CiberRato Assignment 2

Rodrigo Santos and Rui Santos

Universidade de Aveiro, 3810-193 Aveiro

rodrigo.l.silva.santos@ua.pt, ruipedro99@ua.pt

Abstract. Para o desenvolvimento do Assignment 2 foram aplicadas diferentes estratégias para atacar o maior desafio do projeto, de localizar o robô no labirinto com precisão. A primeira estratégia adotada tem por base a fórmula de movimento do robô, com a qual sabendo a potência que será introduzida nos motores é possível calcular a posição final do robô. Após testes, este método provou ser insuficiente, pois a acumulação de erro era muito grande. Para a complementar esta estratégia foi desenvolvida outra com base nos sensores do robô, na qual a sua posição é ajustada consoante os valores lidos pelo sensor dianteiro. Por fim, e para obter uma previsão final mais fidedigna, foi utilizado o resultado de ambas as estratégias e calculada uma posição prevista final combinando as duas previsões, em que a estratégia baseada na fórmula de movimento possui um peso de 20% e a estratégia baseada nos sensores tem um peso de 80%. Assim foi possível desenvolver e concluir com sucesso o desafio proposto.

Keywords: CiberRato, Labirinto, Estratégia, Motores, Sensores, Bússola, Robô, Beacons.

1 Introdução

Neste relatório serão abordadas as diferentes estratégias aplicadas para a concretização do Assignment 2, da unidade curricular Robótica Móvel Inteligente, do 5º ano do Mestrado Integrado em Engenharia de Computadores e Telemática, da Universidade de Aveiro.

Este projeto consiste no desenvolvimento de um agente que comanda um robô simulado que se encontra num labirinto desconhecido e pretende cumprir um conjunto de desafios. O robô terá de navegar pelo labirinto sem bater nas paredes (uma vez que é penalizado por isso), mapeá-lo e escrever o resultado num ficheiro. O robô terá também de encontrar um conjunto de *beacons*, calcular o caminho fechado mais curto entre todos estes pontos e escrever o resultado num ficheiro. Para se movimentar, o robô dispõe de 2 motores que sofrem de ruído. Para se localizar, tem acesso a uma bússola com ruído e 4 sensores também com ruído.

Tendo em conta os diferentes objetivos do desafio proposto foram aplicadas várias estratégias, algumas das quais provenientes do Assignment 1 da mesma unidade curricular.

A primeira seção aborda o principal problema deste desafio, que é o de localizar o robô no labirinto sem o auxílio do GPS, uma vez que este se encontra indisponível. Para tal foram utilizadas técnicas de previsão da posição atual, com base nas fórmulas de movimento do robô e também com os valores lidos dos sensores do robô.

Na segunda seção é abordado o nosso método de avaliação do agente, e são discutidos os resultados obtidos nesta avaliação. O nosso agente foi executado múltiplas vezes ao longo do seu desenvolvimento para garantir o seu correto funcionamento e para que fossem feitos ajustes às técnicas utilizadas, principalmente, na localização do robô de acordo com a sua performance.

2 Estratégia

Tal como foi feito no Assignment 1, e tendo em conta o objetivo de mapear todo o labirinto, a estratégia aplicada segue o conceito de percorrer o mapa de 2 em 2 unidades de GPS, parar, fazer o reconhecimento das paredes e espaços livres à volta do robô e decidir o que fazer de seguida.

2.1 **Mapa**

Na criação do mapa, foi utilizado um array bidimensional com 55 unidades de comprimento e 27 unidades de largura inicializado a 0. Para a determinação destes valores, foi necessário considerar que o robô poderia surgir em cada uma das pontas do mapa 29x15. Na sua construção, foram utilizados os valores obtidos pelos sensores do robô que foram posicionados a 0, 90, -90 e a 180 graus. Quando estes valores fossem superiores a 1, seria atribuída à posição correspondente do mapa 55x27 o símbolo de uma parede que podia ser "|" ou "-" dependendo da sua posição relativamente ao robô. Se os valores fossem inferiores a 1, seria atribuído um espaço (" ") à posição correspondente, simbolizando uma posição livre inexplorada. Foram também atribuídos "X" a todas as posições em que o robô passou. Apesar do robô não andar para trás, a utilização do sensor traseiro mostrou-se relevante na posição inicial, em que este regista por exemplo um espaço, não correndo assim o risco de se "esquecer" que existe uma posição inexplorada.

