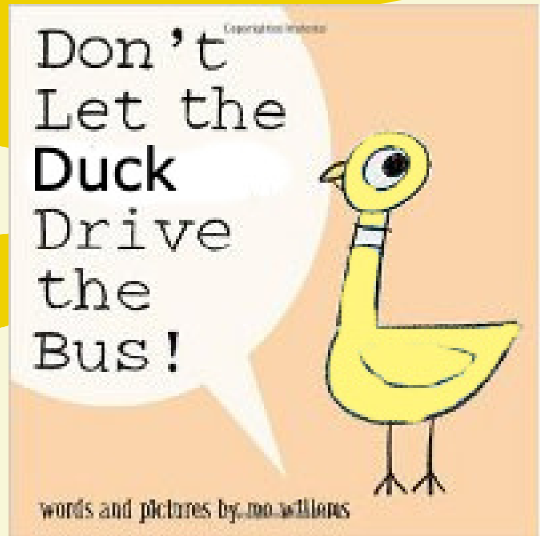
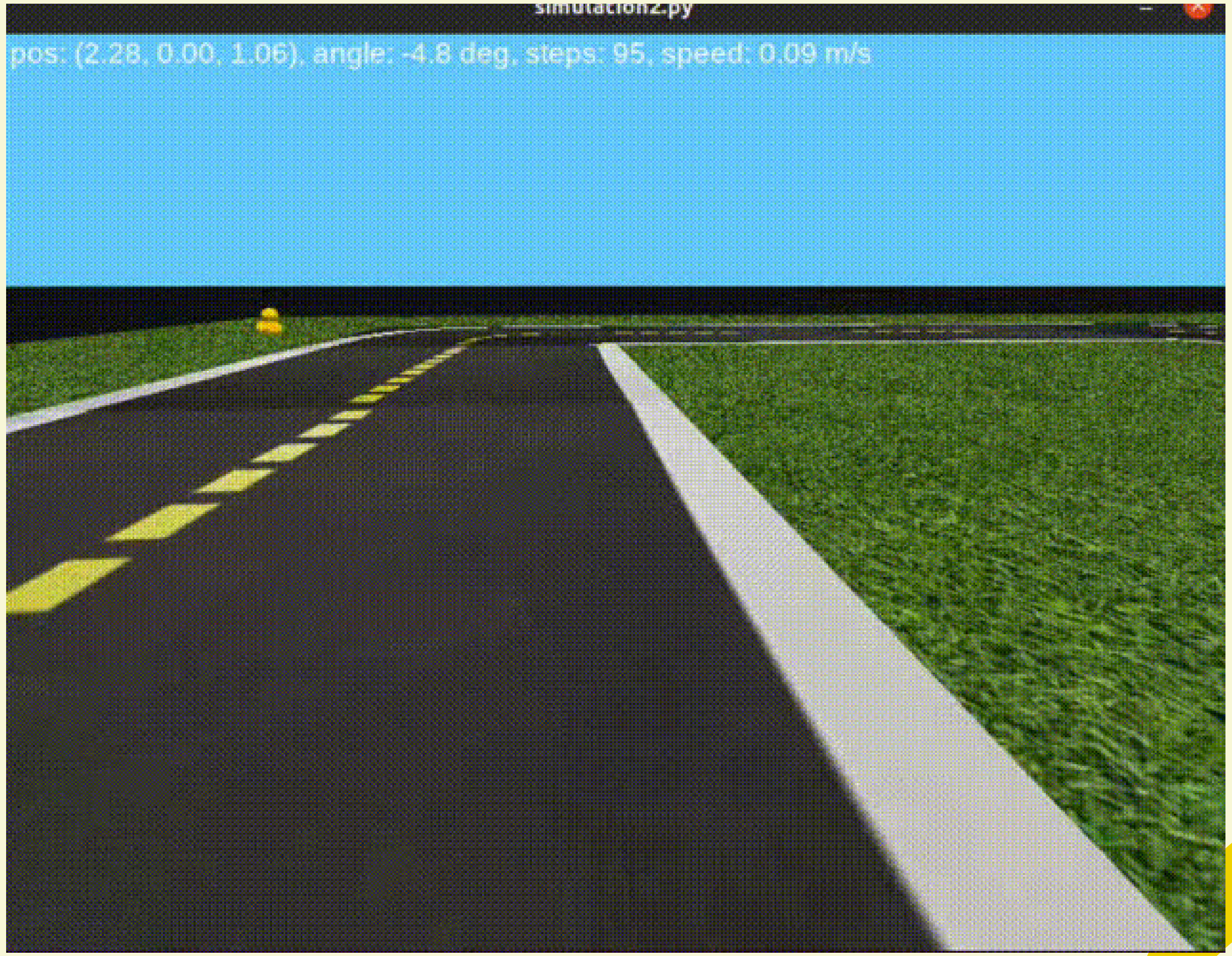
LANEQUACKER

