Documentação TP1

Data: 11 de Setembro de 2025

Rafael Santos Oliveira

1 Questão 1 e Questão 2

Para a primeira e segunda questão, foi criada uma nova cena no CoppeliaSim, na qual foram adicionados seis objetos: um robô móvel (Pioneer 3DX) e cinco outros elementos distintos, distribuídos pelo ambiente, como móveis e pessoas. A cena gerada pode ser visualizada na Figura 1.

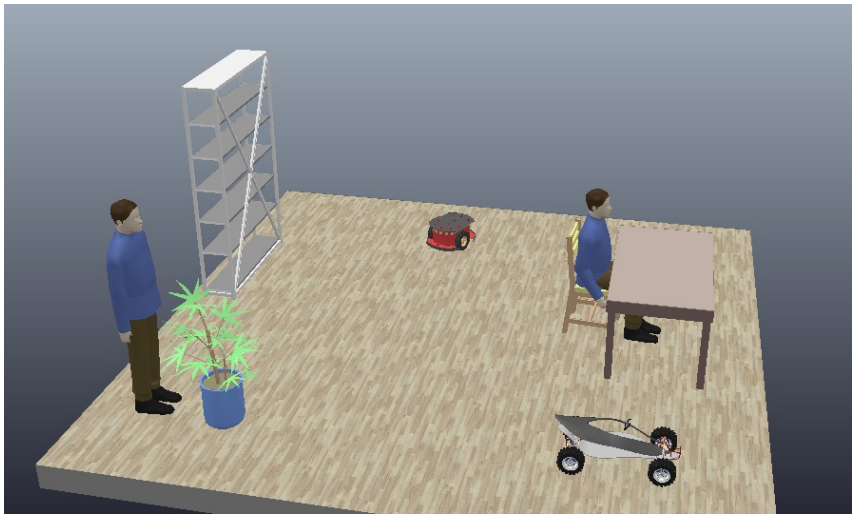

Figura 1: Cena do CoppeliaSim com o robô móvel e os cinco objetos adicionais distribuídos no ambiente.

Em seguida, foi elaborado um diagrama para representar todos frames dos objetos presentes na cena, conforme mostrado na Figura 2.

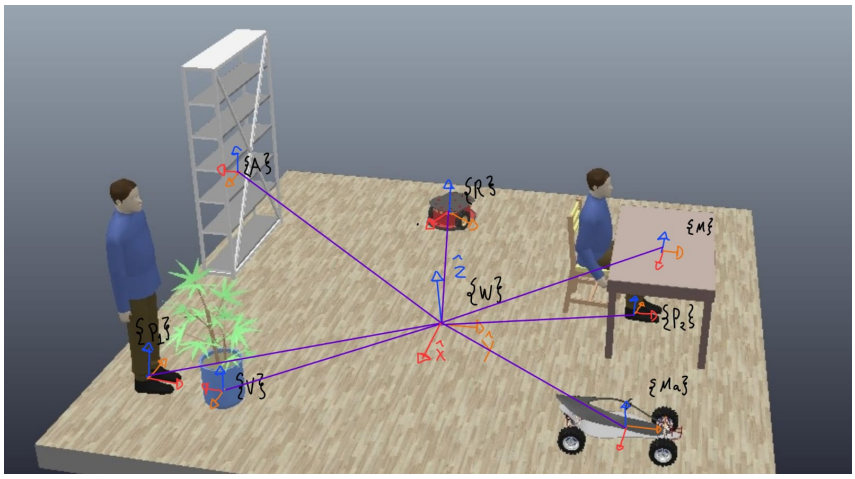

Figura 2: Representação dos sistemas de coordenadas dos objetos na cena.

Tabela 1: Itens adicionados à cena e suas respectivas abreviações.

Abreviação	**Descrição do objeto**
W	Mundo (referencial global)
P_1	Pessoa 1
P_2	Pessoa 2
V	Vaso
Ma	Manta
A	Armário
R	Robô (Pioneer 3DX)
M	Mesa

Essa nomenclatura foi utilizada durante o desenvolvimento do código. A cena utilizada para a questão 2 e questão 3, está em "src/cena1.ttt".

