FORMULA ZERO

Distributionally Robust Online Adaptation via Offline Population Synthesis

Aman Sinha*, Matthew O'Kelly*, Hongrui Zheng*, Rahul Mangharam, John Duchi, Russ Tedrake

Stanford Artificial Intelligence · Penn Engineering · CSAIL

Image: James Gilleard

Overview

Population Synthesis

Online Adaptation

Experiments

Balancing Performance and Safety

Current AV technology still struggles in non-cooperative scenarios like merging due to competing objectives:

- **Maximize performance:** negotiate the merge without delay or hesitation
- **Maintain safety:** avoid catastrophic failures and crashes

Racing (autonomously) highlights this performance safety tradeoff.

Videos: Mobileye and Formula 1

Autonomous Racing

In autonomous racing, the ego-agent must lap a racetrack in the presence of other agents deploying **unknown** policies.

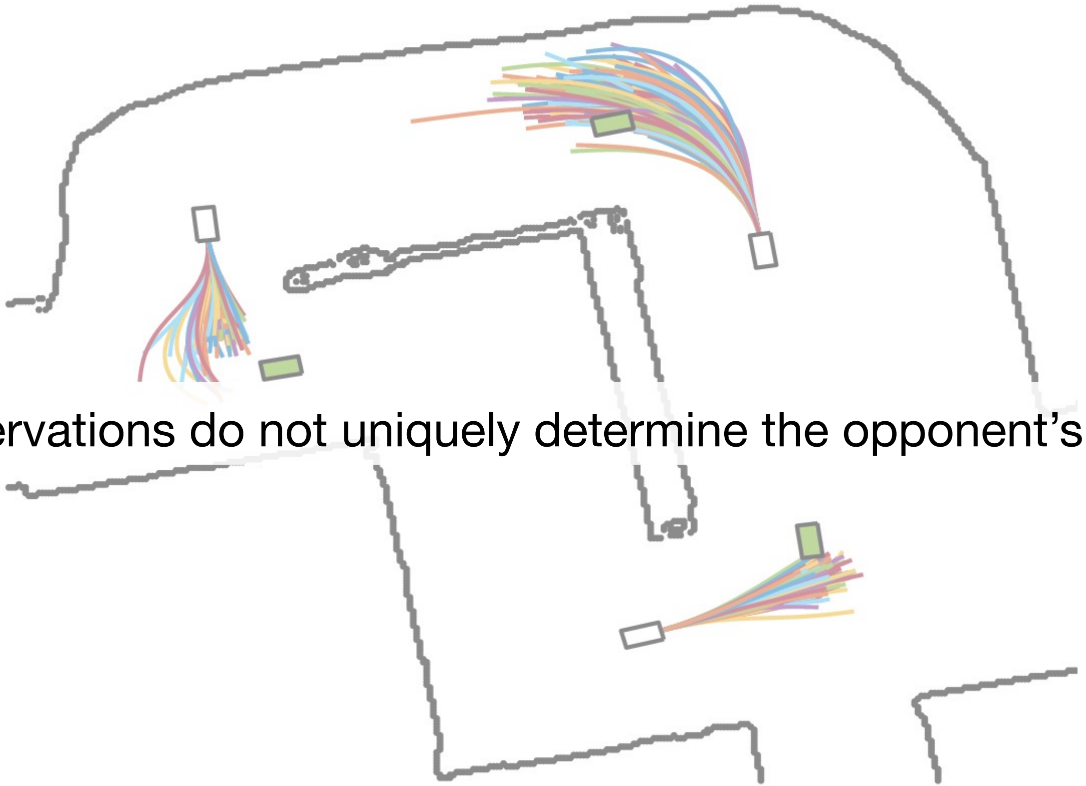
The agent wins by:

- Completing the race first
- Crashing automatically results in a loss

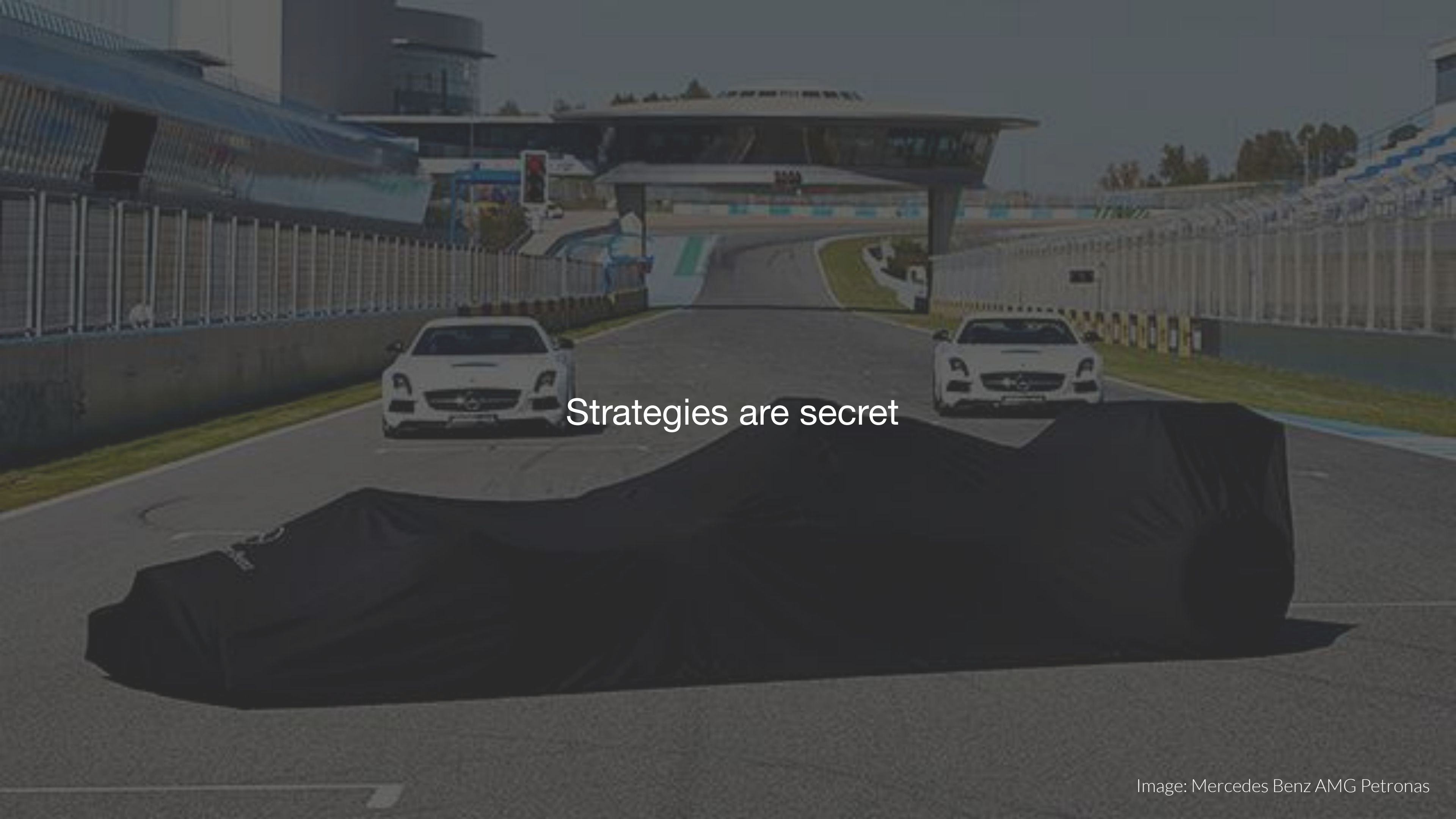
Our simulation and hardware platform is open-source: https://f1tenth.org

Crashing is expensive and dangerous

Image: Formula 1



Sensor observations do not uniquely determine the opponent’s behavior



Strategies are secret

Image: Mercedes Benz AMG Petronas

Robust Reinforcement Learning

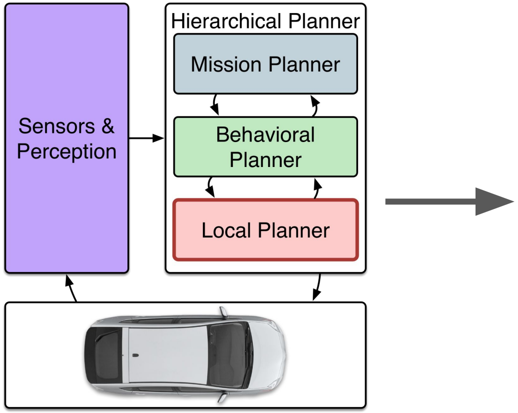
$$\text{maximize} \sum_t \lambda^t \mathbb{E}[r(o(t))]$$

If we knew the opponent's behavior we wouldn't need an ambiguity set

$$\text{maximize} \inf_{P_{sa} \in \mathcal{P}} \sum_t \lambda^t \mathbb{E}[r(o(t))]$$

- We capture uncertainty in the behaviors of other agents through an ambiguity set, $\mathcal{P}$
- A larger ambiguity set, $\mathcal{P}$, ensures a greater degree of safety while sacrificing performance against a particular opponent
- Two challenges: learning P_{sa} offline (without expert demonstrations) and adjusting $\mathcal{P}$ online.

Population Synthesis

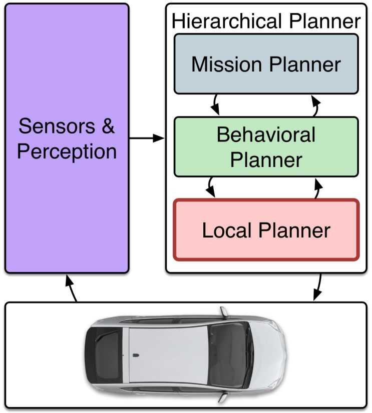


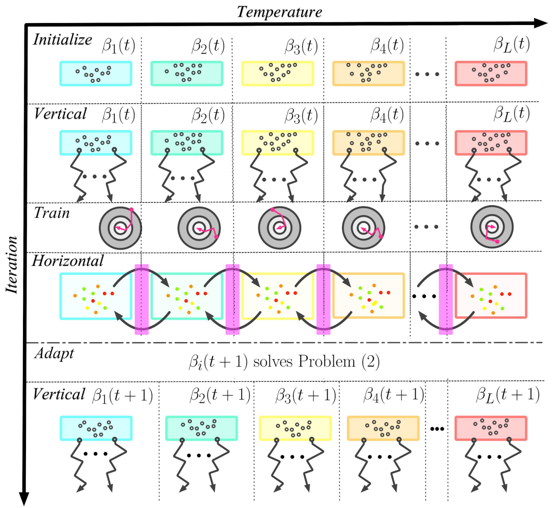
Parameterized Policy:

1. Goal Generator: Inverse Autoregressive Flow weights
2. Goal Evaluator: non-differentiable cost function weights

(Kingma et al 17)

Population Synthesis





Parameterized Policy:

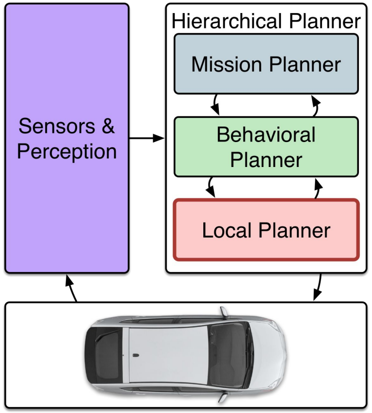
1. Goal Generator: Inverse Autoregressive Flow weights
2. Goal Evaluator: non-differentiable cost function weights

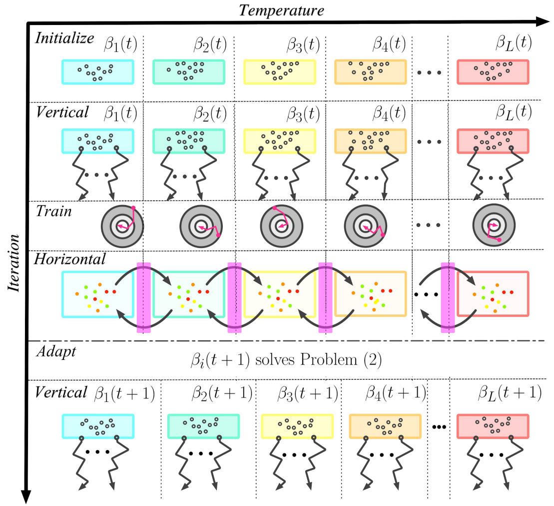
Population Synthesis:

