

# Robotic challenge solver using the CiberRato simulation environment

Manuel Couto<sup>[93285]</sup> and Rafael Dias<sup>[95284]</sup>

Universidade de Aveiro  
<https://www.ua.pt/>

**Abstract.** Dentro do ambiente de simulação de robôs CiberRato foram implementados três diferentes tipos de desafios (Control challenge, Mapping challenge e Planning challenge). A linguagem escolhida para desenvolver os diferentes agentes foi python devido à sua fácil implementação e à experiência dos autores nesta linguagem. Foi desenvolvido código para todos os os agentes dos diferentes três desafios funcionarem sem qualquer tipo de problemas. O primeiro desafio tem como objetivo o robô percorrer um mapa com um circuito fechado no menor tempo possível e sem colisões com as paredes à sua volta. O segundo desafio tem como objetivo o robo explorar o mapa na sua totalidade e fazer o seu mapeamento para um ficheiro "mapping.out". O terceiro desafio tem como objetivo também do mapeamento do circuito mas após esse mapeamento terá de mostrar num ficheiro "planning.out" o caminho mais rápido para percorrer o mapa passando por determinadas zonas chamadas de beacons. Para a implementação deste ultimo desafio é necessário a implementação de um algoritmo relacionado com a procura do caminho mais rápido.

**Keywords:** CiberRato · Robot · Agent.

## 1 Introduction

O Ambiente de Simulação de Robôs CiberRato simula o movimento de robôs dentro de um labirinto e este foi utilizado para avaliar os agentes desenvolvidos. O robô simulado está equipado com 2 motores (esquerda e direita) e 3 leds (visiting, returning and finish). Em termos de sensores, inclui um GPS, uma bússola, quatro sensores de obstáculos, um sensor beacon, um sensor ground, e um sensor de colisão. Os sensores disponíveis dependem do desafio a ser resolvido. O robô simulado navega num mapa/labirinto retangular delimitado que pode ser vista como uma matriz bi-dimensional de células de tamanho fixo. Cada célula é um quadrado com comprimento lateral igual a duas vezes o diâmetro do robô. O tamanho máximo da arena é de 7 células de altura e 14 células de largura. As células podem ser referenciadas pela sua posição no labirinto onde a célula (0,0) está localizada no canto inferior esquerdo e a célula (13,6) está localizada no canto superior direito. Um labirinto é definido pela colocação de paredes finas entre células adjacentes. As células alvo são detetáveis pelo sensor ground.

Para o desafio de controlo foi implementado um agente que compara as distancias de todos os sensores às paredes e conforme essa informação desviar-se-á

caso encontre uma parede relativamente próxima de si, percorrendo o mapa todo desviando se das paredes e continuando sempre a seguir a direção pretendida no menor tempo possível.

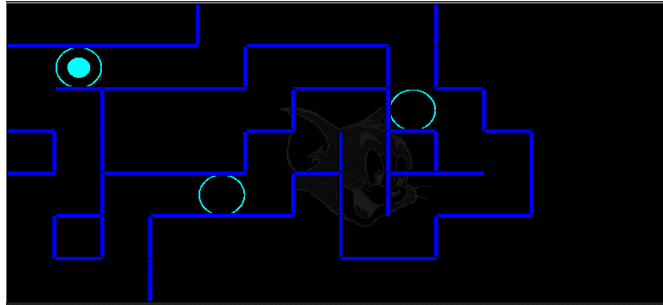
Para o desafio de mapping foi implementado um agente capaz de explorar o circuito e escrever num ficheiro o mapeamento do circuito na sua totalidade. Para efetuar o mapeamento do circuito foi utilizado o algoritmo de busca de caminho A\*.

Para o desafio de planning foi implementado um agente relativamente parecido com o agente de mapping só que neste caso, após o mapeamento de todo o circuito, este irá determinar o caminho mais rápido entre os vários beacons até voltar à posição inicial.

## 2 Challenges

### 2.1 C1 challenge

O objectivo deste desafio centra-se em desenvolver um agente adequado para controlar o movimento de um robô através de um circuito fechado desconhecido o mais rápido possível e sem colidir com as paredes circundantes. A figura 1 foi o exemplo de um labirinto durante a implementação do código neste desafio.



**Fig. 1.** Exemplo de Labirinto do desafio C1

O sensor GPS e a bússola não estão disponíveis neste primeiro desafio ao contrário dos restantes. A ideia por detrás deste desafio foi utilizar os sensores de obstáculos para controlar o movimento do robô no labirinto não o deixando colidir com as paredes fazendo o percurso da forma mais rápida. A pontuação (e portanto a classificação) é baseada no número de células completadas ao longo do circuito fechado e no número de colisões com as paredes.

