UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Robótica Móvel

Planejamento de caminhos – PRM/RRT

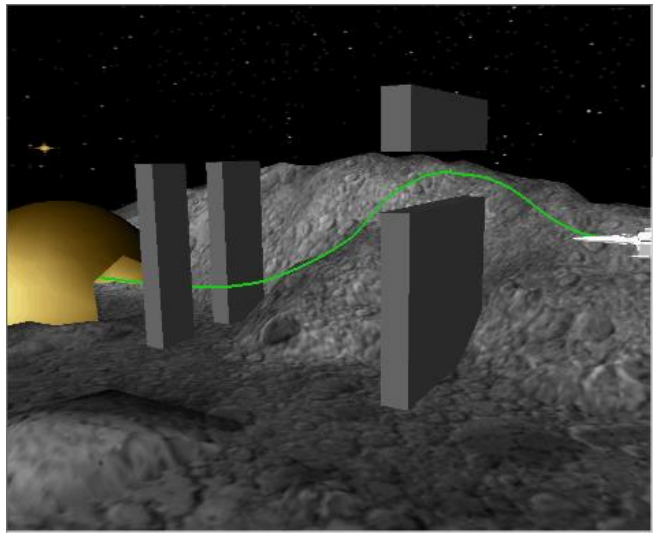
Prof. Douglas G. Macharet
douglas.macharet@dcc.ufmg.br

DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

VERLab
Visão e Robótica

Introdução

- Técnicas determinísticas apresentam dificuldades em ambientes complexos e robôs com vários DoF
  - Discretição do espaço/estados
  - Determinação do $\mathcal{C}$-space
  - Tarefas complexas e caras



http://lavalle.pl/rrt/gallery_xwing.html

- É fácil testar se uma configuração é válida!

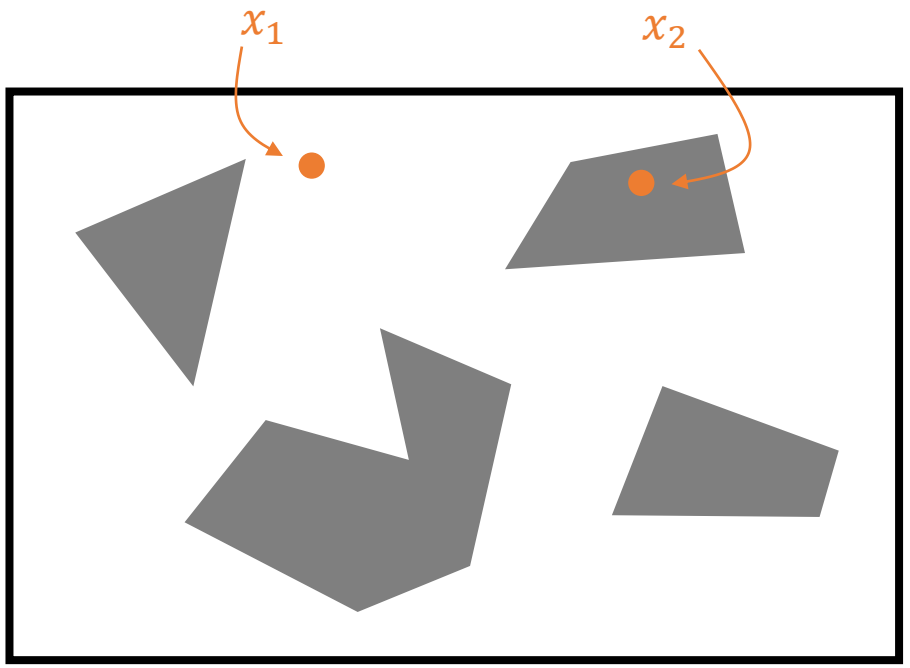
Introdução
Abordagens baseadas em amostragem

- Ideia geral
  - Gerar e distribuir um conjunto aleatório de configurações válidas no ambiente e tentar conectá-las em um grafo/árvore
- Completude e eficácia vs. Simplicidade e eficiência
- Métodos
  - Probabilistic Roadmaps (PRM)
  - Rapidly-Exploring Random Tree (RRT)

Probabilistic Roadmaps

Construção

- Usando uma distribuição qualquer de probabilidades, gerar uma amostra x no domínio do $\mathcal{C}$-space $(\mathbb{R}^2, SE(2), SE(3))$
  - Distribuição uniforme é a mais simples
- Verificar que a amostra gerada é válida → $q(x) \in \mathcal{C}_{free}$
- Conectar a amostra/vértice aos k vizinhos mais próximos



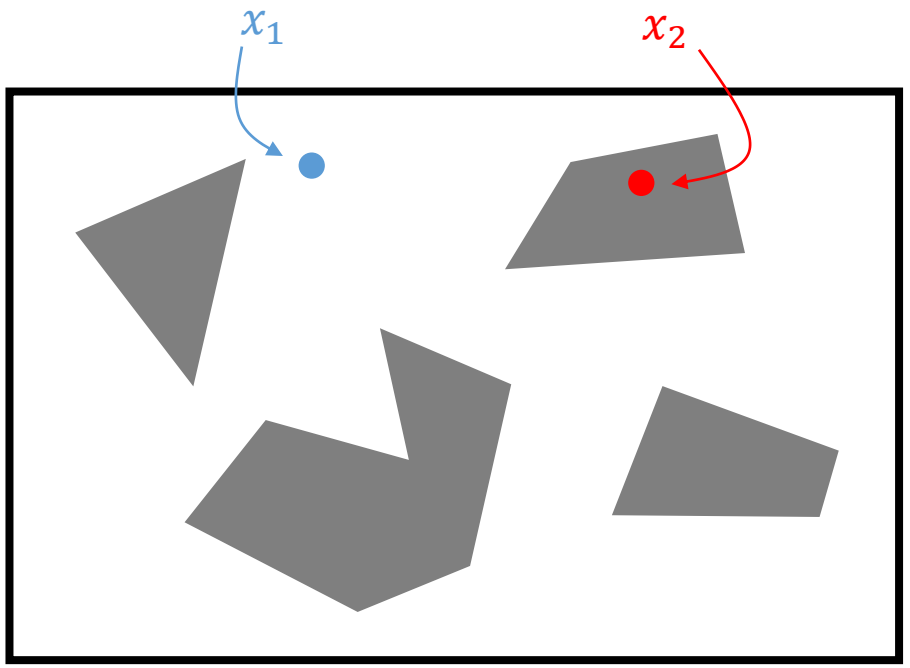
Utilização

- Adicionar q_{start} e q_{goal} ao grafo
- Realizar a busca pelo melhor caminho

Probabilistic Roadmaps

Construção

- Usando uma distribuição qualquer de probabilidades, gerar uma amostra x no domínio do $\mathcal{C}$-space $(\mathbb{R}^2, SE(2), SE(3))$
  - Distribuição uniforme é a mais simples
- Verificar que a amostra gerada é válida → $q(x) \in \mathcal{C}_{free}$
- Conectar a amostra/vértice aos k vizinhos mais próximos



Utilização

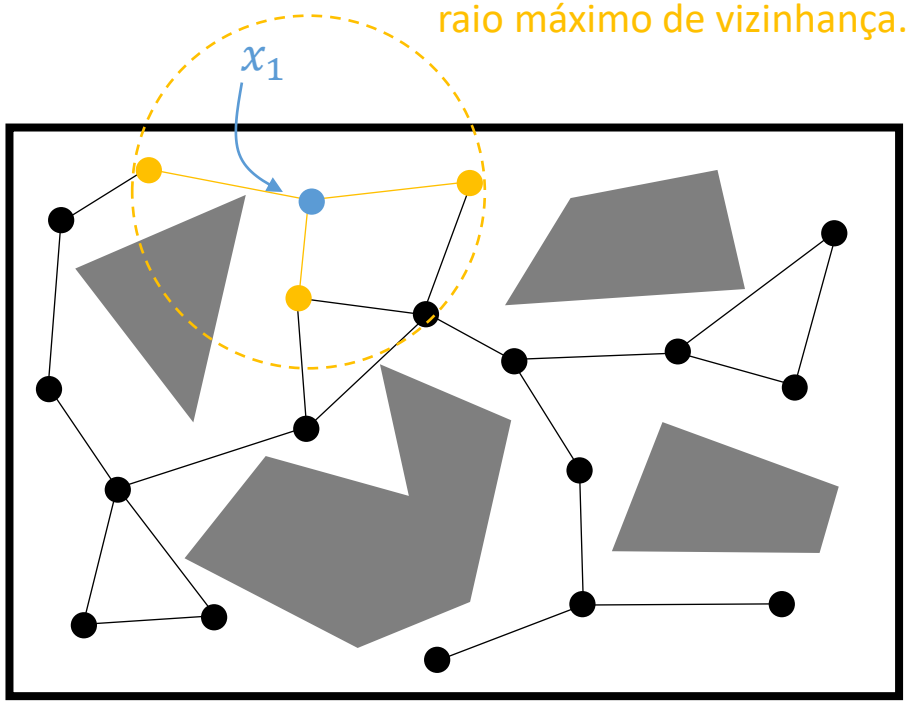
- Adicionar q_{start} e q_{goal} ao grafo
- Realizar a busca pelo melhor caminho

Probabilistic Roadmaps

Construção

- Usando uma distribuição qualquer de probabilidades, gerar uma amostra x no domínio do $\mathcal{C}$-space $(\mathbb{R}^2, SE(2), SE(3))$
  - Distribuição uniforme é a mais simples
- Verificar que a amostra gerada é válida → $q(x) \in \mathcal{C}_{free}$
- Conectar a amostra/vértice aos k vizinhos mais próximos

Pode ser combinado com um raio máximo de vizinhança.



