PLANEJAMENTO DE CAMINHOS

MÉTODOS BASEADOS EM AMOSTRAGEM

Introdução:

Os métodos determinísticos de planejamento de caminhos são adequados para espaços de configuração de baixa dimensão, onde geralmente os obstáculos são descritos por modelos contínuos. Para problemas de planejamento em espaços de configuração de dimensão elevada e onde os obstáculos são descritos na forma de grade de ocupação, métodos probabilísticos são mais adequados.

Princípio:

- Amostrar aleatoriamente o espaço de configuração.
- Testar se as configurações amostradas estão no espaço livre.
- Tentar ligar pares configurações amostradas no espaço de configuração livre por caminhos livres.
- Construir uma rede de caminhos livres entre as configurações amostradas no espaço de configuração livre, incluindo a as configurações inicial e final.
- Buscar um caminho entre q_{ini} e q_{fin} na rede de caminhos construída.

Mapa de Rotas Probabilístico:

O método de planejamento Mapa de Rotas Probabilístico (PRM – *Probabilistic Road Map*) é executado em duas etapas: Construção do Mapa de Rotas e Busca de Caminho.

- Construção do Mapa de Rotas:

- Parte-se do pressuposto que o espaço de trabalho é estático.
- Esta etapa é computacionalmente custosa, mas pode ser feita off-line, previamente à busca de caminho.
- Nesta etapa é construído um grafo R(N,E), (com N igual ao seu conjunto de nós e E seu conjunto de arestas), contido no espaço de configuração livre $\mathbf{C_L}$, cujo objetivo é capturar a conectividade de $\mathbf{C_L}$, onde $\mathbf{C_L}$ tem dimensão m, sendo m o número de graus de liberdade do robô.
- Configurações aleatórias são geradas sucessivamente em $\mathbf{C_L}$.
- Cada configuração q_{nova} gerada é armazenada em um nó do grafo.
- Através de um planejamento local, verifica-se quais configurações vizinhas q_k às quais q_{nova} pode se conectar.
- Caso não haja colisão no caminho entre q_{nova} e q_j, e se ambas não pertencerem ao mesmo componente conexo do grafo R (para evitar gerar laços no grafo), criar aresta nova em E entre estes dois nós.

- Busca de Caminho no Mapa de Rotas:

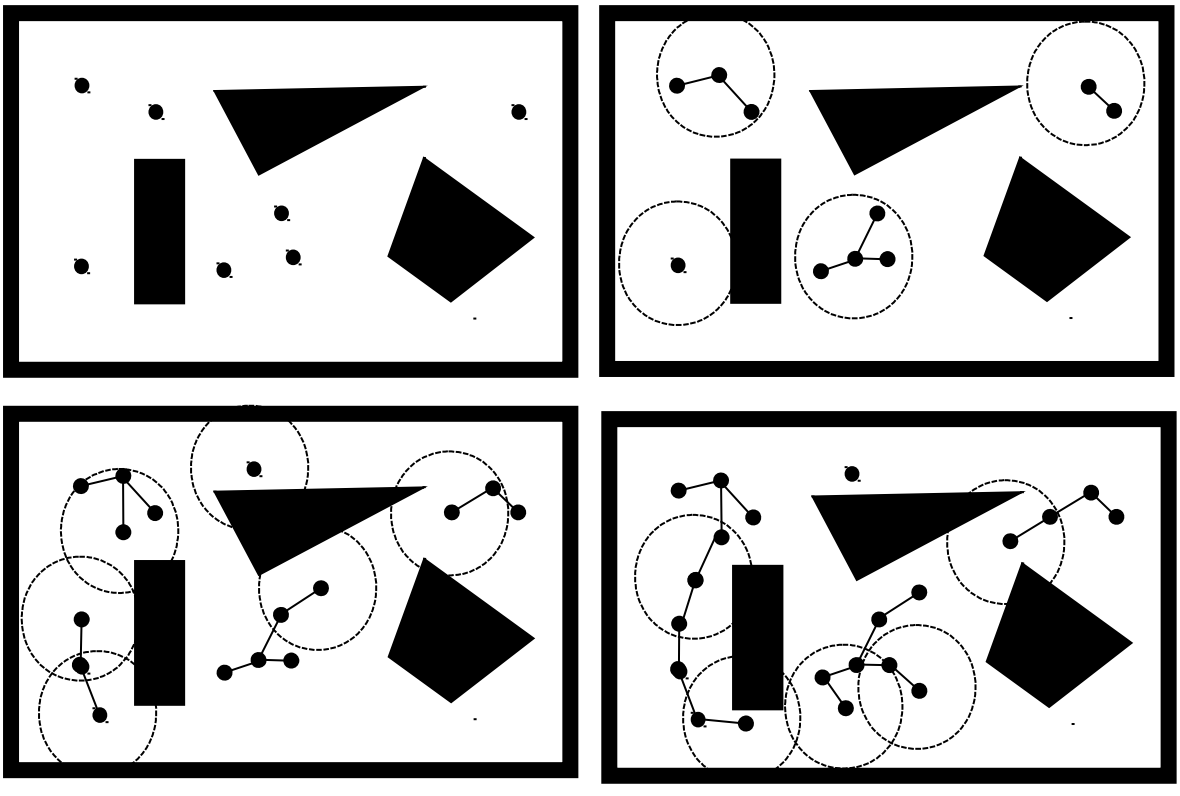
- Uma vez construído o mapa de rotas, o mesmo pode ser usado para buscar um caminho livre entre q_{ini} e q_{fin}.
- Esta etapa pode ser feita rapidamente, usando o grafo construído na etapa de construção.
- Liga-se q_{ini} à configuração mais próxima no grafo R, q_{ini}^R e liga-se q_{fin} à configuração mais próxima no grafo R, q_{fin}^R.
- Inicialmente, verifica-se se q_{ini}^R e q_{fin}^R estão no mesmo componente conexo de R. caso não estejam, reportar falha.
- Caso q_{ini}^R e q_{fin}^R estejam no mesmo componente conexo de R, buscar caminho entre elas no grafo R e retornar o caminho composto pelos nós q_{ini}, q_{ini}^R, sequência de nós ligando q_{ini}^R a q_{fin}^R, q_{fin}^R e q_{fin}.

