INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ
CAMPUS TERESINA CENTRAL
CURSO TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOAO VICTOR DE SOUSA DOS SANTOS

ANÁLISE DE DESEMPENHO E GENERALIZAÇÃO DO ALGORITMO SOFT ACTOR-CRITIC (SAC) INTEGRADO À OTIMIZAÇÃO DE TRAJETÓRIA K1999 EM VEÍCULOS AUTÔNOMOS SIMULADOS

TERESINA
2026

JOAO VICTOR DE SOUSA DOS SANTOS

ANÁLISE DE DESEMPENHO E GENERALIZAÇÃO DO ALGORITMO SOFT ACTOR-CRITIC (SAC) INTEGRADO À OTIMIZAÇÃO DE TRAJETÓRIA K1999 EM VEÍCULOS AUTÔNOMOS SIMULADOS

Trabalho de Conclusão de Curso (Artigo Científico) apresentado como exigência parcial para obtenção do diploma do Curso de Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Teresina.

Orientador: Prof. Me. Francisco Marcelino Almeida de Araújo

TERESINA
2026

ANÁLISE DE DESEMPENHO E GENERALIZAÇÃO DO ALGORITMO SOFT ACTOR-CRITIC (SAC) INTEGRADO À OTIMIZAÇÃO DE TRAJETÓRIA K1999 EM VEÍCULOS AUTÔNOMOS SIMULADOS

Joao Victor De Sousa Dos Santos[1]
Francisco Marcelino Almeida de Araújo[2]

RESUMO

O desenvolvimento de veículos autônomos simulados (AWS DeepRacer) enfrenta o desafio da instabilidade de convergência e do elevado custo exploratório inerentes ao algoritmo *Soft Actor-Critic* (SAC) em espaços de ação contínuos. Este trabalho propõe avaliar o impacto da integração da otimização geométrica de trajetória (K1999) como um guia estruturado para a exploração do SAC, visando mitigar o caos exploratório inicial e o sobreajuste (*overfitting*) geométrico. Utilizou-se o ambiente em nuvem da AWS para treinar dois modelos SAC por 6 horas cada: um controle (trajetória central) e um proposto (K1999). A função de recompensa incorporou decaimento exponencial de distância e velocidade à linha ideal. Os resultados indicam que o modelo K1999 mitigou o caos exploratório, consolidando sua política ótima 74 iterações antes da linha de base. Em testes de generalização (*zero-shot*) em pista inédita, a abordagem reduziu as colisões em aproximadamente 66% e o tempo de volta pela metade. Conclui-se que a injeção de heurísticas pré-calculadas reduz a entropia improdutiva e previne falhas catastróficas do SAC, embora o modelo ainda apresente taxas residuais de *overfitting* ao lidar com cenários totalmente desconhecidos.

Palavras-chave: aprendizado por reforço; SAC; K1999; AWS DeepRacer; condução autônoma.

ABSTRACT

The development of simulated autonomous vehicles (AWS DeepRacer) faces the challenge of convergence instability and the high exploratory cost inherent to the *Soft Actor-Critic* (SAC) algorithm in continuous action spaces. This work evaluates the impact of integrating geometric trajectory optimization (K1999) as a structured guide for SAC exploration, aiming to mitigate initial exploratory chaos and geometric *overfitting*. The AWS cloud environment was used to train two SAC models for 6 hours each: a control (centerline) and a proposed model (K1999). The reward function incorporated exponential distance and velocity decay to the ideal line. Results show that the K1999 model mitigated exploratory chaos, consolidating its optimal policy 74 iterations earlier than the baseline. In *zero-shot* generalization tests on an unseen track, the approach reduced collisions by approximately 66% and cut the lap time in half. In conclusion, injecting pre-calculated heuristics reduces unproductive entropy and prevents catastrophic failures of SAC, although the model still exhibits residual rates of *overfitting* when dealing with entirely unknown scenarios.