Presented by Joshua Choi, Marcus Ha,
Martin Bobarshad

PROBLEM



- **It is difficult for ducks to drive!**
  - Continuous and Dynamic Environment
  - Partial Observability (Limited Sensor Input)
  - Complex, Non-discrete Action and Observation Spaces



PROJECT GOALS

1. Train ducky bot for lane-following, obstacle avoidance, and stopping as needed

2. Expected outcome: See Duckiebot driving in action!

APPROACHES

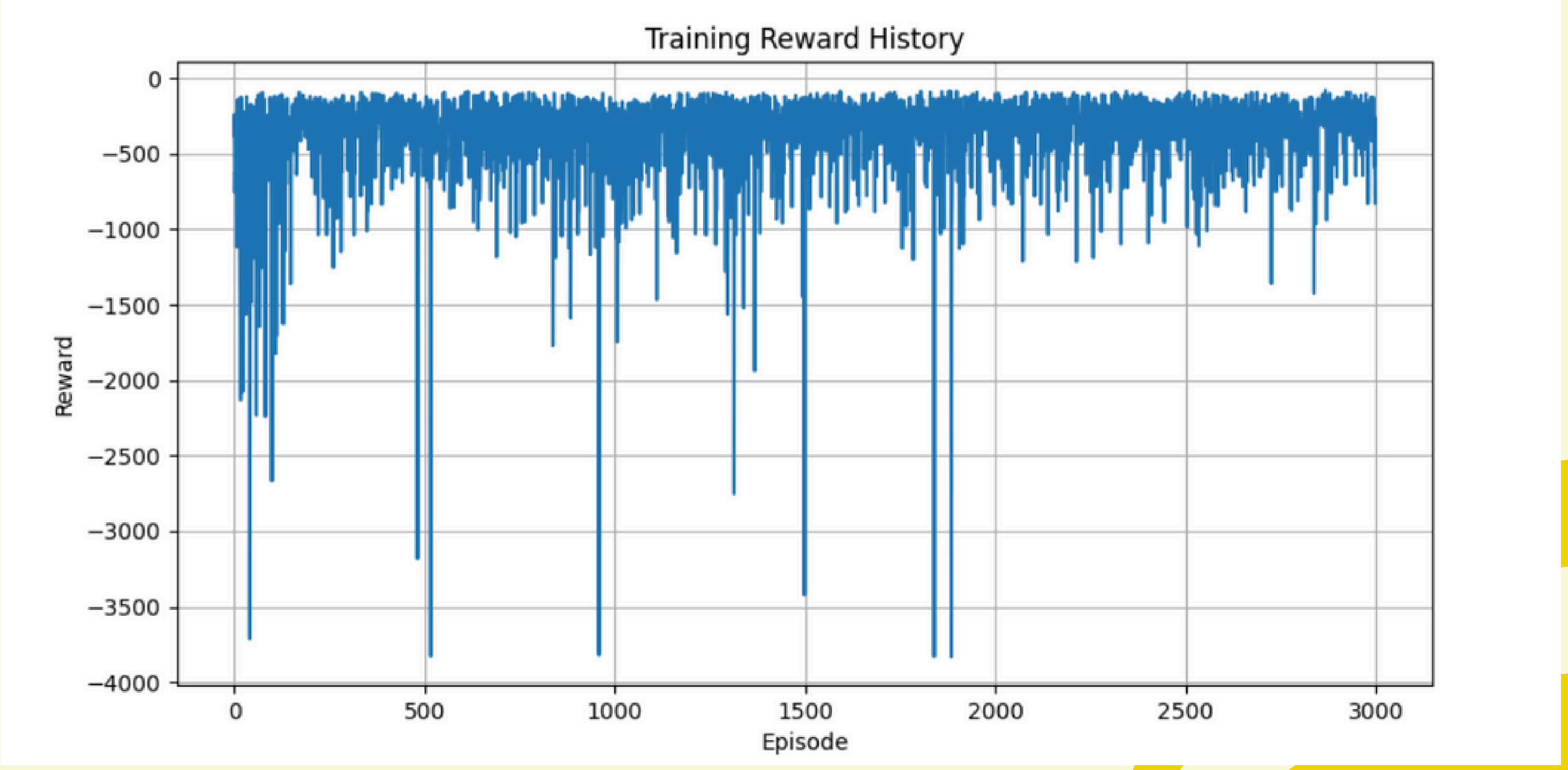
1. **Soft Actor Critic (SAC)**
2. Proximal Policy Optimization (PPO)

