Universidade de Aveiro
Dep. de Engenharia Mecânica

Robótica Espacial

Trabalho Prático nº 2 - Maio/Junho 2025

Planeamento de trajetória para um *rover* planetário em ambiente desconhecido

1 Objetivo e descrição geral

O objetivo principal do trabalho é simular em ambiente Matlab o planeamento local de caminhos de um *rover* planetário para exploração num ambiente desconhecido. O robô parte de uma posição inicial que é o referencial de partida correspondente ao ponto onde foi pousado na superfície planetária, ou seja, as coordenadas (0,0,0). O robô não dispõe de um mapa global do terreno mas tem sensores a bordo que lhe permitem medir as distâncias aos obstáculos que o cercam. Não tendo um mapa, o robô deve seguir uma direção dada, por exemplo, pela posição do sol num dado momento e que se pode considerar fixa durante a missão (Figura 1). A tarefa do *rover* é navegar para essa direção até ficar a uma certa distância linear do ponto de partida e que se admite poder ser estimada pela odometria a bordo.

Figura 1: *Ilustração concetual de um* rover *que deve deslocar-se em direção ao sol evitando obstáculos.*

2 Passos principais do procedimento

Os passos principais do procedimento esperado são os seguintes:

1. Pressupõe-se que o robô parte inicialmente da posição (0,0,0) e que não tem qualquer conhecimento do terreno que deve explorar.
2. O programa carrega os objetivos da missão (leitura de um ficheiro com a lista de pontos a visitar).
3. Com base nos limites dos objetivos da missão (pontos de passagem e destino final), o programa deve criar uma grelha de ocupação inicialmente vazia que servirá para ir mapeando o espaço livre e ocupado durante o movimento ao longo de toda a missão.
4. O robô mede as distâncias aos obstáculos à sua volta com um sensor equivalente a um LiDAR. Será fornecida uma função para esse efeito que simula a medição da distância até intersetar um obstáculo

ao longo de uma direção, ou que indica uma distância máxima se não houver obstáculo até esse limite. Há também uma distância mínima que é o limite do sensor, como explicado adiante.

5. Em cada posição que o robô for ocupando, usando as medições que efetuar, preenche as células da grelha que sejam possíveis de determinar com essa informação.
6. Com a grelha atualizada a cada instante, e com a informação disponível, através de um método como os campos de potencial, o robô deve determinar a direção para onde deve ir numa das 8 células que rodeia a sua posição atual.
7. O robô repete este procedimento desde o passo 4 enquanto não atingir o ponto de passagem para onde se dirige. Quando o atingir, o robô deve dirigir-se ao próximo ponto de passagem e repetir os procedimentos desde o passo 4 desta lista.
8. Quando atingir o último destino da missão, o movimento termina e a missão fica concluída. Nessa altura, ou eventualmente também já durante o processo, o robô deve fazer o registo das posições que foi ocupando, ou seja, os pontos do caminho percorrido desde a partida.

3 Elementos disponibilizados

Para o desenvolvimento do programa serão fornecidos os seguinte elementos:

- Um ficheiro com um mapa codificado de todo o ambiente (e.g. ficheiro `scene.mat`)
- Esse ficheiro pode ser carregado para o programa com `load scene.mat` que cria a variável `scene`
- Esse cenário/mapa pode ser acedida (através da variável `scene`) pelas seguintes funções fornecidas:
  - `function visualizeScene3D(scene,key)`
  - `function [distance, hitPoint] = singleBeamLiDAR(scene,pose,theta,key)`
- Ficheiro `tp2_config.txt` com os destinos intermédios a percorrer (sub-objetivos). Os destinos são dados em coordenadas polares: direção e distância linear em relação ao ponto de partida.

O mapa/cenário contido da variável `scene` inclui todos os obstáculos (forma de prismas) mas a informação está codificada e não será acessível sem uma chave (argumento `key`). Se a chave for omitida nos argumentos, as funções indicadas usam uma chave de defeito. Para os ficheiros fornecidos será usada uma chave de defeito e portanto não será preciso usar qualquer chave, exceto se indicado nalgum caso. Assim, as funções `visualizeScene3D()` e `singleBeamLiDAR()` podem ser chamadas sem o argumento `key` e usarão o chave de defeito.

