PLANEJAMENTO DE CAMINHOS

MÉTODOS BASEADOS CAMPOS DE POTENCIAL

Princípio:

- Considerar o robô, representado como um ponto no espaço de configuração, como uma partícula sob a influência de um campo de potencial artificial **U**, cujas variações locais refletem a estrutura do espaço livre.
- Tipicamente, a função de potencial é uma soma de potenciais repulsivos, (geralmente com influência local), que afastam o robô dos obstáculos e um potencial atrativo que empurra o robô em direção ao alvo.
- O planejamento é realizado iterativamente. A cada iteração, a força artificial $F(q) = -\nabla \mathbf{U}(q)$ induzida por **U** na configuração q define a direção de movimento mais promissora e o incremento de posição nesta direção.

Características:

- Desenvolvido originalmente para contorno de obstáculos, aplicável quando não se dispõe de um modelo *a priori* dos mesmos, mas são percebidos *on-line*. $\Rightarrow$ Método "Local".
- Ênfase em eficiência em tempo real em detrimento da garantia de alcançar o alvo. $\Rightarrow$ Método incompleto, (pode falhar na busca de um caminho, mesmo existindo um), mas de implementação simples; geralmente rápido, eficiente e confiável para a maioria das aplicações.
- Método de otimização baseado em gradiente ("Descida da Ladeira"). $\Rightarrow$ Sujeito a problemas de mínimos locais da função de potencial, (o seu maior problema). Abordagens de solução:
    1. Definir a função de potencial de modo a ter um ou uns poucos mínimos locais. $\Rightarrow$ tornar o método "Global".
    2. Incluir técnicas para escapar dos mínimos locais.

Campo de Potencial – Caso Translacional:

- $\mathbf{W} = \mathbb{R}^n$ $\quad C = \mathbb{R}^n$ $\quad (n = 2 \text{ ou } 3)$.

- Função de Potencial: $\quad \mathbf{U}(q) = \mathbf{U}_{atr}(q) + \mathbf{U}_{rep}(q)$

  - $\mathbf{U}_{atr}(q)$ = Função de potencial atrativo, associada ao alvo e independente da região de C-Obstáculos.
  - $\mathbf{U}_{rep}(q)$ = Função de potencial repulsivo, associada à região de C-Obstáculos e independente do alvo.

- Força artificial: $\quad F(q) = -\nabla\mathbf{U}(q) = F_{atr}(q) + F_{rep}(q)$

  - $F_{atr}(q) = -\nabla\mathbf{U}_{atr}(q)$ = Força atrativa (empurra em direção ao alvo).
  - $F_{atr}(q) = -\nabla\mathbf{U}_{rep}(q)$ = Força repulsiva (afasta o robô dos obstáculos).

- Gradiente da função de potencial:

  - $\nabla\mathbf{U}(q) = \partial\mathbf{U}(q)/\partial q \Rightarrow$

    - $C = \mathbb{R}^2 \Rightarrow \quad q = [x \quad y]^T$

      $\Rightarrow \quad \nabla\mathbf{U}(q) = \quad [\partial\mathbf{U}/\partial x \quad \partial\mathbf{U}/\partial y]^T.$

    - $C = \mathbb{R}^3 \Rightarrow \quad q = [x \quad y \quad z]^T$

      $\Rightarrow \quad \nabla\mathbf{U}(q) = \quad [\partial\mathbf{U}/\partial x \quad \partial\mathbf{U}/\partial y \quad \partial\mathbf{U}/\partial z]^T.$

Potencial Atrativo: (atrai o robô na direção do alvo):

- Potencial Paraboloide: $\mathbf{U}_{atr}(q) = \xi.[\rho_{fin}(q)]^2/2$

onde, $\rho_{fin}(q) = ||q - q_{fin}||$ = distância euclidiana ao alvo.
ξ = fator de ganho positivo.

Obs.:

- $\mathbf{U}_{atr}(q) \geq 0$, com mínimo em q_{fin}.
- $\rho_{fin}(q)$ diferenciável em $\forall q \in C$.

Força atrativa: $F_{atr}(q) = -\nabla\mathbf{U}_{atr}(q) = - \xi.[\rho_{fin}(q)].\nabla\rho_{fin}(q)$
$= - \xi.(q - q_{fin})$.

Obs.:

- $F_{atr}(q)$ converge linearmente para zero ao se aproximar da configuração alvo. $F_{atr}(q)$ tende a infinito para $\rho_{fin} \rightarrow \infty$.
- Estabilidade assintótica pode ser alcançada adicionando uma força dissipativa proporcional à velocidade dq/dt.

- Potencial Cônico: $\mathbf{U}_{atr}(q) = \xi.\rho_{fin}(q)$

Obs.:

- $\mathbf{U}_{atr}(q) \geq 0$, com mínimo em q_{fin}.
- $\rho_{fin}(q)$ diferenciável em $\forall q \in C$, exceto em $q = q_{fin}$.

Força atrativa: $F_{atr}(q) = - \xi.\nabla\rho_{fin}(q) = - \xi.(q - q_{fin})/||q - q_{fin}||$

Obs.:

- $||F_{atr}(q)||$ = constante em $\forall q \in C$, exceto em $q = q_{fin}$.
- $F_{atr}(q)$ não possui características estabilizantes, visto que não tende a zero quando $q \rightarrow q_{fin}$.
- Alternativa: mesclar perfil cônico (longe de q_{fin}) com perfil parabólico (nas vizinhanças de q_{fin}).

Potencial repulsivo: (barreira de potencial em torno de CB):

$\Rightarrow$ É desejável que $\mathbf{U}_{rep}(q)$ não afete o movimento do robô quando este estiver suficientemente longe dos obstáculos. Exemplo:

$$\mathbf{U}_{rep}(q) = \begin{cases} (\eta/2).[(1/\rho(q)) - (1/\rho_0)]^2 & \text{se } \rho(q) \leq \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

onde, η = fator de ganho positivo.
$\rho(q) = \min_{q' \in CB}\|q - q'\|$ = distância de q a CB.
ρ_0 = Distância de influência (parâmetro positivo).

Obs.:

- $\mathbf{U}_{rep}(q)$ é positivo para $\rho(q) < \rho_0$, tendendo a infinito quando o robô se aproxima do obstáculo.
- $\mathbf{U}_{rep}(q)$ é nulo quando o robô se afasta de CB mais longe do que ρ_0.

- Se CB = região convexa com limites diferenciáveis por partes, então $\rho(q)$ é diferenciável para $\forall q \in C_L$. A força repulsiva é:

$$\mathbf{F}_{rep}(q) = -\nabla\mathbf{U}_{rep}(q)$$

$$\mathbf{F}_{rep}(q) = \begin{cases} \eta.[(1/\rho(q)) - (1/\rho_0)].(1/\rho(q))^2.\nabla\rho(q) & \text{se } \rho(q) \leq \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

- Seja q_C tal que $\|q - q_C\| = \rho(q)$. O gradiente $\nabla\rho(q)$ é o vetor unitário apontando de q_C para q.
- Se CB é uma região não convexa, então $\rho(q)$ é diferenciável para $\forall q \in C_L$, exceto para os pontos para os quais existe mais de um q_C tal que $\|q-q_C\| = \rho(q)$ (pontos do diagrama de Voronoi). Nos dois lados do diagrama, $\mathbf{F}_{rep}(q)$ assume sentidos opostos. $\Rightarrow$ O robô ficará oscilando entre estes dois lados.

Solução:

- Decompor CB em componentes convexas CB_k, k = 1,...,r, com uma função de potencial associada $\mathbf{U}_k(q)$. O potencial repulsivo total é a soma dos potenciais gerados por cada componente convexa:

$$\mathbf{U}_{rep}(q) = \sum \mathbf{U}_k(q)$$

$$\mathbf{U}_k(q) = \begin{cases} (\eta/2).[(1/\rho_k(q)) - (1/\rho_0)]^2 & \text{se } \rho_k(q) \leq \rho_0 \\ 0 & \text{se } \rho_k(q) > \rho_0 \end{cases}$$

onde $\rho_k(q)$ é a distância de q a CB_k. A força repulsiva correspondente a $\mathbf{U}_{rep}(q)$ é dada por:

$$F_{rep}(q) = \sum F_k(q)$$

$$F_k(q) = -\nabla \mathbf{U}_k(q)$$

Obs.:

- CB_k pode ser construída decompondo **A** e **B** em componentes convexas e computar o C-Obstáculo correspondente a cada par de componentes convexas de **A** e **B**.
- Decompor CB em componentes pequenas pode resultar numa força repulsiva que é muito maior do que a força do C-obstáculo maior equivalente. $\Rightarrow$ Ponderar cada potencial $\mathbf{U}_k(q)$ proporcionalmente ao tamanho da região CB_k.
- Diferentes ganhos η e distâncias de influência ρ_0 podem ser utilizados para diferentes componentes de CB. Exemplo: se q_{fin} está próxima de CB_k, é razoável diminuir ρ_0 a menos do que a distância de q_{fin} a CB_k, de modo a evitar que o potencial repulsivo $\mathbf{U}_k(q_{fin})$ impeça que o robô atinja q_{fin}.

