ML for CPS/IoT: An Industry Perspective

Guest Lecture for ECE209AS

Bharathan Balaji

Research Scientist @ Amazon

AWS DeepRacer

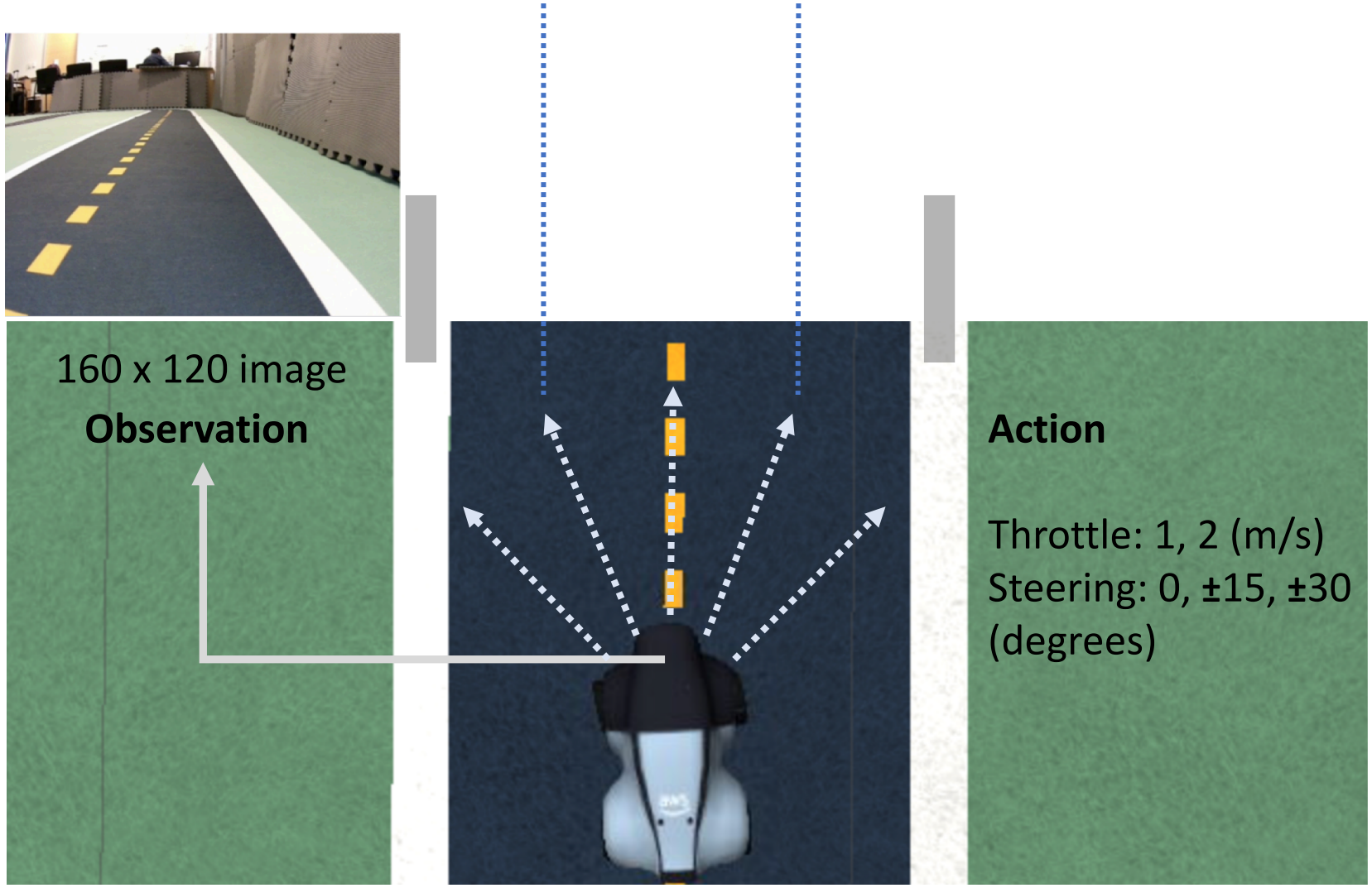
https://aws.amazon.com/deepracer/

Objective: Autonomously Race on a Track

Much simpler than real self-driving cars

- No people
- Traffic rules are simple: follow the track

Imitation Learning



Bojarski, Mariusz, et al. "End to end learning for self-driving cars." *arXiv preprint arXiv:1604.07316* (2016).

Labels Labels

- Need a lot of labels to cover all the scenarios in the real world
- Typical form of ML used in industry today
  - Autonomous driving – obstacle detection, lane keeping, etc.
  - Predictive maintenance
  - Activity recognition
  - Defect recognition
- Labeling services exist to scale human work. E.g. SageMaker GroundTruth

Methods to reduce labeling

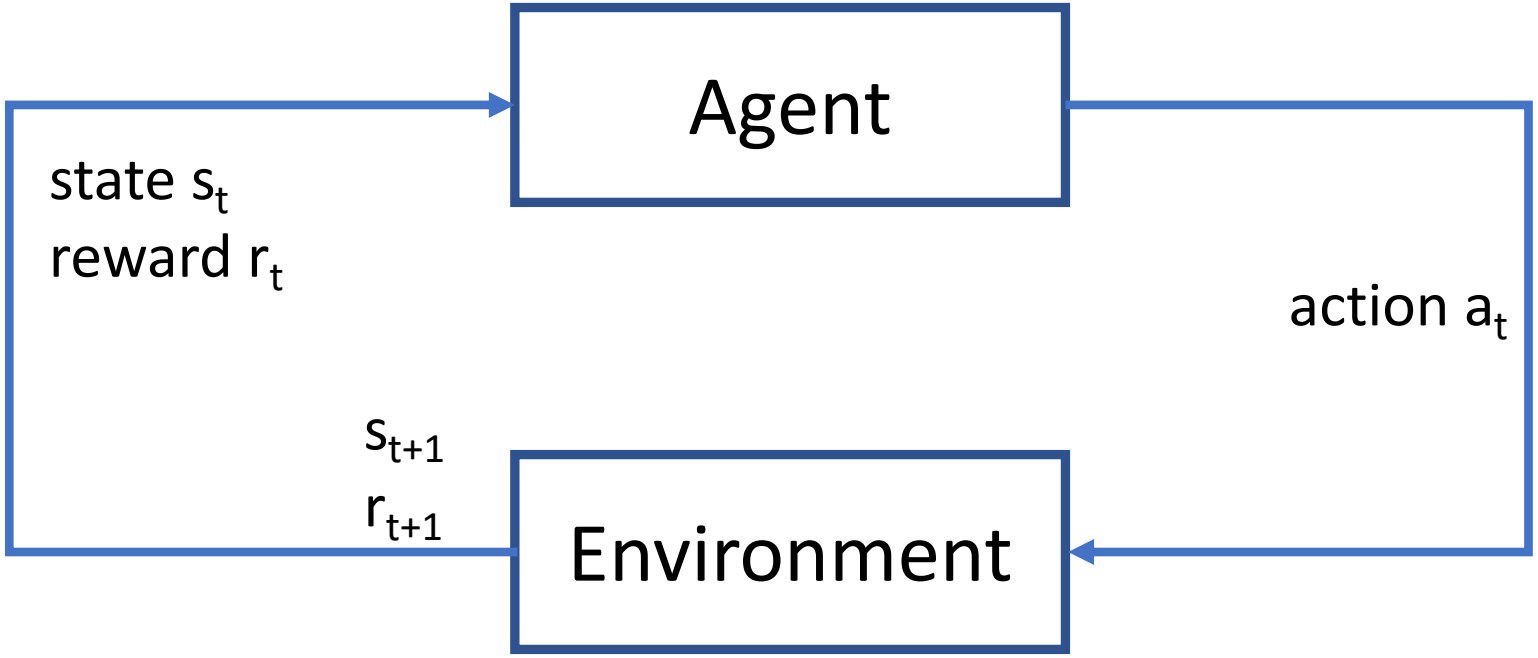
- Active Learning
  - Only ask to label those data points which the model is not confident about
- Transfer Learning
  - Use a model trained for another task as warm start
- Domain adaptation
  - Train the model to create common features for source and target domain
- Self-supervised learning
  - Create artificial labels and “pre-train” a model. E.g. cut out a portion of image and ask the model to predict the cutout portion
- Meta learning
  - Train a “meta” model learns across multiple tasks, and trains with lesser labels on new tasks

The problem with imitation learning

- Our predictions impact the data we collect
- Say the car incorrectly predicts left instead of right in one scenario, it encounters sequence of inputs not similar to the training dataset
- Counterfactual problem: what would have happened if I had taken left instead of right?

Reinforcement Learning

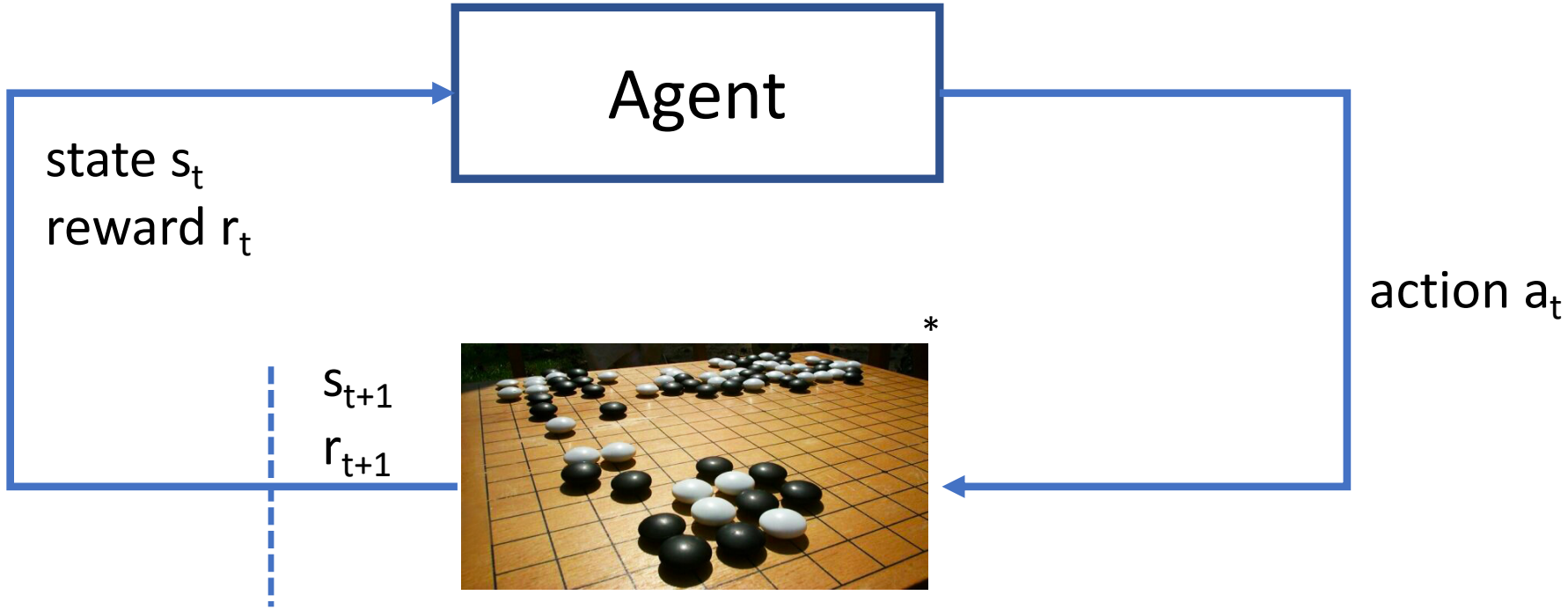
- Terminology
  - State, *s*
  - Action, *a*
  - Reward, *r*
  - Horizon, H



- Policy: π given input state *s* gives action *a.* Parametrized by θ
- Objective:

