

PLANEJAMENTO DE CAMINHOS

MÉTODOS BASEADOS EM MAPA DE ROTAS

Princípio: Capturar a conectividade de C_L na forma de uma rede de curvas unidimensionais \mathbf{R} , (**Mapa de Rotas**), que é usada como um conjunto de caminhos padrão. O planejamento se reduz a buscar um caminho em \mathbf{R} entre q_{ini} e q_{fin} . Exemplo: Grafo de Visibilidade, Diagrama de Voronoi, Rede de Caminhos Livres, Método da Silhueta, etc.

Planejamento baseado em Grafo de Visibilidade:

- Aplicação: $\mathbf{W} = \mathbf{R}^2$, robô \mathbf{A} poligonal e com orientação fixa, com obstáculos \mathbf{B}_i poligonais.
- Princípio: Construir um caminho semi-livre entre q_{ini} e q_{fin} formado por uma linha poligonal através dos vértices de \mathbf{CB} .

\Rightarrow Existe um caminho semi-livre entre q_{ini} e q_{fin} se e somente se existe uma linha poligonal simples, $\tau \in cl(\mathbf{C}_L)$, cujos pontos extremos são q_{ini} e q_{fin} e tal que seus vértices $\in \mathbf{CB}$.

Prova: Se existe um caminho semi-livre, existe um caminho de comprimento euclidiano mínimo, τ , que, para ser o mais curto, deve ser também localmente mínimo. Assim, qualquer sub-caminho τ' de τ deve ser o mais curto entre os seus extremos, portanto, em qualquer configuração q , se q não é vértice de \mathbf{CB} , τ deve ter curvatura zero \Rightarrow os vértices de τ são vértices de \mathbf{CB} .

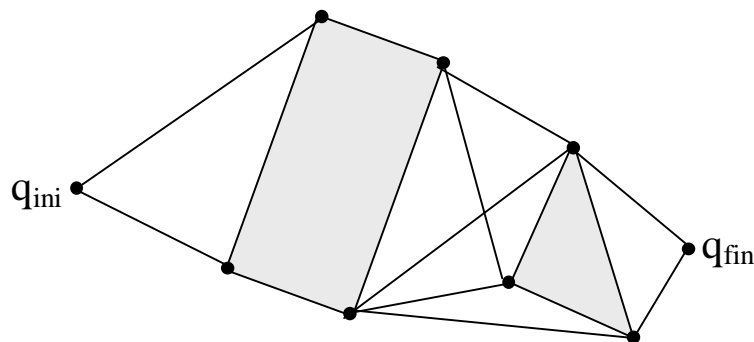
\Rightarrow Para achar um caminho livre entre q_{ini} e q_{fin} , é suficiente considerar o conjunto de linhas poligonais em $cl(\mathbf{C}_L)$ através dos vértices de \mathbf{CB} (Grafo de Visibilidade).

\Rightarrow O Grafo de Visibilidade contém o caminho mais curto, (em métrica euclidiana em \mathbf{R}^2), entre q_{ini} e q_{fin} .

Grafo de Visibilidade - Grafo não direcional, G :

- Os Nós de G são q_{ini} e q_{fin} e os vértices de CB .
- Dois nós de G são conexos por um arco se e somente se o segmento que os une é um eixo de CB ou está contido inteiramente em C_L , com possível exceção dos seus extremos.

Exemplo:



Algoritmo de Planejamento:

1. Construir o Grafo G .
2. Buscar um caminho em G de q_{ini} até q_{fin} .
3. Se o caminho é encontrado, retorná-lo, se não, reportar falha.

Construção do Grafo de Visibilidade:

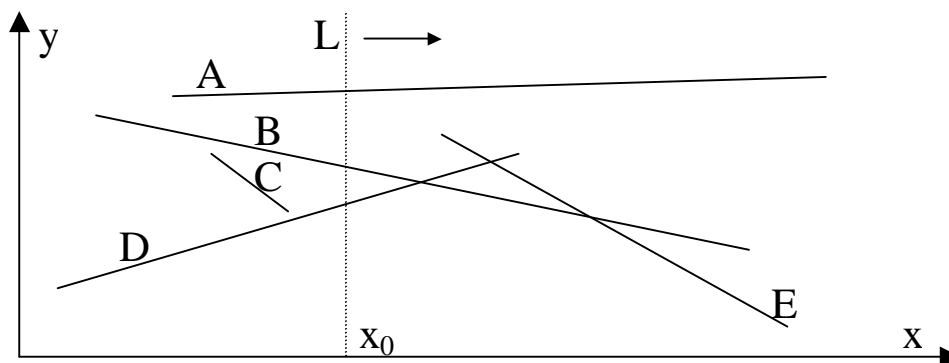
1. Tomar pares (X, X') de nós de G .
2. Se X e X' são extremos do mesmo eixo em CB , conectá-los por um arco em G .
3. Se X e X' não são extremos do mesmo eixo em CB , computar as interseções da linha suporte de (X, X') com CB . Caso não houver nenhuma interseção em (X, X') , ligar os nós por um arco em G .

Observação: complexidade computacional de ordem $O(n^3)$.

Algoritmo de Varredura de Linha - $O(n^2 \cdot \log(n))$:

1. Para cada ponto $X_i \in G$, computar a orientação α_{ij} da semi-reta com origem em X_i e passando por cada um dos outros pontos $X_j \in G$.
2. Ordenar as orientações α_{ij} .
3. Varrer as orientações α_{ij} no sentido anti-horário, de 0 a 2π .
4. Para cada α_{ij} , computar as interseções da semi-reta com CB.
5. Determinar o ponto de interseção P mais próximo a X_i .
6. Se P está contido no segmento X_iX_j , existe interseção, portanto não deve ser criado um arco entre os seus extremos; se não, o arco é criado.

Algoritmo de Varredura de Linha – Caso Geral:



Dadas as seguinte estruturas de dados:

- S = Lista Status de **segmentos ativos** na abscissa corrente ordenados pelas ordenadas das interseções da reta de varredura L com os segmentos ativos. Se o segmento X_1 ocorre antes de X_2 em S , X_1 está *Abaixo* de X_2 e X_2 está *Acima* de X_1 . Por exemplo, em $x = x_0$, $S = \{D, B, A\}$.
- E = Fila de eventos futuros, ordenados de acordo com as abscissas dos pontos extremos dos n segmentos ou dos c pontos de interseção.
- L = Lista ordenada por abscissas dos pontos de interseção.
- Q = Fila auxiliar usada internamente pelo algoritmo.

Dadas as seguinte operações sobre as estruturas de dados:

- $\text{INSERIR}(X, S)$: insere o segmento X em S .
- $\text{APAGAR}(X, S)$: apaga o segmento X de S .
- $\text{ACIMA}(X, S)$: retorna segmento acima de X em S , se existir.
- $\text{ABAIXO}(X, S)$: retorna segmento abaixo de X em S , se existir.
- $\text{TROCAR}(X_1, X_2, S)$: troca a ordem entre os segmentos consecutivos de S , X_1, X_2 .
- $\text{PRIMEIRO}(E)$: remove a menor abscissa de E , retornando-a.
- $\text{INSERIR}(x, E)$: insere a abscissa x em E .
- $\text{MEMBRO}(x, E)$: determina se a abscissa x é um membro de E .
- $\text{INSERIR}((X_1, X_2), Q)$: insere par de segmentos (X_1, X_2) em Q .
- $\text{INSERIR}(x, L)$: insere abscissa x em L .

Procedimento:

- A reta L varre os pontos críticos (x_c, y_c) , (extremos ou interseções de segmento), da esquerda para a direita.
- A menor abscissa (próximo evento) em E é removida a cada iteração e S é atualizada de acordo com o seu tipo:
 - Se x_c é abscissa do extremo esquerdo de um segmento X , X é adicionado a S .
 - Se x_c é abscissa do extremo direito de um segmento X , X é removido de S .
 - Se x_c é abscissa do ponto de interseção dos segmentos X_1 e X_2 , X_1 e X_2 são trocados em S .
- Checa-se a interseção entre todo par de segmentos que se torna adjacente em S .
- Toda abscissa removida de E que corresponde a um ponto de interseção novo é inserida em L .
- No final, L conterà a lista ordenada das abscissas de todos os pontos de interseção.
- Complexidade computacional: $O((n+c).\log(n))$.

Procedimento VARREDURA_DE_LINHA

começar

colocar as abscissas dos 2.n pontos extremos dos n segmentos em E .