Campo de Potencial – Caso Geral:

- $\mathbf{W} = \mathbb{R}^n$ $\quad C = \mathbb{R}^n \times SO(n)$ $\quad (n = 2 \text{ ou } 3)$.

Dada uma métrica d em C,

Potencial atrativo e Força atrativa:

$$\mathbf{U}_{atr}(q) = \xi.[\rho_{fin}(q)]^2/2$$
$$F_{atr}(q) = -\nabla\mathbf{U}_{atr}(q) = -\xi.[\rho_{fin}(q)].\nabla\rho_{fin}(q)$$

onde $\rho_{fin}(q) = d(q,q_{fin})$.

Potencial repulsivo e força repulsiva:

$$\mathbf{U}_{rep}(q) = \begin{cases} (\eta/2).[(1/\rho(q)) - (1/\rho_0)]^2 & \text{se } \rho(q) \le \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

$$\mathbf{F}_{rep}(q) = \begin{cases} \eta.[(1/\rho(q)) - (1/\rho_0)].(1/\rho(q))^2.\nabla\rho(q) & \text{se } \rho(q) \le \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

onde $\rho(q) = \min_{q' \in CB}\|q - q'\|$.

Problema:

- Para uma dada métrica d geralmente não há método eficiente e simples para computar $\rho(q)$ e seu gradiente $\nabla\rho(q)$.

Solução:

- Definir os potenciais atrativos e repulsivos em **W** e combinar seus efeitos em vários pontos de **A**.

Potencial Atrativo:

- Considerar N pontos de controle a_i, sujeitos ao campo de potencial atrativo, (com i=1,..., N e N = dimensão de **W**), selecionados em **A**, e que determinam univocamente q de **A**.
- Seja ξ um ganho positivo e P um ponto em **W**. Para cada ponto a_i, definir uma função de potencial atrativo, $V_{atr}^{\ i} : \mathbf{W} \rightarrow \mathbb{R}$:

  $$V_{atr}^{\ i}(P) = (\xi/2).||P - a_i(q_{fin})||^2$$

- Cada potencial $V_{atr}^{\ i}$ induz um campo de forças em **W**:

  $$F_{atr}^{\ i}(q) = -\nabla V_{atr}^{\ i}(P) = -\xi.(P - a_i(q_{fin}))$$

onde $P = a_i(q)$. Somente o ponto de controle $a_i(q)$ é sensível a ação desta força. A força atrativa total sobre o robô é:

$$F_{atr}(q) = \sum F_{atr}^{\ i}(q)$$

Seja $\mathbf{U}_{atr}^{\ i}(q) = V_{atr}^{\ i}(a_i(q)) \Rightarrow F_{atr}^{\ i}(q) = -\nabla U_{atr}^{\ i}(q)$. Assim, o potencial atrativo em C e a força induzida correspondente são:

$$\mathbf{U}_{atr}(q) = \sum V_{atr}^{\ i}(a_i(q)) \qquad F_{atr}(q) = -\nabla \mathbf{U}_{atr}(q)$$

Obs.: Os vários pontos de controle competem para atingir suas respectivas posições alvo, o que pode produzir mínimos locais em regiões confinadas de **W**. Solução: definir um ponto "lider" $a_1(q)$:

$$\mathbf{U}_{atr}(q) = V_{atr}^{\ 1}(a_1(q)) + \varepsilon.\sum_{i>1} V_{atr}^{\ i}(a_i(q))$$

onde ε é um ganho positivo pequeno. a_1 é puxado fortemente pelo alvo. O potencial sobre os outros pontos contribui para alcançar a orientação alvo. $\mathbf{U}_{atr}$ é boa quando q_{fin} esativer longe de CB, mas pode levar a um mínimo local perto de q_{fin} para q_{fin} perto de CB.

Potencial Repulsivo:

- Definir uma função de potencial repulsivo, $V_{rep} : \mathbf{W} \rightarrow \mathbb{R}$:

$$\mathbf{V}_{rep}(P) = \begin{cases} (\eta/2).[(1/\rho(P)) - (1/\rho_0)]^2 & \text{se } \rho(P) \leq \rho_0 \\ 0 & \text{se } \rho(P) > \rho_0 \end{cases}$$

onde η é um ganho positivo, $\rho(P)$ é a distância euclidiana do ponto P a **B** e $\rho_0(P)$ é a distância de influência de **B**.

- Sejam Q pontos de controle a_i, sujeitos ao campo de potencial repulsivo, (com i=1,..., Q), selecionados no limite de **A**.
- Quando **A** está na configuração q, V_{rep} exerce uma força repulsiva sobre o ponto $P = a_i(q)$:

$$\mathbf{F}_{rep}^{i}(q) = \begin{cases} \eta.[(1/\rho(P))-(1/\rho_0)].(1/\rho(P))^2.\nabla\rho(P) & \text{se } \rho(P) \leq \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

- Se **B** é uma região convexa com limite diferenciável por partes, ρ é diferenciável em todo C_L. Se **B** é uma região não convexa, a mesma pode ser decomposta em componentes convexas às quais pode-se associar potenciais repulsivos separados.

- A força resultante sobre o robô é: $\mathbf{F}_{rep}(q) = \sum \mathbf{F}_{rep}^{i}(q)$

Seja $\mathbf{U}_{rep}^{i}(q) = V_{rep}^{i}(a_i(q)) \Rightarrow F_{rep}^{i}(q) = -\nabla U_{rep}^{i}(q)$. Assim, o potencial repulsivo em C e a força induzida correspondente são:

$$\mathbf{U}_{rep}(q) = \sum V_{rep}^{i}(a_i(q)) \qquad F_{atr}(q) = -\nabla \mathbf{U}_{atr}(q)$$

- Para assegurar que **A** não chegue perto de **B** sem ser repelido, pode-se combinar um pequeno número de pontos de controle fixos e um ponto de controle variável, dependente da configuração q.

- O ponto de controle variável é o ponto $a' \in \partial\mathbf{A}$ que é mais próximo a **B** na configuração corrente, o qual é solução de:

  $\min_{b\in\mathbf{B}}\|a'(q) - b\| = \min_{a\in\mathbf{A},b\in\mathbf{B}}\|a(q) - b\|$

- A força repulsiva aplicada sobre o ponto P = a' é:

$$\mathbf{F}_{rep}^{i}(q) = \begin{cases} \eta.[(1/\rho(P))-(1/\rho_0)].(1/\rho(P))^2.\nabla\rho(P) & \text{se } \rho(P) \le \rho_0 \\ 0 & \text{se } \rho(q) > \rho_0 \end{cases}$$

- Se **A** e **B** são polígonos convexos, o cálculo de $\min_{a\in\mathbf{A},b\in\mathbf{B}}\|a(q)-b\|$ e do par de pontos que satisfazem esta distância mínima pode ser realizado em um tempo $O(n_A + n_B)$, onde n_A e n_B são o número de vértices de **A** e **B**.

- Se **A** e **B** são não convexos, podem ser decompostos em componentes convexas $\{\mathbf{A}_k\}$ e $\{\mathbf{B}_l\}$. Para cada par $(\mathbf{A}_k,\mathbf{B}_l)$ pode-se definir um ponto de controle variável a_{kl}' no limite de $\mathbf{A}_k$ que seja o mais próximo de $\mathbf{B}_l$.

- Todos os pontos de controle variável a_{kl}' estão sujeitos simultaneamente ao campo de potencial repulsivo.

Planejamento de Caminhos guiado por Potencial:

- Abordagem: considerar o robô no espaço de configuração como uma partícula de massa unitária movendo-se sob a influência de um campo de forças artificial $F = -\nabla U$.

- A cada configuração q, a força F(q) determina a aceleração da partícula.

- Conhecendo a dinâmica de **A** e supondo potência de atuadores ilimitada, pode-se calcular os esforços a serem aplicados a cada instante pelos atuadores de modo que o robô se comporte efetivamente como uma partícula.

- Os esforços calculados são os comandos enviados aos servo controladores do robô.

