GoToGol

Carlos Araújo
Daiane Mendes
João Paulo Castilho
Pedro Olímpio

MO651 - Robótica Móvel

27 de Novembro de 2019

Agenda

- Introdução
- Trabalhos Relacionados
- Metodologia
  1. Lógica Fuzzy
  2. Aprendizagem por Reforço
- Resultados
- Conclusões

Introdução

GoToGol

- Ir para o objetivo
- Evitar obstáculos

Introdução

GoToGol

- Ir para o objetivo
- Evitar obstáculos

Abordagens

- Lógica *Fuzzy*
- Aprendizagem por Reforço
  - *Q*-Learning
  - Deep *Q*-Network (DQN)

Introdução

GoToGol

- Ir para o objetivo
- Evitar obstáculos

Abordagens

- Lógica *Fuzzy*
- Aprendizagem por Reforço
  - *Q*-Learning
  - Deep *Q*-Network (DQN)

Objetivo

- Avaliar a diferença de desempenho na realização do trajeto

Introdução - Cenário *Easy*

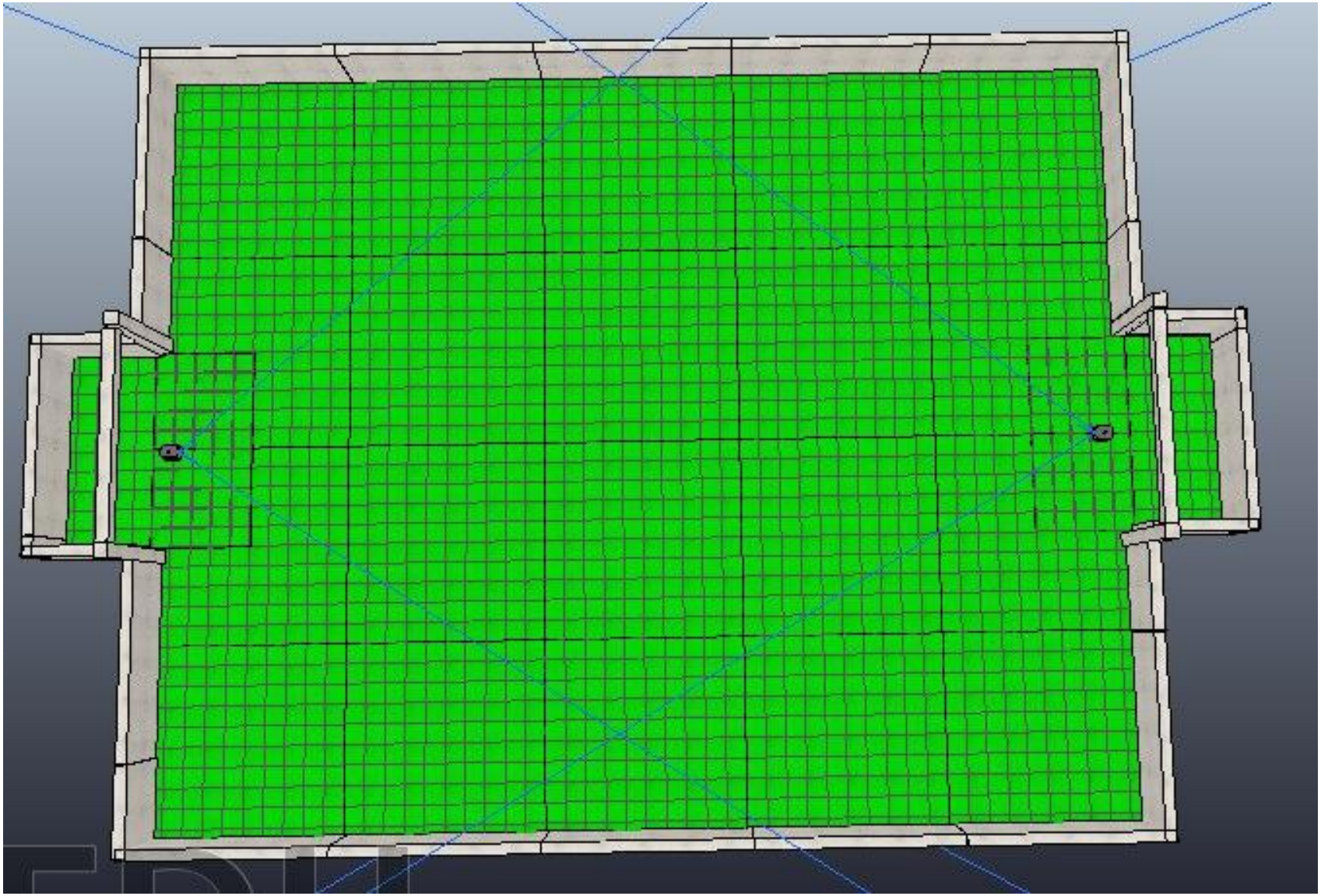


Figura: Cenário sem obstáculos.