Para armazenar o mapeamento feito no array bidimensional criado, é necessário traduzir as coordenadas GPS, em coordenadas do array, para este efeito foi criada uma variável global que guarda o fator da escala a ser usada. Tendo por base as coordenadas iniciais e sabendo que o array começará a ser escrito na posição (27, 13), a escala pode ser calculada subtraindo a posição inicial do array bidimensional à posição GPS inicial. Deste modo, numa função à parte é calculada a posição de cada coordenada GPS a traduzir para coordenadas no array de armazenamento, subtraindo à coordenada GPS a escrever, o valor da escala.

Para evitar com que o robô fique preso e não se obtenha qualquer resultado após *timeout* a escrita do mapa no ficheiro é feita a cada 10 paragens do robô.

Todo este processo foi desenvolvido no Assignment 1 e adaptado para o presente desafio.

2.2 **Movimento**

Tal como no Assignment 1, durante o movimento do robô os motores têm ruído, podendo assim haver alterações na trajetória desejada do robô. Para contornar este problema, foi criada uma função que verifica a direção de movimento pretendida. Se os valores de direção da bússola não estiverem dentro dos limites de -1 e +1 graus do ângulo ideal para o tipo de movimento atual, é feita uma correção de acordo com o desvio que o robô tem em relação à direção do movimento, certificando assim que o robô andará sempre na direção desejada.

O deslocamento do robô tem em conta a sua inércia, portanto, foi necessário adicionar um mecanismo que o previne de exceder a posição destinada. Tendo isto em conta, no movimento simples de 2 unidades de deslocamento, o robô dá o valor máximo de 0.15 a ambos os motores até a distância ao destino ser inferior a 0.15. Posteriormente, quando a distância ao destino for menor do que 0.4, os valores dados aos motores são bastante mais reduzidos (0.04). Quando a distância ao destino for inferior a 0.04 então o robô sai do estado de deslocamento e aplica o negativo de 80% da potência teórica do último ciclo aos motores, para que este consiga parar no local pretendido.

A rotação do robô segue a mesma estratégia do movimento simples, diminuindo o valor enviado aos motores quando o ângulo se aproxima do pretendido.

2.3 **Algoritmo de Pesquisa**

Após o robô fazer a atualização do mapa com base nos sensores, deve prioritizar o deslocamento na direção dos espaços (“ ”) que representam células livres inexploradas, mais especificamente nos espaços à sua frente para não perder tanto tempo com tantas rotações. No caso do robô não ter um espaço a 2 unidades de distância, deve aplicar o algoritmo de pesquisa A* para obter o caminho mais curto entre a posição atual e os diversos espaços no mapa e escolher o espaço cujo caminho é o mais curto. Quando não existirem mais espaços no mapa significa que a sua elaboração está concluída.

Na aplicação do algoritmo de pesquisa A*, o mapa é transformado numa matriz de zeros e uns em que os zeros representam as posições onde o robô se pode deslocar e os uns representam as posições onde não se pode deslocar. Na elaboração deste mapa de input ao algoritmo de pesquisa foi definido que os zeros correspondiam às células onde o robô já esteve (“X”) e espaços (“ ”) e os uns correspondiam aos restantes símbolos do mapa. Após receber o percurso do caminho mais curto, este é transformado para uma *stack* de movimentos e o robô entra no estado de “consumo da *stack*”. Enquanto a *stack* tiver movimentos, o robô irá consumi-los e percorrer o percurso até chegar ao seu destino.

Este procedimento descrito nesta seção é repetido até que o mapa esteja completamente explorado, ou seja, não haja mais espaços por explorar no mapa. Por fim, é impresso o mapa obtido no ficheiro.