2 Questão 3

Para a terceira questão, foi desenvolvido um código em Python disponível em `src/notebook-tp1.ipynb`. Nesse código foi criada a classe `ObjetoCoppelia`, utilizada para definir de forma simples as coordenadas dos objetos no formato $[x, y, \theta]$. A classe possui os seguintes métodos principais:

- `getPosition` — retorna a posição atual do objeto;
- `getObjetoOriginal` — obtém o identificador original do objeto no CoppeliaSim;
- `setNewPosition` — define uma nova posição para o objeto.

Além disso, outros métodos foram implementados no segund bloco do Jupyter Notebook:

- `plot_intercessao`: plota o vetor que sai de um objeto em direção a outro objeto;
- `frame_global_to_robot`: responsável por realizar a mudança do referencial global para o referencial do robô. O cálculo é feito usando a matriz de transformação homogênea, que considera apenas as coordenadas x e y, mantendo z inalterado. As matrizes utilizadas são:

$$T_{obj} = \begin{bmatrix} \cos\theta_o & -\sin\theta_o & x_o \\ \sin\theta_o & \cos\theta_o & y_o \\ 0 & 0 & 1 \end{bmatrix}$$

$$T_{robot}^{-1} = \begin{bmatrix} \cos\theta_r & \sin\theta_r & -x_r\cos\theta_r - y_r\sin\theta_r \\ -\sin\theta_r & \cos\theta_r & x_r\sin\theta_r - y_r\cos\theta_r \\ 0 & 0 & 1 \end{bmatrix}$$

A transformação é obtida pela multiplicação $T_{robot}^{-1} \cdot T_{obj}$, que resulta na posição do objeto no referencial do robô.

- `plot_frame`: plota os frames de todos os objetos. Caso seja passada a variável `True`, o cálculo é feito no referencial do robô; caso seja `False`, no referencial global.

Com todos os métodos definidos, foram plotados os objetos tanto no referencial global quanto no referencial do robô. Além disso, foram impressos os valores calculados e comparados com aqueles obtidos diretamente pela `RemoteAPI`. Os resultados podem ser observados nas Figuras 3 e 4, bem como nas tabelas seguintes.

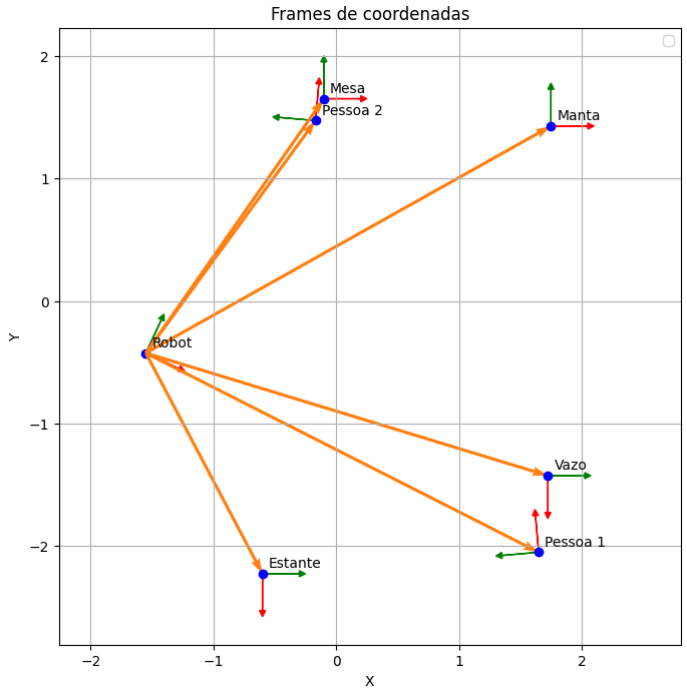

Figura 3: Objetos no referencial global.