Introdução - Cenário *Hard*

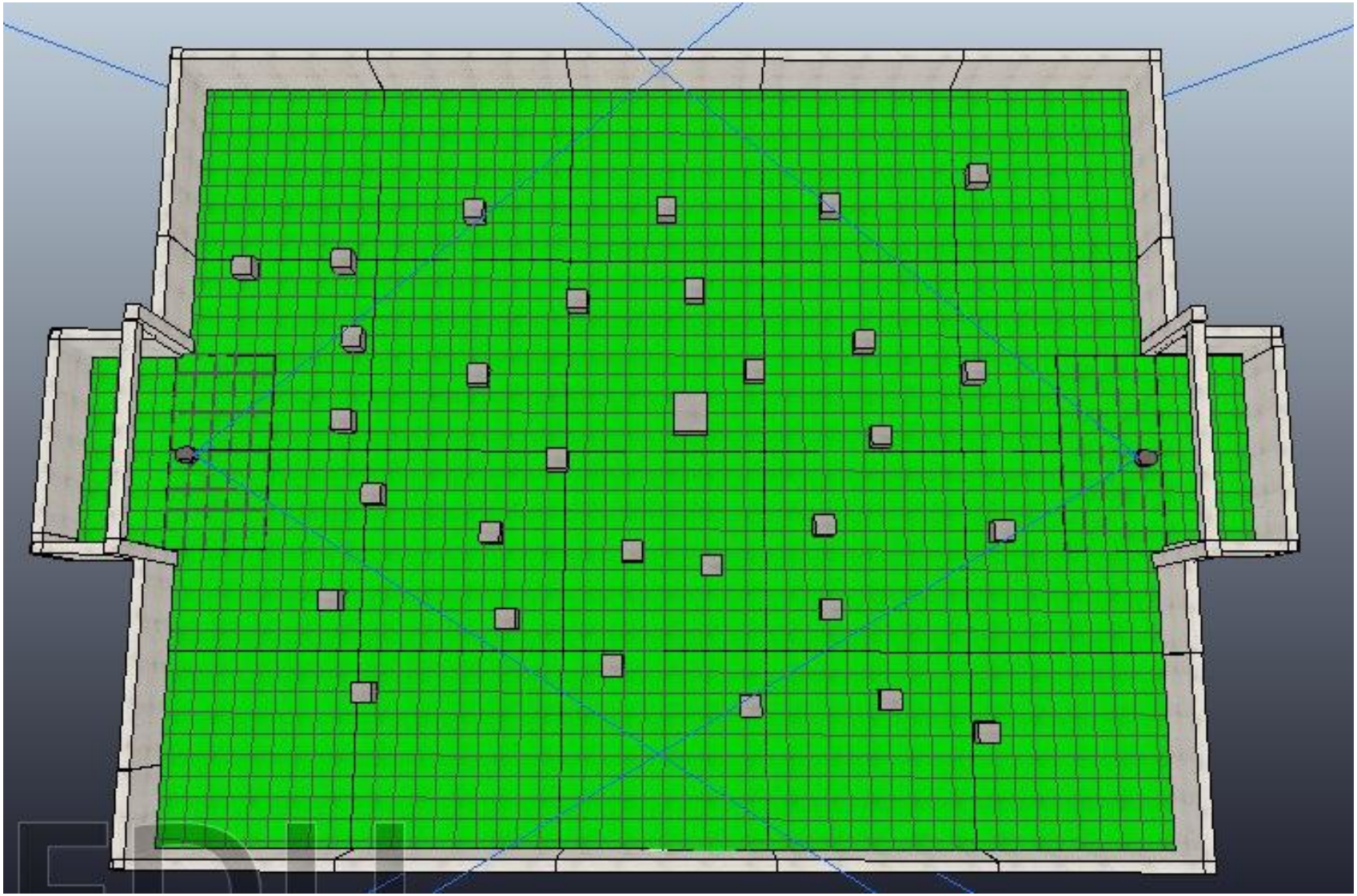


Figura: Cenário com obstáculos.

Trabalhos Relacionados

Trabalhos analisados

- Comum o uso de Lógica Fuzzy em robótica [1] [2]
- Aplicação em cenário real, evitar obstáculos e seguir parede [3]

---

[1] B. Wakileh and K. Gill, "Use of fuzzy logic in robotics, "Computers in Industry, vol. 10, pp. 35–46, 1988.

[2] H. Vashisth and Peng-Yung Woo, "Application of fuzzy logic to robotic control", 1996, pp. 1867–1872 vol.3.

[3] C.-H. Chen, C.-C. Wang, Y. Wang, and P. Wang, "Fuzzy logic controller design for intelligent robots, "Mathematical Problems in Engineering, pp. 1–12, 2017.

Trabalhos Relacionados

Trabalhos analisados

- Vasta é a literatura sobre a utilização de aprendizagem por reforço em robótica [4]
- Apresenta um *benchmark* de algoritmos de aprendizagem por reforço em um cenário real [5]

---

[4] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey" The International Journal of Robotics Research, vol. 32, pp.1238–1274, 2013.

[5] R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots" inCoRL, 2018.

