Trabalho Prático 2 – Planejamento e Navegação

Robótica Móvel

08/08/2024

1. Introdução

As aplicações da robótica móvel estão ficando cada vez mais complexas, que variam de drones que constroem pontes de cordas até carros que se dirigem. Mas por mais impressionantes que sejam os algoritmos e as tarefas executadas por eles, todos esses projetos se apoiam nos mesmos fundamentos, a navegação de um robô sobre um ambiente, conhecido ou desconhecido. Portanto, visando fixar os conceitos de controle, navegação e planejamento de caminhos, implementamos o algoritmo de Roadmap para robôs holonômicos e Campos Potenciais para robôs não holonômicos.

2. Implementação

Os algoritmos foram implementados em Python, com as bibliotecas matplolib, numpy, networkx e a API do Coppelia.

2.1 Roadmap

A técnica de Roadmap (mapa de caminhos) refere-se a um método utilizado para planejar e executar os movimentos de um robô móvel em um ambiente. Essa técnica envolve a criação de um mapa do ambiente em que o robô está e a identificação de rotas seguras e eficientes para o robô navegar.

Existem algumas abordagens para criar roadmaps, e nesse trabalho optamos por implementar a **Graph-based maps**. Nesta abordagem, o ambiente é representado como um grafo, onde os nós representam posições no espaço e as arestas representam conexões entre essas posições. Com base nisso, unimos as informações com o mapa do ambiente e definimos quais são válidos ou não. Em seguida, utilizamos um algoritmo de busca em grafos para planejar o caminho que o robô deverá seguir.

Tomamos como base a implementação fornecida pelo professor, e fizemos algumas modificações para os mapas que foram criados. No primeiro passo então, como visto em aula, a partir do mapa do ambiente previamente conhecido, traçamos as linhas do grid (nos exemplos foram utilizadas células de um metro) e definimos os nós do nosso grafo no centro de cada célula. Em seguida, a partir da análise dos pixels da imagem, identificamos os obstáculos, representados pela cor preta na imagem, presentes no ambiente e definimos quais dos nós são válidos, removendo assim aqueles cuja célula contém alguma parte de um obstáculo. Por fim, agora apenas com os nós válidos do grafo, utilizamos a função *shortest_path* da biblioteca *networkx* que roda um algoritmo e nos retorna o menor caminho entre dois nós no grafo.

Com este caminho em mãos, agora precisamos fazer com que o robô siga cada um dos nós até chegar no nosso goal. Para isso, o primeiro passo foi pegar as posições no mapa referente a cada um dos nós. Porém com as posições em mãos, notamos que elas não estavam em relação ao referencial global da cena, visto que o referencial global do mapa estava localizado no canto inferior esquerdo, enquanto o referencial global da cena

fica localizado no centro. Para colocarmos todos os pontos em relação ao referencial global, fizemos uma transformação de cada coordenada do referencial inicial para o referencial global da cena, localizado no (0, 0).

- Posições consideradas para o referencial global do mapa na cena

  x = - (tamanho total do mapa em x / tamanho da célula) / 2
  y = - (tamanho total do mapa em y / tamanho da célula) / 2)

Por fim, para movimentar o robô, utilizamos como base também, outro trecho de código visto em aula para a movimentação de um robô holonômico. Para que o robô seguisse todo o caminho, foi criado um looping onde cada um dos pontos referentes aos nós do grafo, é tratado como um goal menor. Assim, a cada goal alcançado, o looping é rodado novamente e temos um novo goal, até que o robô chegue ao seu objetivo final.

Como melhoria após a conclusão, optamos por fazer uma função que remove os pontos intermediários de uma linha, ou seja, sempre que o caminho do robo conter uma linha reta, ao invés de termos todos os nós que formam essa linha, temos apenas o primeiro, onde começa a movimentação naquela direção linear específica, e o último. Isso faz com que o robô tenha uma movimentação mais fluida, e não fique parando a cada nó do caminho encontrado, apenas naqueles onde é necessário realizar uma mudança de direção.