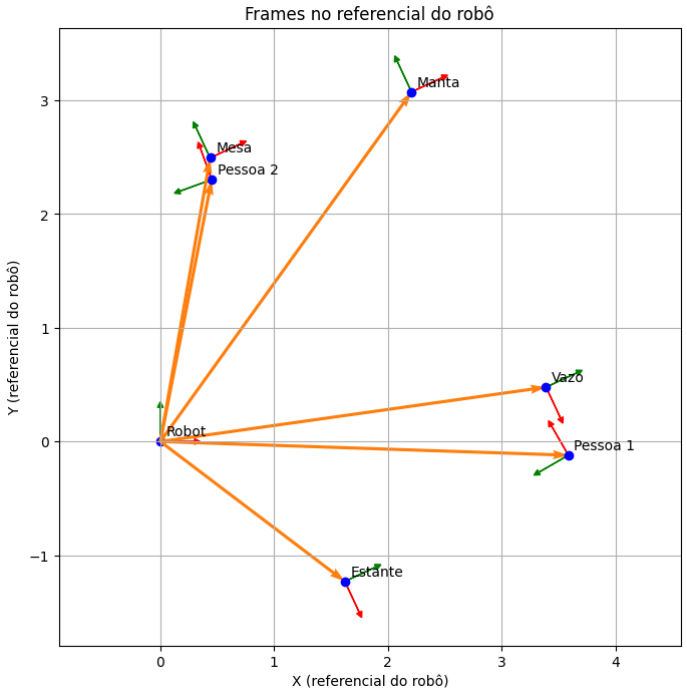

Figura 4: Objetos no referencial do robô.

Tabela 2: Coordenadas dos objetos no referencial global.

Objeto	**Coordenadas** $[x, y, \theta]$
Robô	$[-1.550,\ -0.425,\ -0.436]$
Pessoa 1	$[1.650,\ -2.050,\ 1.658]$
Pessoa 2	$[-0.167,\ 1.471,\ 1.483]$
Vaso	$[1.725,\ -1.425,\ -1.571]$
Estante	$[-0.600,\ -2.225,\ -1.571]$
Mesa	$[-0.100,\ 1.650,\ 0.000]$
Manta	$[1.750,\ 1.427,\ 0.000]$

Tabela 3: Coordenadas dos objetos no referencial do robô, incluindo comparação com transformação da API.

Objeto	**Transformação (API)**	**Transformação (calculada)**
Robô	$[0.000,\ 0.000,\ 0.000]$	$[0.000,\ 0.000,\ 0.000]$
Pessoa 1	$[3.587,\ -0.120,\ 2.094]$	$[3.587,\ -0.120,\ 2.094]$
Pessoa 2	$[0.452,\ 2.303,\ 1.920]$	$[0.452,\ 2.303,\ 1.920]$
Vaso	$[3.391,\ 0.478,\ -1.134]$	$[3.391,\ 0.478,\ -1.134]$
Estante	$[1.622,\ -1.230,\ -1.134]$	$[1.622,\ -1.230,\ -1.134]$
Mesa	$[0.437,\ 2.493,\ 0.436]$	$[0.437,\ 2.493,\ 0.436]$
Manta	$[2.208,\ 3.073,\ 0.436]$	$[2.208,\ 3.073,\ 0.436]$

3 Questão 4

Na quarta questão foi repetido o mesmo procedimento da Questão 3, utilizando o mesmo código, porém alterando a posição do robô na `cena1` em três configurações distintas. Em todos os casos os plots foram gerados no referencial do robô e as coordenadas dos objetos foram comparadas com aquelas fornecidas diretamente pela RemoteAPI, confirmando a consistência dos resultados.

Tabela 4: Coordenadas dos objetos no referencial do robô — Posição 1.

