Trabalho Prático 1 – Ferramentas e Transformações

Robótica Móvel

08/08/2024

1. Introdução

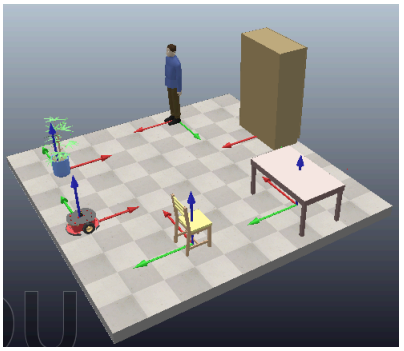
Para aprendermos a desenvolver softwares para robótica móvel é necessário saber alguns conceitos básicos, como transformação de coordenadas e uso de ferramentas de simulação. Este trabalho prático visa levar os alunos a praticar esses fundamentos criando cenas, realizando transformações e praticando o uso do software, CoppeliaSim, e da API, coppeliasim_zmqremoteapi_client em Python, definidas para este trabalho e disciplina.

2. Questões e Implementação

Para fixar esses conhecimentos foi peido a realização de algumas tarefas, as quais discutiremos abaixo.

1. Crie uma nova cena, adicione um robô móvel e outros cinco elementos diferentes.

Essa questão foi simples, uma vez que o CoppeliaSim apresenta uma GUI que torna fácil essa tarefa devido ao seu sistema de segurar e arrastar.



2. Defina os sistemas de coordenadas de todos os itens. Em seguida, você deverá representar os sistemas de coordenadas e as transformações entre eles utilizando um diagrama.

A partir desta necessidade, foi preciso recorrer à programação para responder às questões. Embora a definição dos eixos e o desenho do diagrama tenham sido realizados manualmente, foi indispensável escrever código para efetuar as transformações. Com o objetivo de implementar as transformações de coordenadas discutidas em sala de aula, desenvolvemos três funções distintas, cada uma representando uma matriz de rotação em um eixo específico, além de uma função para gerar a matriz de rotação final, que recebe o ângulo em radianos para cada eixo. Além disso, foi preciso criar uma função para construir a matriz homogênea, combinando a matriz de rotação com a de translação e adicionando a linha de perspectiva e escala.

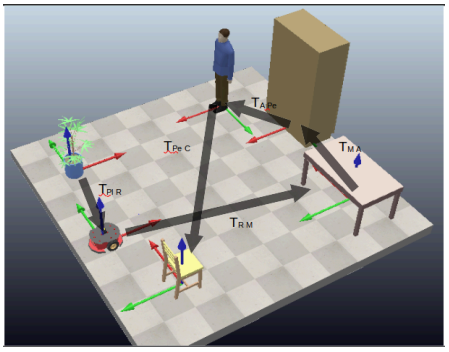
Por fim, criamos uma função para executar as transformações, que recebe como entrada a transformação da origem para o referencial que estamos utilizando, o ponto que se tornará o novo referencial em relação ao antigo, o ponto que será colocado nesta nova perspectiva em relação à origem, e o conjunto de ângulos em graus pelos quais ele se orienta em relação ao novo referencial. Com todas essas funções prontas, basta coletar as posições de cada elemento em relação à origem e, em seguida, calcular as transformações, fornecendo os resultados das transformações anteriores como entrada para as funções subsequentes.

Durante a execução dessa questão um dos grandes desafios foi lembrar de rotacionar o vetor de transposição antes de criar a matriz homogênea. A ausência desse passo afetava a transformação, uma vez que a translação não estaria referente aos eixos corretos.

$$R_Z(\alpha)\ R_Y(\beta)R_X(\gamma)$$

$$\begin{bmatrix} c\alpha & -s\alpha & 0 \\ s\alpha & c\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} c\beta & 0 & s\beta \\ 0 & 1 & 0 \\ -s\beta & 0 & c\beta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & c\gamma & -s\gamma \\ 0 & s\gamma & c\gamma \end{bmatrix}$$

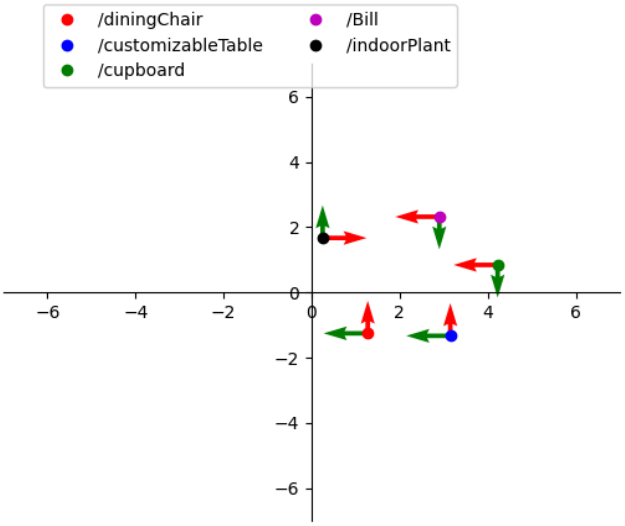
$$\left[\begin{array}{ccc|c} & {}^{A}_{B}R & & {}^{A}P_{BORG} \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$