- Abordagem aplicável para geração de caminhos *on-line*.

- Se um modelo dos obstáculos é conhecido *a priori*, a mesma abordagem pode ser utilizada para efetuar planejamento de caminhos *off-line*.

Planejamento em Profundidade (*Depth First*):

- Construir o caminho como um produto de segmentos de caminho sucessivos partindo da configuração inicial q_{ini}.
- Cada segmento é orientado na direção e sentido do negativo do gradiente da função de potencial calculado na configuração alcançada pelo segmento prévio.
- A amplitude do segmento é escolhida de modo a que este permaneça dentro do espaço de configuração livre.

- Sejam q_i e q_{i+1} as extremidades do i-ésimo segmento do caminho e $x_j(q_i)$ as coordenadas de q_i expressas numa carta (U,ϕ), (com j = 1,...,m). A força artificial expressa na base β induzida pela carta no espaço tangente $T_q(C)$ é dada por:

  $$[F]_\beta = -[\nabla U]_\beta = [-\partial \mathbf{U}/\partial x_1 \ ... \ -\partial \mathbf{U}/\partial x_m]^T$$

- Denominando $t_j(q_i)$ = componente j do vetor unitário $t(q_i)=F(q_i)/||F(q_i)||$ em β, as coordenadas da configuração q_{i+1} são obtidas na iteração i por:

  $$x_j(q_{i+1}) = x_j(q_i) + \delta_i . t_j(q_i), \qquad j = 1, ..., m.$$

onde δ_i é o comprimento do i-ésimo incremento medido de acordo com a métrica euclidiana de $\mathbb{R}^m$.

- O segmento $q_i q_{i+1}$ é a imagem inversa em C do segmento de reta entre $\phi(q_i)$ e $\phi(q_{i+1})$ em $\mathbb{R}^m$.

Exemplo:

- **A** = robô planar movendo-se em $\mathbb{R}^2$.
- Parametrização: $q = (x,y,\theta) \in \mathbb{R}^2 \times [0,2\pi)$.

$x(q_{i+1}) = x(q_i) + \delta_i.[\partial \mathbf{U}(x,y,\theta)/\partial x]$

$y(q_{i+1}) = x(q_i) + \delta_i.[\partial \mathbf{U}(x,y,\theta)/\partial y]$

$\theta(q_{i+1}) = \theta(q_i) + \delta_i.[\partial \mathbf{U}(x,y,\theta)/\partial \theta] \quad \mod 2.\pi$

- Pode-se normalizar os deslocamentos ao longo do eixo θ em relação aos deslocamentos ao longo dos eixos x e y, parametrizando $q = (x,y,\phi) \in \mathbb{R}^2 \times [0,2\pi.R)$, impondo $\phi = \theta.R$, onde $R = \max_{a \in \partial \mathbf{A}} \|O_A - a\|$ = máxima distância entre a origem O_A e o limite do robô $\partial \mathbf{A}$:

$\phi(q_{i+1}) = \phi(q_i) + \delta_i.[\partial \mathbf{U}(x,y,\phi)/\partial \phi] \quad \mod 2.\pi.R$

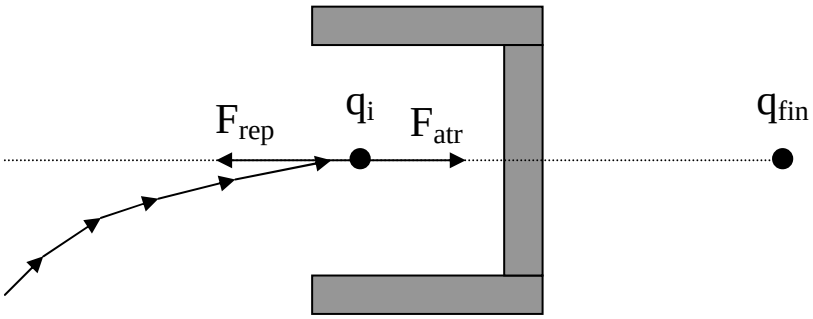
ou, de modo equivalente,

$\theta(q_{i+1}) = \theta(q_i) + (\delta_i/R).[\partial \mathbf{U}(x,y,\theta R)/\partial \phi] \quad \mod 2.\pi$

- Tipicamente, δ_i é pré-especificado como um valor δ suficientemente pequeno.
- δ_i deve ser escolhido pequeno o suficiente de modo que a direção da força e o seu mapeamento no sistema de coordenadas local mantenha seu significado ao longo do segmento $q_i q_{i+1}$.
- δ_i deve ser escolhido pequeno o suficiente de modo que não aconteça nenhuma colisão ao longo do segmento $q_i q_{i+1}$.

- Seja L o limite superior para o deslocamento máximo de um ponto de **A** durante o movimento de q_i para q_{i+1} de comprimento δ, se L for menor do que a menor distância entre **A** e **B**, então é garantido que o segmento $q_i q_{i+1} \subset C_L$. Caso contrário, pode-se escolher $\delta_i = \delta/2$. Proceder iterativamente até garantir caminho livre entre q_i e q_{i+1}.
- O incremento δ_i não deve passar além de q_{fin}. Escolher δ_i menor do que a distância euclidiana entre $\phi(q_i)$ e $\phi(q_{fin})$, onde ϕ é o mapeamento que define a carta (U,ϕ) adotada.

- Devido ao movimento ser feito em incrementos discretos δ_i, ao atingir um mínimo local, o robô permanece oscilando em torno do mesmo.



Procedimento:

- Detectar a situação de mínimo local checando sucessivas configurações e verificando se elas não estão "muito próximas" e se aconteceram "retrocessos".
- Escapar do mínimo local. Um método simples consiste em movimentar-se durante um certo tempo ao longo de uma direção pré-especificada, (exemplo: tangente à superfície eqüipotencial, para contornar os obstáculos). Retomar posteriormente o planejamento em profundidade.

Planejamento Melhor Primeiro:

- Decompor o espaço de configuração numa grade regular fina m-dimensional, GR, através da discretização dos m eixos de coordenadas de C.

Exemplo: se **A** = objeto em vôo livre em $\mathbb{R}^2 \Rightarrow$ GR é formada pelas configurações ($k_x.\delta x$, $k_y.\delta y$, $k_\theta.\delta\theta$), com k_x, k_y, $k_\theta \in \mathbb{Z}$, aplicando aritmética módulo 2π em θ. O incremento angular $\delta\theta$ é uma fração inteira de 2π.

- Configurações p-vizinhas da configuração $q \in$ GR são todas aquelas em GR tendo até p, ($1 \leq p \leq m$), coordenadas que diferem exatamente um incremento (em valor absoluto) das coordenadas correspondentes de q.
- Existem 2m 1–vizinhas, $2m^2$ 2-vizinhas, ..., 3^m-1 m-vizinhas.

Pressupostos:

- q_{ini} e q_{fin} são configurações em GR.
- Se duas configurações vizinhas em GR estão em C_L, o segmento de reta que as conecta em $\mathbb{R}^m$ também está em C_L.
- GR é um retangulóide que limita as possíveis posições de **A**.

- O método de Planejamento Melhor Primeiro (PMP) constrói iterativamente uma árvore T cujos nós são configurações em GR e sua raiz é q_{ini}.
- A cada iteração examina-se as vizinhas da folha de T que tem o menor potencial, são retidas as vizinhas que ainda não estiverem na árvore cujo potencial for menor que um limiar M grande e as instala em T como sucessoras da folha corrente.
- O algoritmo termina quando se atinge q_{fin} (sucesso), ou quando o subconjunto livre de GR acessível a partir de q_{ini} tiver sido completamente explorado (falha).
- Cada nó possui um ponteiro para o seu nó pai. Ao atingir q_{fin}, o caminho é gerado traçando os ponteiros de q_{fin} até q_{ini}.

Estruturas de dados e operações utilizadas no algoritmo PMP:

- T = árvore de busca.
- ABERTA = lista que contém as folhas de T ordenadas por valores crescentes da função de potencial.
- PRIMEIRO(ABERTA) = remove e retorna a configuração de ABERTA com menor potencial.
- INSERIR(q,ABERTA) = insere q em ABERTA.
- VAZIA(ABERTA) = retorna **Verdadeiro** se a lista ABERTA é vazia ou **Falso**, caso a lista não for vazia.

procedimento PMP
começar
instalar q_{ini} em T; /* inicialmente T é uma árvore vazia */
INSERIR(q_{ini},ABERTA); marcar q_{ini} como **Visitada**;
/* inicialmente todas as configurações em GR são marcadas como **Não Visitada** */
SUCESSO ← **Falso**;
enquanto ¬VAZIA(ABERTA) **e** ¬SUCESSO **faça**
começar
q ← PRIMEIRO(ABERTA);
para cada vizinha q' de q em GR **faça**
se U(q') < M e q' é **Não Visitada, então**
começar
instalar q' em T com um ponteiro para q;
INSERIR(q',ABERTA);
marcar q' como **Visitada**;
se q'=q_{fin}**, então** SUCESSO ←**Verdadeiro**;
fim;
fim;
se SUCESSO**, então**
retornar o caminho gerado traçando os ponteiros em T de q_{fin} até q_{ini};
se não¸ retornar falha;
fim;

Observações:

- PMP segue uma aproximação discreta do negativo do gradiente da função de potencial U(q) até atingir um mínimo local.