Objeto	**Transformação (API)**	**Transformação (calculada)**
Robô	$[0.000,\ 0.000,\ 0.000]$	$[0.000,\ 0.000,\ 0.000]$
Pessoa 1	$[2.570,\ -0.898,\ 2.094]$	$[2.570,\ -0.898,\ 2.094]$
Pessoa 2	$[-0.565,\ 1.526,\ 1.920]$	$[-0.565,\ 1.526,\ 1.920]$
Vaso	$[2.374,\ -0.300,\ -1.134]$	$[2.374,\ -0.300,\ -1.134]$
Estante	$[0.605,\ -2.007,\ -1.134]$	$[0.605,\ -2.007,\ -1.134]$
Mesa	$[-0.579,\ 1.716,\ 0.436]$	$[-0.579,\ 1.716,\ 0.436]$
Manta	$[1.191,\ 2.296,\ 0.436]$	$[1.191,\ 2.296,\ 0.436]$

Tabela 5: Coordenadas dos objetos no referencial do robô — Posição 2.

Objeto	Transformação (API)	Transformação (calculada)
Robô	[0.000, 0.000, 0.000]	[0.000, 0.000, 0.000]
Pessoa 1	[−3.690, −2.024, 0.524]	[−3.690, −2.024, 0.524]
Pessoa 2	[−1.267, 1.111, 0.349]	[−1.267, 1.111, 0.349]
Vaso	[−3.092, −1.828, −2.705]	[−3.092, −1.828, −2.705]
Estante	[−4.800, −0.059, −2.705]	[−4.800, −0.059, −2.705]
Mesa	[−1.077, 1.125, −1.134]	[−1.077, 1.125, −1.134]
Manta	[−0.497, −0.645, −1.134]	[−0.497, −0.645, −1.134]

Tabela 6: Coordenadas dos objetos no referencial do robô — Posição 3.

Objeto	Transformação (API)	Transformação (calculada)
Robô	[0.000, 0.000, 0.000]	[0.000, 0.000, 0.000]
Pessoa 1	[2.842, −4.182, 1.484]	[2.842, −4.182, 1.484]
Pessoa 2	[1.664, −0.399, 1.309]	[1.664, −0.399, 1.309]
Vaso	[3.024, −3.580, −1.745]	[3.024, −3.580, −1.745]
Estante	[0.596, −3.964, −1.745]	[0.596, −3.964, −1.745]
Mesa	[1.761, −0.234, −0.175]	[1.761, −0.234, −0.175]
Manta	[3.544, −0.775, −0.175]	[3.544, −0.775, −0.175]

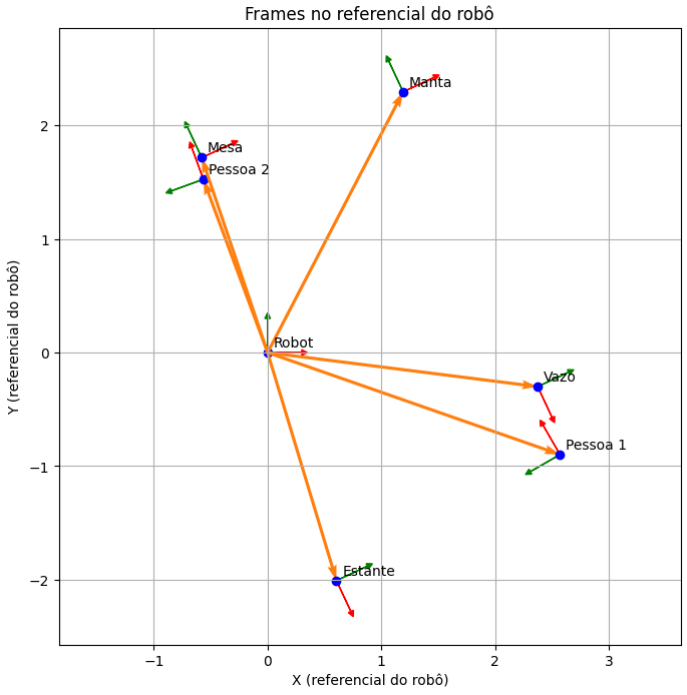

Figura 5: Objetos no referencial do robô na posição 1.

