Paper Presentation:

Short-Term Trajectory Planning in TORCS using Deep Reinforcement Learning

Emilio Capo
emilio.capo@mail.polimi.it
Computer Science and Engineering (CSE)

POLITECNICO MILANO 1863

HONOURS PROGRAMME HP-SR in Information Technology

Outline

- Motivations
- Problem and Solution Design
- Experimental Setup
- Results

Motivations

The Industry's Need for Coherent AI

Believability of racing games

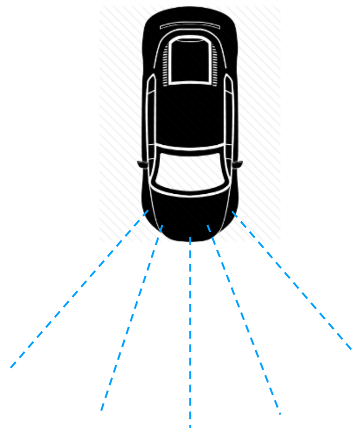
- **Physics:** High quality of simulation (aerodynamics, weather, collisions, …)
- **Graphics:** Aiming at photorealism
- **Real Pilots & Cars**

The problem of AI

- **Complexity of Simulation:** Developing an artificial agent is an hard task
- **Simplified Physics:** Using simplified physics models leads to incoherent behaviour

Racing AI: General Approach

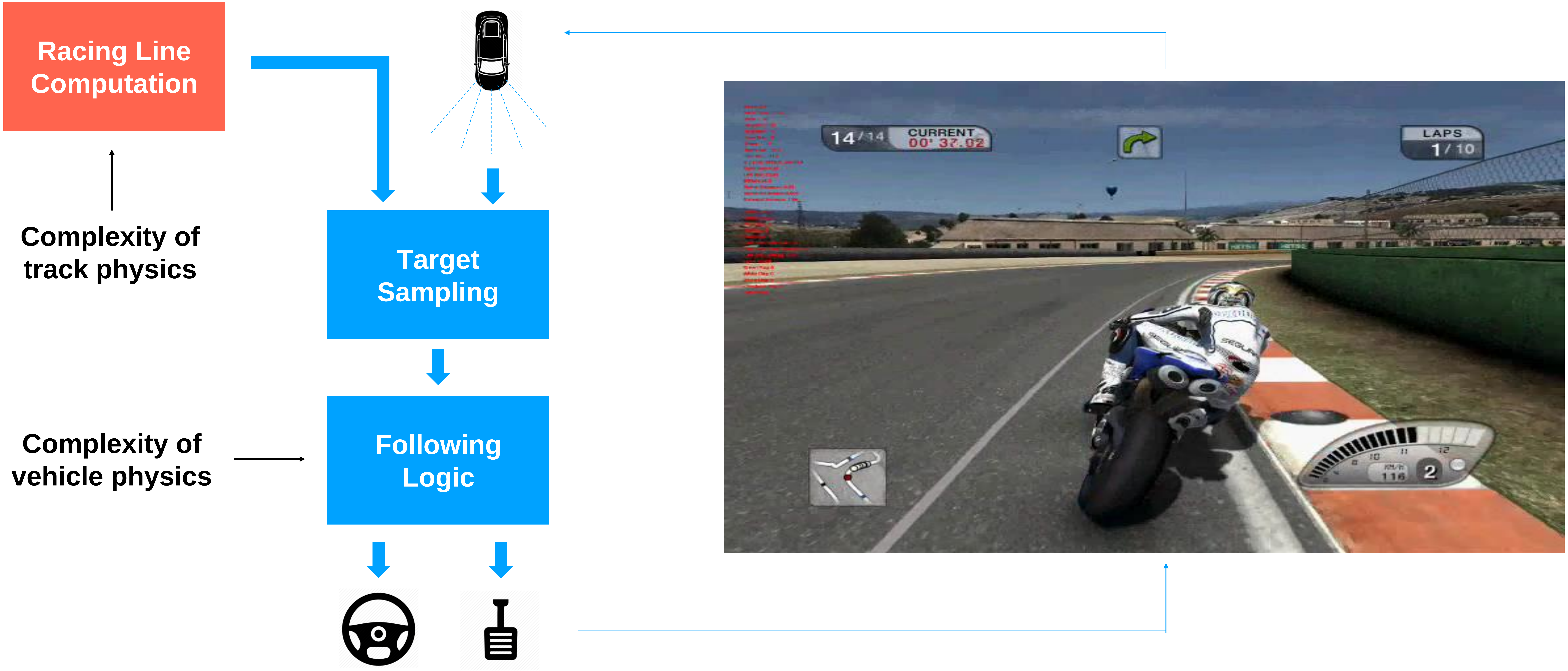
Racing Line Computation



Target Sampling

Following Logic

Racing AI: General Approach



Deep Reinforcement Learning

Reinforcement Learning

- **Abstraction from the environment**
- **Data generated through interaction**

Deep Learning

- **Management of large input spaces**
- **Huge amount of data needed**

Promising Approach

- **Solved Complex Problems** (Go, DOTA, …)
- **Simplifies Development**

Racing AI: General Approach

Racing Line Computation

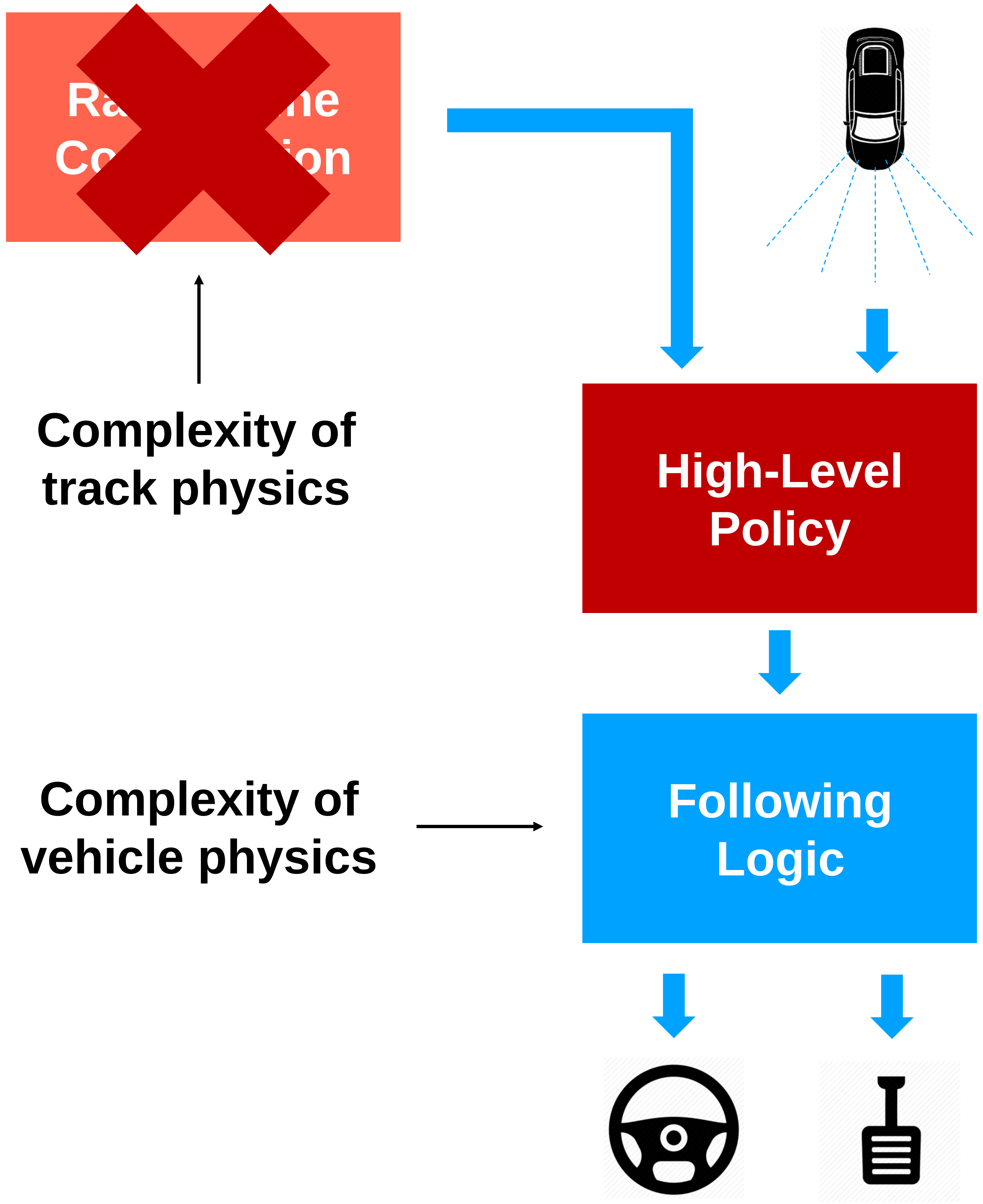
Complexity of track physics

Target Sampling

Following Logic

Complexity of vehicle physics

Racing AI: Our Approach



Problem and Solution Design

The Open Racing Car Simulator (TORCS)

Open-Source Racing Simulator

- **Different Game Modes:** Practice, competition, etc.
- **Physics Engine:** Aerodynamics, traction, fuel, etc.

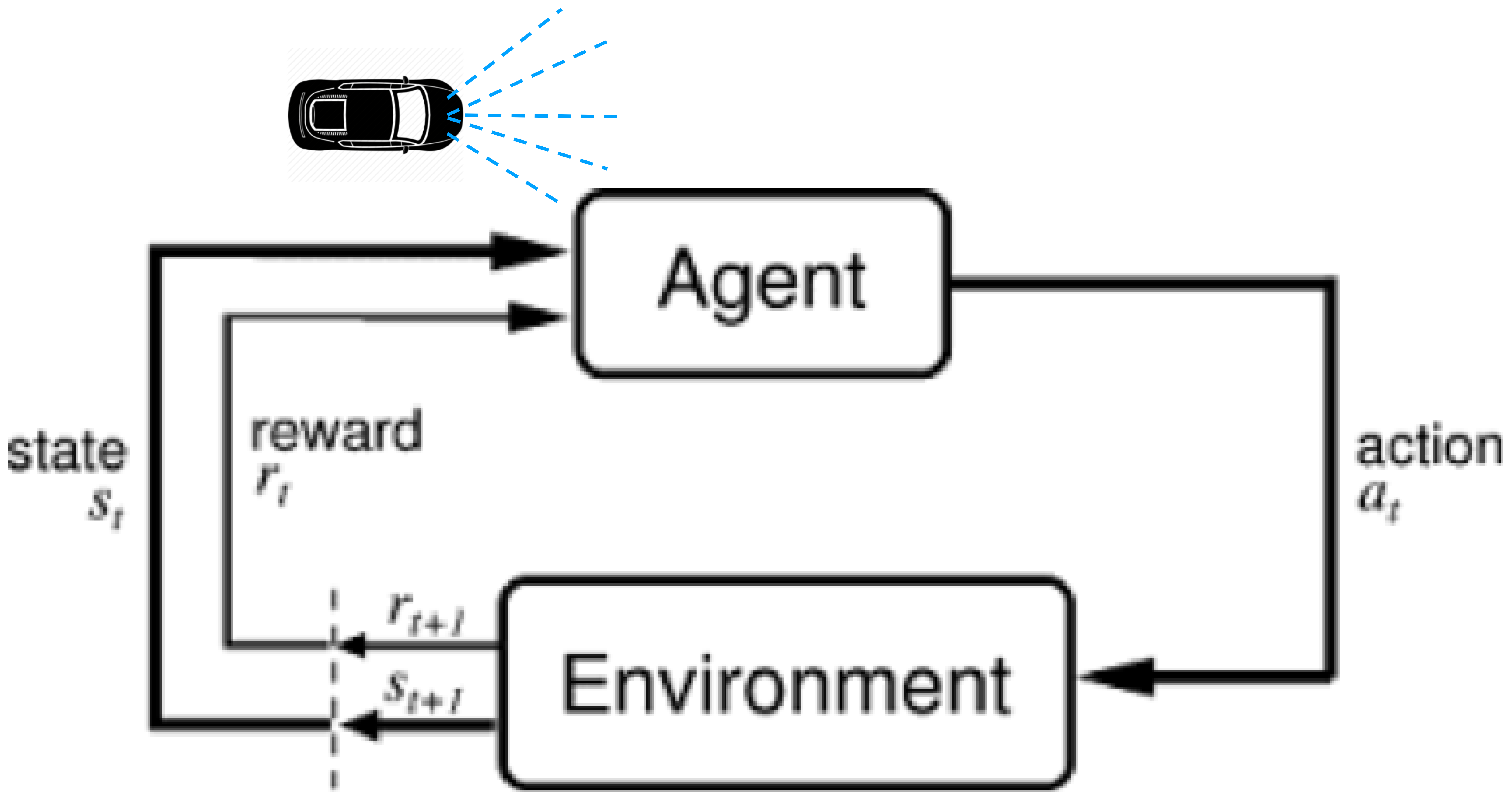
Client-Server Architecture

- **Server:** Wrapper providing numerical information to the client about the race (car, opponents, etc.)
- **Client:** Driving logic taking decisions based on the information received from the server