2.2 Campos Potenciais

O algoritmo de Campos Potenciais reativa é uma técnica projetada para guiar um robô em direção a um objetivo enquanto evita obstáculos captados pelos sensores do robô. Funciona criando um campo de força ao redor do robô, onde o objetivo exerce uma força atrativa e os obstáculos exercem forças repulsivas. Assim, se em direção ao objetivo e desviando-se dos obstáculos.

Esse algoritmo é especialmente útil para robôs não holonômicos, para os quais a movimentação não é trivial. Para controlar seu movimento, recorremos às equações de De Luca e Oriolo, que nos fornecem a velocidade linear e angular necessárias para guiar o robô de forma eficaz. Essas equações permitem ajustar individualmente as velocidades das rodas do robô, garantindo um movimento suave e preciso em direção ao objetivo enquanto evita obstáculos. Com essas funções prontos criamos outras que fazem transformações de coordenadas, leitura de lasers e a conversão dos dados do laser para coordenadas com o robô de referência.

Tendo o básico já implementado, partimos para o algoritmo em questão. O primeiro passo foi implementar a as forças, implementamos as forças de atração e repulsão de maneira semelhante à vista em sala, utilizando as seguintes equações:

- Atração

$$f_{att} = k_{att} * \left(p_{goal} - p_{atual}\right)$$

- Repulsão

$$d_v = \left(p_{atual} - p_{obs}\right)$$

$$d = ||d_v||$$

$$f_{rep} = \frac{k_{rep}}{d^2} * \left(\frac{1}{d} - \frac{1}{R}\right) * \left(\frac{d_v}{d}\right)$$

Sendo k_{att} a constante de atração e k_{rep} a constante de repulsão, p_{goal} a posição final, p_{atual} a posição atual e p_{obs} a posição dos objetos, e R o raio máximo de atuação dos obstáculos. Vale ressaltar que definimos as constantes k de maneira empírica, após realizarmos algumas simulações observamos que o Pioneer se movimentava melhor nas cenas testadas com $k_{att} = 0.6$ e $k_{rep} = 0.01$.

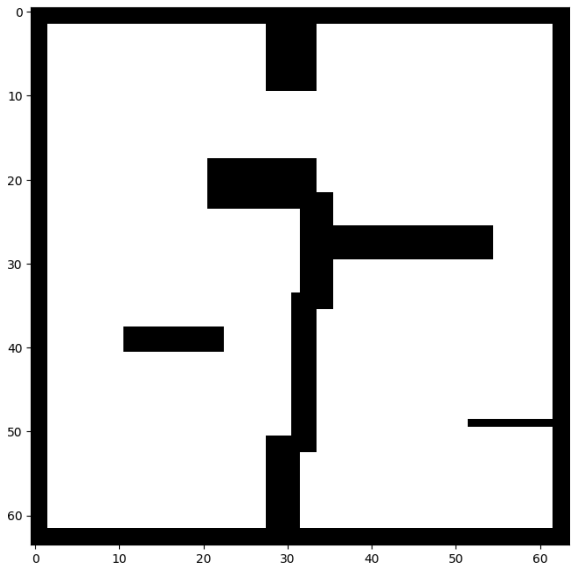
Por fim, um dos grandes problemas desse algoritmo são as zonas de mínimo local, pontos nos quais o vetor de forças resultantes é zero ou muito próximo disso, mesmo não tendo alcançado o destino final. Visando evitar esse problema adicionamos a técnica de forças aleatórias apresentada pelo professor em sala de aula, que consiste em adicionar ao vetor resultante forças aleatórias para evitar as zonas de mínimo local.