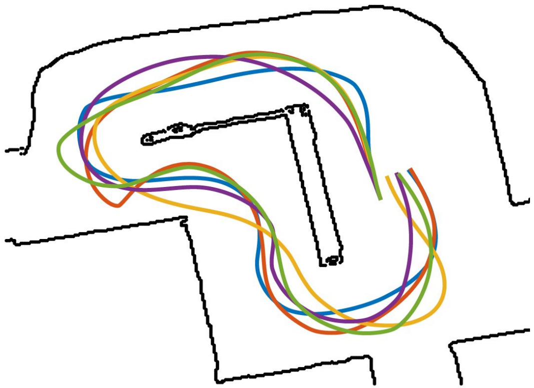
1. Highly-scalable population-based MCMC solution
2. Uses self-play to generate competitive agents

(Marinari & Parisi 92)

Population Synthesis







Parameterized Policy:

1. Goal Generator: Inverse Autoregressive Flow weights
2. Goal Evaluator: non-differentiable cost function weights

Population Synthesis:

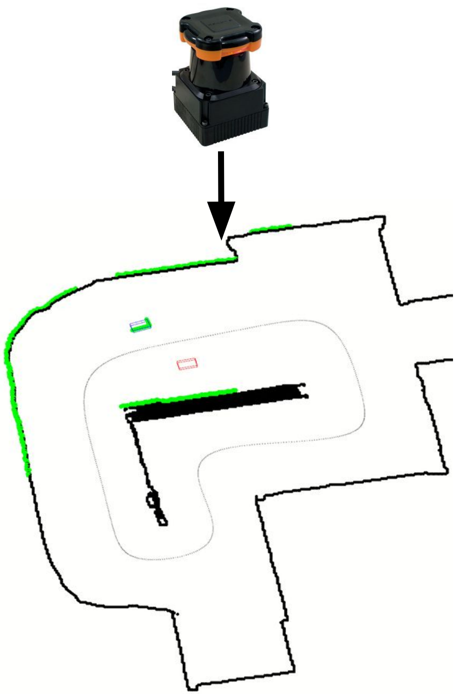
1. Highly-scalable population-based MCMC solution
2. Uses self-play to generate competitive agents

Opponent Prototypes:

1. Elite members of population are described by their policy parameters
2. A diverse subset is selected for online use

(Kulesza et al 12)

Online Adaptation

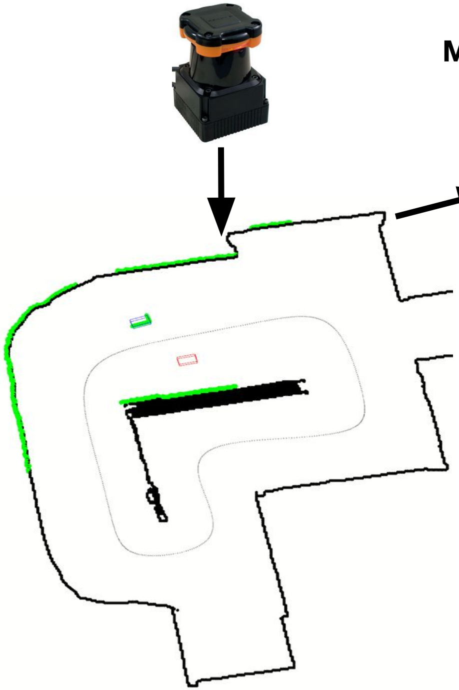


Sensor Measurements

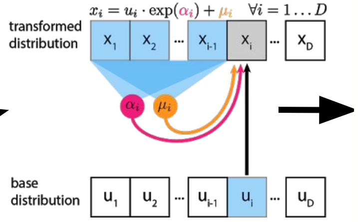
Online Adaptation

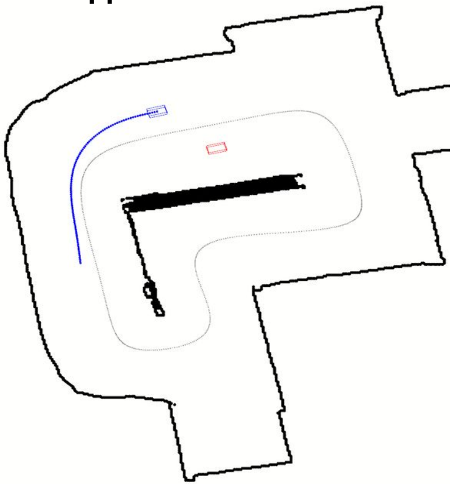
Opponent Prediction

Masked Autoregressive Flow



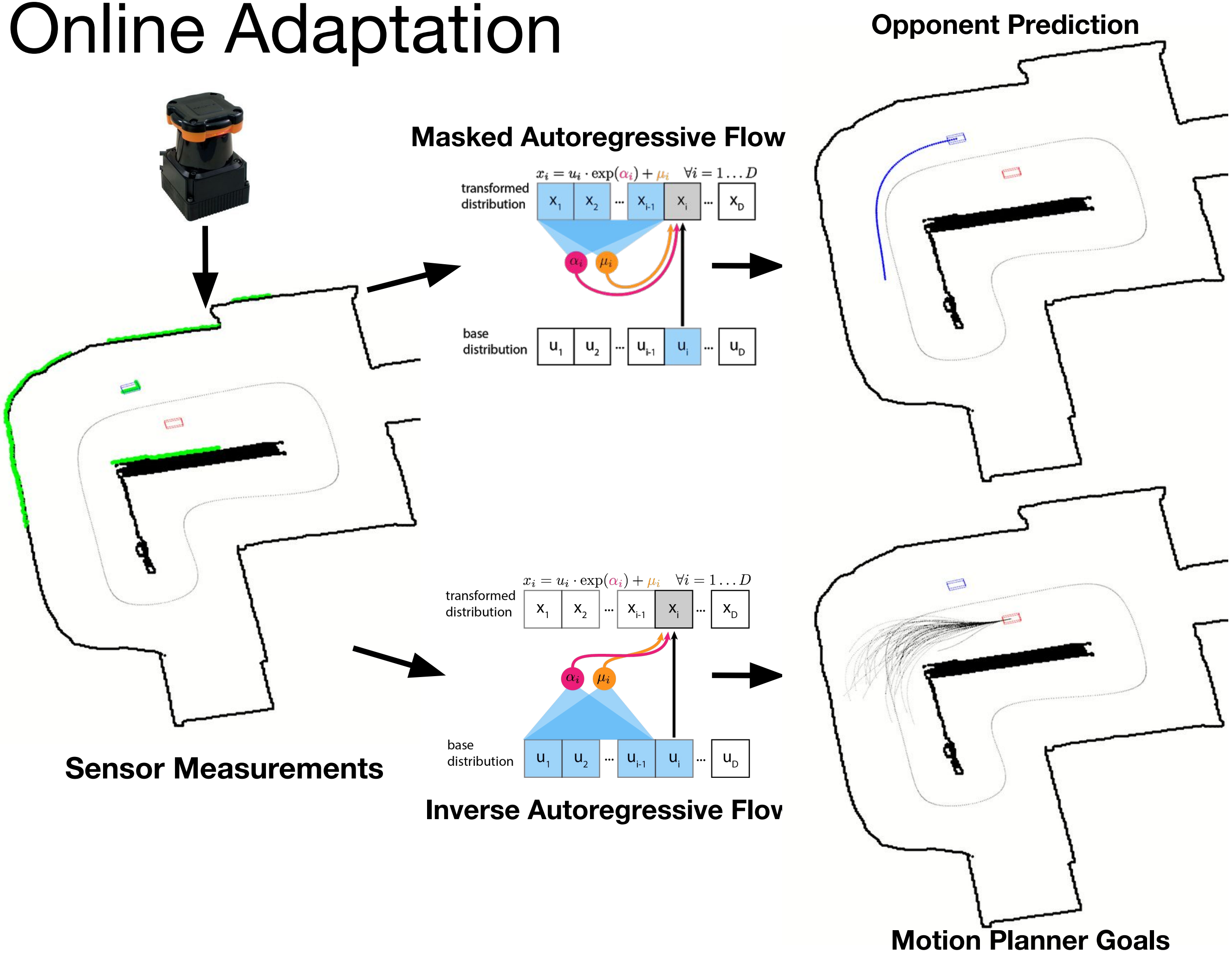
Sensor Measurements





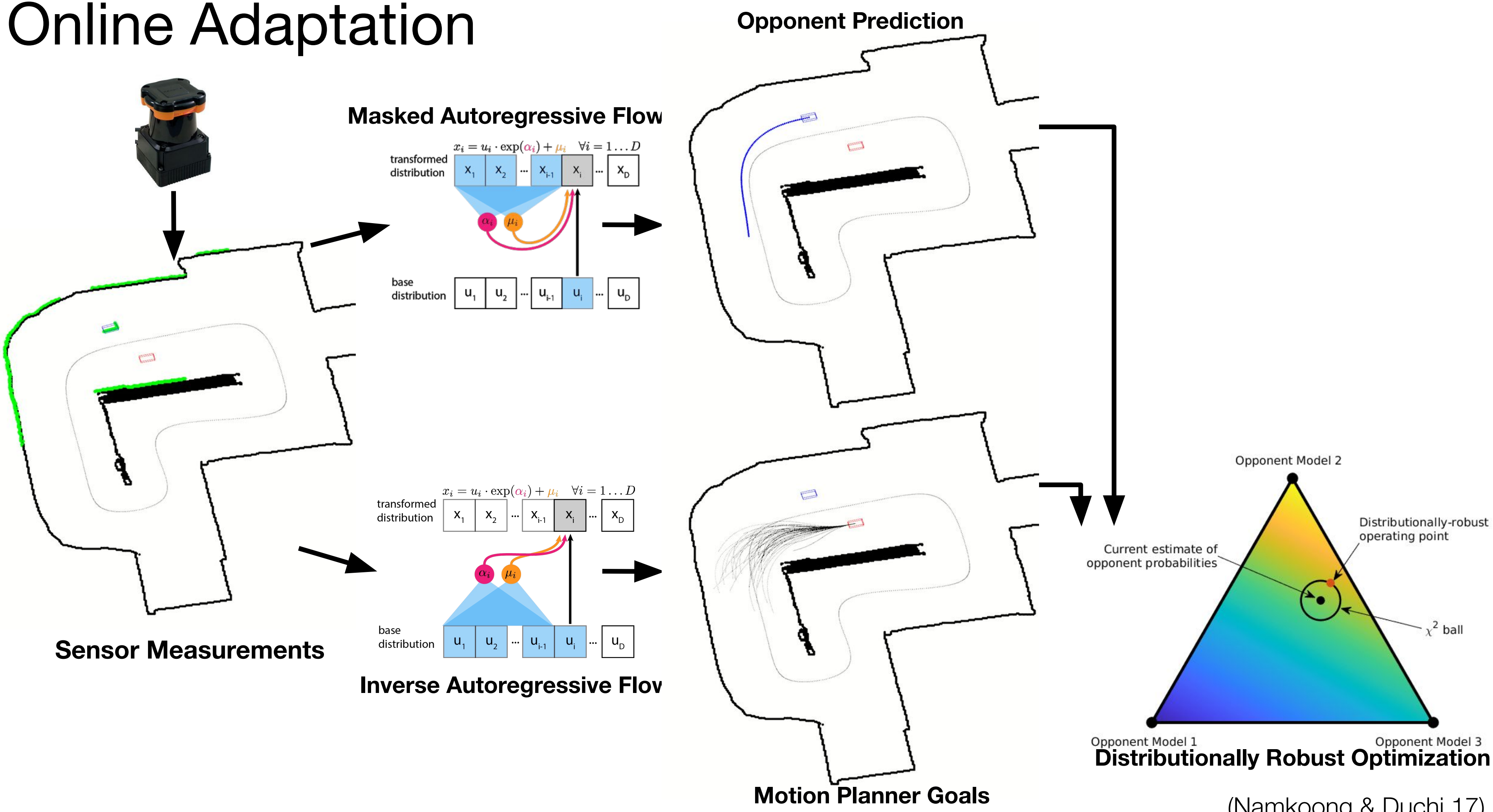
(Lattimore & Szepesvari 20, Papamakarios et al 17)