Reinforcement Learning Scheme

Critical Aspects

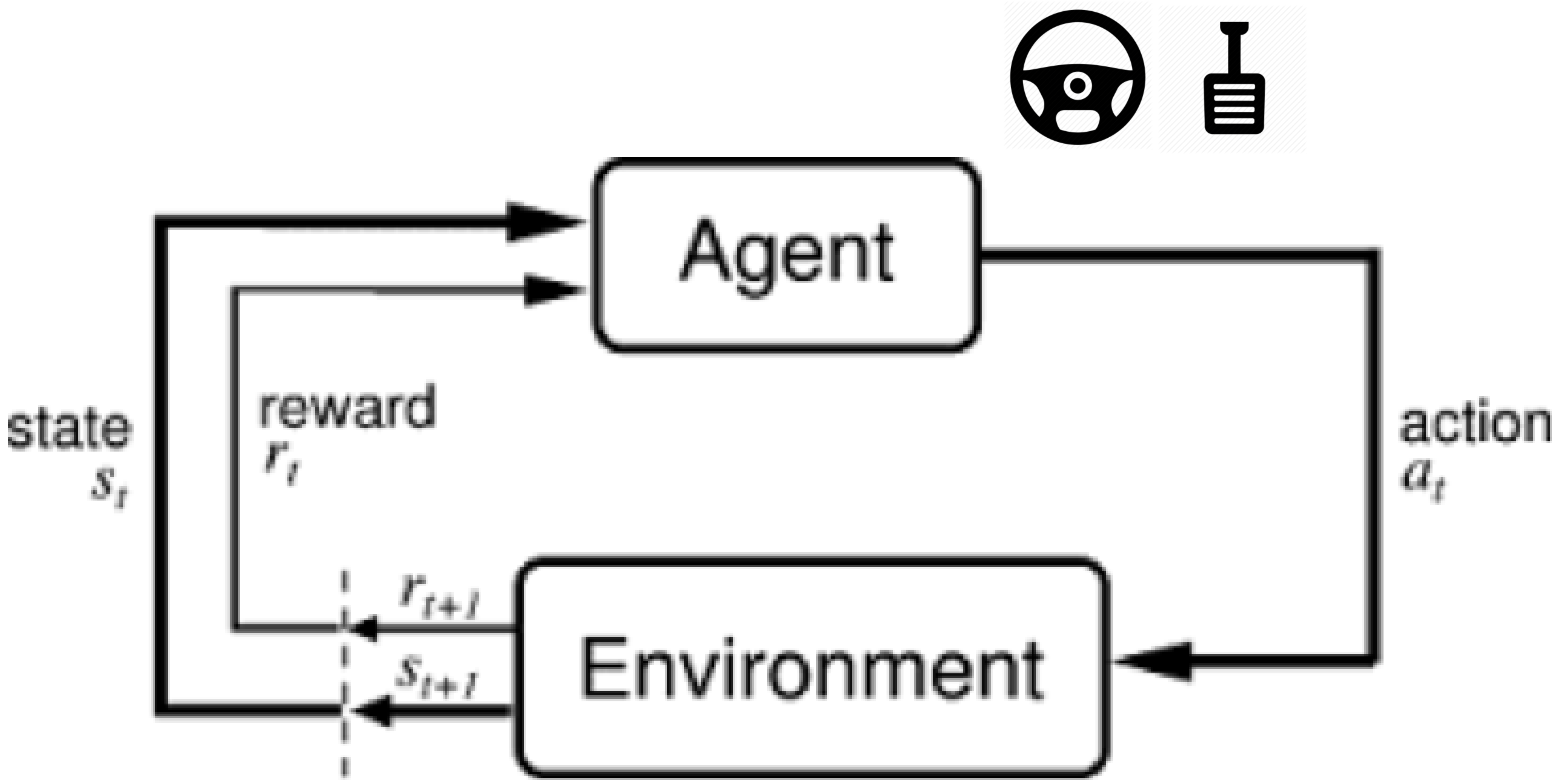
- **State Representation:** The information the agent can use to take decisions;



Reinforcement Learning Scheme

Critical Aspects

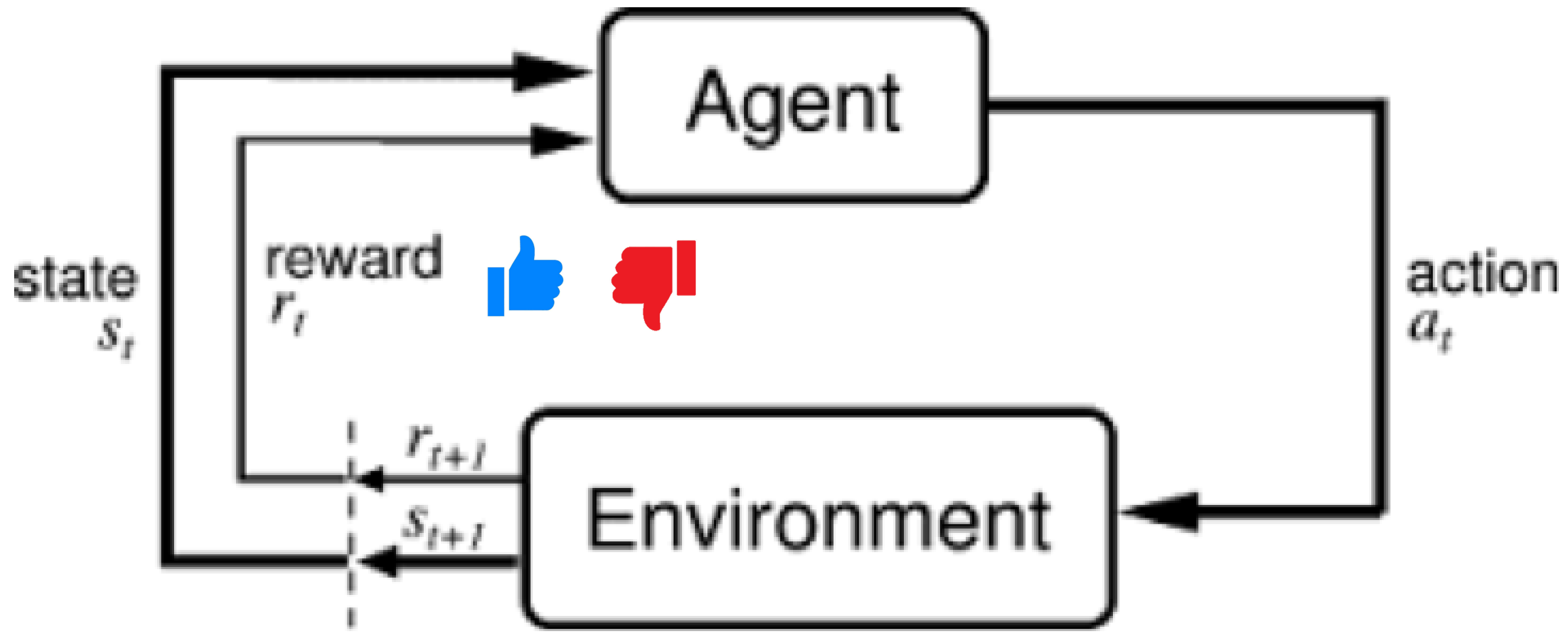
- **State Representation:** The information the agent can use to take decisions;
- **Action Space:** How the agent can interact with the environment;



Reinforcement Learning Scheme

Critical Aspects

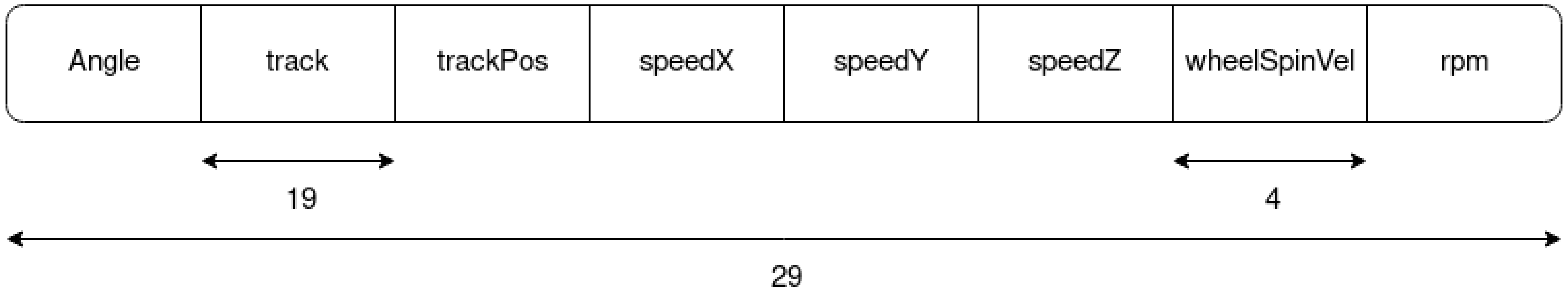
- **State Representation:** The information the agent can use to take decisions;
- **Action Space:** How the agent can interact with the environment;
- **Reward Function:** How to inform the agent about the efficiency of the decisions taken.



State Representation

Numerical Representation

- **Telemetry information:** How the agent’s state is with respect to the environment
- **Internal information:** State of the agent itself

