DeepSpeed
EEE549: Final Presentation

Kaiyuan Tan, Perfect Obumneme, Emmanuel Adeloju, Sung Park, Atharva Hundare

School of Electrical, Computer, and Energy Engineering,
Arizona State University
Contacts: [ktan24, pobumnem, eadeloju, spark259, ahundare]@asu.edu

Arizona State University

Previous Reward Function (First Contest)

1 Strict Off-Track Penalty

$$\text{if off-track} \Rightarrow R = 0$$

2 Lap Completion Bonus

$$\text{if progress} = 100\% \;\Rightarrow\; R \leftarrow R + 100$$

3 Cubic Speed Factor

Let Max speed $= v_p$, Current Speed $= v$

$$g(v) = \begin{cases} 0.5, & v < 1 \\ \left(\max(0.1, \min(1, v/v_p))\right)^3, & v \geq 1 \end{cases}$$

$$R \leftarrow R \cdot g(v)$$

4 On-track Constraints

$d_c =$ Distance from center, $W =$ Track Width,

$$R \leftarrow \begin{cases} R + 1, & d_c \leq 0.4\,W \\ 0.3R, & d_c > 0.4\,W \end{cases}$$

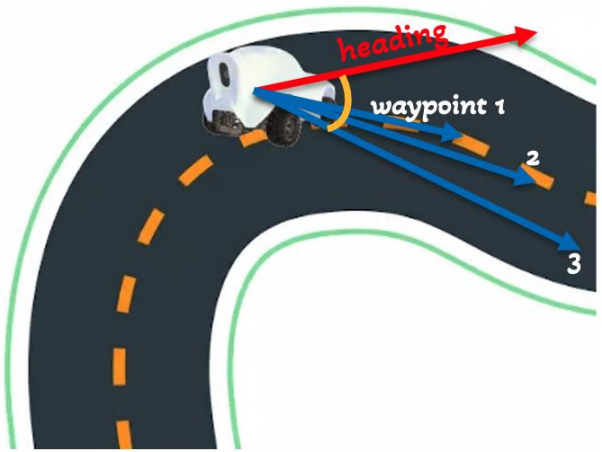
5 Pose Control (Future Direction)

$$\Delta\theta = \text{wrap}_{[-180,180]}(\theta_{\text{track}} - \theta_h)$$

If $|\Delta\theta| > 10°$:

$$e = \frac{|\Delta\theta|}{15°}, \qquad m = \max(0.01,\, 1 - e^2)$$

$$R \leftarrow R \cdot m$$



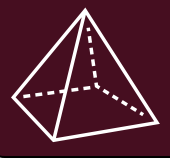
6 Sharp Steering Penalty

$$\text{if } |\text{steering}| > 20° \text{ and } v > 2.5 \Rightarrow R \leftarrow 0.7R$$

First Contest Rank

Rank	Racer	Qualifying time	Gap to 1st	Video	Off-track
1	AccessibleInterface#1358	00:39.199	-	View	10
2	Fast--and--Curious	00:49.395	+00:10.196	View	12
3	TheExplorers	00:50.462	+00:11.263	View	20
4	Horse	00:53.999	+00:14.800	View	8
5	SupervisedBySchumacher	00:55.865	+00:16.666	View	20
6	The-Reinforcers	00:56.927	+00:17.728	View	6
7	PathingNumber#6148	00:57.534	+00:18.335	View	6
8	ModelForgers	00:58.134	+00:18.935	View	14
9	DeepSpeed21	01:03.136	+00:23.937	View	16
10	LoopingCoder#4099	01:08.468	+00:29.269	View	11
11	SequentialMachine#5150	01:15.797	+00:36.598	View	21
12	AbstractRequirements#3057	01:21.463	+00:42.264	View	11
13	MemoryResource#0626	01:21.672	+00:42.473	View	17

Final Reward Function


Foundational Geometry


Track Data Preparation


Target Selection:
Lookahead Strategy


Deriving the Optimal
Steering Angle


Reward Scoring
and Evaluation

Our Final Reward function Sub-components

Foundational Geometry

Our agent is solving a **geometry problem** every timestep.

We represent the car as a point in $\mathbb{R}^2$ with a heading:

$$C = (x_c, y_c, \psi)$$

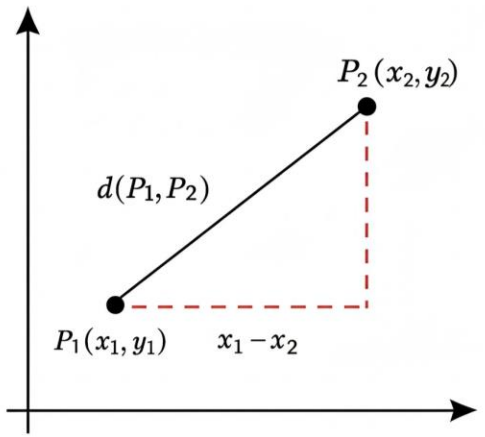
The track is a sequence of waypoints:

$$W = \{(x_i, y_i)\}$$

Everything our policy learns flow from **3** primitives

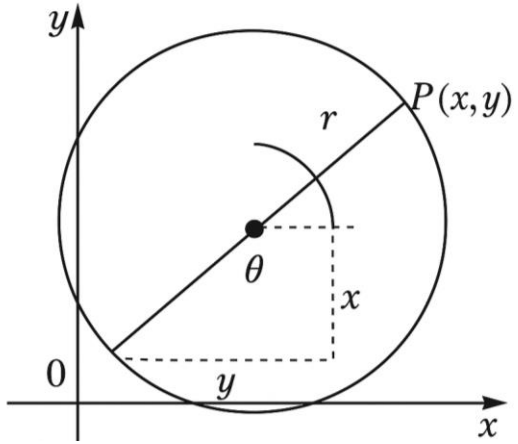
❶ Distance

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



❷ Direction (Polar)