Metodologia

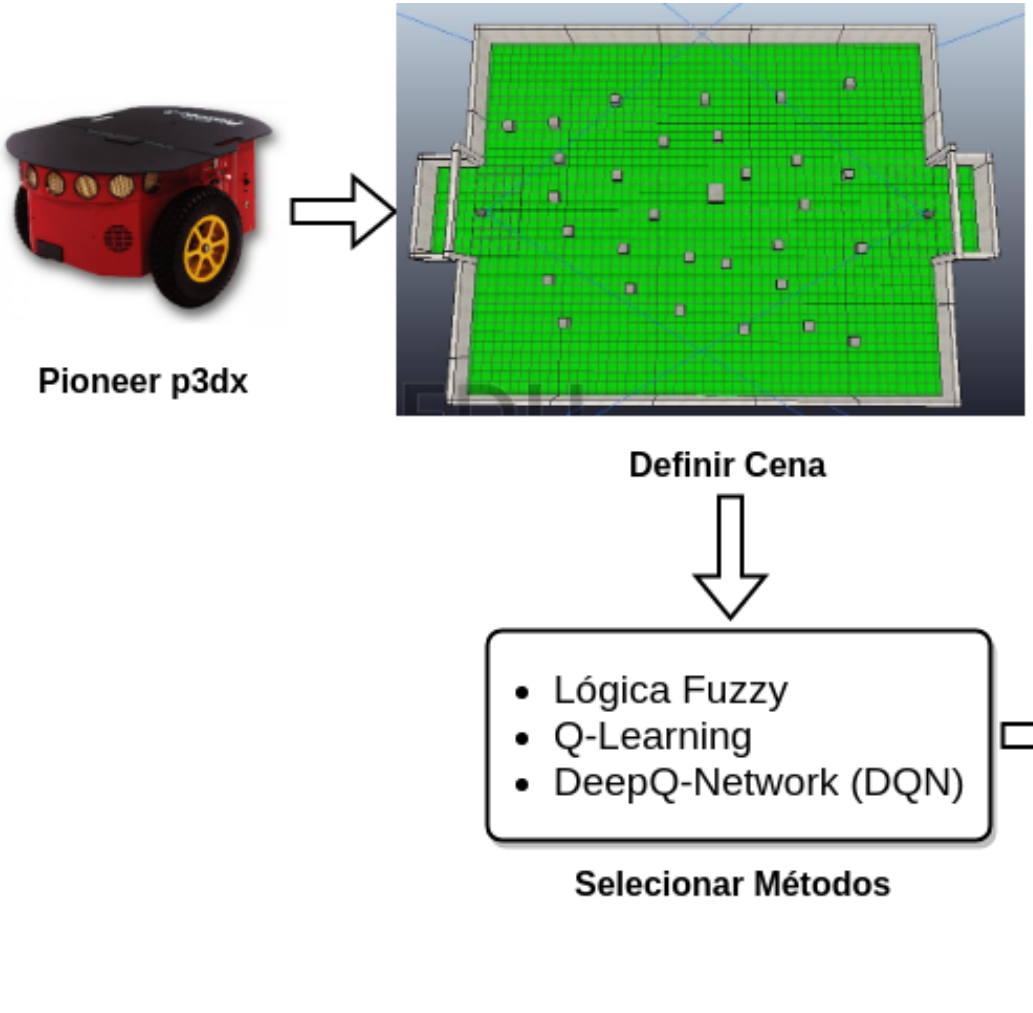


Figura: Metodologia.

Figura: Lógica Clássica x Lógica *Fuzzy*.