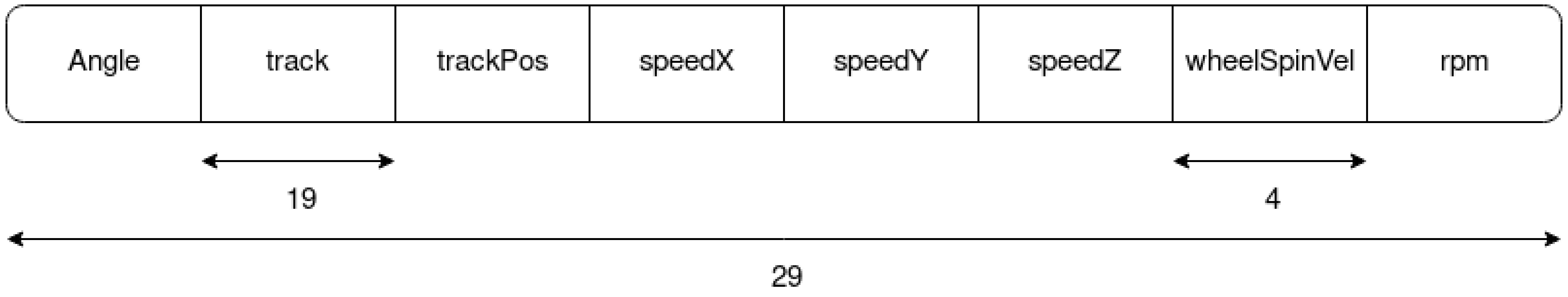


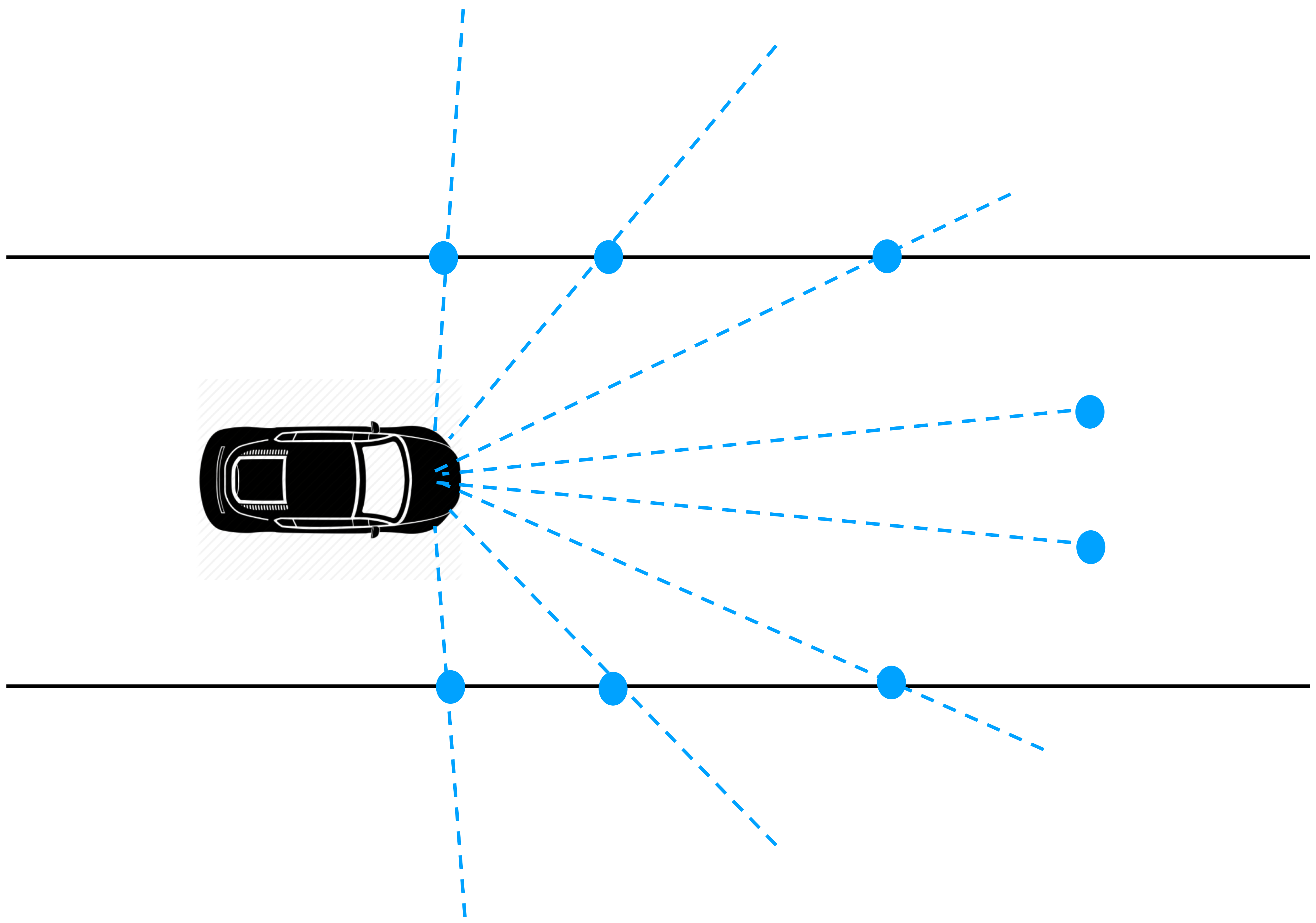
State Representation

Numerical Representation

- **Telemetry information:** How the agent's state is with respect to the environment

- **Internal information:** State of the agent itself

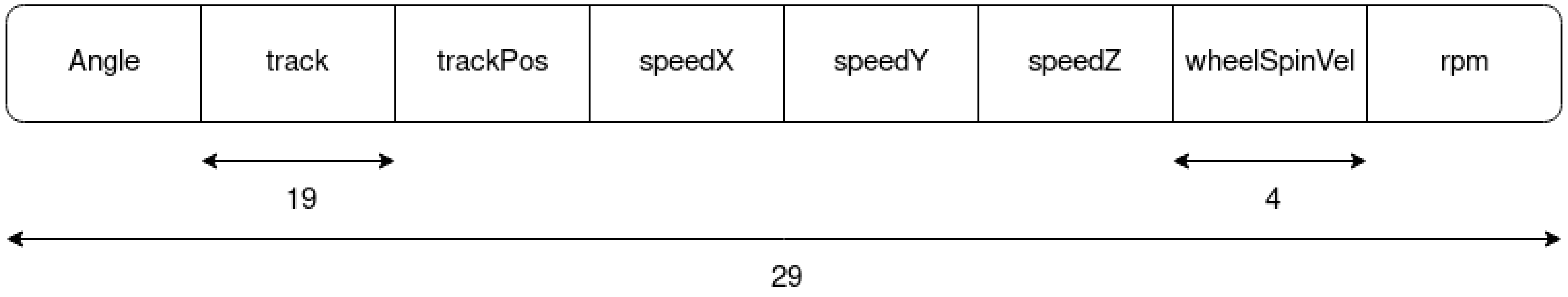




State Representation

Numerical Representation

- **Telemetry information:** How the agent's state is with respect to the environment
- **Internal information:** State of the agent itself



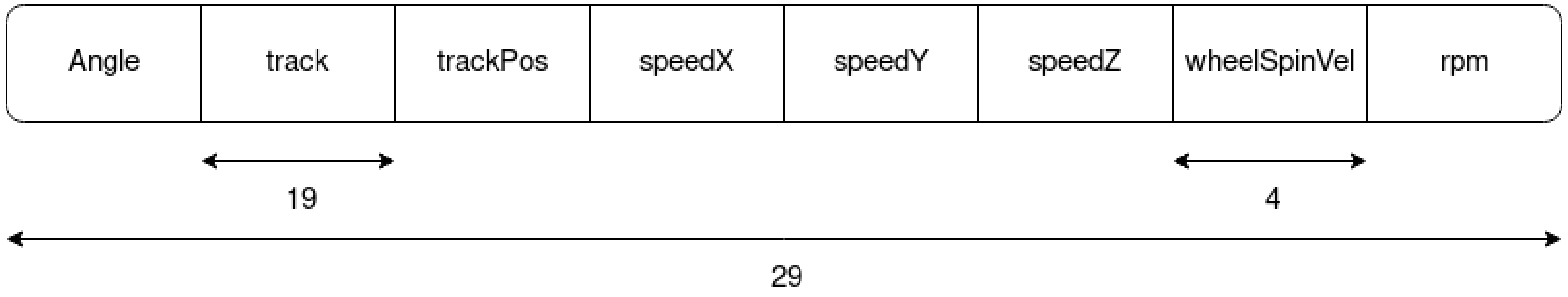
Hybrid Representation…

- **Image:** Telemetry information
- **Numerical:** Internal information

State Representation

Numerical Representation

- **Telemetry information:** How the agent's state is with respect to the environment
- **Internal information:** State of the agent itself