2.4 **Algoritmo de Planeamento**

Foi adicionada lógica que permite ler o valor de *measures.ground* e armazenar as posições dos pontos a encontrar, quando no decorrer da exploração do mapa o valor de *measures.ground* é diferente de -1 (células normais), ou 0 (Ponto Inicial).

A partir do momento em que todos os *beacons* são encontrados, é calculado o melhor caminho de Ponto Inicial → Beacons → Ponto Inicial sempre que o robô pára para "pensar", até o mapa ser totalmente explorado. O cálculo do melhor caminho é feito com recurso a um algoritmo de pesquisa A* diferente do utilizado para a exploração uma vez que este não devolvia o melhor caminho entre os pontos. O caminho é escolhido analisando todas as sequências possíveis de pontos, e calculando o melhor caminho para cada uma, e de seguida o caminho com menor número de células de entre os calculados é escolhido. Quando todas as células são encontradas e o mapa é totalmente explorado, os passos do caminho são escritos num ficheiro *.path* e o robô deverá voltar à posição inicial e executar o *finish()*.

3 Técnicas de localização

Ao contrário dos challenges 2 e 3 propostos no Assignment 1, para este desafio o robô não possui acesso ao GPS, logo para precisar a sua posição a cada instante do seu movimento é necessário prevê-lo.

Para este efeito foram utilizadas algumas estratégias diferentes e, utilizando a previsão calculada em cada uma, é obtida uma previsão final da posição.

3.1 **Fórmula de movimento do Robô**

A primeira estratégia utilizada consistia em monitorizar a potência de saída tendo em conta a inércia. Isto foi feito através da fórmula descrita na seguinte figura:

$$out_t = \frac{in_i + out_{t-1}}{2} * \mathcal{N}(1, \sigma^2)$$

Fig. 1. IIR Filter Formula

Sabendo a potência a aplicar em cada motor do robô é possível prever qual a distância que o robô vai percorrer e, utilizando a direção fornecida pela bússola, é possível saber a que posição o robô irá chegar.

Para guardar o valor da potência de saída (out(t)) foi criada uma variável global para cada motor, bem como a função *update_previous_motors()* chamada sempre que o robô executa o *driveMotors()* e que atualiza as variáveis globais da potência de saída através do *IIR Filter* descrito na *figura 1*. O ruído apresentado pela fórmula é dispensado, não sendo incluído nos cálculos.

Foi criada a função *position_estimator()* que é chamada a cada tick e que é responsável por atualizar a variável global *est_pos* relativa à coordenadas estimada do robô no mapa. Nesta função é adicionado o valor da potência de saída à coordenada

estimada de acordo com o movimento que o robô descreve. Uma vez que são guardados os valores de potência de saída para os 2 motores e estes nem sempre são os mesmos (quando o robô ajusta a orientação de acordo com a bússola, por exemplo), é feita a média da média entre ambos os valores e o valor mais baixo. Esta estratégia levou algum tempo a acertar e chegou-se a esta conclusão quando se percebeu que o robô "pensou" que andou mais do que a distância real.

No entanto, esta não pode ser a única fonte de previsão devido ao ruído nos motores, o que leva a que a posição final calculada não seja a posição real atingida. A direção do robô, é mantida através da bússola, tal como nos challenges do Assignment 1, não permitindo que o robô se desvie muito de uma margem de ângulo definida.

3.2 **Sensores do Robô**

Para complementar a estratégia de previsão utilizando a fórmula de movimento do robô, foram utilizados os valores lidos pelos sensores. Com a utilização da primeira estratégia, reparou-se que o robô ficava um pouco atrás da posição que ele previa estar, isto acontece devido à acumulação de ruído nos motores. Assim, para combater este erro, optou-se por ajustar a posição do robô quando uma parede próxima se encontra à sua frente.

Quando uma parede em frente é detectada a menos de 1 unidade, é estimada a posição do robô de acordo com um conjunto de condições. É utilizada a função *round_base()* para arredondar a atual posição (estimada pela fórmula anteriormente referida) para um múltiplo de 2 (uma vez que as coordenadas das células destino são múltiplas de 2) que será a posição que o robô deve atingir e que dista 0.4 unidades da parede. É feita a diferença entre 0.4 e o inverso do valor lido pelo sensor dianteiro e sendo assim calculada a diferença da distância do robô ao centro da célula destino. Finalmente, deverá ser somado (ou subtraído dependendo do movimento do robô) esta diferença à posição destino, tendo assim a posição do robô estimada através dos sensores.