Algoritmo para a Etapa de Construção do PRM

R(N,E)

Inicializar:
N_{max} = Número máximo de configurações do grafo.
N_{vmax} = Número máximo de configurações vizinhas.
R_V = Raio de vizinhança em torno de configuração nova q_{nova}.

```
1   i ← 1
2   N ← ∅
3   E ← ∅
4   enquanto i < Nmax faça
5         qnova ← configuração aleatória em C
6         se qnova ∈ CL então
7               i ← i+1
8               inserir qnova em N
9               selecionar até Nvmax configurações vizinhas qk
                (k = 1, ..., Nvmax), tal que ||qk-qnova||<RV
10              para j = 1 até Nvmax faça
11                  se (∃ caminho livre entre qk e qnova) e
                    (qk e qnova ∉ mesmo componente conexo de R) então
12                      inserir em E nova aresta conectando qk e qnova
13                  fim se
14              fim para
15        fim se
16  fim enquanto
```



Observações:

- Após construído o Mapa de Rotas, o mesmo pode ser utilizado eficientemente para busca de caminhos (etapa de buscas). Assim, este método é adequado para ambientes estáticos.
- O método pode ser utilizado em espaços de dimensões elevadas, pois após a etapa de construção do Mapa de Rotas, a informação de conectividade é capturada implicitamente em R.
- O número máximo de nós N_{max} depende da topologia de $\mathbf{C}_{\mathbf{L}}$.
- N_{max} pequeno pode no garantir conectividade insuficiente, resultando em um mapa que não contenha a solução, mesmo esta existindo.
- A probabilidade de sucesso (caso exista solução) cresce exponencialmente com o número de amostras.
- O método é probabilisticamente completo. Caso exista solução, a probabilidade de achá-la tende a 1 quando o número de amostras tende a infinito.
- N_{max} grande pode exigir muita memória e um grande tempo de processamento do sistema computacional.
- Vários métodos podem ser utilizados para gerar novas amostras (q_{nova} na linha 5 do algoritmo), influenciando bastante no desempenho do algoritmo.
- A seleção das N_{vmax} configurações vizinhas a q_{nova} situadas a uma distância menor que R_V, (na linha 9 do algoritmo), pode não ser simples. Em espaços de dimensão elevada, esta seleção pode consumir bastantes recursos computacional, exigindo a escolha de estruturas de armazenamento e mecanismos de busca eficientes.
- A busca de um caminho livre entre q_k e q_{nova} (linha 11 do algoritmo) pode ser feita de várias maneiras, usando algum método de planejamento local. Se apenas caminhos constituídos por segmentos de retas são admitidos, um simples teste de visibilidade resolve o problema, caso contrário, outros métodos de planejamento podem ser utilizados, melhorando o desempenho.

- O algoritmo a seguir executa um planejamento local simples entre duas configurações q_j e q_k tentando conectá-las através de um segmento de reta, parametrizado por λ ($\lambda \in [0,1]$), o qual é traçado incrementando o parâmetro λ com resolução $\Delta\lambda$, de q_j a q_k.

Algoritmo de Planejamento Local entre q_j e q_k

PLAN(q_j,q_k)

```
    Inicializar:
    Qj, qk ∈ CL
    Δλ ∈ (0,1), tal que 1/Δλ ∈ ℤ
1   λ ← 0
2   q ← qj
3   enquanto λ < 1 faça
4       se q ∉ CL então
5           retornar Status conexão(qj, qk) = Falso
6       Senão
7           λ ← λ+Δλ
8           q ← (1-λ).qj + λ.qk
9       fim se
10  fim enquanto
11  retornar Status conexão(qj, qk) = Verdadeiro
```

O Algoritmo de Bresenham para traçado incremental de retas também pode ser usado para tentar ligar q_j e q_k em grades regulares.

Árvore Aleatória para Exploração Rápida:

A Árvore Aleatória para Exploração Rápida (RRT – *Rapid-exploring Random Tree*) é um método inicialmente apresentado como um algoritmo de planejamento para buscas rápidas em espaços de alta dimensionalidade.

O princípio básico do algoritmo é polarizar a busca em regiões pouco exploradas do espaço de configuração amostrando pontos no mesmo e empurrando a busca na direção dessas regiões de forma incremental.

A partir da configuração inicial q_{ini}, uma árvore de busca é construída amostrando configurações aleatórias em e tentando conectá-las à arvore corrente, utilizando o espaço circundante para expandir a árvore de busca, levando a uma cobertura uniforme do espaço livre $\mathbf{C_L}$ e, em última instância, alcançando a configuração final q_{fin} ou uma vizinhança Q_{fin} da configuração final.

Assim, uma vantagem do método RRT em relação ao método PRM é que, enquanto neste último, o número de nós é determinado *a priori*, no RRT os nós são gerados de forma incremental.

Versões mais eficientes do método executam uma busca bidirecional, construindo duas árvores, uma partindo de qini e outra de q_{fin}, finalizando a busca quando ambas se encontram. Esta abordagem geralmente não precisa cobrir a maior parte do espaço livre $\mathbf{C_L}$.

O algoritmo abaixo executa a construção incremental da Árvore de Busca T. A cada passo, o algoritmo tenta expandir a árvore adicionando um novo nó que é polarizado por uma configuração escolhida aleatoriamente q_{al}.

Algoritmo para construção da RRT

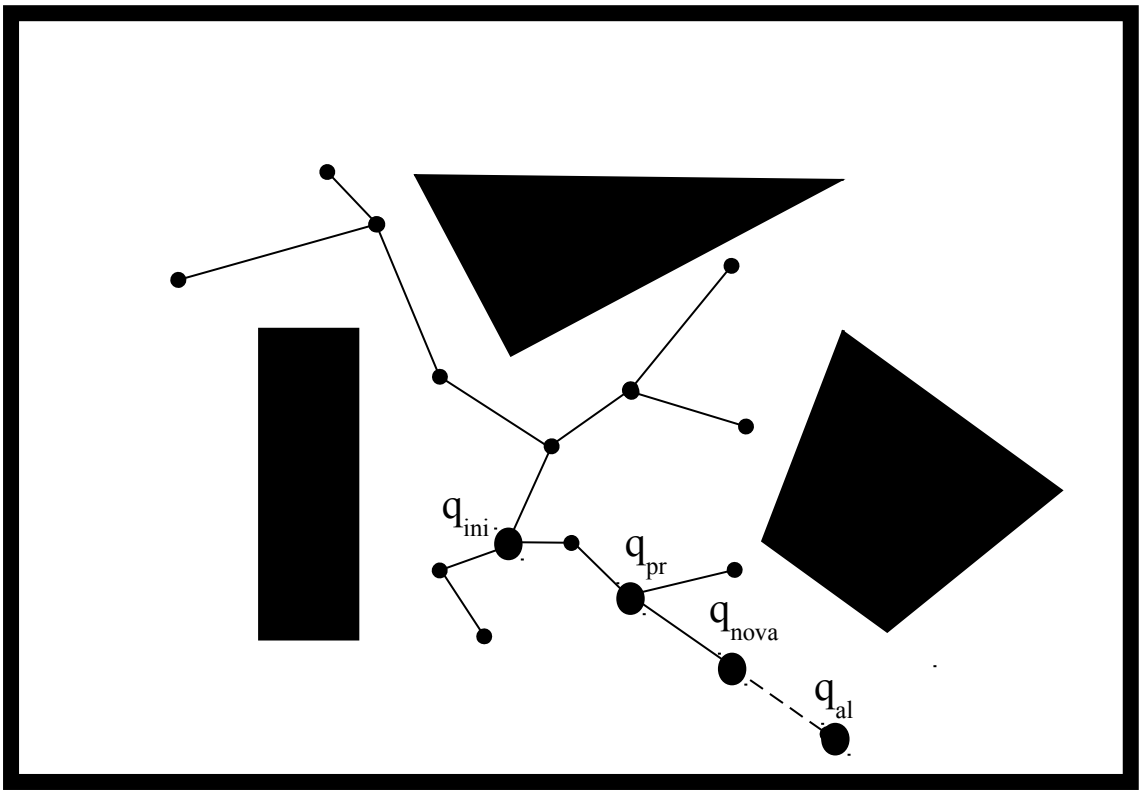
CONSTRUIR_RRT(q_{ini})