Hybrid Representation…

- **Image:** Telemetry information
- **Numerical:** Internal information

… With Racing Line Integration

- **Racing Line:** White
- **Proximity To Racing Line:** Gray

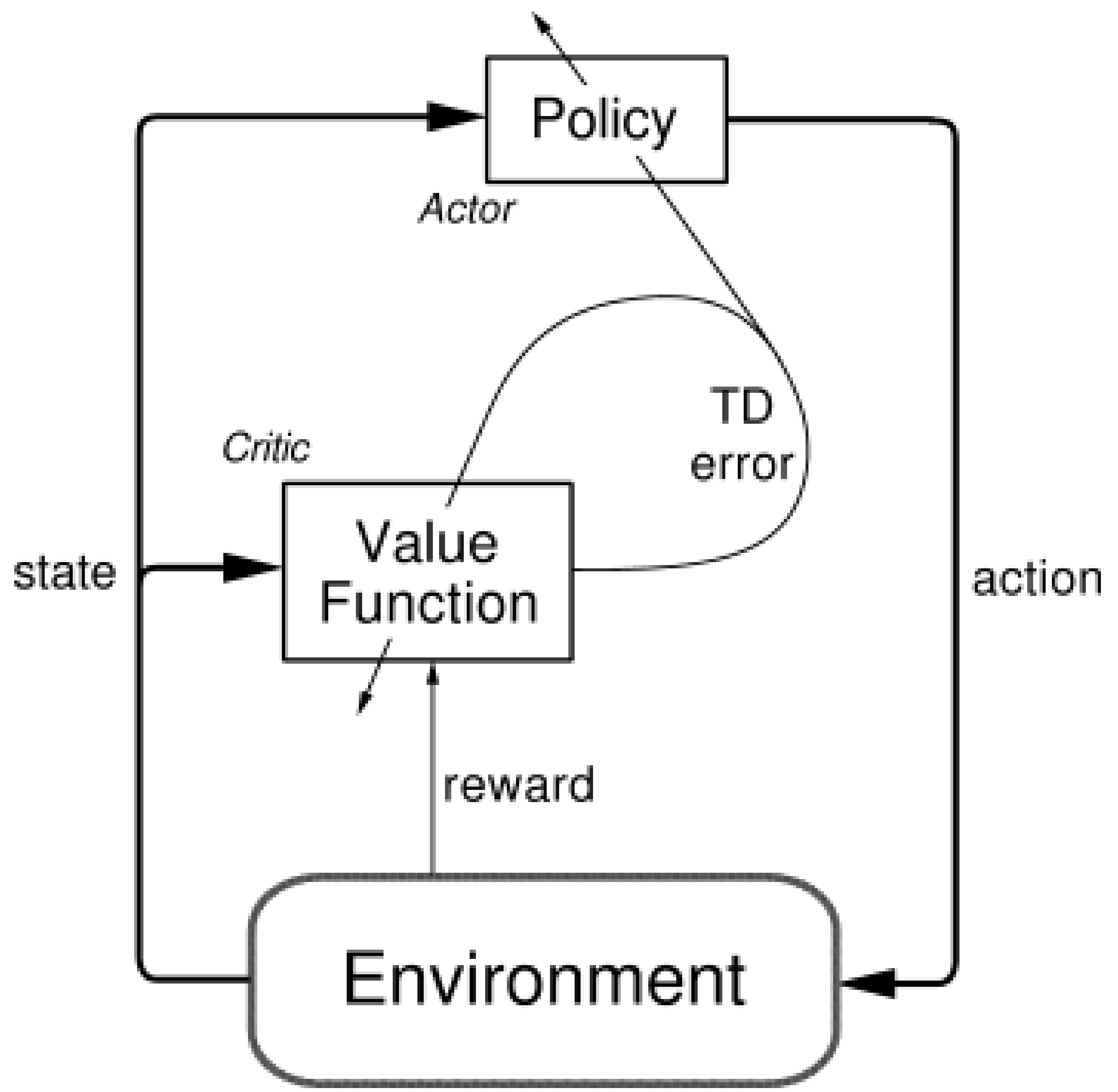
Learning Algorithm: DDPG

Actor-Critic Method

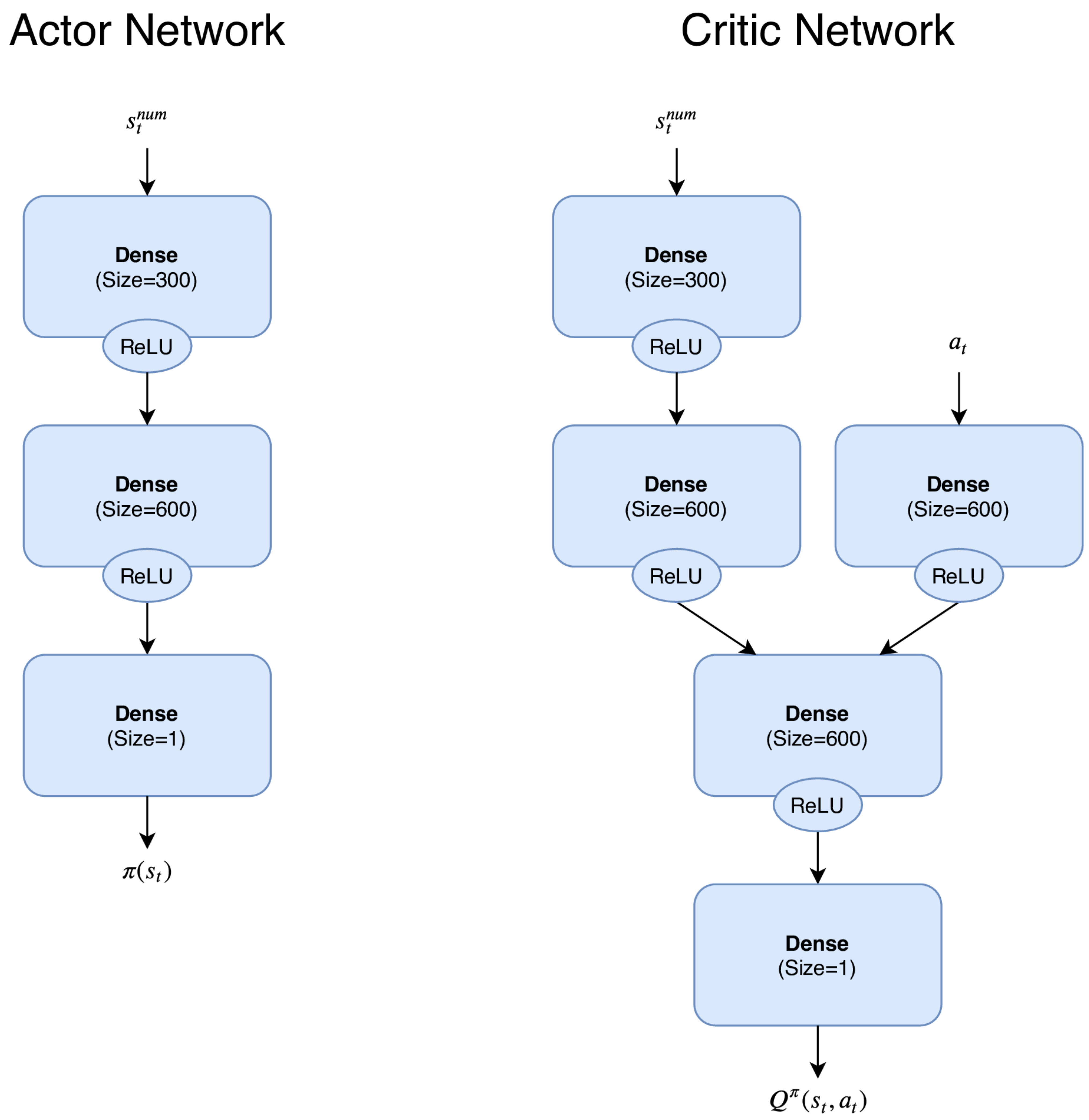
- **Actor Network:** Learns the driving policy
- **Critic Network:** Learns actions' profitability

Core Idea

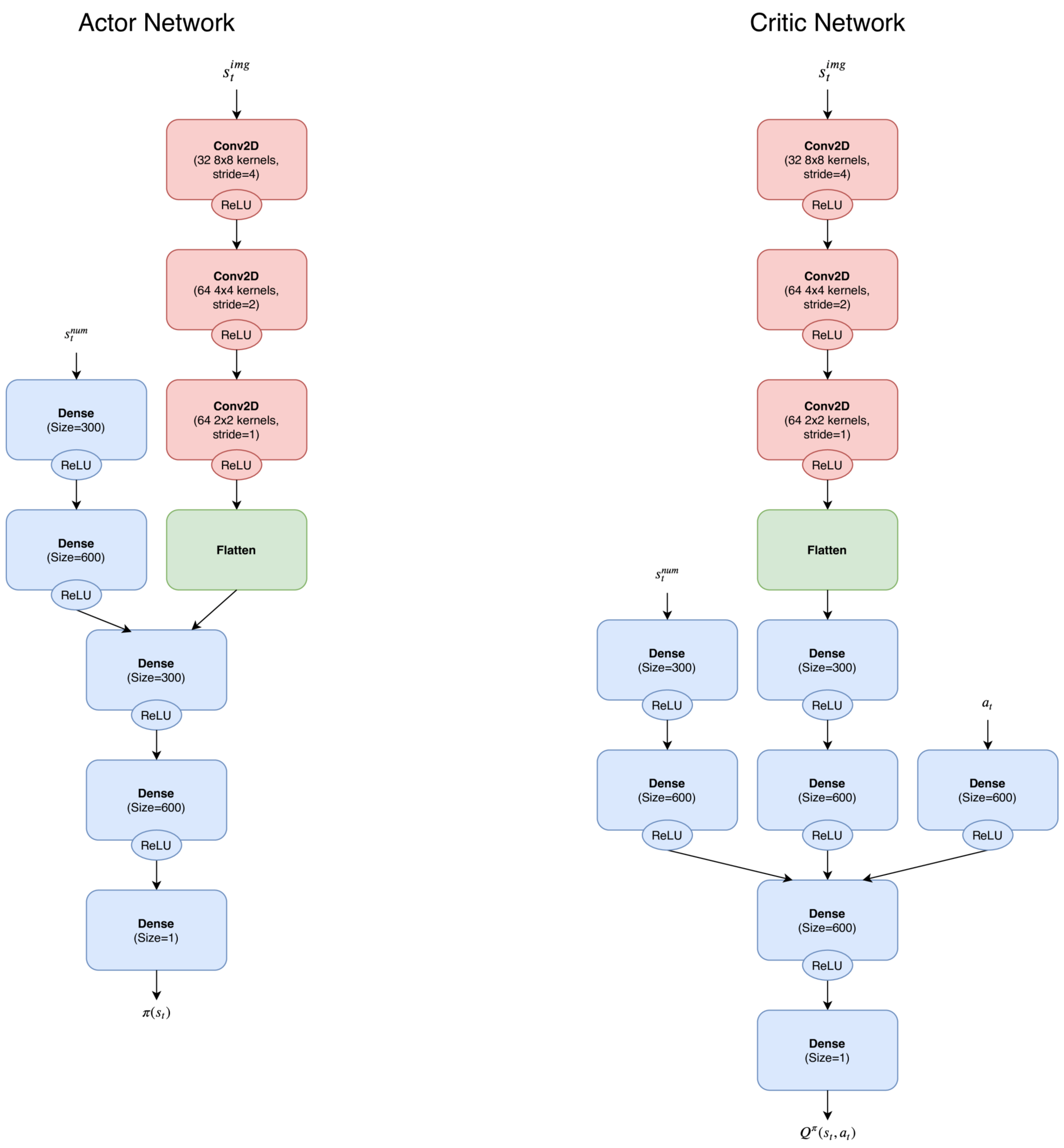
- Update the **Actor** towards the best actions according to the **Critic**
- Generate new experiences from the **Actor** to update the **Critic**



Numerical Networks



Hybrid Networks



Action Space

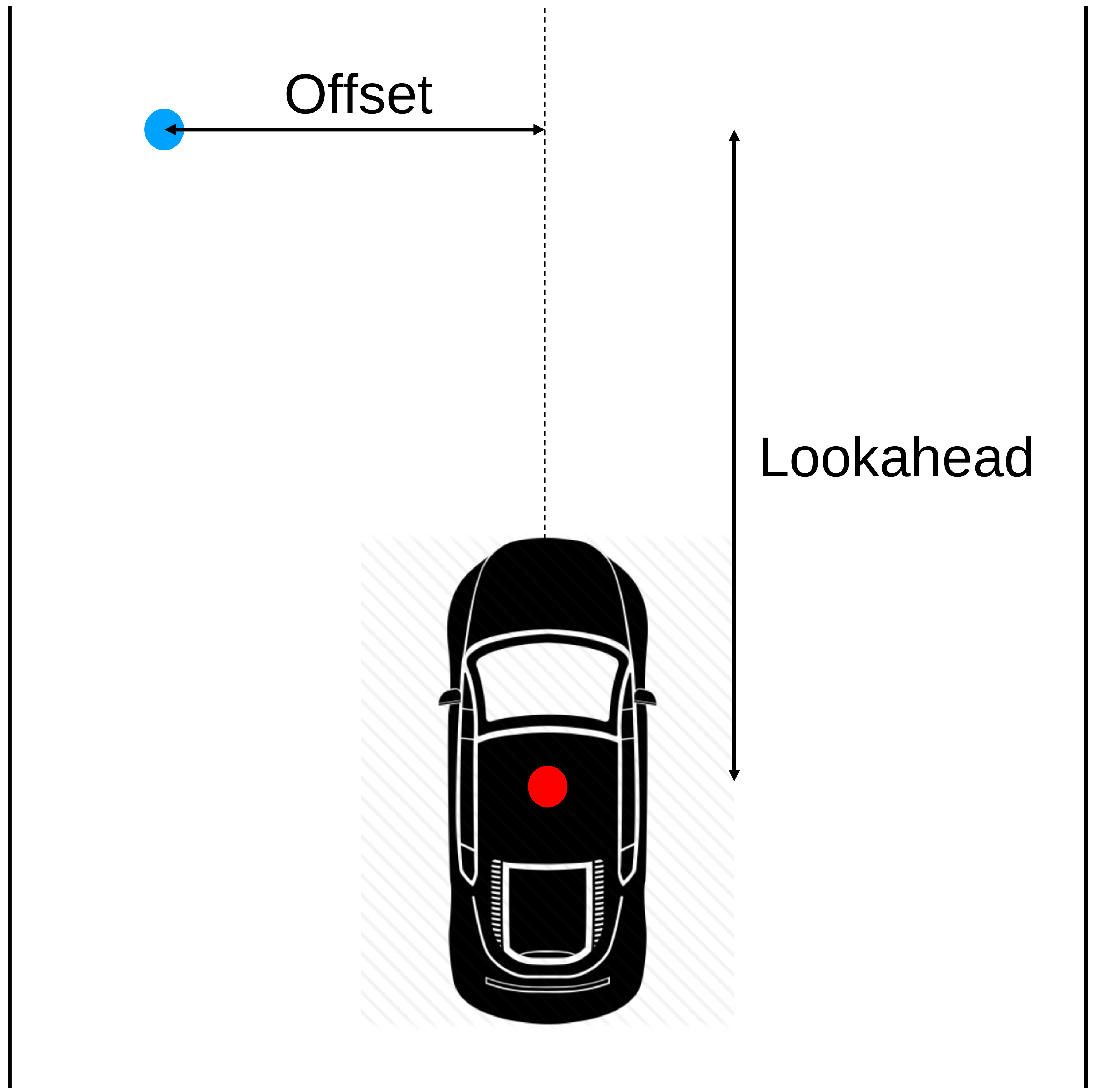
Single Output

- **Offset from track center:** [-1, 1]

Two Outputs

- **Offset from track center:** [-1, 1]
- **Target speed correction:** [-1, 1]