$S \leftarrow \emptyset$; $L \leftarrow \emptyset$; $Q \leftarrow \emptyset$;

enquanto $E \neq \emptyset$, **faça**

começar

$x \leftarrow \text{PRIMEIRO}(E)$;

$p \leftarrow$ ponto extremo ou de interseção do qual x é abscissa;

se p é um ponto extremo esquerdo, **então**

começar

$X \leftarrow$ segmento do qual p é extremo;

$\text{INSERIR}(X, S)$;

$X_1 \leftarrow \text{ACIMA}(X, S)$;

$X_2 \leftarrow \text{ABAIXO}(X, S)$;

se X_1 intersecta X , **então** $\text{INSERIR}((X_1, X), Q)$;

se X_2 intersecta X , **então** $\text{INSERIR}((X, X_2), Q)$;

fim;

se não

se p é um ponto extremo direito, **então**

começar

$X \leftarrow$ segmento do qual p é extremo;

$X_1 \leftarrow \text{ACIMA}(X, S)$;

$X_2 \leftarrow \text{ABAIXO}(X, S)$;

se X_1 intersecta X_2 , **então** $\text{INSERIR}((X_1, X_2), Q)$;

$\text{APAGAR}(X, S)$;

fim;

se não /* p é ponto de interseção */

começar

$\text{INSERIR}(x, L)$;

$(X_1, X_2) \leftarrow$ par de segmentos se intersectando em p ;

/* com $X_1 = \text{ACIMA}(X_2)$ à esquerda de p */

$X_3 \leftarrow \text{ACIMA}(X_1, S)$;

$X_4 \leftarrow \text{ABAIXO}(X_2, S)$;

se X_3 intersecta X_2 , **então** $\text{INSERIR}((X_3, X_2), Q)$;

se X_1 intersecta X_4 , **então** $\text{INSERIR}((X_1, X_4), Q)$;

$\text{TROCAR}(X_1, X_2, S)$;

fim;

enquanto $Q \neq \emptyset$, **faça**

começar

$(X, X') \leftarrow \text{PRIMEIRO}(Q)$;

$x \leftarrow$ abscissa do ponto de interseção de X e X' ;

se $\neg \text{MEMBRO}(x, E)$, **então** $\text{INSERIR}(x, E)$;

fim;

fim;

fim;

Busca do Menor Caminho em G de q_{ini} até q_{fin} :

- Técnicas de busca em grafos. Exemplo: Algoritmo A^* , usando a distância euclidiana ao alvo como função heurística para guiar a busca.
- Grafo $G = (X, A)$, onde X é o conjunto de n nós e A é o conjunto de r arcos.
- G representado por listas de adjacências, uma para cada nó N (lista de todos os nós N' ligados por m arco a N).

Algoritmo A^* $O(r \cdot \log(n))$:

- Aplicável a grafos em que os arcos têm custos associados, por exemplo, distância euclidiana entre os nós.
- Custo de um caminho = soma dos custos dos seus arcos.
- Permite obter o menor caminho em termos da métrica adotada.

Procedimento:

- G explorado iterativamente seguindo caminhos a partir de N_{ini} .
- Para cada nó visitado, um ou mais caminhos são gerados a partir de N_{ini} , mas só o de menor custo é memorizado.
- O conjunto de caminhos gerados forma uma árvore T do subconjunto de G já explorado.
- T é representada através de ponteiros, associados a cada nó visitado, os quais apontam para os correspondentes nós pais.
- Para cada nó N , atribui-se uma função de custo, estimativa do custo do caminho de custo mínimo passando por N :

$$f(N) = g(N) + h(N)$$

- $g(N)$ = custo do caminho entre N_{ini} e N em T corrente.
- $h(N)$ = estimativa heurística do custo $h^*(N)$ do caminho de mínimo custo entre N e N_{fin} .

- $h(N)$ é admissível se e somente se: $\forall N \in G: 0 \leq h(N) \leq h^*(N)$.
- Se $h(N)$ é admissível, A^* garante encontrar o caminho de menor custo entre quaisquer dois nós conexos por G .
- Função heurística admissível trivial (Dijkstra): $h_1(N) = 0$.
- Função heurística admissível, aplicável quando os nós representam pontos em J^n : $\|N_{fin} - N\|$.
- h_2 é mais informada que $h_1 \Rightarrow \forall N \in G: h_1(N) \leq h_2(N)$.
- Se h_2 é mais informada que h_1 , todo nó expandido por A^* usando h_2 pode ser expandido por A^* usando h_1 .
- h é consistente localmente se: $0 \leq h(N') \leq h(N) + k(N, N')$, onde $k(N, N')$ é o custo associado ao arco entre N e N' .
- Se h é consistente localmente, é também admissível.
- h_1 e h_2 são consistentes localmente.

Dadas as seguinte estruturas de dados e funções:

- $G(X, A)$ = Grafo com n nós $\in X$ e r arcos $\in A$, com conectividade representada por listas de adjacências entre nós.
- T = Árvore do subconjunto de G já visitado.
- L = Lista que armazena nós de G ordenados por $f(N)$.
- $k : X \times X \rightarrow J^+$ função que especifica o custo de cada arco.
- $h(N)$: estimativa do custo do caminho mínimo entre N e N_{fin} .
- $g(N)$ = custo do caminho entre N_{ini} e N em T corrente.

Dadas as seguinte operações sobre a lista L :

- $PRIMEIRO(L)$: retorna o nó com menor valor de $f(N)$ em L e o remove da lista.
- $INSERIR(N, L)$: insere o nó N na lista L .
- $APAGAR(N, L)$: apaga o nó N da lista L .
- $MEMBRO(N, A)$: determina se o nó N é membro da lista L .
- $VAZIA(L)$: determina se a lista L está vazia.

Procedimento $A^*(G, N_{ini}, N_{fin}, k, h);$

começar

N_{ini} em T ;

INSERIR(N_{ini}, L); marcar N_{ini} como visitado;

enquanto $\neg VAZIA(L)$, **faça**

começar

$N \leftarrow PRIMEIRO(L);$

se $N = N_{fin}$, **então** sair do laço while;

para cada nó N' adjacente a N em G , **faça**

se N' é não visitado **então**

começar

adicionar N' a T com um ponteiro para N ;

INSERIR(N', L); marcar N' como visitado;

fim

se não, **se** $g(N') > g(N) + k(N, N')$ **então**

começar

redirecionar o ponteiro de N' para N em T ;

se MEMBRO(N', L), **então** APAGAR(N', L);

INSERIR(N', L);

fim

fim;

se $\neg VAZIA(L)$, **então**

retornar o caminho traçando os ponteiros de N_{fin} a N_{ini} ;

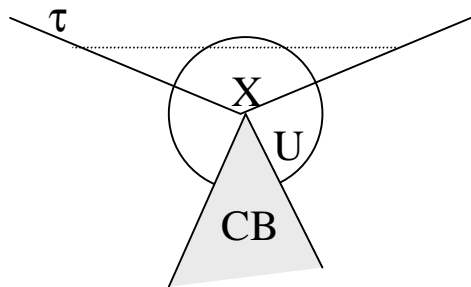
se não reportar falha;

fim;

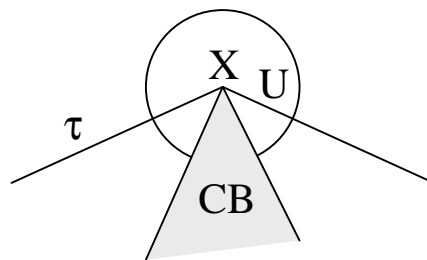
Grafo de Visibilidade Reduzido

- Setor Convexo no vértice X de um caminho poligonal τ é a região convexa fechada entre as duas semi-retas originadas em X e que contém os segmentos de τ adjacentes a X .

\Rightarrow para τ ser o caminho mais curto $\Leftrightarrow \forall X \in \tau$, X deve possuir uma vizinhança U tal que $(CB \cap U) \subset$ setor convexo de τ em X .

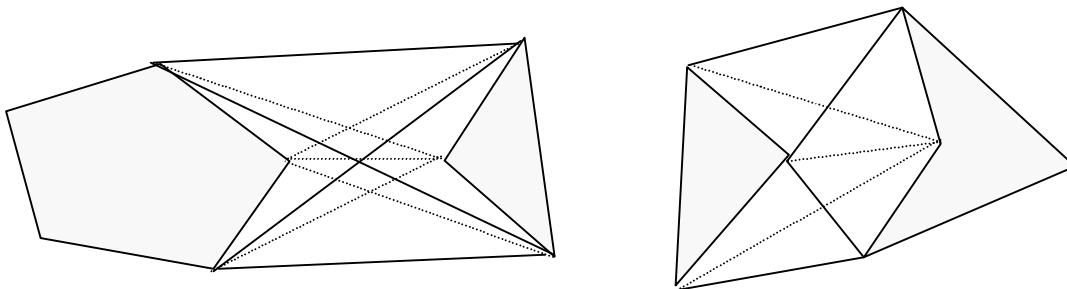


a) Pode ser minimizado.



b) Não pode ser minimizado.

