Euler Agent Technical Datasheet - Autonomous Driving in Super Tux Kart Using Reinforcement Learning

Authored by: Wilson Cedric GENEVIEVE
Safa GUNES
Mahmoud DARWISH
Badr BENHAMMOU

Bachelor in Computer Science, Sorbonne University, Paris, France
January 2025 - May 2025

Supervised by: Olivier SIGAUD
Computer Science Professor and Machine Learning Researcher
ISIR Robotics Laboratory, Sorbonne University

Abstract. Our EulerAgent leverages curvature-based control techniques inspired by the Euler Spiral method to maximize speed and precision in SuperTuxKart races. By dynamically adjusting steering in response to track curvature, the agent optimally balances acceleration and turning for enhanced racing performance.

1 Introduction

The EulerAgent is designed to navigate racing tracks in SuperTuxKart by exploiting the geometric properties of Euler spirals. This method aims to improve the agent's ability to maintain high speed on straight sections while efficiently reducing speed in sharp turns. Unlike the MedianAgent, which relies on selecting future nodes for navigation, the EulerAgent emphasizes curvature estimation to enhance turn handling.

Inspired by optimal racing line strategies used in motorsport, our agent combines curvature-based steering with adaptive acceleration techniques to achieve optimal results.

2 Motivation

The development of the EulerAgent was inspired by research conducted on optimal racing lines, particularly the work described in *Racing Line Optimization* by Ying Xiong at MIT [1]. This research highlights the importance of minimizing curvature to enhance racing efficiency. The Euler spiral, known for its curvature continuity and smooth transitions, is well-suited for approximating optimal racing lines, especially in tight cornering scenarios. This method reduces aggressive steering adjustments and enhances vehicle stability at high speeds.

Our goal was to build an agent that mimics these principles in the context of SuperTuxKart racing tracks. By focusing on curvature-based steering and acceleration control, the EulerAgent achieves improved stability and smoother cornering.

3 The Euler Spiral Concept

The Euler spiral, also known as the clothoid or Cornu spiral, is a curve whose curvature changes linearly with its arc length. This property makes it particularly suitable for designing transition curves in roadways, railways, and racetracks.

In mathematical terms, the Euler spiral satisfies the following condition:

$$k(s) = a \cdot s \tag{1}$$

where $k(s)$ is the curvature at arc length s, and a is a constant parameter controlling the rate of curvature change.

The Euler spiral smoothly connects a straight path to a circular path by gradually increasing curvature. The Fresnel integrals describe the parametric form of the Euler spiral as follows:

$$x(s) = \int_0^s \cos\left(\frac{\theta}{2}s^2\right) ds, \quad y(s) = \int_0^s \sin\left(\frac{\theta}{2}s^2\right) ds \tag{2}$$

Additional equations for calculating key racing metrics are as follows:

$$R = \frac{1}{k(s)} \tag{3}$$

where R is the radius of curvature, inversely related to the curvature.

For optimal turn-in points and apex calculation in the late-apex strategy:

$$Apex_{optimal} = \frac{L}{2} + \frac{R_{min}}{\tan(\theta/2)} \tag{4}$$

where L is the track width and R_{min} is the minimum turning radius required for cornering.

To optimize the steering angle (δ), we use:

$$\delta = \arctan\left(\frac{L}{R}\right) \tag{5}$$

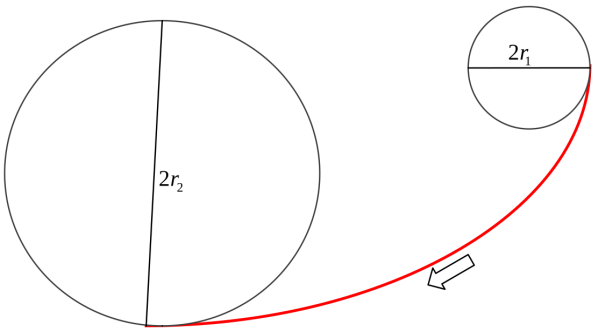

Figure 1: Euler Spiral transition curve representation. The curve smoothly transitions from a straight line into a circular path by gradually adjusting curvature.[1]

4 Optimal Cornering with the Euler Spiral

One of the key applications of the Euler spiral in racing is optimizing cornering strategies. A particularly effective technique involves using a *late-apex racing line*, which allows vehicles to carry higher speeds through turns by delaying the apex point.

This technique is effective for minimizing cornering time and maximizing straight-line speed after the turn. By incorporating curvature analysis inspired by the Euler spiral, the EulerAgent dynamically adjusts its trajectory to mimic this optimal racing behaviour.

