Artificial Intelligence in Racing Games

Videogame Design and Programming
Racing AI in a nutshell

- Strategic System
- Tactical System
- Control System
Control System
Overview

- Control system takes a control action on the basis of
  - current status of the vehicle
  - target position and speed
- Based on car and environment dynamics
- Might involve heuristics or approximations
What about the target?
Understanding the problem...

\[ v_{\text{max}} \approx \sqrt{\mu gr} \]
Understanding the problem...

Shortest path or minimum curvature?
How to find the optimal racing line?

- Expert Design
- Heuristics
- Model-Based Optimization
- Genetic Algorithms
Expert design + Test
Heuristics

Greedy Search
Model Based(1)

\[
\begin{align*}
\frac{dS_a}{dt} &= S_a\left(\frac{2f}{5} + S_{uv} + S_{uv} + S_{ww} + S_{dd} + S_{ddw} + S_{ddw}\right) \\
\frac{dS_{uv}}{dt} &= S_{uv}\left(\frac{2f}{5} + S_{uv} + S_{ww} + S_{dd} + S_{ddw} + S_{ddw}\right) - S_{uv}\left(\frac{4f}{5} + S_{uv} + S_{dd} + \frac{2}{3}(S_{uv} + S_{dd} + S_{ddw} + S_{ddw})\right) \\
\frac{dS_{ww}}{dt} &= S_{ww}\left(\frac{2f}{5} + S_{uv} + S_{ddw} + S_{dd} + S_{ddw}\right) - S_{ww}\left(\frac{3f}{5} + S_{uv} + S_{dd} + S_{ddw} + S_{ddw}\right) + S_{ww}\left(\frac{3}{5} + \frac{1}{2}(S_{uv} + S_{dd} + S_{ddw} + S_{ddw}) + \frac{2}{3}(S_{uv} + S_{dd} + S_{ddw} + S_{ddw})\right) \\
\frac{dS_{dd}}{dt} &= S_{dd}\left(\frac{2f}{5} + \frac{2}{3}(S_{uv} + S_{dd} + S_{ddw} + S_{ddw})\right) \\
\frac{dS_{ddw}}{dt} &= S_{ddw}\left(\frac{2f}{5} + \frac{1}{2}(S_{uv} + S_{ddw})\right) \\
\frac{dS_{ddw}}{dt} &= S_{ddw}\left(\frac{2f}{5} + \frac{2}{3}(S_{uv} + S_{dd} + S_{ddw} + S_{ddw})\right)
\end{align*}
\]

Driver Model & Car Dynamics

\[
\begin{align*}
\min_{\vec{\alpha}} & \quad \vec{\alpha}^T H_S \vec{\alpha} + \vec{B}_S \vec{\alpha} \\
& \quad 0 \leq \alpha_i \leq 1
\end{align*}
\]

\[
\begin{align*}
\min_{\vec{\alpha}} & \quad \vec{\alpha}^T H_{\Gamma} \vec{\alpha} + \vec{B}_{\Gamma} \vec{\alpha} \\
& \quad 0 \leq \alpha_i \leq 1
\end{align*}
\]

Grid search of the best convex combination of SP and MCP
Controllers in racing games are not ideal: models can lead to suboptimal performance.

It might be difficult to deal with any detail of the tracks:
- different type of borders (curbs, barriers, sand, grass)
- bumps and banking
- different friction

One optimal trade-off between MCP and SP on the whole track?
How to extend it?

