Road Vehicles Using Reachability Analysis", IEEE Transactions on Robotics, 2014.





Assume your car is in this zone and its dynamics is given by

$$\dot{X} = f(X)$$

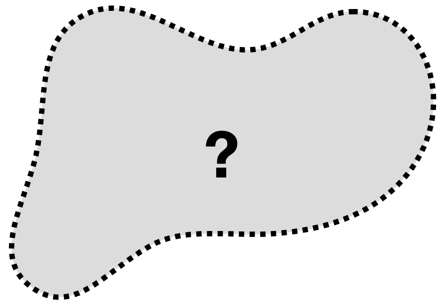
$\longrightarrow$

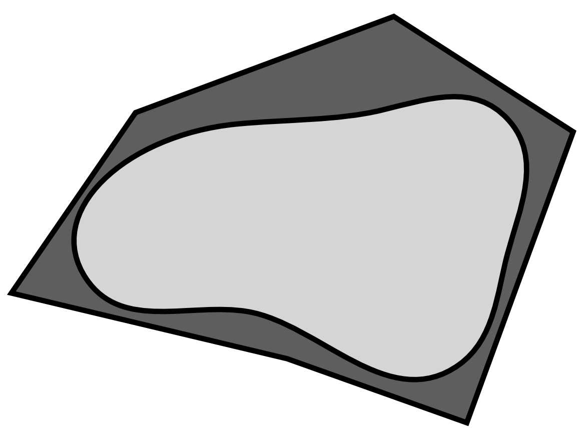
Linearize the system

$$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t + \textbf{Errors}$$

One state-of-the-art's method

M. Althoff, J. M. Dolan, “Online Verification of Automated Road Vehicles Using Reachability Analysis”, IEEE Transactions on Robotics, 2014.





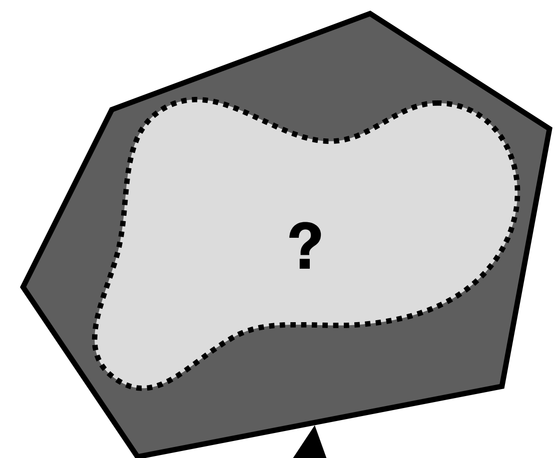
The Errors can be bounded using Lagrange remainders

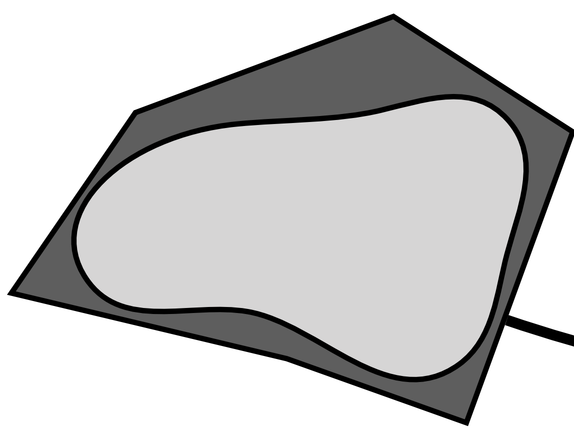
Assume your car is in this zone and its dynamics is given by
$$\dot{X} = f(X)$$

Linearize the system
$$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t + \textbf{Errors}$$

One state-of-the-art's method

M. Althoff, J. M. Dolan, "Online Verification of Automated Road Vehicles Using Reachability Analysis", IEEE Transactions on Robotics, 2014.





Compute an over-approximation of the reachable zone

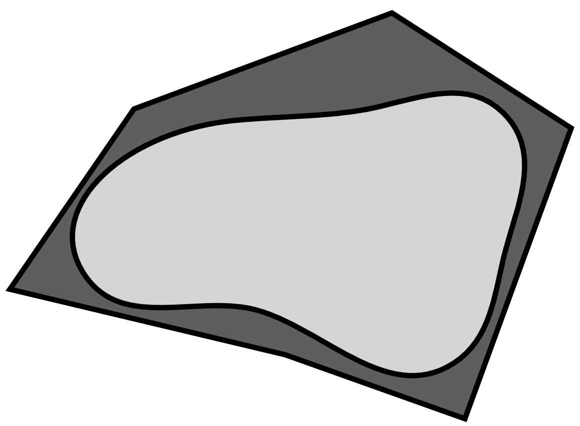
Assume your car is in this zone and its dynamics is given by
$$\dot{X} = f(X)$$

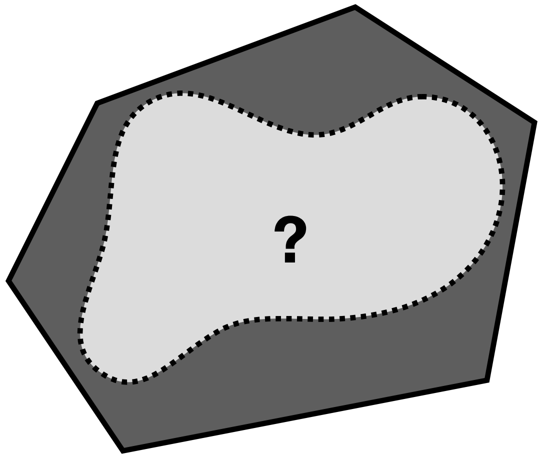
Linearize the system
$$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t + \textbf{Errors}$$

One state-of-the-art's method

M. Althoff, J. M. Dolan, "Online Verification of Automated Road Vehicles Using Reachability Analysis", IEEE Transactions on Robotics, 2014.