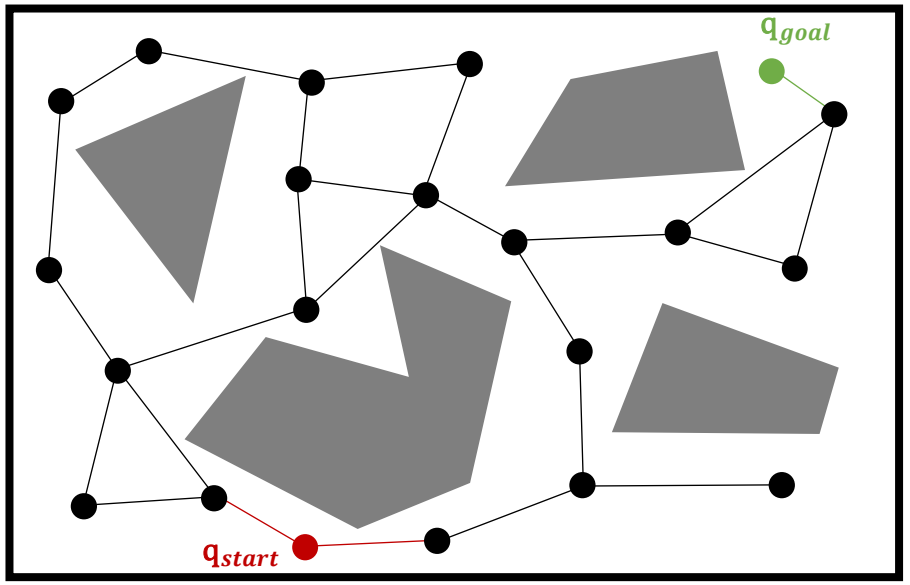
Utilização

- Adicionar q_{start} e q_{goal} ao grafo
- Realizar a busca pelo melhor caminho

Probabilistic Roadmaps

Construção

- Usando uma distribuição qualquer de probabilidades, gerar uma amostra x no domínio do $\mathcal{C}$-space $(\mathbb{R}^2, SE(2), SE(3))$
  - Distribuição uniforme é a mais simples
- Verificar que a amostra gerada é válida → $q(x) \in \mathcal{C}_{free}$
- Conectar a amostra/vértice aos k vizinhos mais próximos



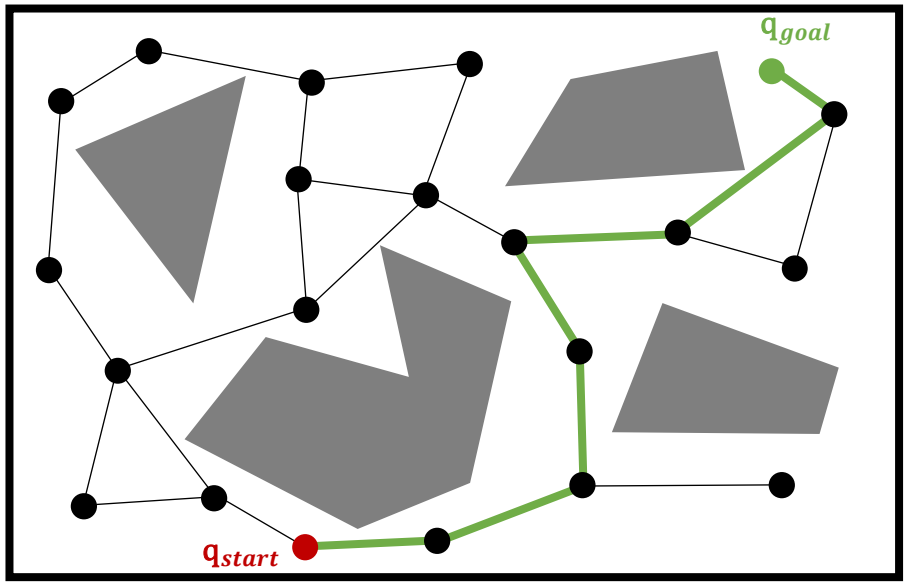
Utilização

- Adicionar q_{start} e q_{goal} ao grafo
- Realizar a busca pelo melhor caminho

Probabilistic Roadmaps

Construção

- Usando uma distribuição qualquer de probabilidades, gerar uma amostra x no domínio do $\mathcal{C}$-space $(\mathbb{R}^2, SE(2), SE(3))$
  - Distribuição uniforme é a mais simples
- Verificar que a amostra gerada é válida → $q(x) \in \mathcal{C}_{free}$
- Conectar a amostra/vértice aos k vizinhos mais próximos



Utilização

- Adicionar q_{start} e q_{goal} ao grafo
- Realizar a busca pelo melhor caminho

Probabilistic Roadmaps
Desafios

- Criação das arestas → Planejador local
  - Encontrar um caminho válido entre duas amostras
  - Linha reta → Robô holonômico (atua em todos os DoF)
  - Robô com restrições → Integrar por Δt usando $\dot{x} = f(x, u)$
- Geralmente maior tempo gasto na checagem de colisão
  - Checar a amostra é rápido, verificar a aresta pode ser custoso
- Qual distribuição usar? E se usarmos outras informações?

Probabilistic Roadmaps
Melhorias

- Pós-processamento
  - Adicionar/remover vértices auxiliares
  - Suavizar/reduzir o caminho atual
- Diferentes distribuições probabilísticas
  - Considerar "tendências" na inserção dos pontos
- Diferentes planejadores locais

Probabilistic Roadmaps

- **Probabilisticamente completo**
  - Com o aumento do número de amostras N (iterações), uma solução (caminho) admissível será encontrada (se houver)

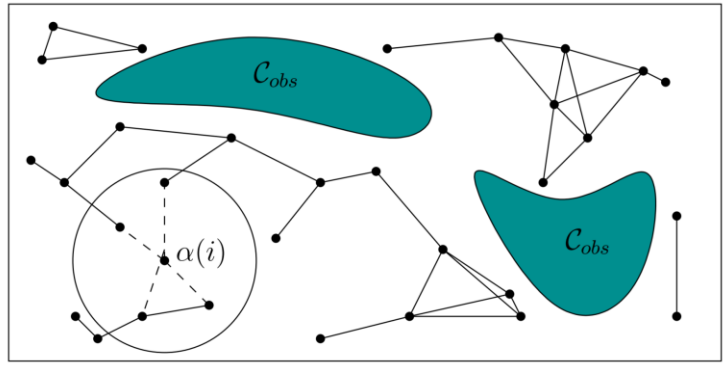
$$\lim_{N \to \infty} \Pr(encontrar\ solução) = 1$$

- **Assintoticamente ótimo**
  - Eventualmente, encontra solução ótima com aumento de amostras

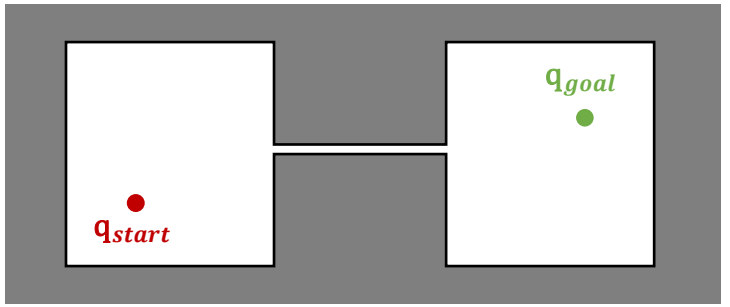
$$\lim_{N \to \infty} \Pr(custo\ solução = ótimo) = 1$$

Probabilistic Roadmaps

- Vantagens
  - Representação *simplificada* do *C*-space
- Desvantagens
  - Não é determinístico
  - Caminhos geralmente ineficientes
  - Dificuldades em certos ambientes
    - Ex: locais com passagens estreitas

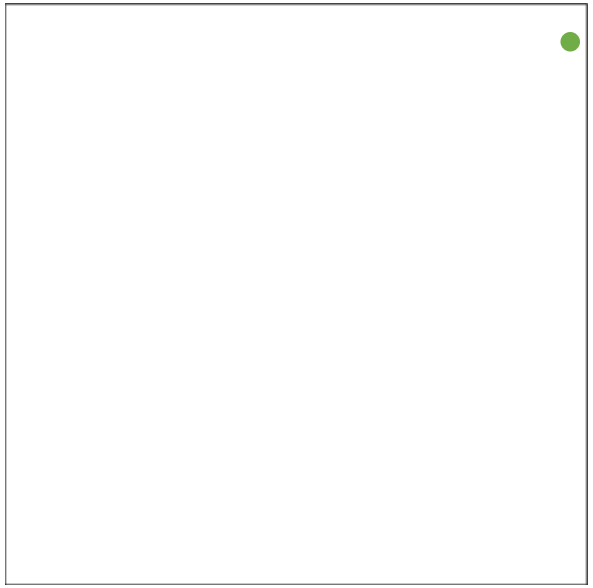


Fonte: *Planning Algorithms*



Rapidly-Exploring Random Tree

- Expandir uma árvore aleatoriamente de maneira incremental a partir da raiz (q_{start}) até um galho estar suficientemente perto do goal (q_{goal})
- Principal característica
  - Viabilidade do caminho, não otimalidade
- Diversas extensões na literatura
  - RRT*, CL-RRT, DMA-RRT, RG-RRT, ...