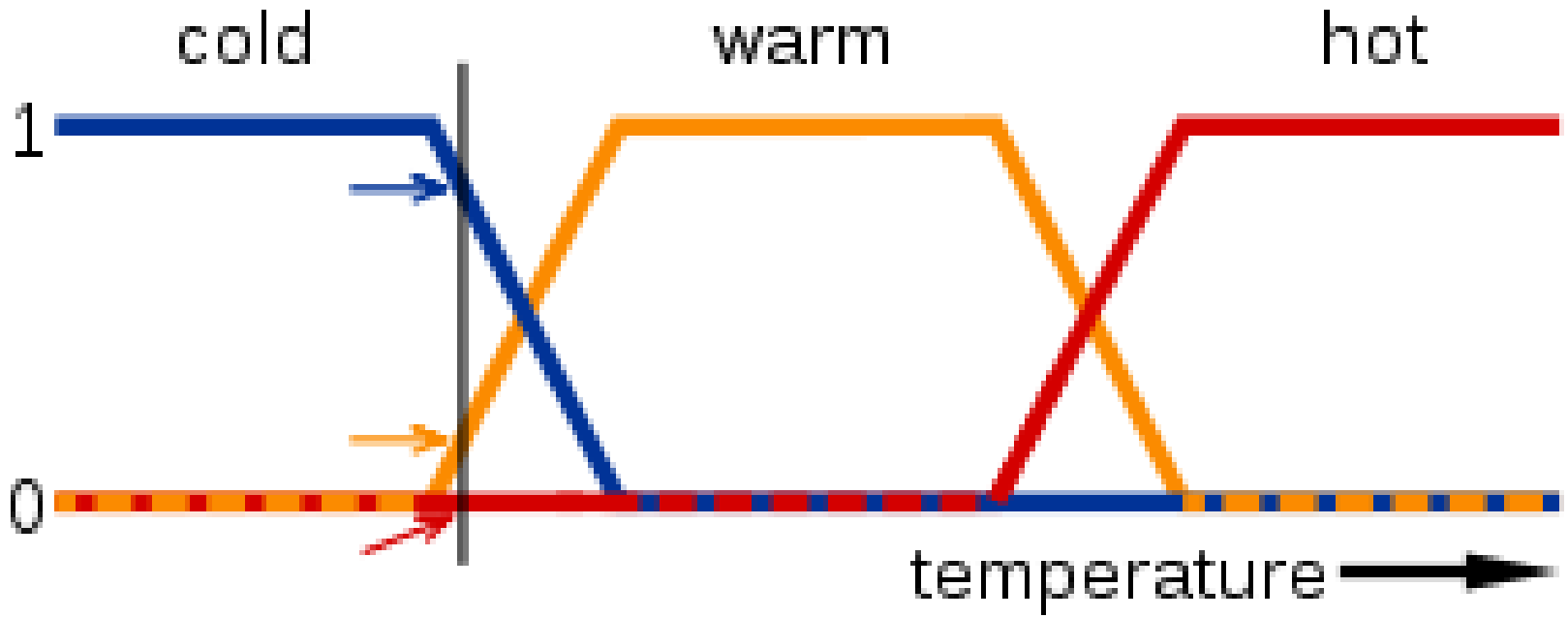


Figura: Exemplo de um sistema *fuzzy*.

Lógica Fuzzy

Regra	Sensor Esquerdo	Sensor Frontal	Sensor Direito	Velocidade	Direção
1	perto	perto	perto	parado	esquerda
2	longe	perto	longe	lento	frente
3	perto	longe	perto	lento	frente
4	longe	longe	longe	rápido	frente
5	perto	perto	longe	lento	esquerda
6	longe	perto	perto	lento	direita
7	perto	longe	longe	rápido	frente