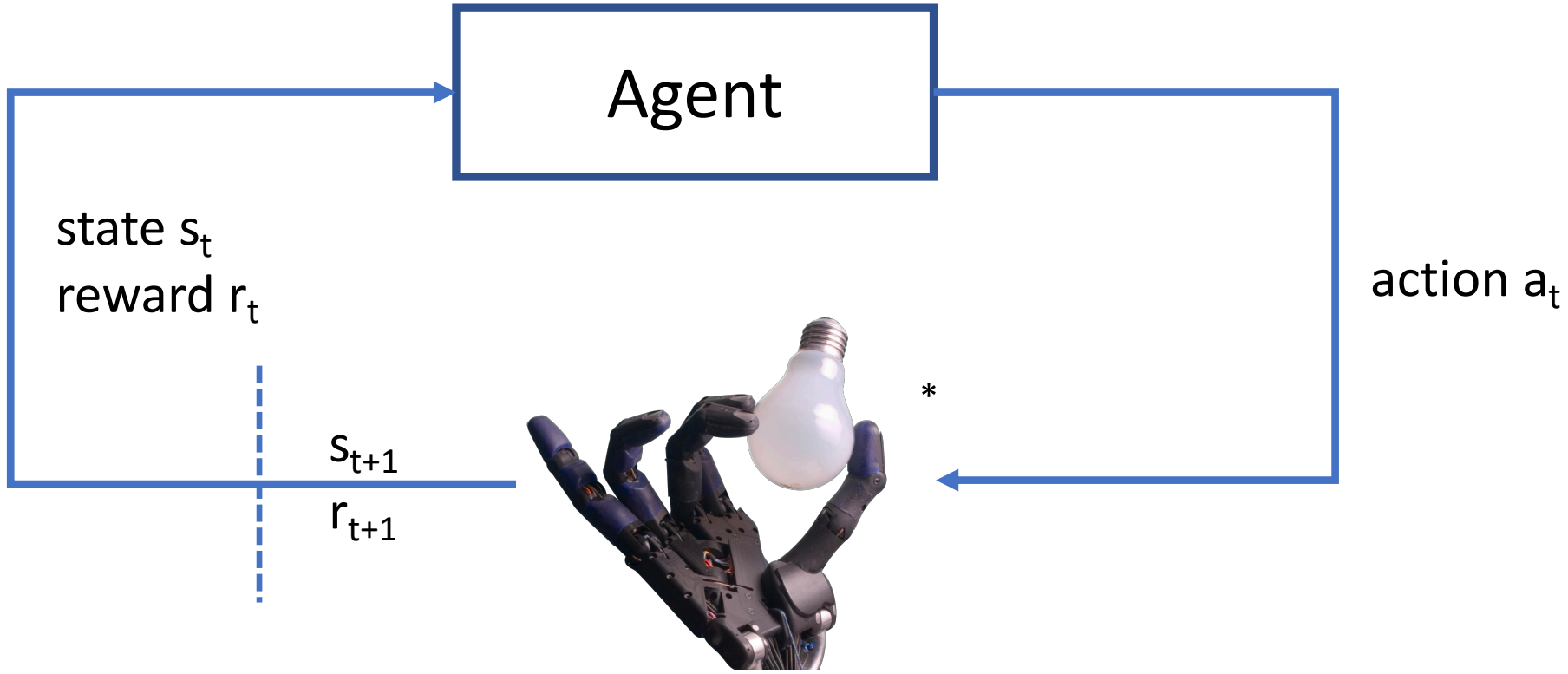
$$J(\theta) = \max_{\theta} \mathbb{E}[\sum_{t=0}^{H} r(s_t, a_t)|\pi_\theta]$$

Games – AlphaGo, Atari, Dota 2



- Mastering the game of Go without human knowledge – Silver et al., 2017
- Mastering the game of Go with deep neural networks and tree search – Silver et al., 2016
- * Image from Wikipedia

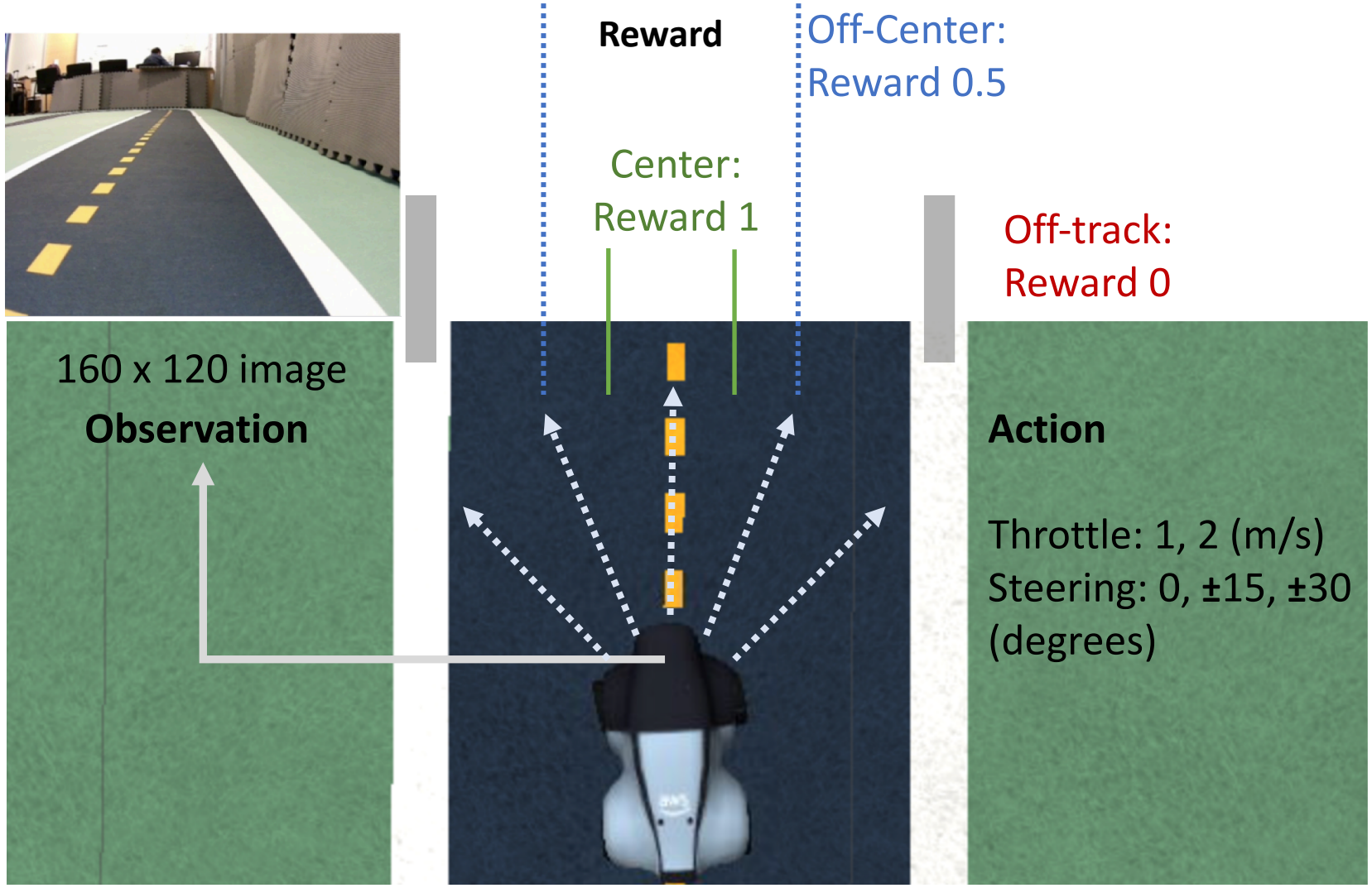
Robotics – Grasping, Navigation, Locomotion



- Learning Dexterous In-Hand Manipulation – OpenAI, 2018
- Learning agile and dynamic motor skills for legged robots – Hwangbo et al., 2019

* Image from Wikipedia

Autonomous Racing with RL: Problem Formulation



RL Algorithm - Policy Gradients

- Objective

$$J(\theta) = \max_{\theta} \mathbb{E}[\sum_{t=0}^{H} r(s_t, a_t)|\pi_\theta]$$

- Initialize a policy with random θ and collect experiences

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\left(\sum_{t=0}^{H} \nabla_\theta \log \pi_\theta(a_t|s_t)\right)\left(\sum_{t=0}^{H} r(s_t, a_t)|\pi_\theta\right)\right]$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

- Use the updated policy to collect more data and iterate

Intuitive Version of Policy Gradients

"Try a bunch of stuff and see what happens. Do more of the stuff that worked in the future."

-RL

Slide inspired by Andrej Karpathy

Q Functions

- Q-function, also called action value function

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + r_{t+2} + r_{t+3} + \ldots | s, a]$$

- With discounting of future rewards. Discount factor, $\gamma \in [0,1]$

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s, a]$$

- Bellman equation form $Q^{\pi}(s,a) = \mathbb{E}_{s',a'}[r_{t+1} + \gamma Q^{\pi}(s',a'))|s,a]$

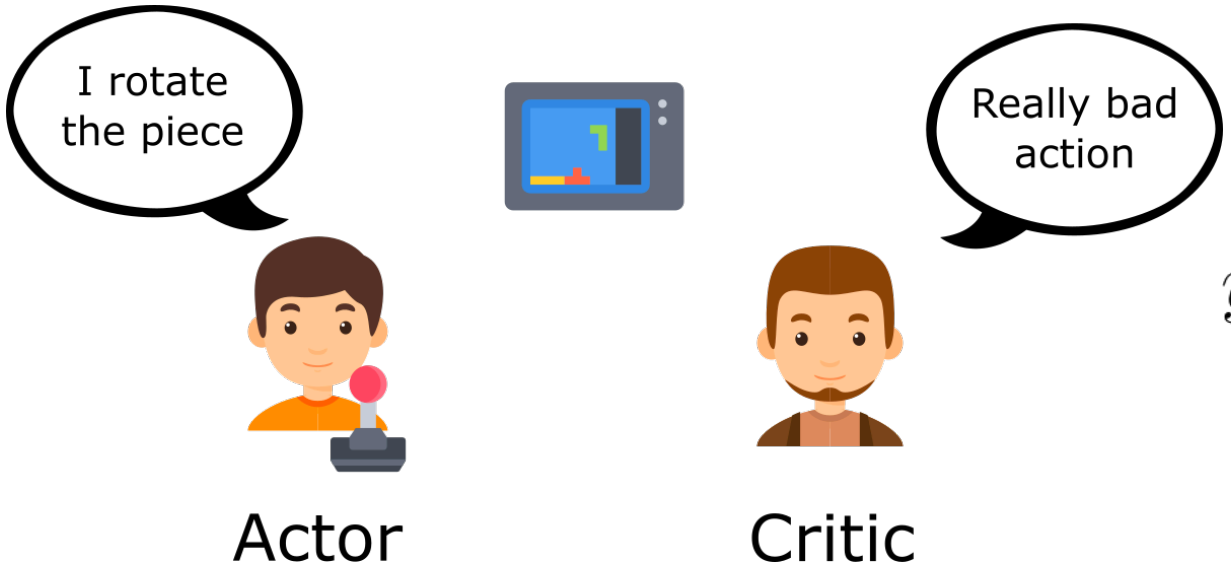
- Value function $V^{\pi}(s) \coloneqq \mathbb{E}_{a_i \sim \pi(\cdot|s_i)}\left[\sum_{i \geq t} \gamma^{i-t} r_i \middle| s_t = s\right]$

RL Algorithm – Actor Critic

- Policy Gradients Update $\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(a|s) \sum_t r(s_t, a_t)$ High variance

- Actor Critic Update $\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(a|s) * Q(s,a)$

Actor Critic



$$y_t \approx r(s_t, a_t) + \gamma \max_{a_{t+1}} Q_\phi(s_{t+1}, a_{t+1})$$

$$L(\phi) = \frac{1}{2}\left(Q_\phi(s,a) - y(s,a)\right)^2$$

Image source: Thomas Simonini
https://www.freecodecamp.org/news/an-intro-to-advantage-actor-critic-methods-lets-play-sonic-the-hedgehog-86d6240171d/

RL Algorithm – Proximal Policy Optimization

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(a|s) * Q(s,a)$$

- Advantage Actor Critic

$$\Delta\theta = \alpha \nabla_\theta \log \pi_\theta(a|s) * \underbrace{(Q(s,a) - V(s))}_{\text{Advantage A(s,a)}}$$

- Importance Sampling

$$\Delta\theta = \alpha \nabla_\theta \underbrace{\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}}_{\text{Importance correction}} * A(s,a)$$

- Trust Region

$$\Delta\theta = \alpha \nabla_\theta \frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} * A(s,a) - \beta \underbrace{\mathbb{E}(KL[\pi_\theta(.|s_t), \pi_{\theta_{old}}(.|s_t)])}_{\text{Trust Region}}$$

RL Algorithm -- Proximal Policy Optimization

- The *Temporal difference (TD) residual*:

$$\delta_t = r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

- The *generalized advantage estimator* (*GAE*):

$$\hat{A}_t = \hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} := \sum_{l=0}^{k} (\gamma\lambda)^l \delta_{t+l}$$

- *Objective* for clipped proximal policy optimization (Clipped PPO):

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \cdot \hat{A}_t, \mathrm{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}_t\right)\right]$$

