Plano de locomoção de robôs

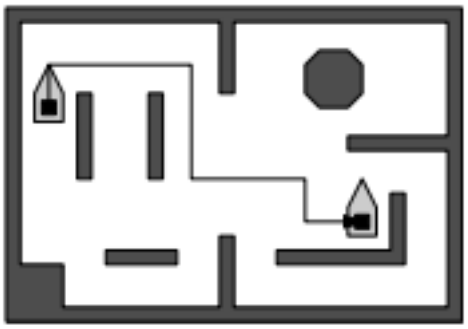
Jiang Zhi, Vinícius Moreno

14 de Dezembro de 2018

1 Introdução

O problema de locomoção de robôs é, achar um caminho entre a posição inicial e a posição final de um robô em um plano, sem colidir com obstáculos. Nesse projeto, considera-se que o robô é um ponto e os obstáculos são vários polígonos disjuntos.

Serão utilizados os algoritmos **ComputeFreeSpace** e **ComputePath**, ambos descritos na seção **13.2** do livro [1]. Como entrada para os algoritmos, será recebido dois pontos, a posição inicial e final do robô, e os polígonos disjuntos. Já como saída, os algoritmos retornarão um caminho entre os pontos, caso exista.



Exemplo do Problema

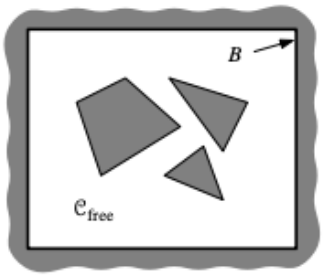
Na figura acima, há um exemplo do plano: os barcos são as posições inicial e final do robô, as áreas escuras são os obstáculos em que o robô deve evitar e a linha que liga os barcos é um caminho possível. Como pré-requisito para os algoritmos, duas estruturas de dados serão utilizadas, explicadas na próxima seção.

2 Estruturas de Dados

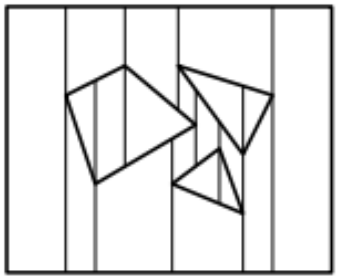
Para resolver o problema, usaremos duas estruturas de dados: um Mapa Trapezoidal, para o plano ser dividido em áreas transponíveis, e um DAG de Decisão, que auxiliará na divisão do plano.

2.1 Mapa Trapezoidal

O plano será dividido em diversos trapézios, de tal forma que cada um possua referência à seus trapézios vizinhos. Para implementar tal mapa, usou-se o **Algoritmo Incremental**, descrito na seção **6.1** do livro [1].



(a) Mapa Original



(b) Mapa Trapezoidal

2.2 DAG de Decisão

Para obter o mapa descrito na seção 2.1, usa-se um DAG de decisão, descrito na seção **6.2** do livro [1]. O processo consiste em incluir, uma a uma, as arestas dos polígonos disjuntos, refinando o Mapa Trapezoidal a cada iteração

2.3 Implementação

Obteremos, primeiramente, um mapa de todo o plano, ou seja, um mapa ainda não trapezoidado, que contém o interior dos polígonos que representam os obstáculos. Ao final do processo de construção, esse mapa será refinado a apenas informação da região fora dos obstáculos.

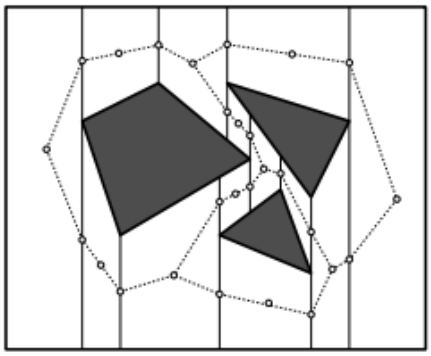
Este refinamento requer uma busca por pontos no Mapa Trapezoidal, para determinar onde cada segmento deverá ser inserido no Mapa. O DAG permite que tais buscas sejam feitas de maneira eficiente.

2.4 Consumo de Tempo

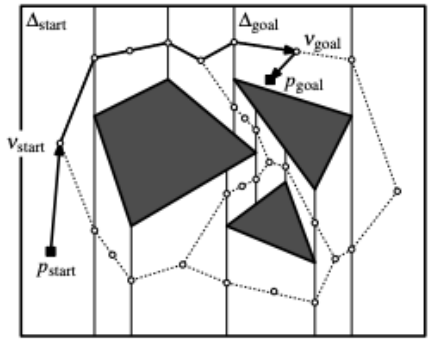
Para a construção do Mapa Trapezoidal e do DAG de decisão, consome-se tempo $O(nlogn)$ e espaço $O(n)$. Além disso, cada busca no DAG é feito em tempo $O(logn)$.

3 Grafo de Caminho

Com o Mapa Trapezoidal pronto, cria-se um grafo: os vértices serão os centros de cada trapézio e, para cada trapézio vizinho, um vértice no meio do segmento de reta entre eles. Já as arestas, serão as ligações entre o centro de cada trapézio e o meio do segmento entre cada vizinho. Um exemplo pode ser visto a seguir:



Logo após, dois novos vértices serão inseridos no grafo, o inicial e final do robô. Será feita, para cada um dos novos vértices, uma outra pesquisa no DAG, descobrindo em qual trapézio cada um se encontra. Encontrando-os, duas novas arestas serão inseridas, cada uma entre o ponto e o centro do trapézio que ele se encontra.