8	longe	longe	perto	rápido	frente

Figura: aa

Ir para o objetivo + Evitar obstáculos − *Q-Learning*

- Treinamos o robô para ir até um ponto objetivo sem obstáculos.

Ir para o objetivo + Evitar obstáculos − *Q-Learning*

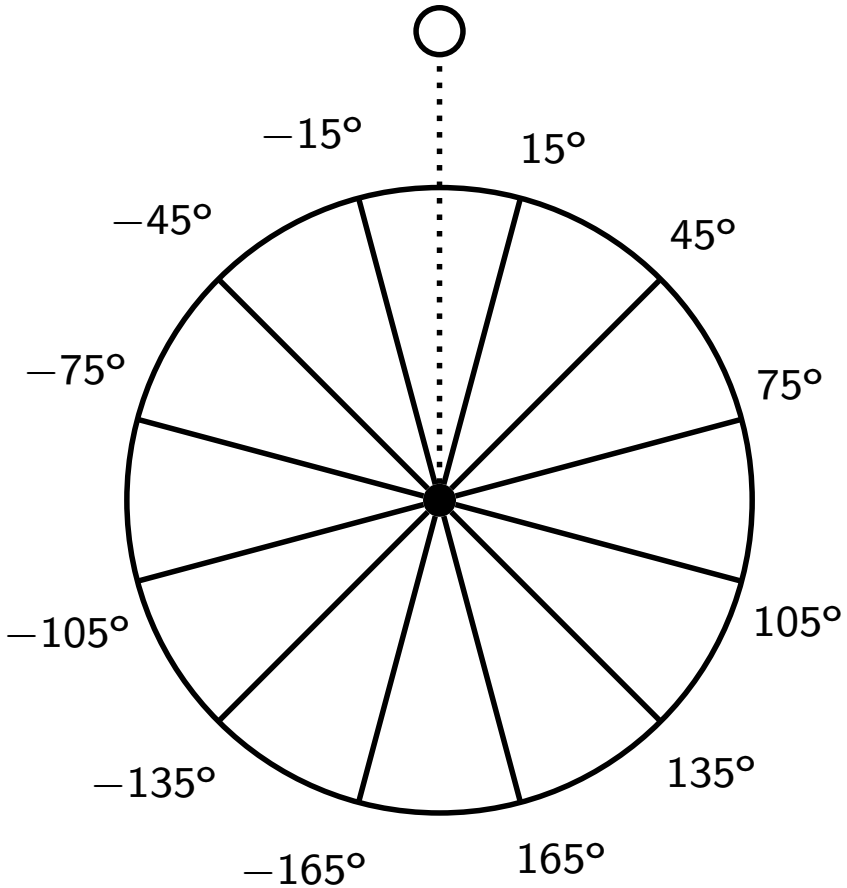
- Treinamos o robô para ir até um ponto objetivo sem obstáculos.
- Treinamos o robô para desviar de objetos que estiverem no seu caminho.