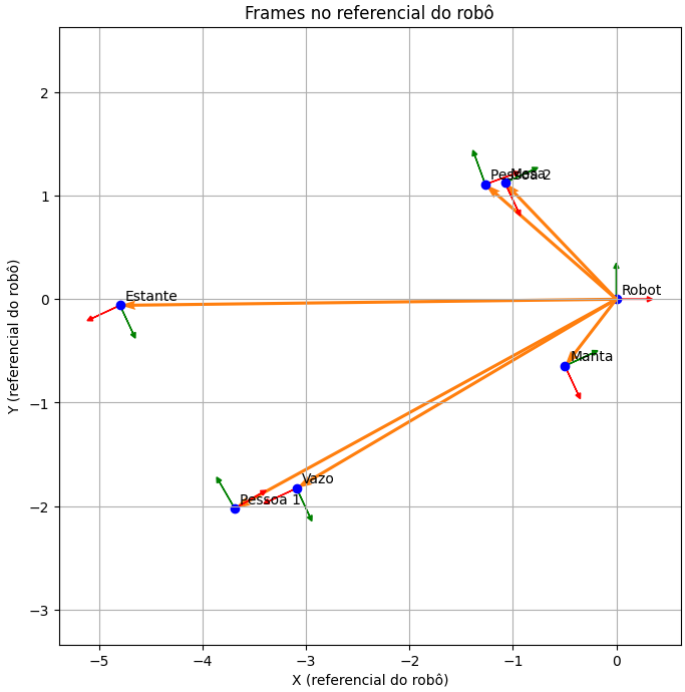

Figura 6: Objetos no referencial do robô na posição 2.

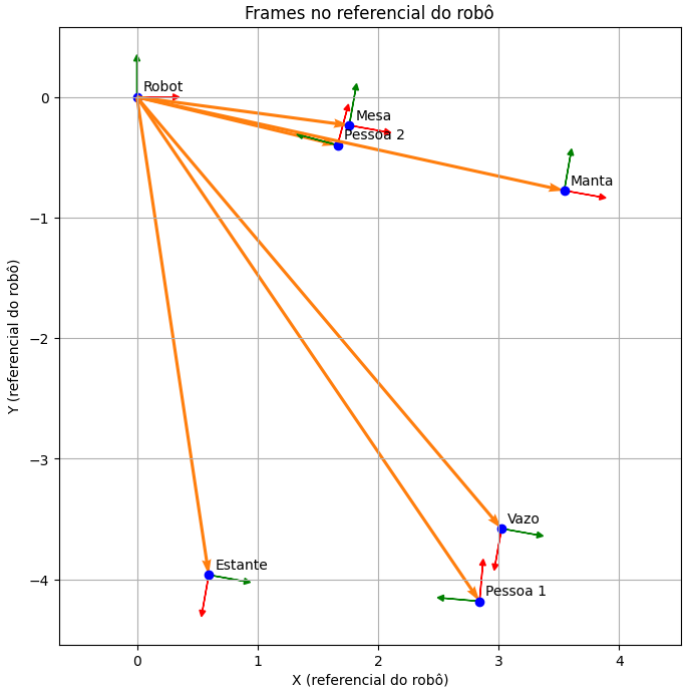

Figura 7: Objetos no referencial do robô na posição 3.

4 Questão 5

Para a quinta questão, foi criada uma nova cena denominada `src/cena2`. Algumas modificações foram realizadas na cena, como a adição de paredes para evitar que o robô

caísse e para melhorar a visualização do plot do sensor laser, como pode ser observado na Figura 8.

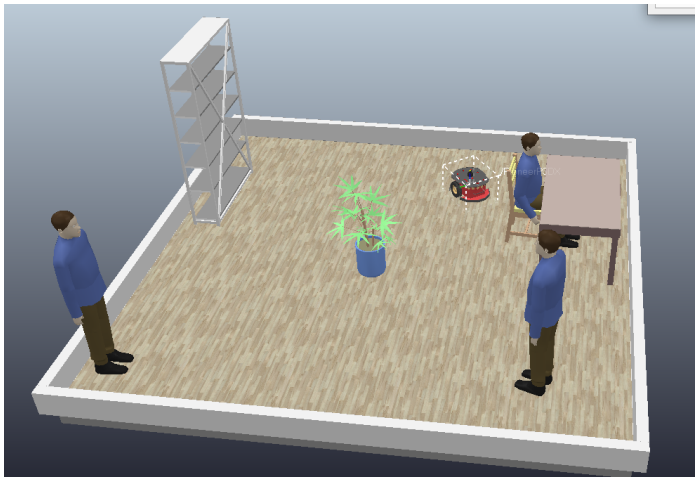

Figura 8: Objetos na cena 2.