[1] Discente do curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal do Piauí (IFPI). E-mail: joawrt8520@gmail.com.
[2] Professor Mestre do Instituto Federal do Piauí (IFPI). E-mail: Francisco.marcelino@ifpi.edu.br.

Keywords: reinforcement learning; SAC; K1999; AWS DeepRacer; autonomous driving.

1 INTRODUÇÃO

O campo da condução autônoma tem experienciado avanços significativos com a aplicação do Aprendizado por Reforço Profundo (*Deep Reinforcement Learning*). Plataformas como o AWS DeepRacer permitem a experimentação de algoritmos de ponta em ambientes de simulação de alta fidelidade, onde agentes devem aprender a navegar em trajetórias complexas através da interação contínua com o ambiente. Entre os algoritmos de destaque, o *Proximal Policy Optimization* (PPO) consolidou-se pela sua estabilidade em espaços de ação discretos (SCHULMAN *et al.*, 2017). Contudo, a transição para espaços de ação contínuos, essenciais para a suavidade de condução em veículos reais, levou à ascensão do algoritmo *Soft Actor-Critic* (SAC) (HAARNOJA *et al.*, 2018).

Apesar da sua elevada eficiência amostral e capacidade de exploração via maximização da entropia, o SAC enfrenta desafios críticos de convergência em ambientes de alta dimensionalidade. Sem uma orientação estruturada, o agente frequentemente sucumbe a um caos exploratório nas fases iniciais do treino, resultando em trajetórias erráticas. Além disso, a literatura aponta o risco de sobreajuste (*overfitting*) geométrico, onde o modelo memoriza características visuais específicas de uma pista, falhando ao ser testado em cenários inéditos (GHIMIRE, 2021).

Para mitigar estas limitações, surge a necessidade de integrar conhecimentos geométricos pré-calculados ao processo de aprendizado. O algoritmo K1999 propõe uma otimização de trajetória baseada na minimização da curvatura e no tangenciamento de curvas, transformando a linha central bruta numa trajetória de corrida física e geometricamente superior (GHIMIRE, 2021). A hipótese central deste trabalho é que a integração de tais heurísticas na função de recompensa pode atuar como um guia para o SAC, reduzindo a entropia improdutiva.

Diante deste cenário, o presente artigo tem como objetivo realizar uma análise comparativa de desempenho e generalização entre duas configurações do algoritmo SAC. O estudo avalia um modelo de controle (treinado sob a trajetória central padrão) e um modelo proposto (integrado à otimização K1999). Através de um protocolo de treinamento de 6 horas na plataforma em nuvem da AWS, pretende-se demonstrar que a orientação geométrica acelera a convergência e atua como um mecanismo de contenção de danos, mitigando a vulnerabilidade do agente e prevenindo falhas catastróficas ao ser submetido a ambientes desconhecidos.

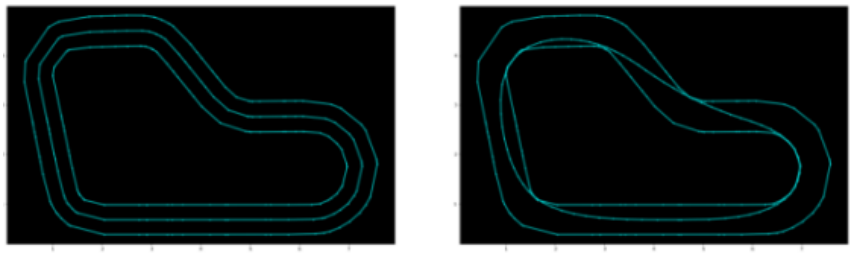
2 FUNDAMENTAÇÃO TEÓRICA

A modelagem de problemas de controle contínuo em veículos autônomos baseia-se na formulação dos Processos de Decisão de Markov (MDP), descrita pela tupla (S, A, P, R, γ). Neste paradigma, o veículo (agente) interage com a pista (ambiente), observando estados contínuos em S (imagens da câmera) e tomando ações em A (aceleração e esterçamento), com o objetivo de maximizar o retorno cumulativo R (SUTTON; BARTO, 2018).