$\Rightarrow$ Ao atingir o mínimo local, o algoritmo opera de modo a "preencher" o "vale" que contém o mínimo, deixando-o plano.

$\Rightarrow$ A partir do "vale" "preenchido", o algoritmo PMP segue novamente $-\nabla U(q)$, descendo uma nova "ladeira" de potencial em direção a um novo mínimo (local ou global).

$\Rightarrow$ É garantido encontrar um caminho livre sempre que este existir dentro do subconjunto livre da grade GR. Caso contrário, uma falha será reportada.

- A complexidade computacional do algoritmo PMP é de ordem $O(m.r^m.\log(r))$, onde r = número de pontos de discretização ao longo de cada eixo de coordenadas do espaço de configuração.
- A maior parte do tempo, PMP explora somente um pequeno subconjunto de GR.

$\Rightarrow$ Em lugar de representar GR explicitamente como um grande arranjo m-dimensional, pode-se representar apenas as configurações instaladas em T.

$\Rightarrow$ Em lugar de testar se uma configuração é Visitada ou não, checa-se se ela está em T ou não.

$\Rightarrow$ Economiza-se um grande espaço de memória, (às custas de uma ligeira perda de eficiência devido à necessidade adicional de checar se a configuração está em T).

$\Rightarrow$ Torna-se possível trabalhar com grade de tamanho ilimitado, (sujeito às limitações de tempo de computação).

- PMP é prático quando $m < 5$. Exemplo: para $m = 3$, com $q=(x,y,\theta)$, PMP é extremamente rápido e confiável com resoluções da grade GR da ordem de 256^3.

$\Rightarrow$ Pode-se acelerar o algoritmo usando pirâmides de grades de várias resoluções.

- Para $m > 5$, o "preenchimento" de vales com mínimos locais se torna intratável, pois o número de configurações discretas em um vale tende a crescer exponencialmente com m.

Outras Funções de Potencial:

Objetivos:

1) Melhorar o comportamento dinâmico local ao longo dos caminhos gerados.
   - Importante quando o caminho é gerado *on-line*, como no algoritmo "Fundo Primeiro".
   - Exemplo: Campo de Potencial Generalizado, função da configuração e da velocidade. $\Rightarrow$ O robô só é repelido se estiver perto de um obstáculo e movendo-se na sua direção.
   - Sujeito a maiores problemas com mínimos locais.
2) Reduzir o número de mínimos locais e/ou o tamanho dos "vales" associados aos mesmos.
   - Mínimos locais são o maior problema dos métodos baseados em potencial.
   - O tipo da função de potencial não é imposta na formulação original do método.
   - Questão: é possível construir uma função $U : C_L \rightarrow \mathbb{R}$ com um mínimo em q_{ini} cujo domínio de atração seja a totalidade do subconjunto de C_L conexo a q_{fin}?
   - Se existir, esta função é chamada Função de Navegação Global. $\Rightarrow$ Com base nesta função, o método "Fundo Primeiro" garante encontrar um caminho entre q_{ini} e q_{fin}, quando este existir.
   - A Função de Navegação Global não existe no caso geral, entretanto, não há restrições à existência de uma função de potencial (Função de Navegação) com um mínimo em q_{fin} cujo domínio de atração inclua a totalidade do subconjunto de C_L conexo a q_{fin}, exceto um número finito de pontos de sela da função de potencial (pontos de equilíbrio instável).
   - A Função de Navegação em forma analítica pode ser obtida para casos particulares. Para uma representação discreta de C_L, este problema é muito mais simples.

Função de Navegação Manhattan:

- Discretizar C numa grade retangulóide GR.
- Cada configuração q em GR é rotulada "0" se $q \in C_L$ e "1" caso contrário. $\Rightarrow GR_L = \{q \in GR / q = \text{"0"}\}$.
- Assume-se que q_{ini} e q_{fin} pertencem a GR_L.

- Função de Navegação Manhattan = distância L^1 a q_{fin}.
- Computada facilmente usando uma "expansão de frente de onda" a partir de q_{fin}:
  - $U(q_{fin})$ é fixado em 0.
  - O potencial de cada 1-vizinha de q_{fin} é fixado em 2.
  - O procedimento prossegue iterativamente incrementando em 1 o potencial das configurações 1-vizinhas à configuração atual em GR_L, cujo potencial não tenha ainda sido computado.
  - Terminar quando o subconjunto de GR_L acessível a partir de q_{fin} tenha sido explorado completamente.

Observações:

- A função de navegação gera potenciais em GR_L com um único mínimo em q_{fin}.
- Pode-se usar o algoritmo de navegação "Fundo Primeiro" para buscar um caminho entre q_{ini} e q_{fin} em GR_L.
- É garantido encontrar um caminho entre q_{ini} e q_{fin} caso este exista em GR_L.
- O caminho gerado em GR_L entre q_{ini} e q_{fin} é de comprimento mínimo (de acordo com a métrica L^1).
- O algoritmo calcula U(q) somente no subconjunto de GR_L conexo a q_{fin}.
- Se $U(q_{ini})$ não foi computado, pode-se concluir imediatamente que não existe um caminho entre q_{ini} e q_{fin} em GR_L.

- A complexidade computacional do algoritmo é linear no número de configurações em GR e independente do número e forma dos C-Obstáculos.
- O algoritmo é eficiente para um espaço de configuração de dimensão baixa (m = 2 ou 3), tornando-se impraticável para dimensões maiores.

procedimento Manhattan
começar
para cada q ∈ GR_L **faça** U(q) ← M; /* M = Nº grande */
$U(q_{fin})$ ← 0; inserir q_{fin} em L_0;
/* L_i = lista de configurações, inicialmente vazia (i=0,1,...)*/
para i = 1, 2, ..., **até** L_i = vazia**, faça**
para cada q em L_i**, faça**
para cada q' 1-vizinha de q em GR_L**, faça**
se U(q') = M**, então**
começar
U(q') ← i+1;
inserir q' no fim de L_{i+1};
fim
fim

Exemplo:

2	1	2	3	4	5		
1	q_{fin}	1	2	3	4		
2	1	2	3	4	5		
3	2			5	6		
4	3			6	7	8	q_{ini}
5	4			7	8	9	10
6	5	6	7	8	9	10	11
7	6	7	8	9	10	11	12

Função de Navegação baseada em Esqueleto:

Princípio:

1. Extrair um subconjunto (m-1)-dimensional S (Esqueleto) de GR_L.
2. Calcular a função de potencial U em S.
3. Computar a função de potencial no resto de GR_L.

1. Cômputo do esqueleto:

- Computar a distância L^1 (Manhattan) $d_1(q)$ de cada configuração $q \in GR_L$ à região de C-Obstáculos CB através de um algoritmo de expansão de frente de onda partindo de ∂CB.
- O subconjunto S (Esqueleto) é computado de forma concorrente com a construção do mapa d_1.
- O esqueleto é construído como o conjunto de configurações onde as ondas emitidas a partir do limite de CB se encontram.
- Para conseguir isto, propaga-se não só os valores de d_1 em q, mas os pontos do limite de GR_L, O(q), que os originaram.
- Seja $q' \in GR_L$, uma configuração 1-vizinha de q. Se a distância L^1, D1(O(q),O(q')) for maior do que um limiar α, duas ondas se encontram em q. $\Rightarrow$ Incluir q em S.

- O esqueleto computado desta maneira é um tipo de Diagrama de Voronoi para a distância L^1.
- O esqueleto computado desta maneira é semelhante ao "esqueleto" extraído de uma região em uma imagem digitalizada usando técnicas de morfologia matemática.