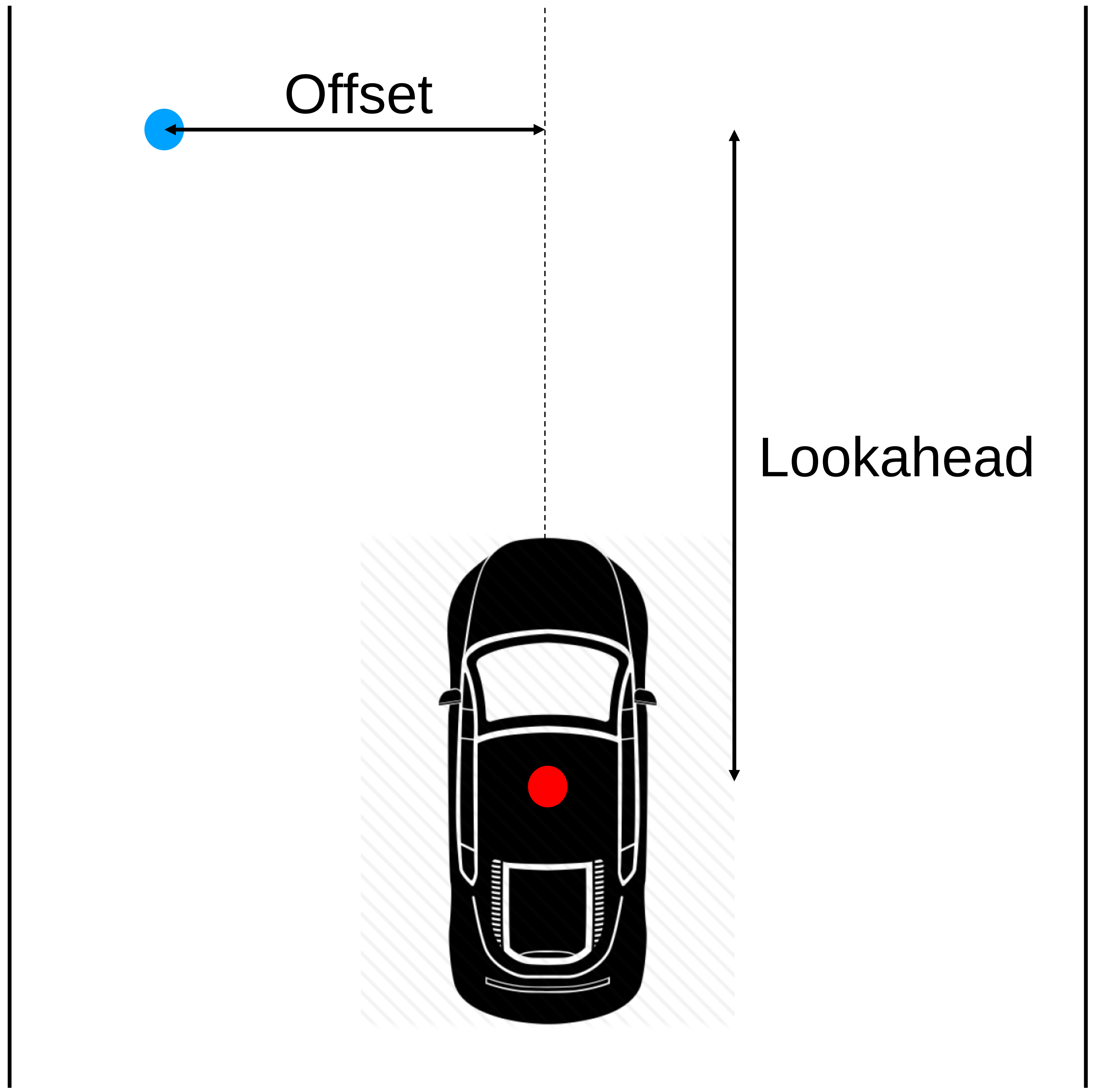
The Lookahead value is computed by the following logic at each step.



Following Logic

Lookahead Computation

$$LookAhead = LookBase + LookScale * currSpeed$$



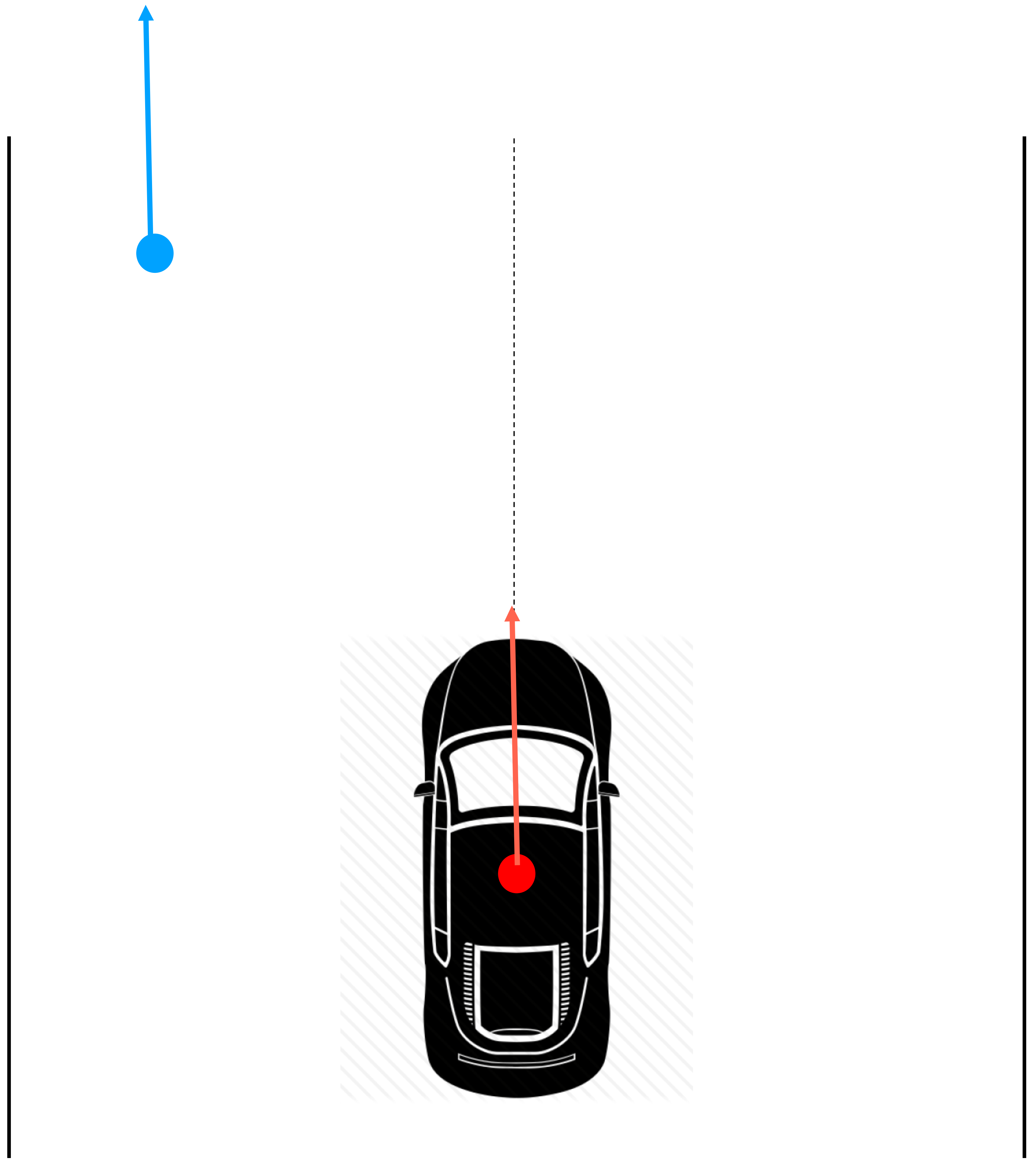
Following Logic

Lookahead Computation

$$LookAhead = LookBase + LookScale * currSpeed$$

Forward Step

- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)



Following Logic

Lookahead Computation

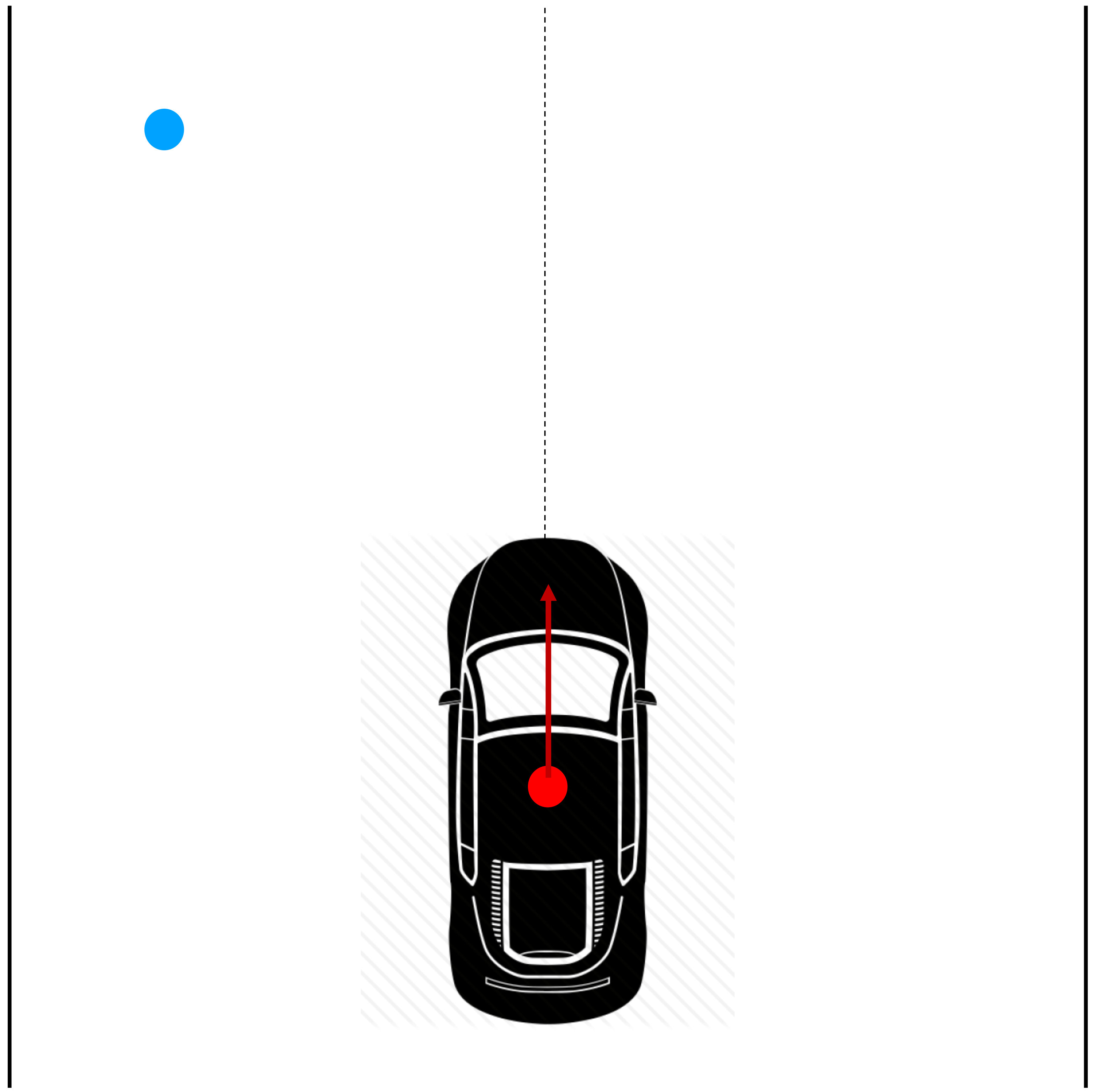
$$LookAhead = LookBase + LookScale * currSpeed$$

Forward Step

- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)