- Uma reta L passando em um vértice X de CB é tangente a CB em X se e somente se o interior de CB está inteiramente em um único lado de L para uma vizinhança U de X .
- Seja L uma reta passando por dois nós X e X' de G . O segmento de L , XX' é um Segmento Tangente se e somente se:
 - se X é vértice de CB , então L é tangente a CB em X ;
 - se X' é vértice de CB , então L é tangente a CB em X' .
- Segmentos tangentes possuem as seguintes propriedades:
 - Entre dois polígonos convexos disjuntos existem exatamente quatro segmentos tangentes: dois são suportes (os dois polígonos ficam do mesmo lado), os outros dois são separadores (os dois polígonos ficam em lados opostos).
 - Se o vértice X é côncavo, (ângulo interno em X maior do que π), nenhum segmento tangente termina em X .



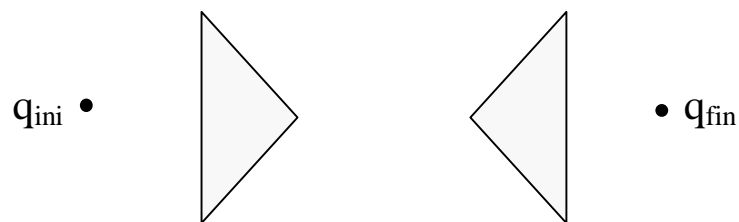
⇒ Alguns arcos do grafo de visibilidade não são necessários.

Procedimento de construção do Grafo de Visibilidade Reduzido:

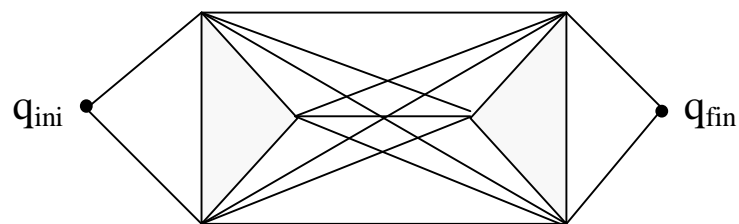
- Se o segmento entre X e X' não é segmento tangente, não precisa ser incluído no grafo.

Observação: qualquer caminho τ que inclui um segmento não tangente entre X e X' é tal que não está situado localmente no setor convexo de τ em X e/ou de X' .

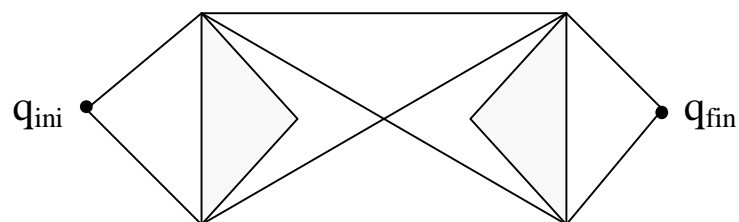
- O subgrafo G' obtido pela remoção de todos os segmentos não tangentes é denominado de Grafo de Visibilidade Reduzido.
- Se existe um caminho semi-livre entre q_{ini} e q_{fin} , G' contém o menor caminho em $cl(C_L)$ entre estas duas configurações.



a) Espaço de Trabalho.



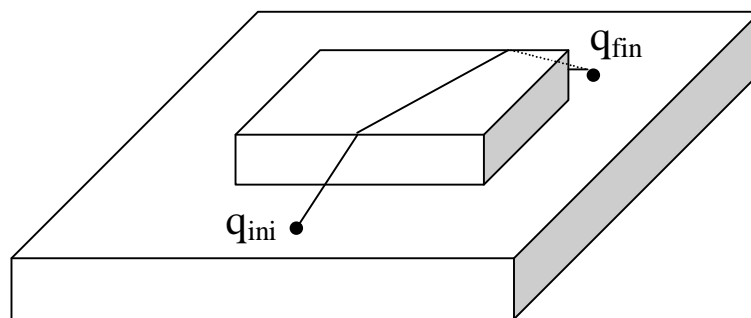
b) Grafo de Visibilidade.



c) Grafo de Visibilidade Reduzido.

Grafo de Visibilidade para espaços de dimensão maior:

- O Problema de gerar o menor caminho semi-livre em um espaço $C = \mathbf{R}^3$ povoado de C-obstáculos poliédricos é NP.
- Para esta situação, o método do grafo de visibilidade pode ser aplicado, mas sem garantir o caminho mínimo.
- Somente se pode garantir que o caminho mínimo será uma linha poligonal cujos vértices estão contidos nos eixos dos C-Obstáculos.
- Caminhos mínimos podem ser obtidos adicionando vértices fictícios nos eixos dos poliedros.



- O método Grafo de Visibilidade não é estendido diretamente ao caso em que o robô é um polígono capaz de transladar-se e girar em um espaço povoado de obstáculos poligonais.
- Neste caso, uma possível maneira de tratar a rotação consiste em "fatiar" o espaço de configuração em um número finito p de intervalos $[\theta_k, \theta_{k+1}]$ e computar a projeção CB_k de CB dentro deste intervalo em \mathbb{J}^2 .
- CB_k é um polígono generalizado, ao qual pode ser associado um grafo de visibilidade G_k .
- Os p grafos G_i s podem ser combinados em um grafo de visibilidade global ligando por um arco cada nó $X \in G_k$ a cada nó $X' \in G_{k+1}$, ($k=1, \dots, p \bmod p$), sempre que o segmento de aberto ligando as duas configurações não intersecta CB_k nem CB_{k+1} . Este método não completo: pode falhar em achar um caminho, mesmo se existir um.

Planejamento baseado em Retração:

- Princípio: Definir um mapeamento contínuo de C_L numa rede de curvas unidimensionais \mathbf{R} contida em C_L , a qual preserva a conectividade do mesmo.
- Seja \mathbf{X} um espaço topológico e \mathbf{Y} um subconjunto de \mathbf{X} . o mapeamento sobrejetor $\rho: \mathbf{X} \rightarrow \mathbf{Y}$ é chamado de Retração de \mathbf{X} em $\mathbf{Y} \Leftrightarrow \rho$ é contínuo e sua restrição a \mathbf{Y} é o mapeamento identidade.
- Seja ρ a retração de \mathbf{X} em \mathbf{Y} . ρ preserva a conectividade de $\mathbf{X} \Leftrightarrow \forall x \in \mathbf{X}$, x e $\rho(x)$ pertencem à mesma componente conexa de \mathbf{X} , ou seja, existe caminho entre x e $\rho(x)$.
- Proposição: Seja $\rho: C_L \rightarrow \mathbf{R}$ uma retração preservadora da conectividade de C_L na rede de curvas unidimensionais $\mathbf{R} \subset C_L$. Existe um caminho livre entre q_{ini} e q_{fin} \Leftrightarrow existe um caminho em \mathbf{R} entre $\rho(q_{ini})$ e $\rho(q_{fin})$.

Prova: seja $\tau: [0, 1] \rightarrow C_L$ um caminho livre entre q_{ini} e q_{fin} . O mapeamento $\rho \circ \tau: [0, 1] \rightarrow \mathbf{R}$ é um caminho livre entre $\rho(q_{ini})$ e $\rho(q_{fin})$, visto que ρ é contínuo. Por outro lado, se existe um caminho em \mathbf{R} entre $\rho(q_{ini})$ e $\rho(q_{fin})$, então, q_{ini} e q_{fin} são conexas por um caminho livre que é a composição de três caminhos: um caminho livre de q_{ini} a $\rho(q_{ini})$, um caminho em \mathbf{R} entre $\rho(q_{ini})$ e $\rho(q_{fin})$ e um caminho livre de $\rho(q_{fin})$ a q_{fin} . O primeiro e o terceiro caminho existem, visto que ρ preserva a conectividade de C_L .

- Método de planejamento baseado em retração:
 - Escolher o mapeamento ρ e construir \mathbf{R} .
 - Construir um grafo \mathbf{G} que represente \mathbf{R} .
 - Determinar os mapeamentos $\rho(q_{ini})$ e $\rho(q_{fin})$.
 - Gerar um caminho livre entre q_{ini} e $\rho(q_{ini})$.
 - Buscar em \mathbf{G} um caminho entre $\rho(q_{ini})$ e $\rho(q_{fin})$.
 - Gerar um caminho livre entre $\rho(q_{fin})$ e q_{fin} .