Seja:

- L_i = lista de configurações (inicialmente vazia) correspondentes a uma mesma frente de onda (mesma distância L^1 = i, com i = 0,1, ...).
- INSERIR(q,L_i) = insere a configuração q no final da lista L_i.
- S = Lista que armazena as configurações do esqueleto.
- INSERIR(q,S) = insere a configuração q no final da lista S.

```
procedimento ESQUELETO
começar
    para cada q ∈ GR_L, faça U(q) ← M    /* M = Nº grande */
    para cada q ∈ GR\GR_L, faça          /* inicializar borda */
        se ∃ q' 1-vizinha de q e q' ∈ GR_L, então
        começar
            d1(q) ← 0;      O(q) ← q;
            INSERIR(q,L0);
        fim;
    para i = 0, 1, ..., até L_i = lista vazia, faça
        para cada q ∈ L_i, faça      /* propagação da onda */
            para cada q' ∈ GR_L 1-vizinha de q, faça
                se d1(q) = M, então
                começar
                    d1(q') ← i+1;   O(q') ← O(q);
                    INSERIR(q',L_i+1);
                fim;
                se não, se D1(O(q),O(q'))>α, então
                /* α inteiro tipicamente entre 2 e 6 */
                    se q ∉ S, então INSERIR(q',S);
                    /* Evita gerar esqueleto de largura
                       igual a duas configurações */
fim;
```

2. Cálculo da função de potencial U em S:

- Primeiro, q_{fin} é conectada a S por um caminho σ seguindo a "subida da ladeira" do mapa d_1 em GR_L.
- O caminho σ é incluído em S.
- Estabelece-se $U(q_{fin}) = 0$.
- U é computada em S através de um algoritmo de expansão de frente de onda restrita a S, partindo de q_{fin}.
- O algoritmo usa o mapa d_1 computado previamente para guiar a expansão.

Seja:

  - Q = lista de configurações em S ordenada por valores decrescentes de d_1.
  - INSERIR(q,Q) insere a configuração q na lista Q.
  - PRIMEIRA(Q) remove e retorna a 1ª configuração de Q.
  - VAZIA(Q) = Verdadeiro se Q = {∅}, Falso caso contrário.
  - L_i = lista de configurações inicialmente vazia, com i=1,2, ...
  - INSERIR(q,L_i) insere a configuração q no final da lista L_i.
  - σ = lista de configurações inicialmente vazia.
  - INSERIR(q,σ) insere a configuração q no final da lista σ.

- q_{fin} é inserida em Q.
- Até que Q esteja vazia, a primeira configuração q de Q é removida.
- Cada configuração q' m-vizinha de q em S cujo potencial não tenha sido computado recebe um potencial U(q') = U(q) + 1 e é inserida em Q.
- O algoritmo termina quando Q ficar vazia (ou seja, quando todas as configurações em S acessíveis a partir de q_{fin} tenham seu potencial calculado).
- A saída do algoritmo é a lista L_0 contendo todas as configurações de S acessíveis a partir de q_{fin}.

```
procedimento U_ESQUELETO
começar
    /* Conexão de q_fin a S */
    para cada q∈ GR_L, faça U(q) ← M    /* M = Nº grande */
    INSERIR(q_fin,σ);       q ← q_fin;
    enquanto q ∉ S, faça
    começar
        selecionar uma vizinha de q' com o maior valor de d_1;
        INSERIR(q',σ);       q ← q';
    fim;
    S ← σ ∪ S;
    /* Cálculo de U(q) em S */
    U(q_fin) ← 0;      INSERIR(q_fin,Q);
    enquanto ¬VAZIA(Q), faça
    começar
        q ← PRIMEIRA(Q);
        INSERIR(q,L_0);
        para cada q' m-vizinha de q em S, faça
            se U(q') = M, então
            começar
                U(q') ← U(q)+1;
                INSERIR(q',Q);
            fim;
    fim;
    /* no fim deste laço, L_0 contém todas as configurações em S
       acessíveis a partir de q_fin. */
fim;
```

3. Cômputo a função de potencial no resto de GR_L:

- O potencial U no resto de GR_L acessível a partir de q_{fin} é computado através de um algoritmo de expansão de frente de onda partindo das configurações em L_0.
- O potencial de cada configuração q' 1-vizinha de cada configuração q em L_0 é estabelecido como U(q') = U(q) + 1.
- O potencial de cada configuração q'' 1-vizinha de cada configuração q' cujo potencial já tenha sido calculado é estabelecido como U(q'') = U(q') + 1.
- O procedimento é repetido iterativamente para todas as configurações de GR_L acessíveis a partir de q_{fin}.

```
procedimento U_GRL
começar
    para i = 0,1,..., até Li ser vazia, faça
        para cada q em Li, faça
            para cada q' vizinha de q em GRL, faça
                se U(q') = M, então   /* M = Nº grande */
                começar
                    U(q') ← U(q) +1;
                    INSERIR(q',Li+1);
                fim;
fim;
```

- A função de potencial gerada em GR_L contém apenas um mínimo em q_{fin}.
- Aplicando o algoritmo de busca em profundidade a partir de q_{ini}, é gerado um caminho que:
    1. Conecta q_{ini} a S.
    2. Segue a curva S, (o mais distante dos C-obstáculos).
    3. Conecta S a q_{fin}.
- A complexidade computacional do algoritmo é de ordem $O(n+n^{(m+1)/m}.\log(n))$, onde n = Nº de configurações em GR.

Aplicação em espaços de maior dimensão:

- As funções de navegação Manhattan e Esqueleto, embora de aplicação geral, na prática só são eficientes para espaços de configuração de pequena dimensão (m = 2 ou 3).
- O tamanho da grade cresce exponencialmente com m.
- Solução: computar potenciais em espaço de trabalho associados a pontos de controle no robô e combiná-los para produzir um potencial de espaço de configuração.

Considerações iniciais:

- Define-se uma grade regular GW de pontos no espaço W construída discretizando os eixos do referencial F_W.
- O incremento δ ao longo dos eixos de F_W deve ser igual ao incremento Δ ao longo dos eixos de translação em GR.
- Assunção: GW é limitada e forma um retangulóide.
- Cada ponto P de GW é rotulado "1" se $P \in B$ e "0" se $P \notin B$.
- $GW_L = \{P / P \in GW$ e P é rotulado "0"$\}$.
- Sejam a_i (com i = 1, ..., Q) pontos de controle em A.
- V_i é a função de potencial associada a a_i, definida sobre GW_L.

- V_i é computada usando a função Manhattan ou Esqueleto, substituindo GR por GW, GR_L por GW_L, q_{fin} por $a_i(q_{fin})$, q por P e q' por P'.
- Para calcular U(q), determina-se a posição de cada $a_i(q)$. O valor V_i é calculado considerando o ponto $P_i \in GW$ mais próximo a a_i. A partir dos V_i's, a função de potencial U(q) na configuração $q \in GR_L$ é calculada através de uma "Função de Arbitragem", G:

$$U(q) = G(V_1(a_1(q)), \ldots , V_Q(a_Q(q)))$$

- Se Q < N, existem múltiplas configurações finais, se Q = N, usualmente q_{fin} é unicamente definida.
- U(q) não precisa ser computada em cada q ∈ GR, mas apenas no pequeno subconjunto de GR onde o planejador precisa conhecer o valor do potencial.
- $U(q) \geq 0$ no espaço livre, com $U(q_{fin}) = 0$.
- U(q) construída a partir de potenciais na grade GW evita computar potenciais sobre a grade GR (bem maior).
- O espaço W é usado como inspiração para obter uma "boa" função de potencial (através da combinação dos V_i's).
- Em geral, a função "Esqueleto" fornece potenciais V_i's que são "melhor" combinados do que aqueles produzidos pela função "Manhattan", pois "aumenta" o espaço de manobra do robô.
- Pode-se facilmente construir uma função U(q) que evite que o robô fique preso em "concavidades" formadas por obstáculos.
- Por outro lado, a combinação dos V_i's produz mínimos locais devidos à competição entre os vários a_i's para atingir suas posições finais. ⇒ É necessária uma Função de Arbitragem.

a) Função de Arbitragem: $U(q) = \sum V_i(a_i(q))$
   - Esta escolha não favorece nenhum ponto de controle, tende a aumentar o número de conflitos e produz muitos mínimos locais.

b) Função de Arbitragem: $U(q) = \min V_i(a_i(q)) + \varepsilon.\max V_i(a_i(q))$
   - ε é uma constante pequena (tipicamente, ε = 0.1).
   - O primeiro termo favorece a atração do ponto que está mais próximo do objetivo. O segundo termo faz o robô girar (menos intensamente) em direção à orientação final.
   - Tende a gerar menos mínimos do que a função a).
   - Cria mínimo fundo quando não há espaço de manobra.

c) Função de Arbitragem: $U(q) = \max V_i(a_i(q))$
   - Tende a aumentar o número de competições entre pontos de controle (mais mínimos locais).
   - Tende a reduzir o domínio de atração dos mínimos.