RL Algorithm -- Proximal Policy Optimization

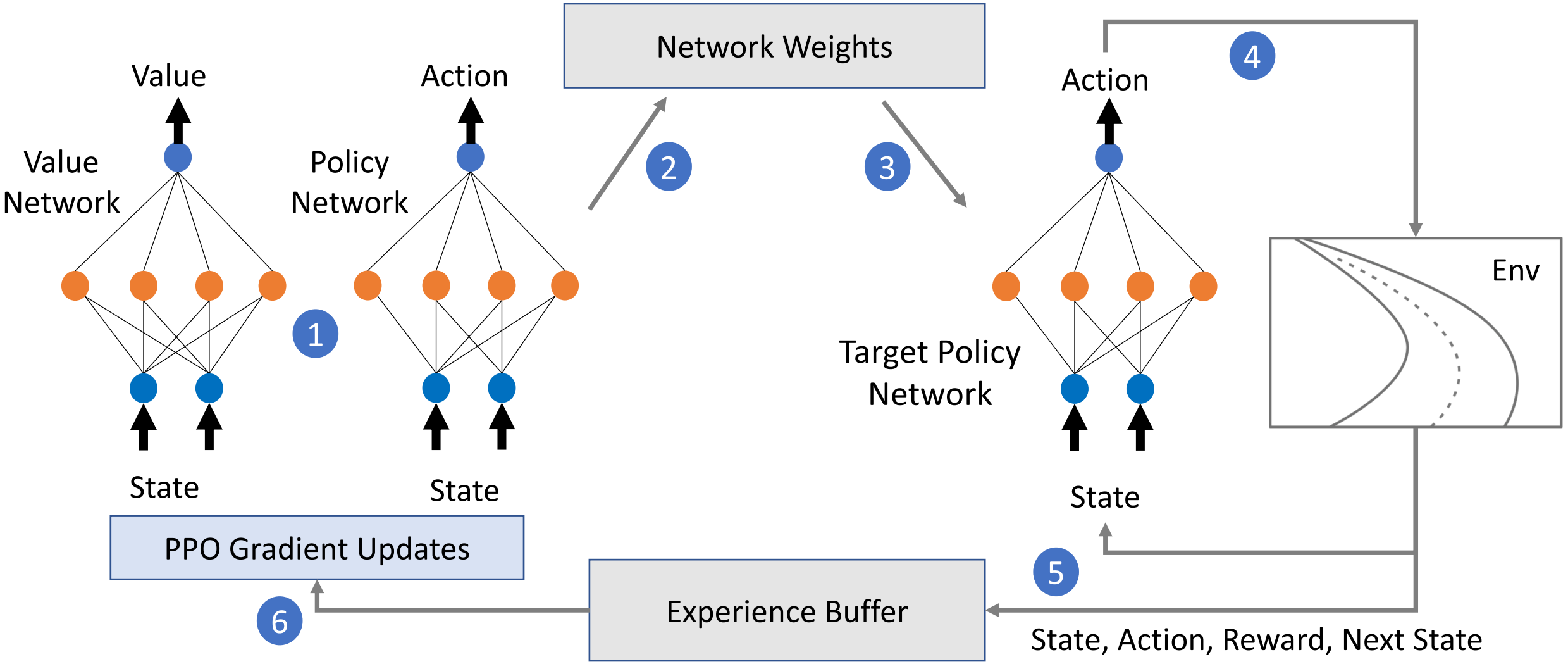
1. Collect $\{s_i, a_i, s_{i+1}, r_i\}$ following $\pi_{\theta_{\text{old}}}(a|s)$
2. Fit $\hat{V}_\phi(s_i)$ to sampled reward sums
3. Compute advantage estimates $\hat{A}_1, \hat{A}_2, \ldots, \hat{A}_T$ using GAE
4. Update policy network: $\theta \leftarrow \theta_{\text{old}} + \alpha \nabla_\theta L^{\text{CLIP}}(\theta)$

Objective for proximal policy optimization (PPO)*:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} \cdot \hat{A}_t, \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}_t\right)\right]$$

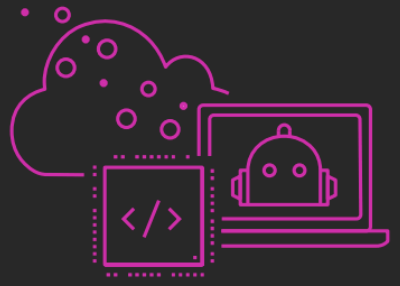
*Schulman, John, et al. "Proximal policy optimization algorithms." *arXiv preprint arXiv:1707.06347* (2017).

Training Procedure



AWS RoboMaker capabilities

Cloud Extensions for ROS



Development Environment

Simulation

Fleet Management

Amazon SageMaker

Build, train, tune, and host your own models

Compliance and audit capabilities

Pay as you go

End-to-end-encryption

BUILD

Pre-built notebooks for common problems

Built-in, high performance algorithms

TRAIN & TUNE

One-click training

Hyperparameter optimization

DEPLOY

One-click deployment

Fully managed hosting with auto-scaling

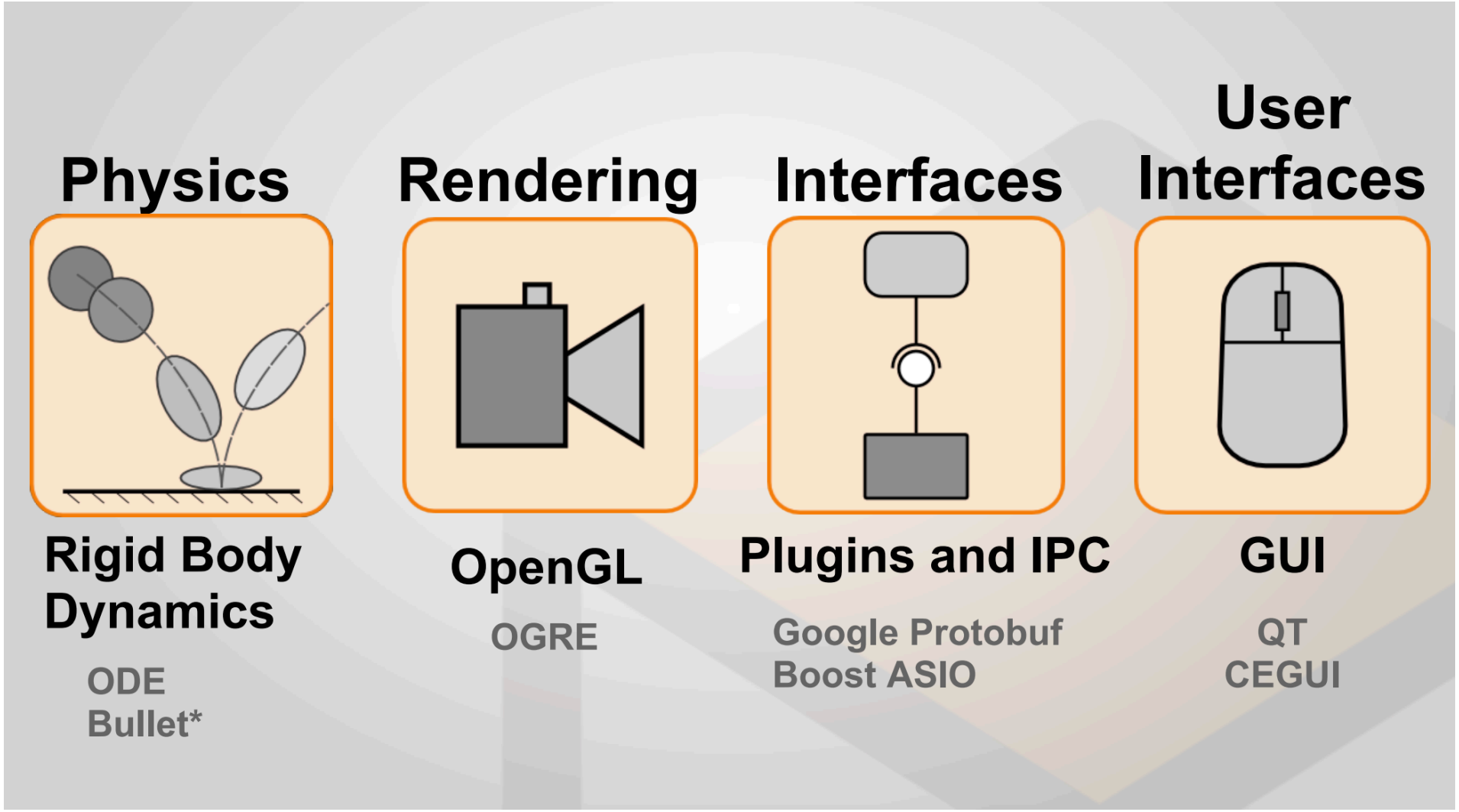
End-to-end VPC support

Metadata and experiment management capabilities

aws

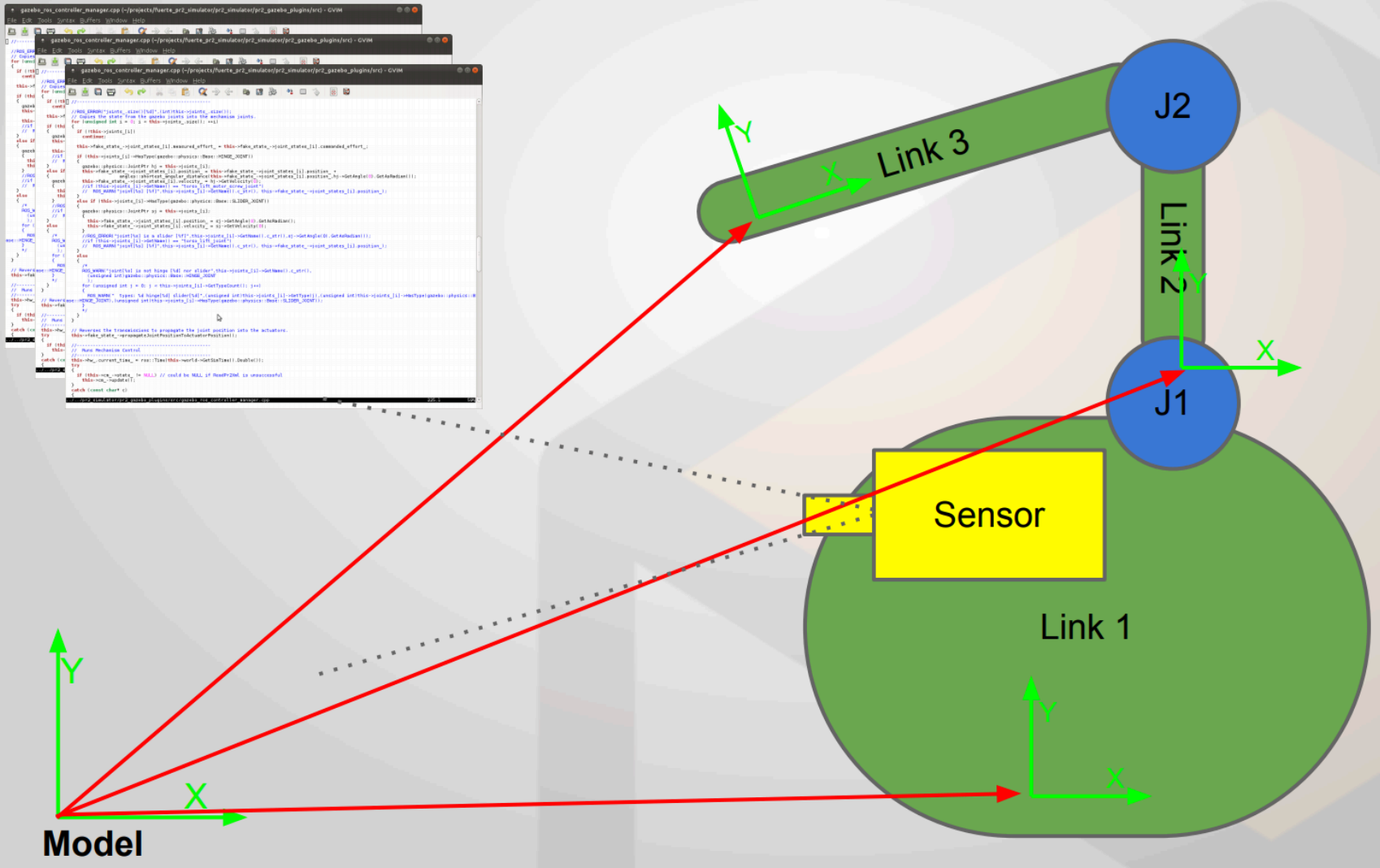
Training in Simulation using RoboMaker

Gazebo – Simulation Orchestrator



What is a Robot (Model)?

A collection of links, joints, sensors, actuators and plugins.