SOFT ACTOR CRITIC (SAC)

- Off-policy actor-critic RL algorithm
  - Small reward for forward movement
  - Large penalty for collision
  - Small/Large penalty for driving off-lane (depending on distance from center)

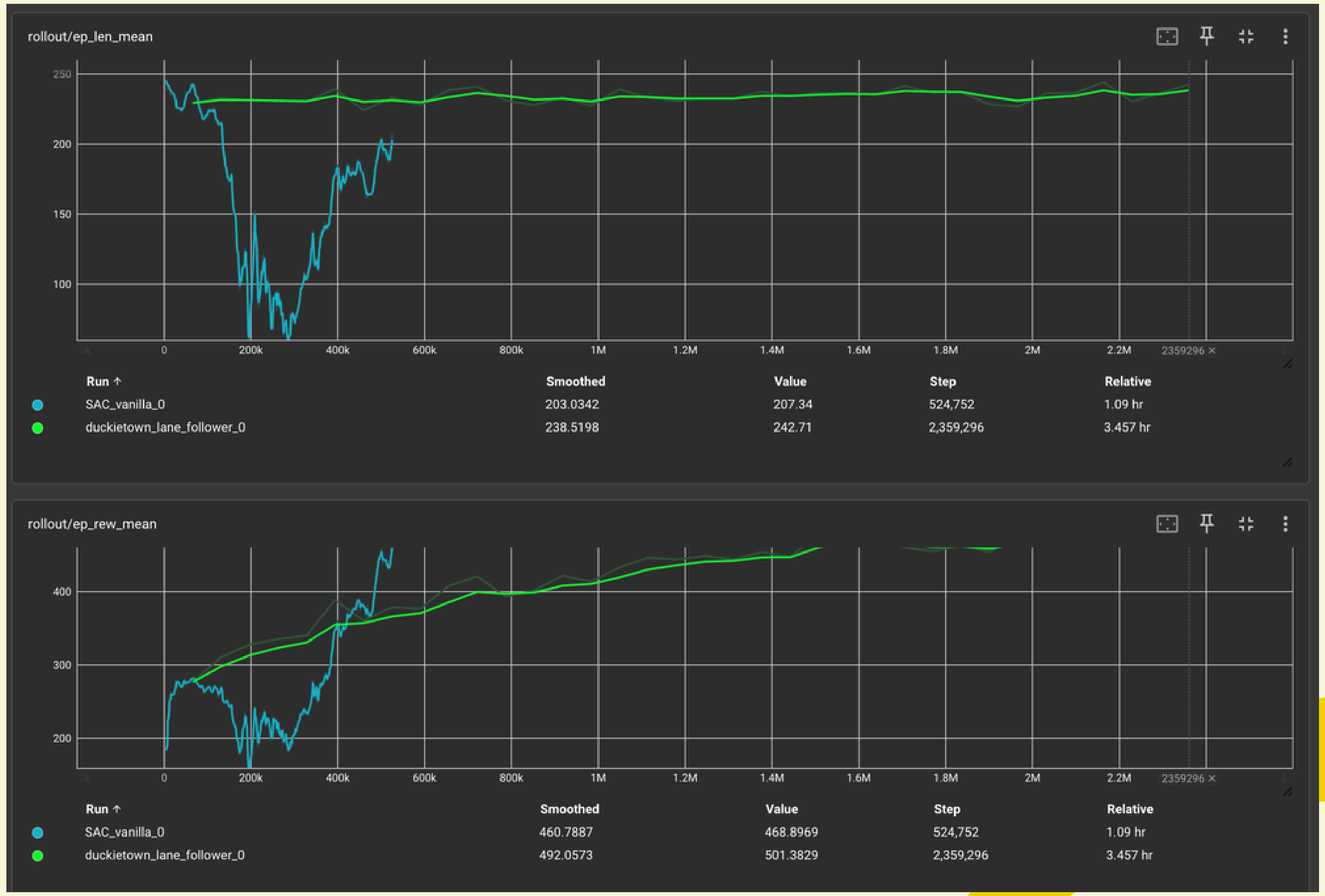
- SAC showed less stable training results, making convergence uncertain.

