Perceção e Controlo
2024-2025

Trabalho 2
Corridor Runner using the
CiberRato simulation environment

Rafael Morgado
104277

universidade de aveiro

Introdução

- Este projeto tem como objetivo desenvolver dois programas distintos para um robô simulado no ambiente CiberRato.
- O primeiro programa utiliza técnicas de **controlo clássico** para navegação.
- Enquanto o segundo usa métodos de **Machine Learning**.
- Cada abordagem será implementada e avaliada de forma independente, permitindo explorar diferentes estratégias de navegação autônoma em labirintos desconhecidos.

Objetivos

- O objetivo deste trabalho é fazer com que o robô navegue autonomamente por um caminho desconhecido, definido por paredes, completando o máximo de score possível num tempo limitado.
- O robô deverá utilizar os seus sensores (linha, obstáculos e sensores de solo) para:
  - Seguir corredores de forma eficiente.
  - Detetar checkpoints para verificar direção e progresso.
  - Evitar obstáculos enquanto mantém altas velocidades.


Clássico

Controlador clássico

(classicAgent.py)

- **Desenvolvimento do agente:**
  - Implementação de um **controlador PID** para ajustar a direção do robô com base nos sensores laterais.
  - O controlador corrige a trajetória do robô, mantendo-o no centro dos corredores.

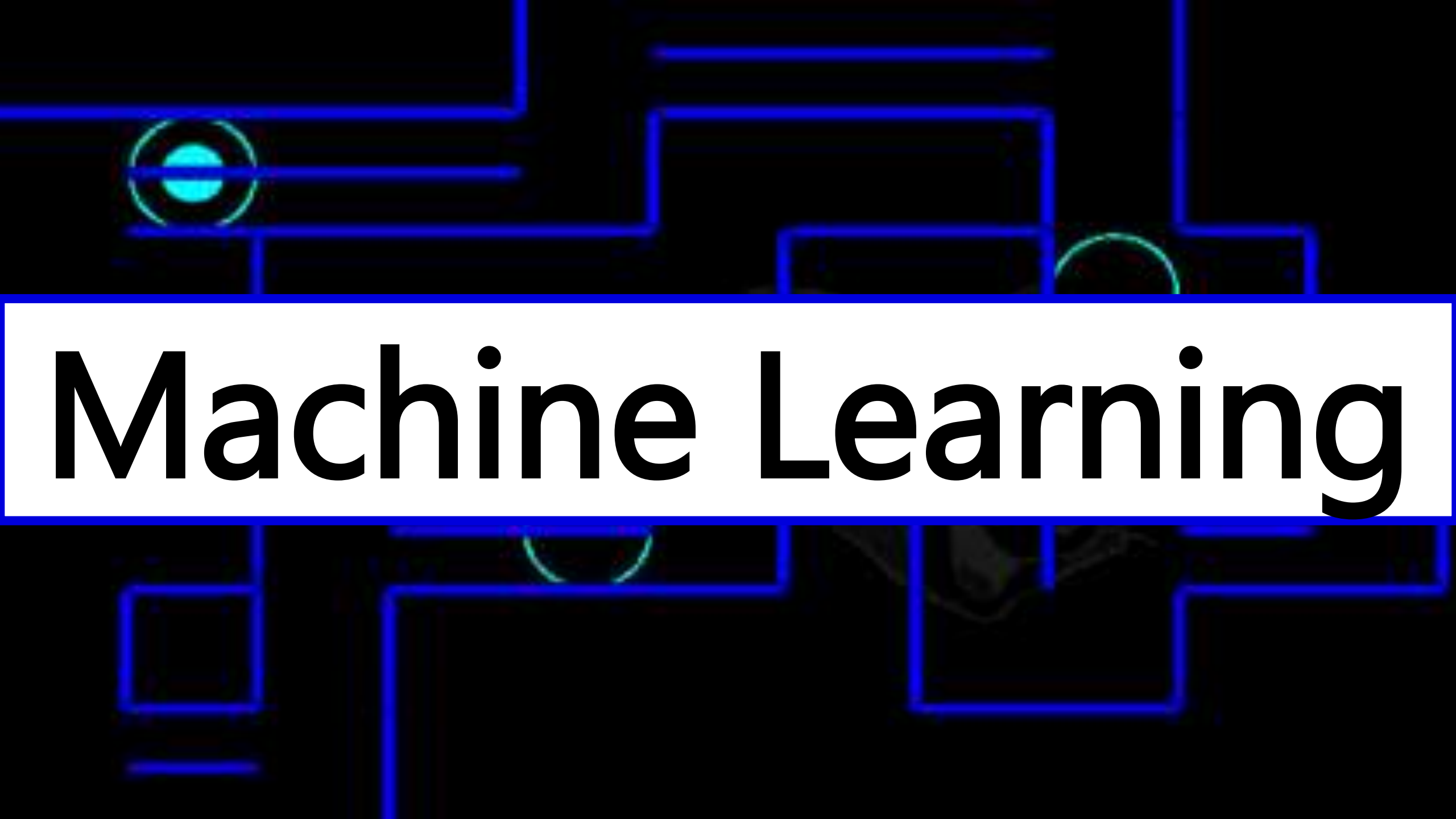
- **Prevenção de colisões:**
  - O robô deteta obstáculos nos sensores frontais e laterais.
  - Foram implementadas reações evasivas, com rotações rápidas para evitar colisões.

