Paper Presentation:
Short-Term Trajectory Planning in TORCS using Deep Reinforcement Learning

Emilio Capo
emilio.capo@mail.polimi.it
Computer Science and Engineering (CSE)
Outline

Motivations

Problem and Solution Design

Experimental Setup

Results
Motivations
The Industry’s Need for Coherent AI

Believability of racing games

- **Physics**: High quality of simulation (aerodynamics, weather, collisions, …)
- **Graphics**: Aiming at photorealism
- **Real Pilots & Cars**

The problem of AI

- **Complexity of Simulation**: Developing an artificial agent is a hard task
- **Simplified Physics**: Using simplified physics models leads to incoherent behaviour
Racing AI: General Approach

Racing Line Computation

Target Sampling

Following Logic

Motivations
Problem and Solution Design
Experimental Setup
Results
Racing AI: General Approach

Complexity of track physics

Complexity of vehicle physics

Racing Line Computation

Target Sampling

Following Logic

Motivations

Problem and Solution Design

Experimental Setup

Results
Deep Reinforcement Learning

Reinforcement Learning
• Abstraction from the environment
• Data generated through interaction

Deep Learning
• Management of large input spaces
• Huge amount of data needed

Promising Approach
• Solved Complex Problems (Go, DOTA, …)
• Simplifies Development
Racing AI: General Approach

- Racing Line Computation
  - Complexity of track physics

- Target Sampling

- Following Logic
  - Complexity of vehicle physics

Motivations

Problem and Solution Design

Experimental Setup

Results
Racing AI: Our Approach

Motivations

Problem and Solution Design

Experimental Setup

Results
Problem and Solution

Design
The Open Racing Car Simulator (TORCS)

Open-Source Racing Simulator

- **Different Game Modes**: Practice, competition, etc.
- **Physics Engine**: Aerodynamics, traction, fuel, etc.

Client-Server Architecture

- **Server**: Wrapper providing numerical information to the client about the race (car, opponents, etc.)
- **Client**: Driving logic taking decisions based on the information received from the server
Reinforcement Learning Scheme

Critical Aspects

- **State Representation**: The information the agent can use to take decisions;
Reinforcement Learning Scheme

Critical Aspects

- **State Representation**: The information the agent can use to take decisions;

- **Action Space**: How the agent can interact with the environment;
Reinforcement Learning Scheme

Critical Aspects

- **State Representation**: The information the agent can use to take decisions;

- **Action Space**: How the agent can interact with the environment;

- **Reward Function**: How to inform the agent about the efficiency of the decisions taken.
State Representation

Numerical Representation

- **Telemetry information**: How the agent’s state is with respect to the environment
- **Internal information**: State of the agent itself
State Representation

Numerical Representation

- **Telemetry information**: How the agent’s state is with respect to the environment
- **Internal information**: State of the agent itself

![Numerical representation diagram](image-url)
State Representation

Numerical Representation

- **Telemetry information**: How the agent’s state is with respect to the environment
- **Internal information**: State of the agent itself

Hybrid Representation...

- **Image**: Telemetry information
- **Numerical**: Internal information
**State Representation**

**Numerical Representation**

- **Telemetry information**: How the agent’s state is with respect to the environment
- **Internal information**: State of the agent itself

**Hybrid Representation…**

- **Image**: Telemetry information
- **Numerical**: Internal information

… **With Racing Line Integration**

- **Racing Line**: White
- **Proximity To Racing Line**: Gray
Learning Algorithm: DDPG

Actor-Critic Method
- **Actor Network**: Learns the driving policy
- **Critic Network**: Learns actions’ profitability

Core Idea
- Update the **Actor** towards the best actions according to the **Critic**
- Generate new experiences from the **Actor** to update the **Critic**
Numerical Networks

Motivations

Problem and Solution Design

Experimental Setup

Results
Single Output

• Offset from track center: [-1, 1]

Two Outputs

• Offset from track center: [-1, 1]
• Target speed correction: [-1, 1]

The Lookahead value is computed by the following logic at each step.
Following Logic

**Lookahead Computation**

\[ \text{LookAhead} = \text{LookBase} + \text{LookScale} \times \text{currSpeed} \]
Looking ahead Computation

\[ \text{LookAhead} = \text{LookBase} + \text{LookScale} \times \text{currSpeed} \]

Forward Step

- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)
Following Logic

Lookahead Computation

\[ \text{LookAhead} = \text{LookBase} + \text{LookScale} \times \text{currSpeed} \]

Forward Step

- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)

Backward Step

- Correct Current Target Speed
Following Logic

Lookahead Computation

\[ \text{LookAhead} = \text{LookBase} + \text{LookScale} \times \text{currSpeed} \]

