Bipedal Walker

Box 2D

Nuno Moreira
Ricardo Ribeiro
Rui Coelho

Definições do Ambiente

Num	Observation	Min	Max	Mean
0	hull_angle	0	2*pi	0.5
1	hull_angularVelocity	-inf	+inf	-
2	vel_x	-1	+1	-
3	vel_y	-1	+1	-
4	hip_joint_1_angle	-inf	+inf	-
5	hip_joint_1_speed	-inf	+inf	-
6	knee_joint_1_angle	-inf	+inf	-
7	knee_joint_1_speed	-inf	+inf	-
8	leg_1_ground_contact_flag	0	1	-
9	hip_joint_2_angle	-inf	+inf	-
10	hip_joint_2_speed	-inf	+inf	-
11	knee_joint_2_angle	-inf	+inf	-
12	knee_joint_2_speed	-inf	+inf	-
13	leg_2_ground_contact_flag	0	1	-
14-23	10 lidar readings	-inf	+inf	-

Cada linha da tabela representa uma variável de observação que é obtida em cada estado do agente durante a simulação

Estados (states)

Um estado é representado por um vetor contínuo que descreve a configuração e dinâmica do agente.

Perceções (percepts)

As perceções referem-se às observações fornecidas ao agente sobre o estado atual do ambiente, e estas são divididas em três grandes grupos:

- Estado do corpo do agente (Hull)
- Estado das articulações das pernas
- Leituras do sensor LIDAR

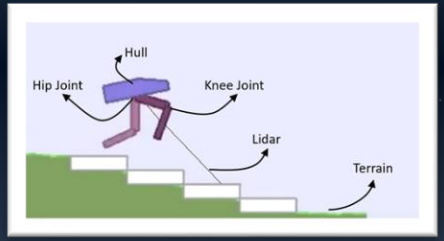
Definições do Ambiente

Ações (Actions)

O espaço de ações é contínuo (intervalo [-1,1]) e representam valores de velocidade para as articulações do Bipedal Walker, tanto no quadril como nos joelhos.

As ações são usadas para alcançar os seguintes objetivos:

- Progredir horizontalmente (andar para a frente).
- Manter o equilíbrio do corpo do robô.
- Alternar as pernas de forma adequada.
- Evitar quedas e obstáculos.



Recompensas (Rewards)

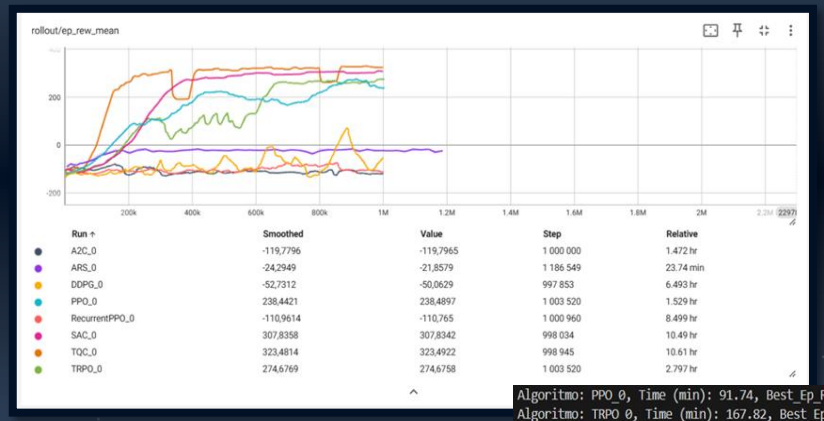
É atribuída uma recompensa pelo movimento para a frente, totalizando mais de 300 pontos até ao final do percurso.

Caso o robô caia, recebe uma penalização de -100 pontos. A manipulação de velocidades nas articulações (quadris e o joelhos) tem um pequeno custo em pontos.

Um agente mais eficiente obterá uma pontuação superior.

Fase de Desenvolvimento

Começamos por treinar 8 algoritmos (modo normal) até 1 milhão de steps para decidir em que algoritmo nos íamos focar com mais detalhe.