Para lidar com a dimensionalidade infinita das ações físicas, o algoritmo *Soft Actor-Critic* (SAC) destaca-se por utilizar uma estrutura *off-policy* baseada na maximização da entropia. Diferente de métodos conservadores, o SAC regula ativamente o equilíbrio entre a exploração de novos traçados e o aproveitamento de caminhos conhecidos, somando o valor de entropia à recompensa (HAARNOJA *et al.*, 2018). Esse mecanismo previne a convergência prematura em ótimos locais, mas costuma gerar um alto custo exploratório nas fases iniciais.

Em ambientes de corrida, seguir a linha central da pista (*centerline*) é uma abordagem subótima. Pilotos utilizam o tangenciamento de curvas (*Apex*) para suavizar a trajetória. O algoritmo de otimização K1999 transfere este conceito para agentes autônomos, minimizando o comprimento total do trajeto ao aplicar um efeito matemático de "elástico tensionado"sobre as coordenadas originais (GHIMIRE, 2021). A Figura 1 ilustra esta otimização.

Figura 1 – Comparativo entre a linha central (a) e a linha de corrida otimizada (b).



(a) Original Race Track (b) Calculated Race Line

Fonte: Adaptado de (GHIMIRE, 2021).

Uma vez definida a trajetória geométrica ideal, o passo subsequente é determinar a velocidade máxima suportada em cada ponto (*waypoint*) da pista, cálculo regido pela física de aderência lateral. A velocidade máxima em curva (v_{max}) antes da derrapagem é definida pela Equação 1:

$$v_{max} = \sqrt{\mu \cdot g \cdot r}$$

Onde μ representa o coeficiente de atrito estático entre os pneus e a pista, g é a aceleração da gravidade e r é o raio da curva no waypoint específico (AU, 2021). Essa

integração cria uma linha teórica ideal, que servirá como base para orientar a função de recompensa do algoritmo SAC.

3 METODOLOGIA

3.1 AMBIENTE DE SIMULAÇÃO E PRÉ-PROCESSAMENTO OFFLINE

Os experimentos foram conduzidos no ecossistema AWS DeepRacer, utilizando integração entre os serviços Amazon SageMaker (processamento de aprendizado por reforço) e AWS RoboMaker (simulação física do agente em escala 1/18) (AU, 2021). O espaço de ação contínuo do agente permitiu ajustes granulares de velocidade (1.0 a 4.0 m/s) e esterçamento ($-30°$ a $30°$).

O cálculo iterativo de trajetórias otimizadas na nuvem consome recursos elevados. Para mitigar isso, desenvolveu-se uma ferramenta de pré-processamento offline em Python. O sistema recebe as coordenadas brutas da pista, aplica o tangenciamento do algoritmo K1999 e determina o perfil cinemático ideal para cada ponto. O resultado é compilado numa tabela de busca rápida (*lookup table*) denominada `TRACK_DATA`, que mapeia cada ponto da pista para as suas coordenadas ideais $[x, y]$ e velocidade máxima suportada. Para atender aos requisitos práticos da engenharia de software e garantir a reprodutibilidade, o script completo desenvolvido para a otimização K1999 encontra-se disponibilizado em um repositório público no GitHub: <https://github.com/LabirasIFPI/tcc-deepracer-sac-com-k1999>.

3.2 ENGENHARIA DA FUNÇÃO DE RECOMPENSA

A função de recompensa customizada (R) foi estruturada como uma soma ponderada, focada em dois pilares geométricos: acompanhamento da trajetória e perfil cinemático, acrescida de um bônus de sobrevivência. A recompensa total do agente em um dado instante é definida pela Equação 2:

$$R_{total} = (w_1 \cdot R_{traj}) + (w_2 \cdot R_{vel}) + b$$

Onde w_1 (0.6) e w_2 (0.4) representam os pesos atribuídos à precisão do traçado e à fidelidade da velocidade, respectivamente, enquanto b representa um bônus constante (0.1) por passo bem-sucedido na pista. Caso o veículo saia da pista (*off-track*), aplica-se uma punição severa (1×10^{-3}) e o bônus é abortado.