What about if the system is stochastic?





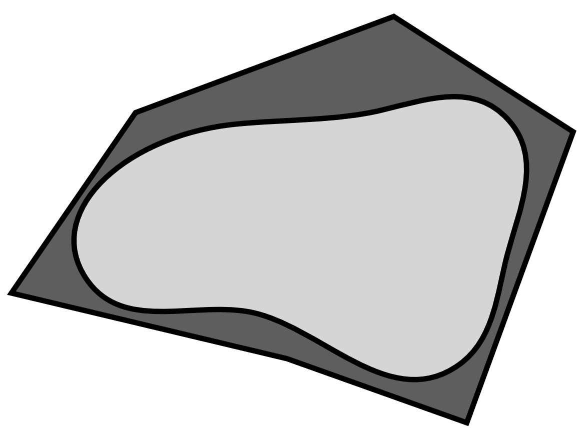
Assume your car is in this zone and its dynamics is given by
$\dot{X} = f(X)$ + **Random**

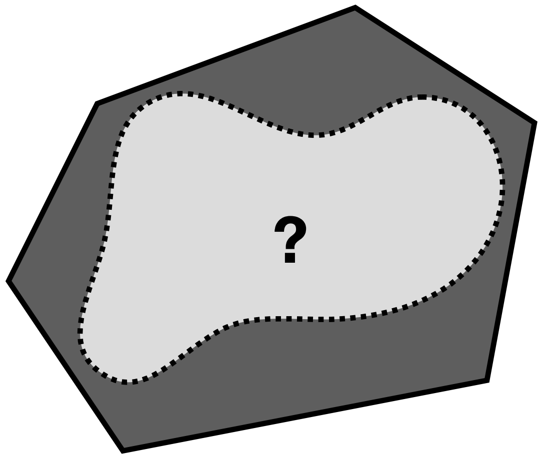
Linearize the system
$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t$ + **Errors**

One state-of-the-art's method

M. Althoff, J. M. Dolan, “Online Verification of Automated Road Vehicles Using Reachability Analysis”, IEEE Transactions on Robotics, 2014.

What about if the system is stochastic?





Assume your car is in this zone and its dynamics is given by

$\dot{X} = f(X)$ + **Random**

Linearize the system

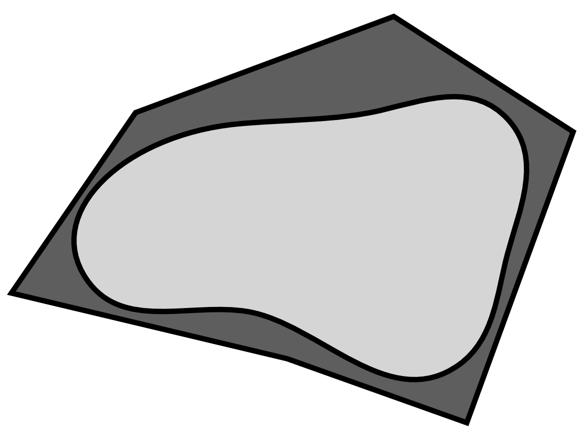
$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t$ + **BiggerErrors**

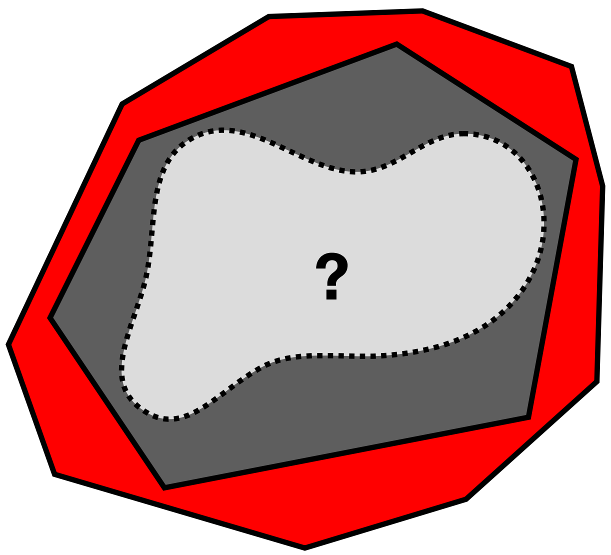
If bounded

One state-of-the-art's method

M. Althoff, J. M. Dolan, "Online Verification of Automated Road Vehicles Using Reachability Analysis", IEEE Transactions on Robotics, 2014.

What about if the system is stochastic?