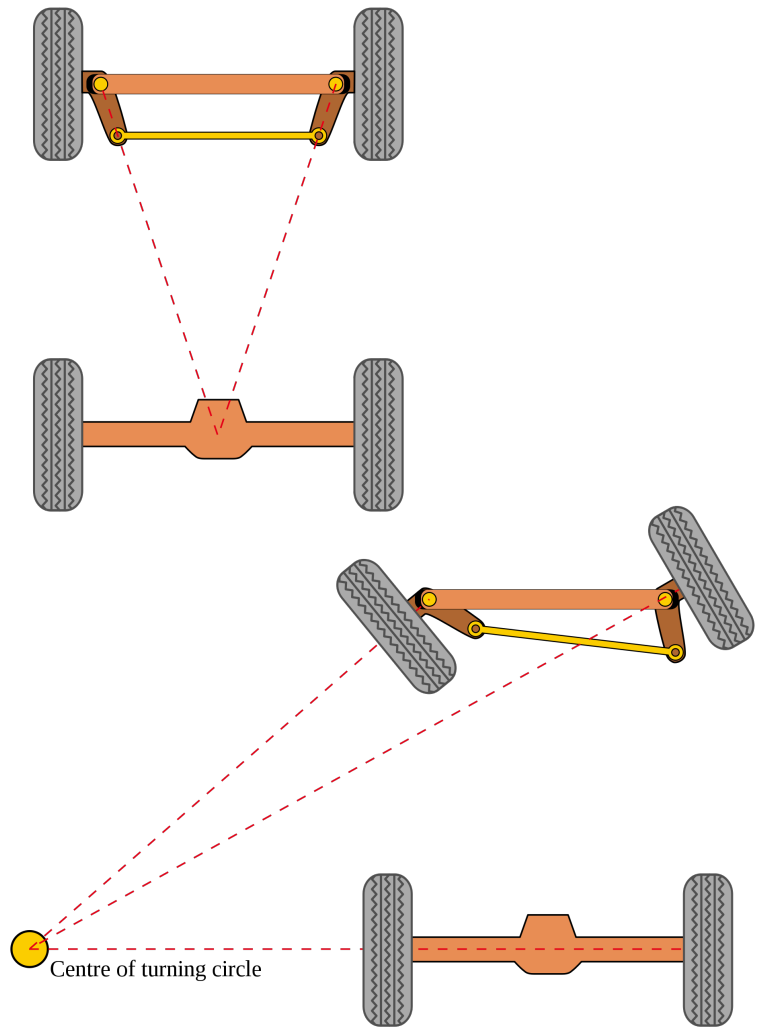


DeepRacer in Simulator

mit-racecar / racecar-simulator

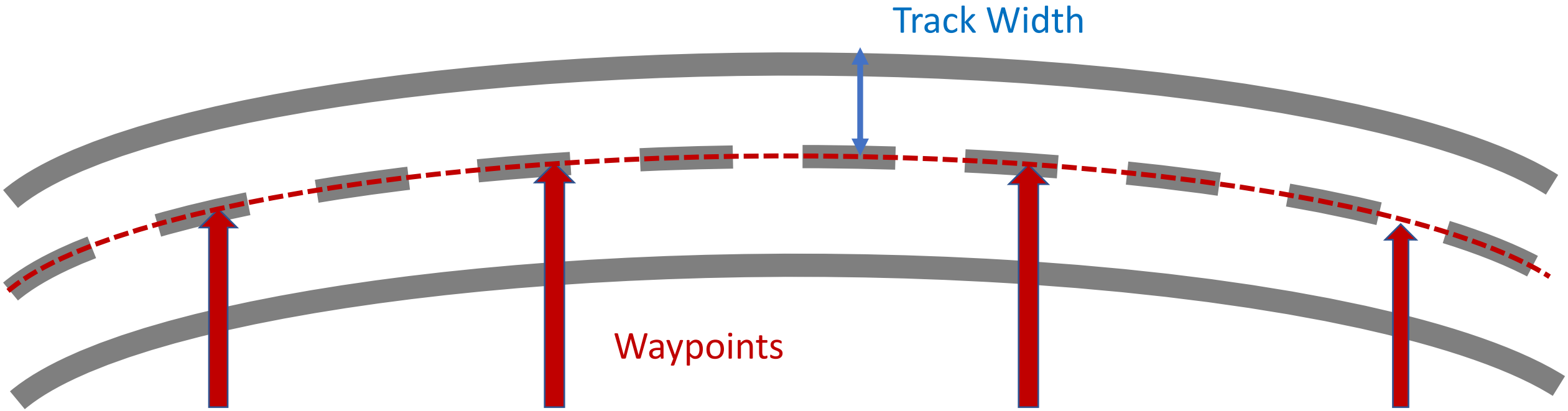
Code | Issues 6 | Pull requests 2

- Unified Robot Description Format
- Ackerman steering model
- Match DeepRacer specs
  - Wheel size
  - Camera height, angle
  - Camera field of view
  - Friction



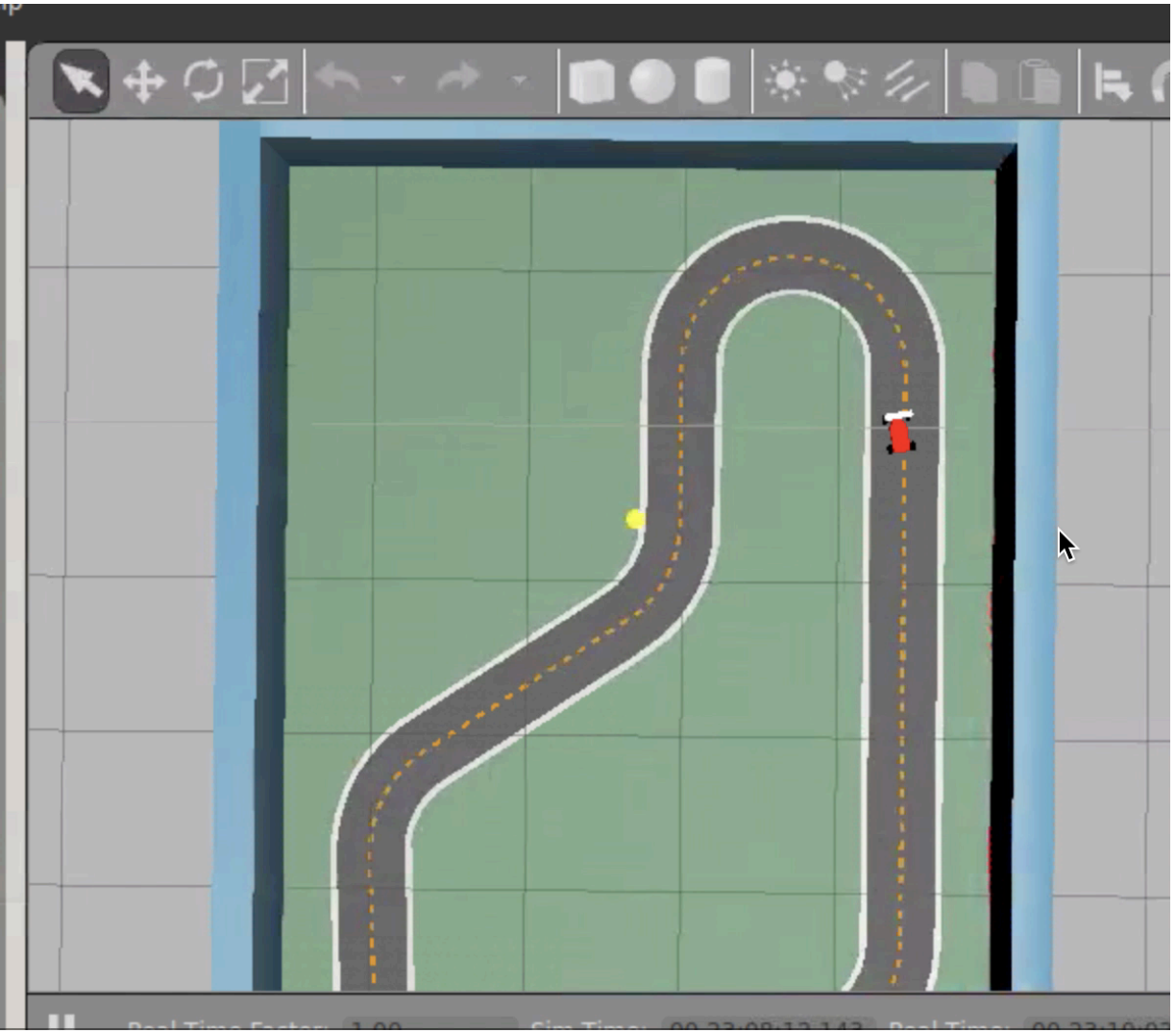
Ackerman Steering

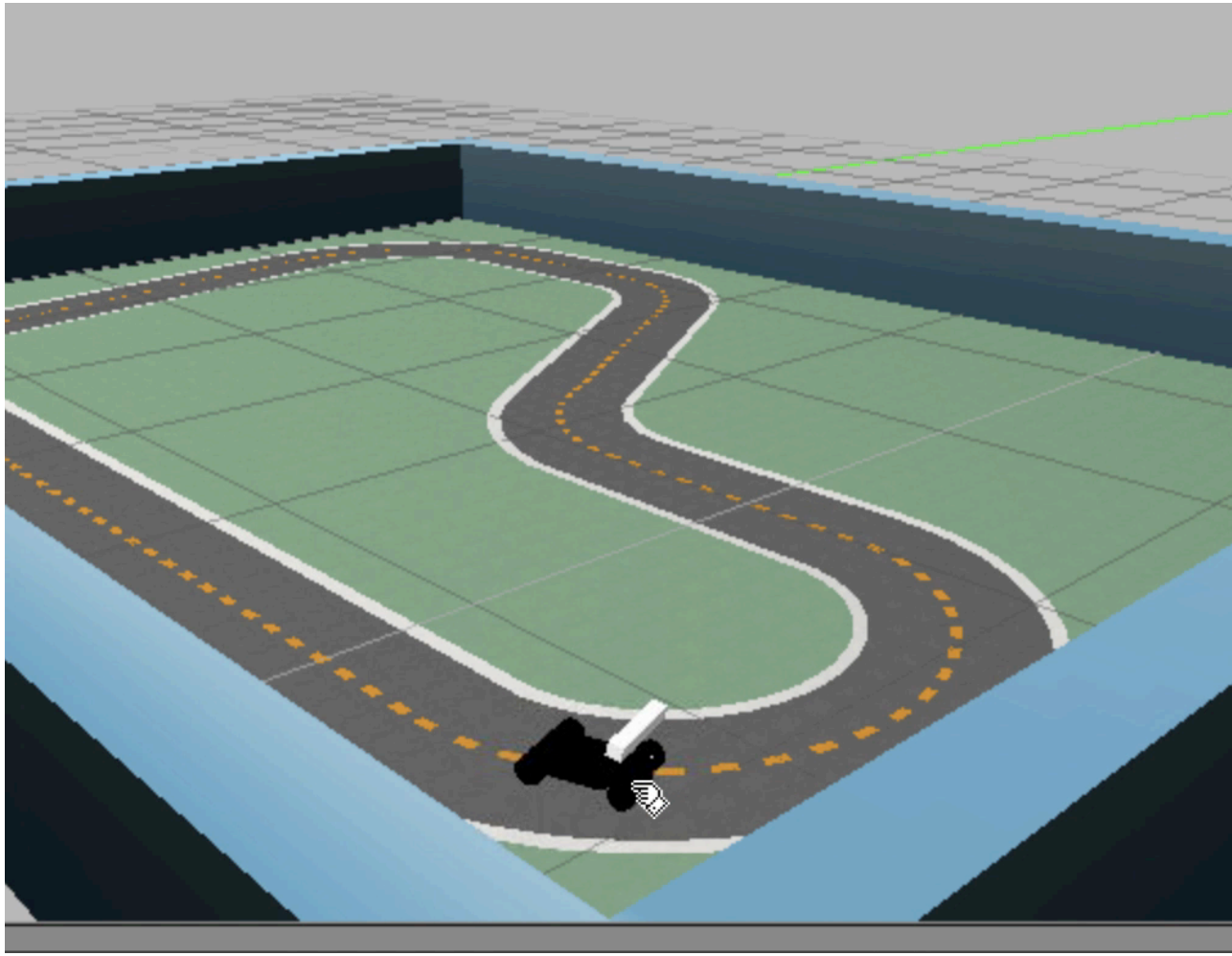
Calculating Rewards in Simulator



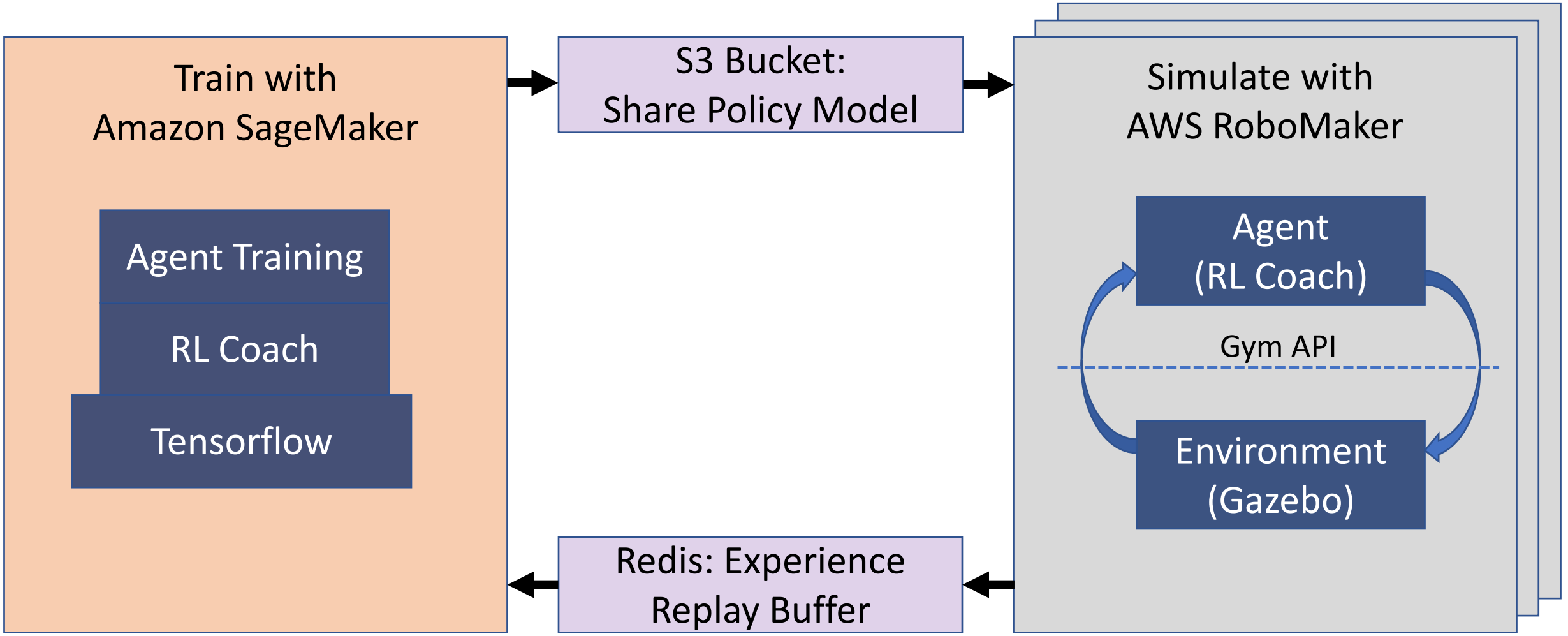
- Waypoints mark progress along the track
- Off-track if car position is away from center by track width