http://msl.cs.uiuc.edu/rrt/

Rapidly-Exploring Random Tree

GENERATE_RRT($x_{init}, K, \Delta t$)
1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$)
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T

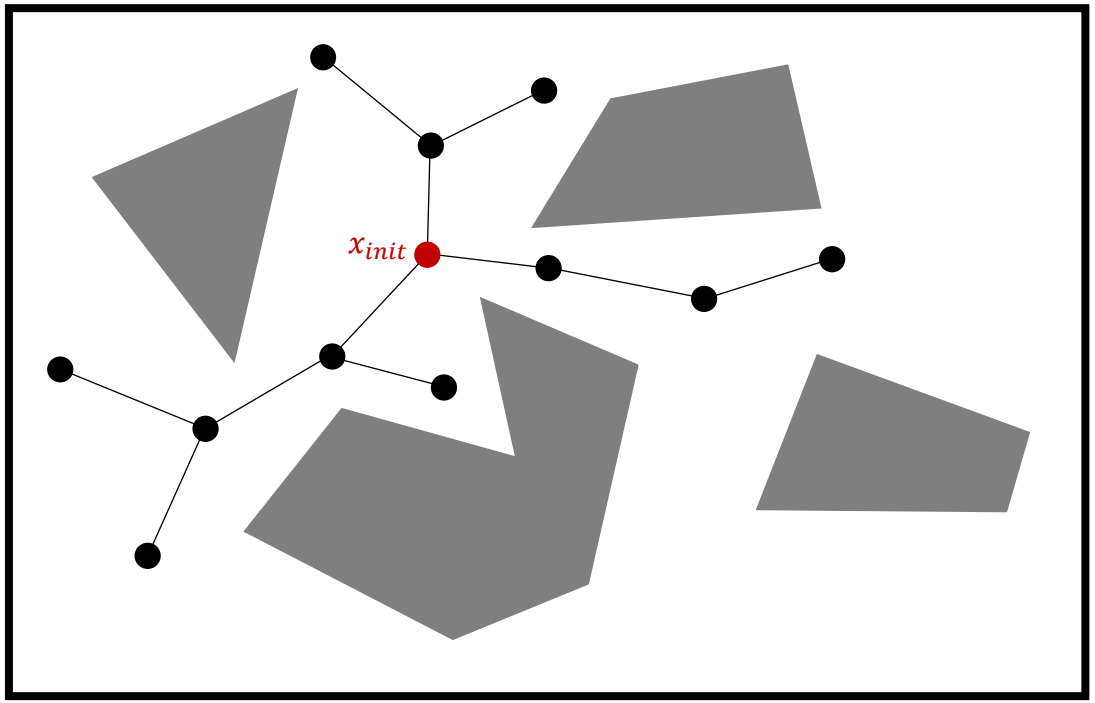
O algoritmo recebe como parâmetros básicos o ponto inicial (raiz), o número máximo de vértices a serem inseridos, e o passo de integração usado para criação do novo estado.

A árvore é iniciada vazia, e enquanto o número máximo de vértices não for inserido os seguintes passos são executados. Também podemos interromper a execução se um caminho até o goal já tiver sido encontrado.

Rapidly-Exploring Random Tree

GENERATE_RRT($x_{init}, K, \Delta t$)

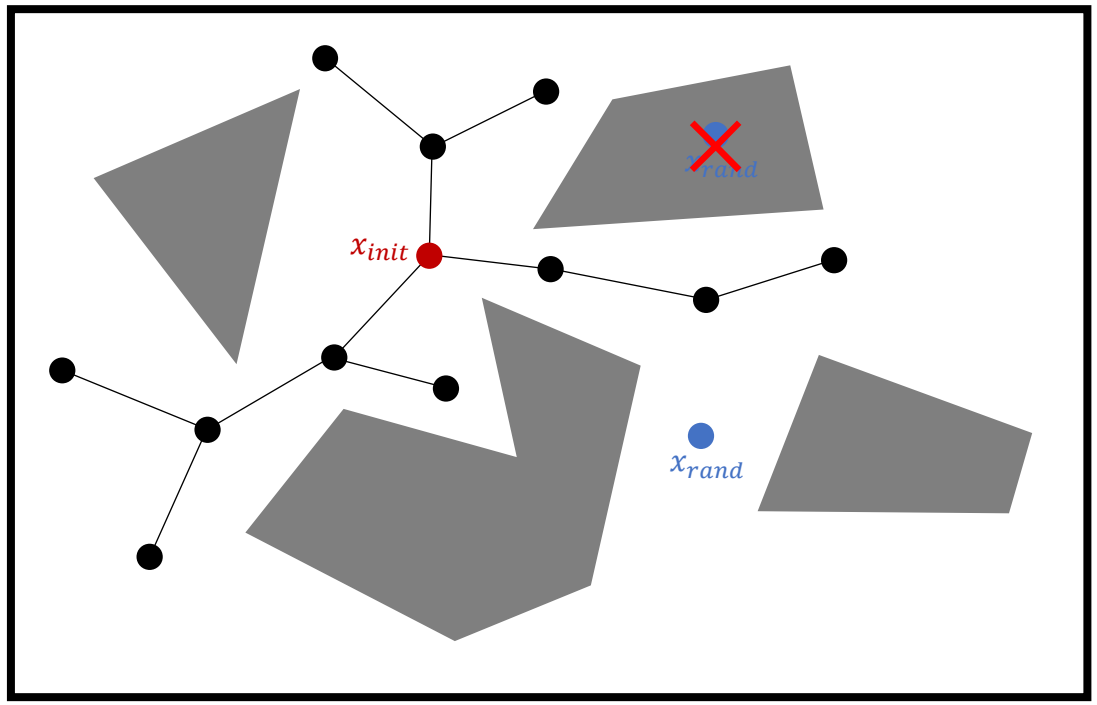
1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T



Rapidly-Exploring Random Tree

GENERATE_RRT($x_{init}, K, \Delta t$)

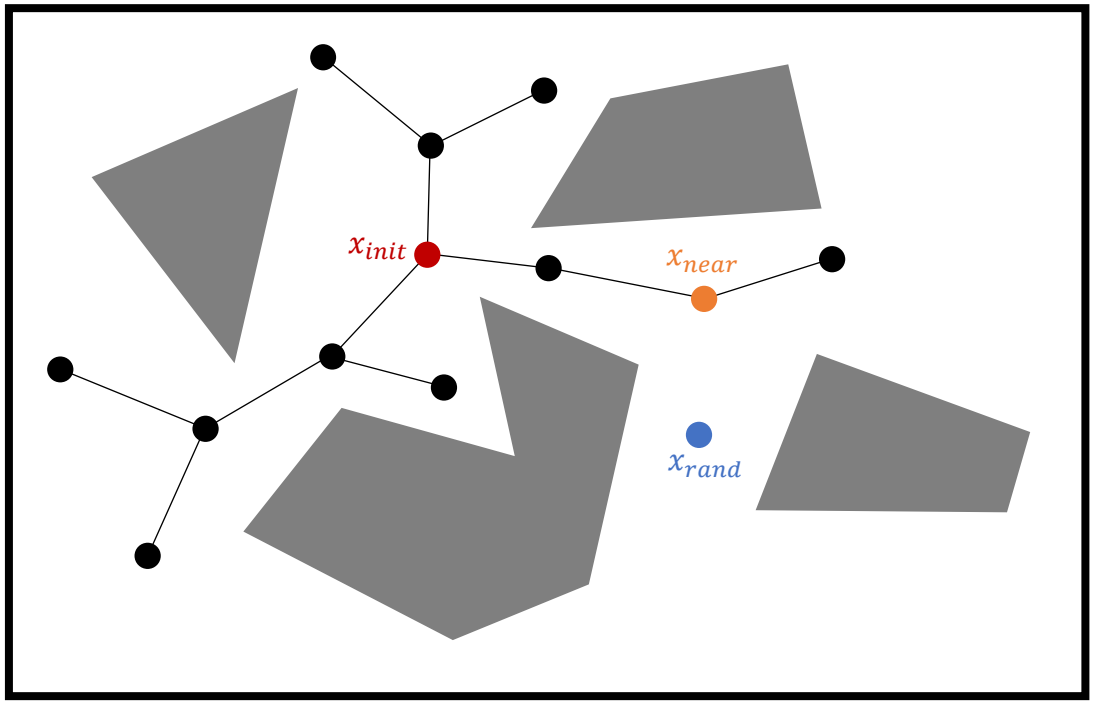
1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T



Rapidly-Exploring Random Tree

GENERATE_RRT($x_{init}, K, \Delta t$)