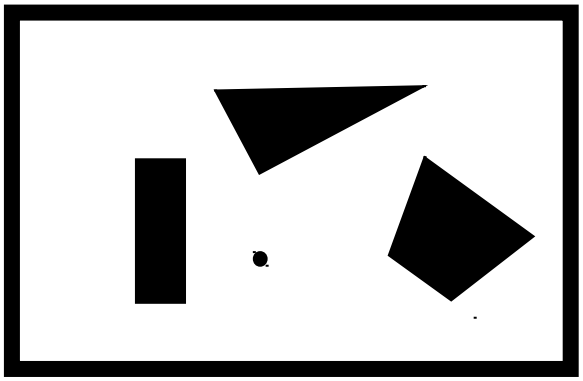
```
     N_max = Número máximo de nós na árvore.
1    T.ini(q_ini)
2    para i = 1 até N_max faça
3         q_al ← configuração aleatória em C
4         EXPANDIR_RRT(T,q_al)
5    Retornar T
```

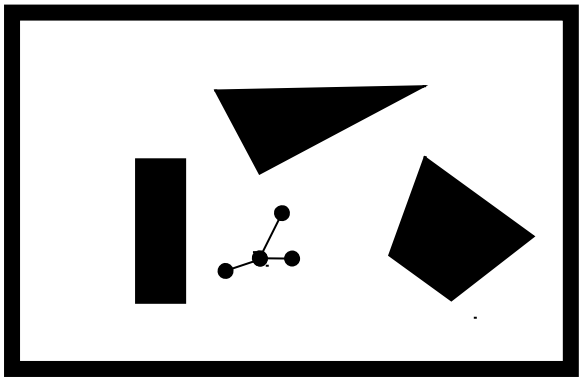
Algoritmo para Expansão da RRT

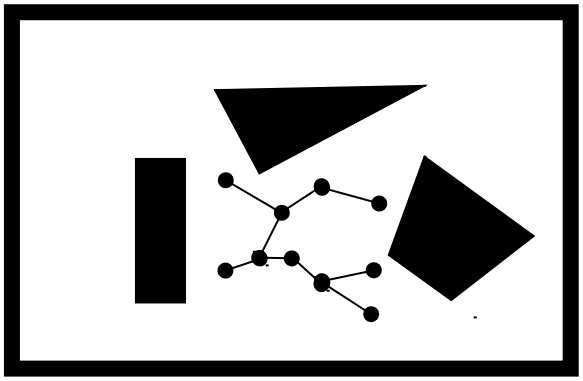
EXPANDIR_RRT(T,q_{al})

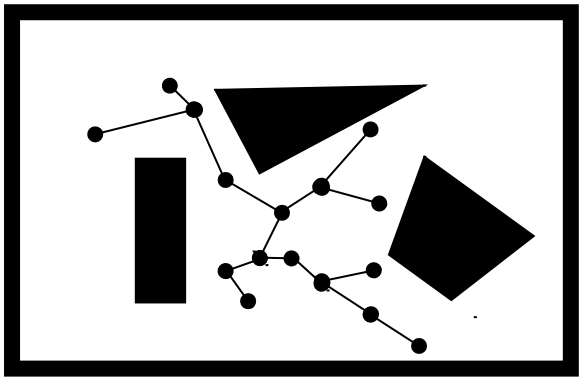
```
1    q_pr ← VIZINHO_MAIS_PRÓXIMO((T,q_al)
2    q_nova ← PLAN(q_pr,q_al)
3    T.adiciona_nó(q_nova)
4    T.adiciona_aresta(q_pr,q_nova)
5    se (q_nova = q_al) então
6         retornar Status conexão(q_pr, q_al) = Alcançada
7    senão
8         retornar Status conexão(q_pr, q_al) = Avanço
9    fim se
10   retornar Status conexão(q_nova, q_k) = Bloqueada
```


q_{ini}
q_{pr}
q_{nova}
q_{al}









Observações:

- Para polarizar a busca em regiões pouco exploradas do espaço de configuração, a probabilidade de escolher um nó da árvore para expansão, na linha 3 do algoritmo CONSTRUIR_RRT(q_{ini}), pode ser atribuída proporcional à área da sua região de Voronoi correspondente. Isto acelera a exploração, garantindo cobertura uniforme de $\mathbf{C}_L$.
  (Ver: Rapidly-exploring random trees: Progress and prospects. S. M. LaValle and J. J. Kuffner. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational* Robotics: New Directions, pages 293--308. A K Peters, Wellesley, MA, 2001).
- Assim, a configuração aleatória q_{al} é selecionada na vizinhança deste vértice.
- O vizinho mais próximo a q_{al} (na linha 1 de EXPANDIR_RRT(T,q_{al})), é escolhido de acordo com a métrica adotada para o espaço de configuração **C**.
- Na etapa de planejamento local PLAN(q_{pr},q_{al}), (linha 2 de EXPANDIR_RRT(T,q_{al})), que tenta conectar q_{pr} a q_{al}, temos três situações possíveis:
  - $q_{nova} = q_{al}$. A configuração selecionada foi Alcançada e a mesma é adicionada à árvore. (Linha 6 de EXPANDIR_RRT(T,q_{al})).
  - $q_{nova} \neq q_{al}$. O planejador avançou até uma configuração q_{nova}, após a qual há um obstáculo. Neste caso, esta última configuração q_{nova} é adicionada à árvore. (Linha 8 de EXPANDIR_RRT(T,q_{al})).
  - O planejador não consegue avançar em direção a q_{al} a partir de q_{pr} (no primeiro deslocamento incremental já colide com um obstáculo – conexão Bloqueada). Neste caso, nenhum novo nó é adicionado à árvore. (Linha 10 de EXPANDIR_RRT(T,q_{al})).

Planejamento de Caminhos baseado em RRT:

- O algoritmo RRT pode ser usado para planejamento de caminhos entre q_{ini} e q_{fin}, uma vez que a medida que a árvore cresce, uma cobertura uniforme de $\mathbf{C_L}$ é alcançada.
- A probabilidade de alcançar uma vizinhança de q_{fin} tende a 1 quando o tempo tende a infinito. Assim, q_{fin} deverá ser alcançada, desde que se tenha suficiente tempo de convergência. Esta abordagem converge devagar.
- RRT-GoalBias é uma variante do método RRT que converge mais rapidamente para o alvo.
- No método RRT-GoalBias, na seleção da configuração aleatória, (linha 3 do algoritmo CONSTRUIR_RRT($q_{ini:}$)), em lugar de escolher uma configuração totalmente aleatória, faz-se esta escolha com uma probabilidade alta ou escolhe-se a configuração final q_{fin} com probabilidade baixa (por exemplo: 0,05). Isto acelera bastante a convergência ao alvo.