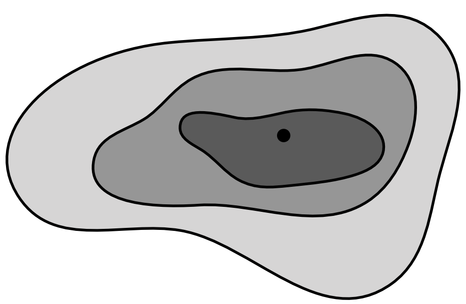
Assume your car is in this zone and its dynamics is given by
$\dot{X} = f(X) +$ **Random**

→

Linearize the system
$X(\Delta t) = X_0 + f(X_0) \cdot \Delta t +$ **BiggerErrors**

If bounded

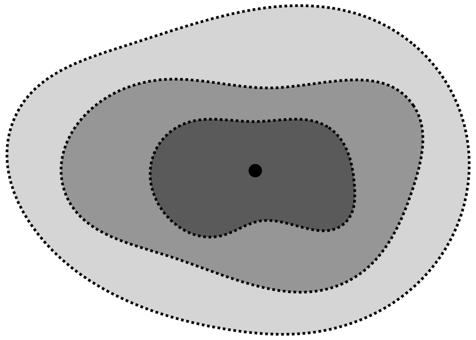
Reachability analysis, for stochastic systems



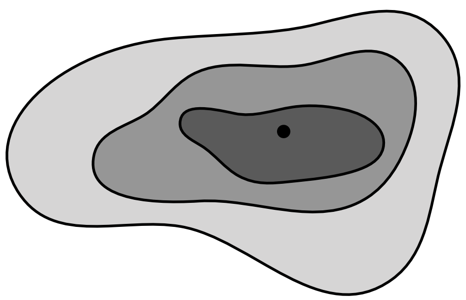
Assume the initial position follows
this distribution (given by moments)
and the dynamics is given by

$$\dot{X} = f(X)$$

Reachability analysis, for stochastic systems



Estimate the distribution after Δt time
(estimate the moments)



Assume the initial position follows
this distribution (given by moments)
and the dynamics is given by

$$\dot{X} = f(X)$$

Discrete-time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$\begin{gathered} x(0) = x_0 \\ x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t) \end{gathered}$$

Discrete-time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$\begin{gathered} x(0) = x_0 \\ x(t+1) = \boxed{F_0(t)} + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t) \end{gathered}$$

vector in $\mathbb{R}^n$

Discrete-time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$

$$x(t+1) = F_0(t) + \boxed{\boxed{F_1(t)}\ \boxed{x(t)}} + \ldots + F_d(t) \cdot x^{[d]}(t)$$

$n \times n$ matrix

vector in $\mathbb{R}^n$

vector in $\mathbb{R}^n$

Discrete-time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= F_0(t) + F_1(t) \cdot x(t) + \ldots + \boxed{\boxed{F_d(t)} \cdot \boxed{x^{[d]}(t)}} \end{aligned}$$

$n \times n^d$ matrix

vector in $\mathbb{R}^n$

vector in $\mathbb{R}^{n^d}$

d-th Kronecker power of $x(t)$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{[2]} = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

Discrete-time polynomial system

States are given by vectors:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Usual assumption: the $F_i(t)$s do not depend on t nor on x_0