$$(r, \theta) = \left(\sqrt{x^2 + y^2}, \arctan 2(y, x)\right)$$



❸ Shortest Rotation

$$\Delta\theta = ((\Delta\theta_{raw} + 180°) \bmod 360°) - 180°$$

Car heading: $\psi = 10°$, **Target direction:** $\theta = 300°$
Raw angle difference: $\Delta\theta_{\text{raw}} = \theta - \psi = 300° - 10° = 290°$

This suggests turning $+290°$, which is obviously wrong! That's a huge spin.

Apply normalization:
Result: $\Delta\theta = 110° - 180° = -70°$
$\leftarrow$ Turn left 70°, not right 290°

Track Data Preparation

Deepracer tracks **sparse waypoints** in a circuit.

Between two waypoints, the true path might **curve**, **dip inward**, or **swing wide** – but the agent only see **two straight-line dots**.

Upsampling waypoints makes the track feel **continuous:**

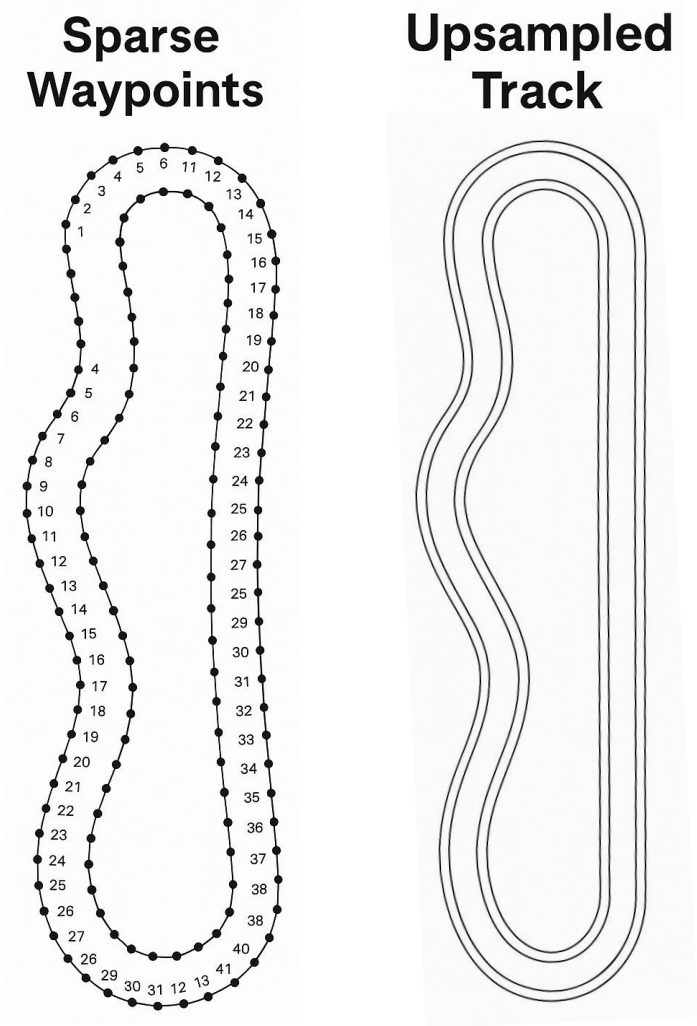
We took each segment: $P_i = (x_i, y_i), \quad P_{i+1} = (x_{i+1}, y_{i+1})$

Then generated intermediate points using linear interpolation:

$$P_{i,k} = (1 - \lambda)P_i + \lambda P_{i+1}, \quad \lambda = \frac{k}{N}$$

Where:
$N = 20$ (upsample factor)
$k \in \{0, 1, 2, ..., N - 1\}$

This yields **20 mini-waypoints per original segment.**
In code terms: *densify the map by a factor of 20.*



2022 Summit Speedway

Target Selection: Lookahead Strategy

1. Firstly, we collect the upsampled waypoints and the position of the car

$$C = (x_c, y_c)$$

Car location

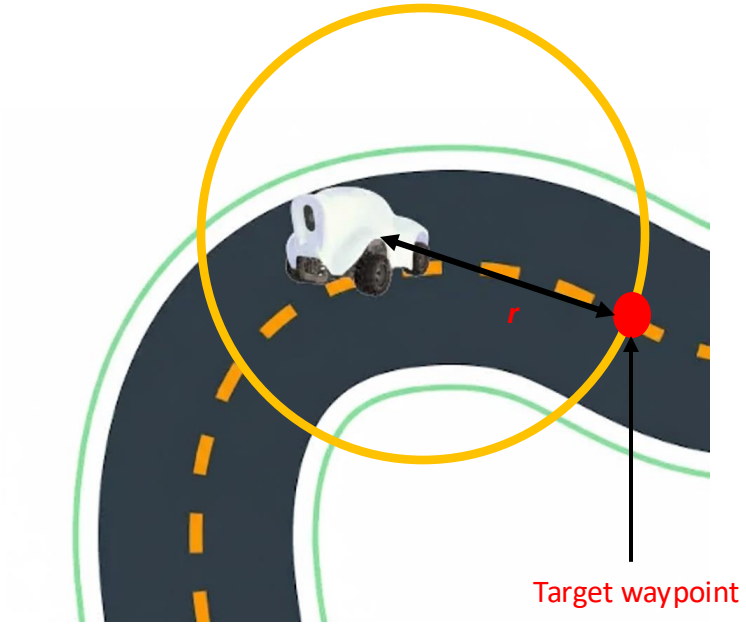
$$W = \{w_0, w_1, w_2, \dots\}$$

We reorder the waypoint list so that waypoint 0 is the closest to the car using our primitives

2. We define a lookahead radius

Radius, $r = 0.9 \cdot \text{track\_width}$
Conceptually: a circle centered at the car.

