Instituto Tecnológico de Aeronáutica

Relatório da disciplina CM-202: Planejamento e Controle para Robótica Móvel

Laboratório 2: Planejamento de Caminho com RRT

Guilherme Müller Bertolino

1. Introdução

A *Rapidly-Exploring Random Tree* (RRT) é uma técnica de planejamento de caminho baseado em amostragem que constrói uma árvore a partir de amostragem aleatória do espaço livre. Esse método garantidamente não encontra o caminho ótimo, mas opera com bom custo computacional, especialmente para espaços de dimensões maiores. O método também é probabilisticamente completo, isto é, a probabilidade de encontrar um caminho factível tende a 1 quando o número de amostras tende ao infinito, mas não é capaz de melhorar a primeira solução encontrada, isso, entretanto, pode ser resolvido usando uma extensão desse método, o RRT*.

2. Metodologia

A implementação da RRT foi feita em `Matlab`, e um pseudocódigo do algoritmo é apresentado na Figura 1.

```
RRT
def rrt(start, goal):
     tree = Tree()
     tree.insert(start, None)
     while not check_stopping_condition():
          if random_uniform() < goal_bias_probability:
               random_node = goal
          else:
               random_node = sample_space()
          nearest = tree.find_nearest(random_node)
          new_node = nearest.extend(random_node, delta)
          if collision_free(nearest, new_node):
               tree.insert(new_node, nearest)
```

Figura 1: Pseudocódigo do algoritmo da RRT.

A operação `extend` consiste em adicionar o nó a árvore caso ele esteja próximo suficiente e, caso contrário, adiciona-se um nó a uma distância Δ do nó mais próximo e contido na reta ligando o nó mais próximo ao nó amostrado:

$$q_{new} = \begin{cases} q_{rand}, ||q_{rand} - q_{neares}|| \leq \Delta \\ q_{nearest} + \Delta(q_{rand} - q_{nearest})/||q_{rand} - q_{nearest}||, ||q_{rand} - q_{nearest}|| > \Delta \end{cases}$$

Na implementação feita, os obstáculos são todos circulares, de modo que para verificar se houve colisão basta checar se o ponto está a uma distância menor do que o raio do obstáculo. A amostragem do espaço livre é feita por *reject sampling*, isto é, amostra-se um ponto aleatório e descarta-se ele caso esteja dentro de um obstáculo, o processo é repetido até que se obtenha um ponto do espaço livre.

3. Resultados

Para teste da implementação feita, foram utilizados quatro cenários:

$$a: \quad q_0 = [1\ 1]^T, q_{goal} = [9\ 9]^T\ \chi_{obs} = \{([5\ 5]^T, 1)\}$$

$$b: \quad q_0 = [1\ 9]^T, q_{goal} = [9\ 1]^T\ \chi_{obs} = \{([5\ 5]^T, 1), ([3\ 3]^T, 1), ([7\ 7]^T, 1)\}$$

$$c: \quad q_0 = [1\ 9]^T, q_{goal} = [9\ 1]^T\ \chi_{obs} = \{([4\ 4]^T, 1), ([4\ 6]^T, 1), ([6\ 4]^T, 1), ([6\ 6]^T, 1)\}$$

$$d: \quad q_0 = [9\ 9]^T, q_{goal} = [9\ 1]^T\ \chi_{obs} = \{([1\ 5]^T, 0.8), ([3\ 5]^T, 1.05), ([5\ 5]^T, 1.05), ([7\ 5], 1.05), ([9\ 5]^T, 1.05)\}$$

3.1. planPathRRT

O caso a foi executado utilizando diferentes probabilidades para o viés em direção ao objetivo, para cada valor.

Para as probabilidades de $p = \{0.01; 0.1; 0.9\}$, uma iteração qualquer dos métodos obtiveram os caminhos apresentados nas Figuras 2, 3 e 4.

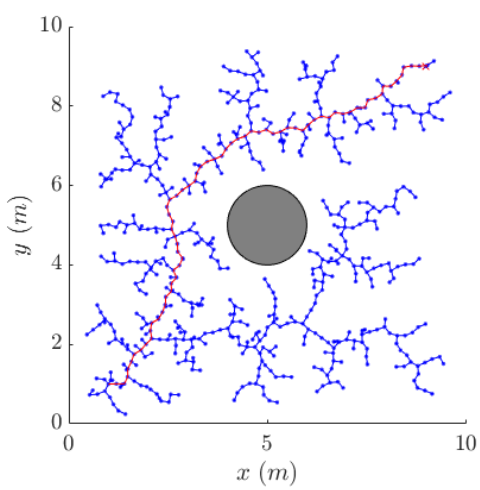

Figura 2: Caminho encontrado no caso 'a' com viés para o objetivo de 0.01.

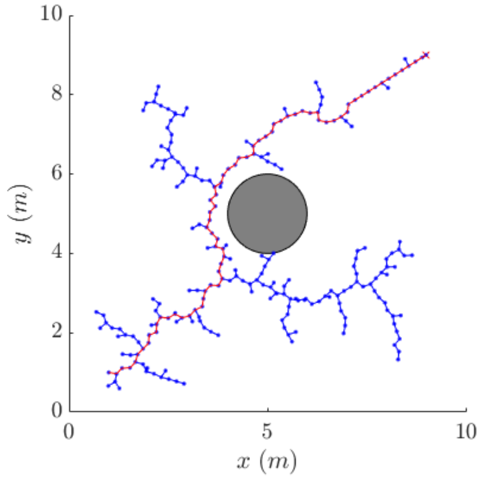

Figura 3: Caminho encontrado no caso 'a' com viés para o objetivo de 0.1.