Example, a bicycle model

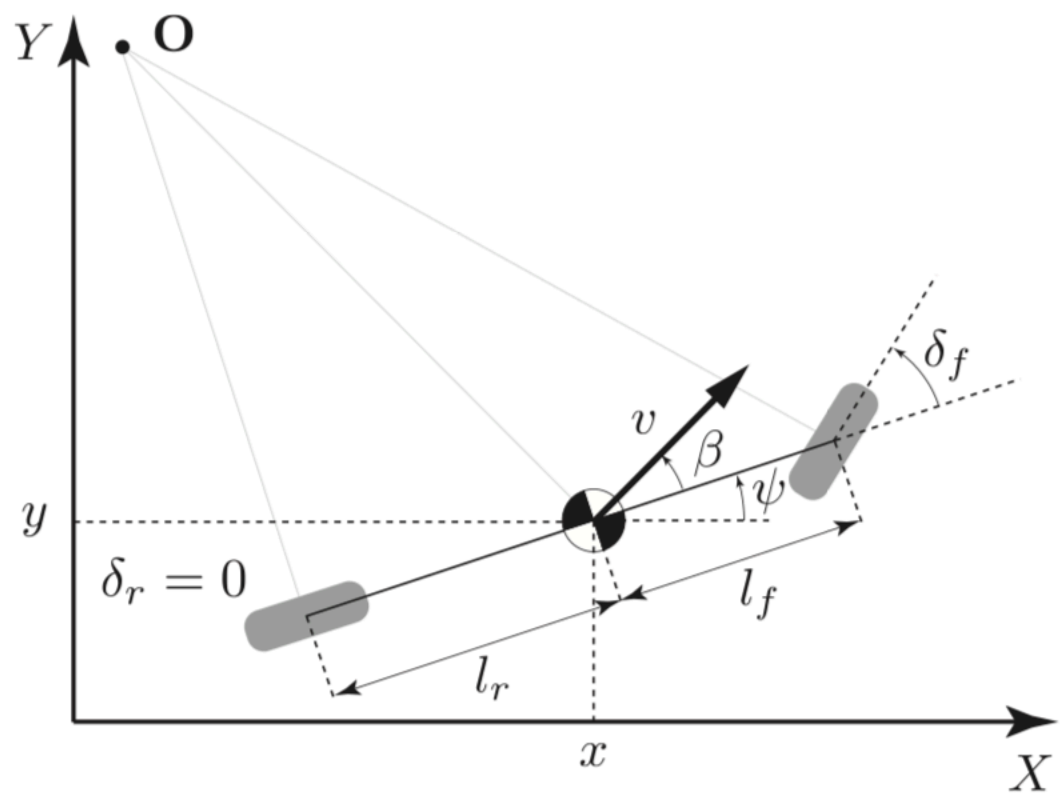


Fig. 1: Kinematic Bicycle Model

$$\dot{x}(t) = v(t) \cdot \cos(\psi(t) + \beta)$$

$$\dot{y}(t) = v(t) \cdot \sin(\psi(t) + \beta)$$

$$\dot{\psi}(t) = \frac{v(t)}{\ell} \cdot \sin \beta$$

$$\dot{v}(t) = a(t)$$

Kong et al., “Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design”, IEEE-IV 2015.