Backward Step

- Correct Current Target Speed



Following Logic

Lookahead Computation

$$LookAhead = LookBase + LookScale * currSpeed$$

Forward Step

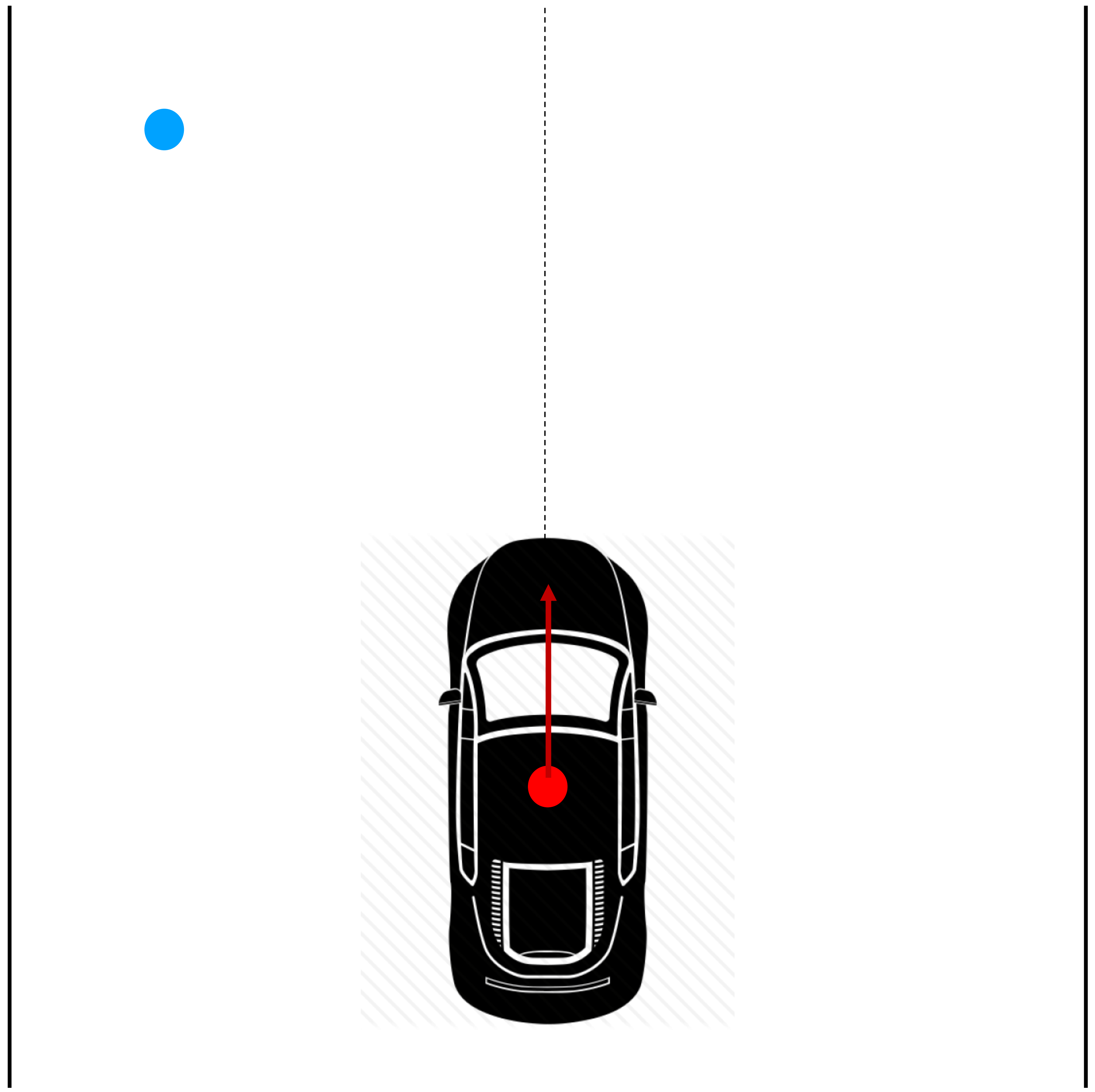
- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)

Backward Step

- Correct Current Target Speed

Heuristic

- Correct Current Target Speed according to proximity to the next corner



Following Logic

Lookahead Computation

$$LookAhead = LookBase + LookScale * currSpeed$$

Forward Step

- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)

Backward Step

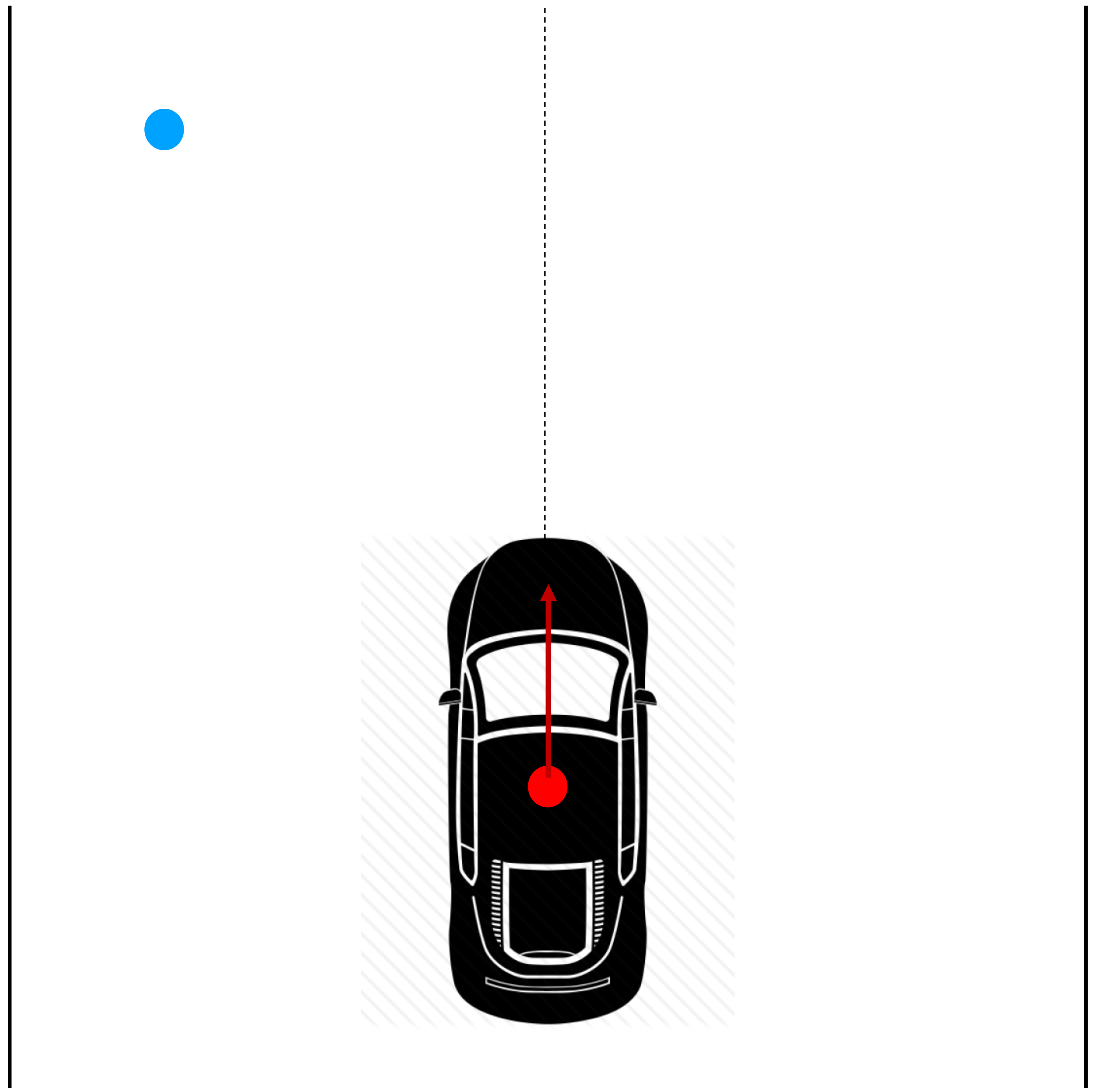
- Correct Current Target Speed

Heuristic

- Correct Current Target Speed according to proximity to the next corner

Agent Correction (Two-Outputs Agents)

$$targSpeed = targSpeed + corrDelta * speedCorr$$



Reward Function

Distance Raced

- P_{curr}: Current car position
- P_{prev} : Previous car position

$$\Delta distRaced = distRaced(P_{curr}) - distRaced(P_{prev})$$

Reward Function

Distance Raced

- $\boldsymbol{P_{curr}}$: Current car position
- $\boldsymbol{P_{prev}}$: Previous car position

$$\Delta distRaced = distRaced(P_{curr}) - distRaced(P_{prev})$$

Complete Reward Function

- **Colliding** (walls or obstacles)
- **Driving backwards**
- **Out of track**

$$r_t = \begin{cases} -100 & \text{if colliding or driving backwards} \\ -1 & \text{if out of track} \\ 100 \cdot \Delta distRaced & \text{otherwise} \end{cases}$$

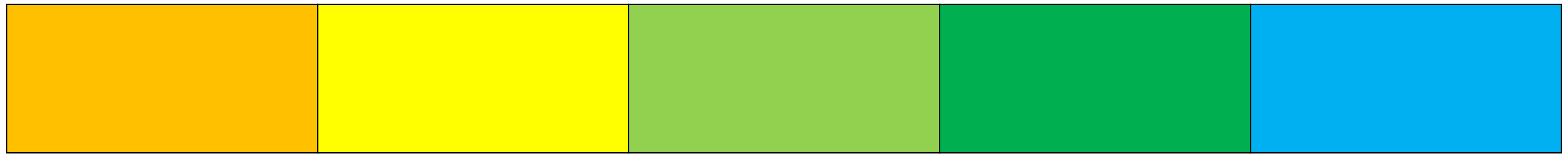
Experimental Setup

Training

Fixed Time Budget

- Each track is given a time budget
- This defines the number of steps that can be spent on that track

1 Batch = 5 Tracks



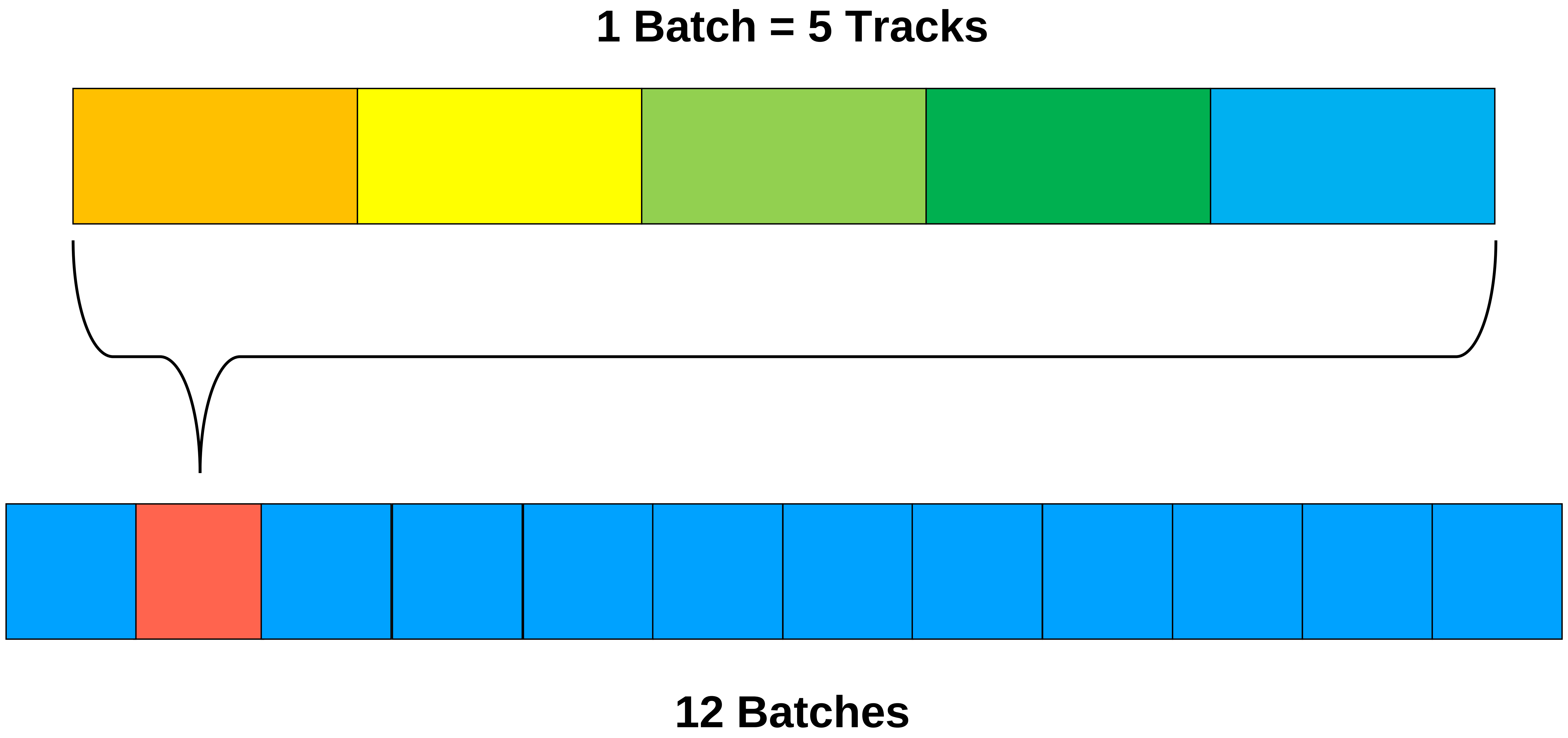
Training

Fixed Time Budget

- Each track is given a time budget
- This defines the number of steps that can be spent on that track