O código deste agente permite comparar as distancias de todos os sensores às paredes e conforme essa informação desviar-se-á caso encontre uma parede relativamente próxima de si, percorrendo o mapa todo desviando se das paredes e continuando sempre a seguir a direção pretendida no menor tempo possível.

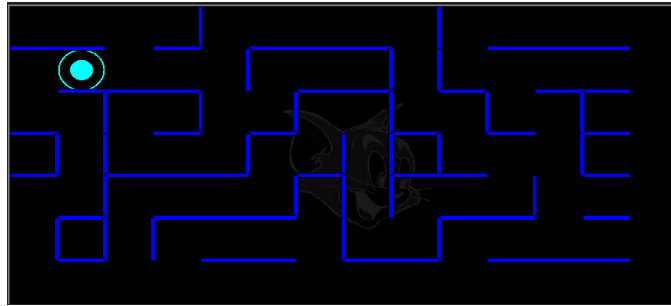
O agente implementado consegue obter um score de 3250, sem colisões, no tempo default de 5000. Para cada volta efetuada sem colisões o score obtido é de 400, ou seja, o número de voltas que o agente faz na totalidade é cerca de 8 ( $3240\%400 = 8,1$ ).



**Fig. 2.** Pontuação e tempo do agente no desafio C1

## 2.2 C2 challenge

O objectivo deste desafio é explorar um labirinto desconhecido, com o objectivo de obter o seu mapa ao imprimi-lo para um ficheiro. A figura 3 mostra um exemplo de um labirinto que foi utilizado durante este desafio.



**Fig. 3.** Exemplo de Labirinto do desafio C2

Sensor GPS e bússola estão disponíveis como foi dito anteriormente, sem ruído. Ao utilizar estes sensores, o agente pode saber, em cada ciclo, onde o robô está posicionado e orientado sobre o labirinto. A navegação poderia ser feita utilizando a abordagem seguida no desafio C1, mas foi mais fácil obter a localização utilizando o GPS e a bússola sem ruído. Colisão com paredes não sofre penalização. Neste desafio a ideia foi de utilizar os sensores de obstáculos para determinar onde estão as paredes em todo o labirinto e para isso foi necessário mover os sensores da esquerda e da direita do robô de  $60^\circ$  para os  $90^\circ$ .

O código deste agente foi implementado com a utilização do algoritmo A\* O agente anda sempre de duas em duas unidades visto que é o tamanho de

uma célula para a outra. Antes do agente executar um movimento deve ser verificado para onde é possível o mesmo se mexer e essa verificação vem através dos sensores. Inicialmente os sensores frontais da esquerda e da direita estão com um ângulo de  $60^\circ$  face ao eixo frontal do agente mas estes foram trocados para um ângulo de  $90^\circ$  o que os faz situar na parte lateral tanto à direita como à esquerda.

O agente tem acesso a uma string de quatro carateres que é armazenada para todas as posições com o conteúdo de "o"s ou "c"s, que significam "open" e "close". O primeiro caratere da string representa se naquela posição o agente já andou para cima, a segunda posição refere-se à direita, a terceira posição à esquerda e a quarta posição para baixo. Com o acesso a esta informação em todas as células, fica mais acessível saber por onde o agente já andou e por onde ele ainda precisa de andar para fazer o mapping da totalidade do circuito.

Ao longo da exploração do agente pelo mapa, este ao saber exatamente onde se encontram as paredes do labirinto com a ajuda dos sensores irá escrever essa informação num ficheiro "mapping.out".

```

- - - - -
|xxxxxx|xxxxxxxx|xxxxxxxx|
- - X - X - - - X X - - X
|xxIxxxxx|xxxxx|x|xxxxxxxx|
X - - - X X - - X - X - - X
|xxx|xxx|xxx|xxx|xxx|xxx|xxx|
- X X - X - X X - X - X - X
  |x|xxxxx|xxx|x|  |xxxxx|xxx|
- X - - - X - X - - X X - X
|xxx|xxxxxxxx|x|x|xxxxx|xxxxx|
X - X - - - X X X - - X - X
|x|  |x|xxxxxxxx|xxx|xxxxxxxx|
X - X X - - X - - X - - - X
|xxxxxxxxxxxxxxxxxxxxxxxxxxxx|
- - - - -

```

**Fig. 4.** Ficheiro output do desafio C3

O tempo necessário para o agente fazer o reconhecimento do mapa e escreve-lo no desejado output é cerca de 2900 em unidades de tempo ( 5000-2100 = 2900).