Search for the best trade-off in *each* segment of the track

Replace models with the actual racing simulator

Replace the grid search with a GA
Genetic Algorithms (1)

- Too many variables!
- Does not exploit any domain information (i.e., SP and MCP)

\[ \alpha_1, \alpha_2, \ldots, \alpha_n \]

NO!
Genetic Algorithms (2)

- Few variables (up to 30-40 in the most complex tracks)
- Exploits the knowledge of SP and MCP
- Continuous by design

Diagram showing tracks labeled with ε₁, ε₂, ..., ε₉.
## Genetic Algorithms (3)

- **Results achieved in a case study:**

<table>
<thead>
<tr>
<th>Track</th>
<th>GA</th>
<th>Model-Based</th>
<th>Heuristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aalborg</td>
<td>69.928</td>
<td>+0.766</td>
<td>+0.834</td>
</tr>
<tr>
<td>Alpine 1</td>
<td>121.481</td>
<td>+1.063</td>
<td>+1.395</td>
</tr>
<tr>
<td>Alpine 2</td>
<td>92.527</td>
<td>+0.549</td>
<td>+1.807</td>
</tr>
<tr>
<td>A-Speedway</td>
<td>24.701</td>
<td>+0.437</td>
<td>+2.857</td>
</tr>
<tr>
<td>Forza</td>
<td>85.210</td>
<td>+0.476</td>
<td>+1.398</td>
</tr>
<tr>
<td>CG-Speedway</td>
<td>39.372</td>
<td>+0.422</td>
<td>+0.748</td>
</tr>
<tr>
<td>Michigan-Speedway</td>
<td>33.866</td>
<td>+0.024</td>
<td>-0.124</td>
</tr>
<tr>
<td>Olethros Road</td>
<td>111.656</td>
<td>+1.270</td>
<td>+2.974</td>
</tr>
<tr>
<td>Ruudskogen</td>
<td>62.732</td>
<td>+0.476</td>
<td>+0.474</td>
</tr>
<tr>
<td>Street 1</td>
<td>75.613</td>
<td>+0.511</td>
<td>-0.933</td>
</tr>
<tr>
<td>Wheel 1</td>
<td>74.887</td>
<td>+0.519</td>
<td>-0.963</td>
</tr>
</tbody>
</table>
How control system uses the racing line?
$v_{max} \approx \sqrt{\mu gr}$

$v_{max} = 110 \text{ Km/h}$

$v_{max} = 200 \text{ Km/h}$
\[ v_{\text{max}} \approx \sqrt{\mu gr} \]

\[ v^* = 110 \text{ Km/h} \]

\[ v^* = 130 \text{ Km/h} \]
Following the racing line

- Control system follows a racing line provided in input
- It is usually programmed based on the following domain knowledge:
  - Car parameters (e.g., engine power, brakes efficiency)
  - Environment parameters (e.g., friction of the asphalt)
  - In-game dynamics (e.g., aerodynamics)
- It is generally fine tuned to guarantee an optimal behaviour
Tactical System
Tactical system

- Performs complex maneuvers
  - Follows a preceding vehicle taking its slipstream
  - Overtakes when appropriate
  - Blocks following vehicles
- Handles specific situations
  - Avoids imminent collisions
  - Recovers the vehicle if it gets stuck against a border
Behavior Decomposition

Level 0
Effectors
- Accelerator
- Brake
- Gear
- Steer
- Repair
- Refuel

Level 1
Control
- ABS
- Speed
- Track
- Position

Level 2
Planning
- Brake Delay
- Trajectory
- Obstacle Avoidance
- Racing Line

Level 3
Strategy
- Overtaking Strategy
- Navigation
- Start Strategy
- Pit-Stop Strategy
How to design complex behaviors?

Programmed Heuristics

Domain Expert Rules

Learning
Examples of Heuristics

- Alternative racing lines for overtaking

- Programmed recovery policy
Learning driving behaviors
Learnig Overtaking Behavior: Problem Definition

- **State Space**
  - Frontal distance from the opponent car
  - Lateral distance from the opponent car
  - Distance from the side of the track
  - Speed difference

- **Action**
  - move 1m on right
  - keep current trajectory
  - move 1m on left

- **Reward**
  - +1 overtake completed
  - -1 collision or out of track
  - 0 otherwise

![Diagram of overtaking behavior](image)
Learning Brake Delay: Problem Definition

- **State Space**
  - Frontal distance from the opponent car
  - Distance from the next turn
  - Speed difference

- **Action**
  - Do **not** brake

- **Reward**
  - +1 overtake completed
  - -1 collision or out of track
  - 0 otherwise