Why Physics Matters



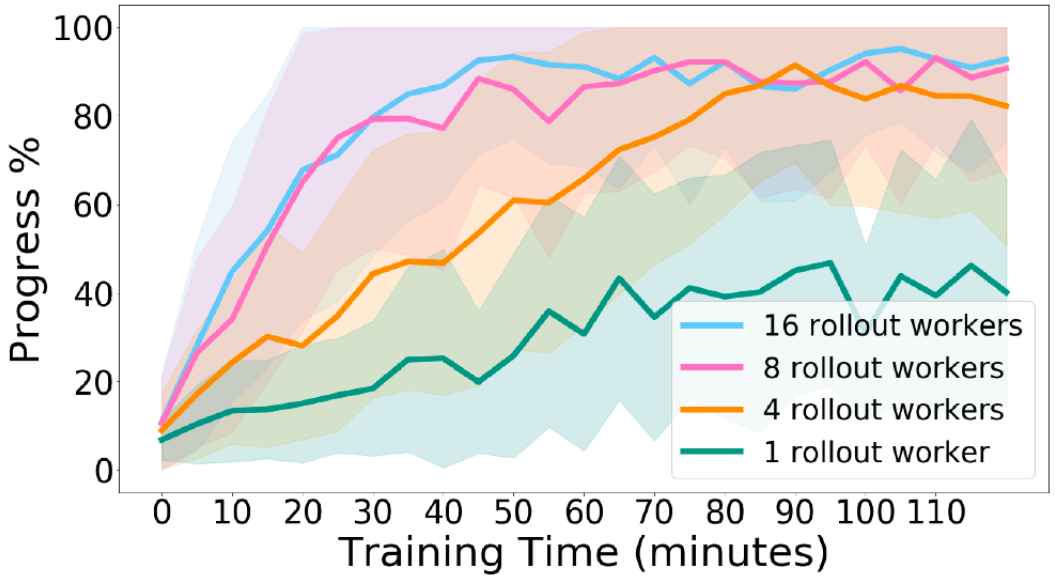


Decoupled Distributed Training

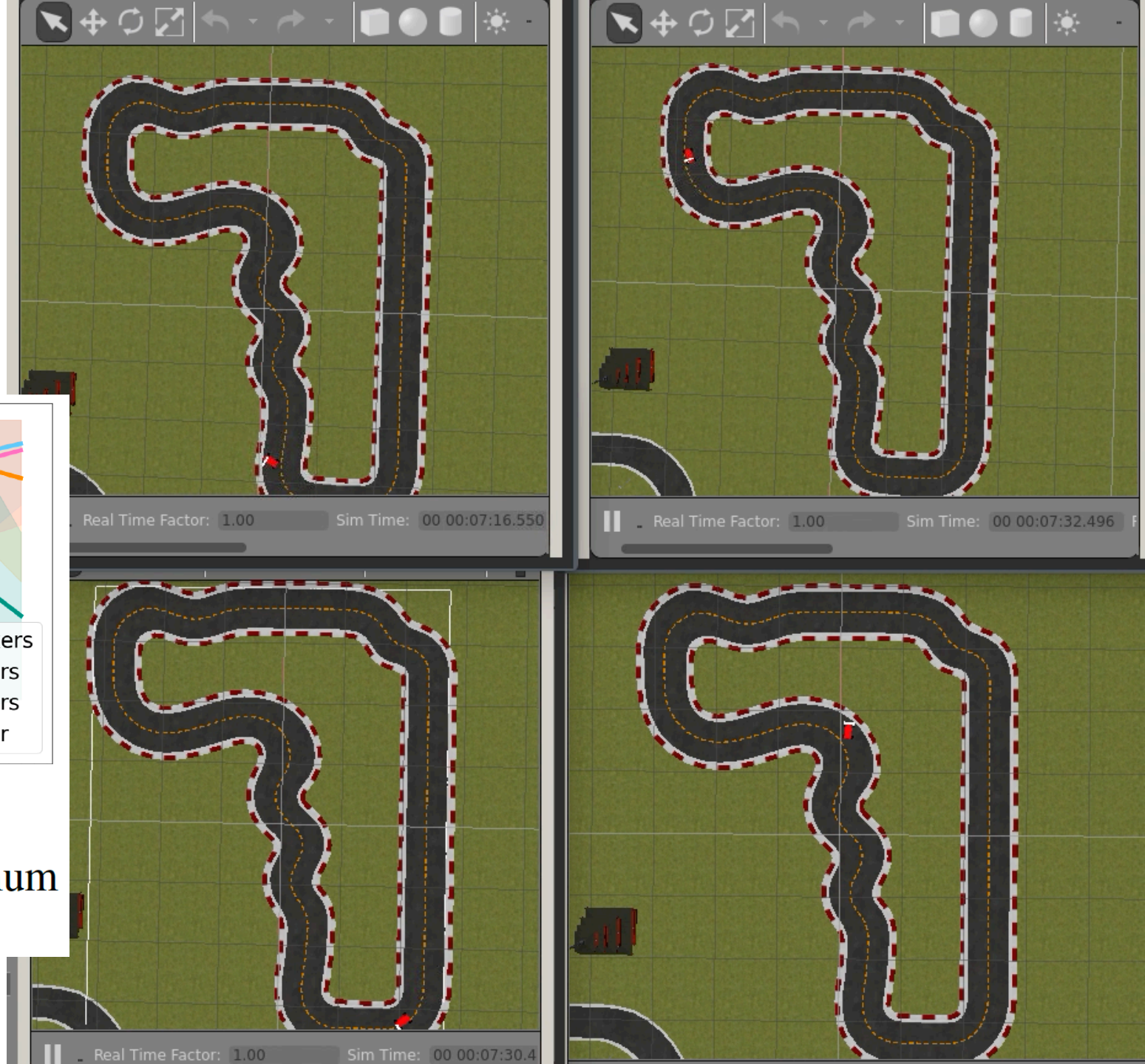


Distributed Rollouts

Successful Sim2Real with just 5 minutes of training



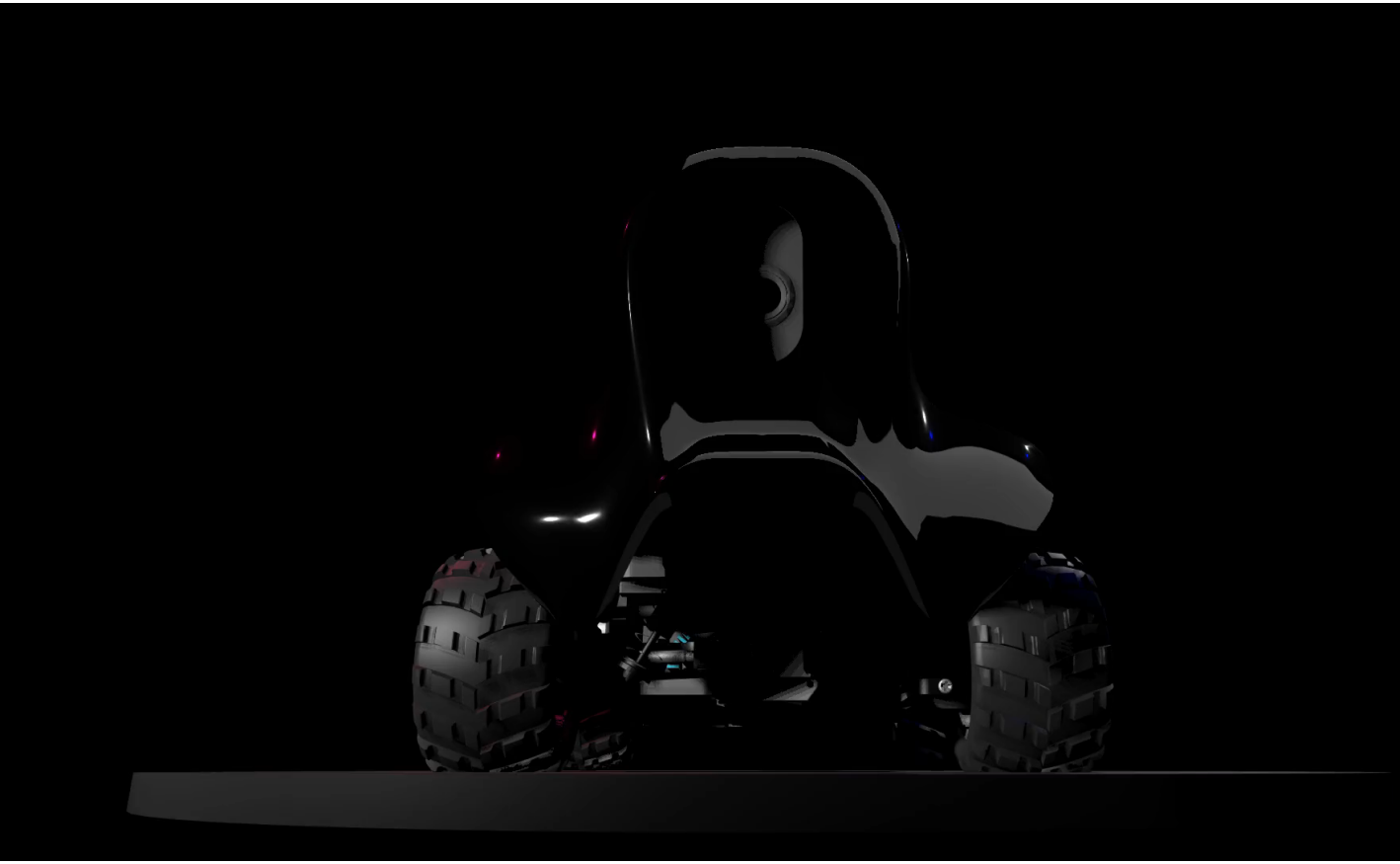
(c) Training with Track B and maximum throttle of 1.67 m/s



What if we have no simulator?

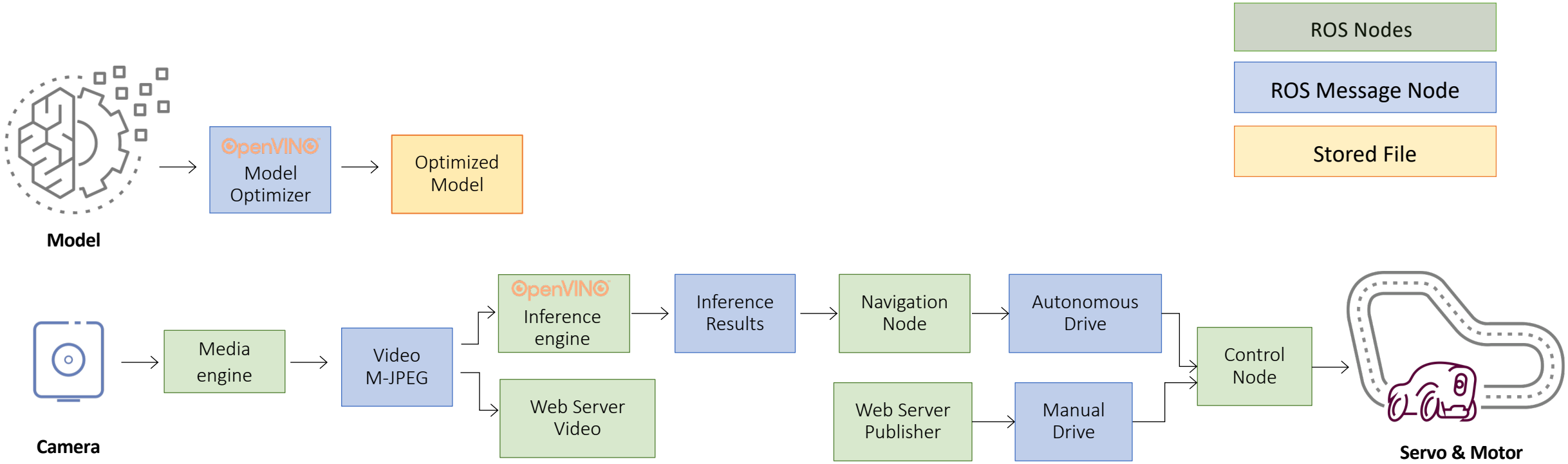
- Create an ML model that acts like a simulator: Model based RL
- Learn from historical data: Offline RL
- Learn directly from real world interactions (but safety is an issue)
  - https://www.youtube.com/watch?v=eRwTbRtnT1I