**Fig. 5.** Tempo ao finalizar o desafio C2

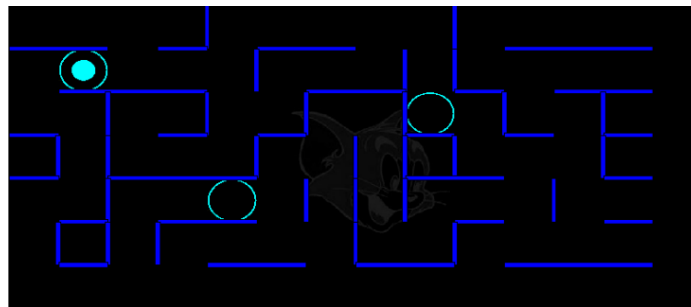
O score obtido após a finalização do desafio comparando o "mapping.out" fornecido pelo programa do simulador com o "mapping.out" criado pelo nosso agente foi de 3110.

```
miinds@miinds:~/Desktop/RMI/ciberRato-simulation-challenges$ awk -f simulator/mapping_score.awk simulator/mapping.out pClient/mapping.out
mapping score:3110
```

**Fig. 6.** Pontuação do desafio C2

### 2.3 C3 challenge

O objectivo deste desafio é explorar um labirinto desconhecido a fim de localizar um número de pontos-alvo e calcular um melhor caminho fechado que permita visitar esses pontos-alvo, começando e terminando no ponto de partida. A figura 7 mostra um exemplo de labirinto adequado a este desafio, que estará disponível durante o desenvolvimento.

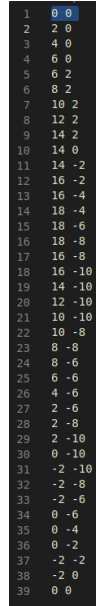


**Fig. 7.** Exemplo de Labirinto do desafio C3

O sensor GPS e a bússola estão disponíveis, sem ruído. Ao utilizar estes sensores, o agente pode saber, em cada ciclo, onde o robô está posicionado e orientado no labirinto. Assim, a localização no labirinto não será um problema. A navegação pode ser feita utilizando a abordagem seguida no desafio C1, mas

é mais fácil de o fazer utilizando o GPS e a bússola sem ruído. Foi-nos ainda disponibilizado a variável `nBeacons` e `ground` que nos ajudam a detectar os "beacons". A colisão com paredes não será penalizada. A ideia por detrás deste desafio é explorar o labirinto apenas até que os pontos-alvo(Beacons) sejam localizados e de seguida identificar o melhor circuito fechado possível começando e acabando na posição inicial, passando por todos os pontos-alvo. Por conseguinte, o labirinto não precisa necessariamente de ser explorado em a sua totalidade.

O código deste agente foi relativamente idêntico ao código do agente de mapeamento. Após o mapeamento de todo o circuito e com a ajuda do algoritmo A\* foi possível calcular o caminho mais rápido desde o ponto inicial passando pelos beacons e voltando ao mesmo. Este caminho foi escrito para um ficheiro "planning.out" com as devidas coordenadas para a resolução da volta ao circuito pelo caminho mais curto.



1	0 0
2	2 0
3	4 0
4	6 0
5	6 2
6	8 2
7	10 2
8	12 2
9	14 2
10	14 0
11	14 -2
12	16 -2
13	16 -4
14	18 -4
15	18 -6
16	18 -8
17	16 -8
18	16 -10
19	14 -10
20	12 -10
21	10 -10
22	10 -8
23	8 -8
24	8 -6
25	6 -6
26	4 -6
27	2 -6
28	2 -8
29	2 -10
30	0 -10
31	-2 -10
32	-2 -8
33	-2 -6
34	0 -6
35	0 -4
36	0 -2
37	-2 -2
38	-2 0
39	0 0

**Fig. 8.** Ficheiro output do desafio C3

### 3 Conclusion

Com este trabalho conseguimos desenvolver e melhorar competências na área da simulação de robôs.

Os objetivos esperados para cada um dos desafios foram alcançados na sua totalidade. Todos os agentes implementados funcionam na totalidade para o objetivo de cada desafio.

Em termos de melhoramento do código implementado para os diferentes agentes, realçamos que o código do agente de controlo poderia ser melhorado para uma versão onde o agente completasse as voltas ao circuito em menos tempo. Face aos restantes desafios, o agente conclui os mesmos de uma maneira acertada para qualquer tipo de labirinto e com um tempo que achamos ser bastante positivo.

### References

1. “Principles of Robot Motion: Theory, Algorithms, and Implementations”, Howie Choset et al., MIT Press, Boston, 2005
2. “Introduction to Autonomous Mobile Robots”, Second Edition, Roland Siegwart et al., MIT Press, 2011
3. “Artificial Intelligence: A Modern Approach”, 3rd edition, Russel and Norvig, Pearson, 2009