Online Adaptation

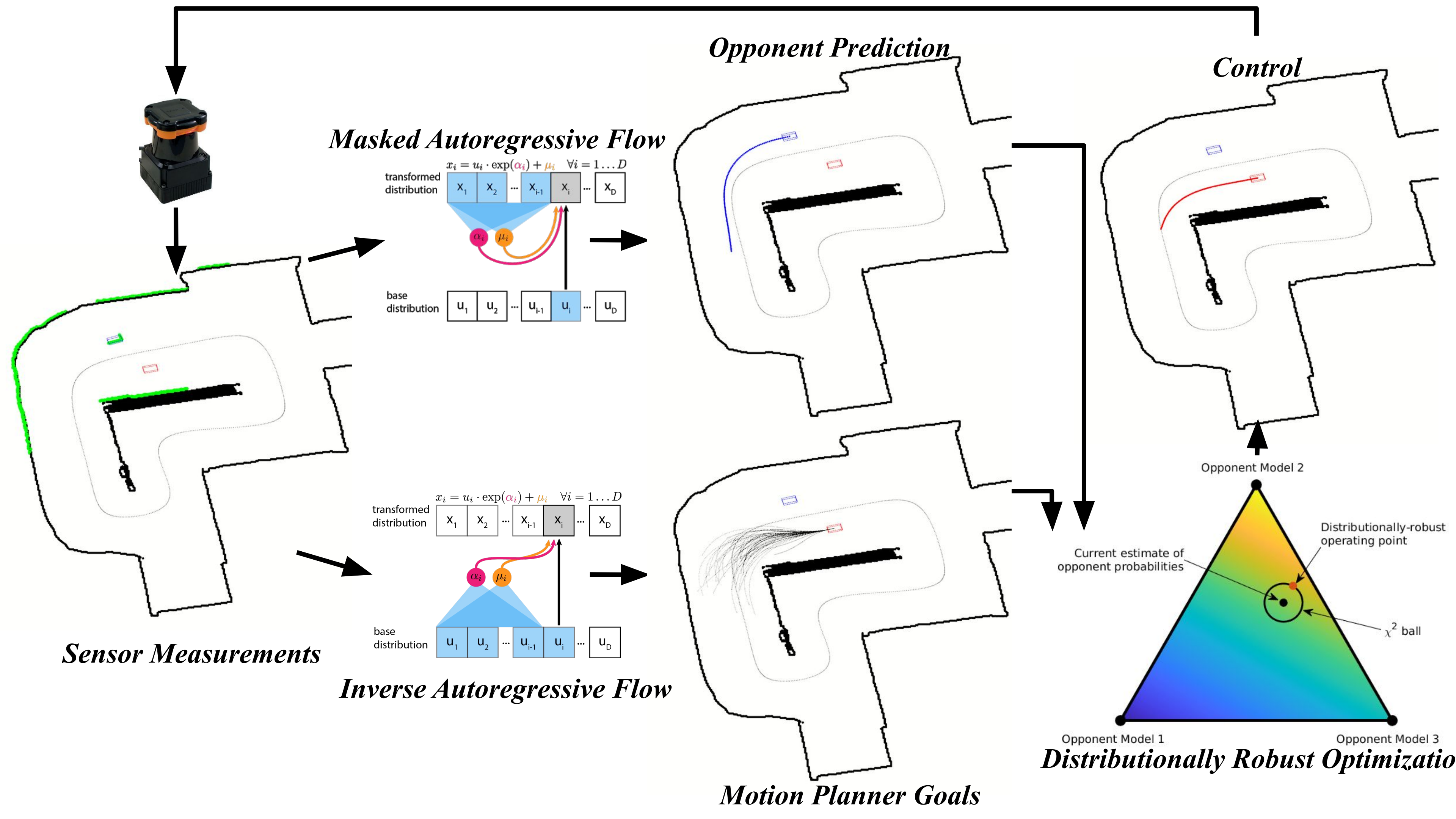


(Kingma et al 17)

Online Adaptation



(Namkoong & Duchi 17)


Opponent Prediction
Control
Masked Autoregressive Flow
$x_i = u_i \cdot \exp(\alpha_i) + \mu_i \quad \forall i = 1 \ldots D$
transformed distribution
base distribution
Inverse Autoregressive Flow
Sensor Measurements
Motion Planner Goals
Opponent Model 1
Opponent Model 2
Opponent Model 3
Distributionally-robust operating point
Current estimate of opponent probabilities
χ^2 ball
Distributionally Robust Optimizatio

Related Work

- Robust RL/control
  - Robust MDP (Nilim, El Ghaoui 05)
  - POMDP (Kaelbling et al 98)
  - Adversarial RL (Pinto et al 17, Mandlekar et al 17)

- Belief-space planning (Kochenderfer 15, Galceran et al 15, Van Den Berg et al 11)
- DRO (Ben-Tal et al 13, Namkoong & Duchi 17)
- Bandits (Lattimore & Szepesvari 20)
- Quality-diversity algorithms (Mouret & Clune 15)
- Simulated tempering (Marinari & Parisi 92)

Overview

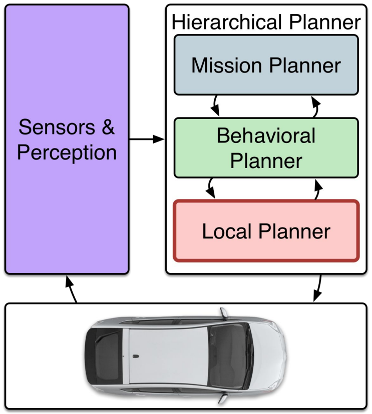
Population Synthesis

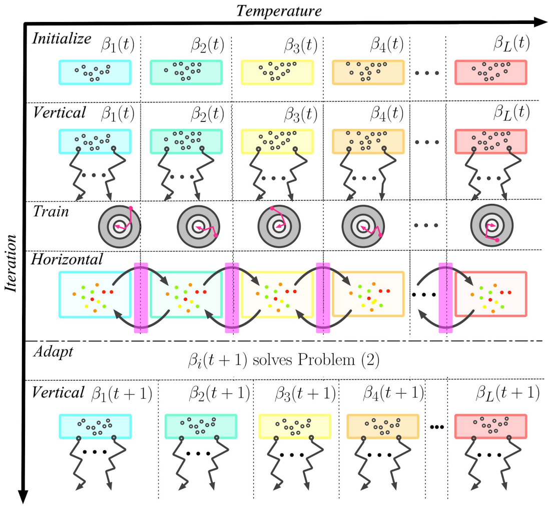
Online Adaptation

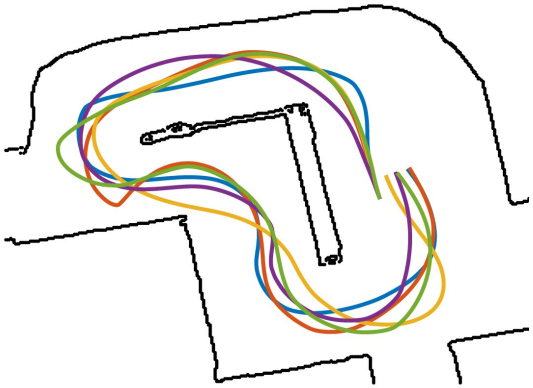
Experiments

Population Synthesis

The goal of offline population synthesis is to generate a diverse set of competitive agent behaviors.



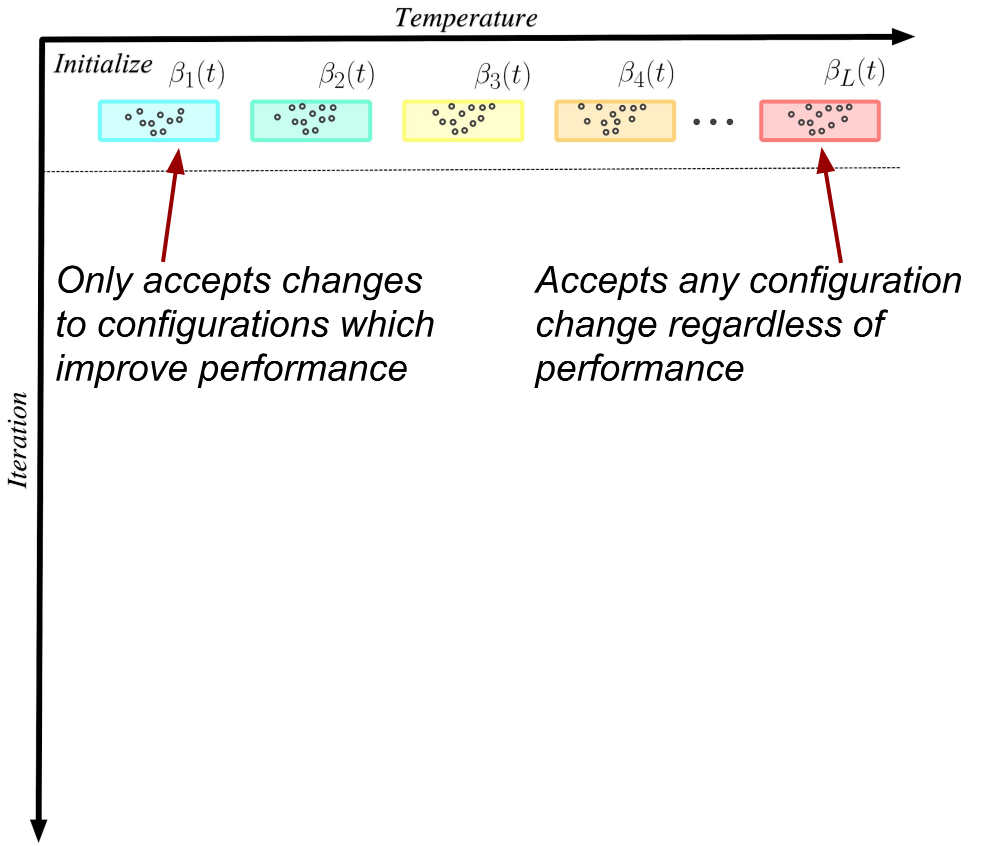




In our AV application, θ parametrizes a neural network used to sample trajectories to follow, x is a weighting of various cost functions that the vehicle uses to select trajectories from the samples, and $f(x, \theta)$ is the simulated lap time.

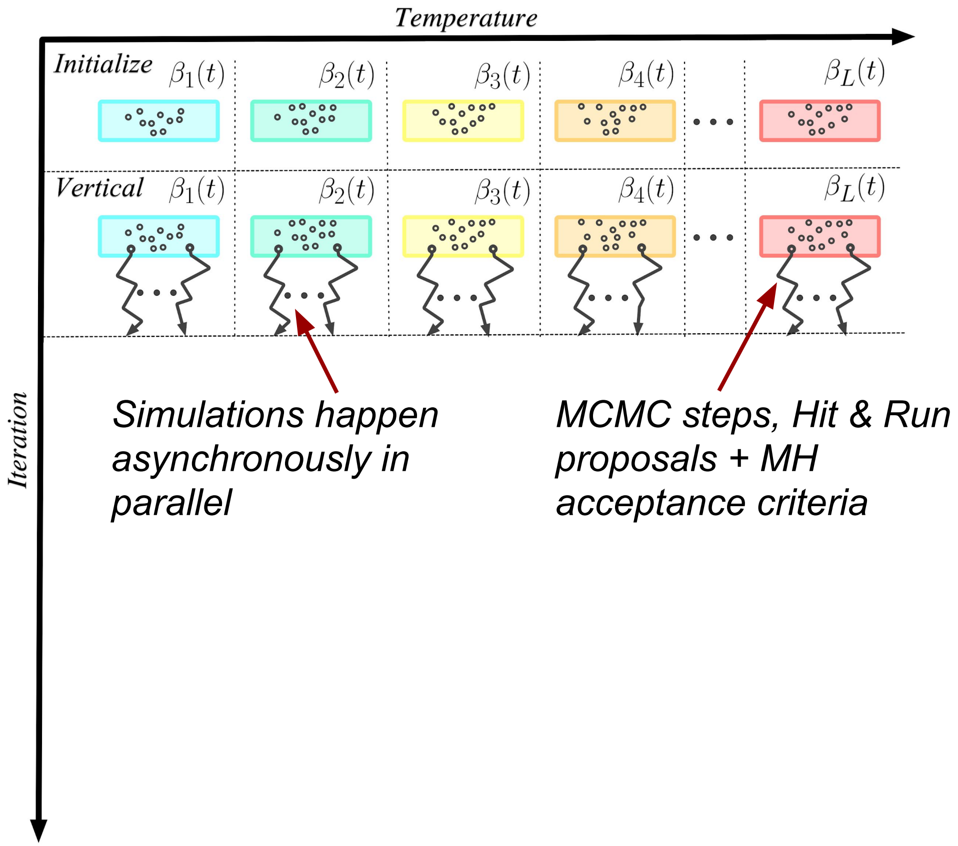
Step 1: Initialize Populations

- Builds off of a concept known in MCMC literature as parallel tempering (Marinari & Parisi 92)
- Initialize several “baths” of configurations that are composed of both differentiable and non-differentiable parameters
- Unlike parallel tempering we maintain populations at each level