A função `visualizeScene3D(scene)` tem como único propósito mostrar o mapa global para efeitos de *debug* visual dos alunos, mas os dados do mapa não estarão acessíveis. A invocação dessa função criará uma janela com uma vista 3D do mapa, como o ilustrado na Figura 2.

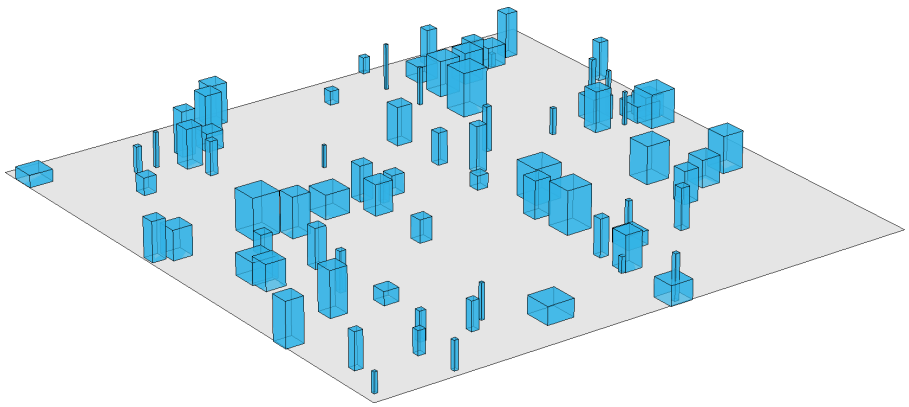

Figura 2: *Ilustração do mapa global com `visualizeScene3D(scene)`. Para visualização apenas.*

4 A função `singleBeamLiDAR()`

A função `singleBeamLiDAR()` fornecida aceita como argumentos o mapa/cenário (`scene`), a pose corrente do robô no formato `pose=[x,y,`ϕ`]`, e a direção do ângulo de medição (θ) vista do referencial do robô. Também aceita um 4º parâmetro que é a chave para descodificar a cena, mas pode ser omitida.

A função devolve no primeiro valor de retorno na variável `distance` a medição (distância) ao obstáculo mais próximo na direção pedida ou um limite de medição do sensor. A função retorna também como segundo valor na variável `hitPoint` as coordenadas (x,y) do ponto de medição no referencial global (seja obstáculo ou seja o limite de medição).

A recomendação é que se a leitura vier igual ao limite máximo de medição ela não se deve considerar porque se trata de espaço vazio pelo menos até àquele ponto e não se sabe se é obstáculo real ou limite de medição do sensor.

Se porventura a medida for pedida a partir de um ponto que está "dentro" de um obstáculo ou a uma distância inferior a um dado limite desse obstáculo, a medida dada será o valor mínimo de medição que o sensor pode dar e, nesse caso, o ponto dado por `hitPoint` pode não ter muito significado.

Esta função pode ser chamada o número de vezes que for necessário e funciona para qualquer direção pedida. Como referido, o ângulo θ é o ângulo medido em relação à frente do *rover*, portanto se o *rover* tiver uma orientação ϕ, quando se pedir uma medida na direção θ, o que a função devolve é efetivamente a medida na direção $(\phi + \theta)$ no referencial inicial do *rover*.

A função pode ser chamada o número de vezes que for útil e nas direções que interessar. É claro que se forem medições muito densas (grande resolução do LiDAR) pode haver implicações computacionais. Por exemplo, na Figura 3 ilustram-se os pontos medidos pelo LiDAR na posição de partida em que se usaram 49 feixes no intervalo de $-120°$ a $+120°$. Algumas medições não tem significado porque são o limite de medição do sensor.