Forward Step

- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)

Backward Step

- Correct Current Target Speed

Heuristic

- Correct Current Target Speed according to proximity to the next corner
**Following Logic**

**Lookahead Computation**

\[ \text{LookAhead} = \text{LookBase} + \text{LookScale} \times \text{currSpeed} \]

**Forward Step**
- Compute Curvatures (Local and Target)
- Compute Maximum Target Speed (Local and Target)

**Backward Step**
- Correct Current Target Speed

**Heuristic**
- Correct Current Target Speed according to proximity to the next corner

**Agent Correction (Two-Outputs Agents)**

\[ \text{targSpeed} = \text{targSpeed} + \text{corrDelta} \times \text{speedCorr} \]
**Distance Raced**

- $P_{curr}$: Current car position
- $P_{prev}$: Previous car position

$\Delta distRaced = distRaced(P_{curr}) - distRaced(P_{prev})$
Reward Function

Distance Raced

- $P_{curr}$: Current car position
- $P_{prev}$: Previous car position

\[
\Delta distRaced = distRaced(P_{curr}) - distRaced(P_{prev})
\]

Complete Reward Function

- Colliding (walls or obstacles)
- Driving backwards
- Out of track

\[
r_t = \begin{cases} 
-100 & \text{if colliding or driving backwards} \\
-1 & \text{if out of track} \\
100 \cdot \Delta distRaced & \text{otherwise}
\end{cases}
\]
Experimental Setup
Fixed Time Budget

- Each track is given a time budget
- This defines the number of steps that can be spent on that track

1 Batch = 5 Tracks
Training

**Fixed Time Budget**
- Each track is given a time budget
- This defines the number of steps that can be spent on that track

**Uniform Experience**
- All tracks are given the same total number of steps
- Avoids bias towards easier tracks

---

Motivations | Problem and Solution Design | Experimental Setup | Results
Training

Fixed Time Budget
- Each track is given a time budget
- This defines the number of steps that can be spent on that track

Uniform Experience
- All tracks are given the same total number of steps
- Avoids bias towards easier tracks

Episode Termination
- Out of time budget
- Collision
- Driving backwards

Motivations
Problem and Solution Design
Experimental Setup
Results
Simple Gaussian Noise

- $\mu$: 0
- $\sigma$: 0.2

Update Rule

- $T_{\text{exp}}$: 3 batches
- $\alpha_{\text{max}}$: 1.0
- $\alpha_{\text{min}}$: 0.0

In a preliminary experiment, we also tried to apply Ornstein-Uhlenbeck noise and sine noise, but we found no relevant advantage.
Baselines

Randomly Initialized Networks
• Single-Output
• Two-Outputs

Low-Level Agents
• **Input**: Numerical/Hybrid
• **Output**: Acceleration/Brake/Steering
Baselines

Randomly Initialized Networks
• Single-Output
• Two-Outputs

Low-Level Agents
• Input: Numerical/Hybrid
• Output: Acceleration/Brake/Steering

SnakeOil
• Input: Numerical
• Rules: Fixed, Human-Designed
• Output: Low-Level

Autopia
• Input: Numerical
• Rules: Fuzzy, Human-Designed
• Output: Low-Level
Testing

**Metric of Interest**

- Distance raced in a fixed time
Testing

Metric of Interest
- Distance raced in a fixed time

Trained Agents (LL and HL)
- Uniformly sampled checkpoints
- The best checkpoint is used for testing

\[ p(c_i) = \text{mean}(d_{c_i,t_i \in T_{train}}) - 0.5 \cdot \text{std}(d_{c_i,t_i \in T_{train}}) \]
Testing

Metric of Interest
• Distance raced in a fixed time

Trained Agents (LL and HL)
• Uniformly sampled checkpoints
• The best checkpoint is used for testing

Episode Termination
• Out of time
• Collision
• Driving backwards

7 Checkpoints per Agent

\[ p(c_i) = \text{mean}(d_{c_i,t_i \in T_{train}}) - 0.5 \cdot \text{std}(d_{c_i,t_i \in T_{train}}) \]
Results
Single-Output Agents

Basics

- Improvement over random policy

Low-Level Comparison

- Improvement over LL-N (Mueda is the only exception)
- Completely overcomes LL-H

Bot Comparison

- Improvement over SnakeOil (performance and generalization)
- Suboptimal with respect to Autopia

