Robótica Espacial
Aula prática nº 14
Análise e Processamento de LiDAR 2D

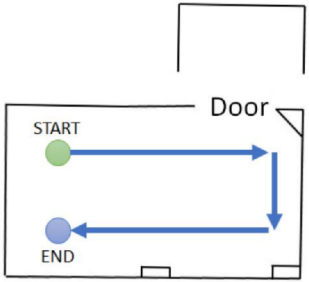
Vitor Santos

Universidade de Aveiro

5 junho 2025

Descrição geral

- Um certo robô navegou num certo espaço com uma geometria aproximada similar à ilustrada e recolheu dados sensoriais com um LiDAR 2D durante esse processo.
- Esses dados estão disponíveis no ficheiro `lidarScans.mat` fornecido.



- O problema a resolver tem várias etapas principais:
  - Representação dos dados
  - Visualização dinâmica dos dados
  - Estudo das diferenças entre scans sucessivos
  - Estimação da trajetória do robô
  - Criação de uma grelha de ocupação do espaço
- Para estes exercícios são necessárias as *toolboxes* **Navigation** e **Lidar** do Matlab

Exercicio 1a - Representação e Visualização de scans LiDAR

- Carregar a sequência de scans 2D

```
load lidarScans.mat
```

  - A variável `lidarScans` que fica disponível pelo carregamento é um array (matriz) de dados do tipo `lidarScan`

- Verificar que a variável `lidarScans` é uma matriz com uma linha e 690 colunas
- Colocar o scan 250 numa variável `s`:

```
s=lidarScans(250);
```

- Verificar que a estrutura de dados na variável `s` tem 4 campos com os seguintes nomes e dimensões:

```
Ranges: [271 x 1 double]
Angles: [271 x 1 double]
Cartesian: [271 x 2 double]
Count: 271
```

Os significados são:

```
% Ranges (range in meters)
% Angles (angle in radians)
% Cartesian (x,y coordinates in meters)
% NB. In the lidar coordinate frame, positive x
%     is forward and positive y is to the left.
% Count (num of measurements in the scan)
```

Exercicio 1b - Representação e Visualização de scans LiDAR II

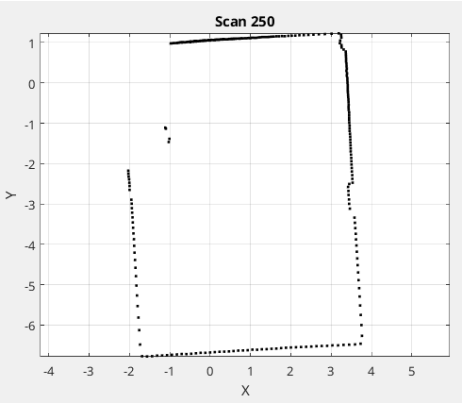
- Representar o scan 250 usando as coordenadas cartesianas com eixos monométricos, uma grelha de fundo e nomes das coordenadas 'X' e 'Y':

```
sCart=s.Cartesian;
plot(sCart(:,1), sCart(:,2),'.k');
axis equal; grid on
xlabel('X');  ylabel('Y')
```

- Verificar, por adição no gráfico anterior, que os dados que estão nas variáveis polares são os mesmos:

```
[x,y]=pol2cart(s.Angles,s.Ranges);
hold on
plot(x, y,'or');
```

- Isto significa que o tipo de dados `lidarScan` tem informação redundante (formato cartesiano e polar).

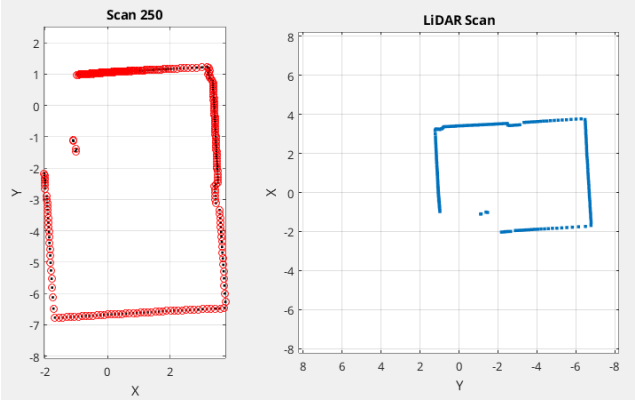


Exercicio 1c - Representação e Visualização de scans LiDAR III

- Uma forma alternativa de representar o gráfico é a de usar o método `plot` de `lidarScan`:

```
s.plot; % or plot(s)
```