AWS DeepRacer Car Specifications

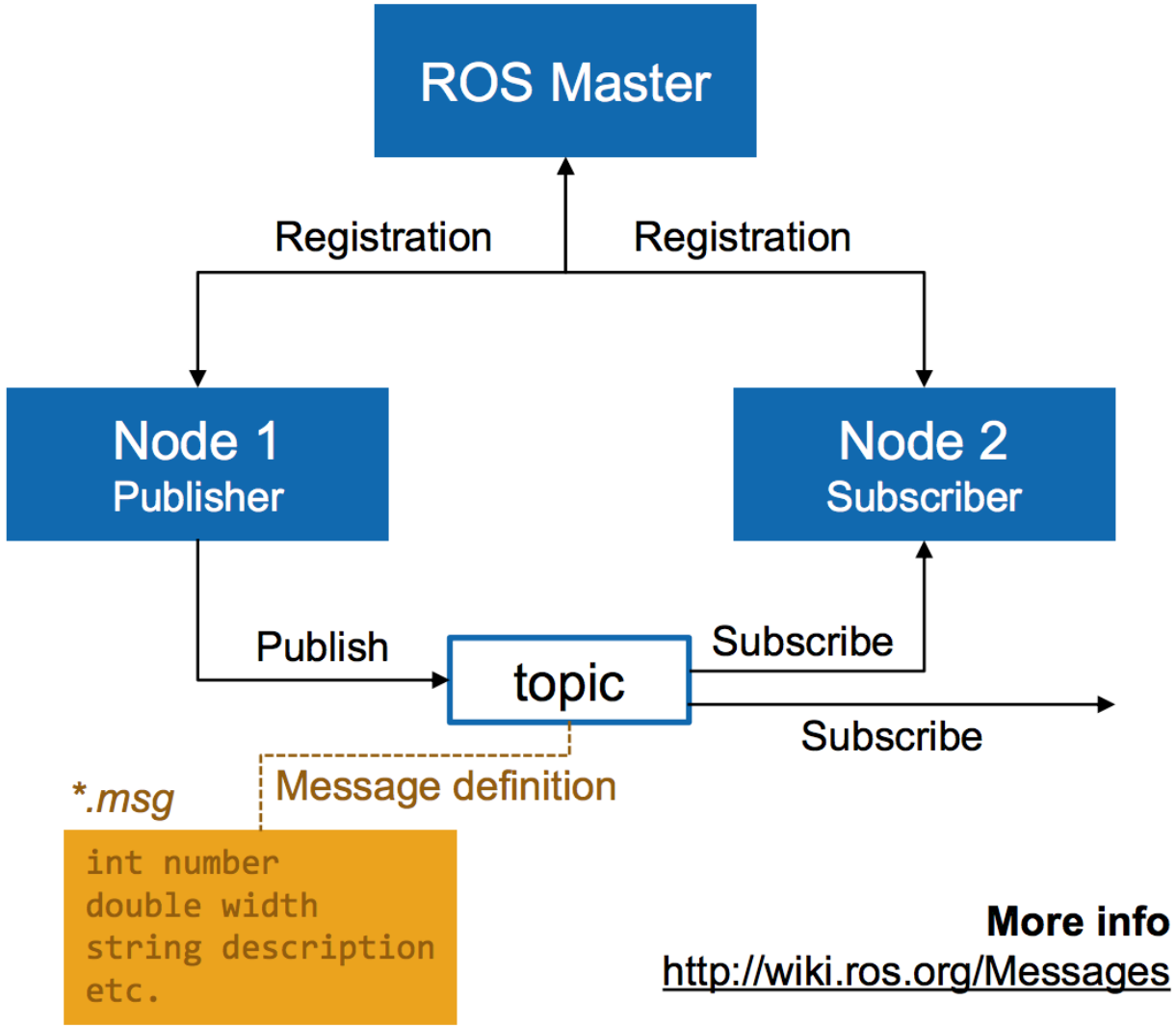


CAR	18th scale 4WD with monster truck chassis
CPU	Intel Atom™ Processor
MEMORY	4GB RAM
STORAGE	32GB (expandable)
WI-FI	802.11ac
CAMERA	4 MP camera with MJPEG
DRIVE BATTERY	7.4V/1100mAh lithium polymer
COMPUTE BATTERY	13600mAh USB-C PD
SENSORS	Integrated accelerometer and gyroscope
PORTS	4x USB-A, 1x USB-C, 1x Micro-USB, 1x HDMI
SOFTWARE	Ubuntu OS 16.04.3 LTS, Intel® OpenVINO™ toolkit, ROS Kinetic

AWS DeepRacer Software Architecture

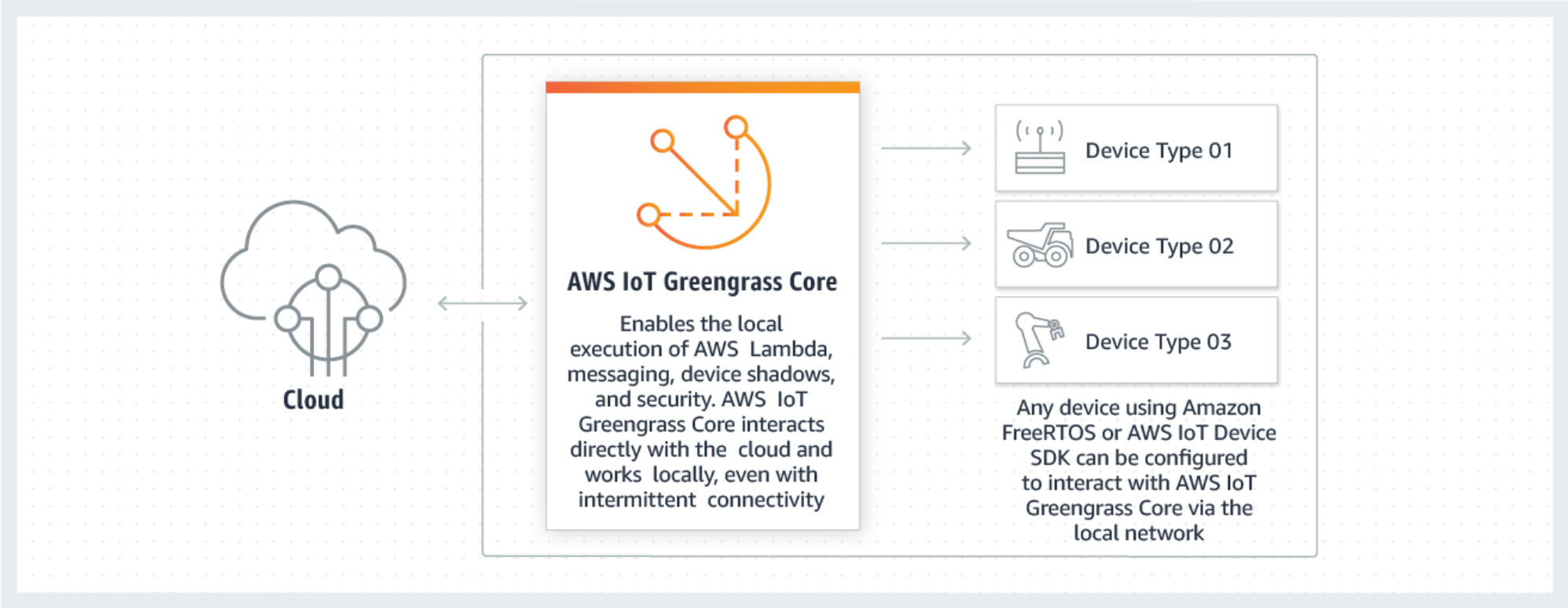


ROS in a Nutshell



Components of a typical Edge ML deployment

- GreenGrass



Lambda runtime, Shadows implementation, Message manager, Group management, Discovery service, Over-the-air update agent, Stream manager, Local resource access, Local machine learning inference, Local secrets manager

Components of a typical Edge ML deployment

- GreenGrass
- Model compilation – SageMaker Neo, Apache TVM, Tensorflow XLA

Components of a typical Edge ML deployment

- GreenGrass
- Model compilation
  - Kind of like going from Python to C++
- Neural network accelerator – NVIDIA Xavier, Google TPU, AWS Inferentia
  - Small GPU like hardware specialized for dot product, convolutions, etc.
- Model compression - network distillation, architecture pruning
  - Quantization (e.g. 16 bits)
  - Latency (e.g. 10 ms)
  - Memory (e.g. 100 MB)

Simulation-to-Real Domain

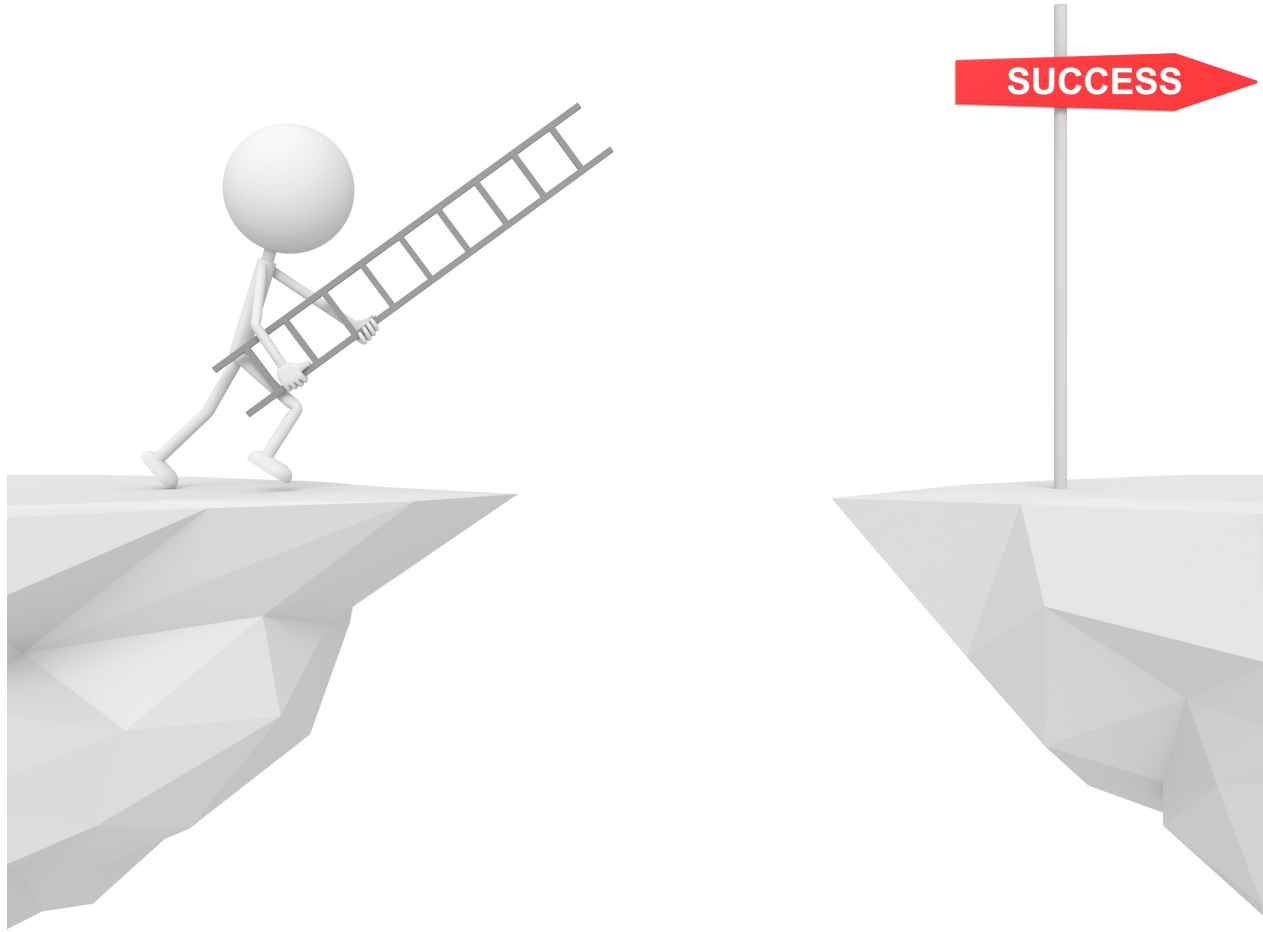
SIM-to-REAL CHALLENGE

Train model using simulated images, but race car using real world images

STRATEGIES

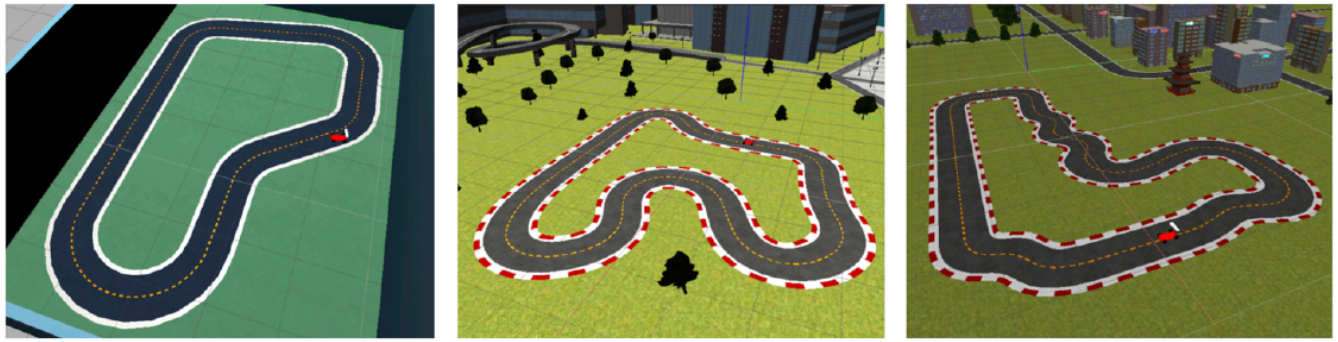
Environment Control

Domain Randomization



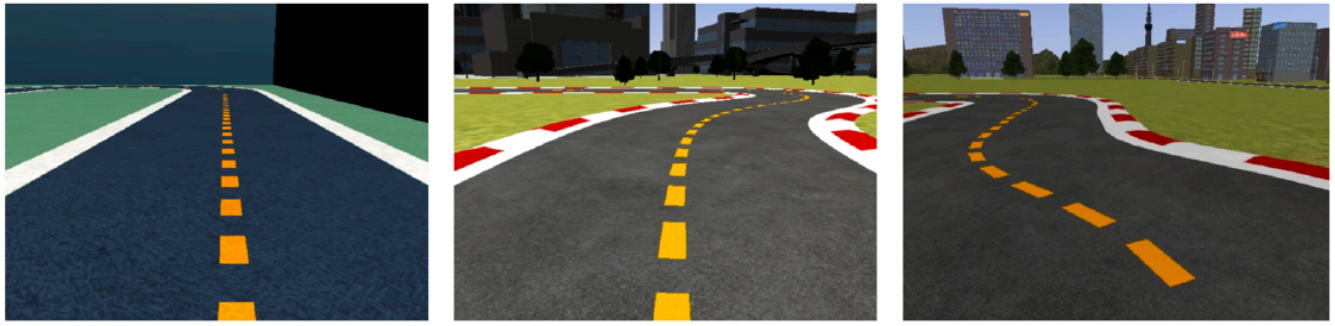
Train and evaluate on a variety of tracks