3. Testes

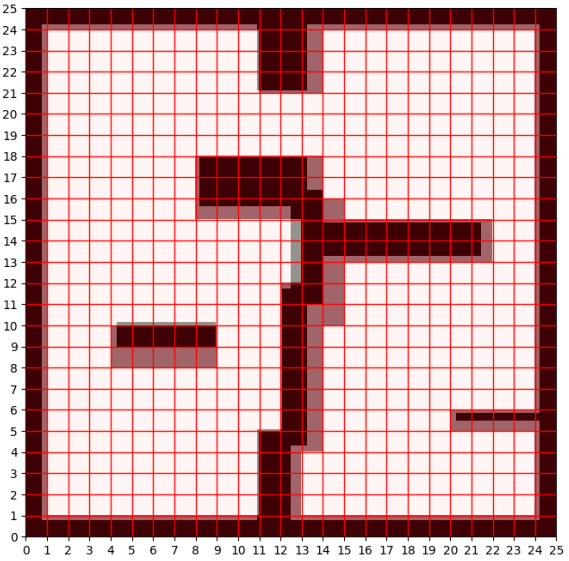
Para fazer testes criamos quatro cenas, duas para cada algoritmo. As cenas destinadas ao algoritmo de Roadmap acabaram sendo mais complexas uma vez que esse é completo, enquanto as de Campos Potenciais foram mais simples uma vez que a chance de cair num mínimo local é grande mesmo com a adição de forças aleatórias.

Na técnica de Roadmap, fizemos inicialmente um teste inicial com um mapa menor de 25x25 metros e em seguida um teste em um mapa mais complexo e com o dobro do tamanho.