Planejamento Probabilístico:

Princípio:

- O Planejador de Caminhos Probabilístico (PCP) constrói e pesquisa um grafo G cujos nós são os mínimos locais da função de potencial U(q), os quais são conectados por um arco se o planejador tiver gerado um caminho entre eles.
- O método não requer função de potencial específica. Assume-se que $U(q) \geq 0$ é uma grade retangulóide GR com incrementos Δ_i ao longo dos eixos x_i e com mínimo global $U(q_{ini}) = 0$.
- O planejador parte de q_{ini} executando movimentos de acordo com a técnica **"Melhor Primeiro"** até alcançar um mínimo local, q_{loc}. Se $U(q_{loc}) = 0$, sucesso.
- Ao atingir um dado mínimo local q_{loc}, PCP procura escapar do mesmo através de movimentos Aleatórios.
- Cada movimento aleatório é seguido por um movimento "melhor primeiro" que leva a um mínimo local. Se este é diferente de q_{loc}, é inserido no grafo G como sucessor de q_{loc}.
- Portanto, dois nós adjacentes no grafo G são conexos por um caminho composto por um trecho "melhor primeiro" seguido por um trecho "aleatório".
- O grafo G é construído incrementalmente até encontrar a configuração q_{fin} ou até atingir um critério de parada.
- Após encontrar um caminho, o mesmo deve ser transformado num caminho suave.

Movimento “Melhor Primeiro”:

- O planejador opera sobre a grade do espaço de configuração GR na qual U(q) é definida.
- A partir da configuração atual, o robô deve ser movimentado para a melhor configuração vizinha (aquela com menor U(q)).
- O planejamento prossegue iterativamente até atingir um mínimo local q_{loc}, cujas vizinhas possuem todas $U(q) > U(q_{loc})$.
- Se o método de cálculo de potencial não garantir que U(q) tenda a infinito no limite de CB, antes de escolher uma configuração como sucessora da configuração atual, deve-se verificar se a mesma está em C_L.

Movimento Aleatório:

- Movimentos aleatórios partem de um mínimo local q_{loc}.
- Consistem de uma série de t passos tais que a projeção de cada passo ao longo dos eixos x_i, (i = 1, ... , m), é escolhida aleatoriamente Δ_i ou $-\Delta_i$.
- Cada passo é escolhido independente dos anteriores.
- Este caminho converge para um movimento browniano quando cada Δ_i tende a zero.
- Assumindo, sem perda de generalidade, que q_{loc} está na origem do sistema de coordenadas, a configuração atingida após t passos é uma variável aleatória $Q(t) = (Q_1(t), \ldots , Q_m(t))$ que satisfaz as duas propriedades seguintes:

  - Densidade de $Q_i(t)$: $p_i(q_i) = [\exp(-q_i^2/(2\Delta_i^2.t))]/[\Delta_i(2\pi.t)^{1/2}]$

  - Desvio Padrão de $Q_i(t)$: $D_i = \Delta_i.(t)^{1/2}$

- Um passo aleatório pode levar a colisão com CB. Deve-se checar se a configuração gerada está em C_L, caso contrário, o planejador deve selecionar um novo passo aleatório.

- A duração t do caminho deve ser escolhida com cuidado. Se for muito pequena, o robô tem pouca probabilidade de escapar do mínimo. Se for muito grande, o planejador desperdiçará tempo em que poderia navegar por "melhor primeiro".
- Define-se **Raio de Atração**, $R_i(q_{loc})$ de qualquer mínimo local q_{loc} como a distância de q_{loc} ao ponto de sela mais próximo na direção x_i. $\Rightarrow$ $R_i(q_{loc})$ é a mínima distância que o robô deve andar para escapar do mínimo local q_{loc}.
- Se for possível estimar a estatística de R_i, de $D_i = \Delta_i.(t)^{1/2}$ pode-se fazer uma estimativa boa da duração t do caminho aleatório:

$$t = \max_{i \in [1,m]}[R_i(q)/\Delta_i]^2$$

- Entretanto, como não se faz nenhuma assunção a respeito da distribuição dos obstáculos, nenhuma estatística forte pode ser inferida a respeito de U(q) e dos R_i's.
- Como qualquer modificação na configuração do robô não deverá causar que qualquer ponto de A se mova mais do que o comprimento L dos lados de GW, pode-se estimar $t = L/\delta^2$, onde δ é o incremento ao longo dos eixos de GW. $\Rightarrow$ Esta estimativa assume implicitamente que todos os R_i's são iguais.
- Outra possibilidade é considerar que R_i é uma variável aleatória com distribuição de Laplace e com valor esperado L. Neste caso, t é escolhido como uma variável aleatória T com a seguinte densidade de probabilidade:

$$p(t) = \delta.[\exp(-\delta(L.t)^{1/2})]/[2(L.t)^{1/2}]$$

- Neste caso, o valor esperado de t é: $T = L/\delta^2$ (maior duração do caminho aleatório).
- A cada passo, o planejador checa U(q) em relação a $U(q_{loc})$. Se for menor, o planejador termina o caminho aleatório.

Busca no Grafo:

- O Grafo G é construído incrementalmente, a partir das combinações de movimentos "Melhor Primeiro" e Aleatório.

- 1ª Estratégia de busca "Melhor Primeiro":
  - Gerar iterativamente sucessores do mínimo local de menor potencial.
  - Limitar em um máximo K predefinido, o número de movimentos aleatórios gerados a partir de um mesmo mínimo local.
  - Problema: o mesmo mínimo pode ser alcançado várias vezes, o que é difícil e custoso de detectar.
  - Problema: a estratégia pode desperdiçar muito tempo explorando grandes domínios de atração de mínimos locais que contenham dentro deles pequenos mínimos locais imbricados.

- 2ª Estratégia de busca:
  - A partir de um mínimo local q_{loc}, gerar até um máximo de K movimentos aleatórios.
  - Cada movimento aleatório é seguido imediatamente por um movimento "melhor primeiro", atingindo um mínimo local q'_{loc}.
  - Se $U(q'_{loc}) > U(q_{loc})$, esquecer q'_{loc}. Caso contrário (e se $q'_{loc} \neq q_{fin}$), gerar o sucessor de q'_{loc} da mesma maneira.
  - Se nenhum dos K movimentos aleatórios a partir de q_{loc} consegue atingir um mínimo com potencial menor do que q_{loc}, considerar q_{loc} um beco se saída e retomar a busca a partir do mínimo considerado mais recentemente cujos K sucessores não tenham sido todos gerados.
  - Problema: apesar de apresentar bons resultados, também pode despender muito tempo com mínimos imbricados.
  - Problema: se um mínimo local baixo foi alcançado, pode ser bastante difícil alcançar um outro mínimo menor.

- 3ª Estratégia de busca:
  - Modificação da 2ª Estratégia.
  - Em lugar de armazenar o grafo G completo, memorizar apenas o caminho gerado τ conectando q_{ini} à configuração atual q.
  - A cada mínimo local, gerar até um máximo de K (tipicamente, K $\cong$ 20) movimentos aleatórios/"melhor primeiro".
  - Se um destes movimentos atingir um mínimo q'_{loc} mais baixo, inserir o caminho entre q_{loc} e q'_{loc} no final do caminho τ corrente, continuando a busca a partir de q'_{loc}.
  - Caso contrário, selecionar aleatoriamente, (de acordo com uma distribuição normal), uma configuração de retrocesso $q_{trás}$ no subconjunto de τ corrente formado por movimentos aleatórios.
  - A busca é retomada, a partir de $q_{trás}$, usando um movimento "melhor primeiro". Como $q_{trás}$ foi gerada por movimento aleatório, esta nova busca pode levar a um mínimo inexplorado.
  - Observação: se o primeiro mínimo local q_{loc} for um beco sem saída, este mecanismo de retrocesso não pode ser aplicado.
  - $\Rightarrow$ Selecionar aleatoriamente um dos mínimos locais q'_{loc} atingidos a partir de q_{loc} e retomar a busca em movimento "melhor primeiro" a partir de uma configuração selecionada aleatoriamente no caminho aleatório que leva a q'_{loc}.