Every waypoint inside the circle is "too immediate or noisy".
We want a point just beyond this bubble to aim at.



3. Drive forward and find target

For each waypoint w_i in the driving order:
$d(w_i, C) = \sqrt{(x_i - x_c)^2 + (y_i - y_c)^2}$

Stop at the first waypoint where: $d(w_i, C) > r$

$Target = w_j$ s.t. $d(w_j, C) > r$ and j is minimal.
All edge cases fallback to the closest waypoint.

Deriving the Optimal Steering Angle

```
def get_target_steering_degree(params):
    tx, ty = get_target_point(params)
    car_x = params['x']
    car_y = params['y']
    dx = tx-car_x
    dy = ty-car_y
    heading = params['heading']
    _, target_angle = polar(dx, dy)
    steering_angle = target_angle - heading
    return angle_mod_360(steering_angle)
```

We chose our target, $T = (t_x, t_y)$

Determine car position, $C = (x_c, y_c, \psi)$

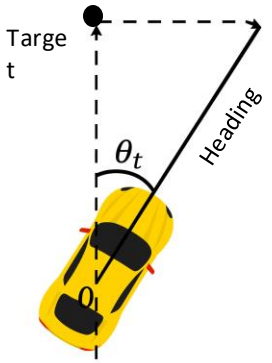
Compute the target-relative vector

$$\hat{v} = (dx, dy) = (t_x - x_c, t_y - y_c)$$

Compare target direction vs car heading

We use our primitive to convert to polar coordinates:

$$(r, \theta) = polar(dx, dy) = \left(\sqrt{x^2 + y^2}, \arctan 2(y, x)\right)$$



Car heading: $\psi = params[heading]$,
Raw steering request: $\Delta\theta_{raw} = \theta - \psi$

Normalize to shortest rotation (primitive):
s.t. $\Delta\theta \; \epsilon \; [-180°, 180°]$

Reward Scoring and Evaluation

```python
def score_steer_to_point_ahead(params):
    best_stearing_angle = get_target_steering_degree(params)
    steering_angle = params['steering_angle']
    error = (steering_angle - best_stearing_angle) / 60.0
    score = 1.0 - abs(error)
    return max(score, 0.01)

def reward_function(params):
    return float(score_steer_to_point_ahead(params))
```

From our geometry pipeline:
we already know the best steering angle, θ_{best}

The simulator gives us the actual steering: θ_{car}

Compute the steering error: $e = \Delta\theta_{car} - \Delta\theta_{best}$

$$\begin{cases} > 0 \rightarrow \text{car is turning too much to the right} \\ < 0 \rightarrow \text{car is turning too much to the left} \\ = 0 \rightarrow \text{perfect alignment} \end{cases}$$

We normalize the error: $\frac{e}{60}$

$$\begin{cases} \pm 60^\circ & \text{mismatch} = \text{total failure} \\ < \pm 10^\circ & \text{mismatch} = \text{good} \\ < \pm 5^\circ & \text{mismatch} = \text{excellent} \end{cases}$$

$$Score = 1 - |e|$$

$$\begin{cases} |e| = 0 \Rightarrow \text{score} = 1.0 \text{ (perfect)} \\ |e| = 0.2 \Rightarrow \text{score} = 0.8 \text{ (pretty good)} \\ |e| = 0.6 \Rightarrow \text{score} = 0.4 \text{ (poor)} \\ |e| \geq 1.0 \Rightarrow \text{score} \leq 0 \text{ (bad)} \end{cases}$$

Final reward is max(score, 0.01) to avoid reward collapse

The car is rewarded for being close to ideal, not simply surviving.

Reward Function Highlight

- We do **NOT** incentive our Agent for **SPEED**
- We do **NOT** incentive our Agent for wheels **ONTRACK**
- We do **NOT** incentive our Agent for **PROGRESS**
- We **ONLY** incentive our Agent for **good STEERING decisions**

Observations:

A car that aligns its heading with the future of the track will do this:

- Enter corners earlier
- Cut apexes naturally
- Exit with straightened steering

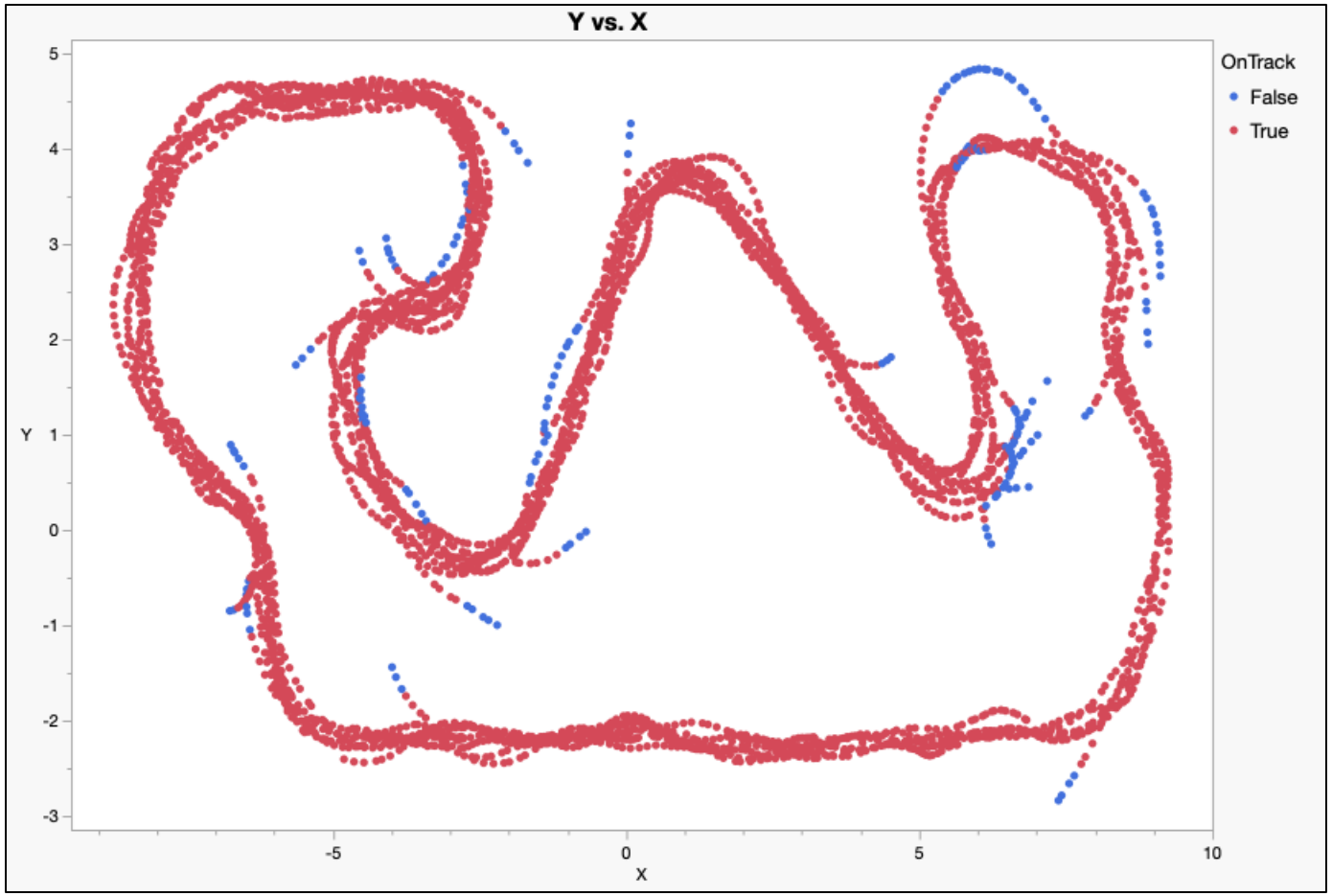
These are the recipe for **SPEED** and **STABILITY…**

Agent Configuration and Hyperparameters

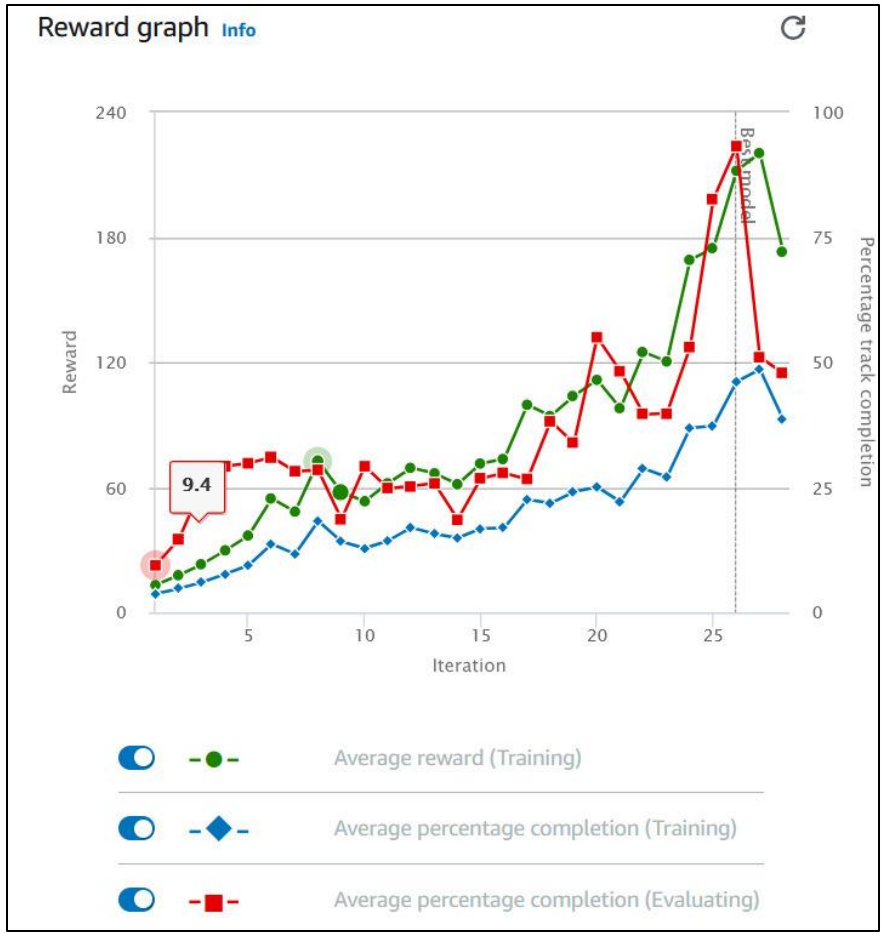
Category	Item	Value
Action space	Type	Continuous
	Speed range	0.5m/s : 3.7 m/s
	Steering angle range	28° : – 27°
Framework	Library	TensorFlow
Reinforcement learning	Algorithm	PPO