O componente de trajetória (R_{traj}) utiliza uma função de decaimento exponencial contínuo para avaliar a precisão do agente em relação à linha ótima do `TRACK_DATA` (Equação 3):

$$R_{traj} = e^{-\beta \cdot \Delta d}$$

Onde Δd representa a distância euclidiana entre o veículo e a coordenada ideal, e $\beta = 3.0$ é o hiperparâmetro de sensibilidade. De forma análoga, o componente cinemático (R_{vel}) alinha a ação ao perfil de velocidade (Equação 4):

$$R_{vel} = e^{-\gamma \cdot \Delta v}$$

Onde Δv é o erro absoluto entre a velocidade atual e a ideal estipulada pelo K1999, com fator de decaimento $\gamma = 2.0$. Esta formulação força o algoritmo SAC a frear e acelerar nos exatos pontos determinados pela física de aderência. Buscando evidenciar a aplicação técnica no escopo da Análise e Desenvolvimento de Sistemas, a transcrição dessa modelagem matemática para a lógica de programação em Python foi implementada na função de recompensa conforme o trecho a seguir:

```
# Calculo de decaimento exponencial (Distancia e Velocidade)
dist_reward = math.exp(-3.0 * distance_to_target)
speed_reward = math.exp(-2.0 * abs(speed - target_speed))

# Recompensa total (Pesos + Bonus de sobrevivencia)
reward = (0.6 * dist_reward) + (0.4 * speed_reward) + 0.1
```

3.3 CONFIGURAÇÃO EXPERIMENTAL

O protocolo consistiu no treinamento de dois modelos SAC por um período ininterrupto de 6 horas cada, utilizando o traçado *re:Invent 2018*. O Modelo A (Controle) operou sobre a trajetória central padrão com velocidade constante, enquanto o Modelo B (Proposto) integrou os dados de otimização geométrica K1999. Os hiperparâmetros foram mantidos idênticos (Tabela 1) para isolar o impacto da trajetória otimizada.

Tabela 1 – Hiperparâmetros de treinamento para os Modelos A e B.

Hiperparâmetro	**Valor (Modelos A e B)**
Batch Size	256
Learning Rate	0.0003
Discount Factor (γ)	0.99
SAC *Alpha* (Entropy)	0.2

Fonte: Elaborado pelo autor (2026).

4 RESULTADOS E DISCUSSÃO

A avaliação de desempenho foi conduzida a partir do cruzamento de métricas de telemetria extraídas do console AWS DeepRacer, partindo da estabilidade de convergência na fase de treinamento, passando pelos testes físicos na pista conhecida,

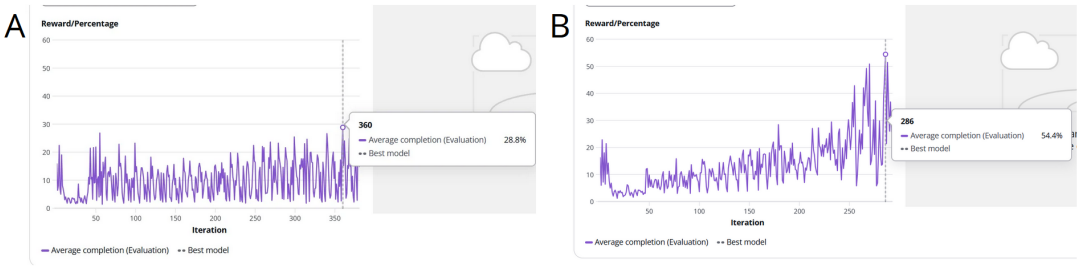
e culminando no teste de generalização geométrica em circuito inédito (*Zero-Shot Learning*).