- O gráfico tem propriedades visuais diferentes mas os dados são os mesmos. O ponto de vista é diferente, mas pode ser ajustado com `view(0,90)`:



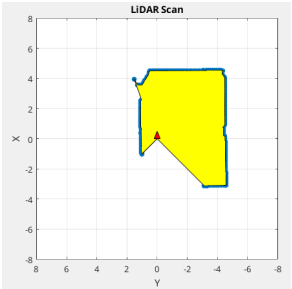
Exercício 2a - Visualização dinâmica e filtragem de scans

- Animar em sequência os scans do lidar do ponto de vista do lidar/robô
- Incluir na animação um objeto fixo triangular que representa o robô, e.g.:

```
robot=[ 0    0.5 0
       -0.2 0    0.2];
```

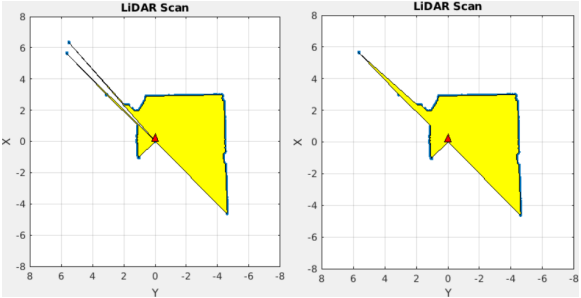
- Para facilitar a visualização do polígono de espaço livre detetado pelo LIDAR, sobrepor nos pontos de scan um polígono fechado cujo primeiro ponto é o centro do robô (0,0).

```
scan = lidarScans(1);
scanCart=scan.Cartesian;
h=scan.plot; h.MarkerSize=20; %LiDAR scan points
hold on; axis equal; grid on; axis([-8 8 -8 8]);
hp=fill([0; scanCart(:,1)], [0; scanCart(:,2)], 'y'); %polygon
fill(robot(1,:), robot(2,:),'r');  %The robot
for n=1:numel(lidarScans)
  scan = lidarScans(n);
  scanCart=scan.Cartesian;
  set(h, 'XData',***,'YData',***) %update LiDAR plot
  set(hp,'XData',***,'YData',***) %update polygon plot
  pause(0.04)
end
```



Exercício 2b - Eliminar medidas erradas

- Algumas medições têm erros e podem ser detetadas por eliminação de medidas demasiado curtas ou demasiado longas. Por exemplo, o scan número 140 apresenta algumas medidas muito longas e outras muito curtas.
- A classe `lidarScan` tem um método designado "removeInvalidData" que se baseia na eliminação de medidas fora de um intervalo admissível.
- Aplicar essa operação com as medidas mínima aceitável de 0.15 m e máxima de 8 m, e o scan original (esquerda) passará a ter o seguinte aspeto (direita).



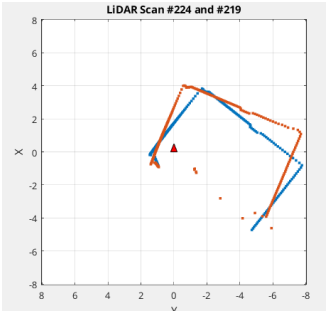
Código a acrescentar dentro do ciclo para filtrar as medidas inválidas:

```
...
minRange = 0.15;
maxRange = 8;
scan = removeInvalidData(scan,...
       'RangeLimits',[minRange maxRange]);
...
```

Exercício 3a - Estudo da diferença entre scans sucessivos

- Adaptar o exercício anterior para representar scans consecutivos (omitindo o polígono)
- scan atual: `currScan = lidarScans(n)`
- scan anterior: `prevScan = lidarScans(n-1)` (ou mais geral: `(n-step)`)

```
...
step=1;
start=1; %first scan to analyse
numScans = numel(lidarScans);
for n=step+start:numScans
  refScan = lidarScans(n-step);
  currScan = lidarScans(n);
  *** % filter and update the 2 scans
  title("LiDAR Scan #"+n+" and #"+(n-step))
  pause(0.04)
end
```



- Ilustração dos scans 224 e 219 (neste caso usou-se step=5). Emula uma situação em que o robô poderia ser muito rápido para o sensor, o que daria medições mais esparsas!