Hyperparameter	Value
Gradient descent batch size	64
Entropy	0.01
Discount factor	**0.99** → **0.7**
Loss type	Huber
Learning rate	0.0003
Number of experience episodes between each policy-updating iteration	20
Number of Epochs	10

Training Results

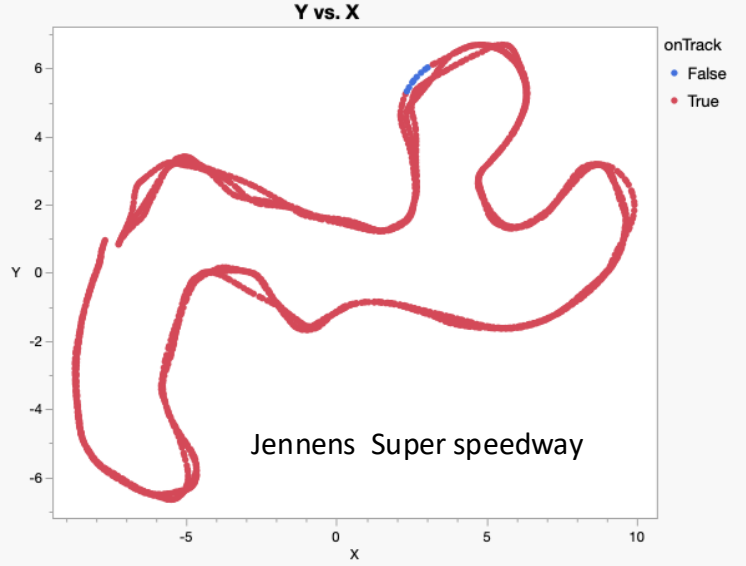


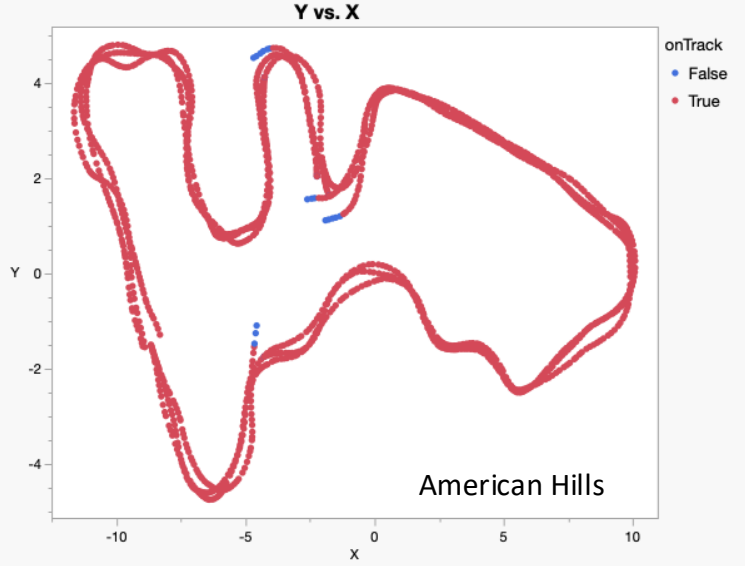
Training Track
Circuit de Barcelona-Catalunya