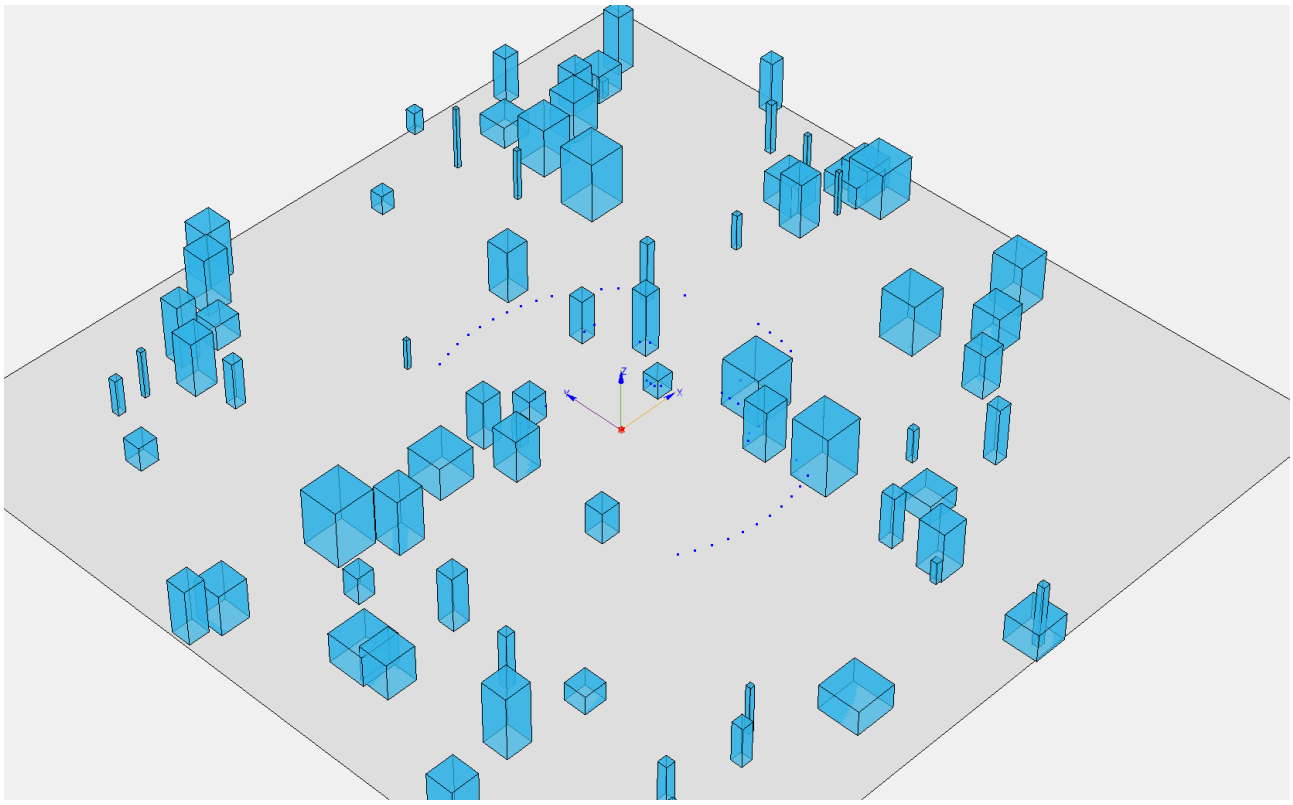

Figura 3: *Pontos detetados pelo LiDAR na posição (0,0,0). Alguns representam os limites do LiDAR e por isso não correspondem a nenhum obstáculo; esses serão descartáveis.*

5 Formato do ficheiro tp2_config.txt

As unidades a considerar no enunciado são metros para comprimentos e radianos para os ângulos, exceto neste ficheiro de configuração onde os ângulos são em graus para mais fácil identificação. Este ficheiro `tp2_config.txt` é um ficheiro ASCII com um formato deste tipo:

```
 40  55
 120 50
-170 45
-50  60
```

Neste exemplo haveria 4 pontos de destino a explorar nas direções 40°, 120°, −170° e −50°, e às respetivas distâncias lineares de 55, 50, 45 e 60 metros, sempre medidas em relação ao ponto de partida do *rover* (0,0,0). Neste caso seriam assim 4 troços de exploração que o *rover* deve percorrer em sequência. Graficamente, estas posições (sub-objetivos) no mapa/cenário seriam os ilustrados na Figura 4.