- It required extensive hyperparameter tuning.



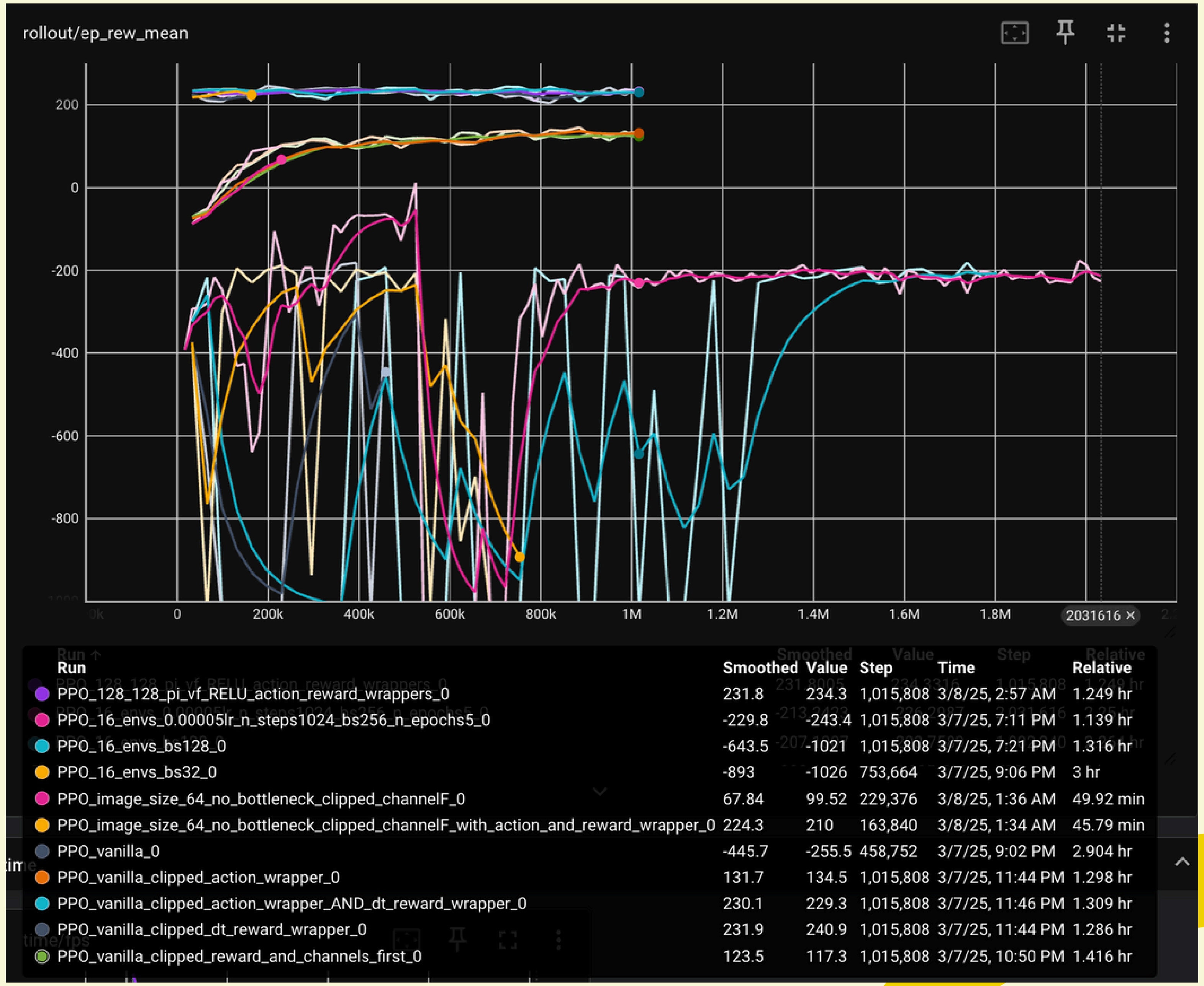
PROXIMAL POLICY OPTIMIZATION (PPO)