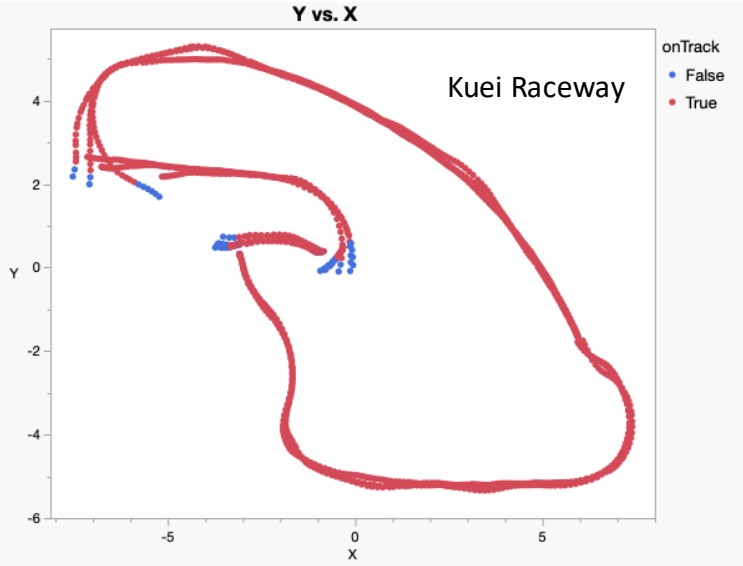


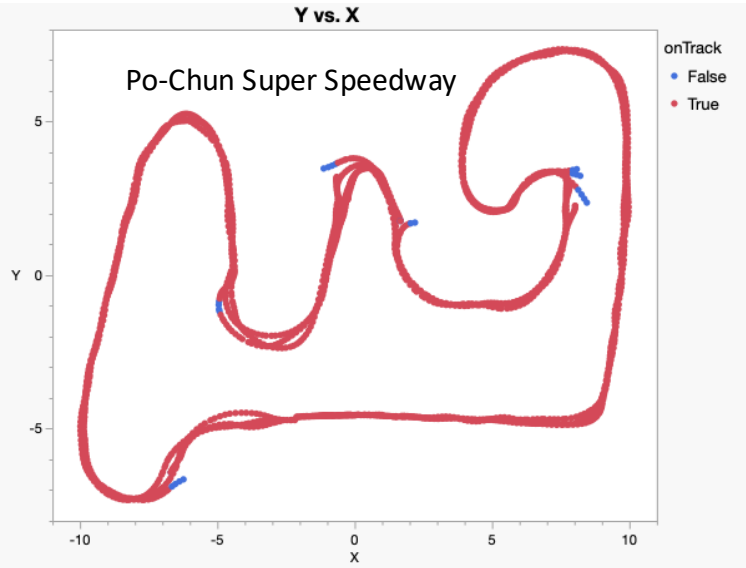
Training Time
190 minutes

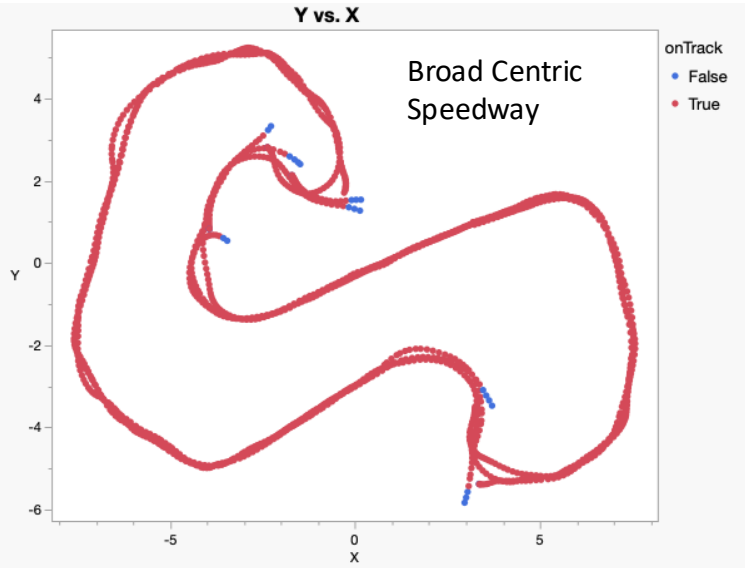
Evaluation (Stable on Different Tracks)