Planejador RRT Bidirecional:

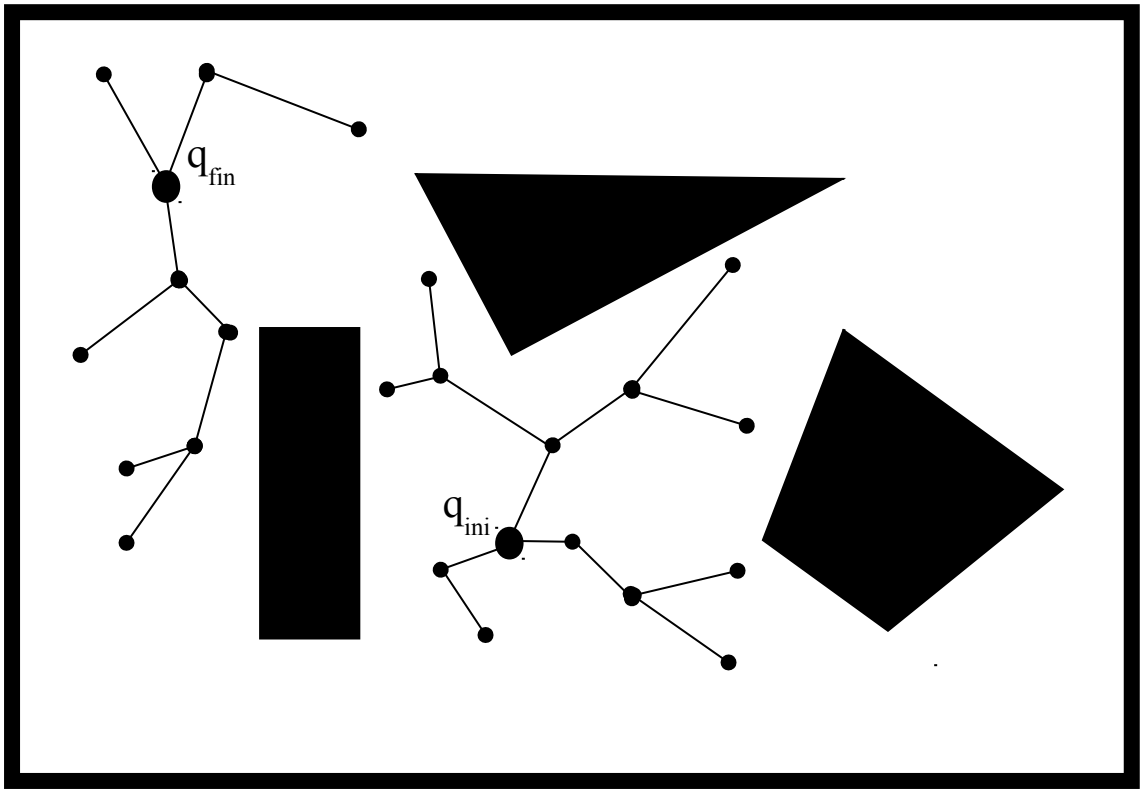
- Um melhor desempenho do método RRT pode ser obtido realizando uma busca bidirecional.
- Duas árvores são construídas concorrentemente: T_{ini} partindo de q_{ini} e T_{fin} partindo de q_{fin}.
- A solução é obtida quando as duas árvores se encontram.
- A construção de T_{ini} e T_{fin} deve ser polarizada de modo a assegurar que as duas arvores se encontrem bem antes de que uma cobertura uniforme de todo $\mathbf{C_L}$ seja alcançada.

Algoritmo RRT Bidirecional

RRT_Bidirecional(q_{ini},q_{fin})

```
1   T.ini(q_ini), T.fin(q_fin)
2   para i = 1 até N_max faça
3       q_al ← configuração aleatória em C
4       se (não(EXPANDIR_RRT(T_ini,q_al) = Bloqueado)) então
5           se (EXPANDIR_RRT(T_fin,q_nova) = Alcançada)) então
6               retornar Caminho(T_ini,T_fin)
7           fim se
8       fim se
9       Alternar(T_ini,T_fin)
10  fim para
11  retornar Falha
```

- A cada iteração, a árvore corrente é expandida e tenta-se conectá-la ao nó mais próximo da outra árvore.
- A seguir, invertem-se os papéis, alternado as duas árvores.



Espuma Probabilística:

O Método de Planejamento baseado em Espuma Probabilística é um algoritmo de planejamento para buscas em espaços de alta dimensionalidade baseado na construção de uma árvore de exploração do espaço livre constituída por estruturas convexas e superpostas denominadas bolhas que garantem um espaço de manobrabilidade mínimo. A espuma se propaga de acordo com uma busca em largura, de forma semelhante aos campos de potenciais baseados em frente de onda, realizando uma cobertura aproximada do espaço livre.

Definições:

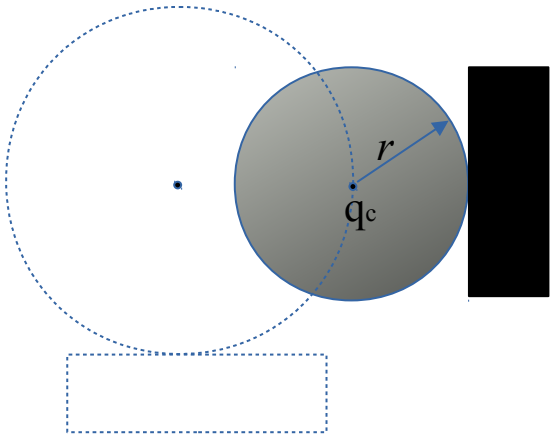
- Uma bolha de raio r centrada na configuração $q_C \in \mathbf{C_L}$ é definida como uma n-bola em $\mathbf{C_L}$:

$$\mathbf{b} = \mathbf{b}(q_C, r) = \{q \in \mathbf{C_L} / \, d(q, q_C) < r\}$$

onde n é a dimensão de **C**, $d(q,q_C)$ é a métrica definida em **C** e r é a mínima distância do centro da bolha q_C ao obstáculo mais próximo em **CB**, tal que:

$$r = min_{q' \in \mathbf{CB}} \, d(q, q_C)$$

Assim, a bolha se expande a partir de q_C até alcançar o obstáculo mais próximo em **CB**.



- Uma Espuma Probabilística $\mathbf{E}$ é um subconjunto de $\mathbf{C_L}$ definido como uma coleção de bolhas $\mathbf{b_i}$, tais que:

i. $\mathbf{E} = \cup \mathbf{b_i}$

ii. $\mathbf{E} \subset \boldsymbol{C_L}$

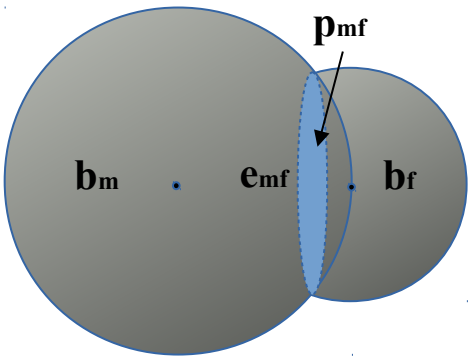
iii. $\mathbf{E}$ é construída incrementalmente a partir de uma bolha inicial $\mathbf{b_{fin}}$ (bolha centrada na configuração inicial $\boldsymbol{q_{fin}}$) propagando uma árvore de busca cujos nós são bolhas filhas criadas aleatoriamente com centro na superfície de bolhas criadas previamente (bolhas mães).