4.1 CONVERGÊNCIA E MITIGAÇÃO DO CAOS EXPLORATÓRIO

Os resultados do treinamento de 6 horas revelaram disparidades fundamentais no comportamento de aprendizagem. O Modelo A (SAC Base), operando sem o balizamento geométrico prévio, demonstrou alta suscetibilidade à estocasticidade de suas ações. A curva de aprendizagem sofreu flutuações violentas, mantendo um teto de conclusão inferior a 30% na maioria das tentativas. A sua melhor política de navegação (*Best Model*) foi registrada tardiamente, apenas na iteração 360.

Em contrapartida, o Modelo B (SAC + K1999) apresentou uma convergência substancialmente mais estável. A injeção das coordenadas de tangenciamento atuou como um trilho virtual, permitindo ao veículo prolongar a sua permanência no traçado. O Modelo B rompeu a barreira dos 50% de conclusão rapidamente e cravou o seu *Best Model* na iteração 286 (74 iterações antes da linha de base), mitigando a entropia improdutiva inerente ao algoritmo, conforme ilustrado na Figura 2.

Figura 2 – Estabilidade de convergência durante o treinamento: Modelo A (esq.) e Modelo B (dir.).



Fonte: Elaborado pelo autor (2026).

4.2 DESEMPENHO NO CENÁRIO CONHECIDO

Para quantificar a eficácia das políticas aprendidas, ambos os modelos executaram 5 tentativas (*trials*) na pista de treinamento (*re:Invent 2018*) (AWS DEEPRACER COMMUNITY, 2026a). A Tabela 2 cruza os tempos de volta com a métrica de reposicionamentos forçados (*resets*).

Ambas as arquiteturas alcançaram 100% de conclusão do traçado, validando o algoritmo SAC. No entanto, o Modelo B demonstrou superioridade esmagadora, cravando a sua melhor volta em 12,769 segundos, menos da metade do tempo da melhor volta do Modelo A (25,532 segundos). A ausência de um referencial de tangenciamento fez com que o Modelo A conduzisse de forma reativa, excedendo os limites da via entre 7

Tabela 2 – Avaliação de tempo e *resets* na pista *re:Invent 2018*.

Tentativa	Tempo (Mod. A)	Resets (Mod. A)	Tempo (Mod. B)	Resets (Mod. B)
1	00:27.791	8	00:12.769	1
2	00:34.849	11	00:12.769	4
3	00:27.992	9	00:18.037	4
4	00:28.023	8	00:15.785	3
5	00:25.532	7	00:15.517	2

Fonte: Elaborado pelo autor (2026).

e 11 vezes. O Modelo B maximizou a inércia pelo trajeto mais curto (*Apex*), chegando a completar o circuito com apenas um *reset*.

4.3 GENERALIZAÇÃO EM CIRCUITO INÉDITO (*ZERO-SHOT LEARNING*)

A etapa final submeteu os agentes à pista *Asia Pacific Bay Loop* (AWS DEEPRACER COMMUNITY, 2026b), um traçado desconhecido e complexo. O teste revelou o fenômeno do sobreajuste (*overfitting*) geométrico na linha de base. Conforme a Tabela 3, o Modelo A entrou em colapso operacional: embora tenha completado as voltas em cerca de 2 minutos e 30 segundos, esse tempo foi inflado por uma média de 50 *resets* por volta, provando que o agente apenas memorizou as coordenadas originais de treino.

Tabela 3 – Desempenho de generalização na pista inédita *Asia Pacific Bay Loop*.

Tentativa	Tempo (Mod. A)	Resets (Mod. A)	Tempo (Mod. B)	Resets (Mod. B)
1	02:29.725	50	01:12.489	15
2	02:31.571	51	01:14.899	17
3	02:37.017	50	01:24.606	20

Fonte: Elaborado pelo autor (2026).