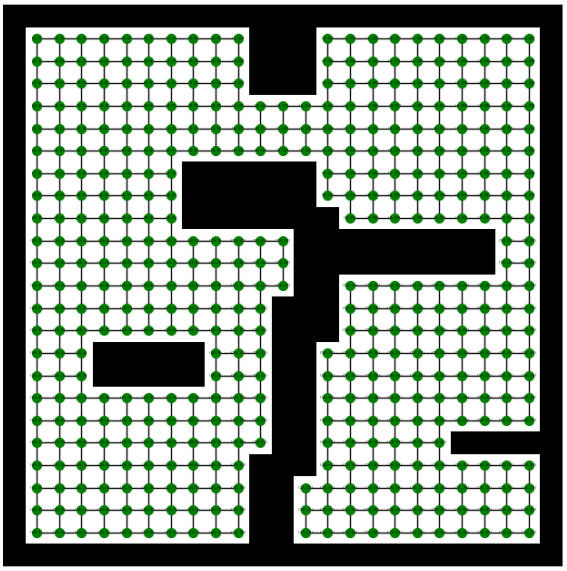
Mapa Simples



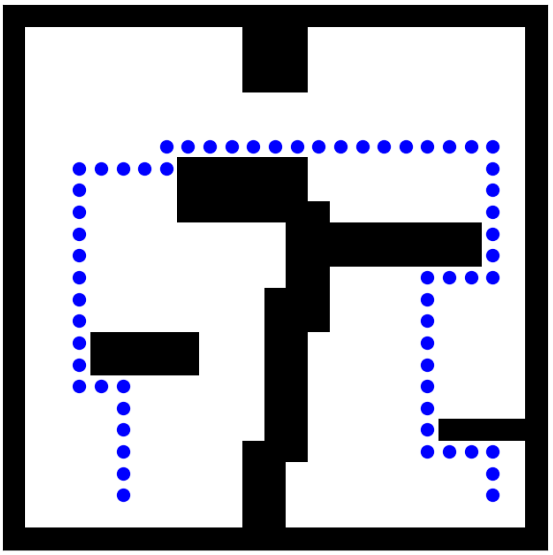
(a) Mapa da cena simples



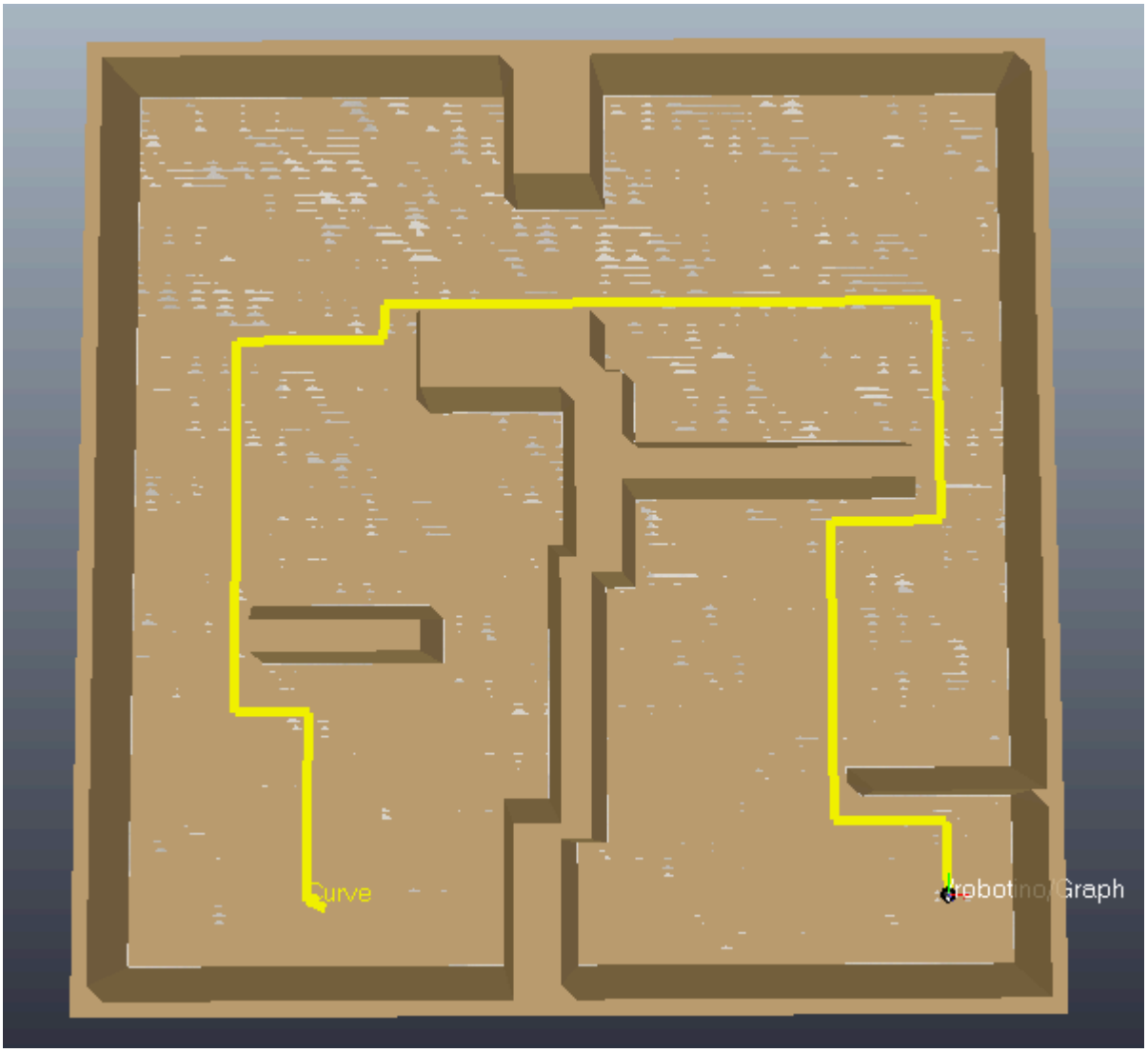
(b) Desenhando grid



(c) Desenhando grafo com nós válidos

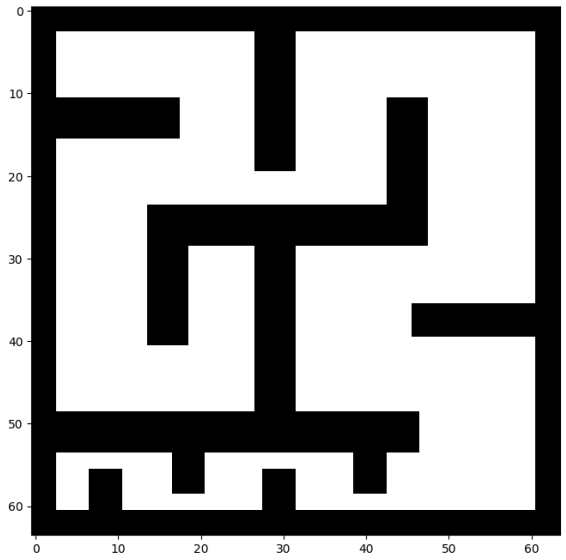


(d) Identificando menor caminho

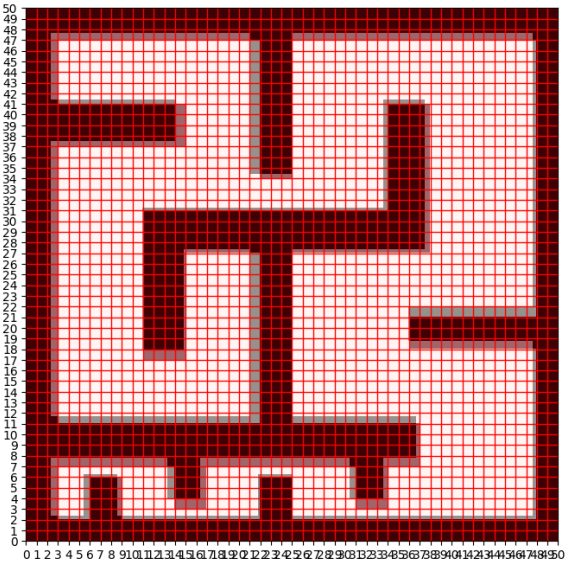


(f) robô percorrendo o caminho proposto no coppelia

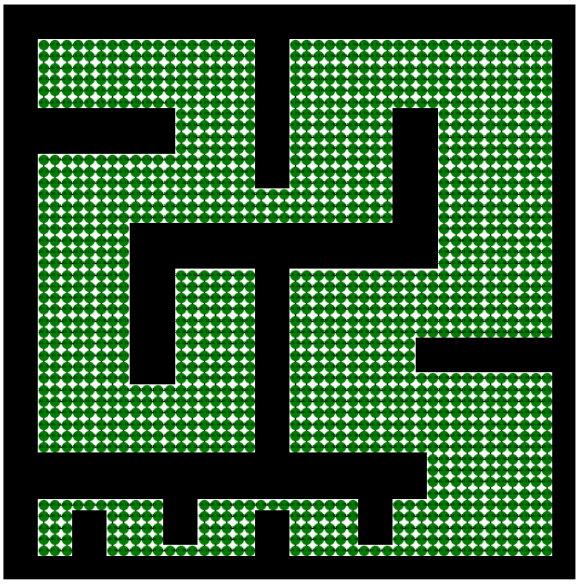
Mapa Hard



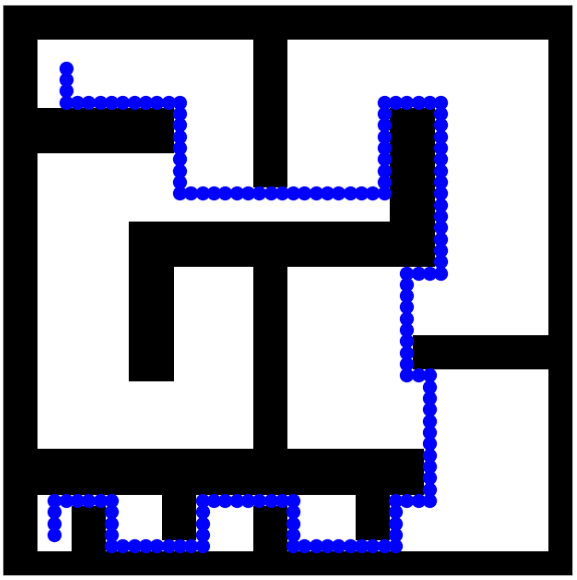
(a) Mapa da cena simples



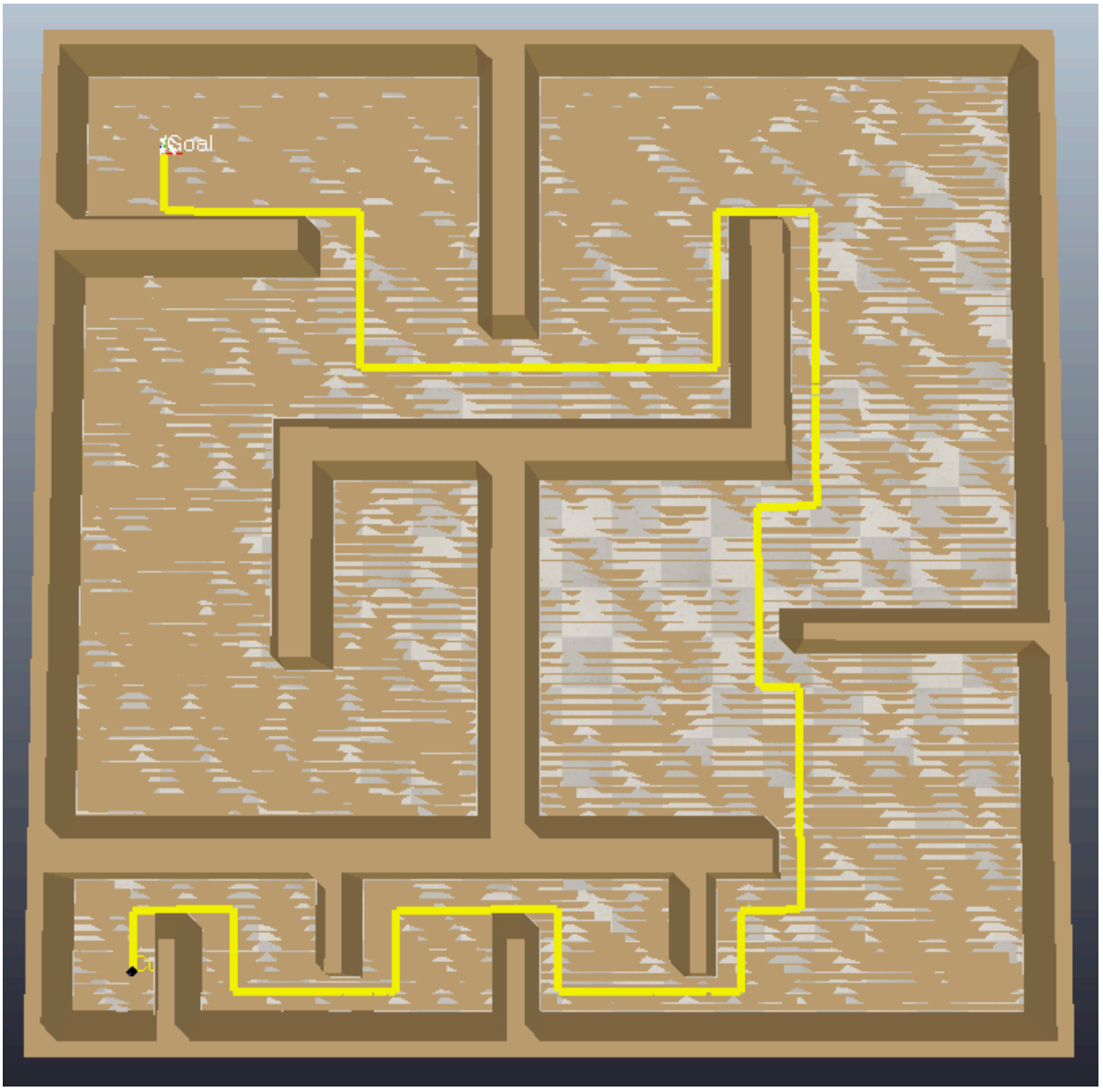
(b) Desenhando grid



(c) Desenhando grafo com nós válidos



(d) Identificando menor caminho

