Trabalho prático II - Planejamento e navegação

Filipe Rodrigues Batista de Oliveira[1], **Lucas Roberto Santos Avelar**[2]

[1]Universidade Federal de Minas Gerais

frbo@ufmg.br, lucasrsavelar@ufmg.br

1. Introdução

Este trabalho apresenta os resultados obtidos ao se implementar um algoritmo do paradigma reativo e outro do paradigma deliberativo, ambos vistos em sala de aula: Campos potenciais e Roadmaps, respectivamente; Sendo utilizado um robô diferencial e um holonômico, separadamente em cada técnica; Também foram empregadas técnicas de controle para o efetivo deslocamento dos robôs no ambiente. Para realizar os testes dos algoritmos, usamos duas cenas no Copellia Sim, um software de simulação para robótica.

2. Implementação

A implementação foi feita através do Jupyter Notebook. Utilizou-se a ”Remote API” do CopelliaSim para a interação com o mesmo, além das bibliotecas: Matplotlib para a visualização dos dados, Numpy para operações vetoriais e armazenamento dos dados e a networkX para operações com grafos.

Na primeira parte, algoritmo de Roadmap foi implementado com auxílio de alguns códigos vistos em sala, como a função de cálculo de matriz de rotação, os códigos de plot da imagem do mapa, de divisão da imagem em células, da criação dos pontos em células válidas e da função de cálculo do caminho mais curto entre os pontos inicial e final definidos. Os trechos de código mais relevantes são a função “estaPerto” e o algoritmo que efetivamente faz o robô seguir o caminho definido pelo roadmap.

A função “estaPerto” é uma função auxiliar para determinar se o robô já alcançou uma determinada posição. Para isso, ela calcula a distância euclidiana entre o ponto atual do robô e o ponto alvo e retorna verdadeiro caso essa distância seja menor que uma distância máxima estabelecida (ou falso, caso contrário). Considera-se essa distância máxima como uma margem de erro, pois em muitos casos o robô não para exatamente na posição desejada (principalmente devido a pequenos erros de locomoção), então determina-se uma distância em que essa diferença seja aceitável.

Na última célula, encontra-se o algoritmo que comanda o robô a seguir o caminho traçado entre o início e o fim. Primeiro, apenas recupera-se informações sobre o robô (handles, rodas etc) e define-se a posição inicial dele na cena do Coppelia. Recupera-se também informações sobre um reference frame auxiliar, que será utilizado posteriormente. São definidas algumas variáveis de cinemática, também vistas em sala, e é valido ressaltar que a posição do robô para o script é dada pelo x recuperado a partir do Coppelia e pelo y recuperado negado. Isso é necessário pois como o eixo y da imagem “cresce para baixo”, o modo encontrado de refletir isso na cena do Coppelia foi utilizar a parte negativa do eixo y, que “cresce” na mesma direção. Assim, foi possível combinar os eixos de coordenadas da imagem e do Coppelia.

Depois disso, inicia-se um laço que itera sobre cada ponto presente no caminho determinado. Para cada ponto, calcula-se a posição atual do robô (utilizando o y negado acima explicado), define-se uma variável que representa o ponto a ser alcançado (isso é necessário para inverter os valores do ponto, que inicialmente estão na forma (y, x)) e o reference frame auxiliar é fixado onde está o objetivo.

A funcionalidade do frame é auxiliar o robô a identificar quando ele deve virar ou se deve apenas seguir, em uma tentativa de criar um controlador “Turn and Go”. Se o frame estiver com valor de y negativo no referencial do robô, significa que o objetivo está à direita e, portanto, ele deve virar para a direita. O contrário ocorre quando o valor de y é positivo.

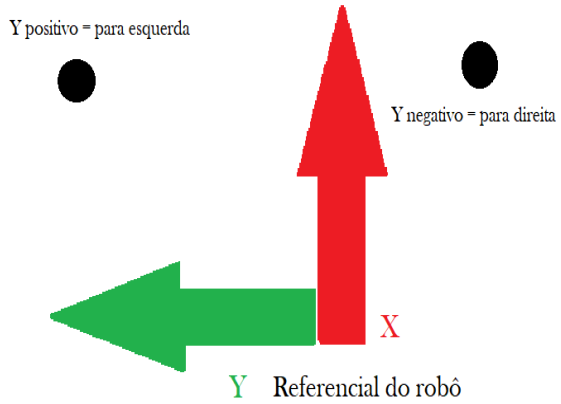

Figure 1. Como funciona o frame auxiliar

Finalmente, quando o robô se encontra na direção certa, basta aplicar comandos de velocidade para que ele se mova em linha reta enquanto o retorno da função ”estaPerto” for falso. Como existem pequenos erros e o robô não se move em uma linha perfeitamente reta, a mesma ideia do frame auxiliar é utilizada para realizar pequenas correções durante o percurso.

