# Robótica Móvel e Inteligente - Assignment 1

Dário Matos (89288), Pedro Almeida (89205)

Universidade de Aveiro

## 1 Introdução

Para a realização deste *assignment* foi utilizado o ambiente de simulação CiberRato. Os agentes foram desenvolvidos em Python devido à familiarização com a linguagem e ao rápido desenvolvimento de código.

Para o desafio C1 (desafio de controlo), o objetivo é percorrer um circuito fechado desconhecido o mais rápido possível evitando ao máximo colisões. A estratégia passou por ler os valores dos sensores de aproximação e foi desenvolver código que garantisse um movimento essencialmente seguro.

Quanto ao desafio C2 (desafio de mapeamento), o objetivo é explorar uma mapa desconhecido e fazer o seu mapeamento. Para isto, foi implementada uma estratégia o mais eficaz possível, que permitisse percorrer o mapa de forma total e dentro do tempo disponível. É assim necessário um algoritmo que permita encontrar caminho para destinos desejados. Durante a deslocação do agente, ia sendo analisado o que estava em seu redor de modo a permitir fazer o mapeamento do mapa desconhecido. Neste desafio foi disponibilizado GPS e bússola, pelo que a deslocação no mapa foi mais fácil de controlar do que no desafio C1.

Para o desafio C3 (desafio de planeamento), o objetivo é explorar um mapa desconhecido de forma a encontrar *targets* espalhados pelo mesmo e calcular o melhor circuito fechado que passe por todos. Para a exploração do mapa foi reaproveitado o trabalho desenvolvido no desafio C2. Já para o cálculo do melhor circuito fechado, foi utilizado um algoritmo que permite resolver o problema do caixeiro viajante.

## 2 Desenvolvimento

### 2.1 Desafio C1

Para o desafio C1 não foi utilizada nenhuma "máquina de estados"para a deslocação do agente, na medida em que o comportamento do agente era de um movimento constante mas seguro, de forma a completar voltas ao circuito sem colisões nem perdas de tempo.

A definição dos movimentos foi feita à base de *if*s. Numa primeira abordagem de resolução deste desafio, foram utilizados os sensores para reger os movimentos para se afastar de obstáculos depois de estar perto dos mesmos. Rapidamente se percebeu que não era a melhor solução, uma vez que muitas vezes o agente ficava demasiado perto das paredes e ter de inverter o caminho dispendia bastante

tempo. Numa primeirda abordagem, foi também desenvolvido um método que permitia detetar se o robo seguia em sentido contrário ao esperado. Isto era possível uma vez que pelo circuito fechado há *targets* numerados o que permitem saber se a ordem de passagem por estes mesmo é a correta. Mais tarde este método foi removido uma vez que o agente foi implementado de uma forma a não ter este problema.

Numa segunda abordagem, foi acrescentado à estratégia anterior a "intenção"de apenas seguir na direção onde não há obstáculos próximos", e ajustar o menor possível a direção, aquando da proximidade de uma parede.

Para a concretização deste planeamento foram efetuadas decisões maioritariamente baseadas nas distâncias dos sensores aos obstáculos. Os valores destas distâncias "limite"foram obtidas por tentativa erro e pequenos cálculos. Depois de o agente ser capaz de completar o circuito fechado sem colisões, foi procurada uma estratégia de modo a também identificar caminhos retilíneos. Nestes momentos é aumentada a velocidade do agente de forma segura o que permite também completar mais voltas ao circuito fechado.

### 2.2 Desafio C2

Para o desafio 2, foi implementada uma uma máquina de estados, com o seguintes estados:

- *go ahead* - Estado de movimento linear até atingir um (*target*) pré-calculado
- *rotate left* - Estado de movimento de rotação ($+90^{o}$)
- *rotate right* - Estado de movimento de rotação ($-90^{o}$)
- *sbinalla* - Estado de movimento de inversão de sentido ($+180^{o}$)
- *mapping* - Estado de decisão. Cálculo do próximo *target* e mapeamento