Ir para o objetivo + Evitar obstáculos – *Q-Learning*

- Treinamos o robô para ir até um ponto objetivo sem obstáculos.
- Treinamos o robô para desviar de objetos que estiverem no seu caminho.
- Juntamos os dois treinamentos adequadamente de modo a fazer o robô chegar ao seu objetivo desviando obstáculos.

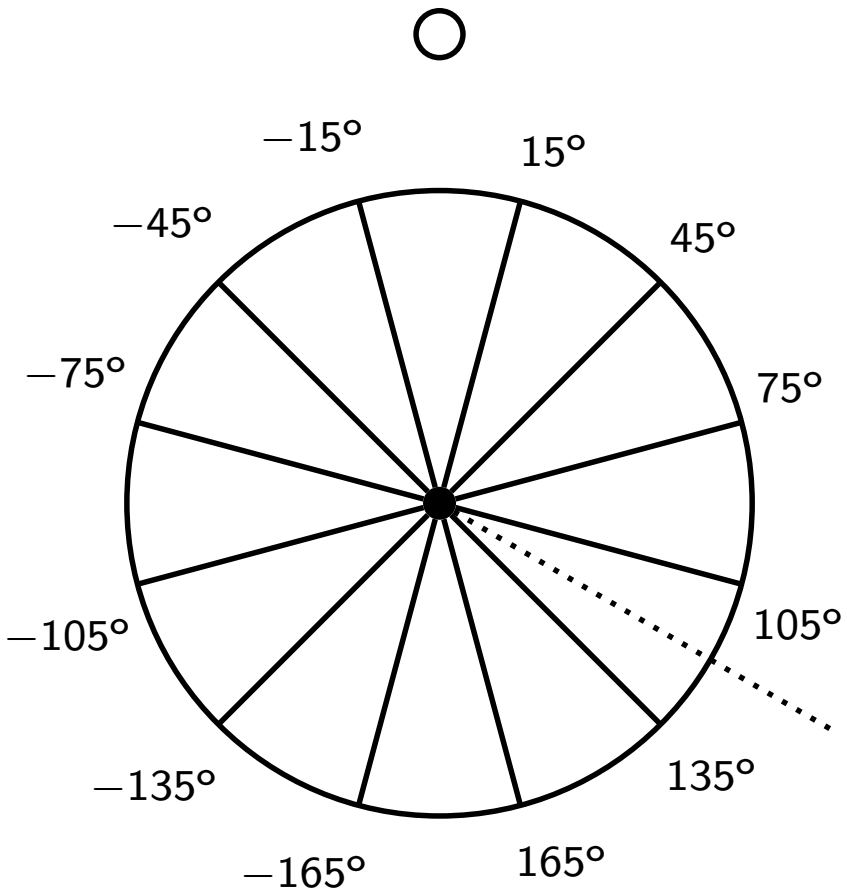
Ir para o objetivo – *Q-Learning*

Conjunto de estados:



Ir para o objetivo – *Q-Learning*

Conjunto de estados:



Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;

Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;

Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Função de recompensas:

Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Função de recompensas:

- $\Delta_a = |old\_angle| - |new\_angle|$;

Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Função de recompensas:

- $\Delta_a = |old\_angle| - |new\_angle|$;
- $\Delta_d = old\_distance - new\_distance$.