- Different backgrounds
- Colors
- Textures
- Printed and duct tape tracks



Track A Track B Track C

(a) Simulation tracks



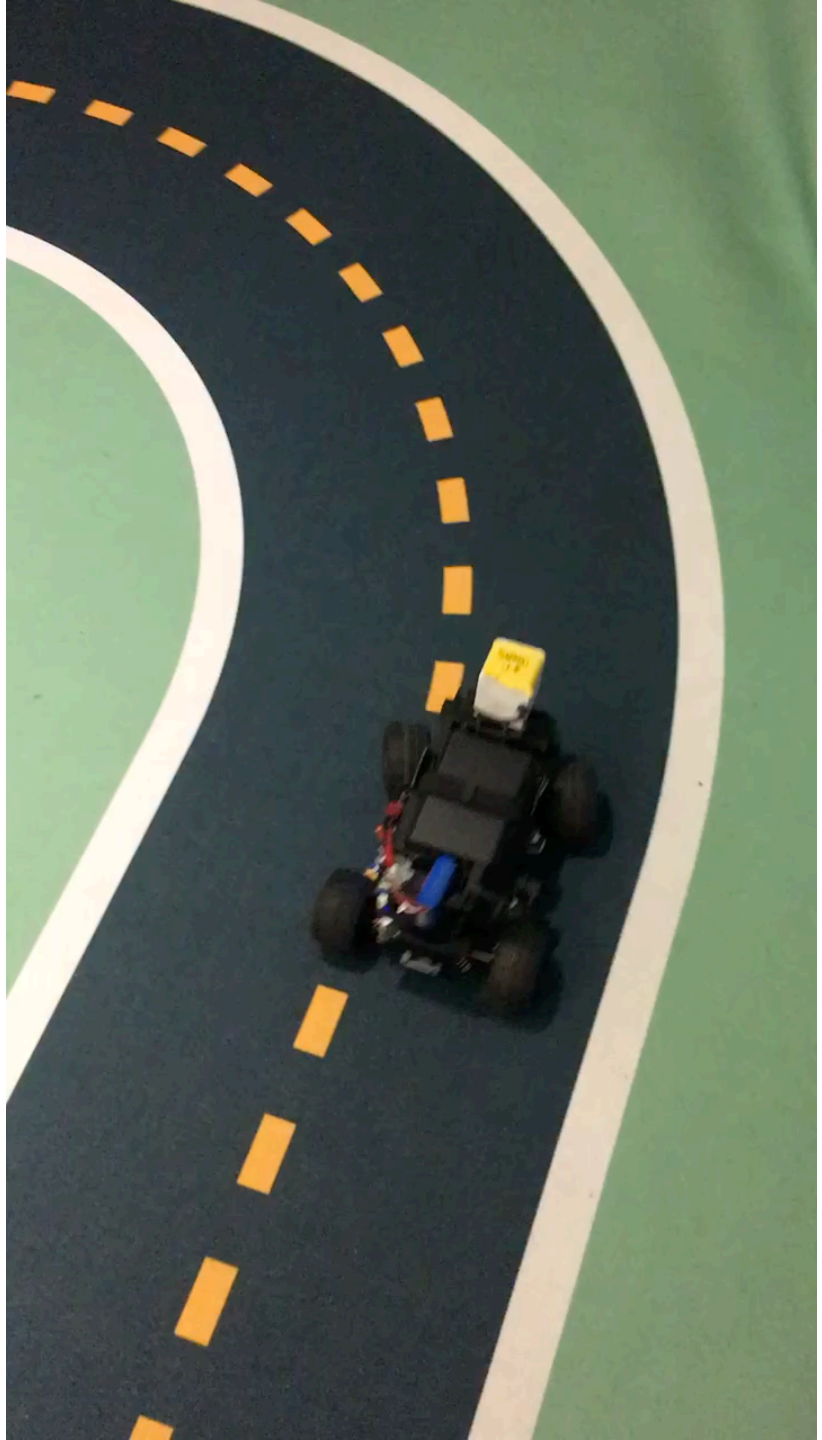
Track A Track B Track C

(b) Camera view of simulation tracks

Track A Replica Track A without barriers Tape Track on Office Carpet

(c) Camera view of real world tracks

Poor Sim2Real Transfer

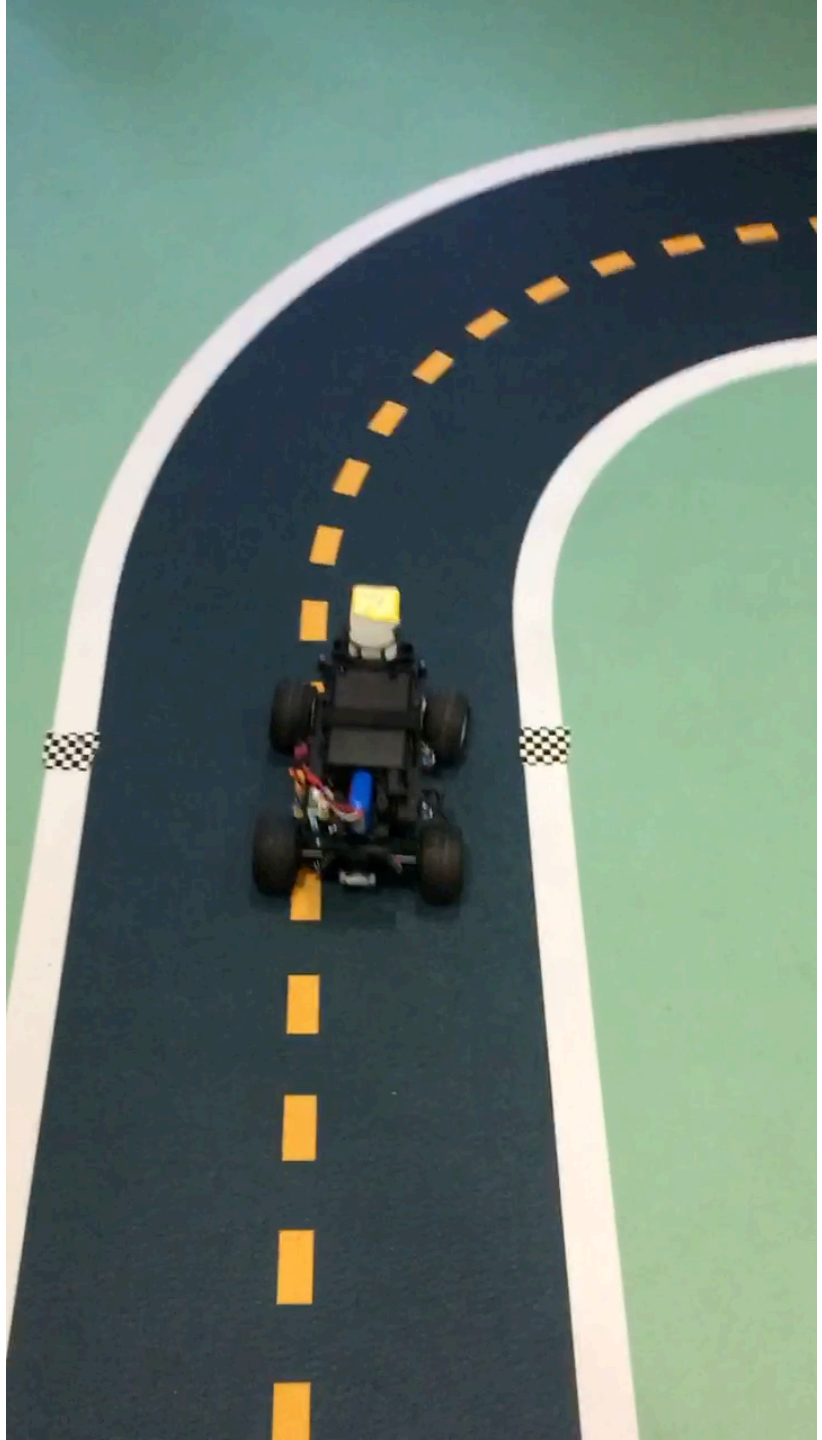


Calibration of Car Model

mit-racecar / racecar-simulator

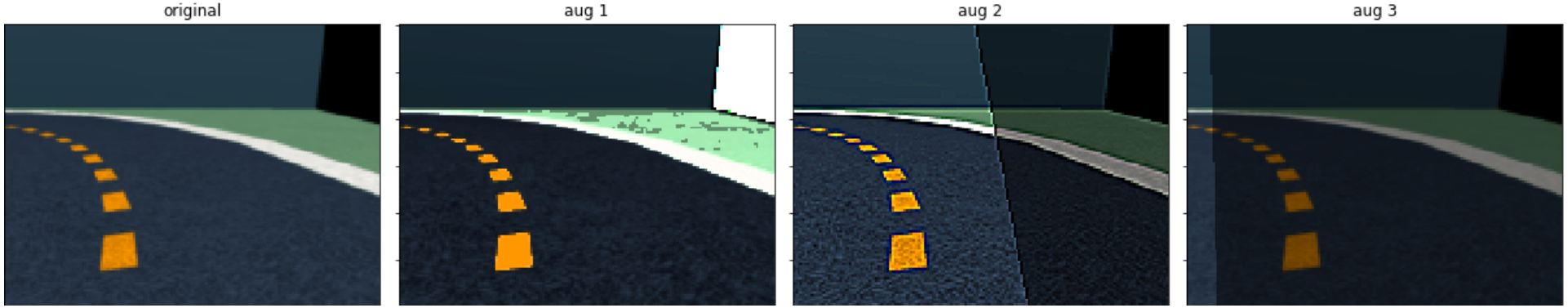
Code | Issues 6 | Pull requests 2

- Unified Robot Description Format
- Ackerman steering model
- Match DeepRacer specs
  - Wheel size
  - Camera height, angle
  - Camera field of view
  - Friction
  - Mass



Domain Randomization

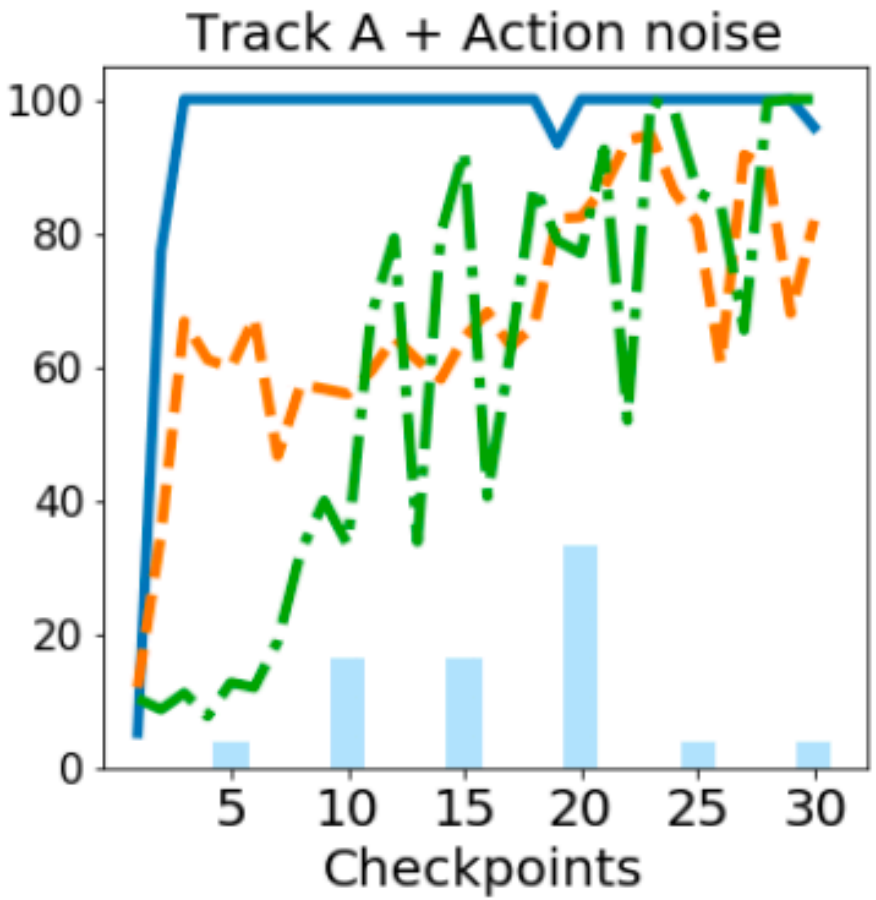
- Perception: color (brightness, hue, contrast, saturation), noise, shift, shadow