<table>
<thead>
<tr>
<th>Bot</th>
<th>Alsoujk-Hill</th>
<th>Brondehech</th>
<th>Coldpeak</th>
<th>Citytrack</th>
<th>Emero-City</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-1R</td>
<td>6999.64</td>
<td>6817.48</td>
<td>6224.37</td>
<td>6463.54</td>
<td>7047.49</td>
</tr>
<tr>
<td>HL-2R</td>
<td>6207.31</td>
<td>5703.68</td>
<td>5326.8</td>
<td>6297.34</td>
<td>6171.61</td>
</tr>
<tr>
<td>Autopia</td>
<td>11481.6</td>
<td>11593.0</td>
<td>11181.7</td>
<td>13597.8</td>
<td>13172.2</td>
</tr>
<tr>
<td>Snake-Oil</td>
<td>6957.07</td>
<td>739.192</td>
<td>6930.21</td>
<td>6972.4</td>
<td>6987.16</td>
</tr>
<tr>
<td>LL-N</td>
<td>1112.28</td>
<td>705.247</td>
<td>1495.89</td>
<td>7714.61</td>
<td>7945.07</td>
</tr>
<tr>
<td>LL-H</td>
<td>127.831</td>
<td>192.635</td>
<td>270.652</td>
<td>338.413</td>
<td>215.725</td>
</tr>
<tr>
<td>HL-N1</td>
<td>9411.82</td>
<td>9310.71</td>
<td>9999.21</td>
<td>11238.2</td>
<td>9229.33</td>
</tr>
<tr>
<td>HL-H1</td>
<td>9413.05</td>
<td>9313.14</td>
<td>10002.4</td>
<td>11238.3</td>
<td>9230.11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bot</th>
<th>Mueda-City</th>
<th>Petit</th>
<th>Ustka-City</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-1R</td>
<td>6965.49</td>
<td>8110.44</td>
<td>6280.97</td>
</tr>
<tr>
<td>HL-2R</td>
<td>6751.92</td>
<td>5599.0</td>
<td>5910.49</td>
</tr>
<tr>
<td>Autopia</td>
<td>13354.1</td>
<td>11513.0</td>
<td>12689.5</td>
</tr>
<tr>
<td>Snake-Oil</td>
<td>6998.45</td>
<td>2158.2</td>
<td>6946.44</td>
</tr>
<tr>
<td>LL-N</td>
<td>9184.19</td>
<td>55.3047</td>
<td>7770.07</td>
</tr>
<tr>
<td>LL-H</td>
<td>218.762</td>
<td>109.677</td>
<td>9.82495</td>
</tr>
<tr>
<td>HL-N1</td>
<td>9041.42</td>
<td>9728.49</td>
<td>10402.0</td>
</tr>
<tr>
<td>HL-H1</td>
<td>9044.97</td>
<td>9734.09</td>
<td>10403.9</td>
</tr>
</tbody>
</table>
Motivations

Problem and Solution Design

Experimental Setup

Results

Two-Outputs Agents

Basics

• Improvement over random policy

Low-Level Comparison

• Improvement over LL-N (completely)
• Completely overcomes LL-H

Bot Comparison

• Improvement over SnakeOil (performance and generalization)
• Suboptimal with respect to Autopia

Single Output

• Slight improvement

<table>
<thead>
<tr>
<th>Bot</th>
<th>Alsoujar-Hill</th>
<th>Brondehach</th>
<th>Coldpeak</th>
<th>Citytrack</th>
<th>Emero-City</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-1R</td>
<td>6999.64</td>
<td>6817.48</td>
<td>6224.37</td>
<td>6463.54</td>
<td>7047.49</td>
</tr>
<tr>
<td>HL-2R</td>
<td>6207.31</td>
<td>5703.68</td>
<td>5326.8</td>
<td>6297.34</td>
<td>6171.61</td>
</tr>
<tr>
<td>Autopia</td>
<td>11481.6</td>
<td>11593.0</td>
<td>11181.7</td>
<td>13597.8</td>
<td>13172.2</td>
</tr>
<tr>
<td>Snake-Oil</td>
<td>6957.07</td>
<td>739.192</td>
<td>6930.21</td>
<td>6972.4</td>
<td>6987.16</td>
</tr>
<tr>
<td>LL-N</td>
<td>1112.28</td>
<td>705.247</td>
<td>1495.89</td>
<td>7714.61</td>
<td>7945.07</td>
</tr>
<tr>
<td>LL-H</td>
<td>127.831</td>
<td>192.635</td>
<td>270.652</td>
<td>338.413</td>
<td>215.725</td>
</tr>
<tr>
<td>HL-N2</td>
<td>9486.19</td>
<td>9382.04</td>
<td>10185.0</td>
<td>11398.6</td>
<td>9314.12</td>
</tr>
<tr>
<td>HL-H2</td>
<td>9411.82</td>
<td>9310.71</td>
<td>9999.21</td>
<td>11238.2</td>
<td>9229.33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bot</th>
<th>Mueda-City</th>
<th>Petit</th>
<th>Ustka-City</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-1R</td>
<td>6965.49</td>
<td>8110.44</td>
<td>6280.97</td>
</tr>
<tr>
<td>HL-2R</td>
<td>6751.92</td>
<td>5599.0</td>
<td>5910.49</td>
</tr>
<tr>
<td>Autopia</td>
<td>13354.1</td>
<td>11513.0</td>
<td>12689.5</td>
</tr>
<tr>
<td>Snake-Oil</td>
<td>6998.45</td>
<td>2158.2</td>
<td>6946.44</td>
</tr>
<tr>
<td>LL-N</td>
<td>9184.19</td>
<td>55.3047</td>
<td>7770.07</td>
</tr>
<tr>
<td>LL-H</td>
<td>218.762</td>
<td>109.677</td>
<td>9.82495</td>
</tr>
<tr>
<td>HL-N2</td>
<td>9226.99</td>
<td>9829.32</td>
<td>10466.4</td>
</tr>
<tr>
<td>HL-H2</td>
<td>9041.42</td>
<td>9728.49</td>
<td>10402.0</td>
</tr>
</tbody>
</table>
Two-Outputs + Racing Line Agent