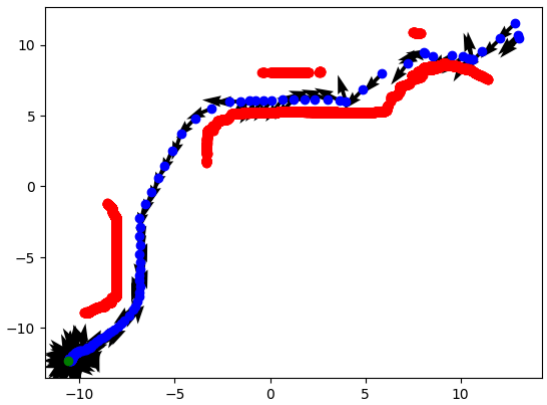


(f) robô percorrendo o caminho proposto no coppelia

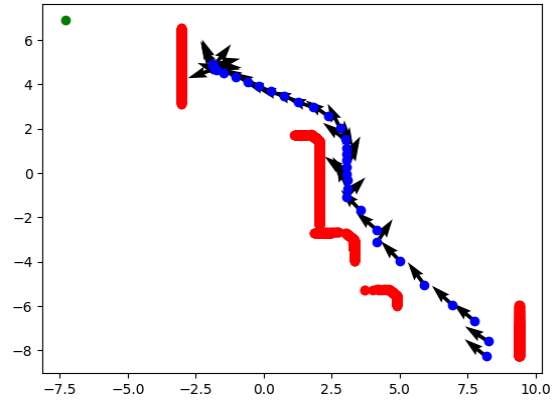
**** Os vídeos com a movimentação do robô na cena estão disponíveis no moodle da disciplina.***

Em ambos os casos obtivemos um bom resultado e o robô conseguiu cumprir o trajeto do início ao fim sem maiores dificuldades.

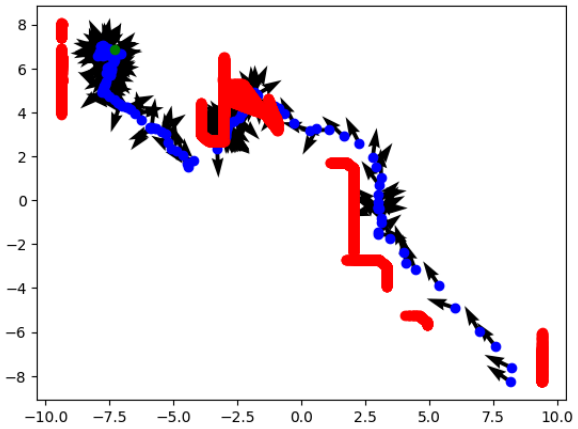
No algoritmo de Campos Potenciais, pudemos ver o algoritmo ter sucesso e também falhar, ficando preso em mínimos locais. Como podemos ver abaixo:



(a) Sucesso, mapa caverna



(b) Fracasso, mapa 2



(c) Sucesso, mapa 2

Como podemos ver no caso de sucesso (a), o robô consegue chegar ao destino final sem problemas, porém com um pouco de dificuldade, devido ao enfraquecimento da força de atração unida às forças aleatórias geradas que atrasam a chegada. Por outro lado, no caso de fracasso (b) vemos que o robô acaba ficando preso em um ponto no qual as forças de atração e repulsão quase se cancelam enquanto as forças aleatórias tentam tirá-lo de lá só que não são suficientes. Tendo isso em mente aumentamos as forças aleatórias sobre o sistema e conseguimos observar elas sendo capazes de retirar o robô do mínimo local, entretanto a movimentação dele ficou bem mais confusa, até se chocando contra a parede no final.

4. 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 controladores e planejamento de caminhos. Nesse processo encontramos como principais dificuldades a transformação do caminho encontrado na imagem do mapa para o caminho real que o robô deveria fazer na cena, além do desafio de evitar ou sair dos mínimos locais e de encontrar as constantes ideias de atração e repulsão para a movimentação do Pioneer ser a melhor possível. Portanto, esse trabalho foi de grande utilidade para fixar fundamentos que usaremos ao decorrer do semestre.

5. 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