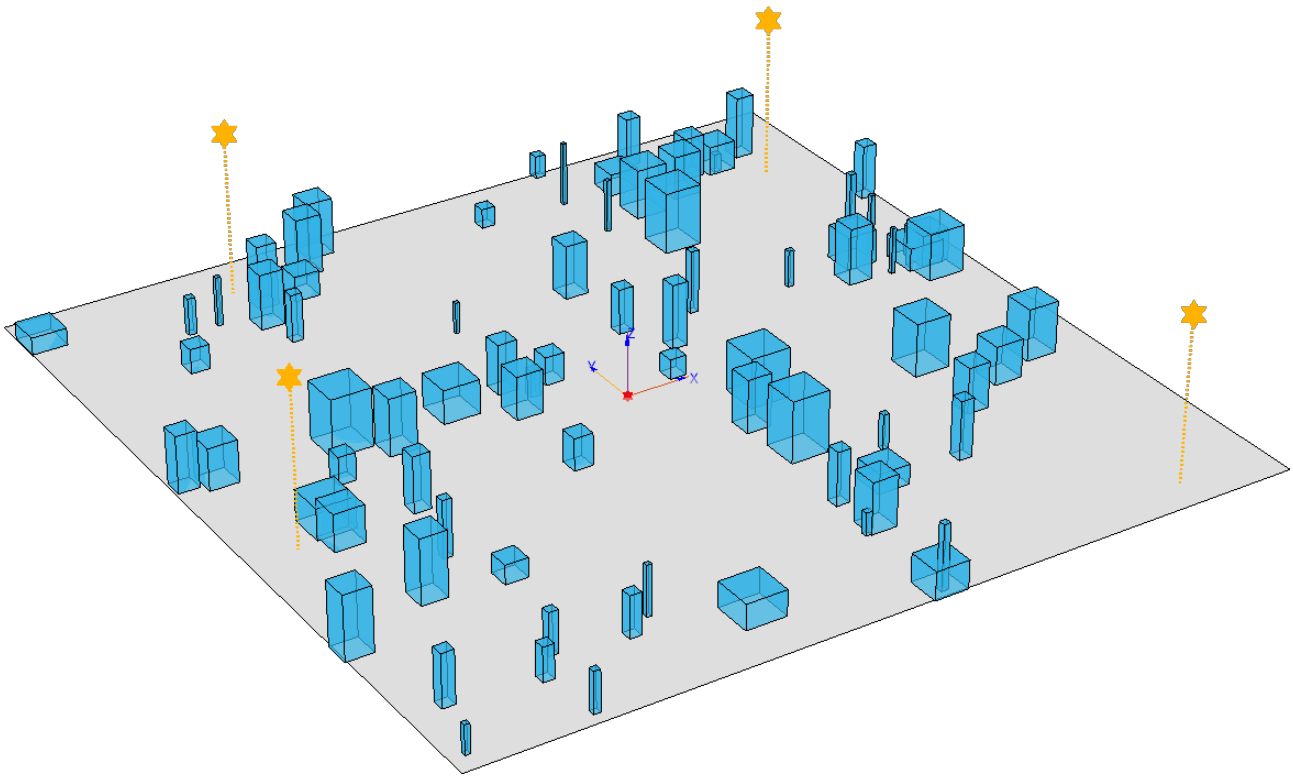

Figura 4: *Ilustração dos sub-objetivos a percorrer pelo* rover *para o ficheiro* `tp2_config.txt` *indicado.*

6 Observações

6.1 Condições da perceção

- A distância máxima que o LiDAR mede é 20 metros.
- A distância mínima que o LiDAR mede é 0.25 metros.
- Considera-se que o *rover* cumpriu um sub-objetivo da missão se o último ponto desse percurso parcial feito pelo robô ficar a menos de 2 metros do ponto real do sub-objetivo.
- Se por ventura houver algum sub-objetivo localizado sobre um obstáculo, pode-se considerar esse sub-objetivo atingido se o *rover* se aproximar dele a uma distância de até 4 metros.