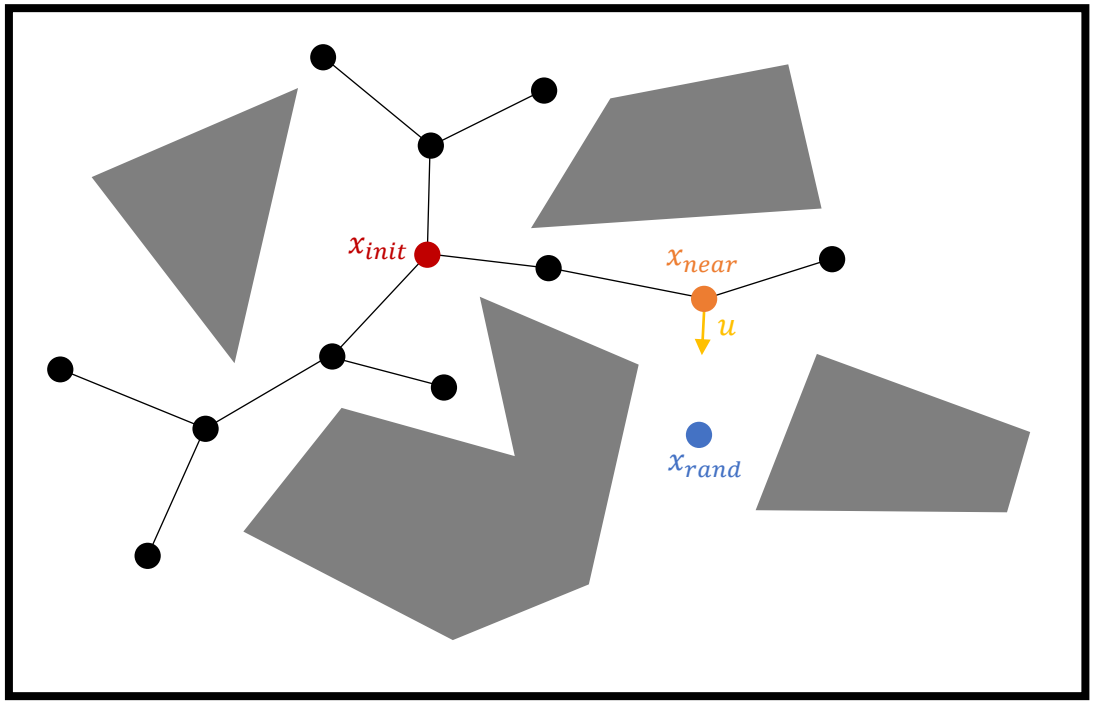
1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T



Rapidly-Exploring Random Tree

GENERATE_RRT($x_{init}, K, \Delta t$)

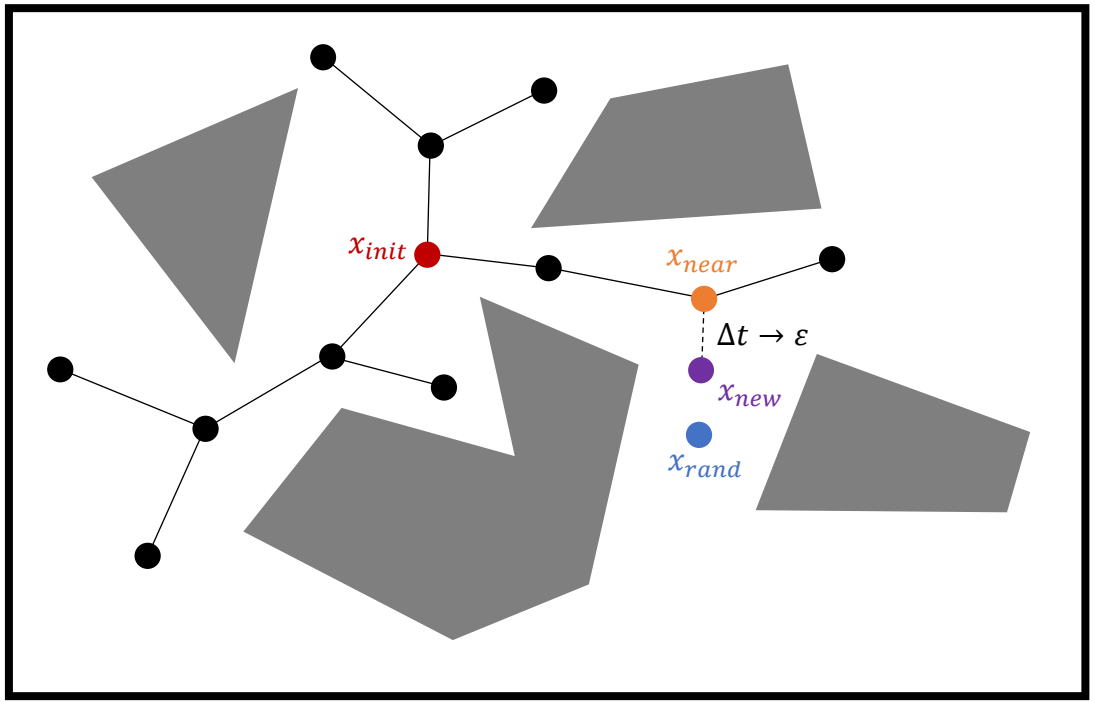
1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T



Rapidly-Exploring Random Tree

GENERATE_RRT($x_{init}, K, \Delta t$)

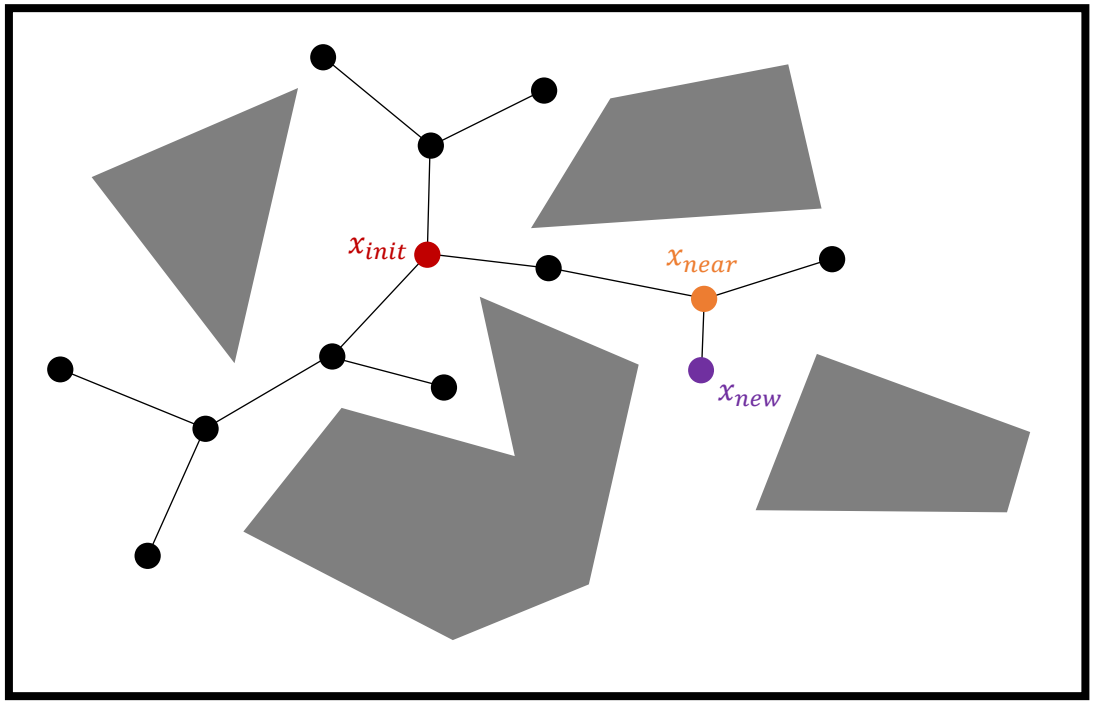
1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T



Rapidly-Exploring Random Tree

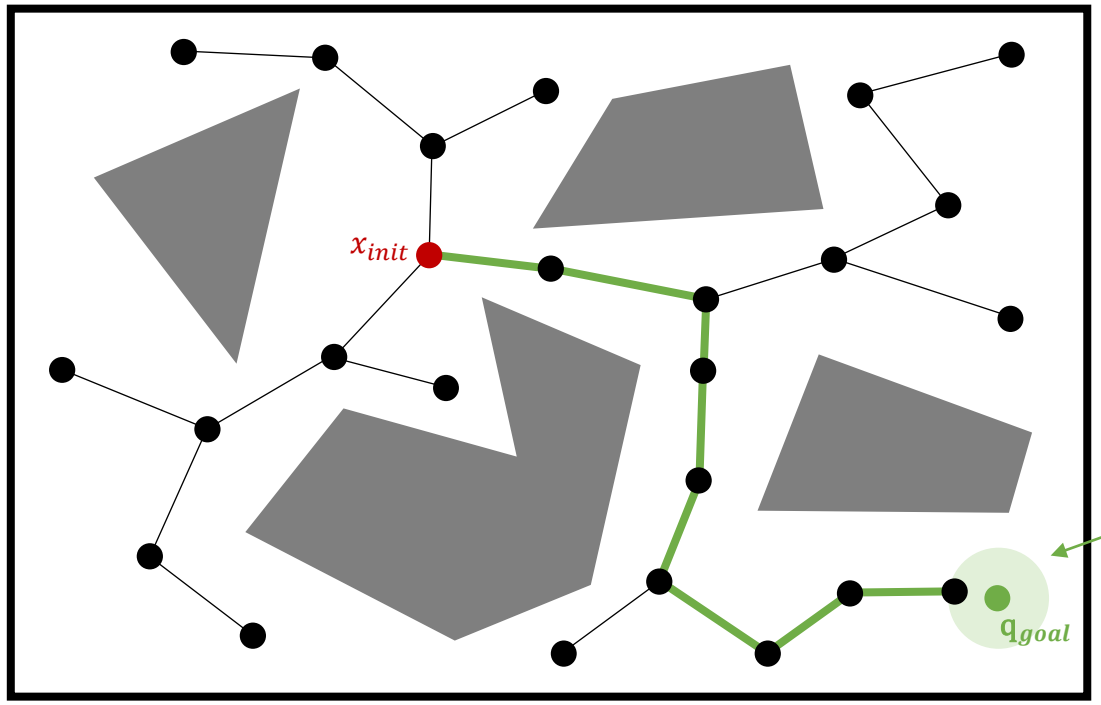
GENERATE_RRT($x_{init}, K, \Delta t$)