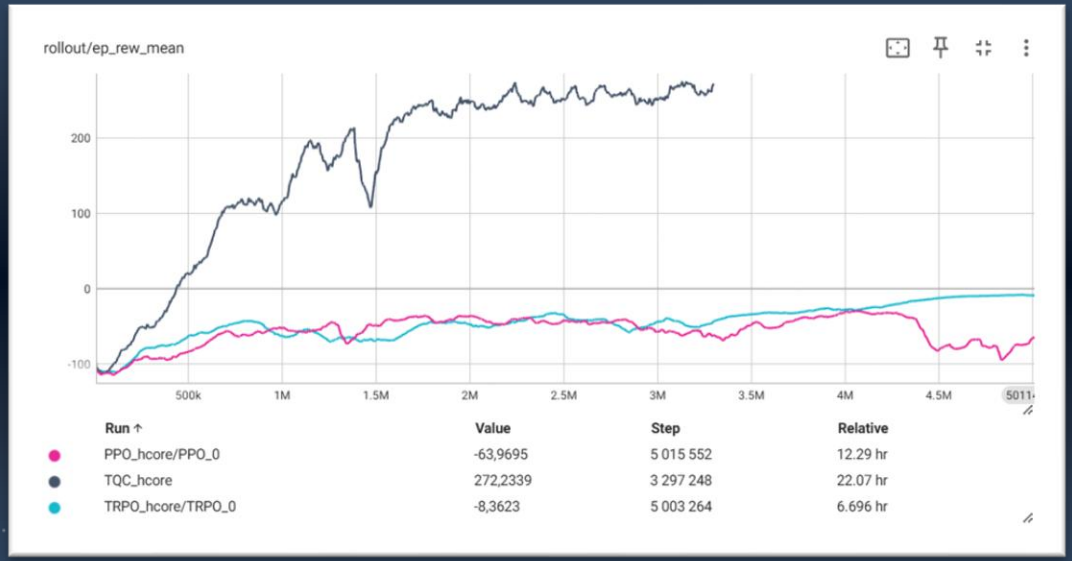


Run	Smoothed	Value	Step	Relative
A2C_0	-119,7796	-119,7965	1 000 000	1.472 hr
ARS_0	-24,2949	-21,8579	1 186 549	23.74 min
DDPG_0	-52,7312	-50,0629	997 853	6.493 hr
PPO_0	238,4421	238,4897	1 003 520	1.529 hr
RecurrentPPO_0	-110,9614	-110,765	1 000 960	8.499 hr
SAC_0	307,8358	307,8342	998 034	10.49 hr
TQC_0	323,4814	323,4922	998 945	10.61 hr
TRPO_0	274,6769	274,6758	1 003 520	2.797 hr

Após análise do gráfico, corremos um script que nos dava o rácio do value por tempo gasto. Os três melhores algoritmos segundo esse rácio são PPO, TRPO e TQC, sendo assim vamo-nos focar nesses algoritmos.

```
Algoritmo: PPO_0, Time (min): 91.74, Best_Ep_Rew_Mean: 276.21, Ratio (Best_Ep_Rew_Mean/Time): 3.010830
Algoritmo: TRPO_0, Time (min): 167.82, Best_Ep_Rew_Mean: 274.77, Ratio (Best_Ep_Rew_Mean/Time): 1.637297
Algoritmo: TQC_0, Time (min): 636.60, Best_Ep_Rew_Mean: 329.05, Ratio (Best_Ep_Rew_Mean/Time): 0.516884
Algoritmo: SAC_0, Time (min): 629.40, Best_Ep_Rew_Mean: 309.32, Ratio (Best_Ep_Rew_Mean/Time): 0.491458
Algoritmo: DDPG_0, Time (min): 389.58, Best_Ep_Rew_Mean: 72.51, Ratio (Best_Ep_Rew_Mean/Time): 0.186131
Algoritmo: RecurrentPPO_0, Time (min): 509.94, Best_Ep_Rew_Mean: -70.99, Ratio (Best_Ep_Rew_Mean/Time): -0.139207
Algoritmo: ARS_0, Time (min): 23.70, Best_Ep_Rew_Mean: -13.87, Ratio (Best_Ep_Rew_Mean/Time): -0.585155
Algoritmo: A2C_0, Time (min): 88.32, Best_Ep_Rew_Mean: -78.31, Ratio (Best_Ep_Rew_Mean/Time): -0.886638
```

