Median 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 MedianAgent navigates SuperTuxKart tracks using curvature control, adaptive acceleration, and path-following methods to balance stability and speed in complex racing environments.

1 Introduction

The primary objective of our MedianAgent is to stay in the middle of the track while maintaining optimal speed through various track configurations. Unlike conventional racing agents, the MedianAgent relies on the *n-th node* path-following technique to determine its position on the track. This approach involves selecting a future node ahead of the kart's current position as a target for movement.

Initially, we attempted a simpler strategy of selecting the immediate next node as our target. However, we discovered that using an *n-th* node (a node farther ahead) improved steering precision and reduced reaction delays. The following diagram illustrates this improved approach:

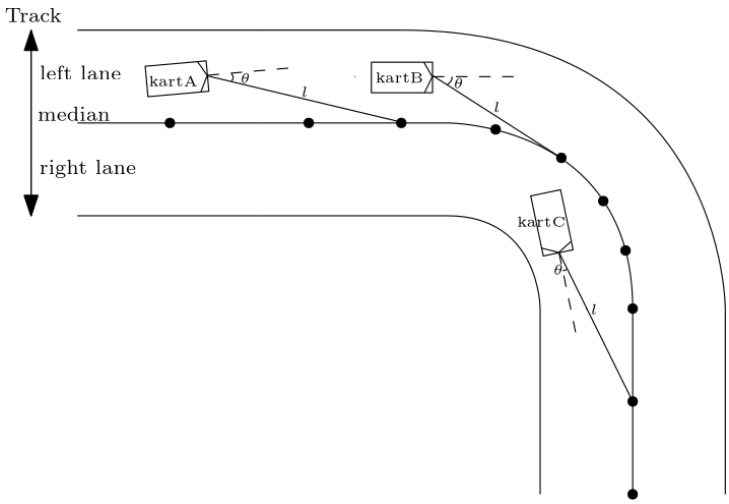

Figure 1: n-th node path-following approach

This represents the most basic implementation of a custom agent, with significant room for improvement, particularly in smoother turns and more efficient trajectory planning.

The challenge with this approach lies in accurately adjusting acceleration, braking, and steering without visual cues or obstacle detection. Our implementation leverages the track's path nodes to create a stable racing strategy.

2 TrackUtils and Path Analysis

The `TrackUtils` script plays a key role in visualizing and analyzing race data. It introduces two key functions:

Algorithm 1 ComputeCurvature Algorithm

Input: List of track nodes
Output: Curvature value
for each pair of consecutive nodes **do**
    Compute $dx = node[i+1].x - node[i].x$
    Compute $dy = node[i+1].y - node[i].y$
    Compute $angle = arctan2(dy, dx)$
    Append $angle$ to direction_changes
end for
if length of direction_changes > 1 **then**
    $curvature = mean(diff(direction\_changes)) \times 10$
else
    $curvature = 0$
end if
return $curvature$

Algorithm 2 ComputeSlope Algorithm

Input: List of track nodes
Output: Slope value
Extract node1 and node2 from node list
Compute $dz = node2.z - node1.z$
Compute $dx = node2.x - node1.x$
Compute $dy = node2.y - node1.y$
Compute $distance = \sqrt{dx^2 + dy^2}$
if distance == 0 **then**
    $slope = 0$
else
    $slope = dz/distance$
end if
return $slope$

3 The MedianAgent Algorithm

Our MedianAgent leverages the `paths_end` values from the observation dictionary provided by the SuperTuxKart environment. The agent selects a node ahead of its current position using the following logic:

Algorithm 3 CalculateAction Algorithm

1: **Input:** Observation data, Lookahead value
2: **Output:** Action tuple (steering, acceleration, drift, nitro)
3: Compute curvature using $compute\_curvature(paths\_end[lookahead - 1])$
4: Compute slope using $compute\_slope(paths\_end[: lookahead])$
5: Determine direction to target using $direction\_to\_target = path\_end - kart\_front$
6: Compute steering: $steering = 0.2 \times direction\_to\_target[0]$
7: Adjust acceleration: $acceleration = max(0.1, 1 - |curvature| + max(0, slope))$
8: Activate drift if $|curvature| > 40$
9: Activate nitro if $|curvature| < 0.02$
10: **return** $(steering, acceleration, drift, nitro)$

Note: The constants used in lines 6, 8, and 9 were chosen arbitrarily through trial and error. They were adjusted iteratively to achieve stable and efficient driving behaviour in most conditions.

4 Experimental Results and Discussion

Our MedianAgent demonstrated significant improvements in completing race circuits with enhanced stability. By adjusting acceleration in uphill segments and enabling drift in sharp turns, the agent was able to outperform previous agents in maintaining control. However, steering performance in extremely narrow curves remains a limitation, as seen in Figures 2, 3, and 4 where aiming at a given nodes could potentially get the kart stuck.

Figure 2: Image of the kart getting off the track FortMagma

Figure 3: Image of the kart getting stuck in the track ZenMagma

Figure 4: Image of the kart getting stuck in the track stk_enterprise

5 Analysis

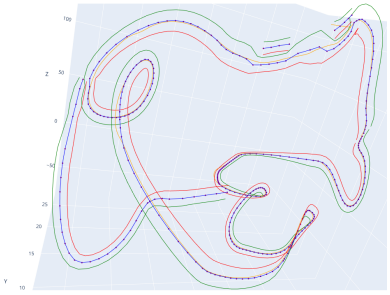

Figure 5: Path of the agent racing in Gran Paradiso

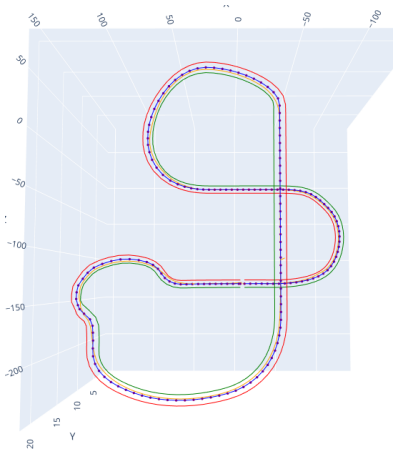

Figure 6: Path of the agent racing in Abyss

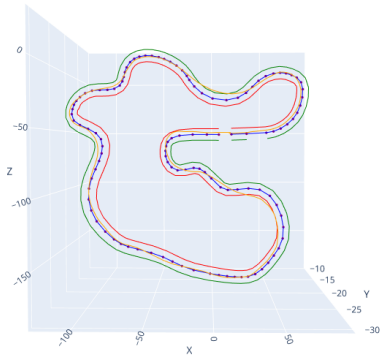

Figure 7: Path of the agent racing in Lighthouse

The MedianAgent effectively follows the middle of the track by using path nodes as reference points (purple points). However, the plots seen in Figures 5, 6 and 7 reveal significant limitations in terms of optimization. Although the MedianAgent (yellow line) generally stays centered, it does not adopt optimal trajectories in sharp turns and complex track sections. This inefficiency results in slower speeds and less precise paths, as demonstrated by the visible gaps between the followed trajectory and the track edges in tight curves.

6 Conclusion and Future Work

The MedianAgent represents a foundational implementation for autonomous driving agents in SuperTuxKart. Future enhancements should primarily consider adding adaptive path selection strategies and improved obstacle avoidance for enhanced racing capabilities.