Planejamento baseado em Diagrama de Voronoi:

- Aplicável a $C = \mathbf{R}^2$ e C_L = interior de uma região poligonal limitada.
- Propriedade: maximiza a distância do robô aos obstáculos.

Definições:

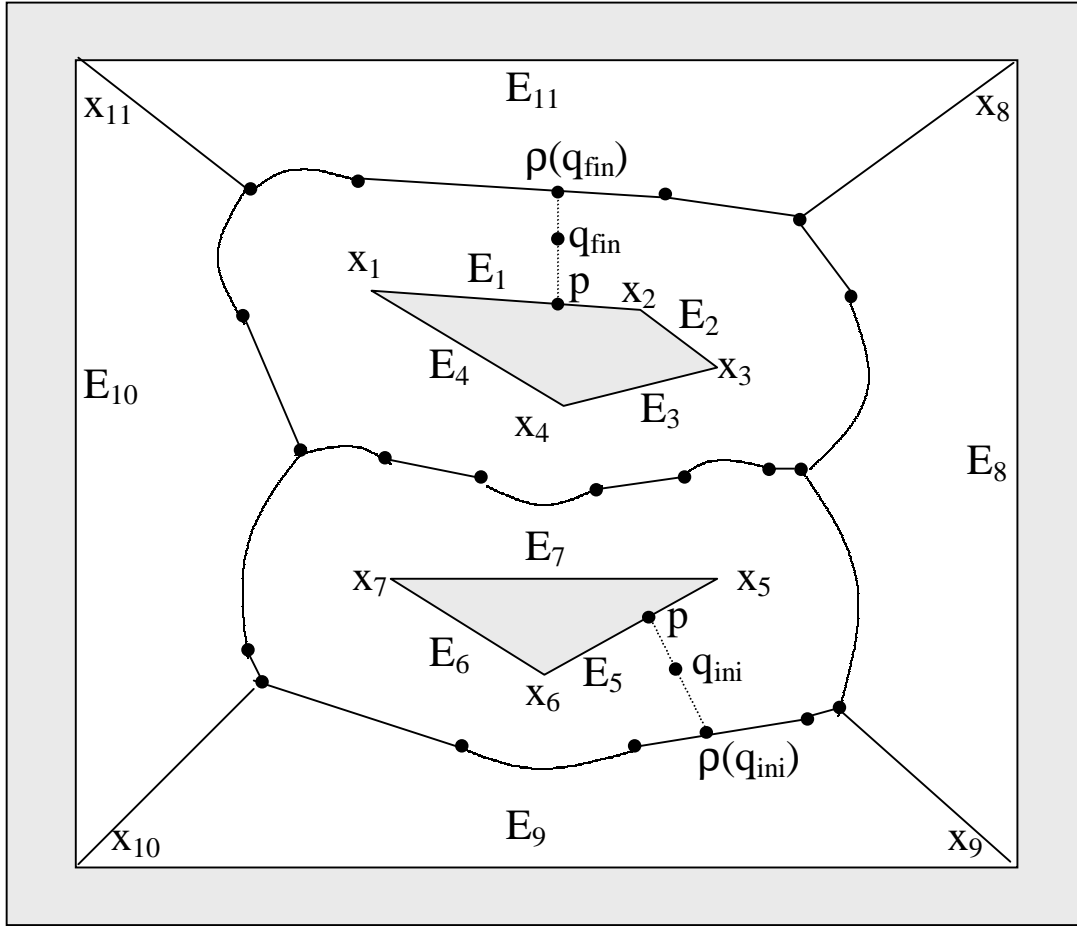
- Seja $\beta = \partial C_L$. $\forall q \in C_L$, Claridade de q = $\text{clar}(q) = \min_{p \in \beta} \|q-p\|$
- Perto de q = $\text{perto}(q) = \{p \in \beta / \|q-p\| = \text{clar}(q)\}$
- Diagrama de Voronoi de C_L :

$$\text{Vor}(C_L) = \{q \in C_L / \text{card}(\text{perto}(q)) > 1\}$$

Onde $\text{card}(X)$ retorna a cardinalidade do conjunto X .

\Rightarrow O Diagrama de Voronoi é o mapa de rotas **R** usado em método de planejamento baseado em retração.

- $\text{Vor}(C_L)$ = conjunto de segmentos de reta e parabólicos:
 - Segmentos de reta entre (eixo, eixo) ou (vértice, vértice).
 - Segmentos parabólicos entre (eixo, vértice).
- \Rightarrow Estes segmentos e os eixos de ∂C_L delimitam regiões de C_L para as quais $\text{card}(\text{perto}(q)) = 1$.
- Dados $q \notin \text{Vor}(C_L)$ e $p \in \partial C_L$, tal que $\|q-p\| = \text{clar}(q)$. Considere a semi-reta L com origem em p e passando por q . O segmento de reta conectando p à interseção mais próxima de L com $\text{Vor}(C_L)$, $\rho(q)$, (imagem de q em $\text{Vor}(C_L)$), segue o gradiente positivo de $\text{clar}(q)$. Além de $\rho(q)$, o gradiente muda de sinal. Para $q = q_{\text{ini}}$ (ou $q = q_{\text{fin}}$), este procedimento permite obter as suas imagens em $\text{Vor}(C_L)$, $\rho(q_{\text{ini}})$ (ou $\rho(q_{\text{fin}})$).



Dado que:

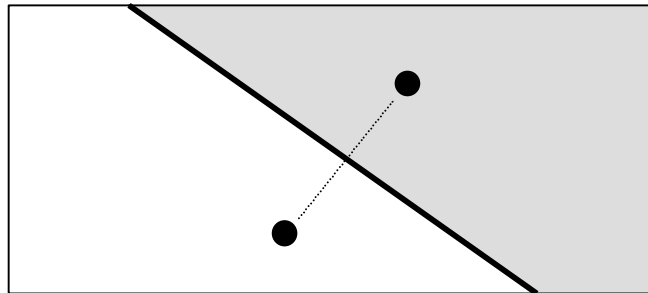
- $\forall q \in \text{Vor}(C_L) \Rightarrow \rho(q) = q$ (mapeamento identidade)
 - $\rho : C_L \rightarrow \text{Vor}(C_L)$ é mapeamento contínuo.
- $\Rightarrow \rho$ é uma retração preservadora da conectividade.

Planificação baseada no mapa de rotas definido por $\text{Vor}(C_L)$:

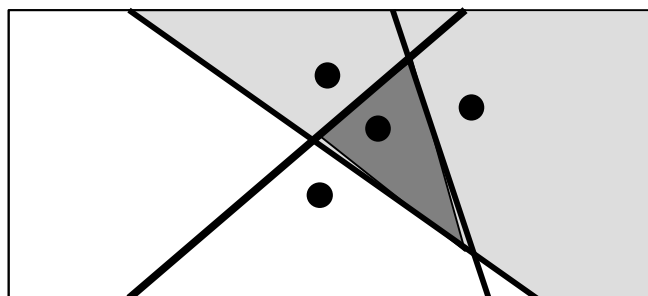
1. Computar $\text{Vor}(C_L) \Rightarrow$ uma vez computado $\text{Vor}(C_L)$ pode ser utilizado para quaisquer q_{ini} e q_{fin} .
2. Computar $\rho(q_{\text{ini}})$ e $\rho(q_{\text{fin}}) \rightarrow$ identificar os arcos de $\text{Vor}(C_L)$ contendo estes pontos.
3. Buscar uma seqüência de arcos A_1, \dots, A_p em $\text{Vor}(C_L)$, tal que $\rho(q_{\text{ini}}) \in A_1$ e $\rho(q_{\text{fin}}) \in A_p$ e, para todo $i \in [1, p-1]$, A_i e A_{i+1} compartilham uma extremidade.
4. Se a busca tem sucesso, retornar $\rho(q_{\text{ini}})$, $\rho(q_{\text{fin}})$ e a seqüência de arcos A_1, \dots, A_p , se não, reportar falha.