1 T.init();
2 **for** $k = 1$ **to** K **do**
3 $x_{rand} \leftarrow$ RANDOM_STATE();
4 $x_{near} \leftarrow$ NEAREST_NEIGHBOR(x_{rand}, T);
5 $u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $x_{new} \leftarrow$ NEW_STATE($x_{near}, u, \Delta t$);
7 T.add_vertex(x_{new});
8 T.add_edge(x_{near}, x_{new}, u);
9 **return** T



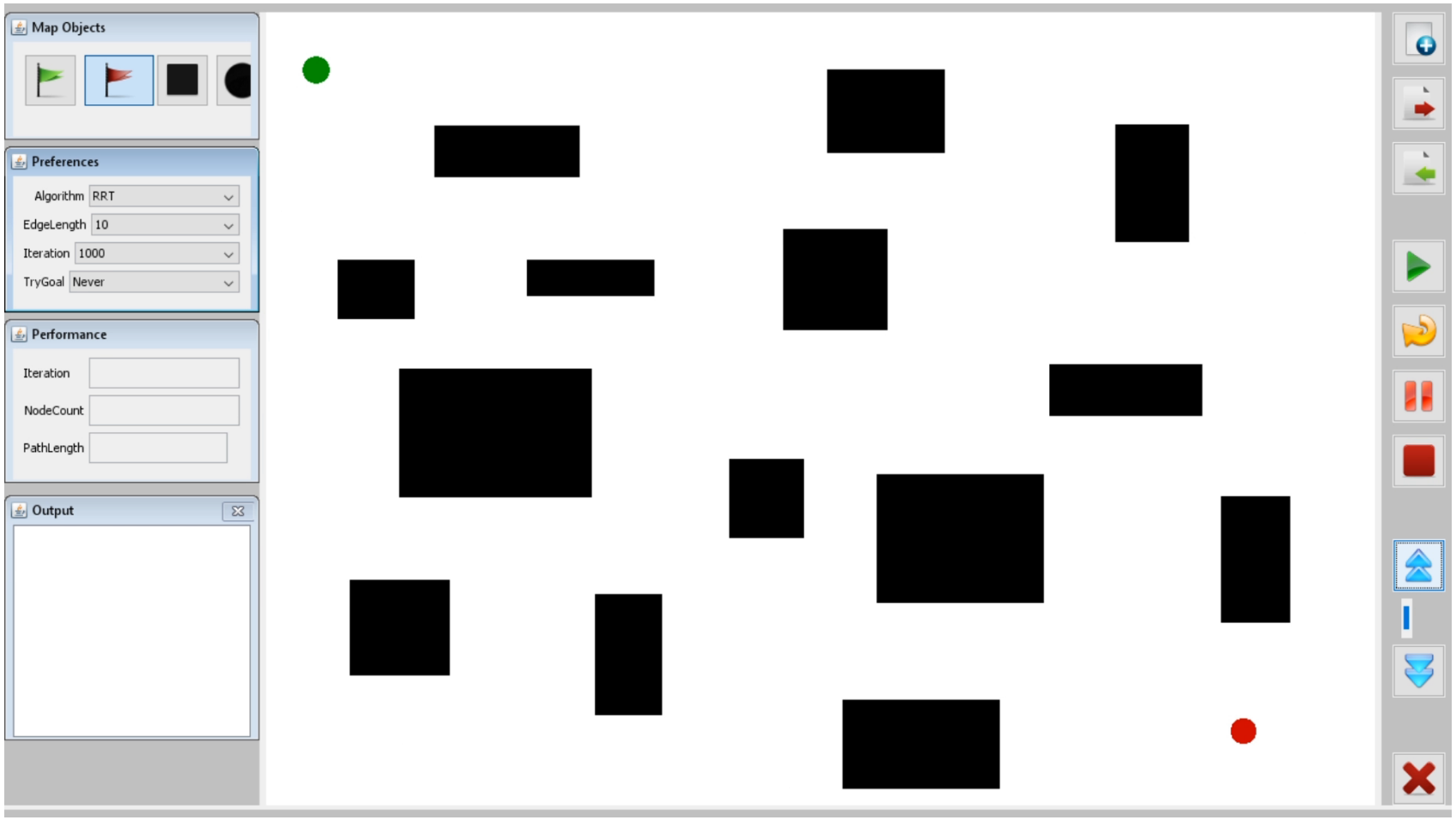
Se não ocorrer nenhuma colisão no trajeto entre x_{near} e x_{new} o vértice e a aresta podem ser adicionados à árvore.

Rapidly-Exploring Random Tree



Geralmente vamos considerar uma região ao redor da posição alvo como suficiente para terminar a execução do algoritmo.

Rapidly-Exploring Random Tree



https://youtu.be/jVKEoXL5-ns

iRRT Simulator: http://correll.cs.colorado.edu/?p=1623

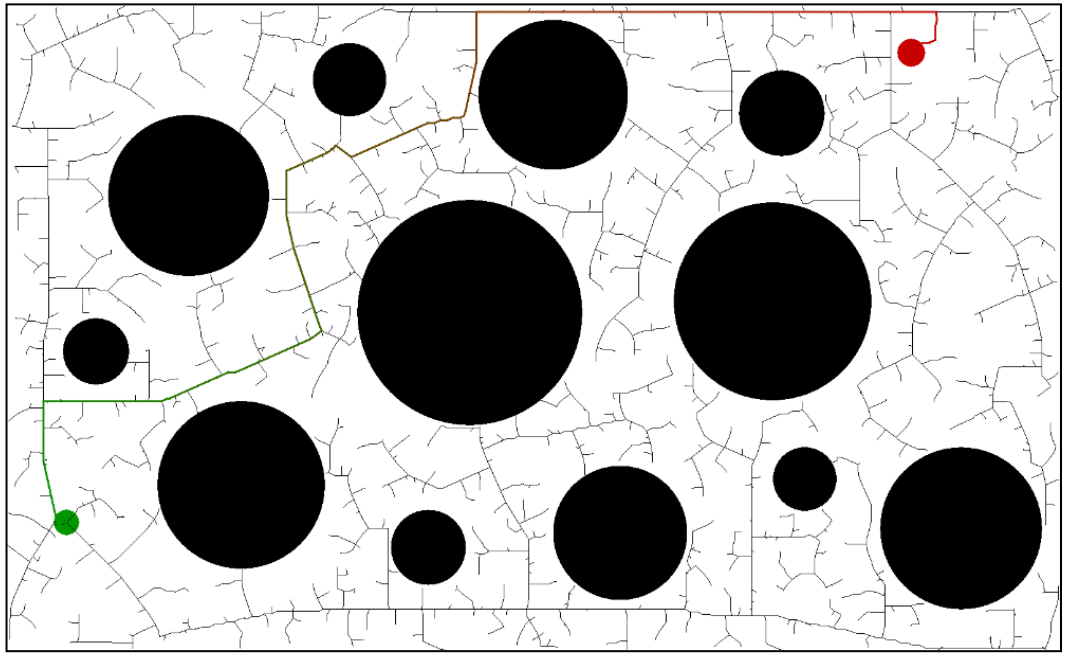
Rapidly-Exploring Random Tree
Melhorias

- Exploração ocorre em todas as direções
  - Perda de tempo olhar em direções desnecessárias
  - Adicionar um bias na busca (direcionar)
    - Porcentagem das amostras é o próprio goal

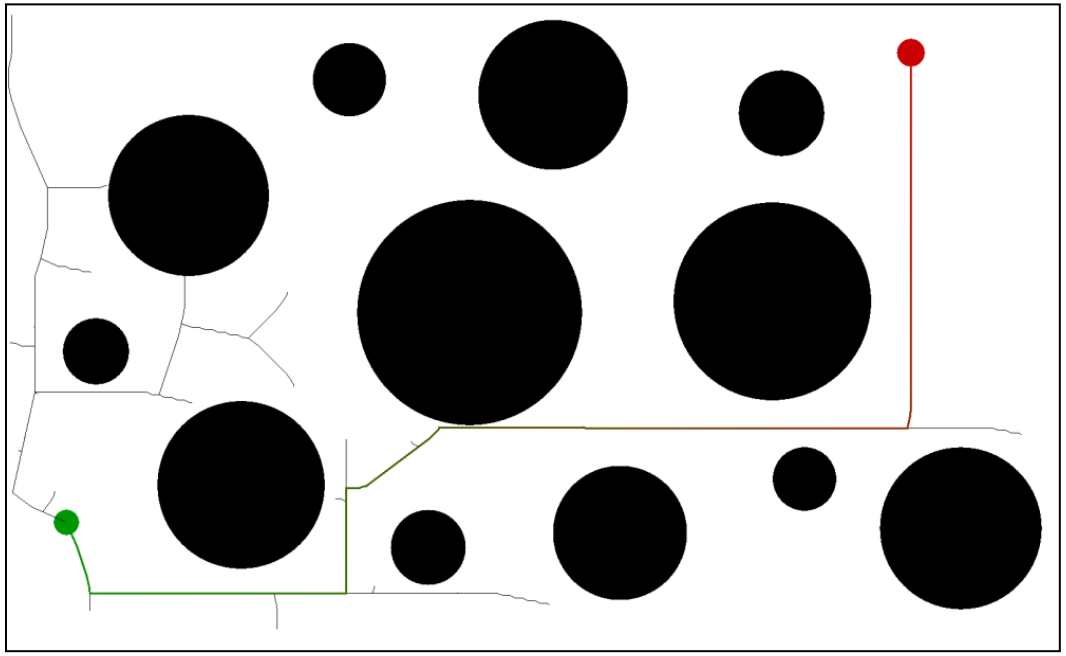
- Árvores bi-direcionais
  - Crescer a árvore em ambas direções
  - Tentar conectar os vértices das árvores