Uma dificuldade que surgiu foi a das paredes de fora do labirinto não terem espessura, distando assim 0.5 unidades do centro da célula em vez de 0.4. Para isso, na elaboração do mapa, o robô regista a diferença entre as coordenadas máxima e mínima de cada eixo. Quando as diferenças corresponderem às dimensões do mapa (-1) o robô passa a reconhecer que uma parede cuja coordenada seja uma máxima ou mínima de cada eixo corresponde uma parede exterior e, portanto, deve aplicar 0.5 na fórmula do cálculo da posição em vez de 0.4.

Outro problema encontrado ao testar o agente surgia quando o robô precisava de virar em posições que não tinham parede em frente, ou seja, a sua posição não era ajustada antes de iniciar o movimento de rotação. Assim foram utilizados os sensores laterais e, caso o seu valor fosse superior a 6.3, significa que o robô poderia colidir ou haver interferência com o sensor frontal, fazendo com que o robô estime a sua posição erradamente. Nestas situações o movimento é ajustado para que o robô se afaste da parede, tentando assim mantê-lo centrado.

3.3 Previsão final

A posição final estimada é calculada através da junção entre as duas estratégias anteriormente referidas, dando um peso de 80% à localização pelos sensores da parede frontal e 20% à localização obtida pela fórmula da *figura 1*.

À medida que o robô faz o seu percurso o erro provocado pelo ruído do motores vai crescendo, principalmente em situações onde o robô tem de percorrer uma grande distância em linha reta. Com este balanceamento de métodos, o erro acumulado pode ser controlado com o auxílio do sensor frontal utilizado no 2º método.

4 Resultados

Após diversos testes e ajustes verificou-se que o robô raramente batia e completava corretamente os diversos *challenges* testados usando vários beacons. Assim, terminava o *challenge 4* com 800 pontos restantes como se pode ver na *figura 2*:

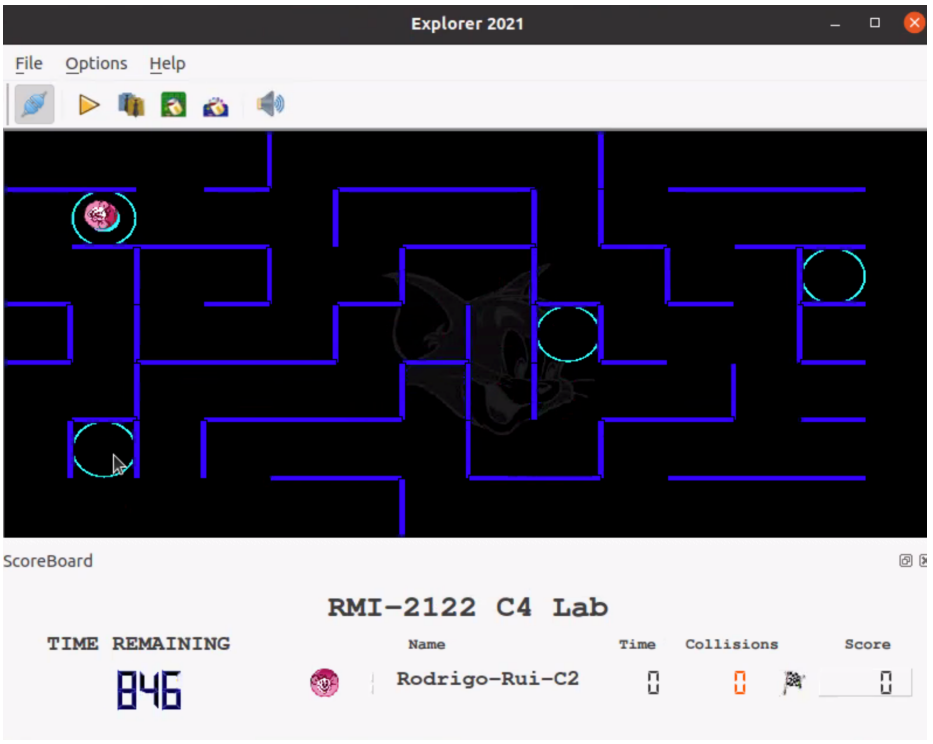

Fig. 2. Finish Challenge 4

Através da ferramenta *Awk*, que compara o conteúdo dos ficheiros *.map* e *.path* gerados pelo agente, foi testado o score obtido. Este foi sempre constante a cada teste e pode ser verificado pelas *figuras 3 e 4.*

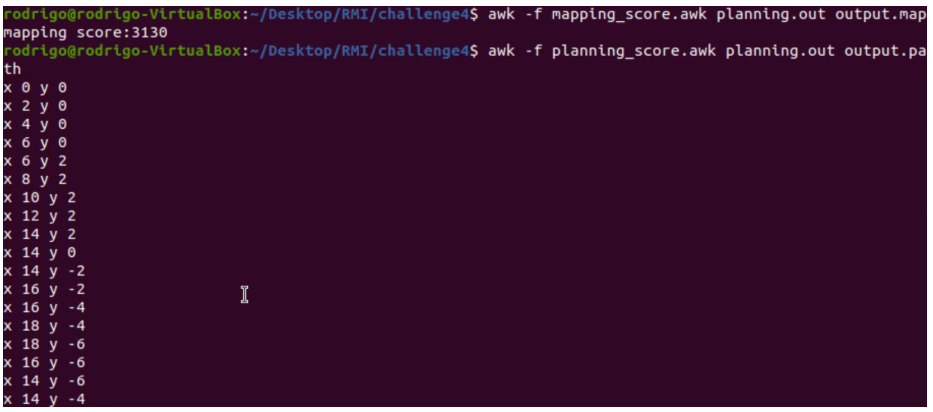

Fig. 3. planning_score e mapping_score relativos ao Challenge 4

Fig. 4. planning_score relativo ao Challenge 4

5 Conclusão

Com o desenvolvimento deste trabalho, aplicou-se os conhecimentos obtidos no projeto anterior, melhorando as estratégias e aprendendo como localizar e movimentar um agente sem que a posição real seja conhecida.

O agente desenvolvido consegue executar todas as tarefas e objetivos propostos para este projeto, no entanto, nem sempre completa o desafio, com uma taxa de falha que ronda 1 por cada 15 execuções. Esta falha ocorre quando o robô se aproxima

demasiado das paredes, depois de estar próximo à extremidade do labirinto e virar sem ter uma parede à frente para aplicar a correção da posição estimada.

Como futuro trabalho, seria pertinente melhorar a movimentação do robô fazendo com que este tenha um movimento contínuo invés de parar a cada 2 células. Isto iria permitir com que o robô fosse substancialmente mais rápido e poderia ser um fator crítico num labirinto mais complexo. No entanto, esta estratégia não foi implementada uma vez que se notou que o robô praticamente não colidia com as paredes e terminava o *challenge 4* com alguma margem (800).

Outra possível melhoria seria ajustar a posição do robô não só através do sensor dianteiro mas também com os laterais, de forma semelhante mas apenas quando este estivesse parado (no centro de uma célula múltipla de 2), evitando com que se "confundisse" com os "bicos" das paredes. Esta estratégia foi seriamente ponderada uma vez que já obtendo a correção de localização usando o sensor frontal, não seria nada complicado de a implementar. No entanto, observou-se que o robô tinha um comportamento bastante satisfatório e raramente batia.

Após colidir com uma parede, o robô poderá ficar desorientado. Para evitar com que isto aconteça, poderia ser pertinente a implementação de um mecanismo que o fizesse andar até uma parede próxima conhecida para que houvesse uma espécie de *reset* da sua atual posição.

Referencias

1. LNCS Homepage, http://www.springer.com/lncs, last accessed 2021/11/20.
2. Slides e material disponibilizado na página do elearning da disciplina