Autonomous Race Strategy via Reinforcement Learning

Autonomous Race Strategy via Reinforcement Learning: Badger Solar Racing

Jesse Schmolze | University of Wisconsin-Madison | Research in Progress

---

Overview

A solar car race is, at its core, an energy management problem under uncertainty. The American Solar Challenge (ASC) is a ten-day, cross-country race spanning over 1,000 miles of varied terrain. A driver receives one speed command every five minutes, meaning 96 sequential decisions determine whether the car finishes the day with charge to spare or strands itself on the side of the road. The fundamental difficulty is that all the relevant variables are coupled: going faster depletes the battery faster, but going too slow on a flat stretch wastes a window when solar input is high. Terrain grade, time of day, and battery state all interact in ways that make intuitive heuristics unreliable and simple rule-based strategies leave substantial distance on the table.

This project builds an autonomous race strategy system that learns to solve this sequential decision problem. The system tells the driver a target speed, updated every five minutes, that maximizes total distance traveled while keeping the battery alive for the full race day. The approach combines a physics-based vehicle simulation, a neural network surrogate trained to mimic that simulation, and a Proximal Policy Optimization (PPO) reinforcement learning agent trained on top of the surrogate.

System Architecture

The pipeline has four components that feed into each other in sequence.

The foundation is a physics simulation built in MATLAB/Simulink. The model captures the full vehicle dynamics relevant to race strategy: motor efficiency curves, battery state-of-charge evolution, aerodynamic drag, rolling resistance, grade effects from GPS elevation data, and solar array output. The car modeled here has a 5.22 kWh usable battery, a motor rated at 2 kW continuous, a drag coefficient of 0.087, and a total mass of 280 kg. Python communicates with the Simulink model via the MATLAB Engine API, allowing the RL

pipeline to drive the simulation programmatically. Each five-minute simulated timestep takes a target speed as input and returns the full updated vehicle state.

The challenge with training an RL agent directly against the Simulink model is computational. Running the simulation the hundreds of thousands of times required for policy convergence is infeasible in wall-clock time. To address this, a surrogate neural network is trained to approximate the simulation's state transition function. Given the current vehicle state and a speed command, the surrogate predicts the next state. It is a feedforward network with three hidden layers of 256 units each, trained on 10,000 state-action-nextstate tuples generated by running the real simulation across randomized initial conditions sampled via Latin Hypercube Sampling. The surrogate runs approximately 1,000 times faster than the real Simulink model, making large-scale RL training tractable.

Both the real simulation and the surrogate are wrapped as standard Gymnasium environments. The observation vector presented to the agent contains 22 features: time elapsed, battery state of charge, velocity, position, motor power and current, net forces, acceleration, solar irradiance, and terrain grade including lookahead values at 5, 10, and 20 kilometers ahead. The action is a single continuous value, a target speed in meters per second bounded between 10 and 50. The PPO agent is trained entirely on the surrogate environment using four parallel rollout workers. The reward signal credits distance traveled per step, penalizes energy consumed, and applies a large terminal penalty if the battery reaches zero.

Current Status and Known Issues

The architecture is validated end-to-end. The Simulink simulation is functional and Python-callable with all physical constants integrated. The data collection pipeline, surrogate training pipeline, and PPO training pipeline are all complete and producing artifacts: a surrogate model file and a trained agent file that load correctly, produce actions in the expected range, and run a full simulated race day without errors. An inference verification test confirms the full chain from surrogate environment through observation vector through policy to speed command.

The baseline comparison framework, which evaluates the trained agent against constant-speed and rule-based alternatives, exposed several issues that need to be resolved before the system is reliable for race day. The most significant is that solar power is currently not modeled in the surrogate. The Simulink solar subsystem outputs zero power under all conditions, meaning both the surrogate and the trained agent operate in a world where the battery only discharges. In reality, solar input is what sustains the car across an eight-hour day. Without it, no policy can survive on a 5.22 kWh pack, and every agent in testing eventually ran the battery to zero. This is the primary issue blocking a valid retrain.

Several secondary issues compound this. The current reward function penalizes energy consumption but has no soft penalty for low state of charge, so the agent has no incentive to preserve battery margin until it is already too late. The surrogate was trained on randomly sampled positions along the course rather than consecutive full-race trajectories, which limits its fidelity to the sequences it will actually be

asked to simulate. Observation normalization scales are misaligned in places, pushing the policy toward out-of-distribution inputs. And the agent was trained for 500,000 timesteps, which is short for a continuous control task of this complexity, where 2 to 5 million timesteps is more typical for stable convergence.

Next Steps

The planned retrain is contingent on first fixing the solar subsystem in Simulink. Once solar input is confirmed working, the sequence is: revise the reward function to include a graduated low-SOC penalty, realign the surrogate's reset conditions with the real race-day starting state, recollect training data using full-race consecutive trajectories with solar active, fix normalization scales, and retrain both the surrogate and the PPO agent at higher timestep budget. The retrained agent will then be validated against the real Simulink environment before integration with live telemetry via Redis for race-day deployment.

The core technical contribution of this project is the surrogate-accelerated RL loop applied to a high-fidelity physics simulation of a real competition vehicle. Solar race strategy is a domain where the state space is continuous, the horizon is long, and the cost of a bad decision compounds over time in ways that make both human intuition and simple heuristics insufficient. The RL framing is a natural fit, and the surrogate architecture makes it computationally viable. The remaining work is correcting the physics and refining the learning signal so the agent is optimizing in a world that accurately reflects the actual race.

Built with Python, MATLAB/Simulink, PyTorch, Stable-Baselines3, and Gymnasium. Questions welcome at jschmolze@wisc.edu.