Rapidly-Exploring Random Tree

Melhorias – Bias



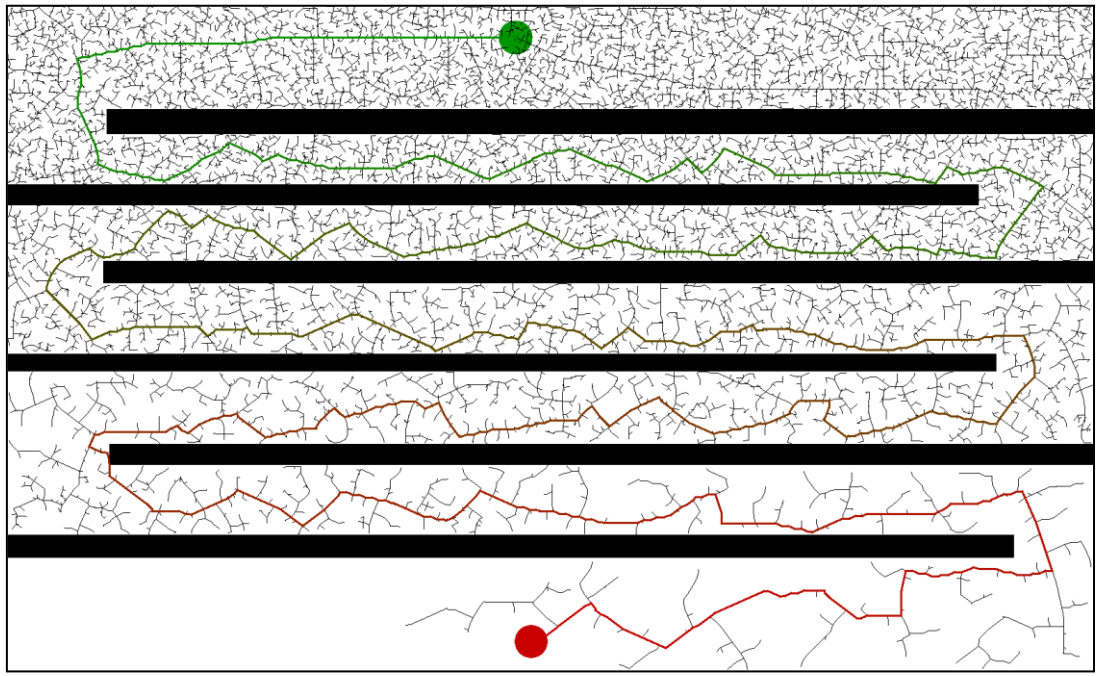
0%



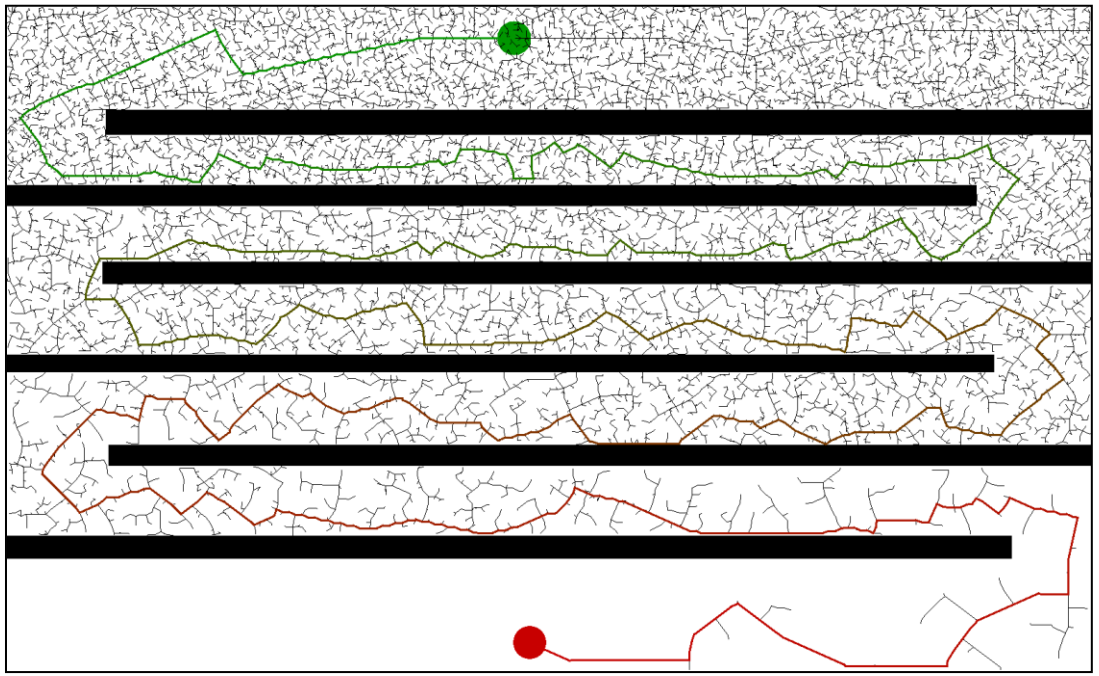
25%

Rapidly-Exploring Random Tree

Melhorias – Bias



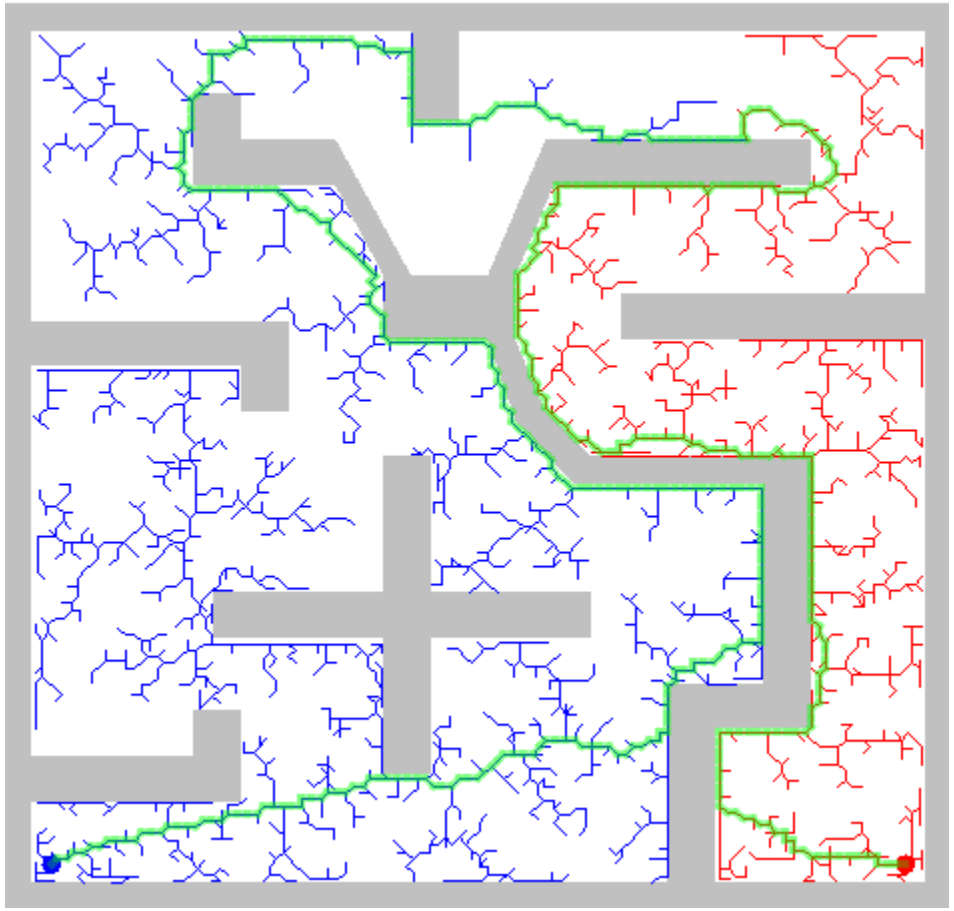
0%



50%

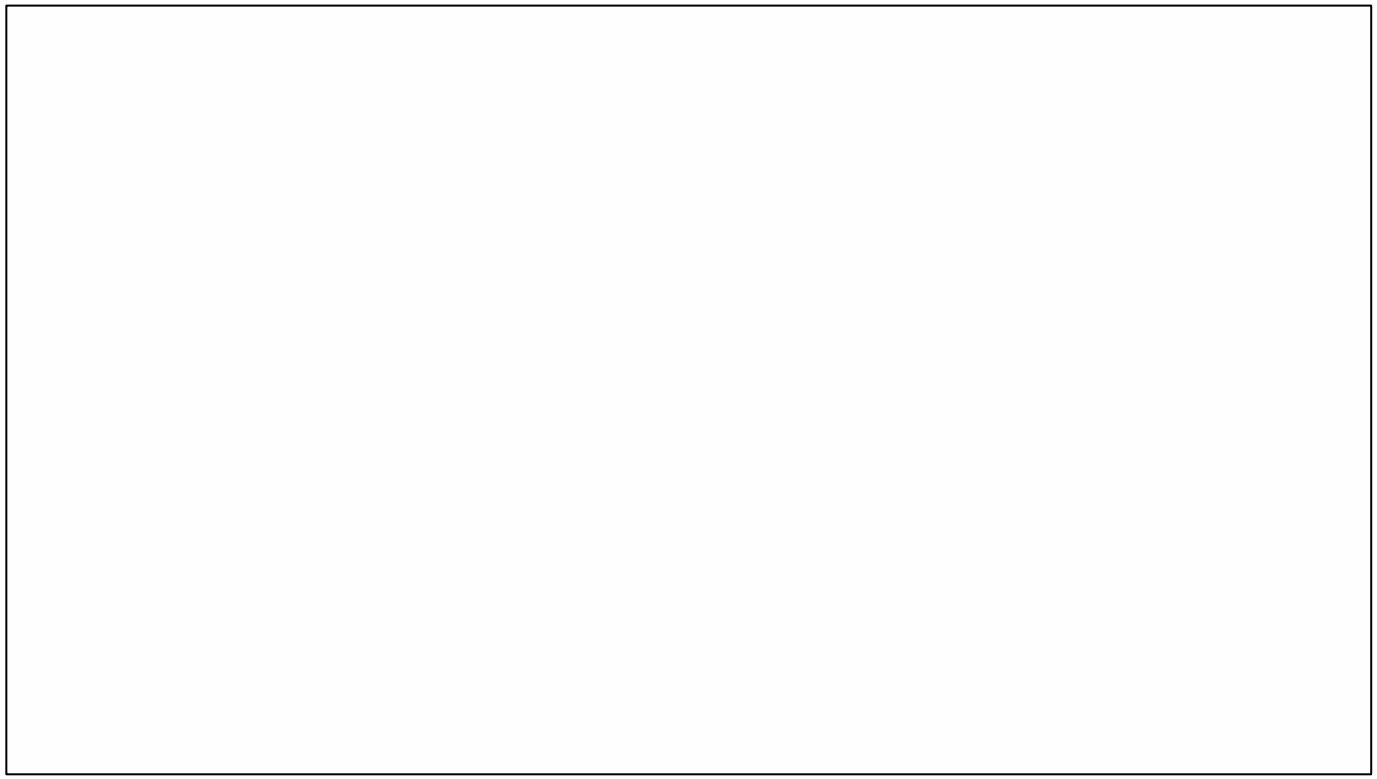
Rapidly-Exploring Random Tree