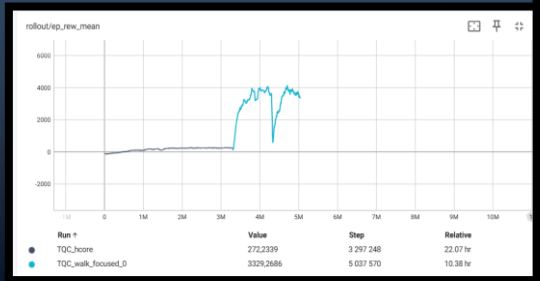
Com foco no PPO, TRPO e TQC, decidimos treinar estes algoritmos até aos 5 milhões de steps, mas desta vez no modo hardcore.



Run	Value	Step	Relative
PPO_hcore/PPO_0	-63,9695	5 015 552	12.29 hr
TQC_hcore	272,2339	3 297 248	22.07 hr
TRPO_hcore/TRPO_0	-8,3623	5 003 264	6.696 hr

Pela análise do gráfico, conclui-se que os values, tanto do PPO como TRPO são muito baixos, pelo que vamos implementar *Reward Wrappers*, de forma a maximizar o value. Já o TQC apresenta um value muito positivo, e o foco sobre este algoritmo é em melhorar o andar do agente.

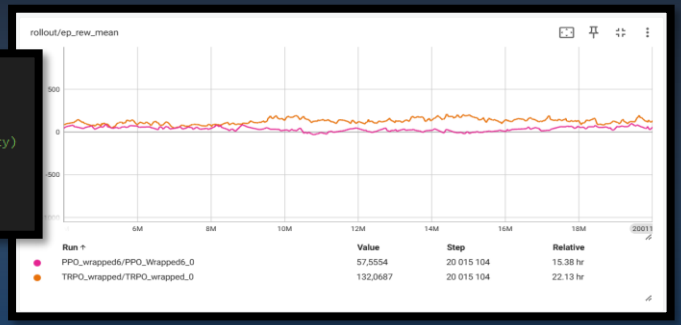
Estes foram os resultados no ambiente com reward wrapper para o TQC., Treinamos por 1 milhão de steps, desde o hardcore mode, totalizando 6 milhões de steps.




```python
# Penalize excessive vertical motion (jumping)
if abs(vertical_velocity) > 1.2:  # Allow small vertical movement
    reward -= 5.0

# Encourage smooth movement (low vertical acceleration)
vertical_acceleration = abs(vertical_velocity - self.last_vertical_velocity)
reward -= vertical_acceleration * 2.0  # Penalize sharp changes in vertical velocity

# Reward forward movement (scaled to avoid overshooting)
reward += forward_velocity * 1.5  # Strong reward for forward motion

# Penalize torso tilting
torso_angle = hull.angle
reward -= abs(torso_angle) * 0.5  # Penalize large tilts

# Encourage alternating leg movement
left_leg_angle = joints[0].angle
right_leg_angle = joints[1].angle
leading_leg = "left" if left_leg_angle > right_leg_angle else "right"

if self.last_leg_position is not None:
    if self.last_leg_position != leading_leg:
        reward += 3.0  # Strong reward for alternating legs
    else:
        reward -= 5.0  # Penalize sticking with one leg

# Penalize asymmetry in leg movement
leg_angle_diff = abs(left_leg_angle - right_leg_angle)
reward -= leg_angle_diff * 0.5  # Penalize imbalance between legs
```