- On-policy learning is preferred for its simplicity and stable updates
  - This is especially useful when using CNN inputs for navigation
  - CNN extracts lane and object features
  - Inputs predict actions like steering and velocity



PPO TRAINING

- Built on Exercise 2 with additional **hyperparameter tuning** for optimization.
- Designed a **custom reward system** to improve learning.
- Trained for **5M timesteps** in both an **empty loop** and a **loop with obstacles**.
- Monitored progress via **TensorBoard** and recorded videos every **250K timesteps**.



CHALLENGE #1: EXCESSIVE IMAGE SIZE AND SLOW TRAINING

Issue:

- Native Duckietown camera resolution
  - **640x480**
- Large observation space resulted in extremely large models **(~5GB)**.
- Training was **excessively slow**, with episodes lasting **too long** (high max_steps).
- Debugging and experimentation became challenging.

Solution:

- Fixed camera resolution
  - **3x84x84 (3 color channels)**
- **Significantly simplified** the observation space.
- Dramatically **decreased model size** and **memory usage**.
- **Training speed improved substantially**, enabling rapid experimentation.

CHALLENGE #2: INEFFECTIVE DEFAULT REWARD FUNCTION

Issue:

- Duckietown's default reward function provided **sparse** and **insufficient feedback**.
- **Poor guidance** for the agent in a continuous, dynamic environment.
- Agent **frequently failed** to stay within lanes or navigate smoothly.

Solution:

- Designed a custom, dense reward function, providing immediate feedback.
- **Implemented multiple reward components:**
  - **Distance traveled along the track reward**
  - **Lane position, Proximity, and Collision penalties**

HYPER PARAMETERS

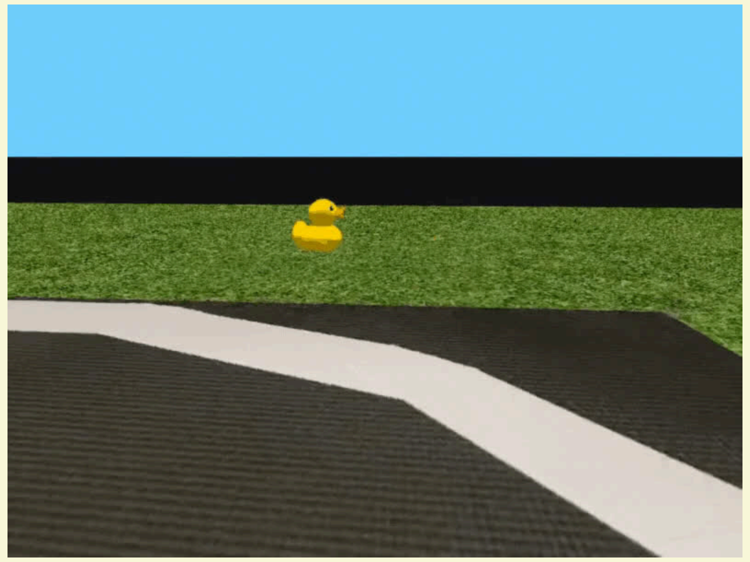
- **Model Parameters**
  - Learning Rate: 0.0001
  - N Steps: 4096
  - Batch Size: 256
  - N Epochs: 10
  - Gamma: 0.98
  - GAE Lambda: 0.95
  - Clip Range: 0.3
  - Entropy Coefficient (ent_coef): 0.05
  - Value Function Coefficient (vf_coef): 0.5
  - Max Gradient Norm: 0.5