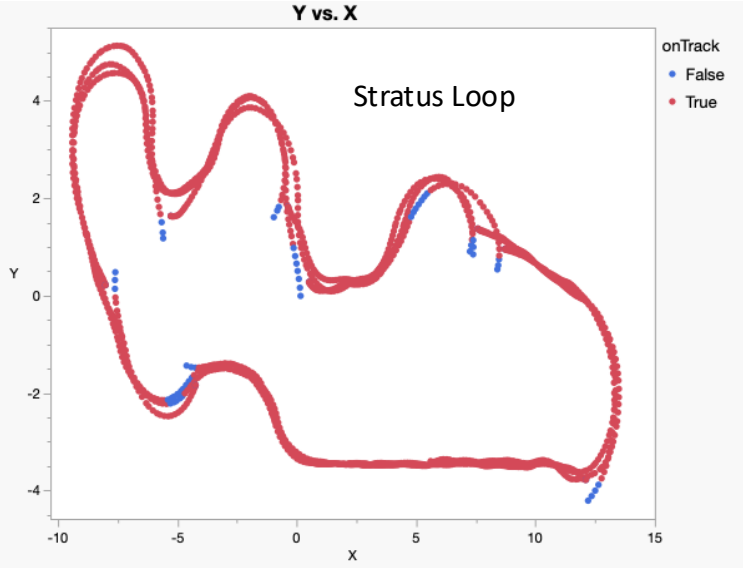




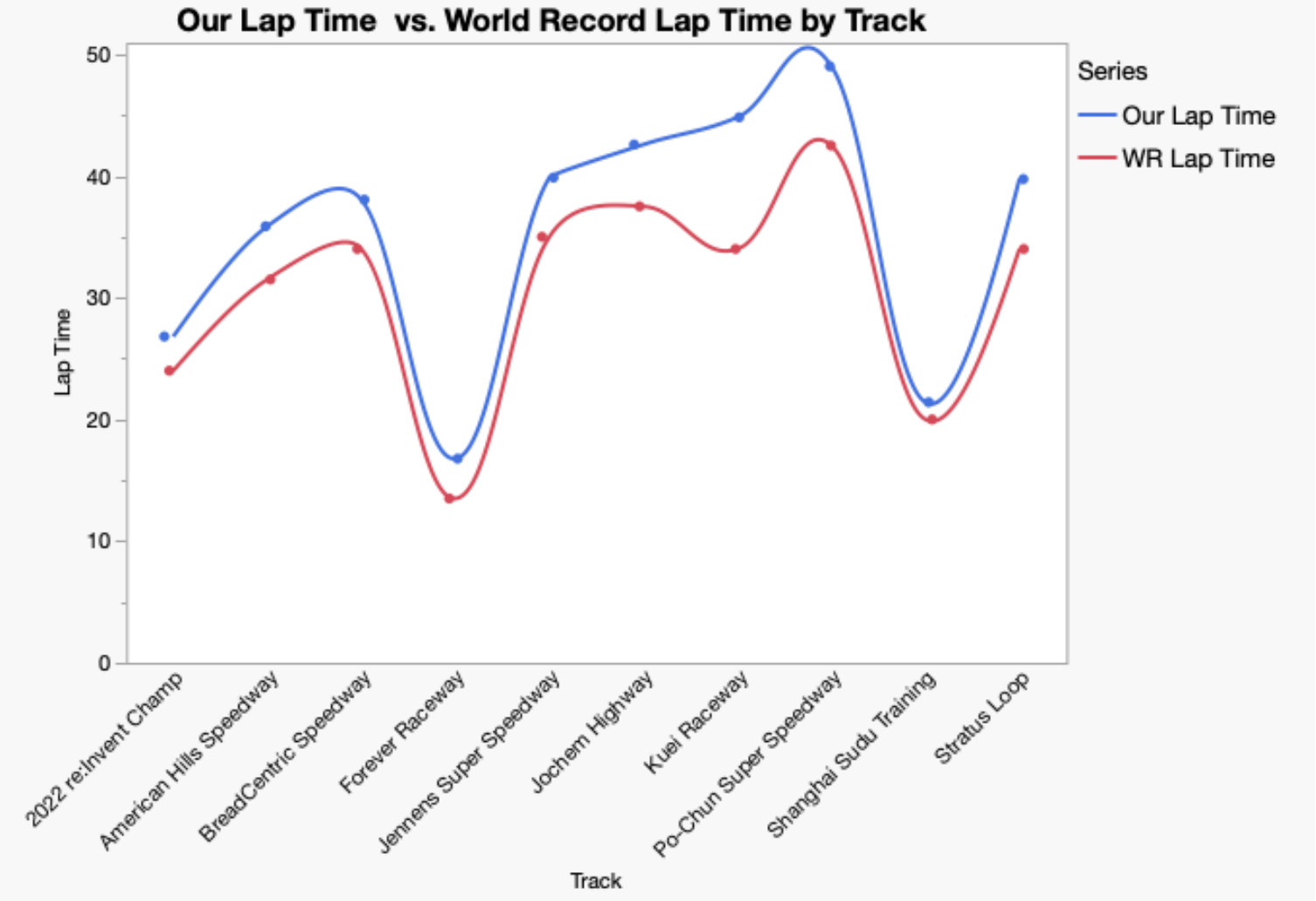






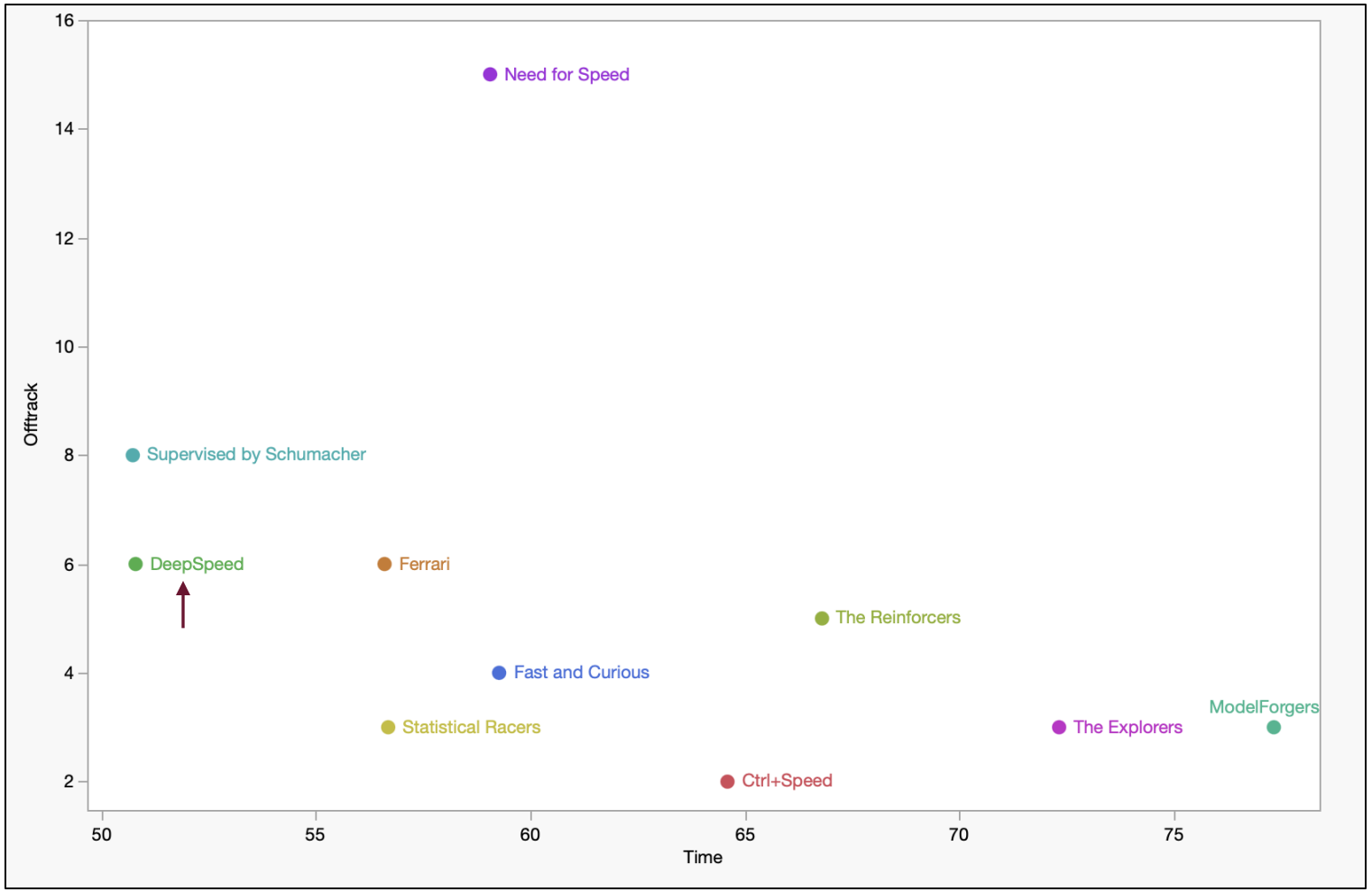


Evaluation (Our Lap Time vs. WR Lap Time by Track)



- **Best Lap Time**: Shanghai Sudu Training
  - ~7% slower
- **Worst Lap Time**: Kuei Raceway
  - ~32% slower

Final Contest Performance (Offtrack vs. Lap Time)



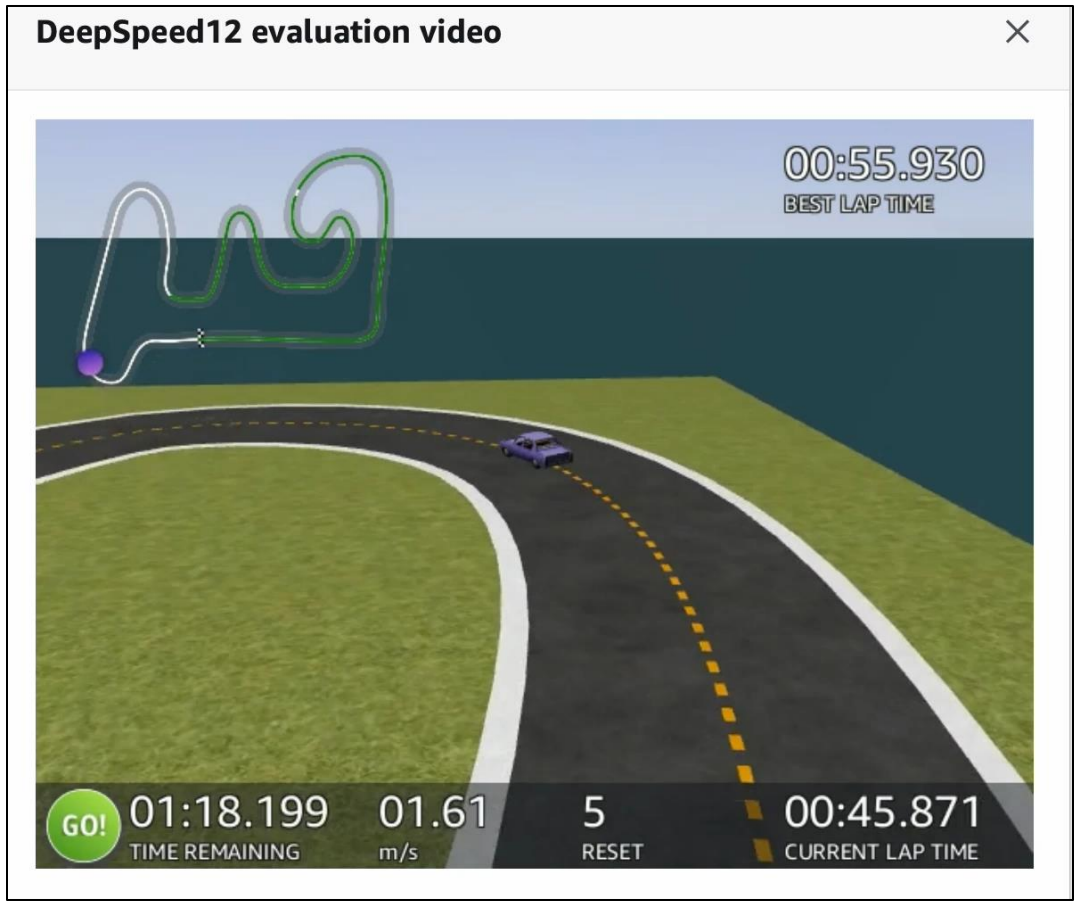
This plot shows the tradeoff Between raw speed and off-track penalties.

Teams that drove cautiously sacrificed lap time.

Teams that pushed too hard accumulated resets.

Top performers found the balance - Need for speed is the extreme outlier.

Future Reward Function Iteration



Corner Stability:
Reduce over-rotations and rare spin-outs on high curvature by penalizing sudden steering jumps

Smooth Acceleration:
Incentivize controlled throttle change to prevent slip

Speed Awareness:
Introduce reward for speed but only when aligned to the correct steering angle

Lap Completion Bonus:
Reward fast laps only after finishing the full

Team Member Contributions

Team Member	Contribution
Kaiyuan Tan	• Create and train models with different reward functions
Perfect Obumneme	• Iterated through several models and reward functions • Typeset all PPT equations and plotted PPT graphs
Emmanuel Adeloju	• Ran different models on different tracks using different reward functions
Sung Park	• Tried different approaches for reward functions and finetuning. • Worked on some Evaluation, and Car-model and Hyperparameters PPT
Atharva Hundare	• Performed the final reward function design,training and evaluation on various tracks.

THANK YOU!