- **Works on top of the driving policy**
Strategic system

- Balance AI skills
  - Determines how fast an AI should race depending to difficulty level, pilot skills, etc.
  - Forces mistakes at a realistic rate
- Handles resources in high simulative titles
  - Manages fuel consumption, tyre wear and damages
  - Chooses when go to pit
Examples of Strategic System

- Rubber band
  - Skill of vehicles behind the player is increased
  - Skill of vehicles ahead of the player is reduced
  - Too simple: has some drawbacks

- Scripted strategy
  - Much more customizable by designers
  - Allows different and, possibly, more realistic strategies
  - Offers more opportunities for research
How to get started?
Simulated Car Racing

- Simulated Car Racing is a **scientific** competition based on The Open Racing Car Simulator (TORCS)
- Competitors are provided with
  - a simple API (Java and C++) to build their own controller
  - a complete sensors/effectors model
- Goal of the competition is developing the fastest controller
- Competition software is open source and is a good starting point to **learn programming a racing AI**

http://cig.ws.dei.polimi.it/?page_id=134

http://groups.google.com/group/racingcompetition
The Open Racing Car Simulator

- TORCS is a state of the art open source simulator written in C++

- Main features
  - Sophisticated dynamics
  - Provided with several cars, tracks, and controllers
  - Active community of users and developers
  - Easy to develop your own controller

- OS Support
  - Linux: binaries and building from sources
  - Windows: binaries and “limited” building from sources support
  - OSX: legacy binaries and no building from sources support 😔
Software Overview

- To make TORCS more easy to use we developed an API based on socket (UDP)
- Values of sensors and effectors are sent through UDP
- 3 components
  - Torcs Patch
  - Server Bot (C++)
  - Client API (C++ and Java)
Main Sensors

- Rangefinders for...
  - ...edges of the track
  - ...opponents
- Speed, RPM, fuel, damage, angle with track, distance race, position on track, etc.
### Sensors (1)