Diagrama de Voronoi de um conjunto de pontos (Sítios):

- Aplicável em um espaço de trabalho bidimensional povoado de obstáculos pontuais.
- Um conjunto de n sítios $P = \{p_1, p_2, \dots, p_n\}$ no plano gera um diagrama de Voronoi associado.
- Célula de Voronoi $V(p_i)$ de p_i é o conjunto de pontos que estão mais próximos de p_i do que de qualquer outro sítio de P .
- Cada par (p_i, p_j) pode ser separado por uma reta eqüidistante aos mesmos e perpendicular ao segmento que os une.
- A reta divide o plano em dois semiplanos, cada um contendo um dos pontos do par.

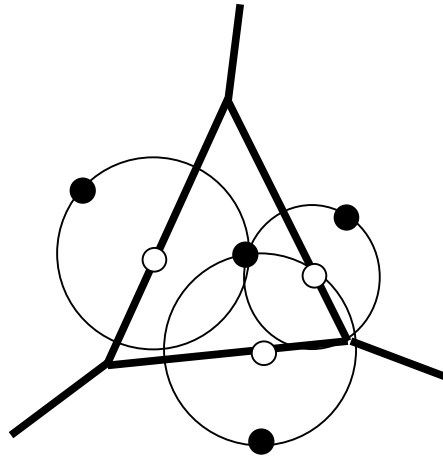


- A célula de Voronoi associada a um ponto p_i é definida pela interseção dos semiplanos que contêm p_i , gerados por todos os pares de pontos que incluem p_i .

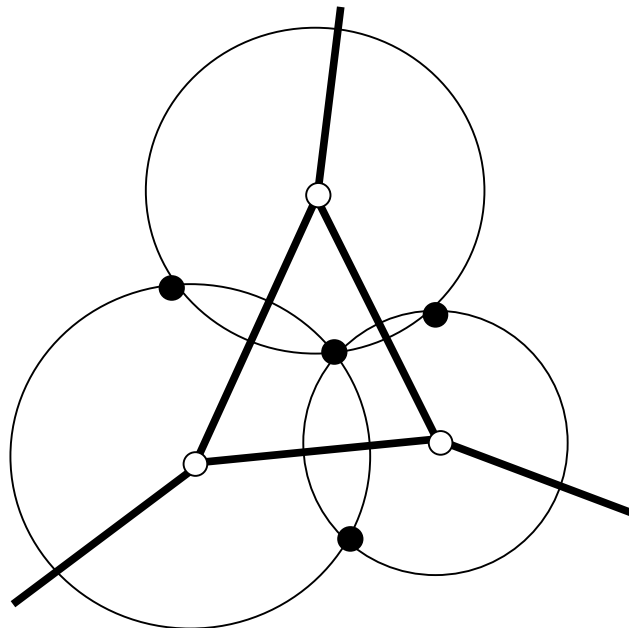


- A interseção dos semiplanos gera uma célula poligonal convexa (eventualmente ilimitada).
- O Diagrama de Voronoi $\text{Vor}(P)$ é definido pelos limites das células de Voronoi.

- Cada par de sítios (p_i, p_j) pertencentes a células adjacentes é separado por um segmento de reta denominado **Arestas de Voronoi**.
- Cada ponto pertencente à Aresta de Voronoi é centro de um círculo passando por p_i e por p_j .



- Cada tripla de sítios (p_i, p_j, p_k) pertencentes a células adjacentes entre si duas a duas define três arestas que se intersectam em um único ponto denominado **Vértice de Voronoi**.
- O Vértice de Voronoi é o centro de um círculo que passa pelos três sítios.



- Algoritmo Trivial para cômputo do diagrama de Voronoi:
 1. Para cada tripla de sítios (p_i, p_j, p_k) em P computar o ortocentro (o ponto equidistante aos três sítios).
 2. Para cada ortocentro x , checar todos os pontos em P de modo a verificar se (p_i, p_j, p_k) são os mais próximos de x .
 - Se existe algum sítio mais próximo de x do que (p_i, p_j, p_k) , rejeitar o ortocentro x .
 3. Para cada ortocentro x remanescente gerado por uma tripla de sítios (p_i, p_j, p_k) , buscar em P os ortocentros y_{ij}, y_{jk}, y_{ki} que compartilham com x os pares de sítios geradores (p_i, p_j) , (p_j, p_k) e (p_k, p_i) , respectivamente.
 - Para cada ortocentro y encontrado associado a um ortocentro x definir a Aresta de Voronoi cujos extremos são x e y .
 - Caso não exista um ortocentro y associado a x para um dado par de seus sítios geradores (p_i, p_j) , definir uma Aresta de Voronoi infinita constituída pela semi-reta com origem em x e perpendicular ao segmento entre p_i e p_j .

Exemplo:

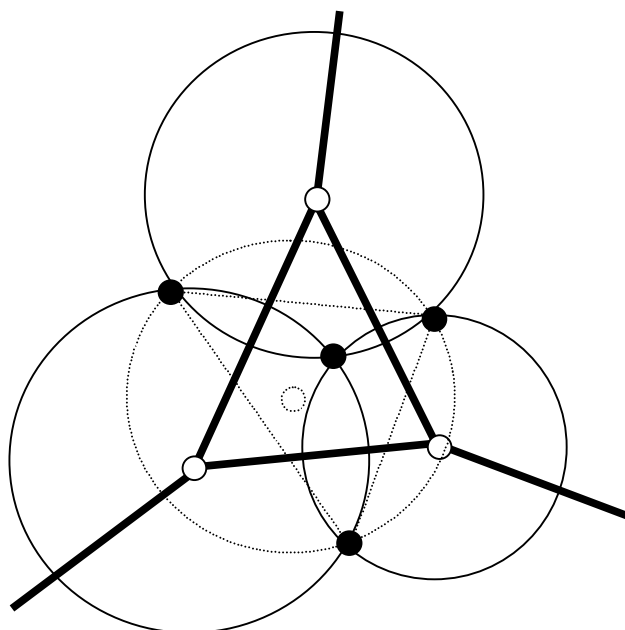


Diagrama de Voronoi de um conjunto de Polígonos:

- Aplicável em um espaço bidimensional povoado de obstáculos poligonais.
- **Algoritmo Aproximado:**
 1. Aproximar os limites dos obstáculos poligonais com um grande número de pontos, resultantes da subdivisão dos lados dos polígonos em pequenos segmentos.
 2. Computar o Diagrama de Voronoi desta coleção de pontos.
 3. Eliminar do Diagrama de Voronoi resultante as arestas que possuem ao menos um dos seus extremos no interior de qualquer obstáculo.
 4. Conectar as configurações inicial e final do robô ao diagrama através de segmentos de reta entre as mesmas e os Vértices de Voronoi mais próximos correspondentes. (Verificar eventuais interseções destes segmentos com os obstáculos).

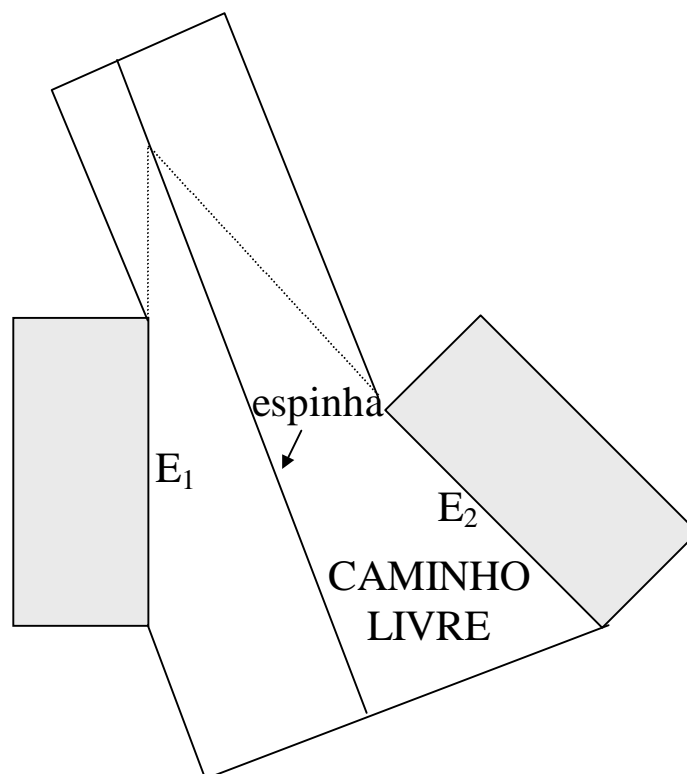
Algoritmo Exato (Trivial):

1. Para cada par (eixo,eixo), (vértice,vértice) e (eixo,vértice) da região de C-Obstáculos, computar as curva equidistante correspondente (reta ou parábola).
2. Computar as interseções entre estas curvas.
3. Considerar apenas os segmentos de curvas cujas extremidades são definidas pelas interseções mais próximas dos seus elementos geradores (eixos ou vértices).