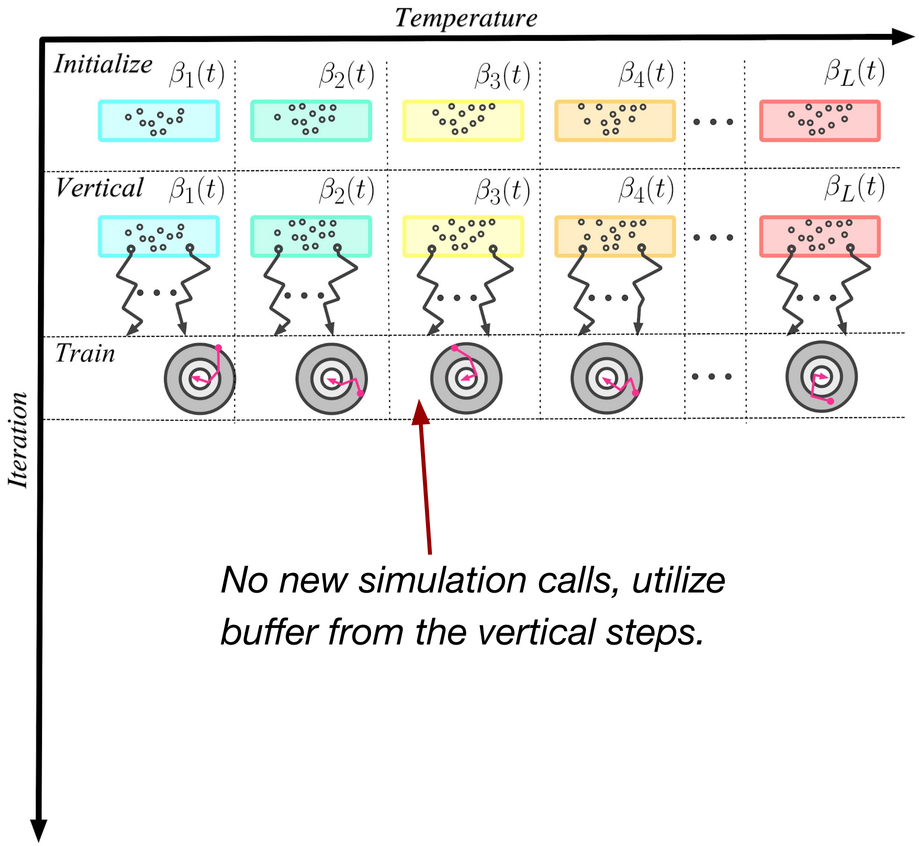
Step 2: Vertical MCMC Exploration

- In the vertical phase of the algorithm we explore the space of non-differentiable parameters using MCMC.
- Each proposal is evaluated by a race simulation between the perturbed configuration and the previous configuration.
- Proposals are accepted according to the standard MH criteria.



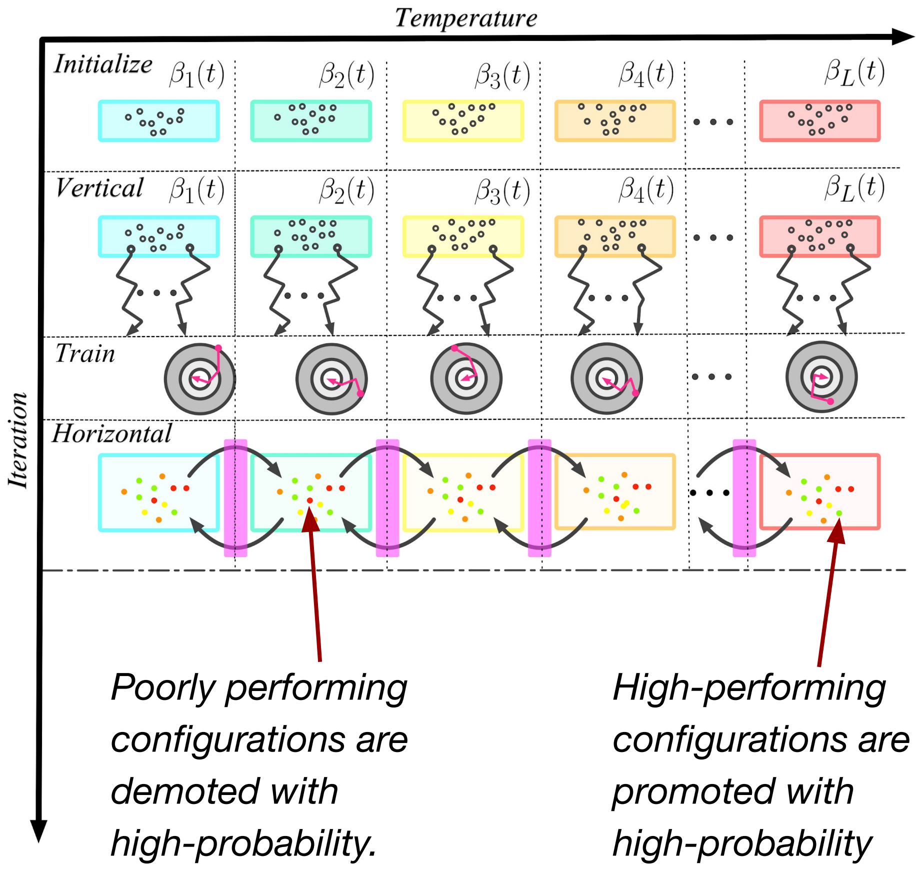
Step 3: SGD Parameter Update

- Run SGD updates on differentiable parameters (e.g. MAF/IAF network parameters).
- The objective is to maximizes the likelihood of the trajectories chosen by the agent with cost functions parametrized by x.



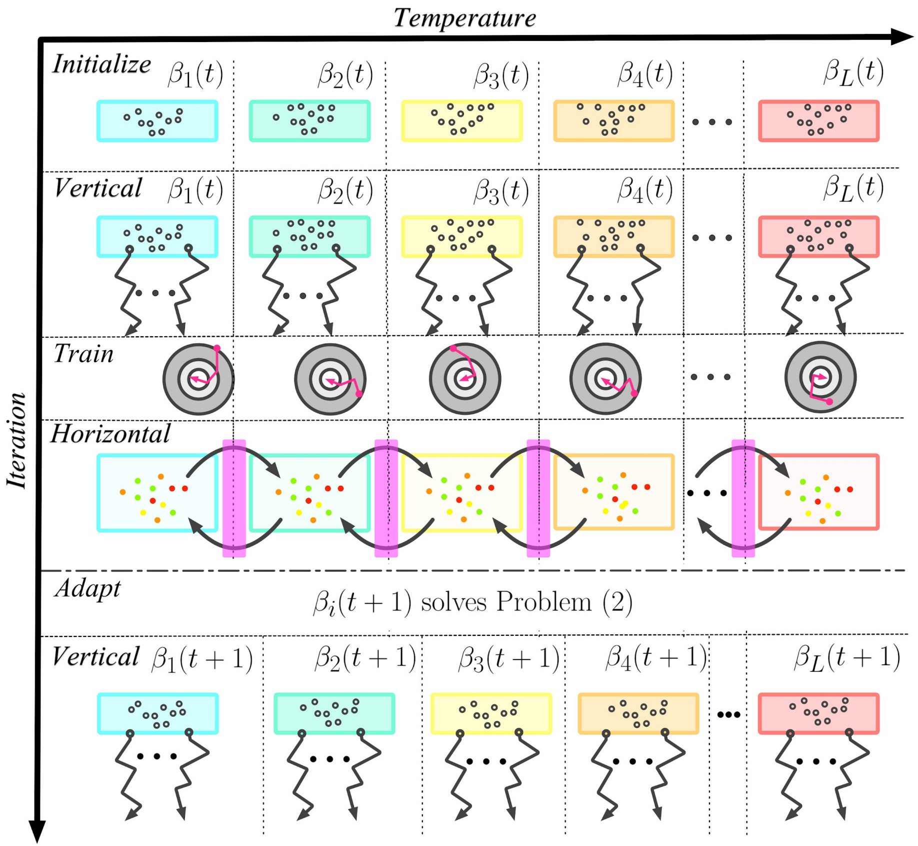
Step 4: Horizontal MCMC Tempering

- Horizontal proposals consist of swapping two configurations in adjacent temperature levels uniformly at random
- The proposal is accepted using standard Metropolis-Hastings (MH) criteria
- This procedure is especially efficient because it doesn't require new simulations.

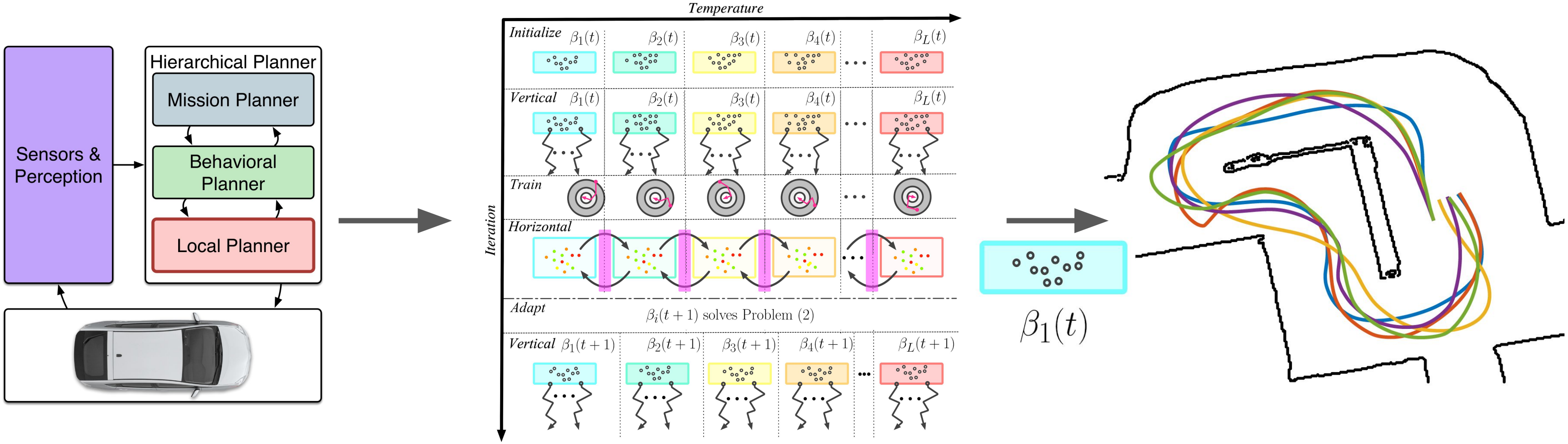


Step 5: Temperature Updates

- Anneal horizontal swap acceptance probability in order to automatically adjust temperature levels.
- This adaptive scheme is crucial in our problem setting, where we a priori have no knowledge of appropriate scales for *f* and, as a result, *β.*



End Result: Population of Opponent Prototypes



When racing against a particular opponent, the agent maintains a belief vector w(t) of the opponent's behavior patterns as a categorical distribution over these prototype behaviors. We then parametrize the ambiguity set as a ball around this nominal belief w(t).