6.2 A grelha de ocupação

- A grelha de ocupação global criada no início do programa começa centrada no robô (0,0,0) e deve ter dimensões suficientes para acomodar a visita de todos os objetivos da missão.
- Essa grelha pode ter a resolução que o aluno desejar e espera-se que seja preenchida à medida que o robô se for movendo e fazendo as medidas com os sensores. Com a grelha corrente, e preenchida com a informação disponível, pode-se fazer a navegação do *rover* com uma ferramenta de navegação local, e aqui recomenda-se usar campos de potencial, embora não seja obrigatório.
- A resolução da grelha pode influenciar o desempenho do programa. Se for muito grosseira pode ser mais difícil encontrar o caminho pelo meio dos obstáculos, mas poderá eventualmente gerar percursos mais curtos quando houver passagem; se for muito fina pode exigir mais recursos computacionais, gerar caminhos ligeiramente mais longos, mas serão certamente caminhos mais seguros (sem colisão com os obstáculos reais).
- Ilustram-se na Figura 5 duas possibilidade de grelhas de ocupação obtidas sobre o mapa real (à esquerda) usando resoluções diferentes. Na primeira, a resolução é mais grosseira limitando as possibilidades de obtenção de caminhos (e até com riscos de caminhos sem saída que podem criar mínimos locais em campos de potencial). Na segunda, a resolução é mais fina e há mais detalhes, mas pode exigir mais recursos computacionais e eventualmente originar caminhos mais sinuosos.

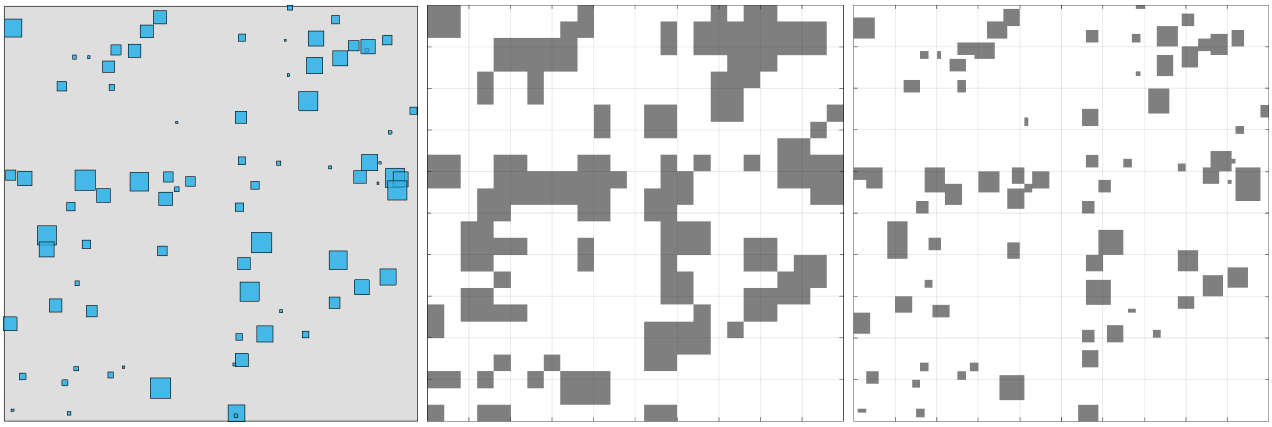

Figura 5: *Cenário real e duas grelhas de ocupação com resoluções diferente.*

- Considera-se que o *rover* consegue passar por passagens que tenham pelo menos 2 metros de largura.

6.3 Risco dos mínimos locais

Se se usarem campos de potencial há o risco de se encontrarem mínimos locais, ou seja, pontos de onde não se consegue sair pela própria metodologia. Nesse caso, será necessário criar uma estratégia para voltar para trás (pelo caminho percorrido em ordem inversa) e de alguma forma precaver que não se volta a percorrer o caminho que leva de novo ao mínimo local.