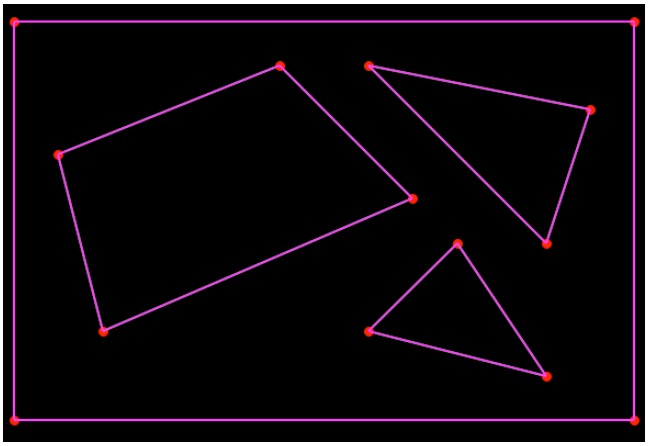


Grafo de Caminho com as posições inicial e final

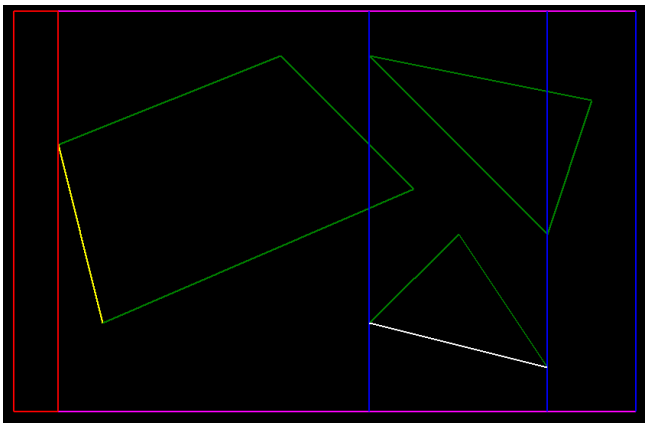
Finalmente, será executado um DFS no grafo, a partir do ponto inicial do robô, tentando chegar em seu ponto final. Caso exista um caminho possível, ele será encontrado.

4 Representação na Plataforma Alexis

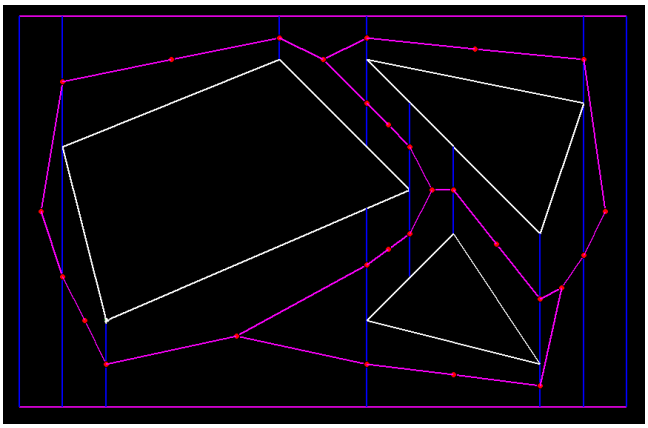
Inicialmente, mostra-se os poligonos iniciais e a região da borda, com arestas de cor rosa e vértices de cor vermelha.



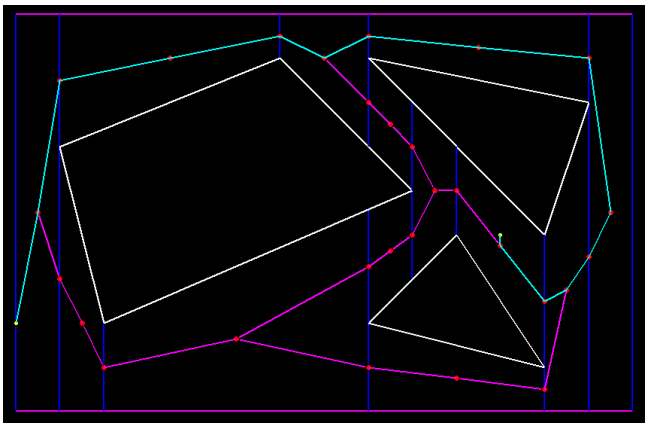
Todos segmentos ficam de coloração verde e o Algoritmo Incremental começa a funcionar. O segmento que está sendo adicionado naquela iteração, fica em cor amarela e, os trapézios que estão sendo adicionados, ficam em cor vermelha. Terminada a iteração, o segmento já adicionado fica em cor branca e os trapézios, também ja adicionados, ficam em cor azul.



Após todos segmentos serem adicionados, as regiões proibidas calculadas e retiradas, o grafo pode ser criado: no centro do trapézio mais à esquerda, destacado em cor amarela, um vértice de cor verde será criado. Logo após, os trapézios vizinhos a ele serão destacados, em verde claro o trapézio à direita superior e em laranja o da direita inferior. O destaque dos trapézios são apagados e é criado um vértice entre cada um dos vizinhos, além das arestas em cor verde entre eles. Terminado tudo, as arestas do grafo ficam em cor rosa e seus vértices em cor vermelha.



Por fim, os vértices inicial e final do robô são adicionados ao grafo, com coloração amarela, e sua aresta ao trapézio que se encontra, criada. O DFS, então, procura um caminho a partir do ponto inicial. A cada vértice que ele visita, a aresta que ele utilizou para chegar é destacada em azul claro. Caso o DFS encontre o ponto final, o caminho continuará em azul claro. Caso não encontre, as arestas do grafo voltam a ter cor rosa e uma mensagem é enviada ao terminal.



5 Referências

[1] Livro - *Computational Geometry, Algorithms and Application - Third Edition, editora Springer, de Mark de Berg, Otfried Cheong, Marc van Kreveld e Mark Overmars*

[2] Site - *Building of Trapezoidal Map* - https://bit.ly/2CetNiX