Drive++
A Safe Autonomous Driving Algorithm

Pedro Castro [*] **Benedikt Kolbeinsson** [*] **Jiaze Sun** [*]
{p.castro18, benedikt.kolbeinsson15, j.sun19}@imperial.ac.uk

1 Introduction

We present a safe and efficient autonomous driving algorithm which performs significantly better than the baseline.

2 Methodology

Observations. At each step of the simulation, the SUMO environment provides egocentric vehicle information describing the current state of the target vehicle as well as additional local track information of neighbouring vehicles and future waypoints. A 2D RGB bird's eye view of the area around the target vehicle is provided. We discard it seeing as it provides no additional information. A list of used observations is available in Table 1.

Table 1: List of observations used by the agent.

Observation	Description
Speed	Normalized
Steering	Normalized
Signed Distance to Center	Normalized
Future Waypoint Headings	Sine of angled relative to current heading
Time-To-Collision	Extended to support 3 lanes, fixed distance calculation
Lane Distance	Extended to support 3 lanes, fixed distance calculation
Social Vehicle Headings	Sine of angled relative to current heading
Speed of Closest Vehicle	Closest vehicle in the current lane normalized by current speed
Proximity Map	Binary flags discretizing vehicle presence in the immediate vicinity

Action Space. In order to control the vehicle's speed, the environment API provides the agent access to the vehicle's throttle and breaking. However seeing as for efficient driving both are mutually exclusive, our agent is formulated to output a single action in the form of acceleration which is post-processed into separate throttle and breaking operations, as imposed by the environment. Similarly, the agent can control the vehicle's direction by modifying its steering angle. We design a residual steering method which limits the next steering angle to be within a range of 25°.

Rewards. In addition to the positive reward r_{env} provided in the baseline, we also incorporate four types of penalties in our reward function to ensure safety and minimize collisions.

- To avoid collisions, we impose the following **crash penalty** to prevent the egocentric vehicle from moving too close to other vehicles:

$$r_{crash} = -5 \cdot I\{\|p_{ego} - p_{closest}\|_2 < 6\},$$

  where I is the indicator function, and p_{ego} and $p_{closest}$ are the positions of the egocentric vehicle and the vehicle closest to it in its current lane, respectively.

- Our vehicle often learns to accelerate to very high speeds after a long training session, which inevitably causes it to go off-road or even flip during sharp turns. To prevent this issue, we impose the following **steering penalty**:

$$r_{steer} = -I\{s > 60\} \cdot \left(\frac{s-60}{20} \cdot \frac{a}{4}\right)^2,$$

[*] These three authors contributed equally.

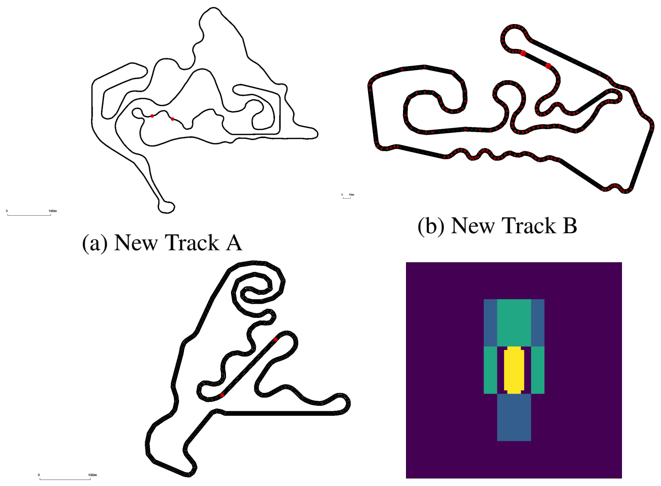
(a) New Track A (b) New Track B (c) New Track C (d) Proximity Map

Figure 1: Examples of new tracks and Proximity Map illustration.

  where s and a are the current speed and steering angle of the egocentric vehicle, respectively.

- The **center penalty** r_{center} helps keep our vehicle on the road by penalizing its absolute lateral distance to the center of its current lane.

- To prevent our vehicle from flipping at high speeds, we include a **flip penalty flag** in our **final reward function**:

$$r_{total} = r_{env} \cdot F + r_{crash} + r_{steer} + r_{center},$$

  where $F = 1$ if $s > 25$ and the height of the vehicle exceeds its normal threshold, and $F = -1$ otherwise.

Training. We used RLLib's **PPO** implementation. Our backbone consists of two 256 fully connected layers with ReLU activations except on the output layer. After intensive hyper-parameter search, we found 4096 mini-batch size with 10 SGD iterations and buffer of size 131072 to be optimal for training. We trained for 24M steps without the steering penalty for the agent to learn to drive quickly. Afterwards, we fine-tuned it for another 6M steps, with the addition of the steering penalty to prioritise safe and comfortable driving. For training, in addition to the provided public tracks, we created **new tracks** using SUMO's NETEDIT tool. Examples can be seen in Figures 1a, 1b and 1c.

3 Safety and comfort

As safety is the highest priority when designing an autonomous vehicle, we tailor our learning strategy in order to satisfy this requirement. Our final agent incorporates multiple novel safety features. (i) To ensure the vehicle's movement is **predicable**, we modified the model to be **deterministic**. (ii) By using residual steering, we are able to **reduce vehicle weaving** which is both inefficient and dangerous as well as uncomfortable for passengers. (iii) Using the future heading errors, our agent can optimally **decelerates in anticipation of future turn angles**. (iv) Our proximity detection allows our agent to be **aware of other obstacles in its surrounding**. (v) We ensemble two models, the main one being extremely safe and reliable while the second is a faster model deployed on tracks with low angle turns.