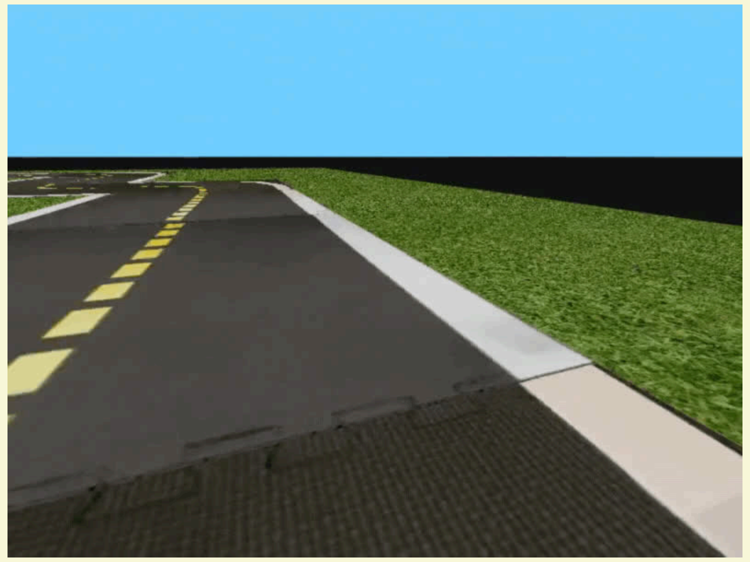
- **Simulator Parameters**
  - Domain Randomization: False
  - Max Steps per Episode: 250
  - Draw Curve: False
  - Draw Bounding Box: False
  - Camera Width: 84
  - Camera Height: 84

- **Training Configuration**
  - Seed: 47
  - Number of Environments: 8
  - Reinforcement Learning Algorithm: PPO
  - Total Timesteps: 5,000,000

CUSTOM REWARD FUNCTION

1. **Encouraging Forward Motion:**
   - The agent is rewarded for moving forward in a stable manner.
   - Prevents the vehicle from spinning in place or getting stuck.
2. **Penalizing Lane Deviations:**
   - Negative rewards are applied when the vehicle moves out of its lane.
   - Encourages alignment with the center of the lane to improve navigation.
3. **Minimizing Collisions**:
   - A heavy penalty is assigned when the vehicle collides with obstacles.
   - Ensures the model learns to avoid objects and drive cautiously.
4. **Discouraging Reverse Driving:**
   - A slight penalty is given when the agent moves in reverse unnecessarily.
   - Reinforces forward driving as the optimal strategy.
5. **Lane-Centric Corrections:**
   - The agent receives feedback based on lane-center distance and angle.
   - Helps refine turning and smooth lane-following behavior.