Na segunda parte, implementou-se o algoritmo de Campos potenciais, sendo utilizado o controlador de **De Luca e Oriolo [1994]** como modelo para a conversão das velocidades obtidas pelo algoritmo, sendo posteriormente limitado por valor pré definido para aumentar a estabilidade no movimento.

Para a navegação, a ideia é usar os sensores para obter a distância do robô até os obstáculos, cujos valores serão concatenados em uma lista, afim de que se obtenha todos os obstáculos naquele momento, sendo passado posteriormente para função do cálculo da ”força” de repulsão, que internamente, vai verificar se a distância para o robô é menor que um valor definido previamente (raio de influência); Caso o invariante anterior não seja satisfeito ou não haja obstáculo (verificado pela presença de uma lista vazia) é retor-

nado uma ”força” nula, senão, é computado a soma de todas as forças de repulsão, que é posteriormente somada a ”força” de atração obtida pela implementação de sua respectiva função. O potencial foi dado por um modelo parabólico conforme descrito nos slides da disciplina.

3. Testes

Serão realizados dois experimentos para cada algoritmo, variando-se o cenário e posições iniciais e finais do robô.

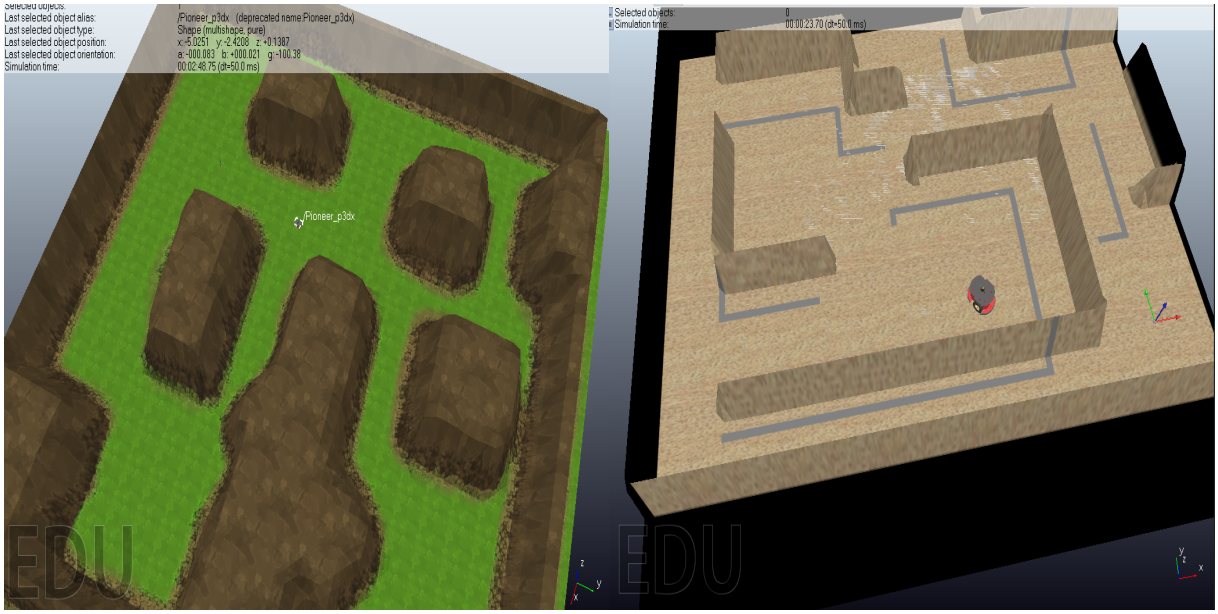

Figure 2. Cenas usadas para teste do algoritmo de campos potenciais, à direita, caso que falha

Para o algoritmo de campos potenciais, escolhemos cenas que apresentassem muitos obstáculos, e os resultados foram bastante insatisfatórios, com o robô demorando muito tempo para realizar o seu percurso (especialmente em regiões estreitas), muitas vezes ficando preso em regiões, os autores creditam este fato a sua natureza ”exploratória” e também a um controle mal feito.
Apesar disso, gostariamos de lembrar que como o mesmo não é completo, nem sempre o robô consegue encontrar um caminho até o objetivo, devido aos mínimos locais que possam ocorrer em determinadas regiões.