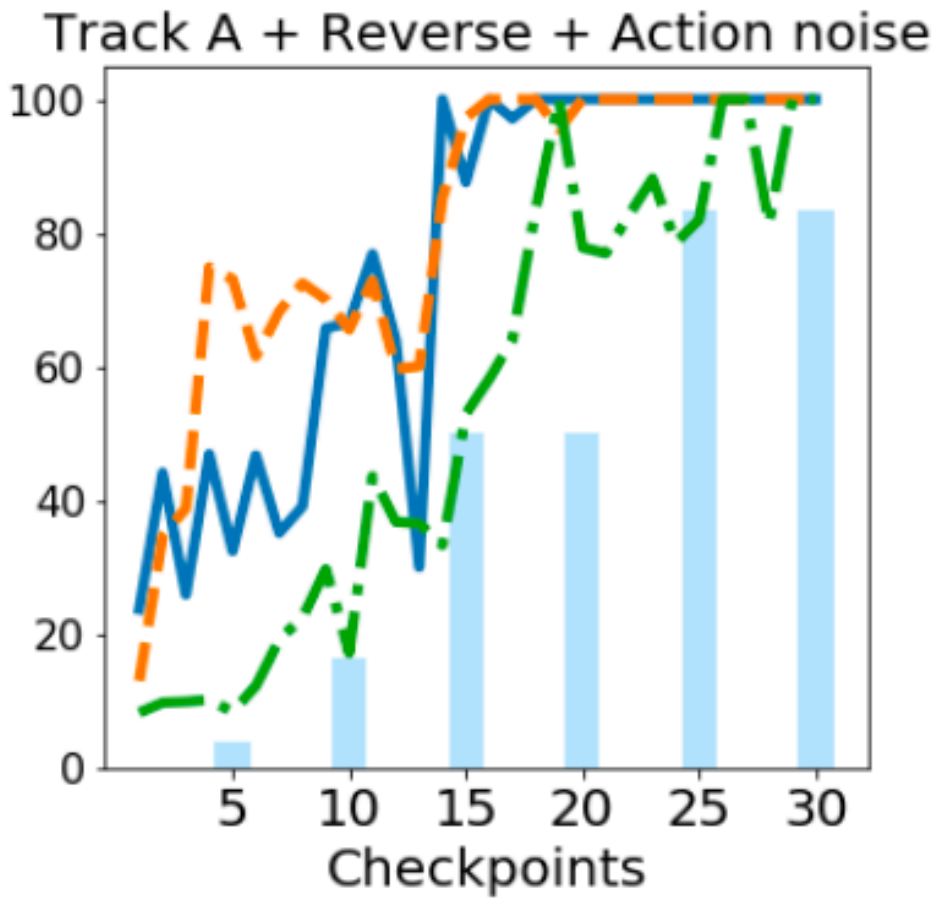


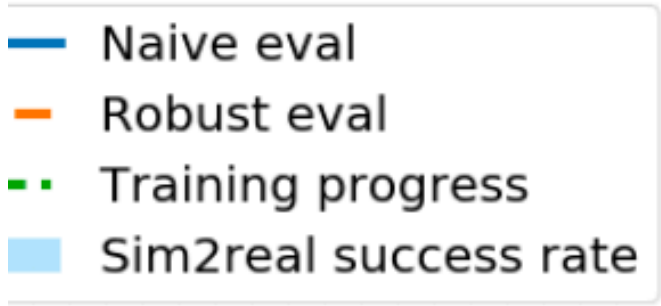
- Dynamics: action noise, driving directions, starting points, tracks
- Regularization: L2, batch normalization, dropout

Train on multiple tracks with distributed rollouts

Robust Evaluation – When should we stop training?

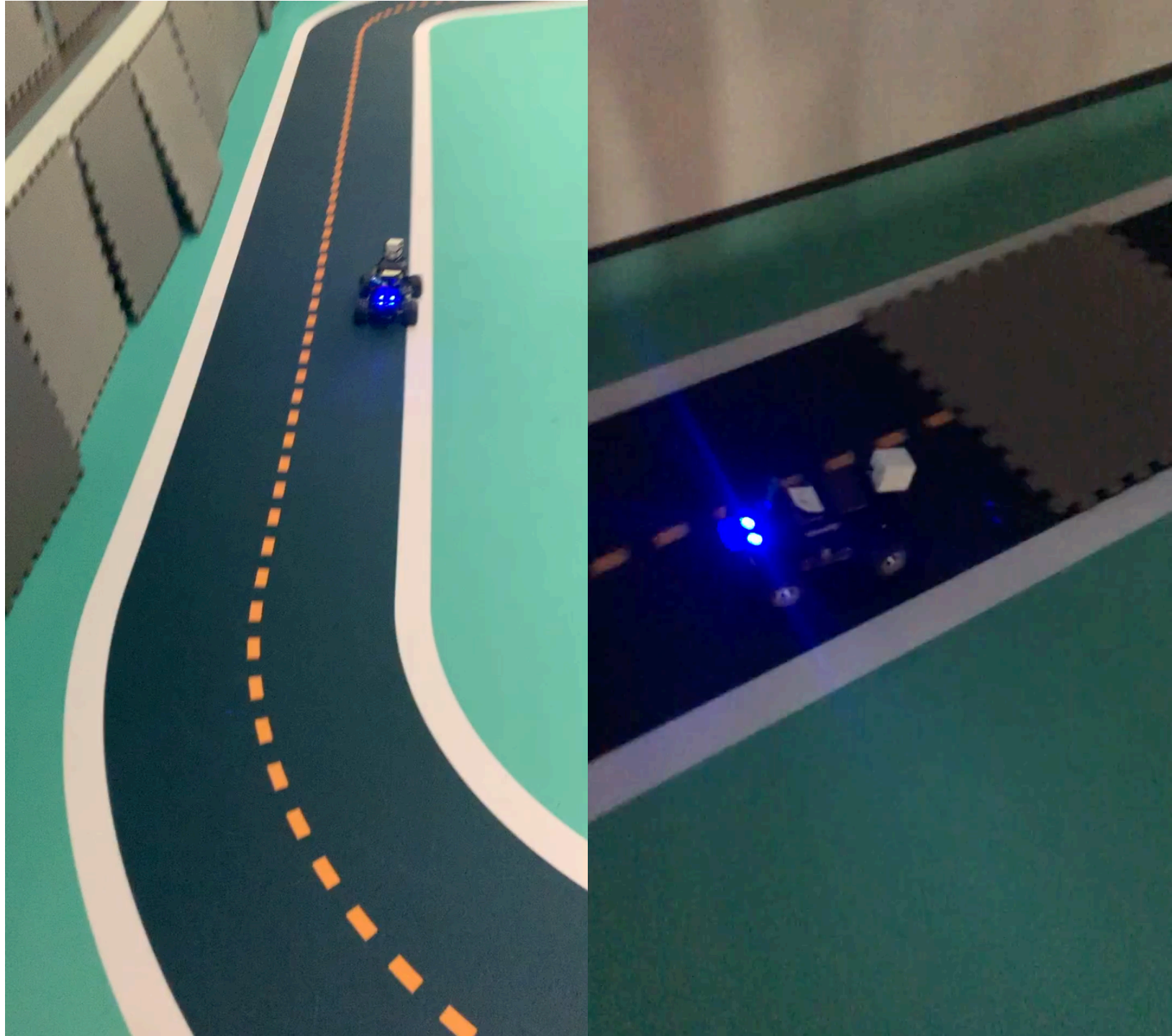






Robust Sim2Real Transfer

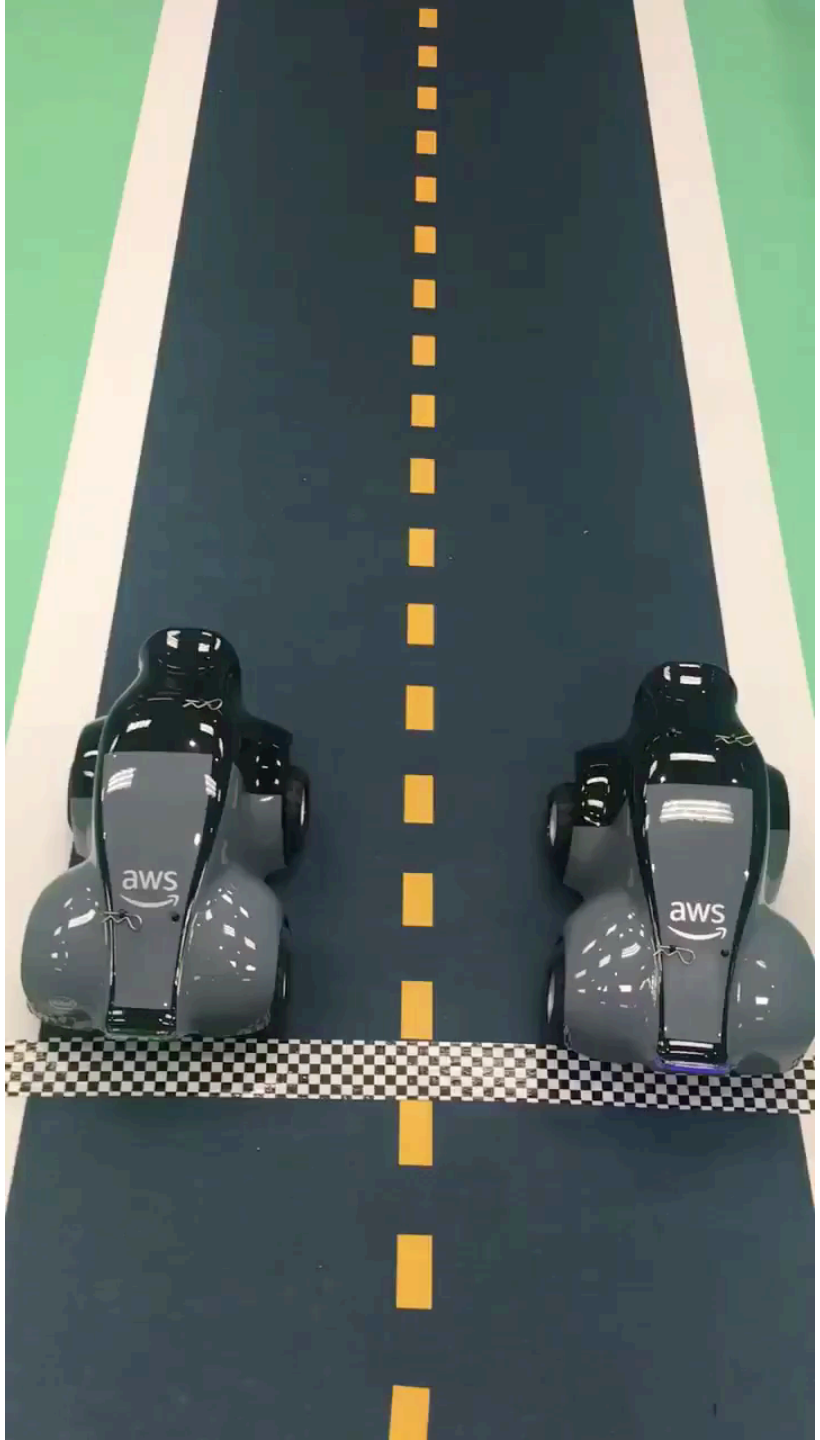
- High speed
- Low light
- Shadows
- Obstacles
- No barrier



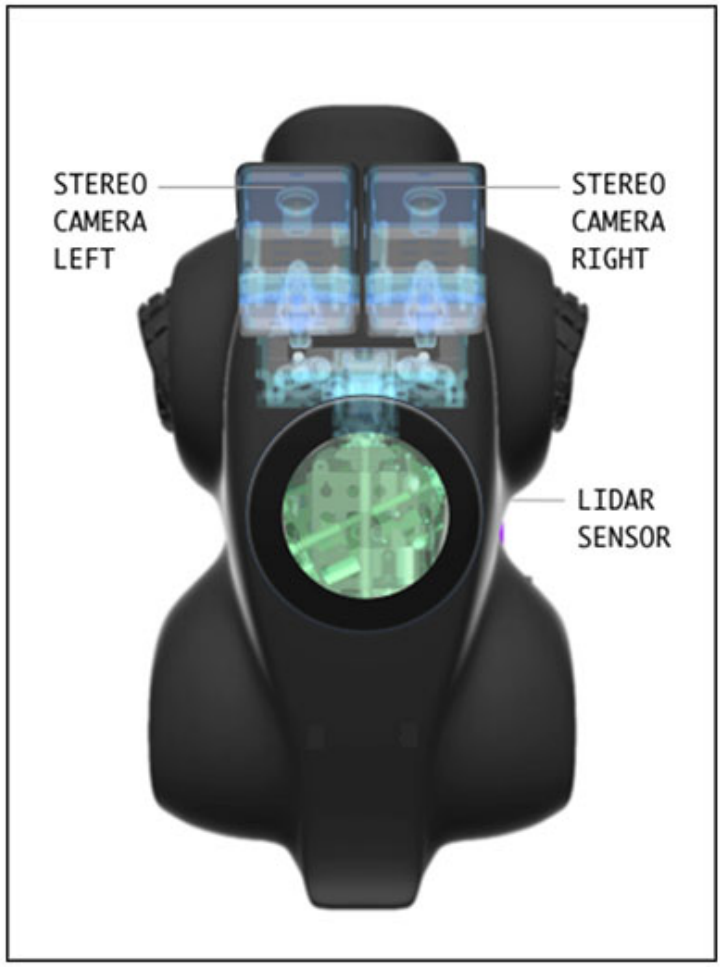
Robust Sim2Real Transfer

- Bright light
- Tape track
- Camera Blur and Shake
- Multiple surfaces

Next Challenge: Multi Car Racing



DeepRacer Evo



Zhang, Kaiqing, et al. "Robust Multi-Agent Reinforcement Learning with Model Uncertainty." Advances in Neural Information Processing Systems 33 (2020).

Thank you!

Questions?