PROJETO PROPOSTO:

Self-Evolving Reinforcement Learning Agent para Controle Adaptativo de AGVs

(Simulado, computacionalmente leve e interpretável)

1. Visão Geral do Projeto

Este projeto propõe o desenvolvimento de um **agente de Reinforcement Learning (RL)** capaz de **ajustar dinamicamente os ganhos de um controlador PID** para o seguimento de trajetória de um **AGV (Automated Guided Vehicle)** em um ambiente simulado e simplificado.

Diferentemente de abordagens que substituem controladores clássicos por modelos opacos, o método utiliza **RL como mecanismo de adaptação dos parâmetros** de um controlador amplamente aceito na indústria, mantendo **interpretabilidade, estabilidade e baixo risco operacional**.

O foco é demonstrar que um agente RL simples pode **igualar ou superar** uma sintonia empírica fixa, especialmente sob **perturbações e mudanças de cenário**, sem exigir grande poder computacional.

2. Objetivo

Desenvolver um **protótipo funcional** onde um agente aprende por interação com o ambiente a:

- Ajustar ganhos PID online
- Minimizar erro de seguimento de trajetória
- Preservar estabilidade do sistema
- Operar com baixo custo computacional

O protótipo não busca entregar um produto final, mas **testar hipóteses e orientar decisões futuras** sobre controle adaptativo em AGVs.

3. Ambiente de Simulação (Leve e Controlado)

O ambiente é um **simulador customizado inspirado no padrão Gym**, projetado para ser:

- Totalmente em CPU
- Sem gráficos 3D
- Sem dependência de GPU
- Reproduzível e rápido

3.1 Estado do Ambiente

O estado observado pelo agente é um vetor de baixa dimensionalidade:

```
s = [
  erro_lateral,
  erro_angular,
  erro_integral,
  derivada_do_erro,
  tipo_de_trajetoria (reta / curva)
]
```

Essas variáveis são suficientes para representar a dinâmica relevante do problema de seguimento de trajetória.

4. Espaço de Ações

O agente **não controla diretamente motores ou atuadores**, o que reduz risco e complexidade. Em vez disso, ele atua **sobre os ganhos do controlador PID**.

Conjunto de ações:

A = {

+ΔKp, -ΔKp,

+ΔKi, -ΔKi,

+ΔKd, -ΔKd,

noop

}

Ou seja:

- aumentar ou diminuir Kp
- aumentar ou diminuir Ki
- aumentar ou diminuir Kd
- manter ganhos atuais

Isso transforma um PID tradicional de ganhos fixos em um **PID adaptativo orientado por aprendizado**.

5. Controlador

O controlador segue a forma clássica:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int e(t)\,dt + K_d \cdot \frac{de(t)}{dt}$$

Com a diferença de que os ganhos **não são estáticos**, mas ajustados dinamicamente pelo agente RL dentro de **limites de segurança previamente definidos**.

Com a diferença de que os ganhos **não são estáticos**, mas ajustados dinamicamente pelo agente RL dentro de **limites de segurança previamente definidos**.

6. Função de Recompensa

A recompensa é projetada para equilibrar **precisão e estabilidade**:

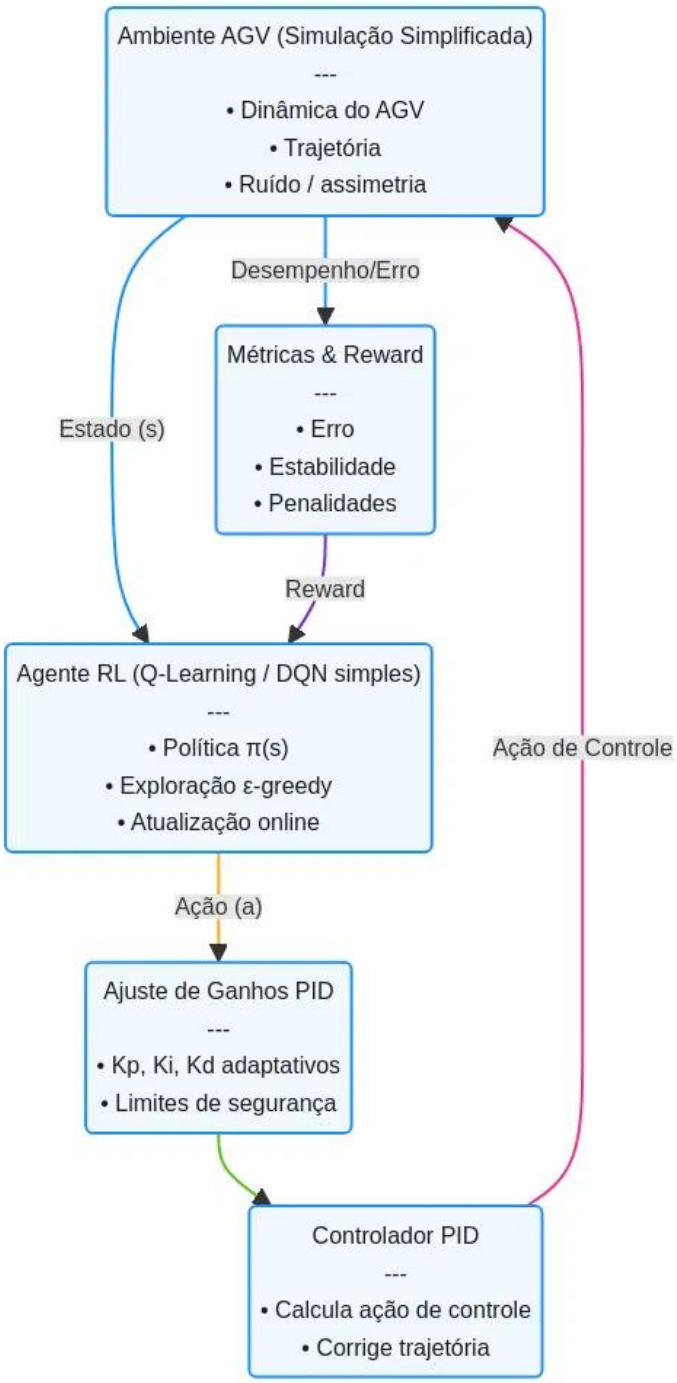
$$R = -\big(|erro| + \alpha \cdot |\dot{erro}| + \beta \cdot variância\big)$$

Onde:

- o erro absoluto incentiva precisão
- o termo derivativo penaliza oscilações
- a variância penaliza instabilidade

Essa formulação evita políticas agressivas e promove comportamento suave e robusto.

7. Arquitetura Geral do Sistema



8. Loop de Treinamento

- Inicializa ambiente
- Inicializa ganhos PID

Para cada episódio:

- reset ambiente
- para cada step:
- observa estado s
- escolhe ação a (ε-greedy)
- ajusta ganhos PID
- executa controle
- observa novo estado s'
- calcula recompensa
- atualiza política

Esse loop caracteriza aprendizado **por experiência**, conforme exigido pelo desafio.

9. O Diferencial: Self-Evolving Agent

O conceito de **self-evolving** refere-se à capacidade do agente de **adaptar sua política ao longo do tempo**, mesmo quando o ambiente muda.

Exemplos:

- mudança no raio da curva
- aumento de ruído de sensor
- atraso de controle
- assimetrias dinâmicas

Estratégias:

- reutilização de políticas aprendidas
- checkpoints de boas políticas
- reaprendizado mais rápido (transfer learning básico)

10. Avaliação Experimental

Comparação direta:

- PID empírico fixo
- PID + RL adaptativo

Métricas:

- Erro médio absoluto (RMSE)
- ITAE
- Estabilidade
- Tempo de acomodação

Análises gráficas:

- erro × tempo
- trajetória desejada × real
- evolução de Kp, Ki e Kd ao longo do tempo

Em vez de substituir controladores clássicos por redes neurais opacas, este trabalho utiliza Reinforcement Learning para evoluir parâmetros de um controlador industrialmente aceito, reduzindo risco e aumentando interpretabilidade.

12. Conclusão

O protótipo demonstra que:

- RL pode ser usado como **camada adaptativa**
- PID continua sendo o núcleo do controle
- O sistema é leve, interpretável e extensível
- A abordagem é viável para aplicações industriais reais