Overview

Population Synthesis

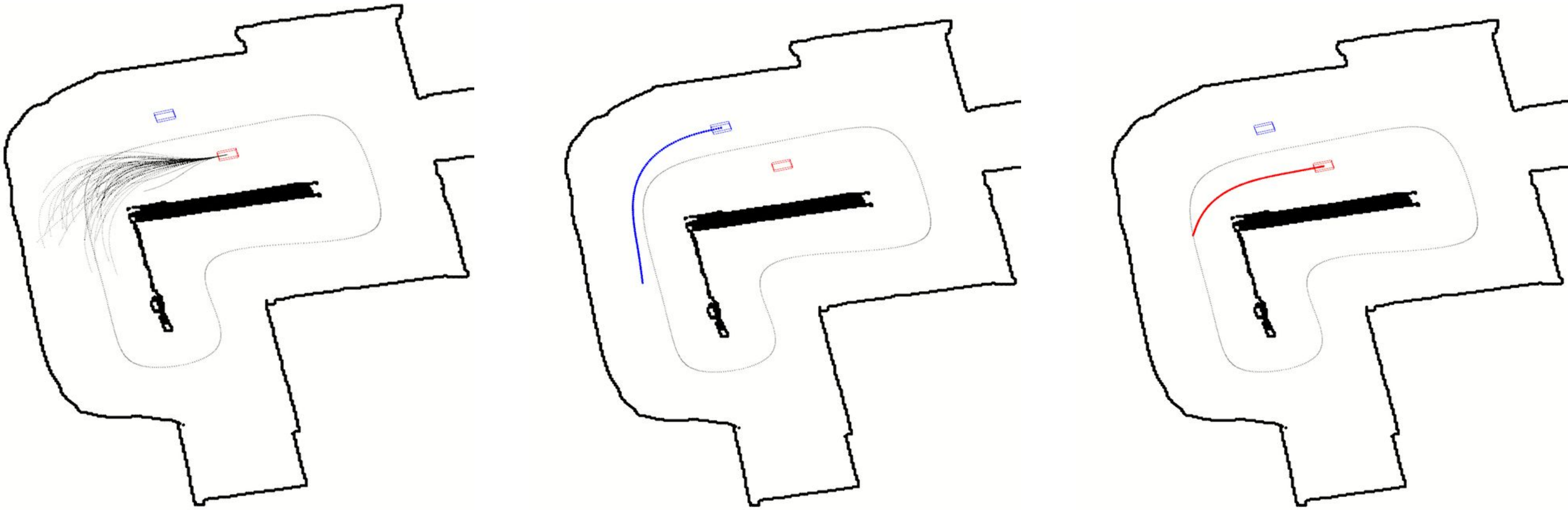
Online Adaptation

Experiments

Distributionally Robust Trajectory Cost

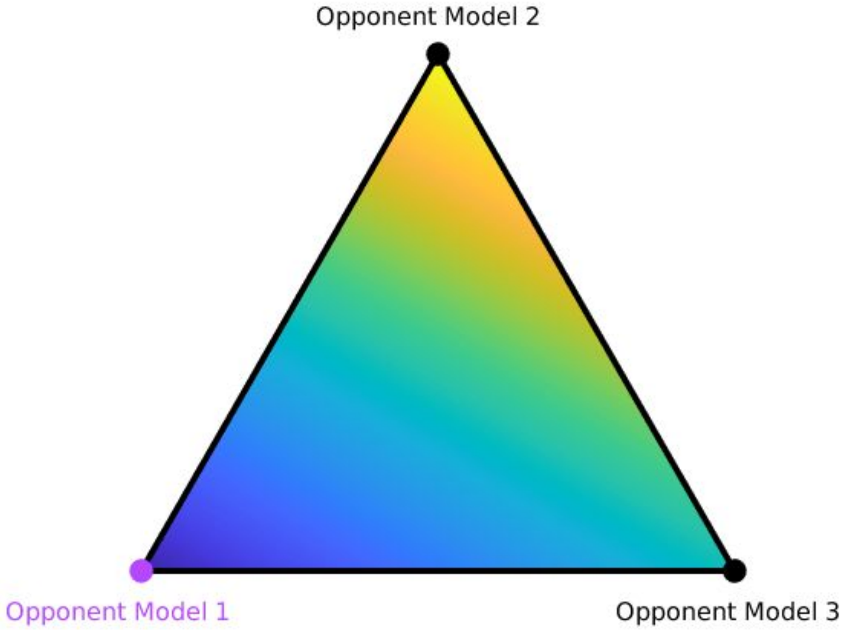
We will investigate how the ego-agent will choose its actions taking into account the opponent behaviors.

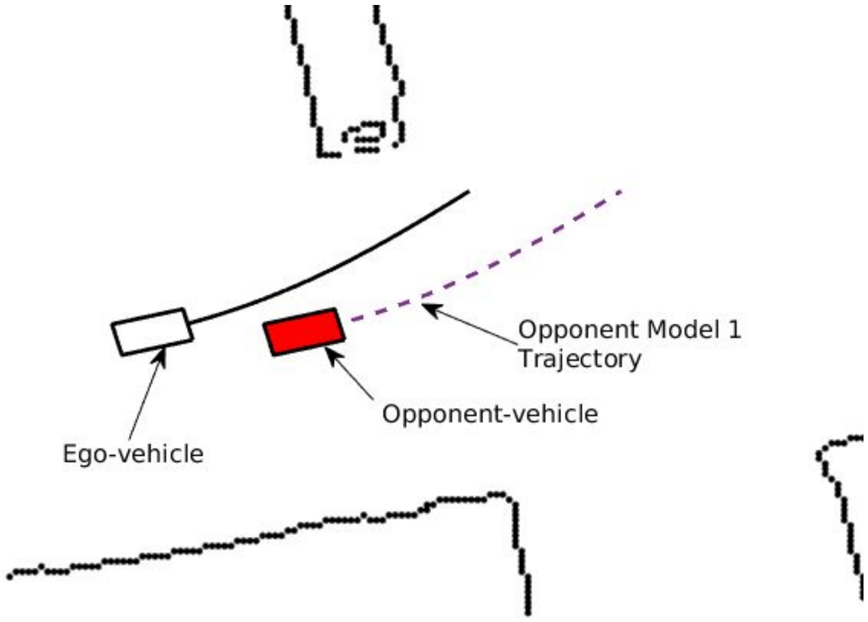
Motion Planner Goals **Opponent Predictions** **Plan**



Distributionally Robust Trajectory Cost

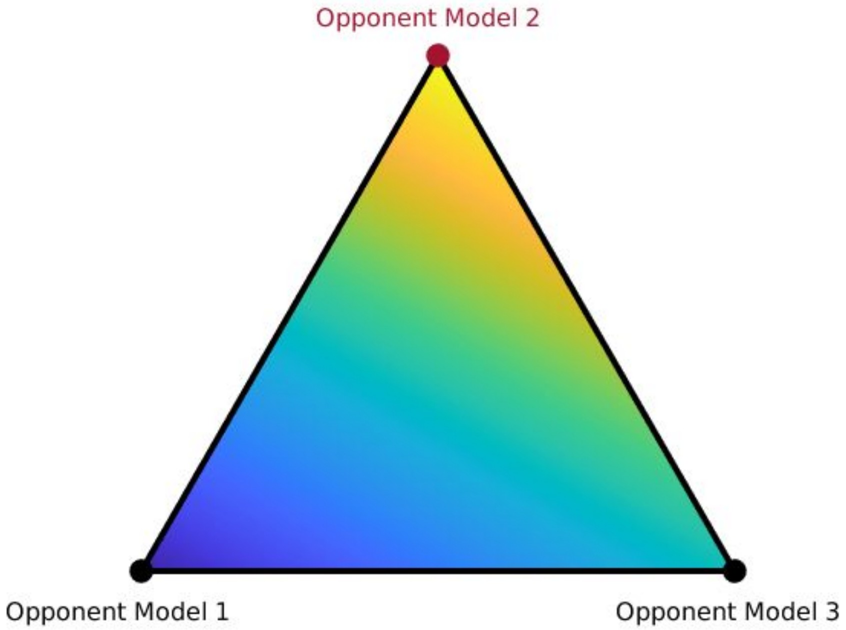
$$c_1(t;p) := -\sum_{s>t} \lambda^{s-t} \mathbb{E}[r(o(s);p)]$$

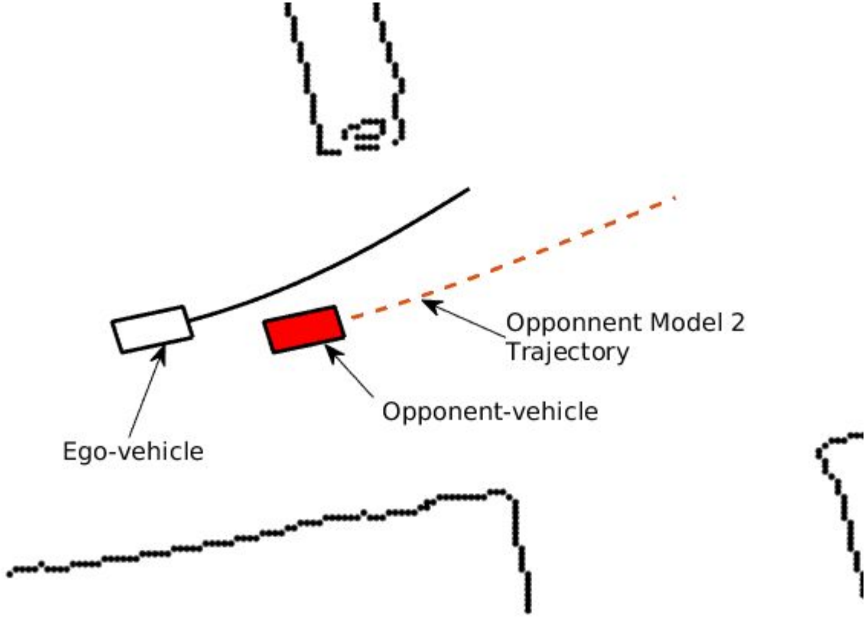




Distributionally Robust Trajectory Cost

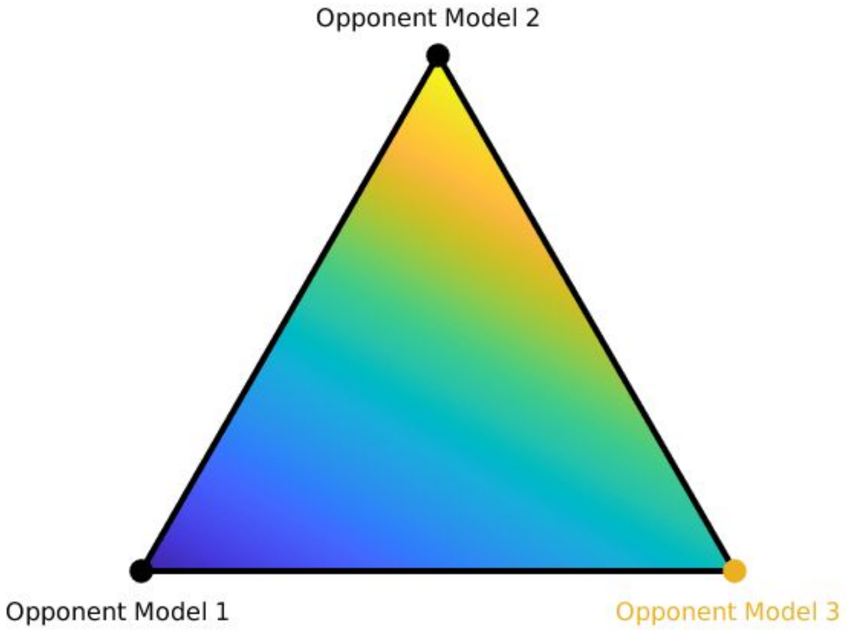
$$c_2(t;p) := -\sum_{s>t} \lambda^{s-t} \mathbb{E}[r(o(s);p)]$$

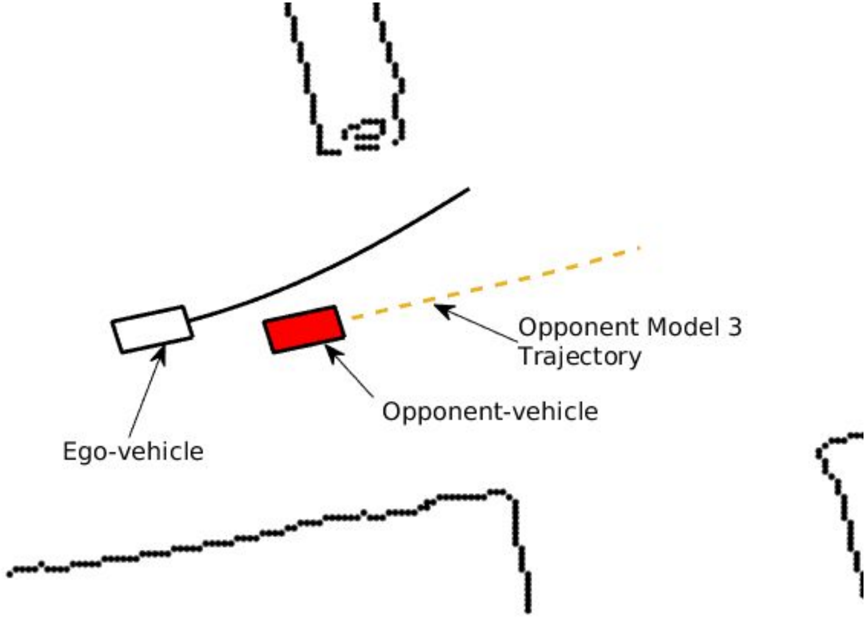




Distributionally Robust Trajectory Cost

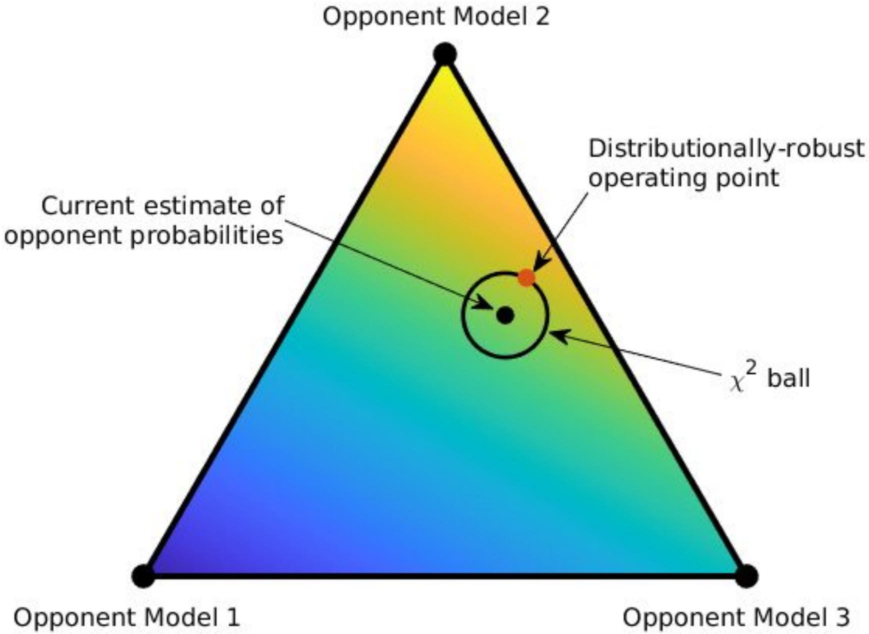
$$c_3(t; p) := -\sum_{s>t} \lambda^{s-t} \mathbb{E}[r(o(s); p)]$$