Estruturas de dados e comandos utilizados no Algoritmo de Planejamento de Caminhos Probabilístico:

- τ = lista de configurações representando o caminho gerado.
- ULT(τ) retorna a última configuração de τ.
- PROD(τ_1,τ_2) retorna o produto dos caminhos $\tau_1 \bullet \tau_2$.
- MELHORPR(q) retorna o caminho gerado a partir de q através de um movimento "melhor primeiro".
- RAND(q,t) retorna o caminho gerado a partir de q através de um movimento aleatório de duração t.
- RAND_T retorna um intervalo de tempo aleatório computado de acordo com a distribuição:

$$p(t) = \delta.[\exp(-\delta(L.t)^{1/2})]/[2(L.t)^{1/2}]$$

- VOLTA(τ, τ_1, ..., τ_K) seleciona uma configuração de retrocesso $q_{trás}$ e retorna o caminho de q_{ini} a $q_{trás}$; se τ inclui um subcaminho gerado por movimento aleatório, o caminho retornado é um subcaminho de τ; caso contrário, é um subcaminho de $\tau \bullet \tau_i$, com i escolhido aleatoriamente entre [1,K].

Algoritmo de Planejamento de Caminhos Probabilístico:

procedimento PCP
começar
$\tau \leftarrow$ MELHORPR(q_{ini}); $q_{loc} \leftarrow$ ULT(τ);
enquanto $q_{loc} \neq q_{fin}$**, faça**
começar
ESCAPE ← Falso;
Para i = 1 **a** K **até** ESCAPE**, faça**
começar
t ← RAND_T;
$\tau_i \leftarrow$ RAND(q_{loc},t);
$q_{rand} \leftarrow$ ULT(τ_i);
$\tau_i \leftarrow$ PROD(τ_i, MELHORPR(q_{rand}));
$q'_{loc} \leftarrow$ ULT(τ_i);
se U(q'_{loc}) < U(q_{loc})**, então**
começar
ESCAPE ← Verdadeiro;
$\tau \leftarrow$ PROD(τ,τ_i);
fim;
fim;
se ¬ESCAPE**, então**
começar
$\tau \leftarrow$ VOLTA(τ, τ_1, ..., τ_K);
$q_{trás} \leftarrow$ ULT(τ);
$\tau \leftarrow$ PROD(τ,MELHORPR($q_{trás}$));
fim;
$q_{loc} \leftarrow$ ULT(τ);
fim;
fim;

Obtenção de um Caminho Suave:

- O caminho obtido através de PCP deve ser suavizado antes da sua execução.
- Procedimento simples:
  - Iterativamente, substituir subcaminhos de τ de comprimentos decrescentes por segmentos de reta no espaço $\mathbb{R}^m$ contendo a grade GR.
  - Discretizar cada novo segmento de acordo com a resolução de GR e checar colisão antes de inseri-lo no novo caminho.
- Inicialmente, o algoritmo tenta substituir subcaminhos cujos comprimentos são da ordem de grandeza do comprimento total do caminho. Então, subcaminhos cada vez menores são considerados até atingir a resolução da grade.

Observações:

- Resultados experimentais mostram que PCP é eficiente e confiável, isto deve-se a que o problema de planejamento tipicamente possui muitas soluções.

$\Rightarrow$ Um procedimento probabilístico de busca global pode achar uma delas, desde que bem informado na maior parte do tempo, o que é garantido pela função de potencial U(q)

- Ponto Fraco: tipicamente, PCP gera diferentes soluções, com diferentes tempos de computação, quando executado várias vezes para o mesmo problema.
- Problema: se o problema não tem solução, PCP não tem como saber, mesmo após longo tempo de computação. $\Rightarrow$ Deve-se impor um tempo máximo limite.
- Para uma classe de problemas é fácil determinar experimentalmente um limite de tempo a partir do qual, se um caminho não foi achado, existe pouca chance de encontrar um.
- O PCP é dito Resolução-Completo Probabilisticamente: se existir um caminho livre em GR, a probabilidade de encontrá-lo tende para 1 quando o tempo de busca tende para infinito.

Planejamento baseado em Funções Harmônicas:

Princípio:

Criar uma função de potencial U(q) no espaço livre C_L cujo Laplaciano seja nulo. No interior de C_L a função é livre de mínimos locais. Uma definição adequada do potencial no contorno de C_L permite estabelecer condições adequadas para navegação com base no negativo do gradiente de U(q).

Equação de Laplace:

A equação de Laplace representa uma classe de equações diferenciais parciais elípticas da forma:

$$\nabla^2 U(q) = 0 \qquad \forall q \in C_L$$

U(q) é a função harmônica solução da equação de Laplace e C_L é a região em que U(q) é definida.

O Laplaciano $\nabla^2 U(q)$ é definido como:

$$\nabla^2 U(q) = \Sigma\, \partial^2 U(q)/\partial q_i^{\,2}$$

Onde q_i são as componentes de q.

Propriedades das funções harmônicas:

a) Superposição: se U_a e U_b são funções de Laplace, qualquer combinação linear delas também é função de Laplace.
b) Valor Médio: se uma função de Laplace bidimensional definida em C_L, então, para qualquer círculo em C_L, a média dos potenciais nesse círculo é igual ao potencial no centro do mesmo. Isto tem consequência direta na ausência de mínimos locais.
c) Princípio do Mínimo e do Máximo (Min-Max): uma vez que para qualquer potencial vale a propriedade do valor médio,

não é possível o surgimento de mínimos locais dentro dessas regiões, ficando os mesmos restritos ao contorno de C_L e a pontos críticos inseridos no seu interior.

Mínimos e Máximos em Funções Harmônicas:

- As regiões internas de U(q) são livres de mínimos ou máximos.
- Mínimos ou máximos ficam restritos ao contorno de C_L e a pontos críticos inseridos no mesmo.
- Pontos críticos inseridos no interior de U(q) podem ser de origem (source) – potencial repulsivo, ou destino (sink) – potencial atrativo.
- U(q) corresponde a um potencial harmônico cujo gradiente leva o robô da origem ao destino.
- Como no resto do potencial não há máximos ou mínimos, seguindo o negativo do gradiente, o robô converge para o destino.

Condições de Contorno de Funções Harmônicas:

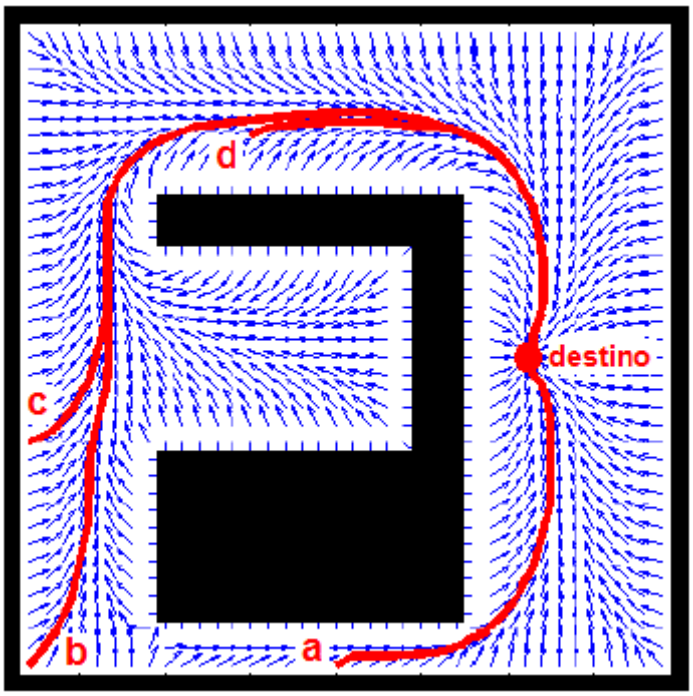
- As regiões de contorno, representadas pelos obstáculos, delimitam o espaço livre (espaço navegável).
- Estas regiões definem a forma do potencial no interior do espaço livre.
- Condições de contorno clássicas:
  - Condição de Dirichlet.
  - Condição de Neumann.

Condição de Contorno de Dirichlet

O potencial U(q) na região de contorno ∂C_L na condição de contorno de Dirichlet é imposto como um valor constante K:

$$U(q) = K = \text{constante}, \qquad \forall q \in \partial C_L$$

- Se um ponto crítico com potencial constante e menor que K é inserido no interior do potencial, U(q) induzirá um gradiente entre o contorno e este ponto crítico que tenderá a seguir na direção normal à região de contorno ∂C_L.