Melhorias – Bi-direcional



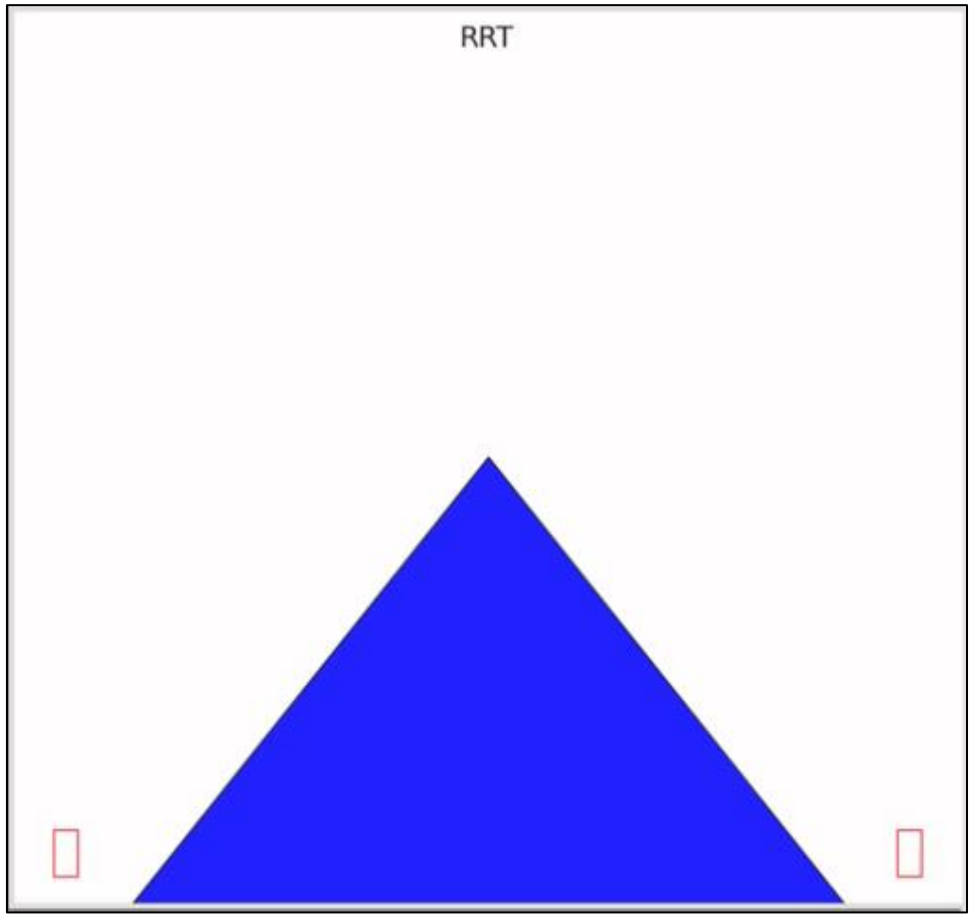
Rapidly-Exploring Random Tree
Melhorias

Informed RRT*



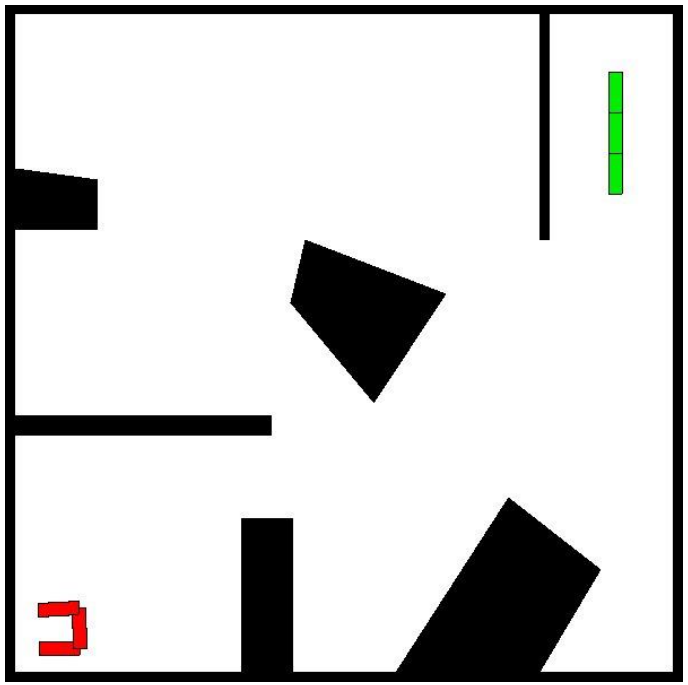
https://www.youtube.com/watch?v=nsl-5MZfwu4

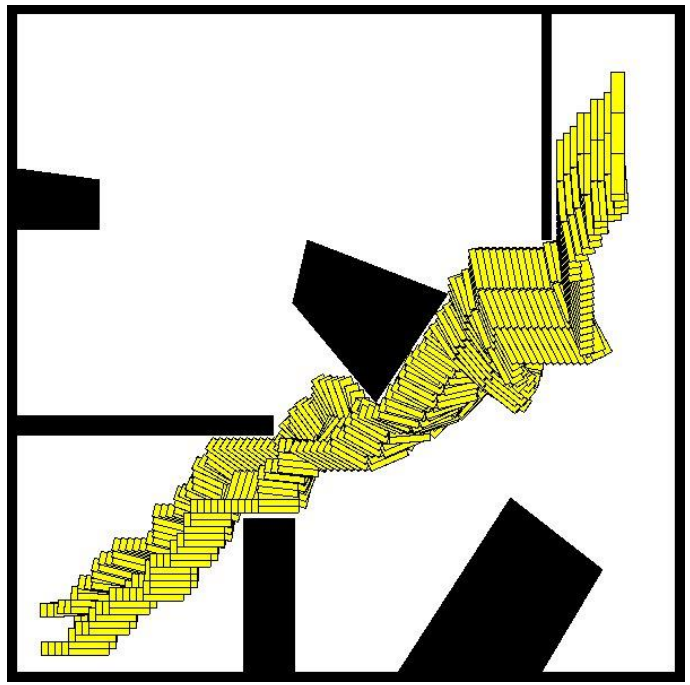
Rapidly-Exploring Random Tree
Múltiplos DoF