Example, a bicycle model

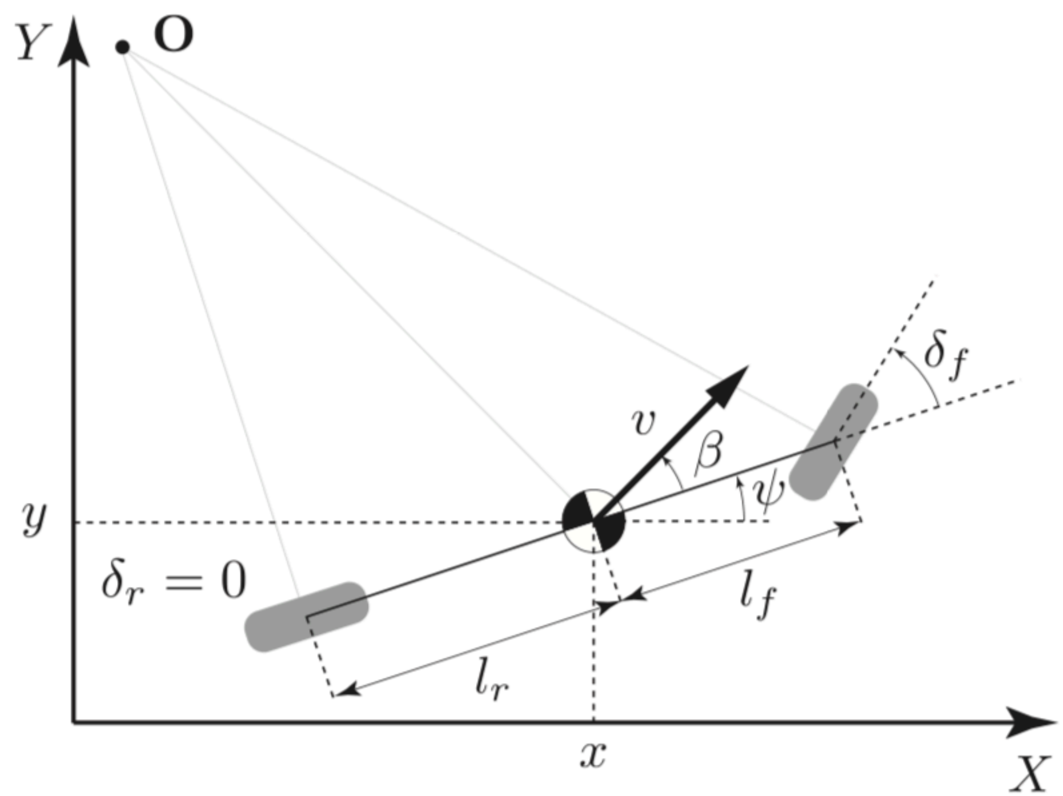


Fig. 1: Kinematic Bicycle Model

$$\dot{x}(t) = v(t) \cdot c(t)$$

$$\dot{y}(t) = v(t) \cdot s(t)$$

$$\dot{\psi}(t) = \frac{v(t)}{\ell} \cdot \sin \beta$$

$$\dot{v}(t) = a(t)$$

$$\dot{c}(t) = -\frac{s(t) \cdot v(t) \cdot \sin \beta}{\ell}$$

$$\dot{s}(t) = \frac{c(t) \cdot v(t) \cdot \sin \beta}{\ell}$$

Example, a bicycle model

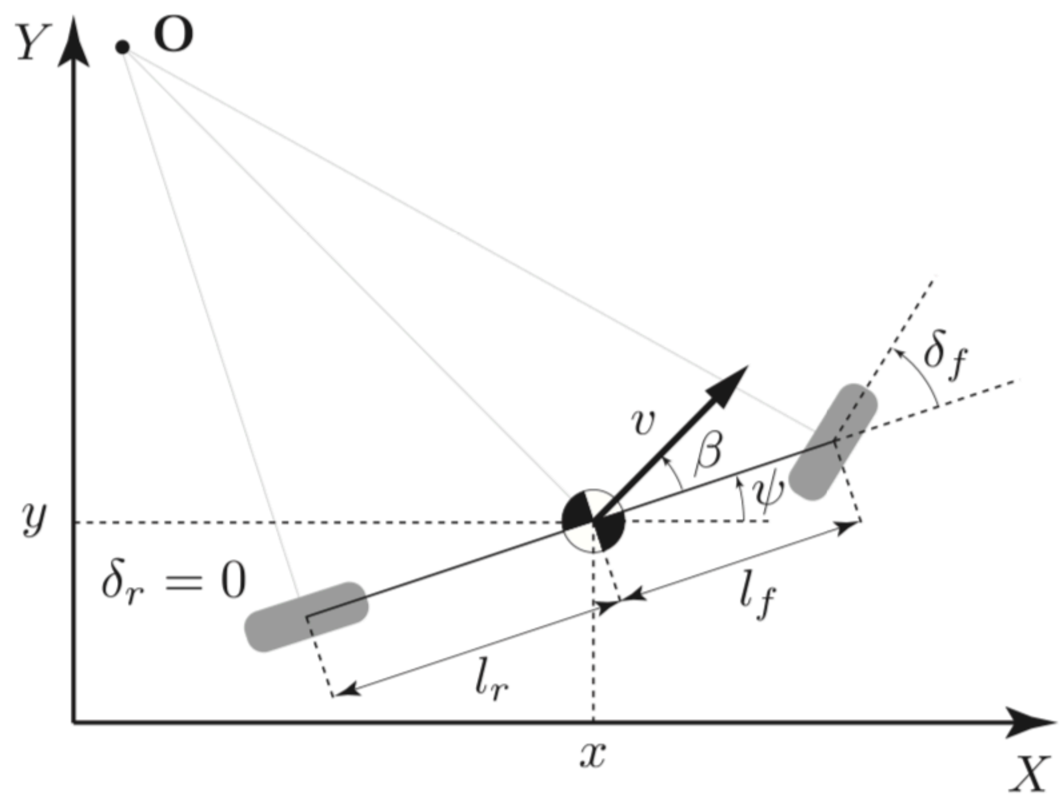


Fig. 1: Kinematic Bicycle Model

$$x(t+\Delta) = x(t) + \Delta c(t)v(t) + \frac{\Delta^2}{2}\left(a(t)c(t) - \frac{s(t)v^2(t)\sin\beta}{\ell}\right)$$
$$y(t+\Delta) = y(t) + \Delta s(t)v(t) + \frac{\Delta^2}{2}\left(a(t)s(t) + \frac{c(t)v^2(t)\sin\beta}{\ell}\right)$$
$$\psi(t+\Delta) = \psi(t) + \Delta\frac{v(t)}{\ell}\sin\beta + \frac{\Delta^2}{2}\frac{a(t)}{\ell}\sin\beta$$
$$v(t+\Delta) = v(t) + \Delta a(t)$$
$$c(t+\Delta) = c(t) - \Delta\frac{s(t)v(t)\sin\beta}{\ell} - \frac{\Delta^2}{2}\left(\frac{c(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)s(t)\sin\beta}{\ell}\right)$$
$$s(t+\Delta) = s(t) + \Delta\frac{c(t)v(t)\sin\beta}{\ell} + \frac{\Delta^2}{2}\left(-\frac{s(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)c(t)\sin\beta}{\ell}\right)$$

Example, a bicycle model

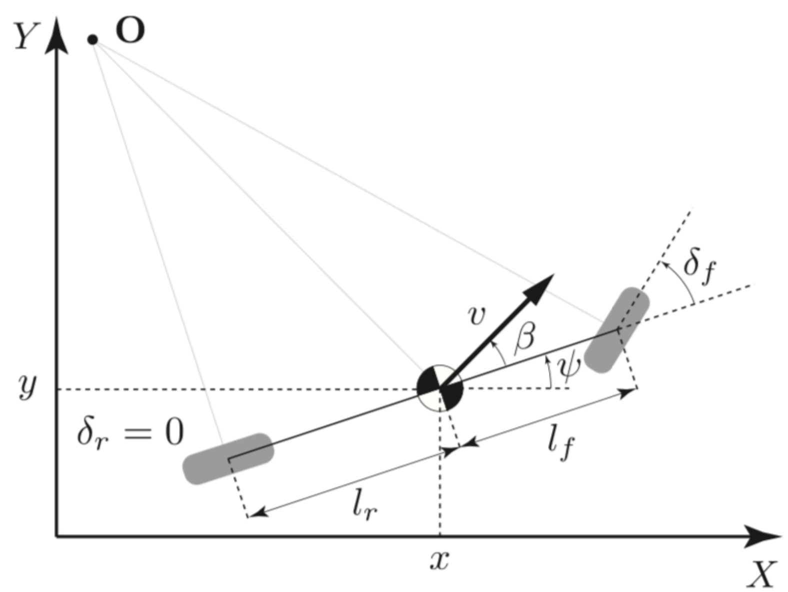


Fig. 1: Kinematic Bicycle Model

$$x(t+\Delta) = x(t) + \Delta c(t)v(t) + \frac{\Delta^2}{2}\left(a(t)c(t) - \frac{s(t)v^2(t)\sin\beta}{\ell}\right)$$
$$y(t+\Delta) = y(t) + \Delta s(t)v(t) + \frac{\Delta^2}{2}\left(a(t)s(t) + \frac{c(t)v^2(t)\sin\beta}{\ell}\right)$$
$$\psi(t+\Delta) = \psi(t) + \Delta\frac{v(t)}{\ell}\sin\beta + \frac{\Delta^2}{2}\frac{a(t)}{\ell}\sin\beta$$
$$v(t+\Delta) = v(t) + \Delta a(t)$$
$$c(t+\Delta) = c(t) - \Delta\frac{s(t)v(t)\sin\beta}{\ell} - \frac{\Delta^2}{2}\left(\frac{c(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)s(t)\sin\beta}{\ell}\right)$$
$$s(t+\Delta) = s(t) + \Delta\frac{c(t)v(t)\sin\beta}{\ell} + \frac{\Delta^2}{2}\left(-\frac{s(t)v^2(t)\sin^2\beta}{\ell^2} + \frac{a(t)c(t)\sin\beta}{\ell}\right)$$