Em seguida, o robô foi plotado em três posições diferentes, permitindo observar o comportamento do sensor laser:

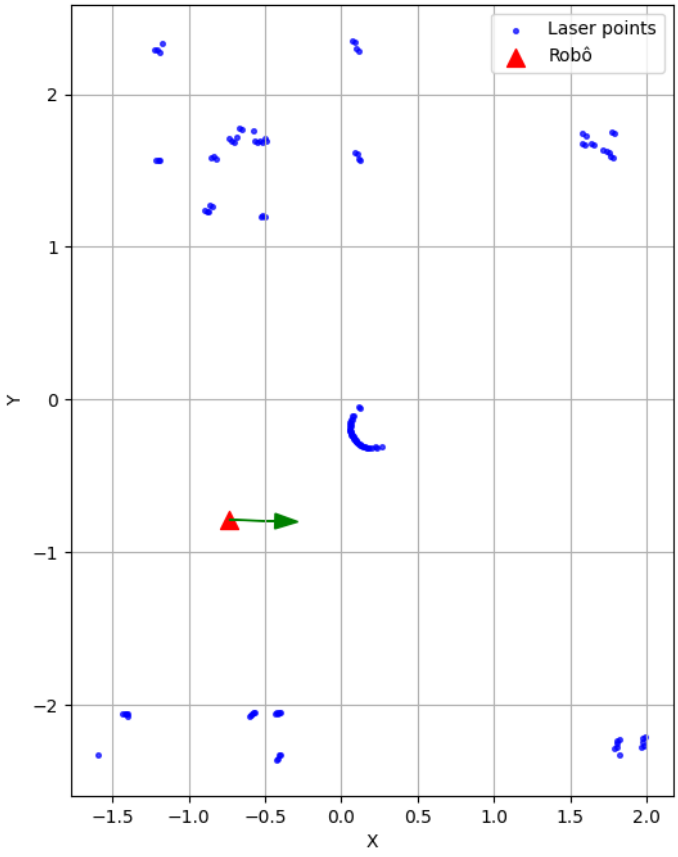

Figura 9: Laser na posição 1 do robô.

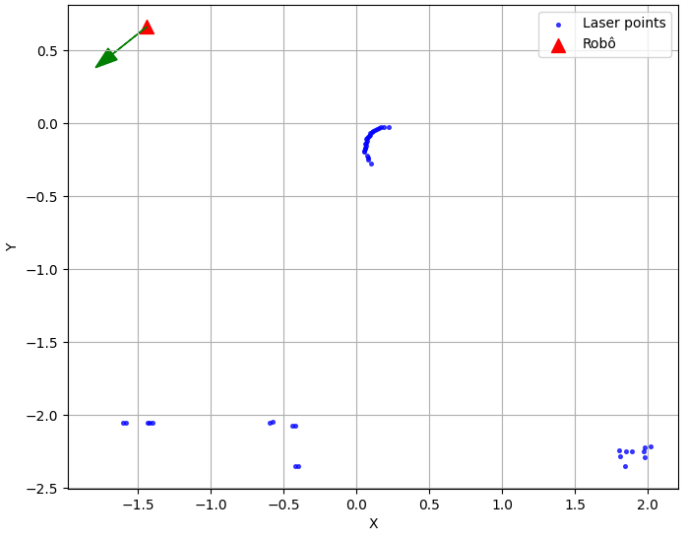

Figura 10: Laser na posição 2 do robô.