Uniform Experience

- All tracks are given the same total number of steps
- Avoids bias towards easier tracks



Training

Fixed Time Budget

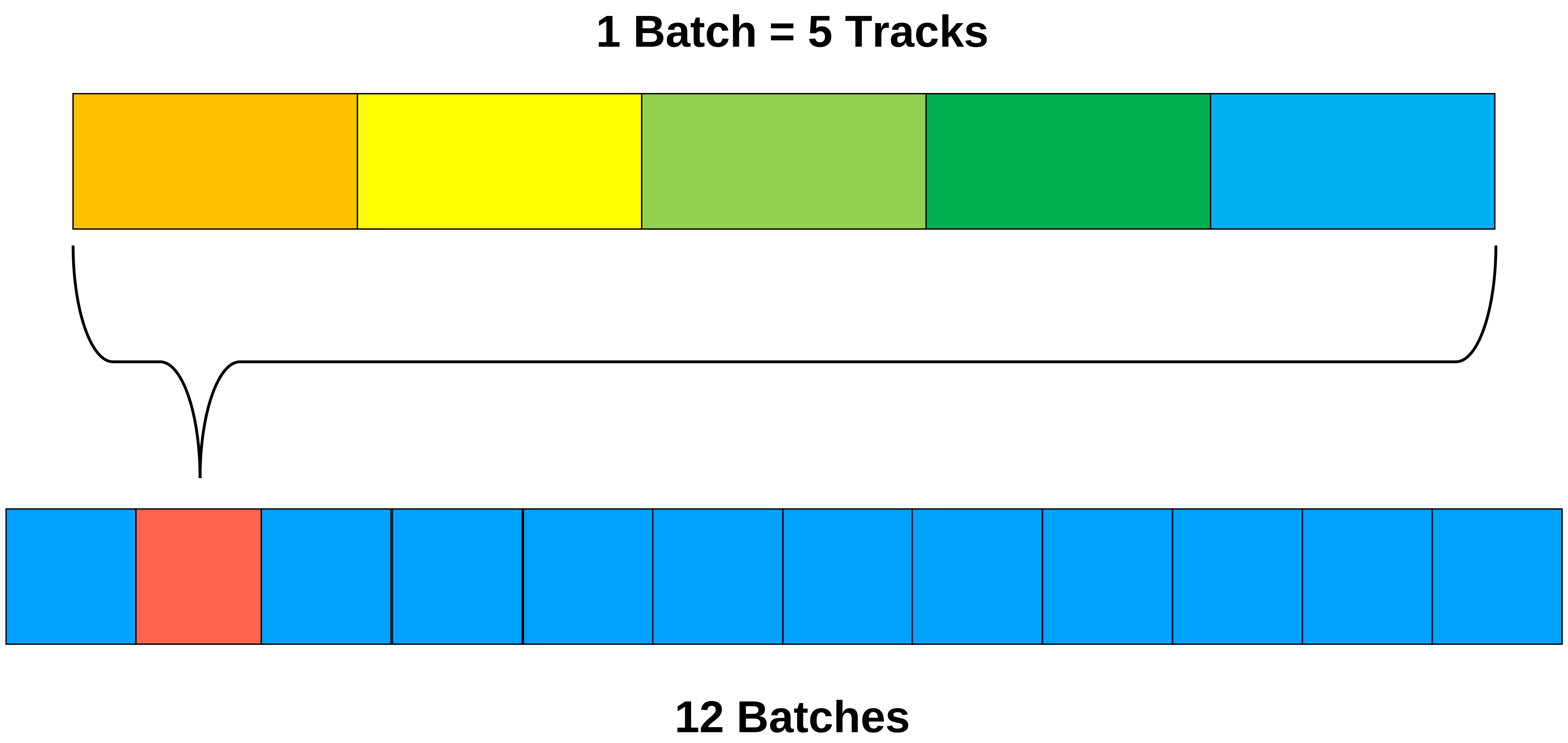
- Each track is given a time budget
- This defines the number of steps that can be spent on that track

Uniform Experience

- All tracks are given the same total number of steps
- Avoids bias towards easier tracks

Episode Termination

- Out of time budget
- Collision
- Driving backwards



Exploration Policy

Simple Gaussian Noise

- **μ:** 0
- **σ:** 0.2

$$a_t = \pi(s_t) + \alpha_t \varepsilon_t$$

Update Rule

- $\boldsymbol{T_{exp}}$ **:** 3 batches
- $\boldsymbol{\alpha_{max}}$**:** 1.0
- $\boldsymbol{\alpha_{min}}$ **:** 0.0

$$\alpha_t \leftarrow \max\left\{\alpha_{min},\ \alpha_t - \frac{\alpha_{max} - \alpha_{min}}{T_{exp}}\right\}$$

In a preliminar experiment, we also tried to apply **Ornstein-Uhlenbeck noise** and **sine noise**, but we found no relevant advantage.

Baselines

Randomly Initialized Networks

- **Single-Output**
- **Two-Outputs**

Low-Level Agents

- **Input:** Numerical/Hybrid
- **Output:** Acceleration/Brake/Steering

Baselines

Randomly Initialized Networks

- **Single-Output**
- **Two-Outputs**

SnakeOil

- **Input:** Numerical
- **Rules:** Fixed, Human-Designed
- **Output:** Low-Level

Low-Level Agents

- **Input:** Numerical/Hybrid
- **Output:** Acceleration/Brake/Steering

Autopia

- **Input:** Numerical
- **Rules:** Fuzzy, Human-Designed
- **Output:** Low-Level

Testing

Metric of Interest

- Distance raced in a fixed time

Testing

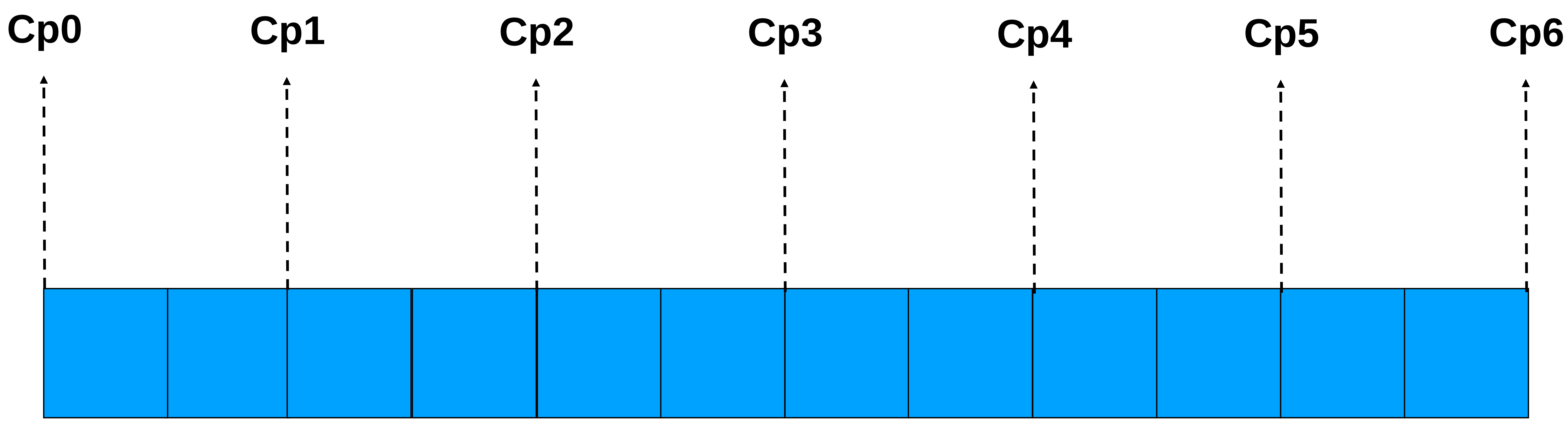
Metric of Interest

- Distance raced in a fixed time

7 Checkpoints per Agent

Trained Agents (LL and HL)

- Uniformly sampled checkpoints
- The best checkpoint is used for testing



$$p(c_i) = mean(d_{c_i, t_i \in T_{train}}) - 0.5 \cdot std(d_{c_i, t_i \in T_{train}})$$

Testing

Metric of Interest

- Distance raced in a fixed time

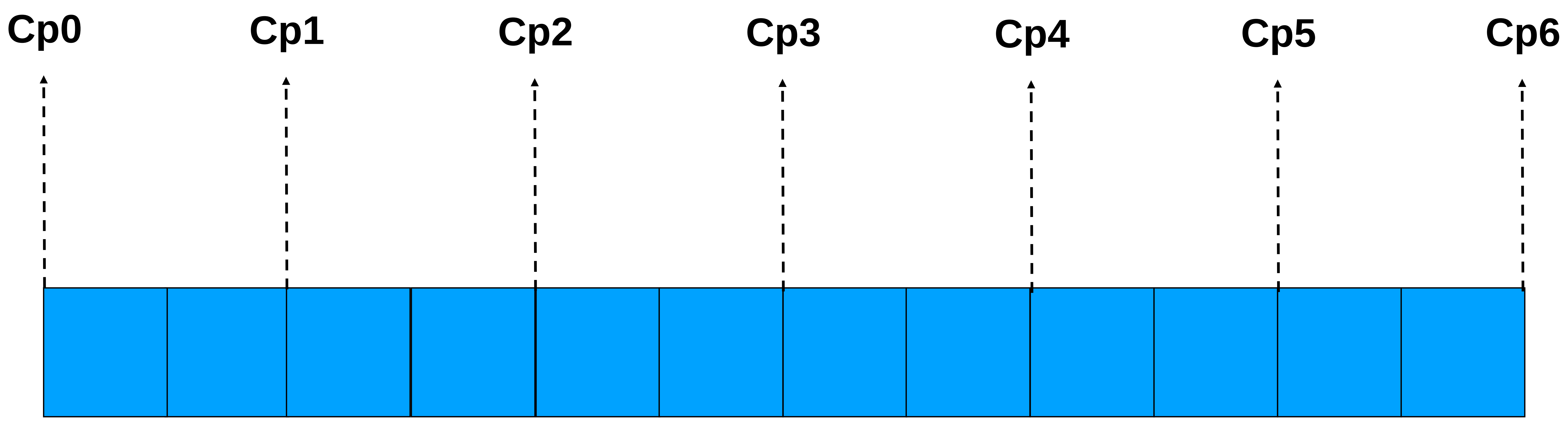
Trained Agents (LL and HL)

- Uniformly sampled checkpoints
- The best checkpoint is used for testing

Episode Termination

- Out of time
- Collision
- Driving backwards

7 Checkpoints per Agent



$$p(c_i) = mean(d_{c_i, t_i \in T_{train}}) - 0.5 \cdot std(d_{c_i, t_i \in T_{train}})$$

Results

Single-Output Agents

Basics

- Improvement over random policy

Low-Level Comparison

- Improvement over LL-N (Mueda is the only exception)
- Completely overcomes LL-H

Bot Comparison

- Improvement over **SnakeOil** (performance and generalization)
- Suboptimal with respect to **Autopia**

Bot	Alsoujlak-Hill	Brondehach	Coldpeak	Citytrack	Emero-City
HL-1R	6999.64	6817.48	6224.37	6463.54	7047.49
HL-2R	6207.31	5703.68	5326.8	6297.34	6171.61
Autopia	**11481.6**	**11593.0**	**11181.7**	**13597.8**	**13172.2**
Snake-Oil	6957.07	739.192	6930.21	6972.4	6987.16
LL-N	1112.28	705.247	1495.89	7714.61	7945.07
LL-H	127.831	192.635	270.652	338.413	215.725
HL-N1	9411.82	9310.71	9999.21	11238.2	9229.33
HL-H1	<u>9413.05</u>	<u>9313.14</u>	<u>10002.4</u>	<u>11238.3</u>	<u>9230.11</u>