Exercício 3b - Transformação entre scans

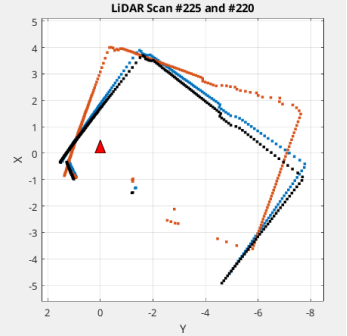
- É possível estimar a transformação geométrica entre scans
- Assumindo o ambiente fixo isso é equivalente a estimar o movimento relativo do robô
- O matlab implementa esse algoritmo com:

```
pose = matchScans(currScan,prevScan)
```

  - onde `pose=[x y` θ`]`
- De forma relacionada, com esta `pose` é possível estimar o scan atual em função do anterior com esta operação:

```
estScan=transformScan(refScan,pose);
```

- se a deteção tiver corrido bem, este processo de `transformScan` deveria recuperar o `currScan` com rigor.
- Em geral isso acontece, mas em alturas de movimento rápido, há alguma diferença, como no exemplo.



Vermelho – scan corrente
Azul – scan anterior
Preto – scan estimado a partir do anterior (deveria ser próximo do corrente – ou seja, houve grande erro)

Exercício 4 - Estimação da trajetória do robô

- Acumular as transformações entre scans sucessivos e estimar o deslocamento do robô entre medições (pose relativa)
- Note-se que as transformações foram todas relativas (incrementais)
- Para poder representar a trajetória real do robô é preciso obter as transformações ou poses absolutas. Isso faz-se de uma forma cumulativa.
- Alterar o código anterior de modo a guardar as poses obtidas em cada iteração.

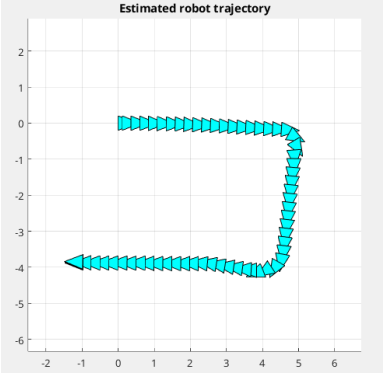
```
initialPose = [0 0 0];          %initial estimate of pose
poseList = zeros(numScans,3); %empty list to calculate poses
poseList(1,:) = initialPose;
```

- Em cada iteração guardar o resultado do scan matching.

```
...
pose = matchScans(currScan,refScan);
poseList(n,:)= pose;
...
```

Exercício 4 (concl) - Representação da trajetória estimada

- Para obter a trajetória do robô, aplicar cumulativamente uma transformação geométrica com base nas poses obtidas.
- Pode-se usar/adaptar a função `accumulatedPoses()` fornecida.
- Desenhar a posição do robô em cada uma destas poses para representar a trajetória estimada.



Exercício 5 - Criação de uma grelha de ocupação

- À medida que o robô navega no espaço é possível observar quais as áreas ocupadas por objetos e quais as áreas navegáveis.
- A criação de um mapa do espaço navegável é útil para planeamento e navegação em geral
- O Matlab permite facilmente criar um mapa usando uma grelha de ocupação.
- Usar a função `occupancyMap()` para criar uma grelha de ocupação.

```
map = occupancyMap(15,15,20);
map.GridLocationInWorld = [-7.5 -7.5]
```

- Usar a função `insertRay` para inserir no mapa as observações feitas pelo sensor LASER.

```
...
currentScan = lidarScans(n);
absolutePose=allPoses(n,:);
insertRay(map, absolutePose, currentScan, 10);
...
```

- Apresentar o mapa resultante e as posições do robô.

```
show(map)
hold on
plot(allPoses(:,1),allPoses(:,2),'bo','DisplayName','Estimatedposition');
```

Exercício 5 - A grelha de ocupação final