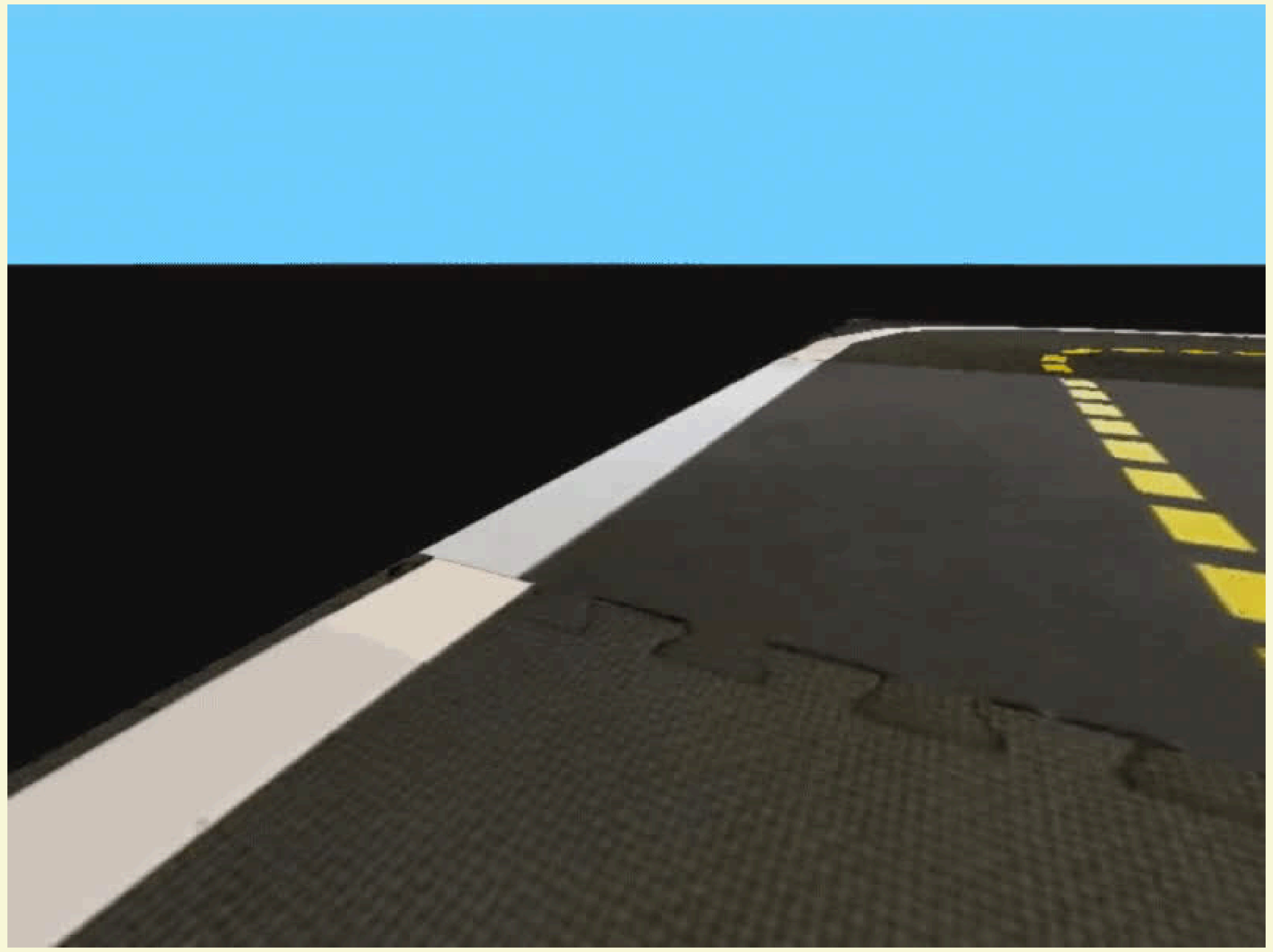
REWARD FUNCTION IMPROVEMENT: PREVENTING REVERSING BEHAVIOR

Additional Observation:

- Early models often reversed to correct their mistakes.
- Frequent reversing resulted in erratic and inefficient driving patterns ("squiggly paths").
- This behavior hindered stable progression and reduced overall rewards.