$$F_0(t) = \begin{bmatrix} 0 \\ 0 \\ \frac{\Delta^2 a(t)\sin\beta}{2} \\ \Delta a(t) \\ 0 \\ 0 \end{bmatrix} \qquad F_1(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & \frac{\Delta^2 a(t)}{2} & 0 \\ 0 & 1 & 0 & 0 & 0 & \frac{\Delta^2 a(t)}{2} \\ 0 & 0 & 1 & \frac{\Delta\sin\beta}{\ell} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -\frac{\Delta^2 a(t)\sin\beta}{2\ell} \\ 0 & 0 & 0 & 0 & -\frac{\Delta^2 a(t)\sin\beta}{2\ell} & 1 \end{bmatrix}$$

Discrete-time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t) \end{aligned}$$

Discrete-time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = \boxed{F_0(t)} + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

vector of random variables

Discrete-time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$

$$x(t+1) = F_0(t) + \boxed{\boxed{F_1(t)}\ \boxed{x(t)}} + \ldots + F_d(t) \cdot x^{[d]}(t)$$

$n \times n$ random matrix

random vector

random vector

Discrete-time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$\begin{aligned} x(0) &= x_0 \\ x(t+1) &= F_0(t) + F_1(t) \cdot x(t) + \ldots + \boxed{\boxed{F_d(t)} \cdot \boxed{x^{[d]}(t)}} \end{aligned}$$

$n \times n^d$ random matrix

random vector

random vector

Discrete-time polynomial stochastic system

States are given by vectors of random variables:

$$x(0), x(1), \ldots, x(t), x(t+1), \ldots \in \Omega \to \mathbb{R}^n$$

satisfying an equation of the form:

$$x(0) = x_0$$
$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Usual assumption: the $F_i(t)$s do not depend on t

($F_i(t)$ and $F_j(s)$ are independent for $t \neq s$)

($F_i(t)$ and $F_i(s)$ are identically distributed)

nor on x_0

(x_0 and $F_i(t)$ are independent)

Carleman linearization

Main idea: transform a finite-dimensional polynomial system into a infinite-dimensional linear system

$$x(t+1) = F_0(t) + F_1(t) \cdot x(t) + \ldots + F_d(t) \cdot x^{[d]}(t)$$

Computing the Kronecker products:

$$x^{[j]}(t+1) = \sum_{k=0}^{jd} A_{j,k}(t) \cdot x^{[k]}(t)$$

where $A_{j,k}(t)$ is computed from the $F_i(t)$. Using:

$$y(t) = [1 \quad x(t) \quad x^{[2]}(t) \quad \ldots \quad]$$

we obtain the following infinite-dimensional linear system:

$$y(t+1) = A(t) \cdot y(t)$$

where $A(t)$ is computed from the $F_i(t)$.

Moment equations

The j-th moments of $x(t)$ are given by $\mathbb{E}(x^{[j]}(t))$.

Equations between the moments:

$$\mathbb{E}(x^{[j]}(t+1)) = \sum_{k=0}^{jd} \mathbb{E}(A_{j,k}(t) \cdot x^{[k]}(t))$$

using the assumptions:

$$\mathbb{E}(x^{[j]}(t+1)) = \sum_{k=0}^{jd} \mathbb{E}(A_{j,k}(t)) \cdot \mathbb{E}(x^{[k]}(t))$$

and $\mathbb{E}(A_{j,k}(t))$ is independent of t. We then obtain:

$$\boxed{\mathbb{E}(y(t+1)) = E \cdot \mathbb{E}(y(t))}$$

where E is computed from the moments of the $F_i(t)$.

Truncated system

M. Forest, A. Pouly, "Explicit error bounds for Carleman linearization", arXiv:1711.02552, 2017.

Fix N, and define E_N the restriction of E to the $\sum_{k=0}^{N} n^k$ raws and columns.