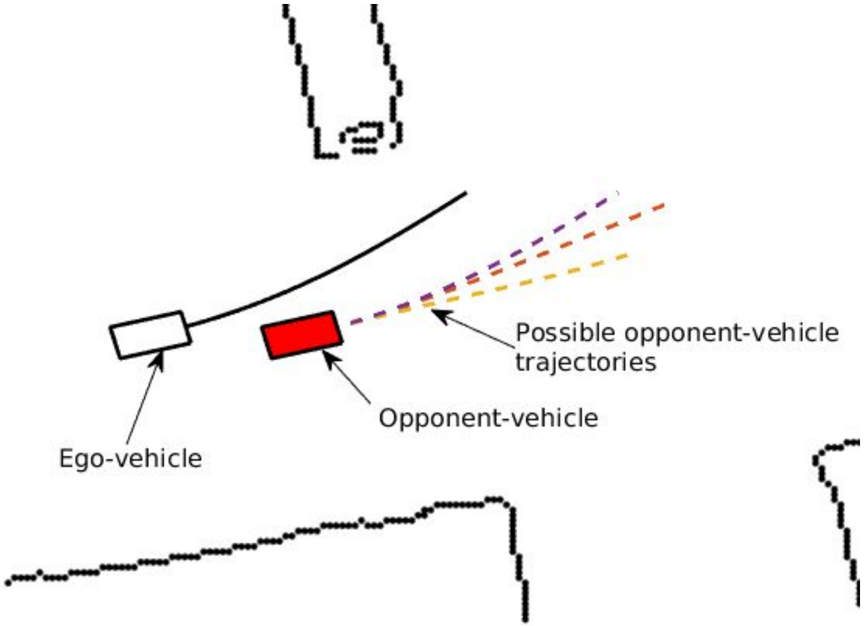




Distributionally Robust Trajectory Cost

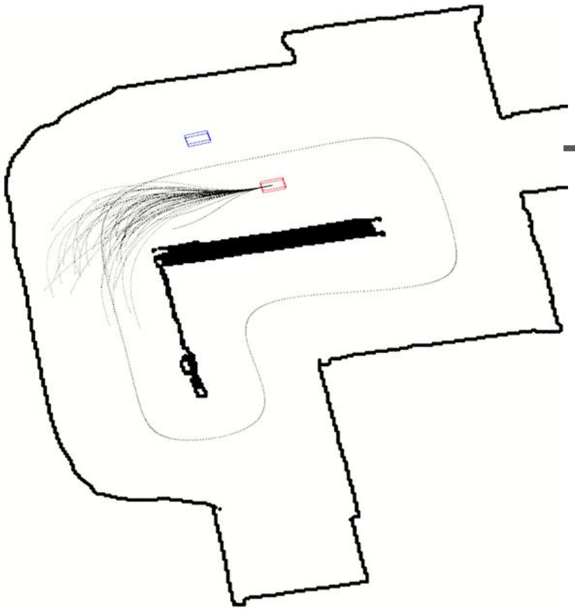
$$\sup_{q:\sum_i w_i \phi(\frac{q_i}{w_i}) \leq \rho} \sum_i q_i c_i(t;p)$$



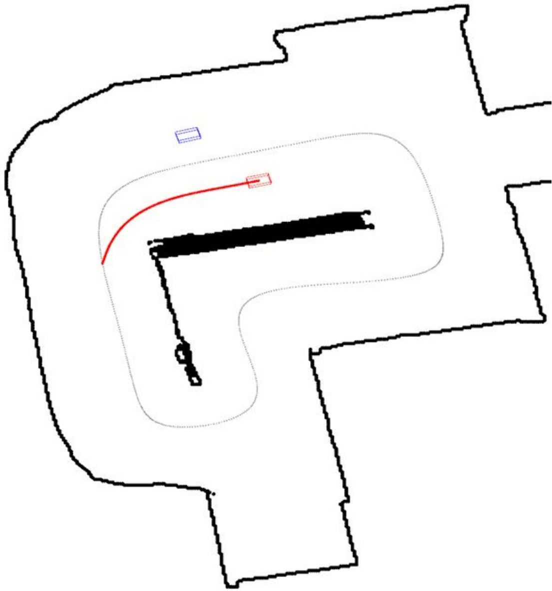


Distributionally Robust Trajectory Cost

We repeat this for every motion planning goal, and select the goal with the lowest robust cost.



$$\sup_{q:\sum_i w_i \phi(\frac{q_i}{w_i})\leq\rho} \sum_i q_i c_i(t;p)$$

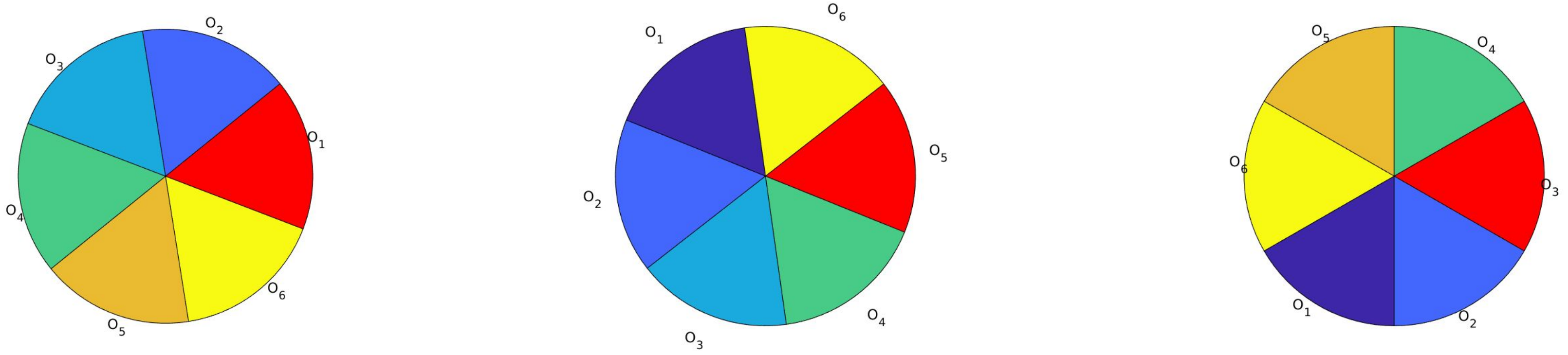


If there are 10 possible opponents and 100 possible motion planning goals we will need to compute 1000 receding horizon costs just to setup the problem!

Efficient Approximation of the Robust Cost

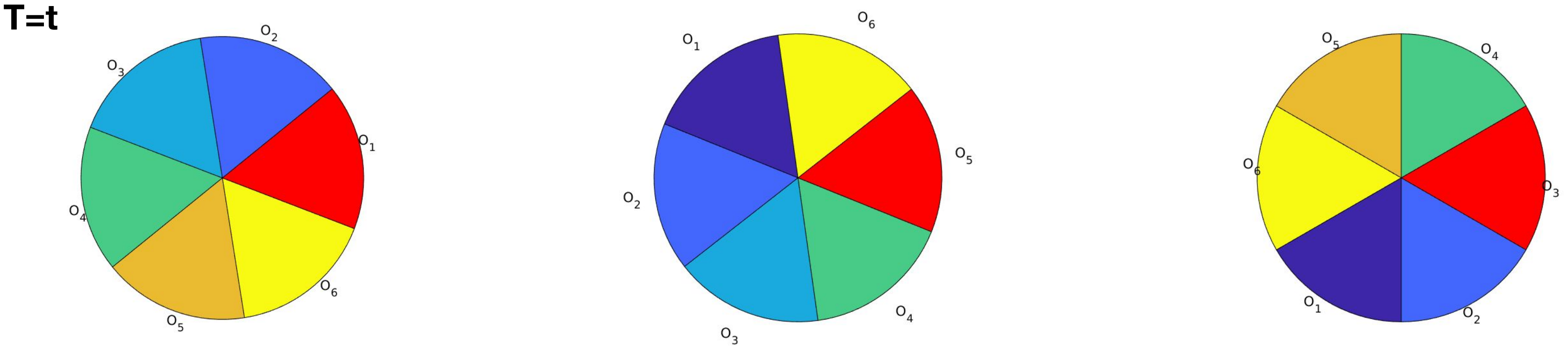
- **Challenge: what happens when there are many possible opponents?**
- At each time step we sample N<d opponent prototypes
- Beliefs begin as a uniform distribution

T=t



Select opponent 1,5, and 3… and compute: $\underset{q \in \mathcal{P}_{N_w}}{\text{maximize}} \sum_k q_k c_{j_k}(t; p)$

Efficient Approximation of the Robust Cost



Select opponent 1,5, and 3… and compute: $\underset{q \in \mathcal{P}_{N_w}}{\text{maximize}} \sum_k q_k c_{j_k}(t;p)$

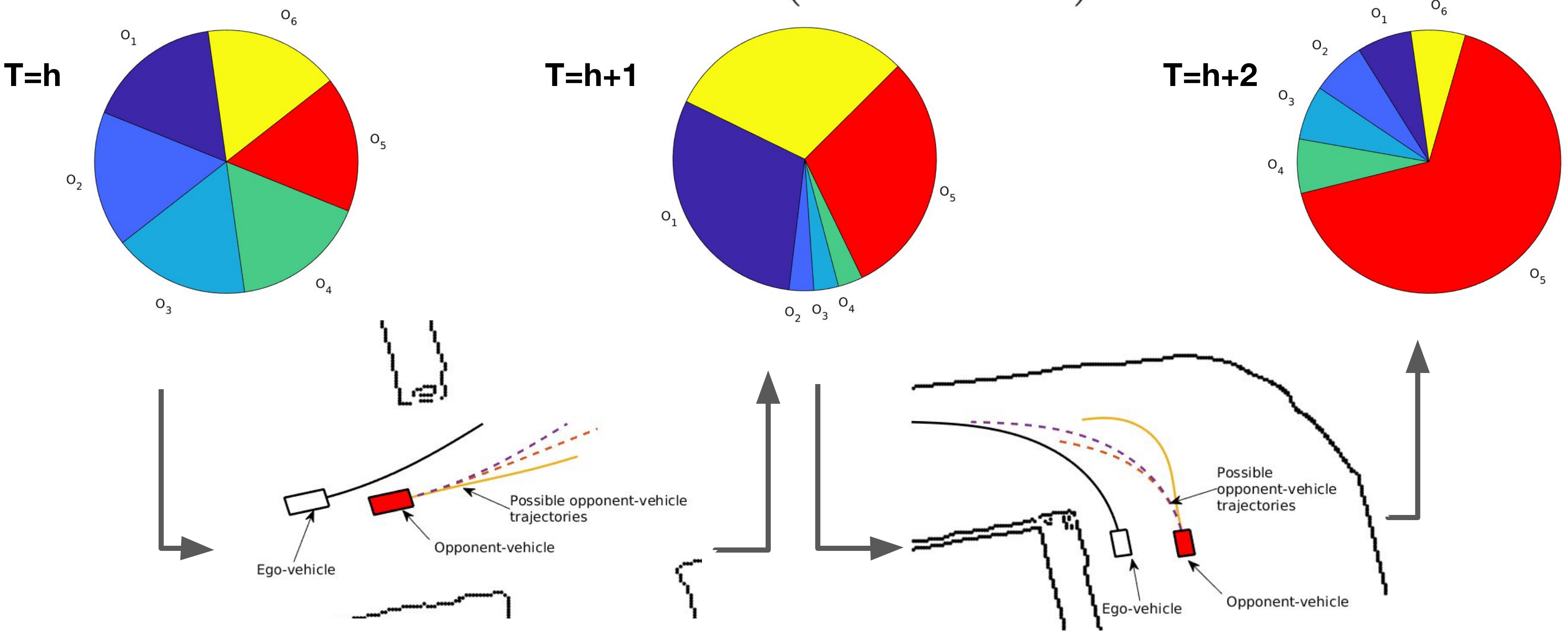
Proposition 1 shows we can bound the approximation quality, see the paper for details:

$$\left| \sup_{q \in \mathcal{P}_{N_w}} \hat{R}(q;p) - \sup_{Q \in \mathcal{P}} R(Q;p) \right| \leq 4A_\rho \sqrt{\frac{\log(2N_w)}{N_w}} + B_\rho \sqrt{\frac{\log \frac{2}{\delta}}{N_w}}$$

Online Adaptation

At each timestep, compute likelihood that the real trajectory was generated by prototype i:

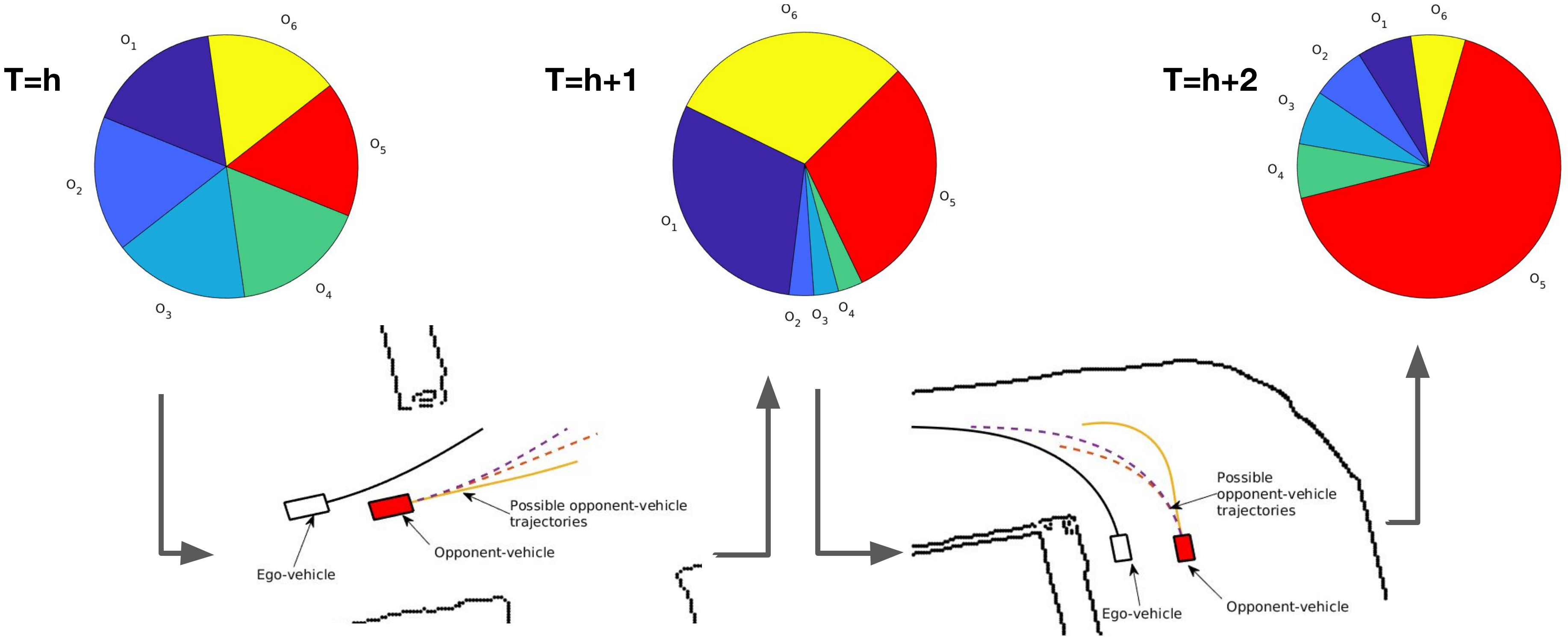
$$l_i^h(t) = \log d\mathbb{P}\left(o_{\mathrm{opp}}^h(t) | G(\theta^{1,i})\right)$$



Online Adaptation

Then we can construct an unbiased estimate of subgradient:

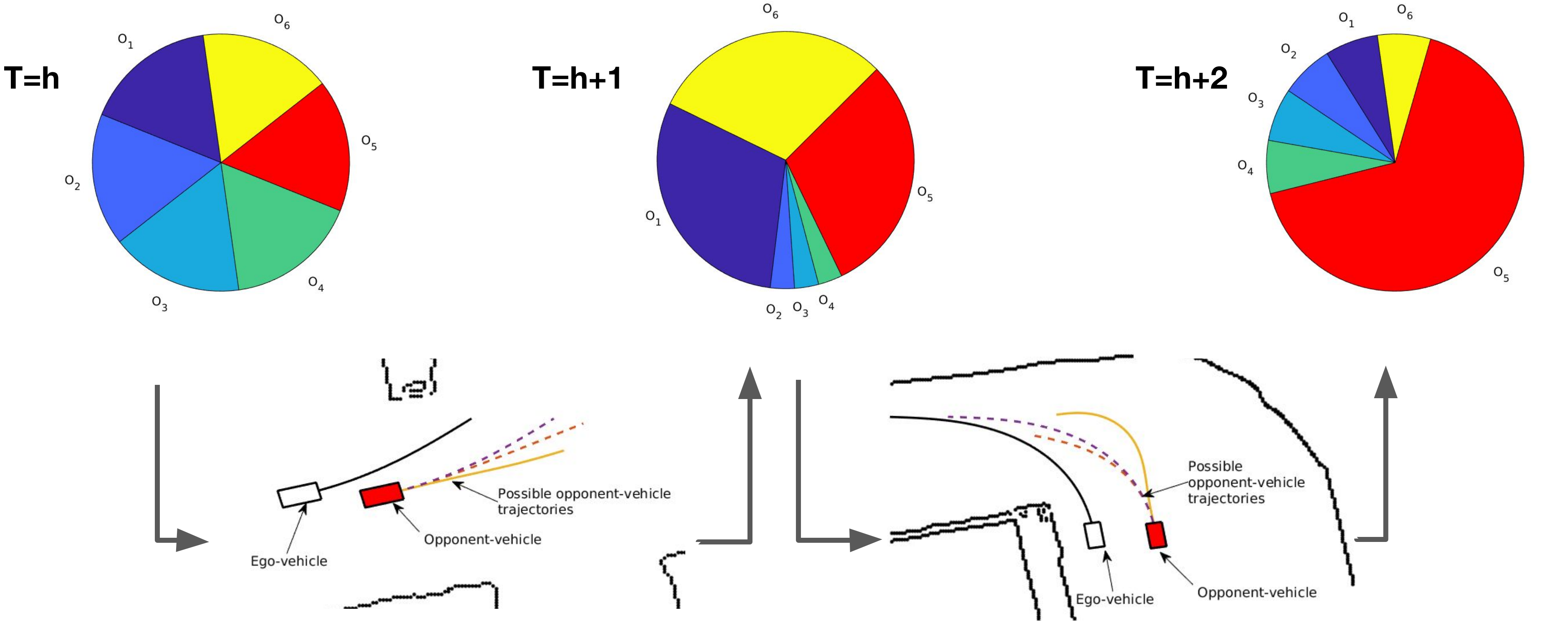
$$L_i(t) := 1 - l_i^h(t)/\bar{l} \qquad \gamma_i(t) = \frac{1}{N_w}\sum_{k=1}^{N_w} \frac{L_i(t)}{w_i(t)} \mathbf{1}\{J_k = i\}.$$



Online Adaptation

Update the belief vector using modified **EXP3** (Auer et al 2002):

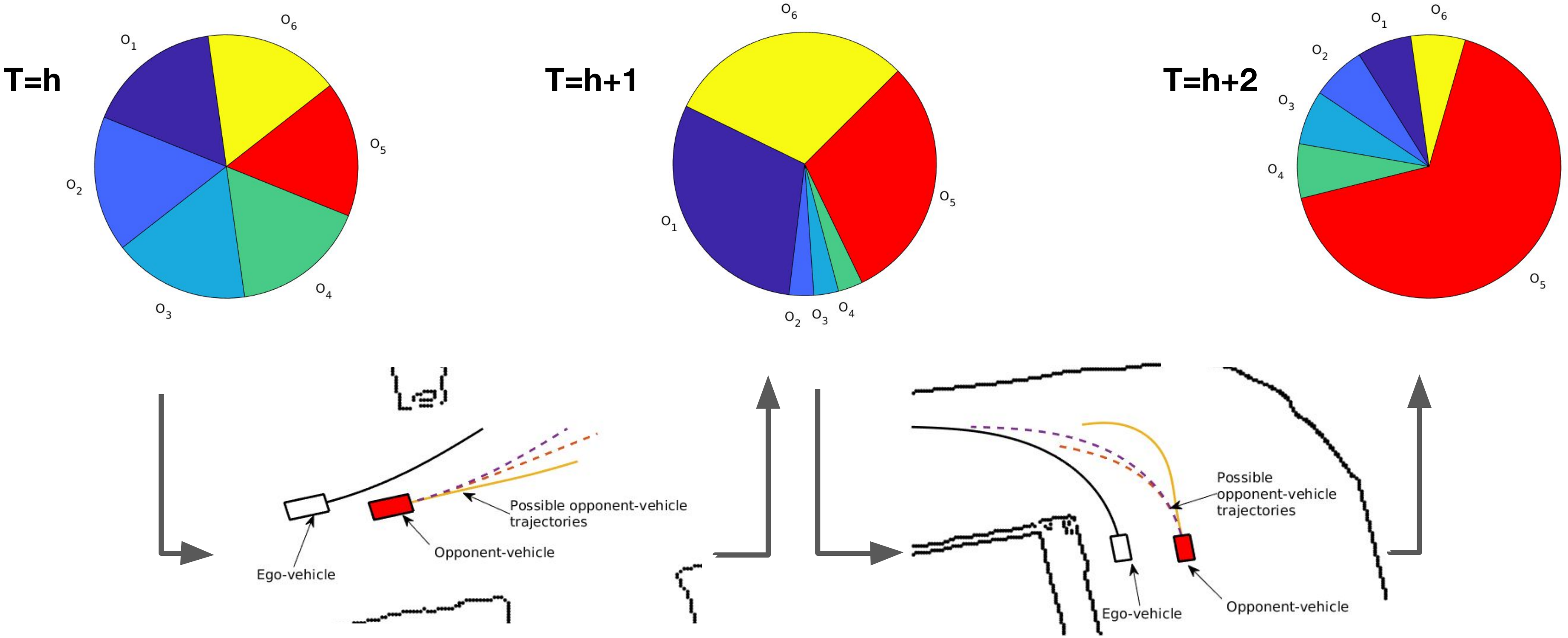
$$w_i(t+1) := \frac{w_i(t)\exp(-\eta_t \gamma_i(t))}{\sum_{j=1}^{d} w_j(t)\exp\big(-\eta_t \gamma_j(t)\big)}$$



Online Adaptation

With the following regret bound:

$$\sum_{t=1}^{T} \mathbb{E}\left[\gamma(t)^T (w(t) - w^\star)\right] \leq \sqrt{2zT\log(d)}$$