3. Considerando que a configuração do robô no referencial global é conhecida e dada por q = [x, y, θ], defina as matrizes de transformação homogêneas que representam as posições de todos os outros elementos da cena no referencial local do robô. Escreva um script que plota esses referenciais e os relacionamentos (vetor) entre eles.

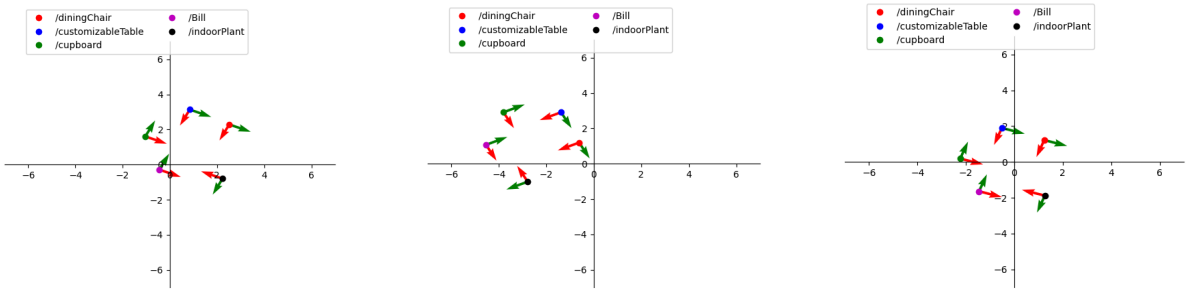
Nessa tarefa foi necessário o uso da API para Python do CoppeliaSim. Unindo as funções fornecidas pela própria biblioteca com algumas funções já feitas anteriormente criamos uma função que calcula a posição de um objeto segundo o referencial de outro, recebendo apenas o caminho para cada objeto. Com essa função completa criamos outra que faria essas transformações de diversos objetos para o mesmo referencial facilitando a execução dessa atividade.

Por fim, com as coordenadas já ajustadas para o referencial do robô, só nos resta implementar uma função para gerar um gráfico dos objetos e seus eixos. Foi uma tarefa simples, uma vez que tínhamos visto exemplos semelhantes na aula.



4. Coloque o robô em outras três posições diferentes da cena e faça os respectivos plots (referenciais e relacionamentos), verificando que a implementação funciona para diferentes casos.

Visando resolver essa questão, criamos uma função destinada a movimentar o robô em X e Y, além de rotacioná-lo sobre o eixo Z. Com essa função fomos capazes de posicionar o robô em três posições diferentes e então gerar os gráficos, comprovando que a implementação anterior estava correta.



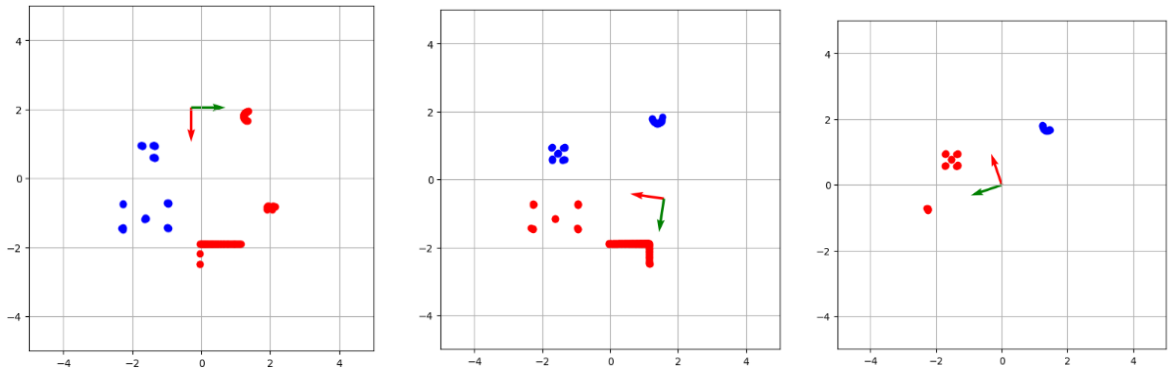
Atenção: Apesar da nova API do CoppeliaSim já iniciar a execução da cena automaticamente ao rodar a célula no notebook, para as questões 5 e 6 é recomendado que as células sejam rodadas com a cena já em execução para evitar possíveis falhas na leitura do laser.