.

Ir para o objetivo – *Q-Learning*

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Função de recompensas:

- $\Delta_a = |old\_angle| - |new\_angle|$;
- $\Delta_d = old\_distance - new\_distance$.

A função devolve $\Delta_a + 50 \times \Delta_d$.

Ir para o objetivo – *Q-Learning*

Treinamento:

- Iniciamos com uma probabilidade de movimentação aleatória de 90%.

Ir para o objetivo – *Q-Learning*

Treinamento:

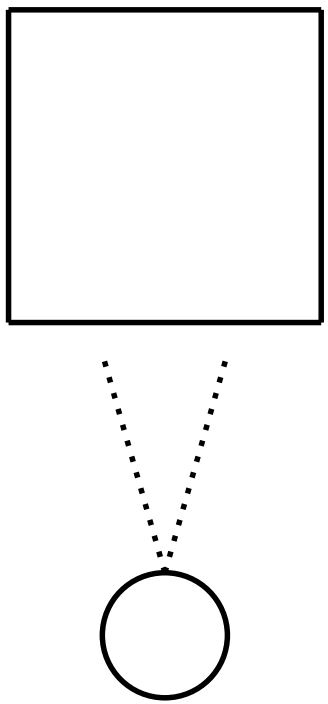
- Iniciamos com uma probabilidade de movimentação aleatória de 90%.
- Iniciamos com uma taxa de aprendizado de 60%.

Ir para o objetivo – *Q-Learning*

Treinamento:

- Iniciamos com uma probabilidade de movimentação aleatória de 90%.
- Iniciamos com uma taxa de aprendizado de 60%.
- Treinamos com um fator de desconto de 0.99.

Evitar obstáculos – *Q-Learning*



4 estados

3 ações

Recompensas:

$+10$ para ir em frente;

0 para virar;

-100 se bater.

$\alpha = 0.5$

$\gamma = 0.8$

Deep Q-Network (DQN)