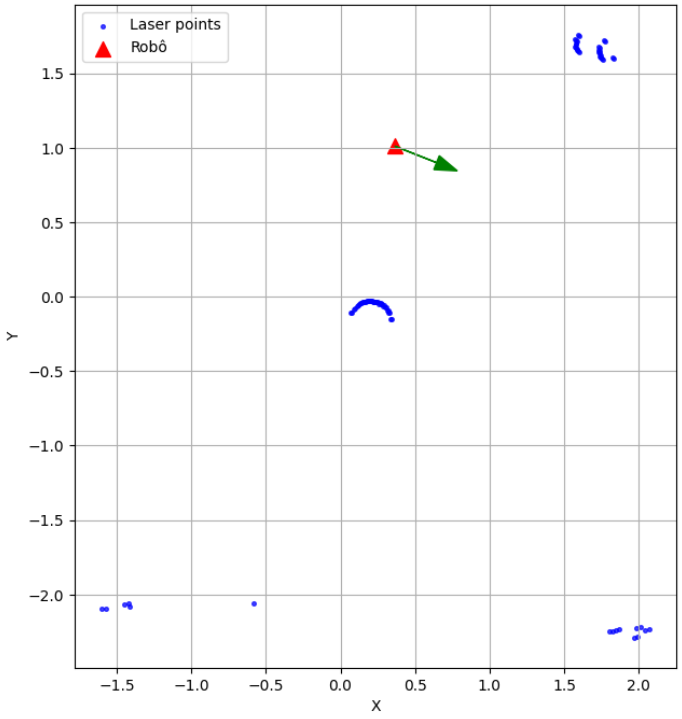

Figura 11: Laser na posição 3 do robô.

A estrutura do código utilizada para esses plots foi baseada no apresentado em aula e inclui:

- `HokuyoSensorSim`: classe responsável pela simulação do sensor laser;
- `transform_laser_to_global`: método responsável por transformar os dados do laser do referencial do robô para o referencial global;
- `plot_laser`: método responsável por gerar os plots das leituras do laser nas diferentes posições do robô.

Dessa forma, foi possível analisar e visualizar o comportamento do sensor laser em diferentes localizações dentro da cena.

5 Questão 6

Para a última questão, os métodos de transformação e a classe responsável pelo controle do sensor laser foram mantidos, conforme implementado na Questão 5. No entanto, nesta etapa, foram armazenadas tanto a trajetória do robô quanto a nuvem de pontos obtida pelo sensor laser.

A Figura 12 ilustra a trajetória do robô e a nuvem de pontos no referencial global, permitindo visualizar a movimentação do robô e as medições do laser ao longo de seu percurso.

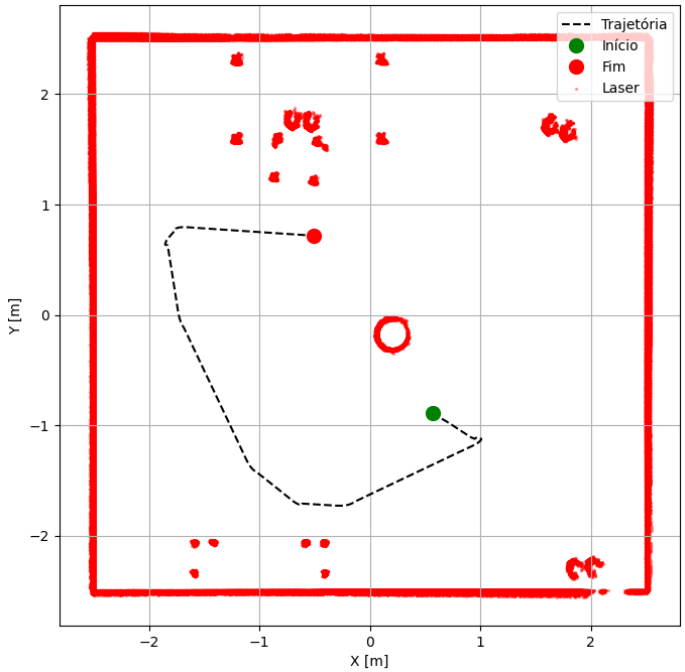

Figura 12: Trajetória do robô e nuvem de pontos obtida pelo sensor laser no referencial global.