5. Substitua o robô adicionado inicialmente na cena pelo robô com laser mostrado em aula. No exemplo que vimos, o plot da leitura do laser está sendo feito no referencial local do laser. Defina as matrizes de transformações $LT\ R$ (laser → robô) e $RT\ W$ (robô → mundo), e em seguida modifique o script original para que os pontos agora sejam plotados no referencial global, de acordo com a posição atual do robô (recuperada pela RemoteAPI). Coloque o robô em outras diferentes posições da cena (e.g., três) e faça os plots das leituras do laser.

Para responder a questão, foi utilizado como base o script para plotar a visão do robô, visto em aula. Esse código base, calcula a posição dos pontos lidos pelos lasers em relação ao robô, a partir da distância e angulação de cada feixe. Com esse ponto em mão,

agora é necessário calcular a transformação desse ponto que está em relação ao robô, para o referencial global. Foi criada então, a função ***calculateGlobalReferance*** que recebe como parâmetro os pontos x e y gerados pelo código base a partir da leitura do laser, e em seguida, com o uso da API para Python do CoppeliaSim resgata a posição e orientação do robô em relação ao referencial global. Com essas informações, bastou utilizar a função de transformação homogênea, implementada anteriormente, para translocar e rotacionar os pontos em relação ao novo referencial.

Para identificar o posicionamento e direção do robô no momento do plot, foi adicionado um frame onde o eixo x é representado pela seta vermelha e o eixo y representado pela seta verde, mantendo o padrão que é utilizado no CoppeliaSim.



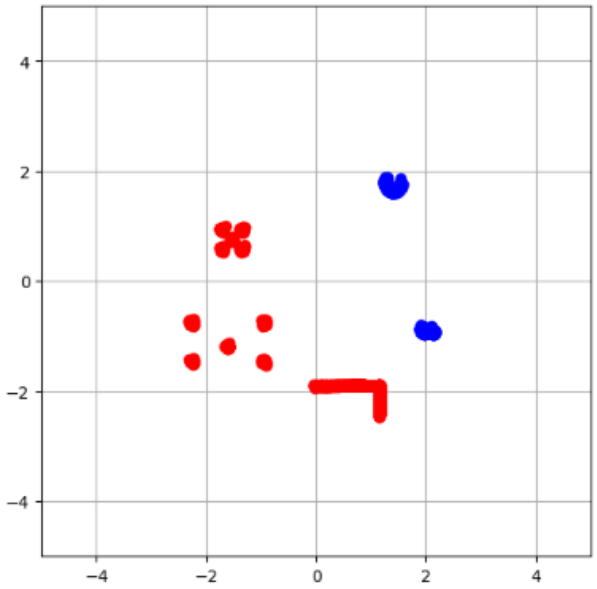
Obs: Assim como no script apresentado em sala, o que é plotado de azul está à direita da visão do robô, e o que está em vermelho está à esquerda.

Inicialmente a maior dificuldade encontrada foi no momento do entendimento da questão, pois ao vê-la pela primeira vez, tivemos um entendimento errado de que o referencial global estaria no canto inferior esquerdo da cena e não no centro como é representado no CoppeliaSim. Após entendermos corretamente como deveria ser implementado o exercício, não houveram grandes dificuldades na implementação, visto que reutilizamos alguns trechos de código feitos nas questões anteriores.

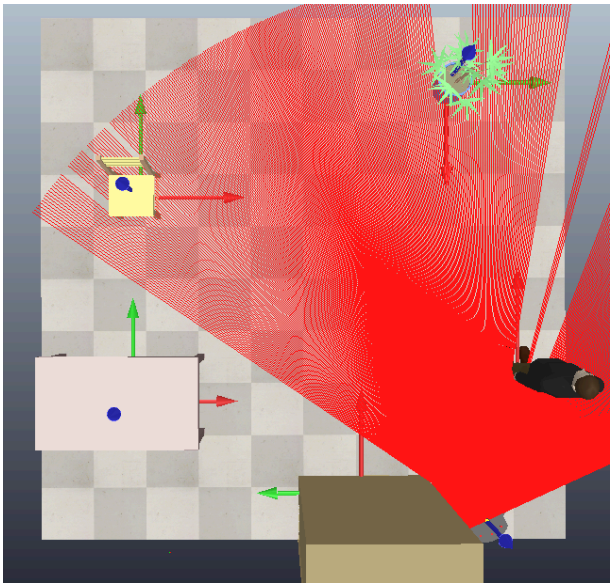
6. Por fim, utilizando o script básico de navegação visto em aula, faça um plot incremental que mostra o caminho executado pelo robô (sequência de posições) ao longo da navegação (por exemplo, usando uma linha tracejada) e combina todas as leituras de laser realizadas ao longo do trajeto. Essas informações devem ser todas representadas em um único gráfico considerando o referencial global.