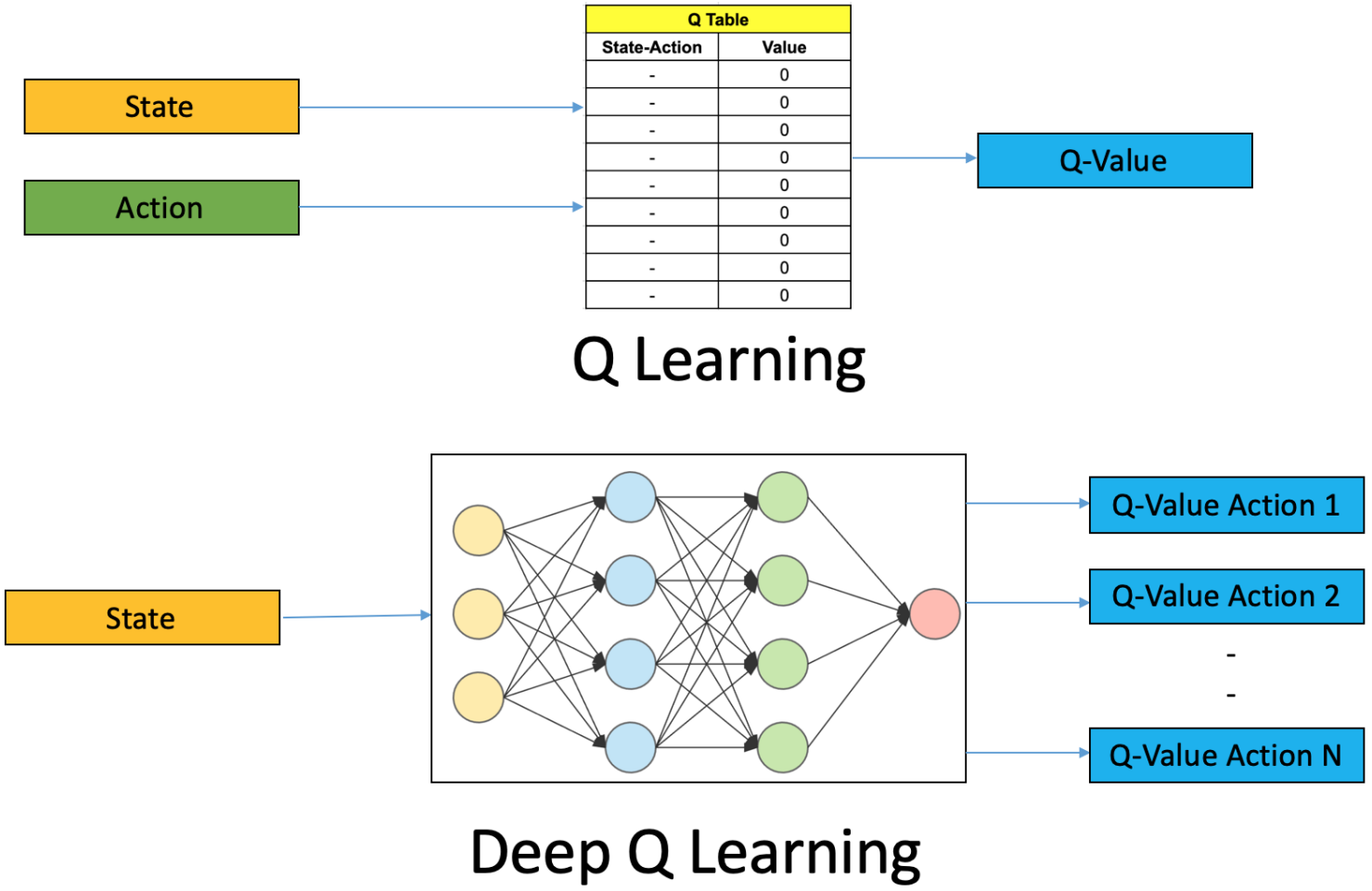


Figura: Comparação *Q-Learning* e Deep Q-Network.

Deep Q-Network (DQN)

Model: "sequential_1"

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 3)	0
dense_1 (Dense)	(None, 32)	128
activation_1 (Activation)	(None, 32)	0
dense_2 (Dense)	(None, 32)	1056
activation_2 (Activation)	(None, 32)	0
dense_3 (Dense)	(None, 32)	1056
activation_3 (Activation)	(None, 32)	0
dense_4 (Dense)	(None, 3)	99
activation_4 (Activation)	(None, 3)	0

Total params: 2,339
Trainable params: 2,339
Non-trainable params: 0

Deep Q-Network (DQN)

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Deep Q-Network (DQN)

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Função de recompensas:

- $\Delta_a = |old\_angle| - |new\_angle|$;
- $\Delta_d = old\_distance - new\_distance$.
- $flag =$ obstáculo detectado

.

Deep Q-Network (DQN)

Conjunto de ações possíveis:

- ir para frente;
- virar à direita;
- virar à esquerda.

Função de recompensas:

- $\Delta_a = |old\_angle| - |new\_angle|$;
- $\Delta_d = old\_distance - new\_distance$.
- $flag =$ obstáculo detectado

A função devolve $\Delta_a + \Delta_d - 2 \times flag$.

Resultados

Figura: Vídeo completo: `https://youtu.be/Q6ETvQM3fdI`.

Conclusões

- Controladores com Lógica *Fuzzy* e *Q-Learning* obtiveram os melhores resultados.
- Entre esses dois o controlador com *Q-Learning* foi melhor.
- Tempo limitado inviabiliza alguns algoritmos de aprendizado por reforço.


T. HANKS