5 Experimental Results and Discussion

Our EulerAgent demonstrated some improvements in completing race circuits by employing optimized curvature control based on Euler Spiral principles. The integration of dynamic steering adjustments and adaptive acceleration helped improve the agent's overall stability. Additionally,

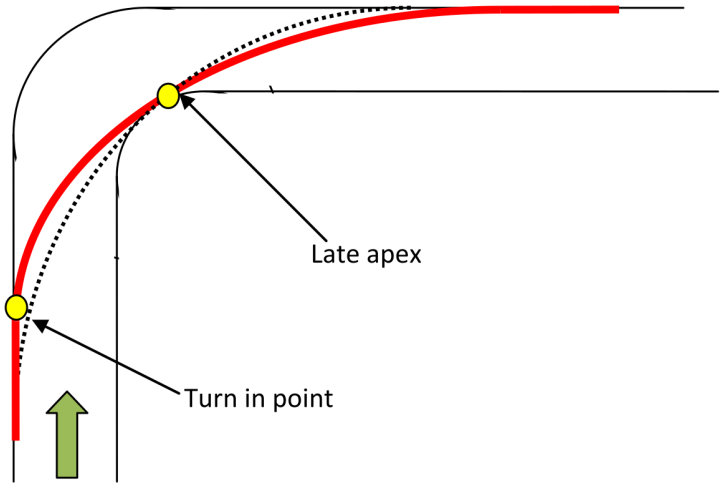

Figure 2: A late-apex racing line shown by the red path. The two yellow round points are the turn in point and the late apex, respectively. To achieve a late apex there should be a late turn in point. The dot line is the center-apex racing line.The turn-in point is delayed to allow a later apex, improving exit speed.[1]

the agent showed better control when managing sharp turns, particularly by adjusting its trajectory smoothly.

However, when comparing the EulerAgent with the MedianAgent, no significant improvement was observed in overall performance. While the EulerAgent showed marginal gains in certain scenarios, such as avoiding kart blockage in a specific turn in the track "black_forest," these improvements were minimal.

The current state of the EulerAgent can be considered a work in progress, and additional testing and parameter tuning are ongoing to enhance its efficiency.

Algorithm 1 EulerSpiralCurvature Algorithm

Input: List of path nodes
Output: Curvature value
if length of path nodes < 3 **then**
    return 0
end if
Select $p1 = path\_ends[0]$, $p2 = path\_ends[\text{midpoint}]$, $p3 = path\_ends[-1]$
Compute vector $v1 = p2 - p1$
Compute vector $v2 = p3 - p2$
Compute $angle = arctan2(cross(v1, v2), dot(v1, v2))$
Compute $distance = ||v1|| + ||v2||$
Compute initial curvature: $curvature = |angle|/distance$
Adjust curvature scaling: $curvature = curvature \times 2.0$
Ensure curvature limits: $curvature = \max(0.01, \min(curvature, 1.0))$
return $curvature$

Algorithm 2 CalculateAction Algorithm (NewEulerAgent)

Input: Observation data, Lookahead value
Output: Action tuple (steering, acceleration, drift, nitro)
if Observation data is None **then**
    return $\{\}$
end if
Extract path endpoint: $path\_end = obs["paths\_end"][lookahead - 1]$
Extract kart front position: $kart\_front = obs["front"]$
Extract path nodes: $path\_ends = obs["paths\_end"][: lookahead]$
Compute curvature: $curvature = euler\_spiral\_curvature(path\_ends)$
Compute slope: $slope = compute\_slope(path\_ends[: 2])$
if $curvature$ is an array **then**
    $curvature = \max(curvature)$
end if
if $slope$ is an array **then**
    $slope = \max(slope)$
end if
Compute acceleration:

$$acceleration = \max(0.5, 1 - |curvature| + \max(0, slope))$$

Compute direction to target: $direction\_to\_target = path\_end - kart\_front$
Compute steering:

$$steering = 0.4 \times direction\_to\_target[0]/(1.0 + |curvature| \times 0.5)$$

if $|curvature| < 0.05$ **then**
    Activate Nitro
else
    Deactivate Nitro
end if
Track agent position:

$$agent\_abs\_pos = env.unwrapped.world.karts[0].location$$

Append agent position to position list:

$$agent\_positions.append(agent\_abs\_pos)$$

return {acceleration, steering, drift, nitro}

References

[1] Ying Xiong. Racing line optimization. Master's thesis, Massachusetts Institute of Technology (MIT), 2010. Available at https://dspace.mit.edu/handle/1721.1/64669?show=full.