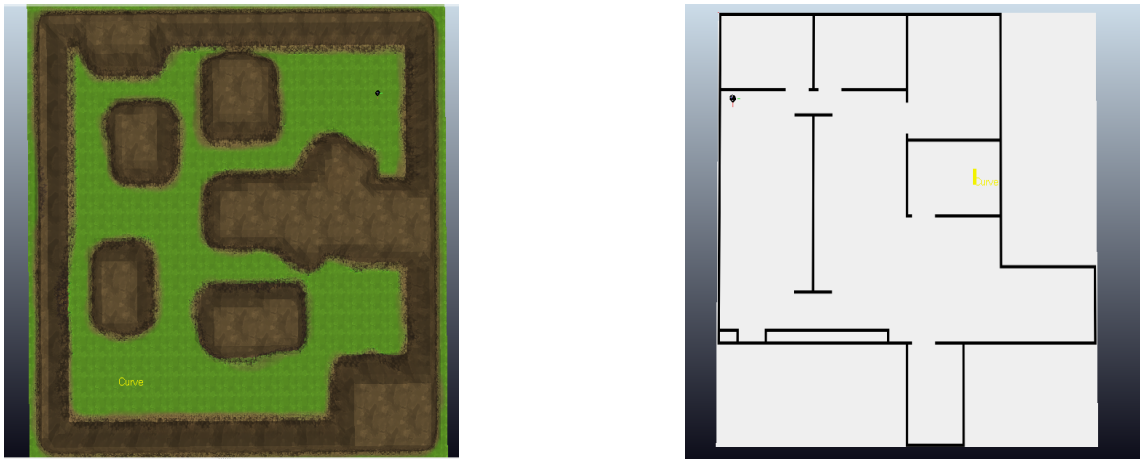

Figure 3. Cenas usadas para teste do algoritmo de roadmaps

Para o algoritmo de roadmaps também foram escolhidas cenas que pudessem forçar o robô a tomar rotas com desvios. A lentidão novamente se mostrou um problema, com o robô levando entre 6 a 10 minutos para completar o percurso. Aplicar uma velocidade menor foi necessário para evitar que o robô saísse muito da rota ao arrancar, provocando desvios com difícil correção. Apesar da demora, o robô obteve êxito em sair do ponto inicial e chegar ao final em ambos os testes, seguindo as rotas predefinidas a partir do grid.

4. Conclusão

Uma das dificuldades encontradas para o algoritmo de campos potenciais foi a de escolher uma boa combinação da parâmetros (raio de influência dos obstáculos, fatores de repulsão e força de atração) de forma a proporcionar a movimentação estável e correta, mas sem dúvida alguma a maior dificuldade foi fazer com que o robô reconhecesse os obstáculos.

Já no algoritmo de roadmaps, a principal dificuldade foi implementar o sistema que efetivamente faria o robô seguir a rota. Comandar o robô a seguir uma linha reta foi relativamente simples, mas houve significativo aumento no grau de dificuldade para compreender como fazer o robô identificar que ele deveria virar, problema que persistiu até surgir a solução de utilizar o frame auxiliar. Outro problema percebido foram os desvios de rota provocados pelos erros e consequentemente pelo fato do robô não realizar uma reta perfeita, que muitas vezes faziam o robô afastar demais do objetivo, a função ”estaPerto” nunca retornar verdadeiro e o programa entrar em loop sem fim. A aplicação dos sistemas de correção de rota foram fundamentais para que esse problema fosse minimizado e para o cumprimento com sucesso do objetivo. Finalmente, outro problema foi a discrepância do tamanho dos obstáculos na cena e no grid, fazendo com que rotas válidas para o roadmap fossem impossíveis na cena, o que foi solucionado com a edição da imagem do mapa para criar uma representação mais realista.

Apesar do grau de complexidade e do esforço para compreender e implementar tudo, pode-se considerar que os avanços realizados foram bastante satisfatórios, visto que é de enorme orgulho ver os robôs capazes de alcançar o objetivo através de uma implementação própria de controladores e paradigmas robóticos.

5. Bibliografia

https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm
D.G Macharet - Slides da disciplina