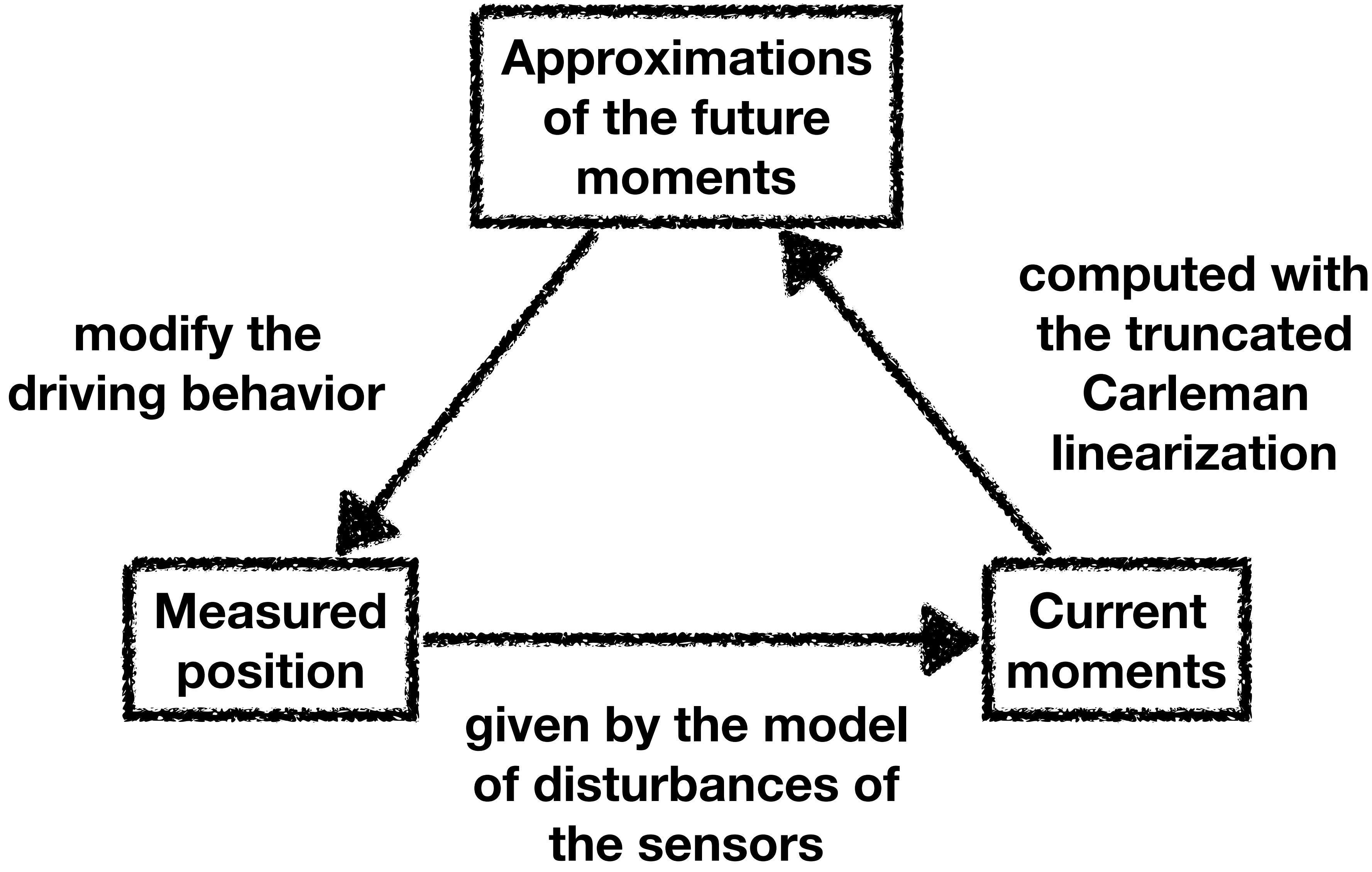
Our estimation of the N-th first moments of $x(t)$ is given by the following system:

$$\widetilde{y}(0) = [1 \quad \mathbb{E}(x_0) \quad \dots \quad \mathbb{E}(x_0^{[N]})]$$
$$\widetilde{y}(t+1) = E_N \cdot \widetilde{y}(t)$$

That is, $\widetilde{y}(t)$ is an approximation of $[1 \quad \mathbb{E}(x(t)) \quad \dots \quad \mathbb{E}(x^{[N]}(t))]$.

Furthermore, we have efficient ways of computing bounds of the errors.

Online computation for autonomous cars



Conclusion: online cost very small, no need to compute $E(N, N)$!

Tail probability analysis

What can we do with the estimated future moments and the bounds on errors?

Proving a Chebyshev-like inequality:

<u>Proposition:</u>

We can compute a bound of

$$\mathbb{P}(\|x(t) - \widetilde{x}_1(t)\| \geq \alpha)$$

where $\widetilde{x}_1(t)$ is our estimation of the first moment of $x(t)$ using:

- our estimation of the second moment of $x(t)$,
- the bounds on the errors of those estimations.

Furthermore, all those values can be computed efficiently.