O Modelo B superou a linha de base em todas as métricas, cravando sua melhor volta em 1 minuto e 12 segundos e reduzindo os *resets* para uma média de 17. Contudo, faz-se necessária uma análise crítica e realista destes resultados. Embora a redução de aproximadamente 66% nas colisões represente um ganho estatístico expressivo e comprove que o K1999 impediu a falha total do SAC, registrar entre 15 e 20 colisões em uma única volta não configura uma navegação autônoma bem-sucedida. Pelo contrário, evidencia que o Modelo B foi apenas "menos catastrófico". Fica demonstrado que, mesmo com a mitigação do caos exploratório via injeção de heurísticas, o algoritmo SAC ainda sofre de *overfitting* geométrico. A alta taxa de reposicionamentos residuais inviabiliza a aplicação segura do modelo em uma corrida real em cenários inéditos sem que haja um retreinamento específico prévio.

5 CONSIDERAÇÕES FINAIS

Esta pesquisa investigou o desafio da instabilidade de convergência e do elevado custo exploratório associados ao algoritmo *Soft Actor-Critic* (SAC) em ambientes de condução autônoma. A hipótese de que a integração das heurísticas geométricas e cinemáticas do algoritmo K1999 mitigaria essas limitações foi empiricamente validada no cenário de treino. A função de recompensa customizada converteu o caos inicial em um aprendizado estruturado, permitindo ao agente consolidar sua política de forma acelerada e reduzir o tempo de volta pela metade em circuito conhecido.

Contudo, a avaliação de generalização (*Zero-Shot*) exigiu uma análise crítica sobre os limites do algoritmo. O uso do K1999 revelou-se um eficiente mecanismo de contenção de danos, impedindo a falha total do agente e reduzindo as taxas de colisões em aproximadamente 66% frente ao modelo base. Todavia, a manutenção de uma média residual de 17 reposicionamentos por volta evidencia que o modelo não alcançou uma condução autônoma fluida e segura em cenário inédito. Conclui-se que o referencial geométrico é capaz de limitar o fracasso exploratório, mas não suprime inteiramente o *overfitting* inerente à natureza visual do SAC sem que haja um retreinamento específico.

Como perspectivas para trabalhos futuros, sugere-se a expansão da camada sensorial (*LiDAR* ou câmeras estéreo) para reduzir a dependência exclusiva da geometria pré-calculada, bem como a transferência do modelo para o ambiente físico (*sim-to-real*) num veículo AWS DeepRacer em escala $1/18$, visando avaliar o impacto de ruídos reais na estabilidade da política de controle.

REFERÊNCIAS

AU, C. M. **AWS DeepRacer**. Dissertação (Final Year Project Report) — City University of Hong Kong, 2021. Citado 2 vezes nas páginas 4 e 5.

AWS DEEPRACER COMMUNITY. **Deepracer-race-data: reinvent_base**. 2026. GitHub. Disponível em: <https://github.com/aws-deepracer-community/deepracer-race-data/blob/main/raw_data/tracks/assets/arn:aws:deepracer:us-east-1::track/reinvent_base/track-resources/reinvent_base.svg>. Acesso em: 14 maio 2026. Citado na página 7.

AWS DEEPRACER COMMUNITY. **Deepracer-race-data: singapore**. 2026. GitHub. Disponível em: <https://github.com/aws-deepracer-community/deepracer-race-data/blob/main/raw_data/tracks/assets/arn:aws:deepracer:us-east-1::track/Singapore/track-resources/singapore.svg>. Acesso em: 15 maio 2026. Citado na página 8.

GHIMIRE, M. **A Study of Deep Reinforcement Learning in Autonomous Racing Using DeepRacer Car**. Dissertação (Undergraduate Thesis (Honors)) — University of Mississippi, 2021. Citado 2 vezes nas páginas 3 e 4.

HAARNOJA, T. *et al.* Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: PMLR. **International Conference on Machine Learning (ICML)**. [S.l.], 2018. p. 1861–1870. Citado 2 vezes nas páginas 3 e 4.

SCHULMAN, J. *et al.* Proximal policy optimization algorithms. **arXiv preprint arXiv:1707.06347**, 2017. Disponível em: <https://arxiv.org/abs/1707.06347>. Acesso em: 20 maio 2026. Citado na página 3.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: An introduction**. 2. ed. Cambridge: MIT press, 2018. Citado na página 4.