<table>
<thead>
<tr>
<th>Name</th>
<th>Range (unit)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>angle</td>
<td>([-\pi, +\pi]) (rad)</td>
<td>Angle between the car direction and the direction of the track axis.</td>
</tr>
<tr>
<td>curLapTime</td>
<td>([0, +\infty)) (s)</td>
<td>Time elapsed during current lap.</td>
</tr>
<tr>
<td>damage</td>
<td>([0, +\infty)) (point)</td>
<td>Current damage of the car (the higher is the value the higher is the damage).</td>
</tr>
<tr>
<td>distFromStart</td>
<td>([0, +\infty)) (m)</td>
<td>Distance of the car from the start line along the track line.</td>
</tr>
<tr>
<td>distRaced</td>
<td>([0, +\infty)) (m)</td>
<td>Distance covered by the car from the beginning of the race.</td>
</tr>
<tr>
<td>focus</td>
<td>([0,200]) (m)</td>
<td>Vector of 5 range finder sensors: each sensors returns the distance between the track edge and the car within a range of 200 meters. Sensor are affected by i.i.d. normal noises with a standard deviation equal to the 1% of sensors range. The sensors sample, every degree, a five degree space along a specific direction provided by the client (the direction is defined with the focus command and must be in the range ([-\pi/2, +\pi/2]) w.r.t. the car axis). Focus sensors are not always available: they can be used only once per second of simulated time. When the car is outside of the track (i.e., pos is less than -1 or greater than 1), the focus direction is outside the allowed range ([-\pi/2, +\pi/2])) or the sensors has been already used once in the last second, the returned values are not reliable (typically -1 is returned).</td>
</tr>
<tr>
<td>fuel</td>
<td>([0, +\infty)) (l)</td>
<td>Current fuel level.</td>
</tr>
<tr>
<td>gear</td>
<td>${-1,0,1,\cdots,7}$</td>
<td>Current gear: -1 is reverse, 0 is neutral and the gear from 1 to 7.</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>lastLapTime</td>
<td>$[0, +\infty)$ (s)</td>
<td>Time to complete the last lap</td>
</tr>
<tr>
<td>opponents</td>
<td>$[0, 200]$ (m)</td>
<td>Vector of 36 opponent sensors: each sensor covers a span of $\pi/18$ (10 degrees) within a range of 200 meters and returns the distance of the closest opponent in the covered area. Sensor are affected by i.i.d. normal noises with a standard deviation equal to the 5% of sensors range. The 36 sensors covers all the space around the car, spanning clockwise from $+\pi$ up to $-\pi$ with respect to the car axis.</td>
</tr>
<tr>
<td>racePos</td>
<td>${1,2,\cdots,N}$</td>
<td>Position in the race with to respect to other cars.</td>
</tr>
<tr>
<td>rpm</td>
<td>$[2000, 7000]$ (rpm)</td>
<td>Number of rotation per minute of the car engine.</td>
</tr>
<tr>
<td>speedX</td>
<td>$(-\infty, +\infty)$ (km/h)</td>
<td>Speed of the car along the longitudinal axis of the car.</td>
</tr>
<tr>
<td>speedY</td>
<td>$(-\infty, +\infty)$ (km/h)</td>
<td>Speed of the car along the transverse axis of the car.</td>
</tr>
<tr>
<td>speedZ</td>
<td>$(-\infty, +\infty)$ (km/h)</td>
<td>Speed of the car along the Z axis of the car.</td>
</tr>
<tr>
<td>sensors</td>
<td>description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>track</td>
<td>Vector of 19 range finder sensors: each sensor returns the distance between the track edge and the car within a range of 200 meters. Sensors are affected by i.i.d. normal noises with a standard deviation equal to the 5% of sensors range. The sensors sample the space in front of the car every 10 degrees, spanning clockwise from $+\pi/2$ up to $-\pi/2$ with respect to the car axis. When the car is outside of the track (i.e., pos is less than -1 or greater than 1), the returned values are not reliable.</td>
<td></td>
</tr>
<tr>
<td>trackPos</td>
<td>Distance between the car and the track axis. The value is normalized w.r.t. to the track width: it is 0 when the car is on the axis, -1 when the car is on the right edge of the track and +1 when it is on the left edge of the car. Values greater than 1 or smaller than -1 means that the car is outside of the track.</td>
<td></td>
</tr>
<tr>
<td>wheelSpinVel</td>
<td>Vector of 4 sensors representing the rotation speed of wheels.</td>
<td></td>
</tr>
<tr>
<td>z</td>
<td>Distance of the car mass center from the surface of the track along the Z axis.</td>
<td></td>
</tr>
</tbody>
</table>
Main Effectors

- Basically 4 main effectors
  - Steering wheel [-1, +1]
  - Gas pedal [0, +1]
  - Brake pedal [0, +1]
  - Gearbox {-1, 0, 1, 2, 3, 4, 5, 6, 7}
### Effectors

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>accel</td>
<td>[0,1]</td>
<td>Virtual gas pedal (0 means no gas, 1 full gas).</td>
</tr>
<tr>
<td>brake</td>
<td>[0,1]</td>
<td>Virtual brake pedal (0 means no brake, 1 full brake).</td>
</tr>
<tr>
<td>clutch</td>
<td>[0,1]</td>
<td>Virtual clutch pedal (0 means no clutch, 1 full clutch).</td>
</tr>
<tr>
<td>gear</td>
<td>-1,0,1, ⋯, 7</td>
<td>Gear value.</td>
</tr>
<tr>
<td>steering</td>
<td>[-1,1]</td>
<td>Steering value: -1 and +1 means respectively full right and left, that corresponds to an angle of 0.785398 rad.</td>
</tr>
<tr>
<td>focus</td>
<td>[-90,90]</td>
<td>Focus direction (see the <em>focus</em> sensors in Table 1) in degrees.</td>
</tr>
<tr>
<td>meta</td>
<td>0,1</td>
<td>This is meta-control command: 0 do nothing, 1 ask competition server to restart the race.</td>
</tr>
</tbody>
</table>