Numerical results, for the bicycle model

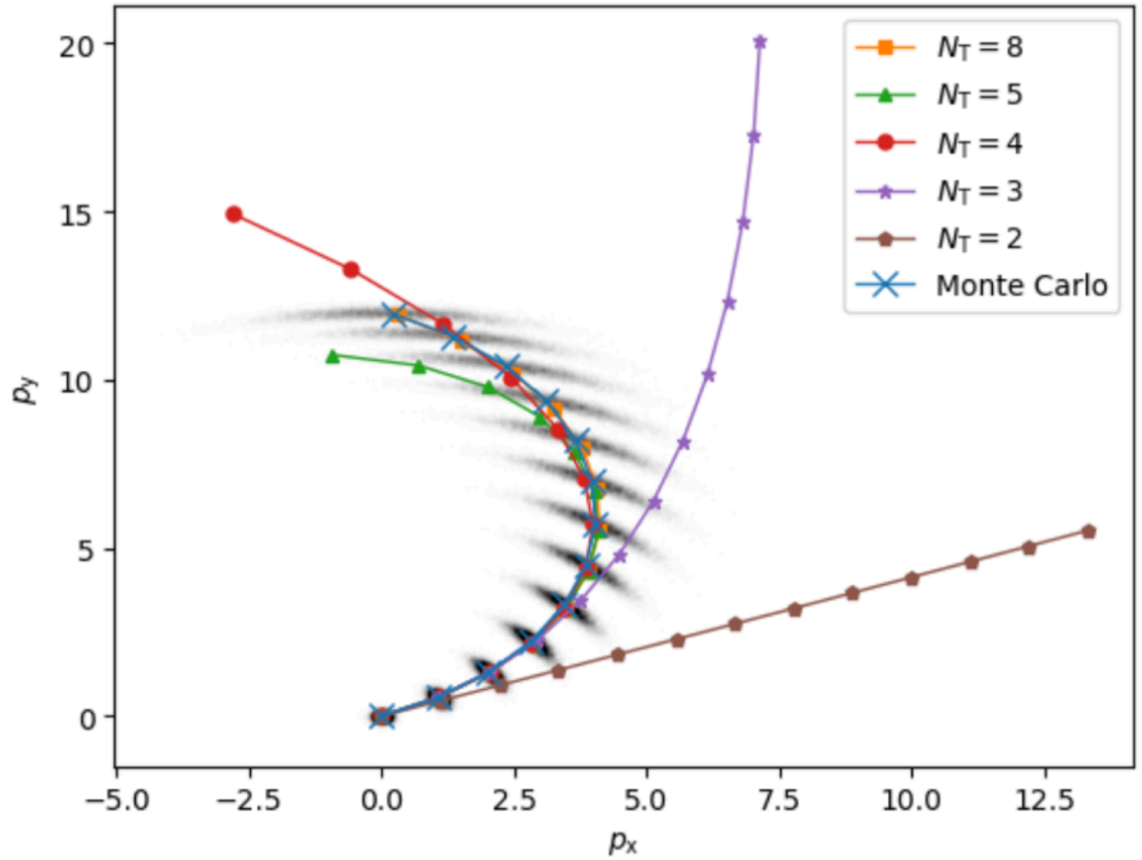


Fig. 6. First moment approximation in vehicle dynamics.

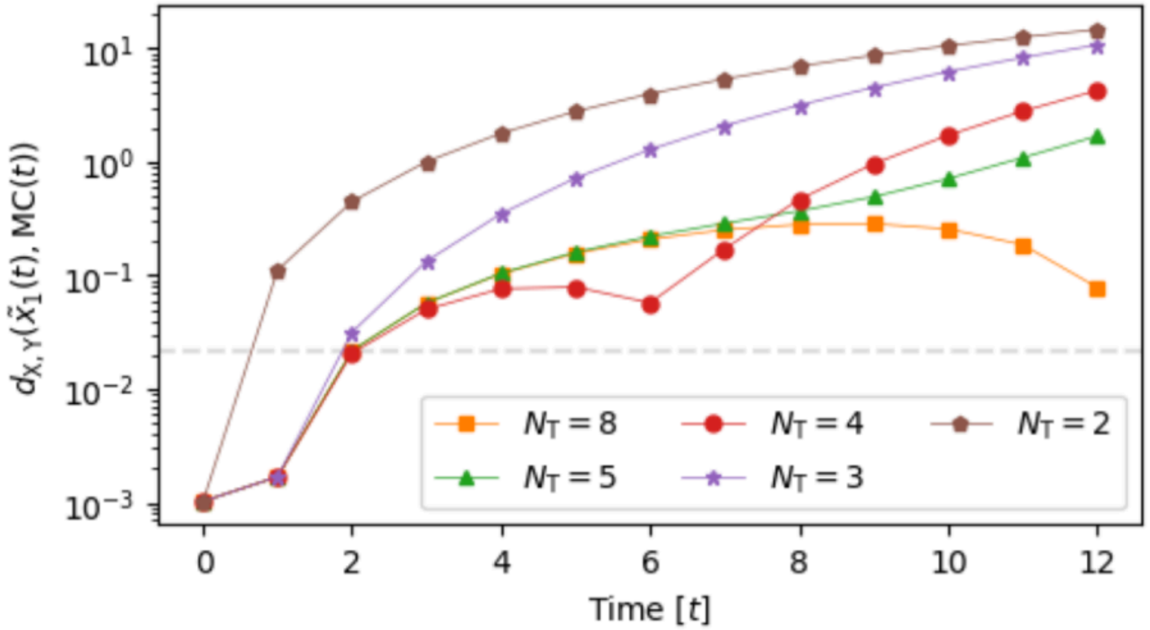


Fig. 7. Distance to the mean of the empirical distribution.

Online computation vs. Monte Carlo

Method	Monte Carlo		Moment propagation			
Parameters	num. samples		N_{T}			
	10	10^4	4	16	64	256
Time (μs)	2.9e10^3	3.4e10^6	11	14	30	93

Future work

- Develop the tail probability analysis
- Reconstruction of the distribution from the moments
- Improving the computation by using the symmetries in the Kronecker powers