Desta forma, caso não haja restrição no raio mínimo das bolhas, a espuma probabilística pode propagar-se incrementalmente, gerando coberturas aproximadas cada vez melhores do subconjunto do espaço livre $\mathbf{C_L}$ conectado a q_{fin}. Assim, quanto mais bolhas são criadas, menor a região não coberta $\mathbf{C_L \backslash E}$. O objetivo é propagar a espuma até alcançar uma bolha $\mathbf{b_{ini}}$ que contenha a configuração inicial $\boldsymbol{q_{ini}}$, ou reportar falha caso $\boldsymbol{q_{ini}}$ não seja encontrada e não seja mais possível propagar a espuma.

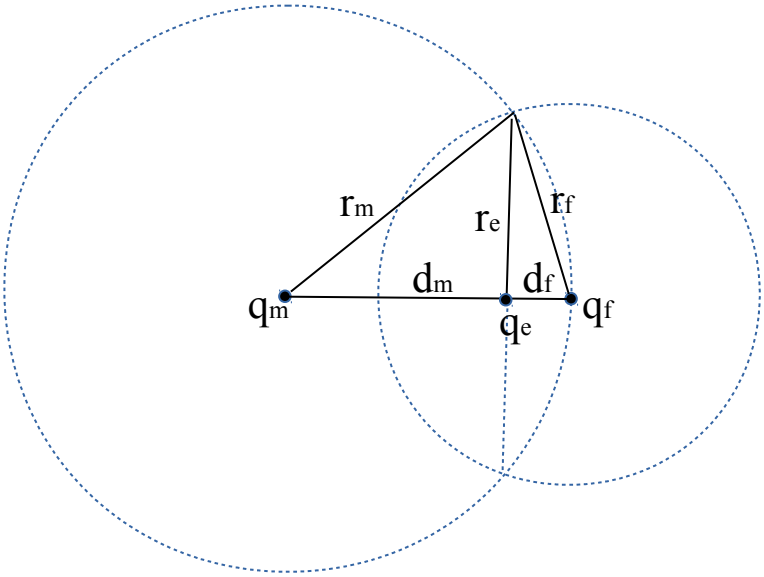
- Duas bolhas $\mathbf{b_m}$ (bolha mãe) e $\mathbf{b_f}$ (bolha filha), possuem uma região de superposição denominada escotilha $\mathbf{e_{mf}} = \mathbf{b_m} \cap \mathbf{b_f} \neq \varnothing$.

Uma vez que bolhas são conjuntos convexos, sempre existe um caminho livre de qualquer configuração em $\mathbf{b_m}$ para a escotilha $\mathbf{e_{mf}}$, bem como existe um outro caminho livre de qualquer configuração na escotilha para qualquer configuração em $\mathbf{b_f}$.

- Região de Passagem $\mathbf{p_{mf}}$ da escotilha $\mathbf{e_{mf}}$ entre $\mathbf{b_m}$ e $\mathbf{b_f}$ é a (n-1)-bola delimitada pela (n-2)-esfera resultante da intersecção das (n-1)-esferas que delimitam as n-bolas $\mathbf{b_m}$ e $\mathbf{b_f}$.



- O raio r_e da escotilha e o centro q_e da escotilha na região de passagem $\mathbf{p_{mf}}$ podem ser obtidos por simples relações geométricas:



$r_e = (r_f/r_m)[(r_m)^2 + (r_f/2)^2]^{1/2}$

O centro da escotilha q_e na região de passagem encontra-se a uma distância d_m de q_m (centro de $\mathbf{b_m}$) ou a uma distância d_f de q_f (centro de $\mathbf{b_f}$) ao longo do segmento de reta entre estes dois centros, onde:

$d_m + d_f = r_m$

$d_m = [(r_m)^2 - (r_e)^2]^{1/2}$

$d_f = [(r_f)^2 - (r_e)^2]^{1/2}$

- Rosário: define-se rosário ρ como uma sequência de bolhas iniciada em $\mathbf{b_{fin}}$ e suas descendentes ao longo de um galho da árvore de busca. O objetivo do algoritmo é encontrar um rosário que alcance a bolha inicial $\mathbf{b_{ini}}$: $\rho = \{\mathbf{b_{fin}}, \mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_{ini}}\}$

Propagação da Espuma Probabilística:

A espuma probabilística se propaga pelo espaço livre construindo coberturas aproximadas da componente de $\mathbf{C_L}$ alcançável a partir de q_{fin}. Esta cobertura é similar à cobertura criada pelos métodos de decomposição aproximada em células convexas.

A propagação da espuma se inicia a partir da bolha mãe $\mathbf{b_{fin}}$ (bolha centrada em q_{fin}), a qual é a raiz da árvore de busca. A partir da bolha final, sucessivas gerações de bolhas filhas são geradas aleatoriamente sobre a superfície livre das bolhas mães da geração prévia e adicionadas ao próximo nível da árvore de busca.

Uma bolha filha é criada selecionando aleatoriamente uma configuração sobre a superfície livre da bolha mãe. A bolha filha é expandida com centro nesta configuração aleatória até ser limitada pelo C-Obstáculo mais próximo. Cada nova bolha filha é incluída na árvore com um ponteiro para a sua bolha mãe correspondente.

A nova bolha filha é gerada se o seu centro criado aleatoriamente sobre a superfície da bolha mãe não está dentro de outra bolha já criada. Além disso, a nova bolha só é criada se o seu raio de expansão é maior que um raio mínimo r_{min}, parâmetro a ser definido pelo projetista. Este raio mínimo é necessário para evitar que seja criado um número excessivo de bolhas filhas, ao mesmo tempo que garante um espaço de manobra mínimo, livre de colisão, no interior da bolha.

Desta forma, a propagação da espuma executa uma busca em largura (breadth-first search) sobre o espaço livre. Ou seja, dado o conjunto de bolhas e uma mesma geração (que estão na mesma altura da árvore de busca), todas as suas possíveis bolhas filhas são expandidas antes de qualquer bolha a maior altura na árvore.

Assim, a Espuma Probabilística se propaga geração a geração, de maneira similar às frentes de onda característica de vários métodos de planejamento baseados em campo de potencial.
Em caso de sucesso, a árvore de busca se propaga a partir de $\mathbf{b_{fin}}$ até expandir uma bolha $\mathbf{b_{ini}}$ que contenha a configuração inicial q_{ini}. Neste caso, a saída do algoritmo é o rosário $\rho = \{\mathbf{b_{fin}}, \mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_{ini}}\}$, obtido descendo a árvore de busca, seguindo os ponteiros que apontam das bolhas filhas para as suas bolhas mães, a partir de $\mathbf{b_{ini}}$: $\mathbf{b_{ini}} \rightarrow \ldots \rightarrow \mathbf{b_2} \rightarrow \mathbf{b_1} \rightarrow \mathbf{b_{fin}}$.

Cada bolha mãe gera até um número máximo de N_f de bolhas filhas. Este parâmetro limita o número de bolhas filhas de maneira a limitar o número de galhos da árvore de busca, limitando, em certo grau a sua complexidade e os recursos computacionais (espaço de armazenamento e capacidade de processamento) necessários para implementá-la. O número máximo de bolhas filhas pode ser escolhido a partir de:

$$N_f = K(\lfloor (r_m/r_{min}) \rfloor)^{n-1}$$

onde n é a dimensão do espaço de configuração, a função $\lfloor (.) \rfloor$ retorna o inteiro mais próximo menor do que o argumento, r_m é o raio da bolha mãe, r_{min} é o parâmetro que define o raio da menor bolha permitida e K é um parâmetro a ser definido pelo projetista. Conceitualmente, K corresponde ao maior número de bolhas filhas que se permite criar sobre a superfície da menor bolha admissível (a bolha de raio r_{min}). A escolha de N_f de acordo com a expressão acima, busca manter constante a densidade de bolhas filhas criadas sobre a superfície da bolha mãe, independente do seu raio r_m.