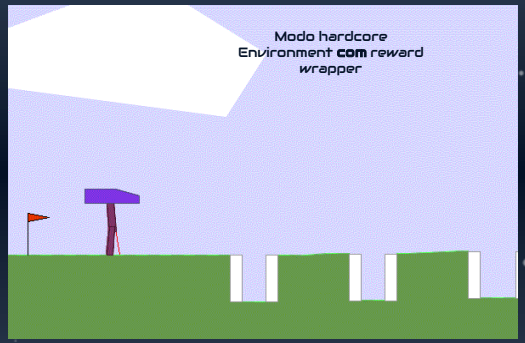
Estes foram os resultados no ambiente com Reward Wrappers para o PPO e TRPO. Treinamos ambos os algoritmos até 25 milhões de steps, apresentando em ambos os casos apresentavam values de rewards positivos.

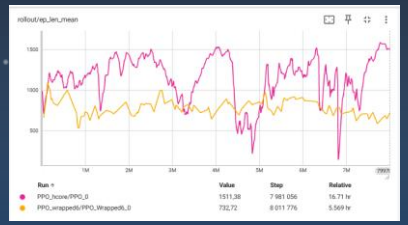
```python
class BalanceRewardWrapper(gym.RewardWrapper):
    def step(self, action):
        obs, reward, done, truncated, info = self.env.step(action)

        # Add a reward for keeping balance (absolute forward velocity)
        balance_reward = abs(obs[2])
        reward += balance_reward

        return obs, reward, done, truncated, info
```



Parte Experimental e Análise de Resultados

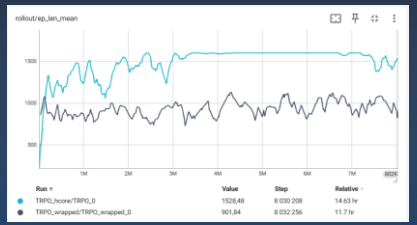
PPO

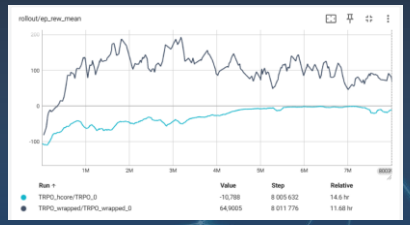




Parte Experimental e Análise de Resultados

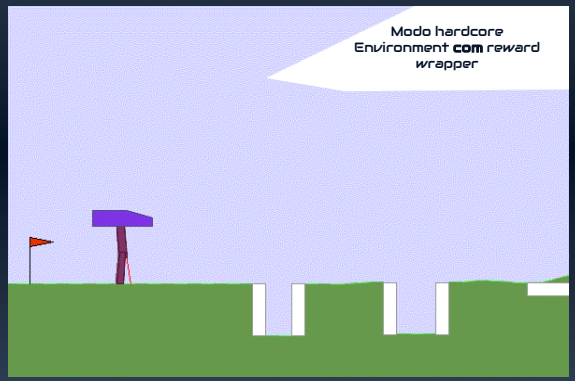
TRPO





Parte Experimental e Análise de Resultados

TQC



A implementação do *reward wrapper* tornou o andar menos robótico, obrigando o robô a usar as duas pernas.

Conclusões

PPO

Vantagens

Simples, versátil e eficiente; boa taxa de aprendizagem e estabilidade.

Desvantagens

Menos robusto em situações extremas; atualizações menos precisas.

TQC

Vantagens

Altamente robusto e eficiente em exploração; desempenho estável.

Desvantagens

Tempo de treino do modelo muito elevado.

TRPO

Vantagens

Alta estabilidade e segurança; bom desempenho em tarefas contínuas

Desvantagens

Complexidade computacional elevada; mais lento que PPO e resultados semelhantes

Trabalho Futuro

No futuro, pretendemos treinar o robô com outros algoritmos e em diferentes contextos. Planeamos criar um novo ambiente e adaptar o robô de forma a que este consiga completar o percurso com sucesso.