Observação: algoritmos ótimos baseados em varredura do plano, (derivados do algoritmo de Steven Fortune) computam o Diagrama de Voronoi com complexidade $O(n.\log(n))$

Planejamento baseado em Rede de Caminhos Livres:

- Aplicável a objeto poligonal transladando-se e girando em um espaço de trabalho poligonal bidimensional.
- Busca manter o robô afastado dos obstáculos.
- Baseado em Caminhos Livres (*freeways*), extraídos do espaço de trabalho e conectados em um grafo, denominado Rede de Caminhos Livres.
- O eixo central do Caminho Livre é chamado de Espinha, a qual é anotada com uma descrição conservativa das orientações livres que o robô assume ao mover-se ao longo da mesma.



- O robô **A** pode mover-se ao longo de um caminho livre se existir um conjunto conexo de orientações de **A** ao longo da sua espinha \Rightarrow Descrição da conectividade de C_L é incompleta.
 - **A** pode passar de um caminho livre para outro, desde que os dois se intersectem e as faixas de orientações livres ao longo de ambas espinhas tenham interseção não vazia.
- \Rightarrow A rede é uma representação dos possíveis movimentos ao longo de espinhas e entre espinhas.

Extração de Caminhos Livres:

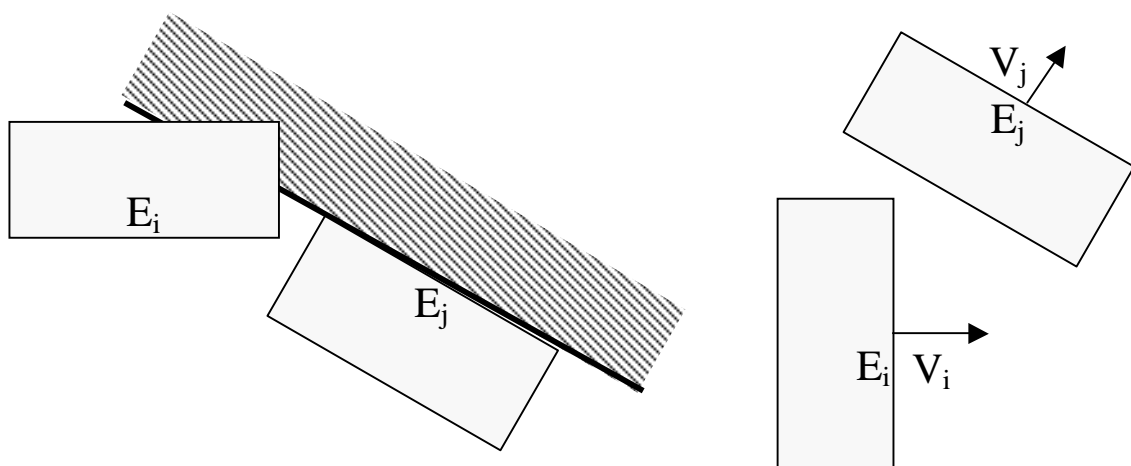
Definição:

- Cilindro generalizado linear reto bidimensional é uma região de \mathbf{R}^2 obtida pela varredura de um segmento de reta (a seção transversal) perpendicular e ao longo de uma linha reta (a espinha), sobre a qual se define uma origem e uma orientação.
- A seção transversal é dividida pela espinha em dois segmentos: as seções transversais direita e esquerda, cujos comprimentos são independentes, funções lineares por partes da abscissa ao longo da espinha.
- As linhas traçadas pelas extremidades da seção transversal são os lados direito e esquerdo do cilindro generalizado (CG).

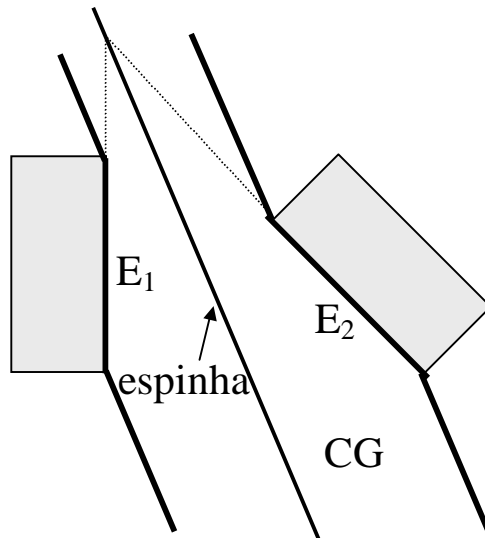
Caminhos livres são extraídos de $W_L = W \setminus B$, considerando pares de eixos de B , (E_i, E_j) , desde que estes "se encarem", ou seja:

1. Ambas extremidades de E_j devem estar no semi-plano externo de E_i (e vice-versa).
2. O produto interno das suas normais externas deve ser negativo.

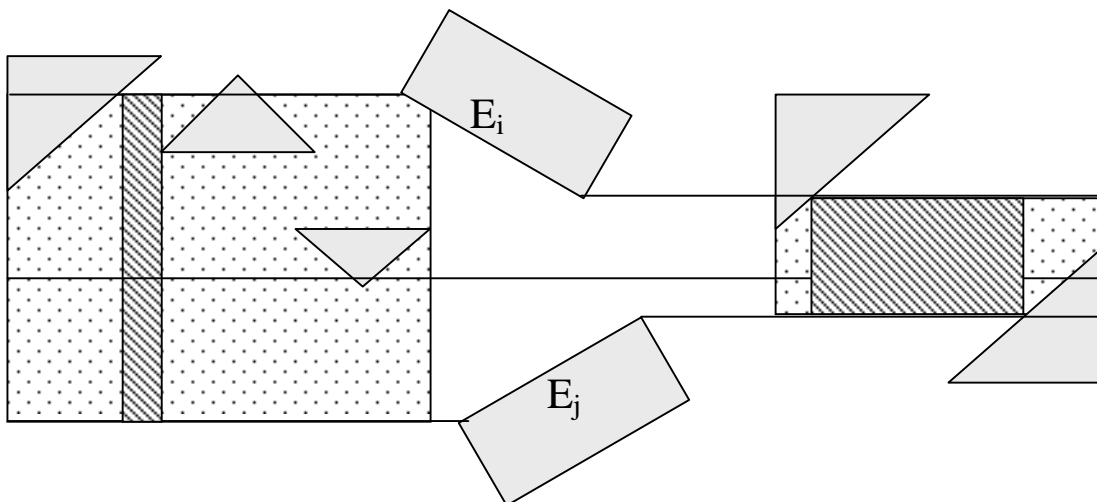
Exemplo de condições não satisfeitas:



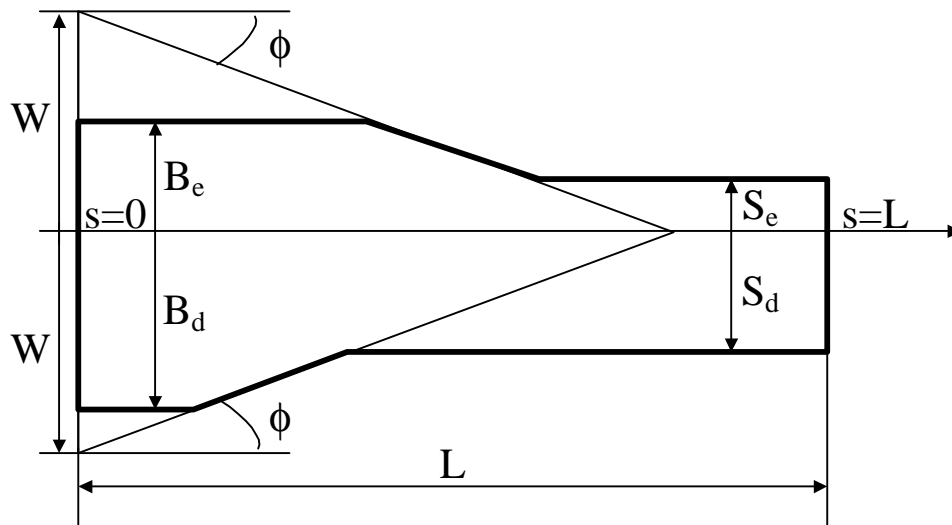
- A espinha do cilindro é a reta bissetriz do ângulo formado pelas retas suportes de E_i e E_j .
- Cada lado do cilindro é o eixo correspondente, estendido em ambas extremidades por semi-retas paralelas à espinha.