Os parâmetros r_{min} e K influenciam na convergência do algoritmo, assim como nos recursos computacionais necessários para a sua execução, e devem ser escolhidos cuidadosamente pelo projetista.

Sorteio de configurações aleatórias sobre a bolha mãe:

Para sortear configurações aleatórias (centros de possíveis bolhas filhas) sobre a superfície da bolha mãe, deve-se ter cuidado para que qualquer ponto da superfície tenha igual probabilidade de ser sorteado. Para garantir esta condição, uma forma de fazer isto é representar um ponto na superfície da bolha em coordenadas polares. A bolha mãe no espaço de configuração **C** de dimensão n é uma n-bola e a sua superfície é uma (n-1)-esfera. Em coordenadas cartesianas, uma configuração q_c sobre a superfície da bolha mãe de raio r_m é representada pela expressão abaixo, onde os ângulos ϕ_1, ϕ_2, …, ϕ_{n-1} são as coordenadas polares, de tal forma que $0 \leq \phi_1 \leq 2\pi$, e $0 \leq \phi_i \leq \pi$, para $i \neq 1$:

$$q_c = r_m \begin{bmatrix} cos(\phi_1) \\ sen(\phi_1)cos(\phi_2) \\ sen(\phi_1)sen(\phi_2)cos(\phi_3) \\ . \\ . \\ . \\ sen(\phi_1)...sen(\phi_{n-2})cos(\phi_{n-1}) \\ sen(\phi_1)...sen(\phi_{n-2})sen(\phi_{n-1}) \end{bmatrix}$$

Assim, para sortear um ponto q_c sobre a superfície da bolha, de forma que cada ponto da mesma tenha igual probabilidade, basta sortear cada um dos ângulos $(\phi_1, \phi_2, \ldots, \phi_{n-1})$ de acordo com uma função de densidade uniforme, dentro do seu domínio e, a partir deles, obter a as coordenadas cartesianas de acordo com a expressão acima.

Algoritmo: Espuma Probabilística

Entrada: q_{ini} (configuração inicial), q_{fin} (configuração final), r_{min} (raio de bolha mínimo), K (número máximo de filhas da bolha mínima), n (dimensão de **C**), **CB** (conjunto de C-Obstáculos).
Saída: ρ =(rosário de $\mathbf{b_{fin}}$ a $\mathbf{b_{ini}}$},

início

E = ∅; /* E= árvore de bolhas da espuma, inicialmente vazia */
L = ∅; /* L= lista de bolhas, inicialmente vazia */
$\mathbf{b_{fin}} \leftarrow$ bolha inicial, expandida a partir de q_{fin} ;
instalar $\mathbf{b_{fin}}$ em **E**; /* Raiz da árvore */
inserir $\mathbf{b_{fin}}$ no final de **L**;
enquanto (L ≠ ∅)
$\mathbf{b_m} \leftarrow$ primeiro(**L**); /* Bolha Mãe corrente */
$r_m \leftarrow$ raio($\mathbf{b_m}$); /* Raio da Bolha Mãe */
$\mathbf{s_m} \leftarrow$ superfície($\mathbf{b_m}$); /* superfície de $\mathbf{b_m}$ */
$N_f = K(\lfloor (r_m/r_{min}) \rfloor)^{n-1}$ /* Número máximo de filhas de $\mathbf{b_m}$ */
repita N_f vezes
$q_C \leftarrow$ configuração amostrada aleatoriamente em $\mathbf{s_m}$;
se ($q_C \not\subset$ int(**E**)) **então faça:**
$\mathbf{b_f} \leftarrow$ bolha filha, expandida a partir de q_C ;
$r_f \leftarrow$ raio($\mathbf{b_f}$); /* Raio da Bolha Filha */
se ($r_f \geq r_{min}$) **então faça:**
instalar $\mathbf{b_f}$ em **E** com ponteiro para $\mathbf{b_m}$;
inserir $\mathbf{b_f}$ no final de **L**;
se ($d(q_f,q_{ini}) \leq r_f$) **então faça:**
$\mathbf{b_{ini}} \leftarrow \mathbf{b_f}$
retorna Sucesso
/* Saída: rosário livre seguindo os ponteiros de $\mathbf{b_{ini}}$ a $\mathbf{b_{fin}}$ */
fim repita
fim enquanto;
retorna Falha /* Não foi possível encontrar o rosário */

fim

Critérios para a escolha de K e r_{min}:

O método de planejamento baseado em Espuma Probabilística caracteriza-se por precisar apenas dois parâmetros, K e r_{min}, para determinar a busca.

- r_{min}:

O parâmetro r_{min} é o menor raio de bolha admissível na construção da espuma probabilística. Quanto menor r_{min}, mais bolhas poderão ser geradas e espaços mais estreitos entre os obstáculos poderão ser explorados. Por outro lado, mais bolhas geradas significa árvore maior, consequentemente uma maior necessidade de recursos computacionais (memória e processamento).

Um critério para escolher adequadamente r_{min} é utilizar, caso disponível, informação sobre a largura da passagem mais estreita, L_{min}, entre os obstáculos presentes em $\mathbf{C}_L$. Para que o rosário tenha possibilidade de ultrapassar a passagem mais estreita, r_{min} deve ser menor do que a metade desta largura. Assim, escolher r_{min} um pouco menor do que $L_{min}/2$ é uma escolha razoável.

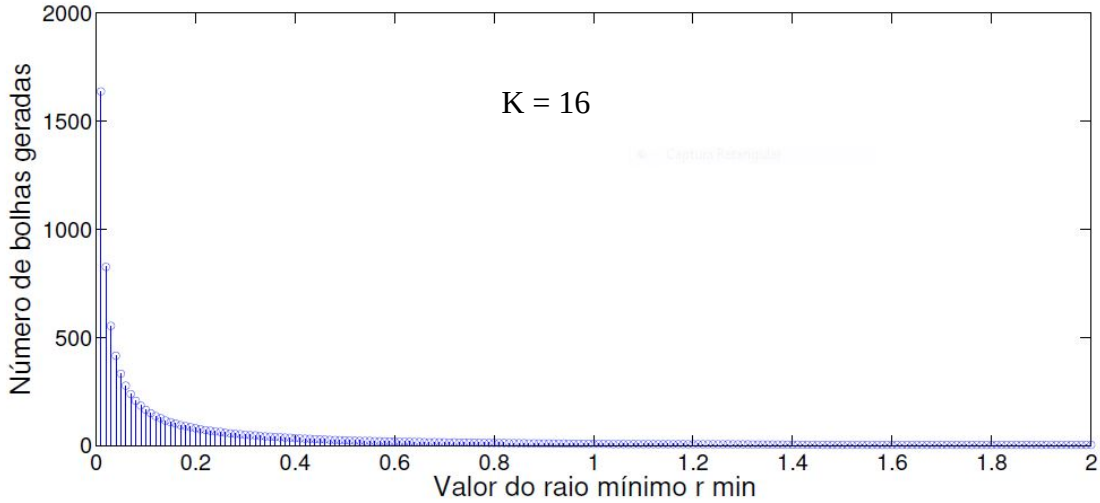
- K:

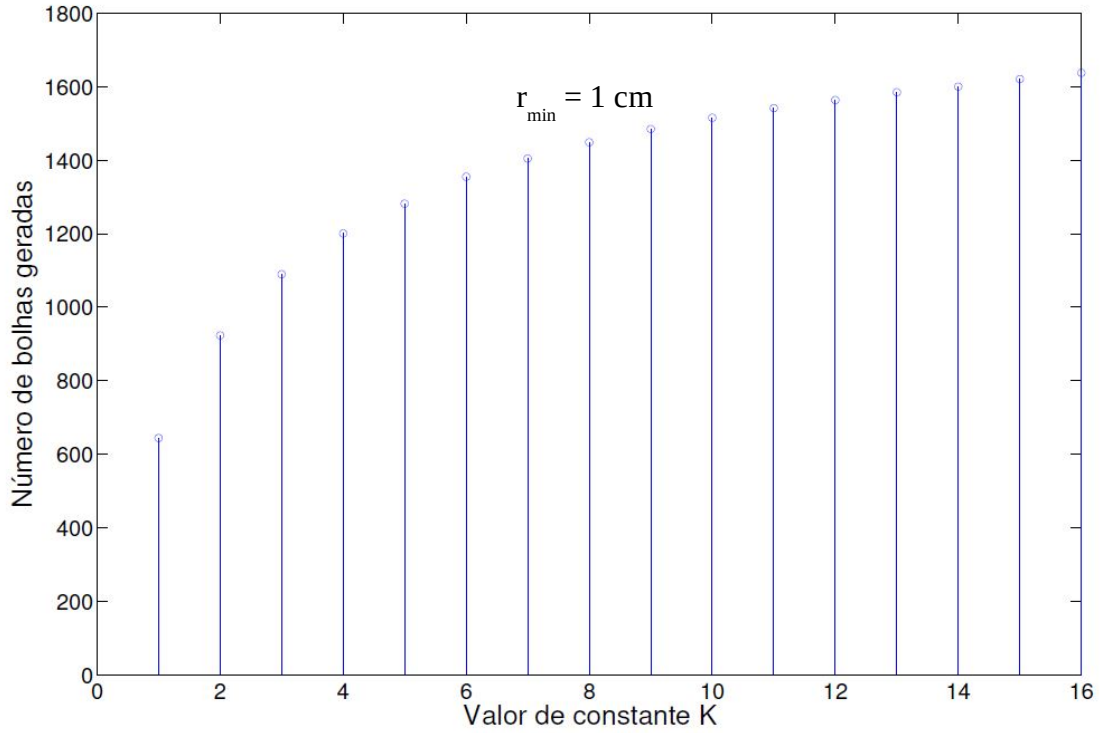
Conceitualmente, o parâmetro K é igual ao número máximo de bolhas filhas permitidas sobre a superfície da bolha de raio mínimo. Para uma bolha mãe de raio r_m maior que r_{min}, o número máximo de bolhas filhas permitido, N_f, aumenta proporcionalmente à (n-1)-ésima potência de (r_m/r_{min}), multiplicada por K, onde n é a dimensão de $\mathbf{C}$. A ideia por trás deste critério é manter a densidade de bolhas filhas na superfície da bolha mãe aproximadamente constante. Assim, bolhas maiores geram mais filhas que bolhas menores. Desta forma, para um mesmo raio de bolha mãe, quanto maior K, mais filhas são geradas na sua superfície (mais ramificações na árvore). Desta forma um K maior implica em uma melhor exploração das vizinhanças da bolha mãe, mas, por outro lado, implica também na necessidade de mais recursos computacionais na execução do algoritmo.

Resultados experimentais em alguns estudos de caso sugerem que ao aumentar K, o número de bolhas filhas aumenta até um certo limite,

tendendo a saturar. Ou seja, após um certo valor, não adianta aumentar mais K, pois o número de bolhas geradas não aumenta de forma correspondente.

As figuras abaixo mostram resultados experimentais que ilustram a relação entre os parâmetros r_{min} e K e o número de bolhas geradas, quando o algoritmo foi executado em um espaço plano, desprovido de obstáculos e limitado por um quadrado de 10 metros de lado.

