Autonomous Driving Using Reinforcement Learning

Miloš Tomić, Katarina Petrović and Đorđe Mijović

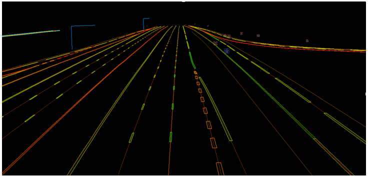
Autonomous Driving

Classical Methods in Driverless Cars

- Heavy reliance on **high definition maps** and **hand-crafted rules**
- Human interactions and rare events are unpredictable
- Difficulty **adapting** to new maps

Reinforcement Learning

- **Trains safely in simulation** on millions of scenarios, including rare and dangerous events
- **Adapts to new maps and environments** without manual reprogramming



Modeling Autonomous Driving as RL problem

In complex environments like city traffic:

- Actions are often **continuous** (steering angle, acceleration)
- The state space is **high-dimensional**
- Policies must be **stable and adaptive**

RL views driving as a **sequential decision-making problem**

- **Agent** = autonomous car
- **Environment** = road network, sensors, other agents

The goal is to learn a **policy** $\pi(a|s)$ that selects the best driving action in each situation.

Policy Gradient Methods

Policy Gradient methods directly learn $\pi_\theta(a|s)$ and work naturally with continuous actions.

- Goal: maximize the expected return $J(\theta)$ by adjusting θ in the direction of the gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t)\, \hat{A}_t\right]$$

- $\hat{A}_t$ is the **advantage estimate**, showing how much better the action was compared to the state value:

$$\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \left[r_{t+l} + \gamma V_\phi(s_{t+l+1}) - V_\phi(s_{t+l})\right]$$

- $V_\phi(s)$ is the critic's value estimate of state s.

Problem – large updates to θ can change the policy too much in one step, destabilizing learning.

PPO: Clipped Surrogate Objective and Full Loss

Idea: Control policy changes by clipping the probability ratio:

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

$$L^{\mathrm{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\hat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big]$$

Full PPO loss to minimize:

$$\mathcal{L}(\theta, \phi) = -L^{\mathrm{CLIP}}(\theta) + c_v\,\mathbb{E}_t[(V_\phi(s_t) - G_t)^2] - c_s\,\mathbb{E}_t[\mathcal{H}(\pi_\theta(\cdot|s_t))]$$

where:

- L^{CLIP} — clipped surrogate policy loss
- $(V_\phi(s_t) - G_t)^2$ — value function (critic) loss
- $\mathcal{H}$ — entropy bonus encouraging exploration
- $G_t = \hat{A}_t + V_\phi(s_t)$ — bootstrapped return

RL applied to Autonomous Driving Simulation

- **Environment:** Custom Manhattan grid (2×2 intersections) built on top of `highway-env` library
- **Agent:** Continuous-control PPO (acceleration, steering) trained to navigate from spawn to goal
- **Observation space:** Vehicle speed, heading, lane offset, next-waypoint bearing, distance to goal
- **Action space:** Continuous $a = [\text{acceleration}, \text{steering}] \in [-1, 1]^2$

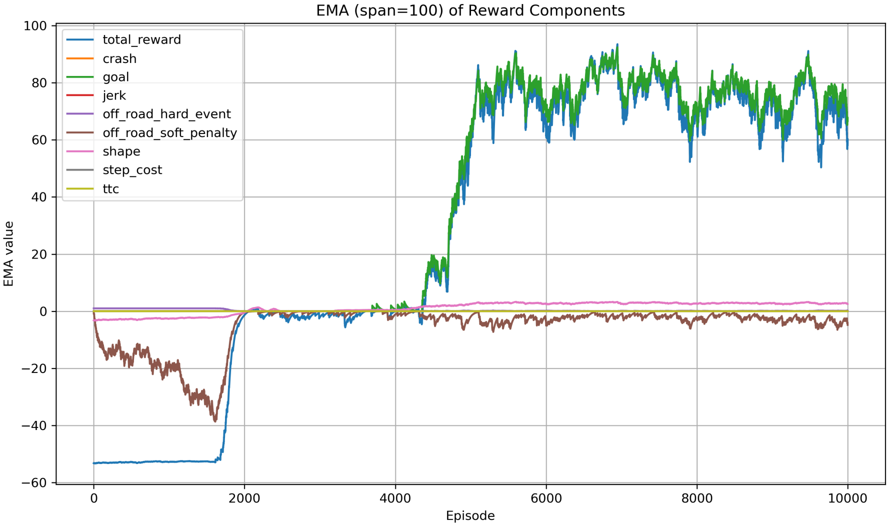
Reward Design

- **Shaping:** Combines distance to goal, lane centering, and heading alignment
- **Turn boost:** Extra shaping pressure near intersections to encourage committed turns
- **Terminal rewards:**
  - Large positive for reaching goal
  - Negative for crashes, going off-road, or timing out
- **Commit bonus:** Reward for choosing the optimal lane through a junction
- Small penalties for step cost and steering jerk

Training Setup and Metrics

- **Algorithm:** Proximal Policy Optimization
- **Policy network:** MLP with Gaussian action head (tanh-squashed)
- **Discount factor:** $\gamma = 0.99$
- **Metrics:**
  - Success rate (reaching goal)
  - Crash/off-road rate
  - Average episode length and cumulative reward

Training Metrics



Thanks for your attention!