7 Funcionamento e resultado esperado do programa

Depois das inicializações e preparação inicial da grelha de ocupação, espera-se que o programa vá preenchendo a grelha de acordo com as medidas obtidas a partir da cada célula onde se encontra o *rover* e que ele se desloque para a célula que determinou como a adequada para avançar na sua missão. Portanto, durante a execução deve estar visível a grelha, a sua atualização e a representação do robô a deslocar-se entre células. O trajeto que o robô executa deve ir ficando assinalo atrás dele, unindo os centros das

células percorridas com uma linha. Os resultados do movimento devem ser guardados num ficheiro como descrito de seguida.

7.1 Ficheiro de resultados

- Para guardar os resultados, o programa deve criar um ficheiro de nome `tp2_NNNNNN.txt` onde `NNNNNN` é o número mecanográfico do aluno
- Em cada linha desse ficheiro devem ficar as coordenadas x e y dos pontos percorridos ao longo da missão (incluído eventuais retrocessos no caminho).
- Cada troço (entre sub-objetivos) consistirá no número de linhas necessárias em duas colunas (x y) e será separado do troço seguinte por NaN NaN.
- Considere-se um exemplo reduzido para ilustrar o formato do ficheiro. Se houvesse 3 troços em que o primeiro tem 3 pontos e o 2º e o 3º têm dois pontos cada, e se guardados na variável FF, ter-se-á a seguinte matriz onde os NaN separam os 3 troços entre si:

```
FF =
1     2
3     4
5     6
NaN   NaN
7     8
9    10
NaN   NaN
11    12
13    14
```

- Esta matriz FF pode ser guardada num ficheiro com o comando `writematrix()`.

8 Avaliação

Aos estudantes será dado um mapa para fazer o desenvolvimento do programa, mas para a avaliação será usado um mapa diferente.

A avaliação levará em conta a verificação de resultados independentemente do algoritmo que for usado para simular a missão.

Para cada troço de exploração (entre o ponto de partida/passagem e o próximo ponto de passagem/destino final) são considerados os seguintes elementos quantitativos:

- Comprimento real do caminho realizado face ao caminho mais pequeno possível.
- Cumprimento da proximidade aos diversos pontos de passagem (sub-objetivos).
- Número de pontos da trajetória do robô que colidem com os obstáculos reais.
- Número de situações de bloqueio em mínimo local resolvidas.

Os três primeiros itens correspondem a requisitos obrigatórios; os dois primeiros juntos têm o mesmo peso na avaliação que o 3º. O 4º item será avaliado como bónus se houver evidências que essas situações ocorreram. Esta apreciação será feita para cada troço da missão e a classificação geral será a média dos resultados obtidos em cada troço.

Como referido antes, deve ser feita a representação animada do preenchimento da grelha de ocupação e do movimento do robô ao longo de todo o caminho realizado. Deve ser criado um filme que mostre esse procedimento e movimento e colocado no youtube.

9 Material a entregar

1. Código Matlab. Um *script* com o programa principal que deve incluir no seu fim todas as funções eventualmente desenvolvidas especificamente para o trabalho. Os programas serão executados com as mesmas toolboxes usadas nas aulas.
2. Um ficheiro com o nome `tp2_link_nnnnnn.txt` onde `nnnnnn` é o número mecanográfico do aluno que deve ter o *link* do filme no youtube.
3. Vídeo colocado *on-line* no YouTube com a simulação do processo. Este filme deve identificar pelo menos o autor, a data, a UC e a Universidade. O filme não deve ter uma duração superior a 45 segundos.

Em resumo, devem ser entregues dois ficheiros: um ficheiro com o código matlab (.m) e outro ficheiro com o link do filme carregado no youtube. Não é preciso entregar nenhum relatório escrito.

Nota Final: os trabalhos serão verificados pelo sistema de plágio MOSS da Standford University, e eventuais situações de plágio no código podem levar a anulação de trabalhos. Todos os elementos ou soluções de terceiros (páginas de internet, LLMs, etc.) devem ser creditados.