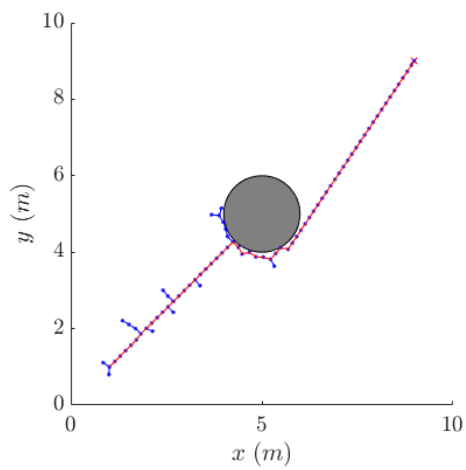

Figura 4: Caminho encontrado no caso 'a' com viés para o objetivo de 0.9.

A partir das Figuras, observa-se que quanto menor o viés para o objetivo, mais explorado é o espaço, conforme o esperado. Como, neste caso, o espaço só possui um obstáculo central, também temos que quanto maior o viés para o objetivo, mais próximo do ótimo será o caminho encontrado, conforme observado nas Figuras.

3.2. Análise 2

Utilizando um viés para o objetivo de 0.1, foram executados os casos b, c e d, de forma a obter as Figuras 5, 6 e 7.

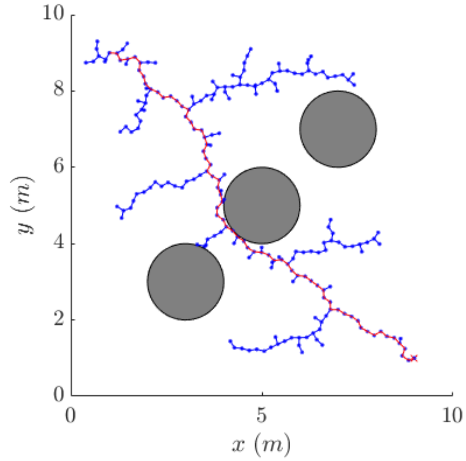

Figura 5: Caminho encontrado no caso 'b'.

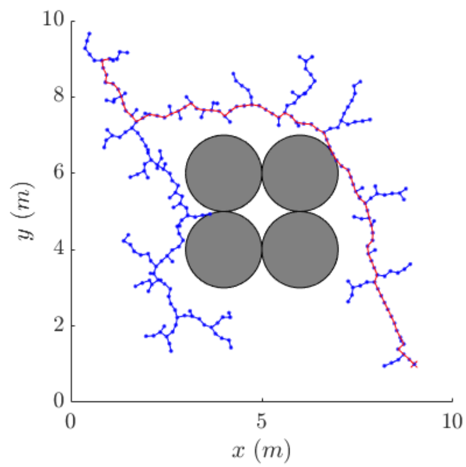

Figura 6: Caminho encontrado no caso 'c'.

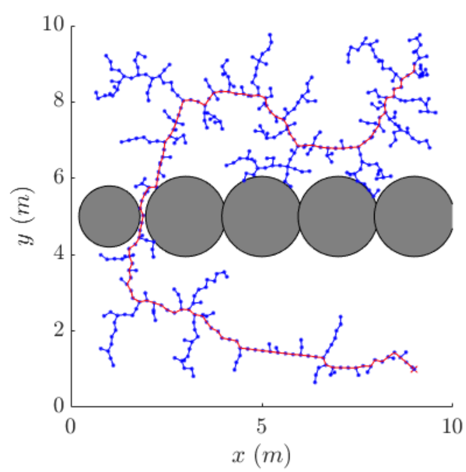

Figura 7: Caminho encontrado no caso 'd'.

Para todos os casos apresentados, observa-se que o RRT foi capaz de encontrar com sucesso um caminho factível através do espaço, mesmo com muitos obstáculos no caminho, apesar deles estarem bem longe de serem ótimos e serem bastante diferentes para cada vez que se roda o método.

3.3. MonteCarloRRT

Os casos a, b, c e d foram rodados 200 vezes em um método de Monte Carlo para obter valores como caminho médio e número médio de iterações, de forma a obter os resultados apresentados na Tabela 1.

Caso	% de sucesso	Caminho médio	N° médio de iterações
a	100%	13.877	238.635
b	100%	13.853	258.040
c	100%	14.911	296.835
d	72.5%	21.412	763.170

Tabela 1: Parâmetros obtidos fazendo um Monte Carlo nos quatro casos.

Na Tabela apresentada, não observa-se nada muito interessante nos casos a, b e c, dado que a RRT conseguiu encontrar um caminho em 100% dos casos, e os três obtiveram um número semelhante para caminho médio e número de iterações. Para o caso d, entretanto, o método não foi capaz de encontrar um caminho com menos de 1000 iterações para quase 30% das tentativas, o que é esperado, já que esse cenário é de fato o mais difícil.