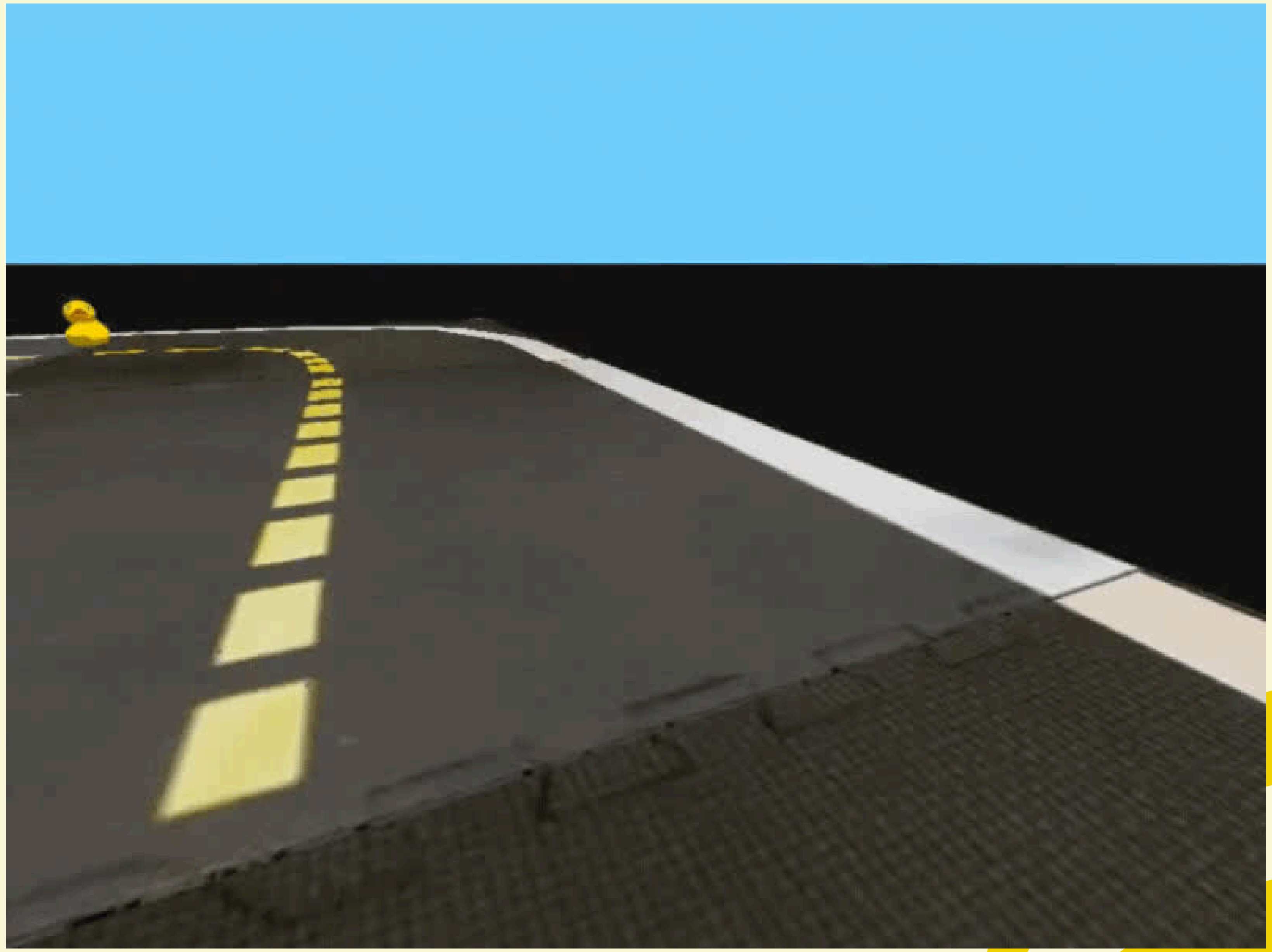
Solution:

- Introduced a severe penalty for reversing movements.
- Agent motivated explicitly to move forwards or sideways—never backward.
- Stabilizing driving results:
  - Smoother trajectories.
  - Consistent lane following.
  - Overall better agent performance and stability.



```python
# Check if reversing (both left and right engine values are negative)
left_engine, right_engine = action
if left_engine < 0 and right_engine < 0:
    # Apply penalty for reversing
    return -25
```

RESULTS



- Our agent can drive quickly, smoothly, and keep centering the lane in the small empty loop.

- Our agent can avoid obstacles in our custom map although inconsistent

THANK YOU