- **Melhoria de desempenho:**
  - Foram feitos ajustes na velocidade base para tornar o robô mais rápido.
  - A Frequência de atualização foi aumentada para garantir respostas rápidas às mudanças no ambiente.

Controlador PID

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \frac{de(t)}{dt}$$

- O controlador PID ajusta a velocidade das rodas do robô com base no erro entre os sensores laterais (**left** e **right**) para manter o robô centralizado no corredor. Ele combina três componentes principais:
  1. **Proporcional (Kp):** Reage proporcionalmente ao erro atual, ajustando rapidamente para corrigir desvios.
  2. **Integral (Ki):** Acumula erros ao longo do tempo para corrigir desvios persistentes, garantindo alinhamento a longo prazo.
  3. **Derivativo (Kd):** Prevê mudanças futuras no erro, amortecendo oscilações e melhorando a estabilidade.


Machine Learning

Modelo de treino

(ciberCorridorEnv.py)

- Neste método é usado o algoritmo **PPO (Proximal Policy Optimization)**.
- O robô recebe informações do ambiente através de sensores e toma decisões ajustando as velocidades de seus motores. O algoritmo **PPO** otimiza as ações do robô para maximizar uma recompensa, que é baseada em seu progresso e penalidades para colisões ou falta de movimento eficiente.
- Após o treino, o modelo é salvo para uso posterior e avaliado em diversos episódios para medir o seu desempenho médio e consistência. A ideia é que, no final do treino, o robô seja capaz de navegar de forma autônoma, evitando obstáculos e progredindo no corredor de maneira eficiente.

Avaliação do modelo

(Agent_model.py)

- O modelo posteriormente salvo será então utilizado neste programa
- Ele será carregado, serão aplicadas as políticas aprendidas ao ambiente e medido o desempenho do agente em múltiplos episódios.
- Durante cada episódio, o agente recebe ações baseadas nas observações do ambiente, coleta recompensas, e verifica se a simulação termina.
- Ao final, o código calcula e exibe estatísticas como a média e o desvio padrão das pontuações, permitindo analisar a eficácia do modelo.

Resultados

- Em ambos os métodos, o robô consegue atingir o objetivo proposto, apresentando vantagens e desvantagens específicas em cada abordagem:
  - **Machine Learning**: Nesse método, o robô é capaz de alcançar pontuações mais altas, mas apresenta instabilidades, como a inversão de direção em alguns episódios, resultando em variações significativas nas pontuações (média de score: 2000-3500).
  - **Controlador Clássico**: Por outro lado, este método oferece maior estabilidade, porém com resultados menos expressivos em termos de pontuação (média de score: 2500-2800).

- Conclui-se que o método de **Machine Learning**, apesar de alcançar pontuações mais elevadas, apresenta maior variabilidade e instabilidade, em parte devido à falta de utilização de checkpoints, o que poderia melhorar significativamente o desempenho e a consistência. Por outro lado, o **Controlador Clássico** oferece resultados mais estáveis e previsíveis, ainda que com pontuações ligeiramente inferiores.