Para resolver essa tarefa, tomamos como base o script de navegação visto em aula e adicionamos três novas lists, ***laser_data_collection***, ***robot_position_collection*** e ***robot_rotation_collection***, que irão guardar respectivamente, todas as leituras feitas pelo laser durante a execução da cena, todas as posições do robô durante a execução da cena e, por fim, todas as orientações do robô em cada momento da execução. Com isso, a cada execução do while que verifica o tempo percorrido de execução da cena, nós salvamos o que foi registrado no laser, juntamente da posição e orientação do robô naquele momento, para que depois possamos colocar a leitura do laser em relação ao referencial global. Em seguida, na função ***plot_all_robot_visualization***, é feito um for na list que contém as

leituras do laser e para cada leitura, executamos a mesma função de plot utilizada na questão 5, porém, dessa vez, enviando a posição e orientação do robô, correspondente ao momento de cada leitura do laser. O que resultou no seguinte plot incremental das visualizações.

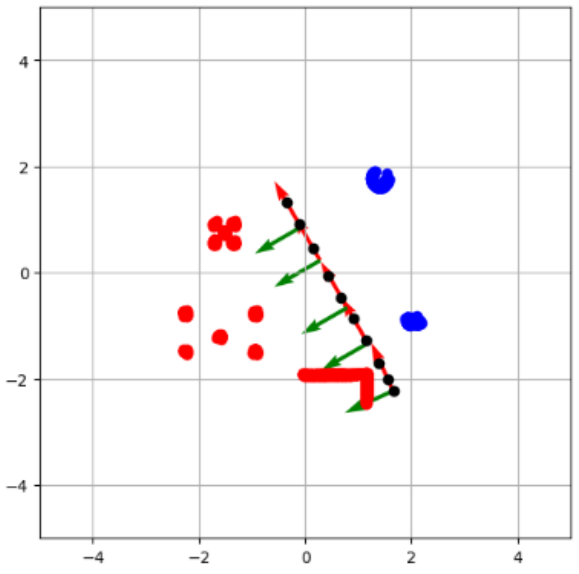


Para indicar a localização do robô, dando um efeito de linha pontilhada, a cada 3 execuções do while que conta o tempo de execução do programa, é plotado um ponto preto indicando a localização exata do robô naquele momento, em relação ao referencial global. Por fim, a cada segundo que passa, é plotado também um frame que indica qual a orientação do robô naquele dado momento.



Posição inicial da cena

3.



Plot final de todas as visualizações do laser

Durante a realização da tarefa uma das principais dificuldades encontradas, foi quanto a unir todas as leituras do laser. Em um primeiro momento, tentamos realizar o plot em tempo de execução, sempre que a leitura era feita na cena, porém o alto número de lasers e posições a serem calculadas para o plot, faziam com que o tempo de cada execução do while ficasse muito alta o que resultava em poucos registros de leitura do laser e consequentemente em uma perda de grande parte das informações. Por isso optamos por adicionar as listas onde são salvos as leituras e posições do robô, e só ao fim da execução da cena, as posições são realmente calculadas no referencial global e os dados plotados no gráfico.

3. Conclusão

Após a conclusão deste trabalho prático podemos afirmar que ele cumpre seu papel, uma vez que este foi muito útil no seu papel de fixar a matéria de transformadas homogêneas, além de nos introduzir ao CoppeliaSim e sua API. Nesse processo encontramos como principais dificuldades a aplicação de maneira correta da transformada e o plot incremental de todas as leituras realizadas pelo laser durante a execução da cena, entretanto esses contratempos acabaram ajudando a fixar melhor os conhecimentos exigidos para sua conclusão. Portanto, esse trabalho foi de grande utilidade na introdução das ferramentas e fixar fundamentos que usaremos ao decorrer do semestre.

4. Bibliografia

- Documentação da API CoppeliaSim, https://manual.coppeliarobotics.com/en/apiFunctions.htm
- GitHub da disciplina, https://github.com/verlab/dcc042-robotica-movel
- Slides de aula, https://homepages.dcc.ufmg.br/~doug/cursos/doku.php?id=cursos:roboticamovel:index