Bot	Mueda-City	Petit	Ustka-City
HL-1R	6965.49	8110.44	6280.97
HL-2R	6751.92	5599.0	5910.49
Autopia	**13354.1**	**11513.0**	**12689.5**
Snake-Oil	6998.45	2158.2	6946.44
LL-N	<u>9184.19</u>	55.3047	7770.07
LL-H	218.762	109.677	9.82495
HL-N1	9041.42	9728.49	10402.0
HL-H1	9044.97	<u>9734.09</u>	<u>10403.9</u>

Two-Outputs Agents

Basics

- Improvement over random policy

Low-Level Comparison

- Improvement over LL-N (completely)
- Completely overcomes LL-H

Bot Comparison

- Improvement over **SnakeOil** (performance and generalization)
- Suboptimal with respect to **Autopia**

Single Output

- Slight improvement

Bot	Alsoujlak-Hill	Brondehach	Coldpeak	Citytrack	Emero-City
HL-1R	6999.64	6817.48	6224.37	6463.54	7047.49
HL-2R	6207.31	5703.68	5326.8	6297.34	6171.61
Autopia	**11481.6**	**11593.0**	**11181.7**	**13597.8**	**13172.2**
Snake-Oil	6957.07	739.192	6930.21	6972.4	6987.16
LL-N	1112.28	705.247	1495.89	7714.61	7945.07
LL-H	127.831	192.635	270.652	338.413	215.725
HL-N2	9486.19	9382.04	10185.0	11398.6	9314.12
HL-H2	9411.82	9310.71	9999.21	11238.2	9229.33

Bot	Mueda-City	Petit	Ustka-City
HL-1R	6965.49	8110.44	6280.97
HL-2R	6751.92	5599.0	5910.49
Autopia	**13354.1**	**11513.0**	**12689.5**
Snake-Oil	6998.45	2158.2	6946.44
LL-N	9184.19	55.3047	7770.07
LL-H	218.762	109.677	9.82495
HL-N2	9226.99	9829.32	10466.4
HL-H2	9041.42	9728.49	10402.0

Two-Outputs + Racing Line Agent

Basics

- Improvement over random policy

Low-Level Comparison

- Improvement over LL-N (completely)
- Completely overcomes LL-H

Bot Comparison

- Improvement over **SnakeOil** (performance and generalization)
- Suboptimal with respect to **Autopia**

Two-Outputs Without Racing Line (HL-H2)

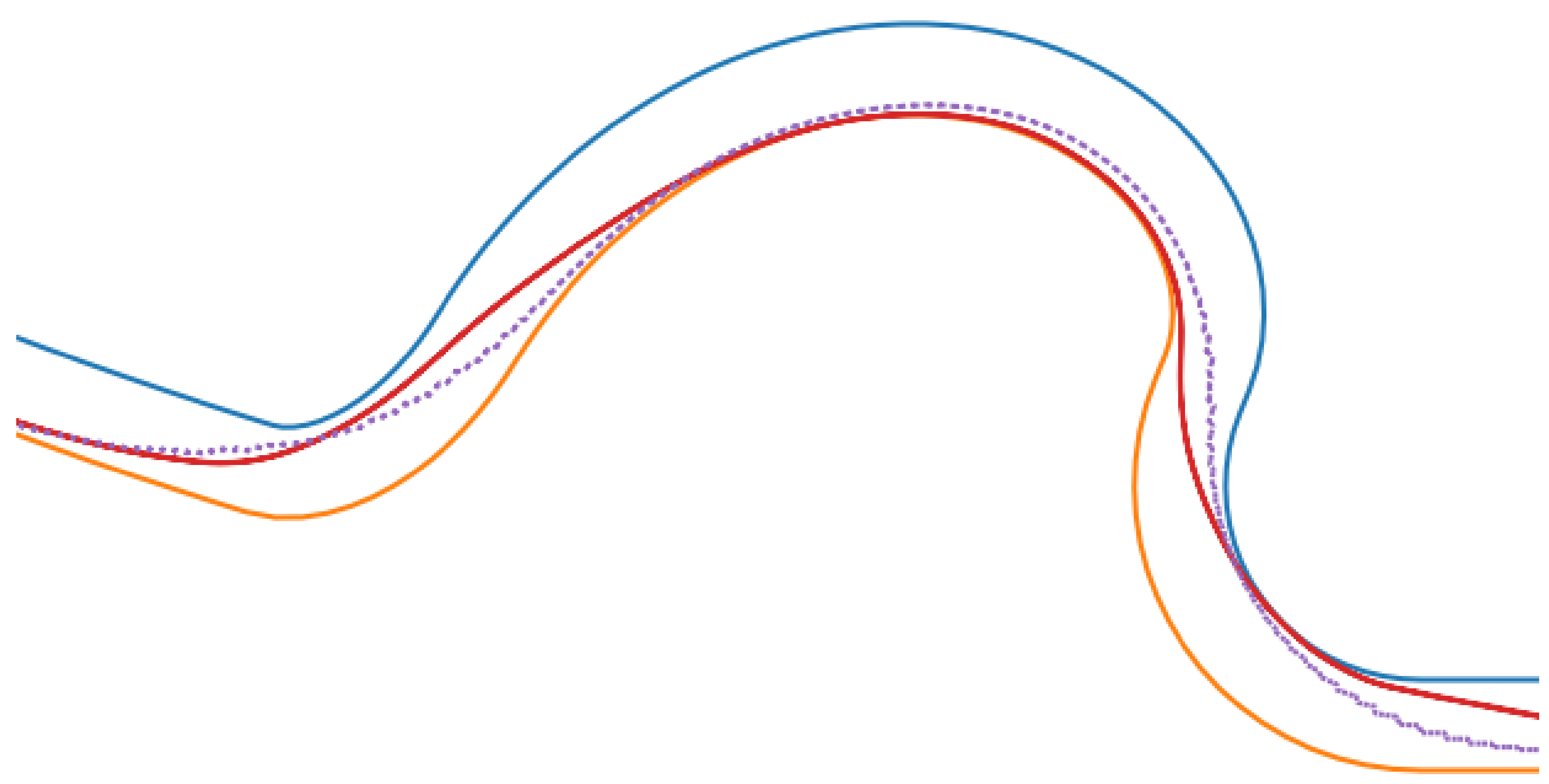
- Slight improvement

Bot	Alsoujlak-Hill	Brondehach	Coldpeak	Citytrack	Emero-City
HL-1R	6999.64	6817.48	6224.37	6463.54	7047.49
HL-2R	6207.31	5703.68	5326.8	6297.34	6171.61
Autopia	**11481.6**	**11593.0**	**11181.7**	**13597.8**	**13172.2**
Snake-Oil	6957.07	739.192	6930.21	6972.4	6987.16
LL-N	1112.28	705.247	1495.89	7714.61	7945.07
LL-H	127.831	192.635	270.652	338.413	215.725
HLR	<u>9486.19</u>	<u>9382.04</u>	<u>10185.0</u>	<u>11398.6</u>	<u>9314.12</u>

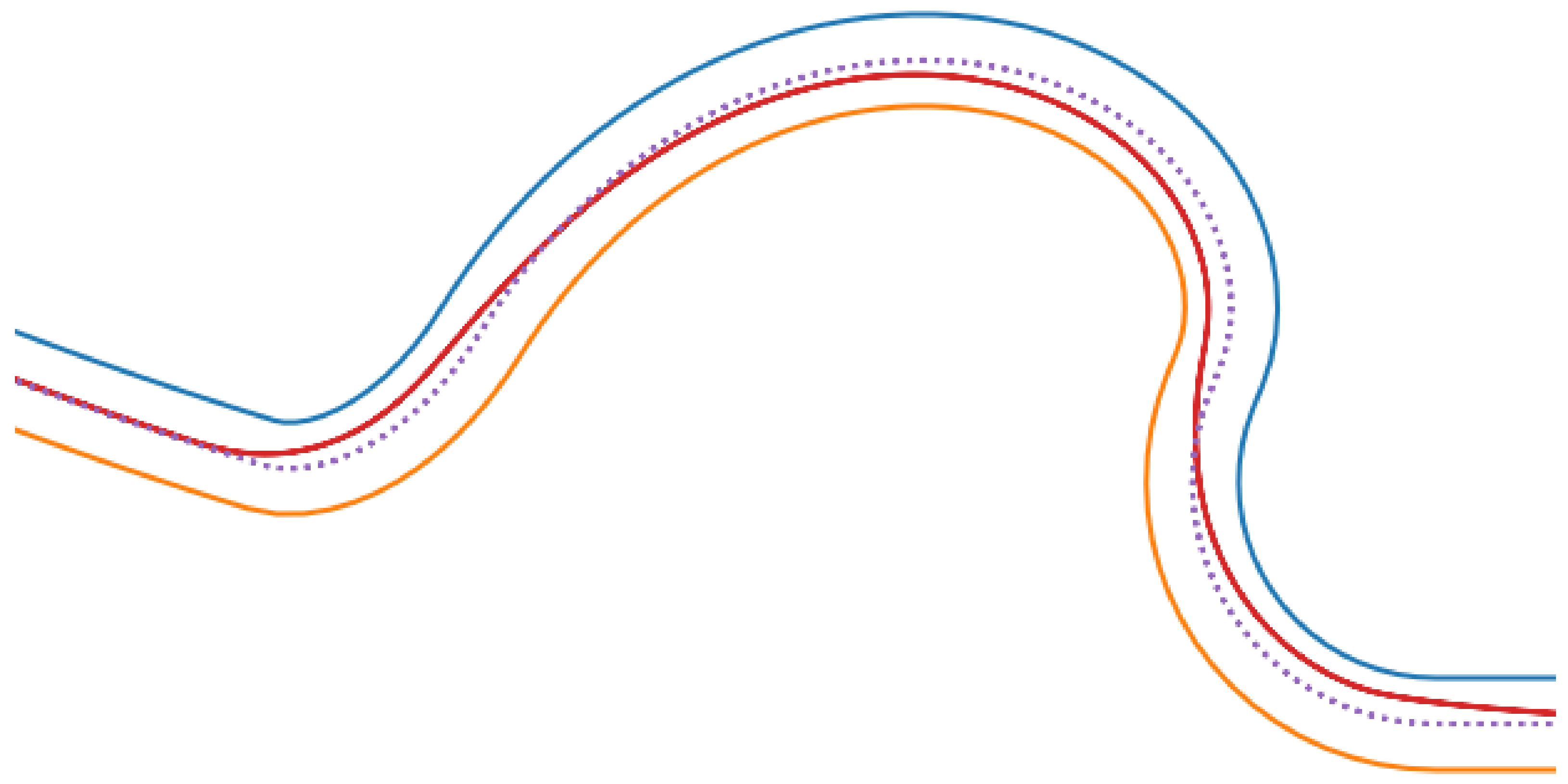
Bot	Mueda-City	Petit	Ustka-City
HL-1R	6965.49	8110.44	6280.97
HL-2R	6751.92	5599.0	5910.49
Autopia	**13354.1**	**11513.0**	**12689.5**
Snake-Oil	6998.45	2158.2	6946.44
LL-N	9184.19	55.3047	7770.07
LL-H	218.762	109.677	9.82495
HLR	<u>9226.99</u>	<u>9829.32</u>	<u>10466.4</u>

Examples of Racing Lines

Following Simplix's Racing Line



Following Learned Racing Line



Future Works

More target points

- A single target point is limiting
- More points allow to build a better racing line approximation

Richer input space

- Enlarge the portion of the track visible to the agent
- This allows for a better planning

Exploration of algorithms

- Perform accurate hyperparameter tuning
- Explore other algorithms (TRPO, PPO, …)

Exploration of reward functions

- Consider embedding racing line information in the reward function
- Learning a general behaviour from specific racing lines

Emilio Capo
emilio.capo@mail.polimi.it
Computer Science and Engineering (CSE)

POLITECNICO
MILANO 1863

HONOURS PROGRAMME HP-SR
in Information Technology