Características:

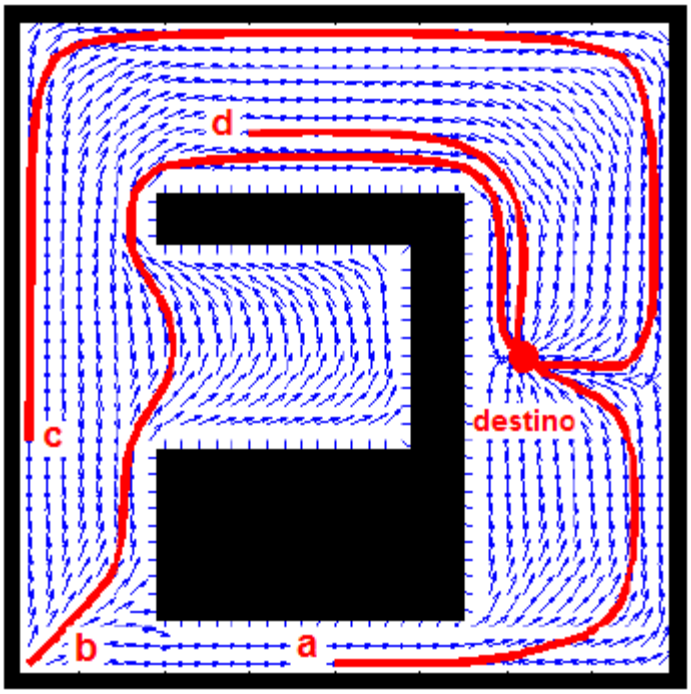
- Do ponto de vista de navegação, a condição de Dirichlet resulta em potenciais que tendem a direcionar o robô por caminhos afastados dos obstáculos.
- Os caminhos gerados são mais seguros.
- O potencial que satisfaz a condição de Dirichlet apresenta dificuldade na geração de caminhos por passagens estreitas.
- Não é necessário introduzir um ponto crítico no ponto de origem do movimento.
- O mapa de potencial pode ser reaproveitado para pontos de origens diferentes.

Condição de Contorno de Neumann

O potencial U(q) na região de contorno ∂C_L na condição de contorno de Neumann é imposto de tal forma que o gradiente de U(q) na direção normal a ∂C_L, **n**, seja nulo:

$$\partial U(q)/\partial \mathbf{n} = 0, \qquad \forall q \in \partial C_L$$

Uma vez que o gradiente possui componente normal nula na região de contorno ∂C_L, esta condição força a criação de um gradiente paralelo ao contorno ∂C_L.



Características:

- Do ponto de vista de navegação, a condição de Neumann resulta em potenciais que tendem a direcionar o robô por caminhos próximos ao contorno dos obstáculos, o que pode gerar caminhos mais longos.
- Os caminhos gerados são menos seguros.
- É necessário introduzir um ponto crítico no ponto de origem do movimento para forçar o surgimento do gradiente, pois sem o mesmo a equação de Laplace e a condição de

contorno são satisfeitas apenas pela solução trivial (potencial de qualquer ponto igual ao potencial do destino).

- O mapa de potencial não pode ser reaproveitado quando o ponto de origem do movimento muda.

Obtenção dos Potenciais Harmônicos

O cálculo dos potenciais harmônicos envolve a solução da equação de Laplace no domínio em que U(q) é definida. (espaço navegável), de modo a satisfazer as condições de contorno.

O planejamento baseado em funções harmônicas tem sido abordado tanto no domínio contínuo como no domínio discreto. No domínio contínuo, usando o princípio da superposição, o campo de potencial é modelado pela superposição de diferentes componentes do campo gerados por diferentes estruturas do ambiente. Assim, o ambiente é decomposto em estruturas primitivas menores, de forma a possibilitar a representação analítica de todo o espaço. Apesar de ser possível aplicar esta abordagem para casos particulares de ambientes simples, este processo é complexo e a complexidade tende a aumentar com o tamanho do mapa e com a densidade dos obstáculos. Por outro lado, a representação do ambiente no domínio discreto permite calcular o potencial através de métodos numéricos e técnicas iterativas de fácil implementação para diferentes tipos de ambientes, o que torna esta abordagem bastante utilizada em aplicações práticas do método.

Para um ambiente bidimensional, uma abordagem comum é discretizar o plano (x,y) em uma grade regular. A grade é obtida discretizando os eixos x e y em intervalos regulares Δx e Δy. Considerando intervalos de discretização iguais para os eixos x e y, $\Delta x = \Delta y = h$, a equação de Laplace pode ser discretizada expandindo a função de potencial U(x,y) em série de Taylor, em torno do ponto (x,y), desprezando as derivadas de ordem superior, de forma a obter aproximações discretas para as derivadas

segundas de U(x,y) em relação a x e a y, respectivamente. Com base nesta metodologia, a equação de Laplace pode ser aproximada no domínio discreto por:

$$[U_{x,y-h} + U_{x,y+h} + U_{x-h,y} + U_{x+h,y} - 4U_{x,y}]/h^2 = 0$$

A aproximação discreta da equação de Laplace pode ser resolvida para o domínio em que U(x,y) é definida e para as condições de contorno especificadas, usando um variado repertório de soluções algorítmicas, como por exemplo, o método recursivo de Gauss-Seidel.

Para robôs que se movimentam no plano e com orientação livre, onde a configuração é dada por q = (x,y,θ), o potencial pode ser discretizado da seguinte forma, (considerando h = 1, para simplificar):

$$U_{x,y,\theta} =[U_{x,y-1,\theta} + U_{x,y+1,\theta} + U_{x-1,y,\theta} + U_{x+1,y,\theta} + U_{x,y,\theta-1} + U_{x,y,\theta+1}]/6$$

Se este robô estiver sujeito a restrição não holonômica, onde só pode executar uma rotação e/ou uma translação no plano (x,y) na direção em que está orientado, então, denominando v o eixo que define esta orientação, o potencial pode ser discretizado da seguinte forma:

$$U_{x,y,\theta} = [U_{v-1,\theta} + U_{v+1,\theta} + U_{v,\theta-1} + U_{v,\theta+1}]/4$$

.

Cálculo direto dos Potenciais Harmônicos

Para o caso planar, q=(x,y), a equação de Laplace discretizada, pode ser reescrita de modo a explicitar o potencial $U_{x,y}$ em função dos potenciais dos seus pontos vizinhos na grade. Considerando o incremento h = 1, para simplificar, temos:

$$U_{i,j} = [U_{i,j-1} + U_{i,j+1} + U_{i-1,j} + U_{i+1,j}]/4$$

Observa-se que o potencial no ponto (i,j) da grade é função do potencial dos seus vizinhos, cada um destes pode ser:

a) Ponto da região navegável, cujo potencial deve ser calculado de forma semelhante ao potencial $U_{i,j}$.
b) Ponto da região de contorno, onde o seu potencial U_C deve ser atribuído de acordo com a condição de contorno adotada.
c) Ponto de destino, onde seu potencial U_D deve ser fixado em um valor fixo mínimo.

Desta maneira, o calculo dos potenciais da grade pode ser formulado da seguinte forma:

$$\mathbf{U} = \mathbf{A}.\mathbf{U} + \mathbf{U_C} + \mathbf{U_D}$$

Onde todos os potenciais dos pontos da região navegável são arranjados em um vetor coluna **U** de dimensão n. $\mathbf{U_C}$ também é um vetor coluna de dimensão n, onde, para a k-ésima linha, o elemento $\mathbf{U_{Ck}}$ assume o valor da soma dos potenciais dos pontos vizinhos ao ponto k que são pontos da região de contorno. $\mathbf{U_D}$ também é um vetor coluna, cujos elementos são todos nulos, exceto o elemento que corresponde ao ponto de destino. Este elemento deve assumir o valor do potencial mínimo atribuído ao ponto de destino. A matriz **A** é uma matriz nxn, cuja diagonal é nula, uma vez que, de acordo com a formulação discreta adotada, o potencial $\mathbf{U}_{i,j}$ depende apenas dos seus pontos vizinhos e não dele mesmo. Os coeficientes de cada linha da matriz **A** devem ser estabelecidos de acordo com os potenciais livres que aparecem na formulação discreta $U_{i,j} = [U_{i,j-1} + U_{i,j+1} + U_{i-1,j} + U_{i+1,j}]/4$. Desta forma, os potenciais na grade podem ser obtidos diretamente resolvendo a expressão:

$$\mathbf{U} = \mathbf{A}.\mathbf{U} + \mathbf{U_C} + \mathbf{U_D}$$

$$\Rightarrow \mathbf{U} - \mathbf{A}.\mathbf{U} = \mathbf{U_C} + \mathbf{U_D}$$

$$\Rightarrow (\mathbf{I} - \mathbf{A})\mathbf{U} = \mathbf{U_C} + \mathbf{U_D}$$

$$\Rightarrow \mathbf{U} = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{U_C} + \mathbf{U_D})$$