Basics
- Improvement over random policy

Low-Level Comparison
- Improvement over LL-N (completely)
- Completely overcomes LL-H

Bot Comparison
- Improvement over SnakeOil (performance and generalization)
- Suboptimal with respect to Autopia

Two-Outputs Without Racing Line (HL-H2)
- Slight improvement

<table>
<thead>
<tr>
<th>Bot</th>
<th>Alsoujk-Hill</th>
<th>Brondehach</th>
<th>Coldpeak</th>
<th>Citytrack</th>
<th>Emero-City</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-1R</td>
<td>6999.64</td>
<td>6817.48</td>
<td>6224.37</td>
<td>6463.54</td>
<td>7047.49</td>
</tr>
<tr>
<td>HL-2R</td>
<td>6207.31</td>
<td>5703.68</td>
<td>5326.8</td>
<td>6297.34</td>
<td>6171.61</td>
</tr>
<tr>
<td>Autopia</td>
<td>11481.6</td>
<td>11593.0</td>
<td>11181.7</td>
<td>13597.8</td>
<td>13172.2</td>
</tr>
<tr>
<td>Snake-Oil</td>
<td>6957.07</td>
<td>739.192</td>
<td>6930.21</td>
<td>6972.4</td>
<td>6987.16</td>
</tr>
<tr>
<td>LL-N</td>
<td>1112.28</td>
<td>705.247</td>
<td>1495.89</td>
<td>7714.61</td>
<td>7945.07</td>
</tr>
<tr>
<td>LL-H</td>
<td>127.831</td>
<td>192.635</td>
<td>270.652</td>
<td>338.413</td>
<td>215.725</td>
</tr>
<tr>
<td>HLR</td>
<td>9486.19</td>
<td>9382.04</td>
<td>10185.0</td>
<td>11398.6</td>
<td>9314.12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bot</th>
<th>Mueda-City</th>
<th>Petit</th>
<th>Ustka-City</th>
</tr>
</thead>
<tbody>
<tr>
<td>HL-1R</td>
<td>6965.49</td>
<td>8110.44</td>
<td>6280.97</td>
</tr>
<tr>
<td>HL-2R</td>
<td>6751.92</td>
<td>5599.0</td>
<td>5910.49</td>
</tr>
<tr>
<td>Autopia</td>
<td>13354.1</td>
<td>11513.0</td>
<td>12689.5</td>
</tr>
<tr>
<td>Snake-Oil</td>
<td>6998.45</td>
<td>2158.2</td>
<td>6946.44</td>
</tr>
<tr>
<td>LL-N</td>
<td>9184.19</td>
<td>55.3047</td>
<td>7770.07</td>
</tr>
<tr>
<td>LL-H</td>
<td>218.762</td>
<td>109.677</td>
<td>9.82495</td>
</tr>
<tr>
<td>HLR</td>
<td>9226.99</td>
<td>9829.32</td>
<td>10466.4</td>
</tr>
</tbody>
</table>
Examples of Racing Lines

Following Simplix’s Racing Line

Following Learned Racing Line
Future Works

**More target points**
- A single target point is limiting
- More points allow to build a better racing line approximation

**Richer input space**
- Enlarge the portion of the track visible to the agent
- This allows for a better planning

**Exploration of algorithms**
- Perform accurate hyperparameter tuning
- Explore other algorithms (TRPO, PPO, …)

**Exploration of reward functions**
- Consider embedding racing line information in the reward function
- Learning a general behaviour from specific racing lines
Emilio Capo
emilio.capo@mail.polimi.it
Computer Science and Engineering (CSE)