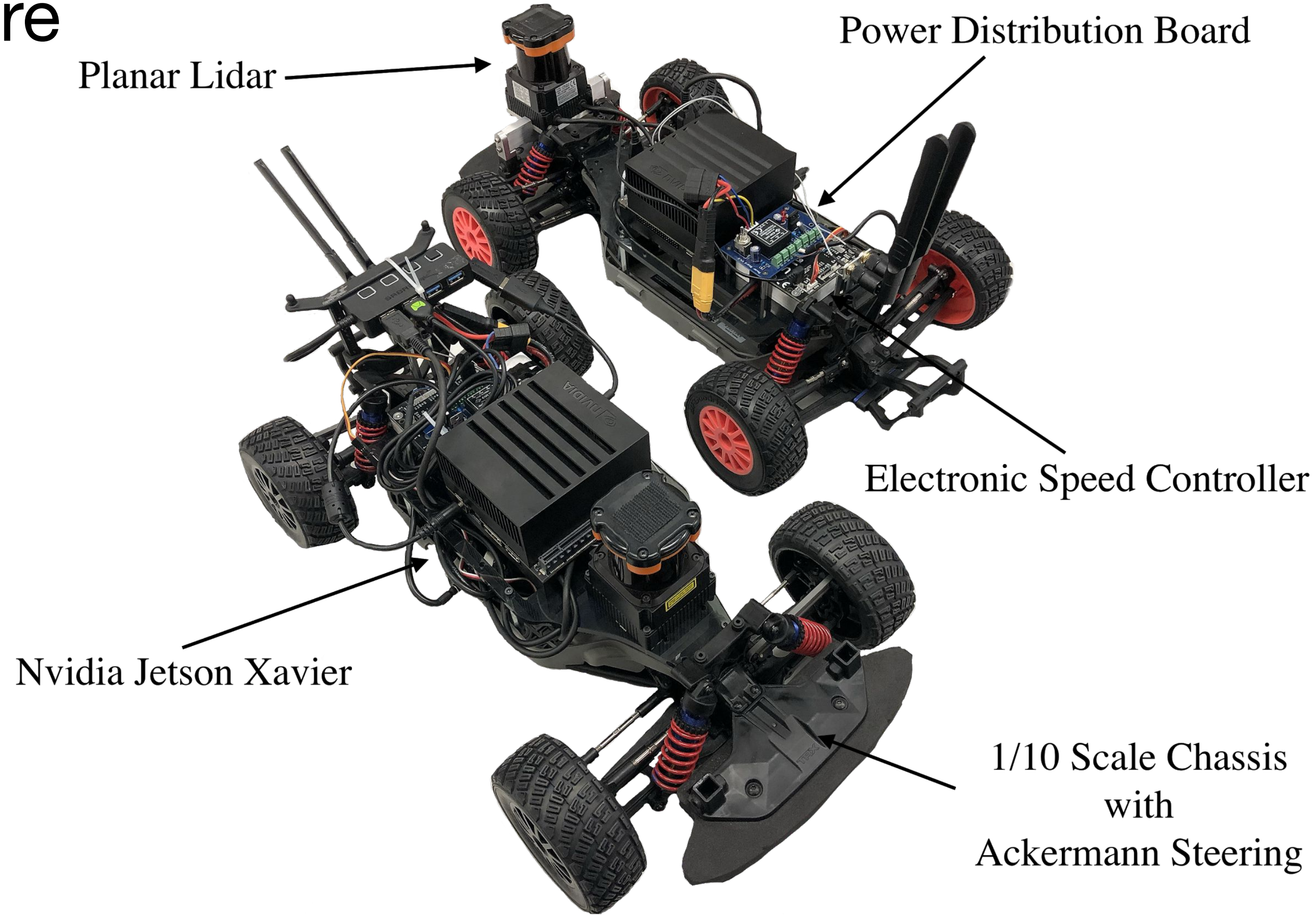
Overview

Population Synthesis

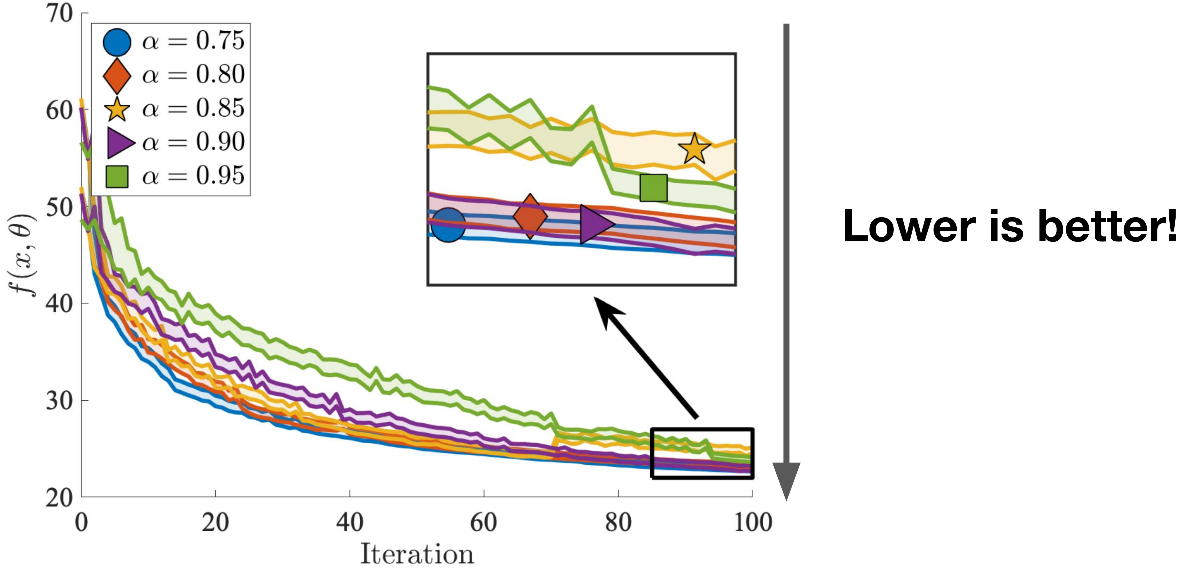
Online Adaptation

Experiments

Hardware

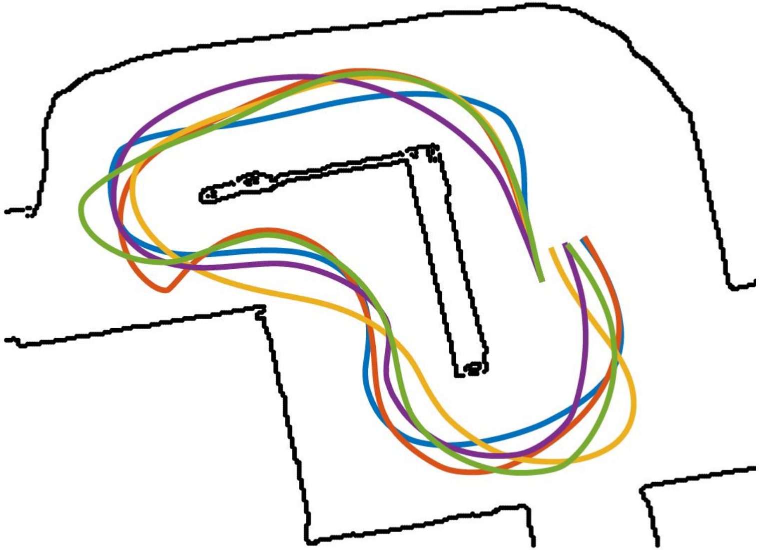

Planar Lidar
Power Distribution Board
Electronic Speed Controller
Nvidia Jetson Xavier
1/10 Scale Chassis
with
Ackermann Steering

Population synthesis results

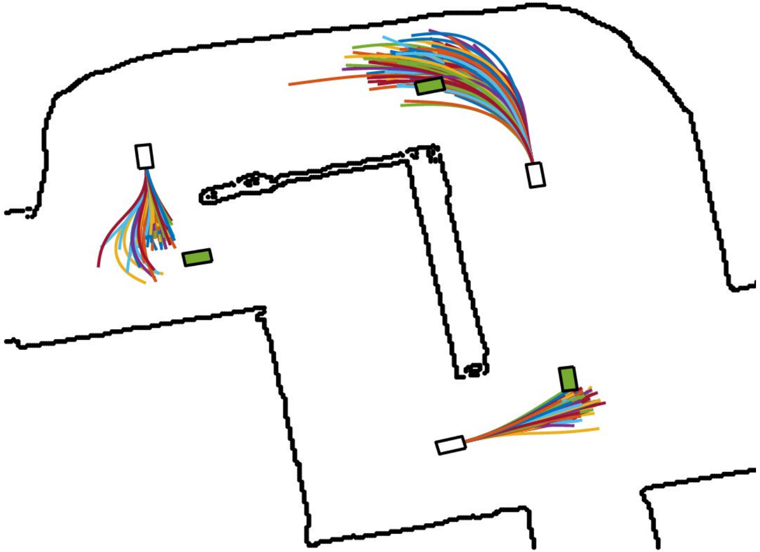


Decrease in average race times over the course of training.

Illustrations of diversity

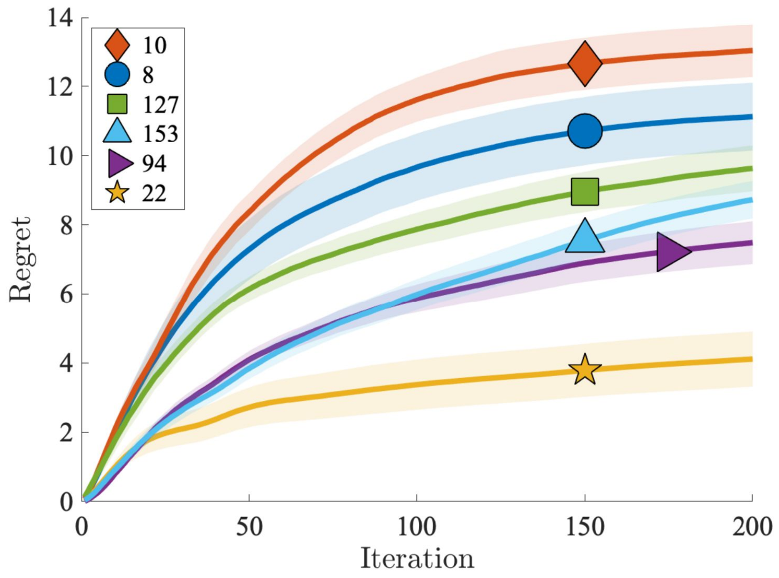


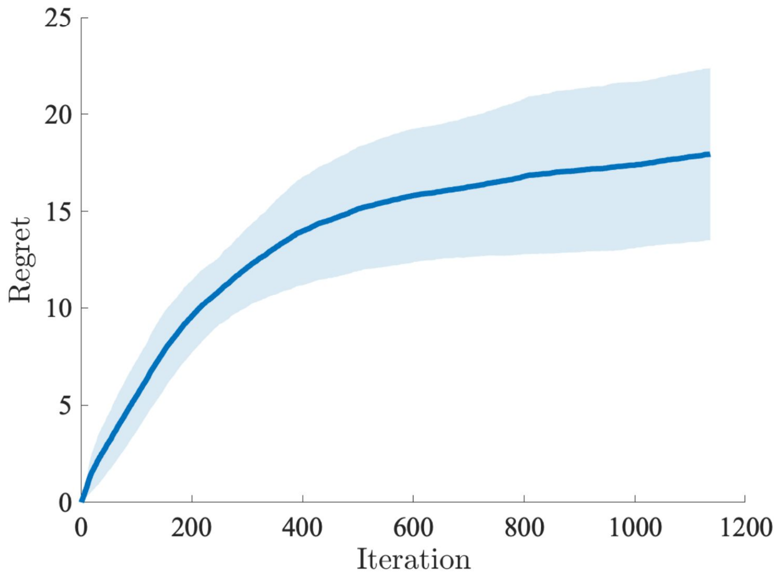
Diversity in performing a lap in isolation (no opponents)



Diversity in maneuvering near an opponent

Regret for opponent identification





In simulation we can identify the opponent model with only ~150 observations

In the real-world we also correctly identify the opponent, but it takes longer...

Balancing safety and performance

By actively identifying the opponent's strategy can we regain the performance of aggressive strategies without the downside of compromised safety?

Agent	% of iTTC values < 0.5s
$\rho/N_w = 0.001$	$\mathbf{7.86 \pm 0.90}$
$\rho/N_w = 0.025$	6.46± 0.78
$\rho/N_w = 0.2$	4.75± 0.65
$\rho/N_w = 0.4$	5.41± 0.74
$\rho/N_w = 0.75$	5.50± 0.82
$\rho/N_w = 1.0$	$\mathbf{5.76 \pm 0.84}$

The larger the robustness-ball the less frequently the agent experiences low time-to-collision events

Balancing safety and performance

By actively identifying the opponent's strategy can we regain the performance of aggressive strategies without the downside of compromised safety?

Agent	Win-rate Non-adaptive
$\rho/N_w = 0.001$	$\mathbf{0.593 \pm 0.025}$
$\rho/N_w = 0.025$	0.593 ± 0.025
$\rho/N_w = 0.2$	0.538 ± 0.025
$\rho/N_w = 0.4$	0.503 ± 0.025
$\rho/N_w = 0.75$	0.513 ± 0.025
$\rho/N_w = 1.0$	0.498 ± 0.025

Larger robustness-balls without adaptivity significantly reduce win-rate

Balancing safety and performance

By actively identifying the opponent's strategy we can regain the performance of aggressive strategies without the downside of compromised safety.

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	**0.593 ± 0.025**	0.588 ± 0.025	0.84
$\rho/N_w = 0.025$	0.593 ± 0.025	0.600 ± 0.024	0.77
$\rho/N_w = 0.2$	0.538 ± 0.025	0.588 ± 0.025	0.045
$\rho/N_w = 0.4$	0.503 ± 0.025	0.573 ± 0.025	0.0098
$\rho/N_w = 0.75$	0.513 ± 0.025	0.593 ± 0.025	0.0013
$\rho/N_w = 1.0$	0.498 ± 0.025	**0.590 ± 0.025**	0.00024

Online adaptivity preserves win rate even when the requested robustness level is high

Balancing safety and performance

By actively identifying the opponent's strategy we can regain the performance of aggressive strategies without the downside of compromised safety.

Agent	Win-rate Non-adaptive	Win-rate Adaptive	p-value
$\rho/N_w = 0.001$	**0.593 ± 0.025**	0.588 ± 0.025	0.84
$\rho/N_w = 0.025$	0.593 ± 0.025	0.600 ± 0.024	0.77
$\rho/N_w = 0.2$	0.538 ± 0.025	0.588 ± 0.025	0.045
$\rho/N_w = 0.4$	0.503 ± 0.025	0.573 ± 0.025	0.0098
$\rho/N_w = 0.75$	0.513 ± 0.025	0.593 ± 0.025	0.0013
$\rho/N_w = 1.0$	0.498 ± 0.025	**0.590 ± 0.025**	0.00024

Online adaptivity preserves win rate even when the requested robustness level is high

Putting it all together on a real racecar