- As regiões de interseção do CG com \mathbf{B} são projetadas na espinha e as fatias correspondentes do CG são eliminadas. Assim, CG é fatiado em um ou vários CG' s truncados.
- Eliminam-se os CG truncados cujos lados não incluem porções de ambos E_i e E_j . A região resultante é um Caminho Livre.



Parâmetros que definem um caminho livre:



Determinação das orientações livres:

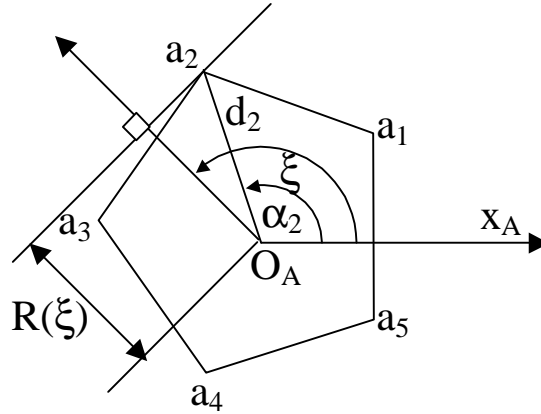
- o movimento do robô ao longo da espinha é a projeção, no plano xy, do seu movimento ao longo de um caminho em C restrito a uma seção transversal (plano) de C perpendicular ao plano xy.
- ⇒ Determinar a faixa de orientações livres é equivalente a computar a seção transversal à região de CB correspondente a um plano passando pela espinha e perpendicular ao plano xy.
- Para maximizar a faixa de orientações livres ao longo da espinha, O_A deve ser localizada de modo a minimizar a máxima distância de O_A a ∂A . Isto é conseguido colocando O_A no centro do menor círculo que contém A.

Semi-reta radial de ângulo ξ , $SR(\xi)$: semi-reta com origem em O_A , formando um ângulo ξ com o eixo x_A .

Função Raio, $R(\xi)$: ínfima das distâncias de O_A para as retas normais à $SR(\xi)$ e que não intersectam o robô \mathbf{A} .

$$\Rightarrow R(\xi) = \max\{d_i \cos(\xi - \alpha_i)\}$$

onde $i \in [1, n_A]$, d_i é a distância de O_A ao vértice a_i e α_i é o ângulo da semi-reta radial passando por a_i .



A imagem inversa do intervalo $[0, r]$ gerado por $R(\xi)$ é a função:

$$R^{-1}(r) = \{\xi \in [0, 2\pi) / R(\xi) < r\} = [0, 2\pi) \setminus \cup I_i$$

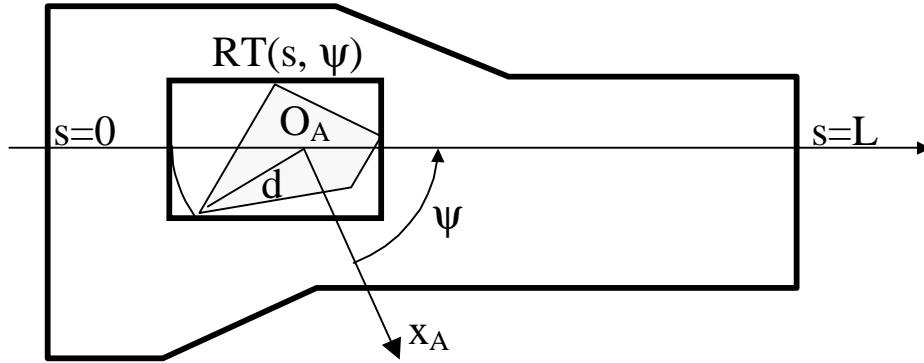
onde,

$$I_i = \emptyset, \quad \text{se } r > d_i.$$

$$I_i = [\alpha_i - |\arccos(r/d_i)|, \alpha_i + |\arccos(r/d_i)|] \bmod 2\pi, \text{ se } r \leq d_i.$$

\Rightarrow Garante-se que uma linha perpendicular a $SR(\xi)$ a uma distância de O_A maior do que r , com $\xi \in R^{-1}(r)$, não intersecta \mathbf{A} .

Retângulo Suporte de \mathbf{A} , $RT(s, \psi)$:



- Os lados direito e esquerdo de $RT(s, \psi)$ são paralelos à espinha, distantes $R(\psi - \pi/2)$ e $R(\psi + \pi/2)$ de O_A , respectivamente.
- O lado frontal situa-se a uma distância $R(\psi)$ de O_A .
- O lado traseiro situa-se a uma distância $d = \max_i \{d_i\}$
- O lado traseiro é uma escolha conservativa, independente de ψ .
- $RT(s, \psi)$ depende da orientação de \mathbf{A} em relação à espinha.

Definindo:

- $F_\Phi(s)$, com $0 \leq s \leq L$, conjunto de orientações ψ tal que $RT(s, \psi)$ permanece no interior do caminho livre Φ .
- $\Rightarrow F_\Phi(s)$ é um subconjunto das orientações livres de \mathbf{A} quando O_A se move sobre a espinha de Φ .

Para $0 \leq s \leq d$: $F_\Phi(s) = \emptyset$

Para $d < s \leq L$: $F_\Phi(s) = R^{-1}(L-s) \cap [(R^{-1}(S_d) + \pi/2) \cap (R^{-1}(S_e) - \pi/2)] \cup [(R^{-1}(B_d) + \pi/2) \cap (R^{-1}(B_e) - \pi/2)] \cap R^{-1}_{-\pi/2, \alpha}(\alpha \cdot s + W) \cap R^{-1}_{\pi/2, \alpha}(\alpha \cdot s + W)]$

onde, $\alpha = -\tan(\phi)$
 $R_{\xi, \alpha} = R(\psi + \xi) - \alpha R(\psi)$
 $R^{-1}_{\xi, \alpha}(u) = \{\psi / R_{\xi, \alpha}(\psi) < u\}$

Observações:

- Caso particular - Φ construído a partir de dois eixos paralelos:

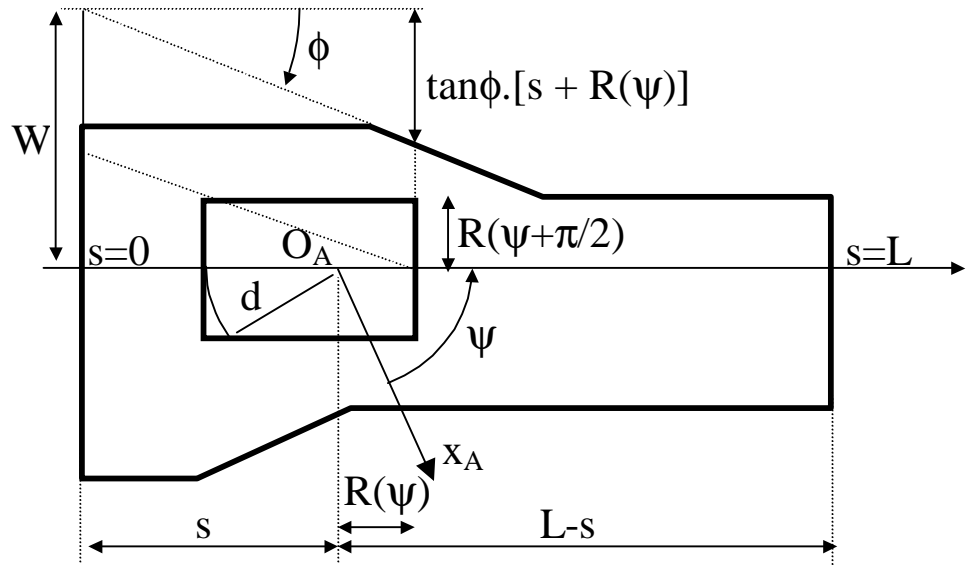
Para $0 \leq s \leq d$: $F_\Phi(s) = \emptyset$

Para $d < s \leq L$: $F_\Phi(s) = R^{-1}(L-s) \cap [(R^{-1}(W)+\pi/2) \cap (R^{-1}(W)-\pi/2)]$