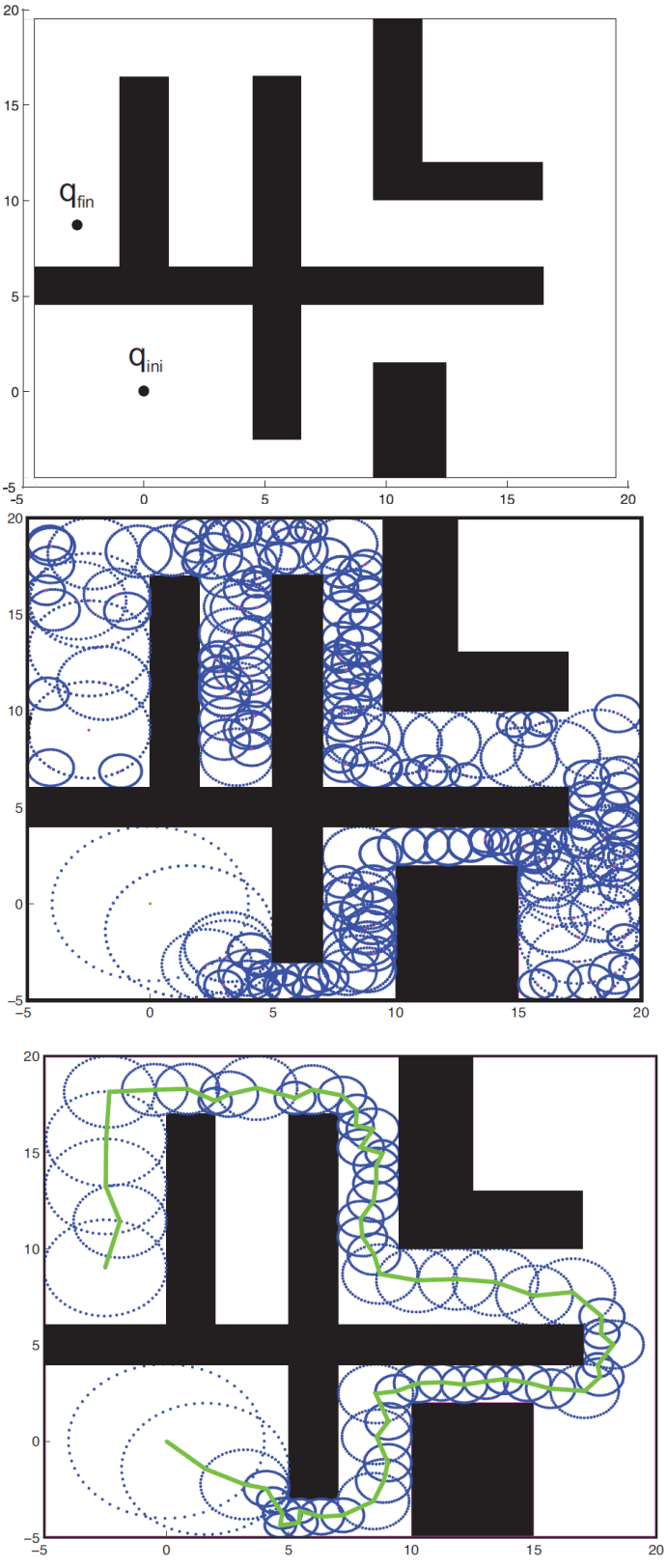




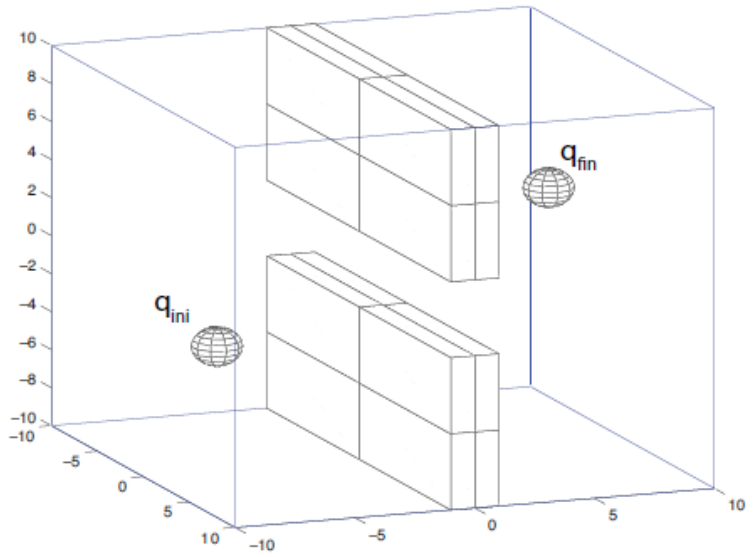
Teoricamente, se r_{min} tendesse a zero e K tendesse a infinito, a espuma poderia cobrir exatamente o subconjunto de $\mathbf{C_L}$ conexo a q_{fin} e o método se tornaria completo. Obviamente, neste caso, os recursos computacionais necessários para executar o algoritmo tenderiam a infinito, inviabilizando a

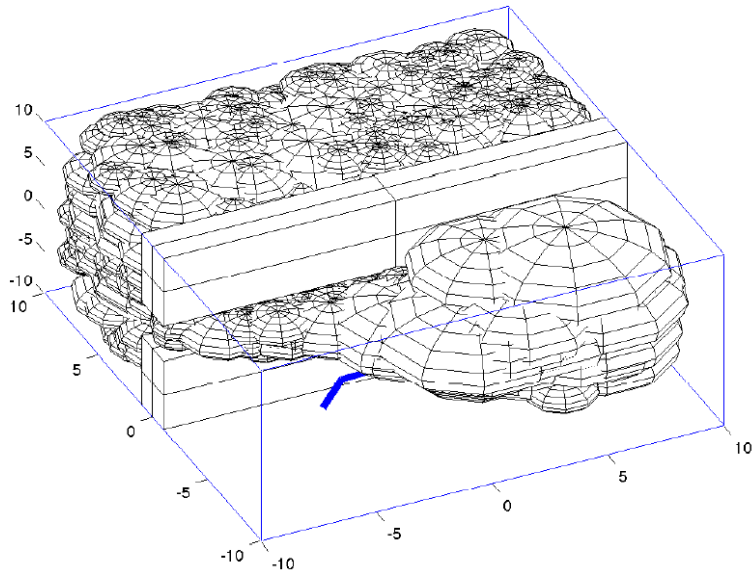
sua aplicação. Assim, na escolha dos parâmetros r_{min} e K é necessário estabelecer um compromisso entre os recursos computacionais disponíveis e o grau de resolução em que a busca é feita.

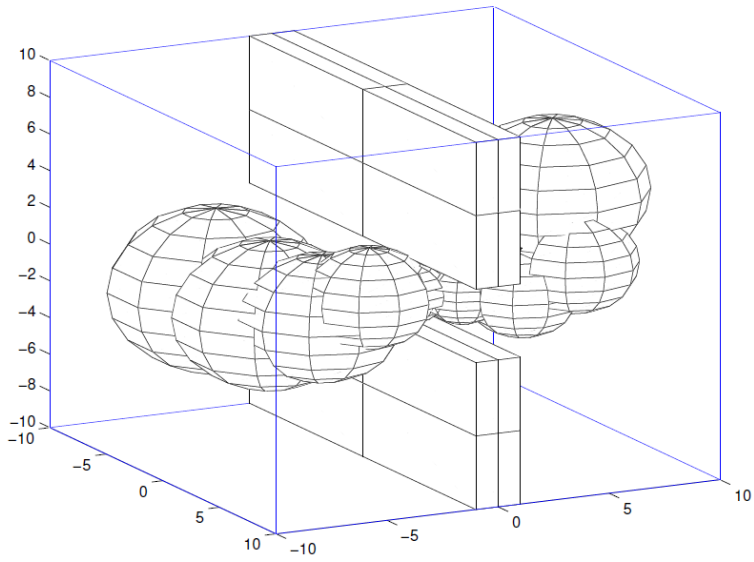
Primeiro exemplo: mapa 2D, espuma probabilística e rosário resultante.



Segundo exemplo: mapa 3D, espuma probabilística e rosário resultante.







Observações:

- O método de planejamento baseado em espuma probabilística baseia-se em árvore de busca, similar aos métodos baseados em RRT.

- A busca em largura (gerações sucessivas de bolhas) é similar à propagação de frentes de ondas característica de alguns métodos baseados em campos de potenciais artificiais.

- O método implementa uma cobertura aproximada de $\mathbf{C}_L$, de maneira similar aos métodos baseados em decomposição aproximada em células convexas. Difere destes em dois aspectos: as bolhas permitem superposição dos seus interiores (as escotilhas) e a adjacência entre as mesmas respeita a conectividade da árvore de busca.

- O método requer a sintonia de apenas dois parâmetros, r_{min} e K, claramente relacionados a aspectos qualitativos da busca.

- Assim como em métodos baseados em RRT, pode-se acelerar a busca construindo duas árvores de busca, uma com raiz em q_{fin} e outra com raiz em q_{ini}. A busca tem sucesso quando se encontra um centro de bolha de uma árvore contido em uma bolha da outra árvore. A saída do algoritmo é um rosário composto dos dois rosários que partem das bolhas na intersecção das duas árvores até as raízes das mesmas.

- A grande desvantagem do algoritmo é o processo de expansão das bolhas, que exige o cálculo da distância ao C-obstáculo mais próximo. Isto pode não ser trivial em espaços de configuração de alta dimensão, ou onde o cálculo da métrica é complexo (por exemplo, envolvendo dimensões de grandezas diferentes: ângulos e comprimentos, e/ou topologia diferente do espaço $\mathbb{R}^n$). Para contornar este problema, pode-se aproximar **CB** como uma união de n-retângulos, onde o cálculo da distância aos obstáculos se torna mais simples.