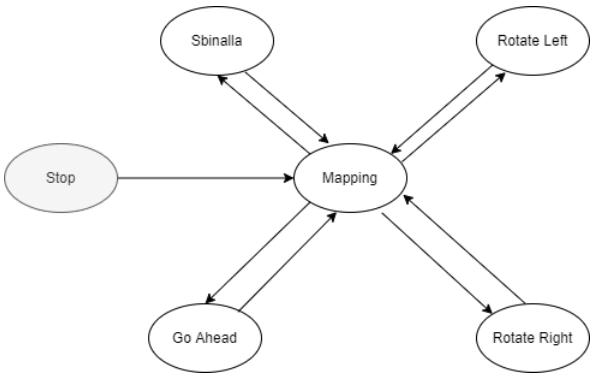


**Figura 1.** Máquina de estados implementada

Ao iniciar, o agente encontra-se no estado *Mapping*. Este estado de decisão é o

estado mais importante, sendo responsável tanto por calcular o próximo movimento do agente como fazer o mapeamento do mapa.

**Cálculo do próximo target:** Sempre que o agente está numa posição que ainda não esteve, analisa o seu redor e através dos sensores de proximidade retira em que direções tem o caminho livre. Das possíveis direções livres encontradas escolhe uma como próximo objetivo com as seguintes prioridades: frente, direita, esquerda e só depois atrás. As restantes direções livres são guardadas para uso futuro. Quando o agente está numa posição já visitada mas que ainda tem outra direção que não foi explorada é essa selecionada como próximo objetivo. Quando o agente se encontra numa posição em que todo o seu redor são posições já visitadas, é calculado o caminho mais curto através do algoritmo *astar* para todas as posições não visitadas conhecidas e é selecionado como próximo objetivo a posição com o menor caminho para lá chegar. Ao entrar no estado *mapping*, caso já haja um caminho definido, o próximo objetivo é sempre o próximo "passo"nesse caminho. Do mesmo modo, caso já haja um objetivo calculado esse é mantido.

**O deslocamento** é feito através da seguinte fórmula:

```
rot = k * (m - r)

r_power = lin - (rot / 2)
l_power = lin + (rot / 2)
```

**O mapeamento** é efetuado no momento em que calcula o próximo *target*: primeiramente, mapeia a sua posição atual como válida (devido a ter lá chegado); seguidamente, conclui se existem paredes nas posições adjacentes à sua, através da análise das distâncias medidas pelos sensores; finalmente, após saber que paredes o rodeiam, conclui acerca das células anexas disponíveis e mapeia a posição "de transição"(de abcissa ou ordenada ímpar, entre células).

No que toca à escrita do mapa num ficheiro de texto, é utilizado um dicionário que guarda as posições e o símbolo correspondente a imprimir. Depois de o agente percorrer a totalidade do mapa, inicia-se a escrita. Todas as posições são guardadas com um de offset de mais 28 na abcissa e subtraídas a 14 na ordenada, de forma a suportar o movimento de duas em duas células por parte do agente. Assim, o mapa terá o dobro do tamanho, e as coordenadas negativas existentes no programa serão retratadas corretas relativamente ao resto do mapa, ao escrever num ficheiro.

### 2.3 Desafio C3

Neste desafio de planeamento, foi reaproveitado o código desenvolvido no desafio C2 que permitia explorar o mapa desconhecido. Durante a exploração do mapa,

foi guardado a localização dos *targets* encontrados. Depois da exploração total do mapa e consequentemente da descoberta de todos os *targets* foi calculado. Com a posição de todos os *targets* foi calculado o caminho de um *target* para todos os outros através do algoritmo *astar*, guardadando os custos associados (número de posições). Posteriormente foi construído um grafo e este mesmo foi passado para um algoritmo que permite resolver o problema do caixeiro-viajante, fazendo todas as permutações possíveis de caminhos e retornando o mais curto. Finalmemte, foi escrito o *output* num ficheiro.

## 3 Conclusão

Após realizar os três challenges, podemos concluir que, embora satisfaçam os objetivos pretendidos, existe espaço para melhorar. Para o C1, retira-se que a utilização de máquina de estados poderia ser benéfica para definição de movimentos mais úteis. Quanto ao C2, conclui-se que a utilização do sensor de GPS é crucial para a visão geral do mapa. Quanto ao C3, uma possível melhoria poderia passar pela utilização de outro algoritmo de procura de melhor caminho, ou de uma implementação mais eficiente do mesmo.

## Referências

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017