- Para $0 \leq s \leq d$, a parte traseira do retângulo ultrapassa o lado esquerdo de Φ .
- Para $d < s \leq L$, $F_\Phi(s)$ representa uma conjunção de restrições na orientação devido aos outros lados do caminho livre Φ .
- O termo $R^{-1}(L-s)$ expressa a restrição na a abscissa da frente do retângulo, a qual deve ser menor do que L .
- O termo $[(R^{-1}(S_d)+\pi/2) \cap (R^{-1}(S_e)-\pi/2)]$ representa restrições de orientação que garantem que o retângulo não ultrapasse os lados direito e esquerdo de Φ (a distâncias S_d e S_e da espinha).
- As restrições acima são muito conservativas. Limitam as orientações válidas àquelas que mantêm o retângulo suporte sobre a espinha e dentro do retângulo de comprimento L e largura $S_d + S_e$, o qual é um subconjunto de Φ .
- O termo $(R^{-1}(B_d)+\pi/2) \cap (R^{-1}(B_e)-\pi/2)$ expressa que a distância da espinha aos lados direito e esquerdo do retângulo não deve ser superior a B_d e B_e respectivamente.
- O termo $R^{-1}_{-\pi/2,\alpha}(\alpha.s+W) \cap R^{-1}_{\pi/2,\alpha}(\alpha.s+W)$ incorpora as restrições na orientação devido aos lados não paralelos, que, em termos de distâncias euclidianas, são expressas como:

$$R(\psi - \pi/2) < W - \tan(\phi).[s + R(\psi)]$$

$$R(\psi + \pi/2) < W - \tan(\phi).[s + R(\psi)]$$



- O cálculo de $R_{\xi, \alpha}^{-1}(u)$, onde $R_{\xi, \alpha} = R(\psi + \xi) - \alpha R(\psi)$, envolve a soma de duas funções raio. Dados:

$$R(\xi) = \max\{d_i \cos(\xi - \alpha_i)\} \quad R'(\xi) = \max\{d_j' \cos(\xi - \alpha_j')\}$$

$$\Rightarrow R(\xi) + R'(\xi) = \max\{d_i \cos(\xi - \alpha_i) + d_j' \cos(\xi - \alpha_j')\}$$

Que pode ser rescrito como:

$$R(\xi) + R'(\xi) = \max\{e_{ij} \cdot \cos(\xi - v_{ij})\}$$

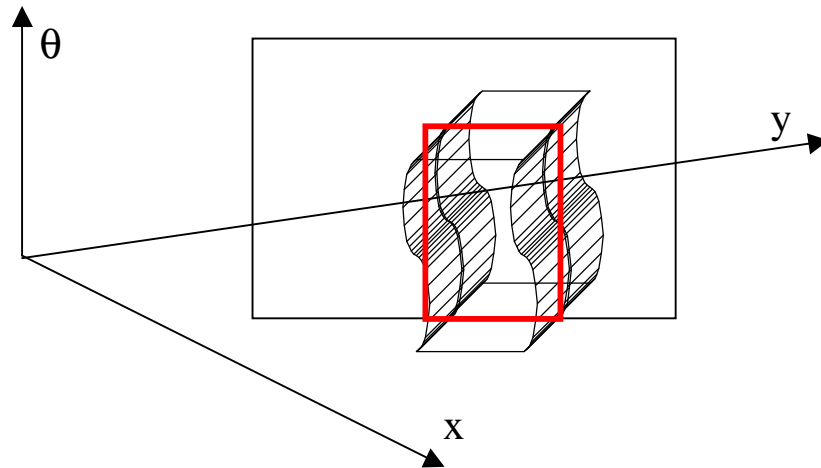
onde $i, j \in [1, n_A]$. Os termos e_{ij} e v_{ij} dependem de d_i , α_i , d_j' e α_j' .

- Assim, $F_\Phi(s)$ é descrito como um conjunto de intervalos conexos máximos.
- Como os raios direito e esquerdo de Φ são funções não crescentes da abscissa s da espinha,

$$\forall s_1, s_2 \in [d, L] : s_1 \leq s_2 \Rightarrow F_\Phi(s_2) \subseteq F_\Phi(s_1)$$

Rede de Caminhos Livres:

- Geometricamente, para cada caminho livre, $F_{\Phi}(s)$ representa o complemento de uma aproximação (por limites superiores) da seção transversal da região de C-obstáculos, contida em um plano que contém a espinha e é perpendicular ao plano xy.



- A partir da representação parcial de CB através de vários caminhos livres, um grafo G não direcionado é construído
- O grafo G é chamado Rede de Caminhos Livres.
- O grafo define todos os possíveis movimentos do robô ao longo das espinhas dos caminhos livres.

Dado X = conjunto de todos os pontos P que sejam interseção de duas espinhas de dois caminhos livres.

Procedimento para criar os nós de G :

Para todo caminho livre Φ **fazer**

Para todo ponto $P \in X$ na espinha de Φ **fazer**

- atribuir s = abscissa de P na espinha;
- criar um nó de G para cada intervalo em $F_{\Phi}(s)$.

Observações:

- Cada intervalo associado a um nó de G é definido em relação a sua própria espinha. \Rightarrow deve ser reparametrizado para um sistemas de coordenadas comum (por exemplo F_w).
- Por simplificação, assumimos que as posições inicial e final estão sobre espinhas de Φ_{ini} e Φ_{fin} , dentro dos intervalos $F_{\Phi_{ini}}(s_{ini})$ e $F_{\Phi_{fin}}(s_{fin})$.

Procedimento para criar os arcos de G :

1. Sejam N_1 e N_2 dois nós de G correspondentes a dois pontos com abscissas s_1 e s_2 na espinha de um mesmo caminho livre, com intervalos de orientações livres I_1 e I_2 , respectivamente. Se $I_1 \cap I_2 \neq \emptyset$, conectar N_1 e N_2 com um arco de G .
 - O robô pode se movimentar livremente entre s_1 e s_2 a qualquer orientação dentro de $I_1 \cap I_2$.
2. Sejam N_1 e N_2 dois nós de G correspondentes a um mesmo ponto P no plano xy pertencente a dois caminhos livres diferentes. sejam I_1 e I_2 os intervalos de orientações livres associados a N_1 e N_2 , respectivamente. Se $I_1 \cap I_2 \neq \emptyset$, conectar N_1 e N_2 com um arco de G .
 - O robô pode se transferir livremente, em P , da espinha de um caminho livre para a espinha do outro em qualquer orientação dentro de $I_1 \cap I_2$.

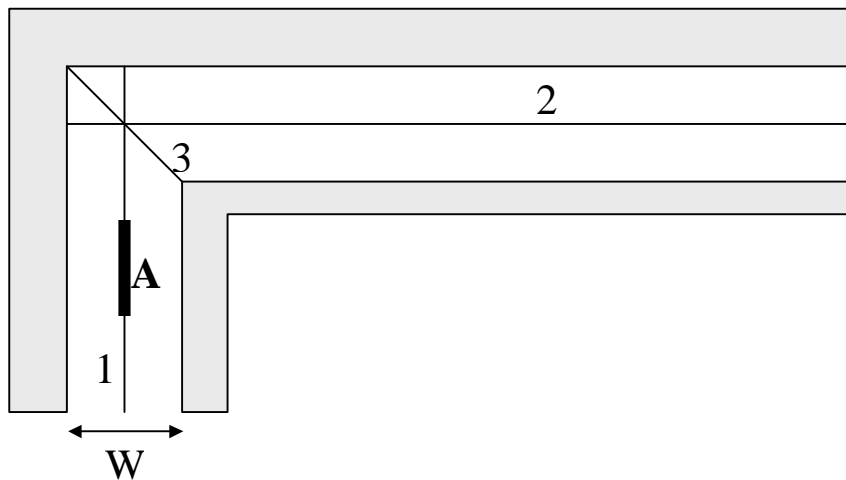
Observações:

- Para construir G , é suficiente calcular $F_{\Phi}(s)$ em um número finito de abscissas ao longo da espinha de cada caminho livre.
- A rede de caminhos livres é analisada para buscar um caminho entre o nó inicial e o nó final.

Limitações da Rede de Caminhos Livres:

- A rede incorpora uma representação conservativa dos limites de CB. \Rightarrow O método não é completo, em certos casos, pode falhar na busca de um caminho livre, mesmo existindo um.

Exemplo:



- O robô **A** é um segmento de reta.
- O robô deve se movimentar ao longo de dois corredores de largura W conectados em ângulo reto por uma esquina.
- Somente três caminhos livres podem ser construídos: Φ_1 (com espinha no meio do corredor 1), Φ_2 (com espinha no meio do corredor 2) e Φ_3 (com espinha entre as paredes externas dos dois corredores).
- A melhor escolha de O_A é no centro do segmento **A**.
- O Método dos Caminhos Livres tem sucesso em achar um caminho ao longo dos dois corredores somente quando **A** é menor do que $W \cdot (2)^{1/2}$.
- Existe um caminho livre entre os dois corredores até quando **A** for menor do que duas vezes esse tamanho.