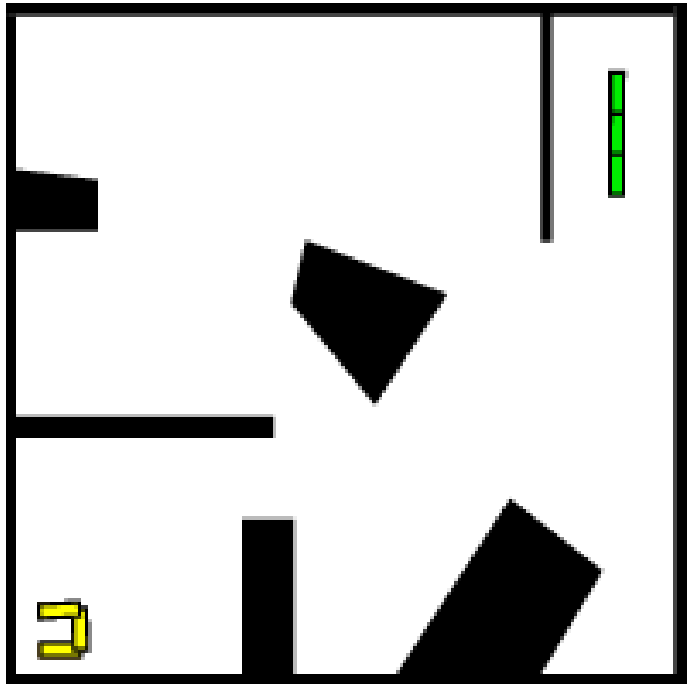


https://youtu.be/rPgZyq15Z-Q

Rapidly-Exploring Random Tree
Múltiplos DoF

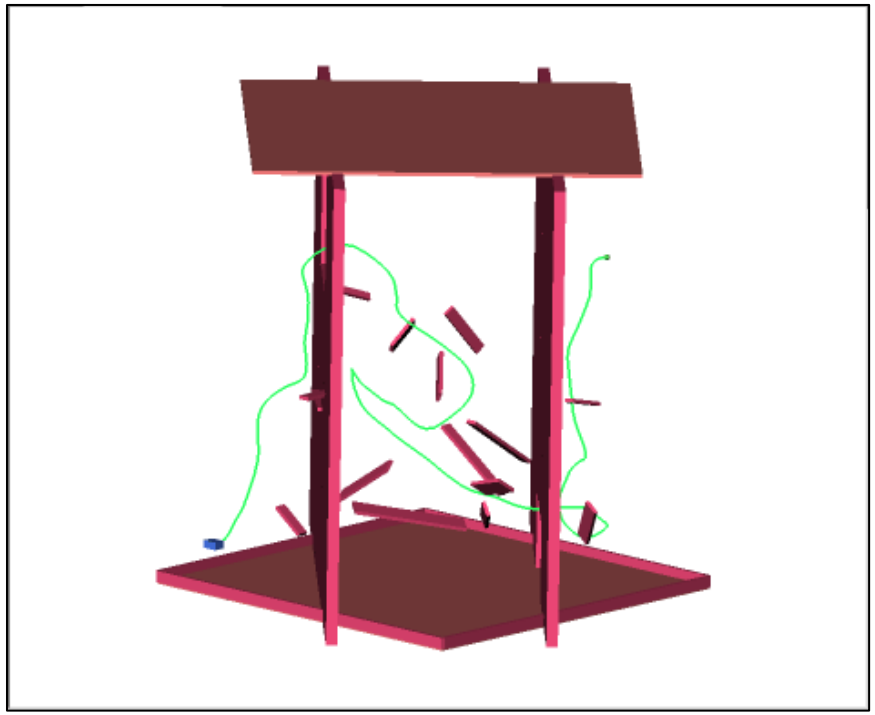


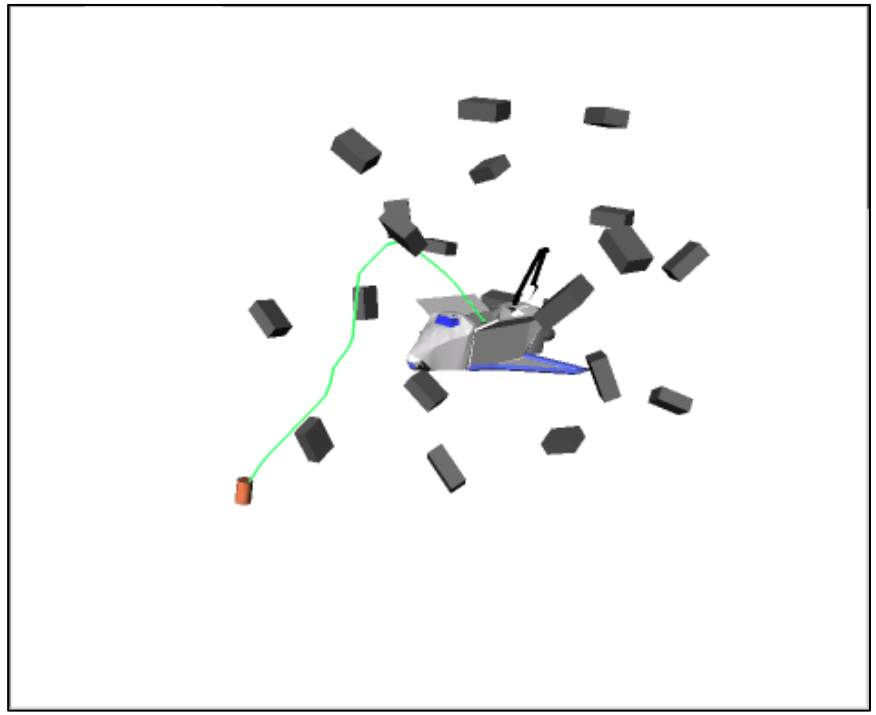




http://lavalle.pl/rrt/gallery_chain.html

Rapidly-Exploring Random Tree
Múltiplos DoF





http://lavalle.pl/rrt/gallery_3dnorot.html
http://lavalle.pl/rrt/gallery_3drot.html

Rapidly-Exploring Random Tree
Restrições cinemáticas

- Dado um estado x e entrada u, temos que

$$\dot{\mathrm{x}} = f(\mathrm{x}, u)$$

- Considerar o modelo cinemático

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & \dfrac{\tan u_\phi}{L} \end{bmatrix} \begin{bmatrix} u_s \\ u_\phi \end{bmatrix}$$

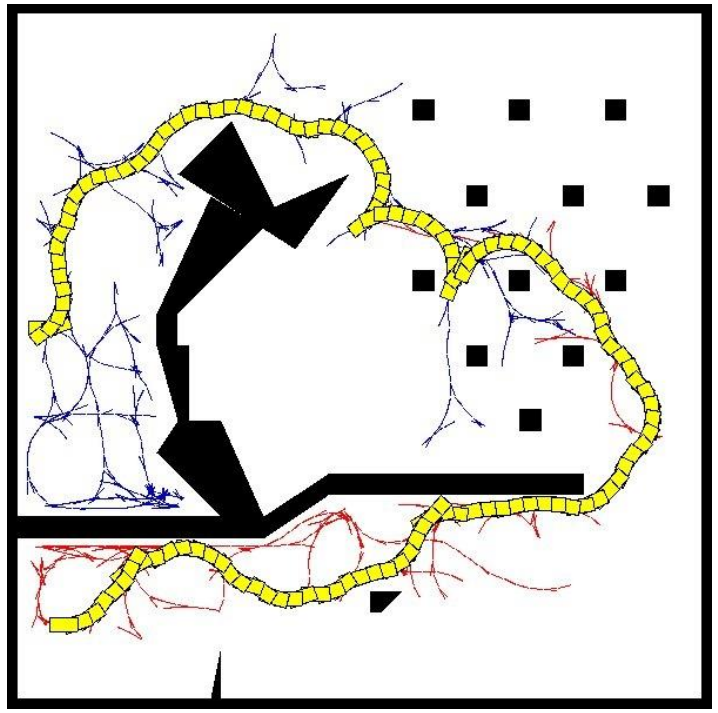
Você também pode usar soluções analíticas, por exemplo, curvas de Dubins!

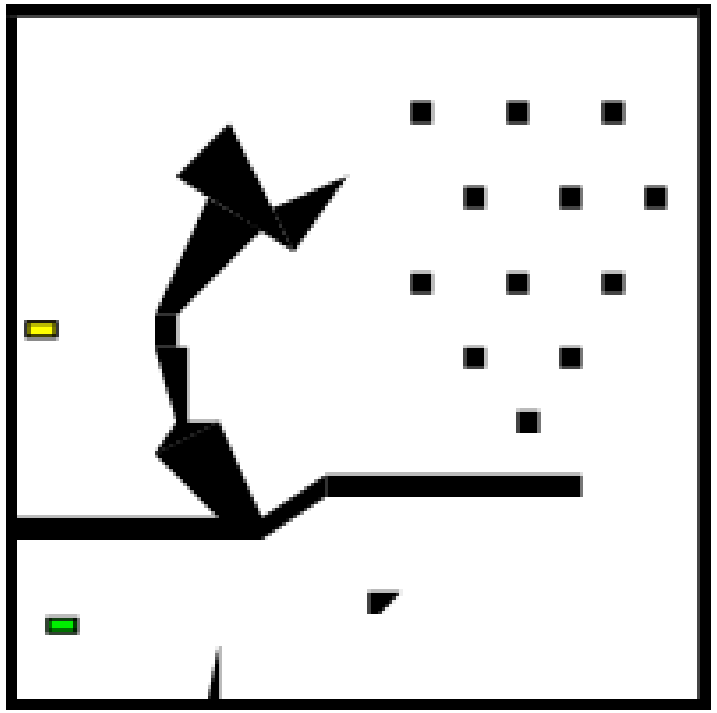
- Deve-se realizar uma integração numérica

Dubins curves: http://planning.cs.uiuc.edu/node821.html

Rapidly-Exploring Random Tree

Restrições cinemáticas





http://lavalle.pl/rrt/gallery_car.html

Rapidly-Exploring Random Tree

- Vantagens
  - Simples de implementar
  - Fácil adicionar restrições mais complexas
  - Probabilisticamente completo

- Desvantagens
  - Não determinístico (assintoticamente ótimo)
  - Caminhos geralmente ineficientes

Considerações finais
PRM vs. RRT

- Probabilistic Roadmaps (PRM)
  - *Multiple-query model*
  - Grafo gerado pode ser usados várias vezes

- Rapidly-Exploring Random Tree (RRT)
  - *Single-query model*
  - Árvore com foco em uma única utilização