Inteligência Artificial para Robótica Móvel: CT-213

Instituto Tecnológico de Aeronáutica (ITA)

Relatório do Laboratório 2 - Busca Informada

Leonardo Peres Dias

31 de março de 2025

Sumário

1 Breve Explicação em Alto Nível da Implementação

1.1 Algoritmo Dijkstra

A implementação do algoritmo de Dijkstra utiliza uma grade de nós, onde cada nó possui atributos como a posição, o custo acumulado g e o nó pai, permitindo a reconstrução do caminho. O procedimento básico é descrito a seguir:

1. Inicializa-se o nó de partida com $g = 0$ e este é inserido em uma fila de prioridade ordenada pelo valor de g.

2. Em cada iteração, o nó com o menor custo acumulado é extraído da fila e marcado como visitado (flag `closed`).

3. São obtidos os sucessores (vizinhos válidos, considerando movimentos 8-conectados e evitando obstáculos) utilizando a função `get_successors`.

4. Para cada vizinho, calcula-se o custo de transição através da função `get_edge_cost`. Se o custo acumulado via nó atual for menor que o custo previamente registrado para o vizinho, atualiza-se o valor de g, define-se o nó atual como seu pai e o vizinho é reinserido na fila.

5. Quando o nó objetivo é extraído, o caminho é reconstruído retrocedendo a partir do objetivo por meio dos atributos pai.

1.2 Algoritmo Greedy Search

Na busca gulosa, a escolha do próximo nó a ser expandido é guiada exclusivamente por uma heurística, definida aqui como a distância Euclidiana até o objetivo. O fluxo de execução é o seguinte:

1. Inicializa-se o nó de partida com $g = 0$, calculando o valor f como a distância até o objetivo.

2. O nó de partida é inserido em uma fila de prioridade ordenada pelo valor f.

3. Em cada iteração, o nó com o menor valor f é extraído e seus vizinhos são expandidos.

4. Para cada vizinho, calcula-se o custo de transição; caso o caminho via o nó atual resulte em um custo menor, atualiza-se o valor de g, recalcula-se f (com base na heurística) e define-se o nó atual como pai.

5. O processo se repete até que o nó objetivo seja expandido, momento em que o caminho é reconstruído.

1.3 Algoritmo A*

O algoritmo A* combina as estratégias de Dijkstra e da busca gulosa, utilizando tanto o custo acumulado g quanto uma heurística admissível para calcular o valor f. O procedimento é detalhado a seguir:

1. Inicializa-se o nó de partida com $g = 0$ e define-se $f = g + h$, onde h representa a distância Euclidiana até o objetivo.

2. O nó de partida é inserido em uma fila de prioridade ordenada pelo valor f.

3. Em cada iteração, extrai-se o nó com o menor valor f e expandem-se seus vizinhos.

4. Para cada vizinho, calcula-se o custo de transição e, se o novo custo acumulado for inferior ao previamente registrado, atualiza-se g, recalcula-se $f = g + h$, e define-se o nó atual como pai.

5. O algoritmo termina quando o nó objetivo é expandido e, então, o caminho de menor custo é reconstruído utilizando os ponteiros de pai.

2 Figuras Comprovando Funcionamento do Código

2.1 Algoritmo Dijkstra

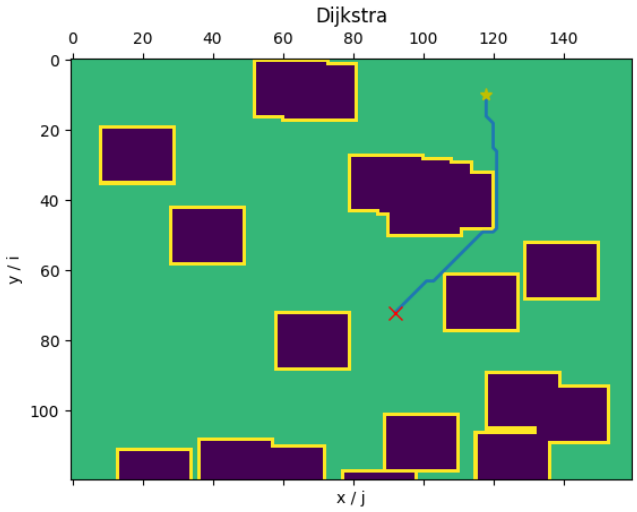

Figura 1: Exemplo de execução do algoritmo Dijkstra.

2.2 Algoritmo Greedy Search

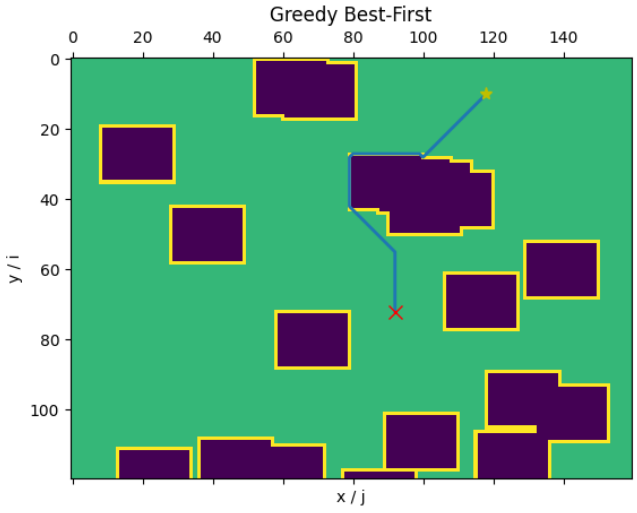

Figura 2: Exemplo de execução do algoritmo Greedy Search.

2.3 Algoritmo A*

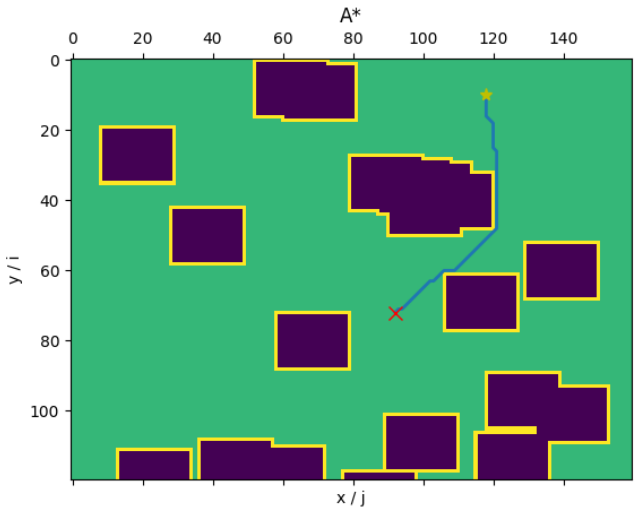

Figura 3: Exemplo de execução do algoritmo A*.

3 Comparação entre os Algoritmos

Algoritmo	**Tempo computacional (s)**		**Custo do caminho**	
	Média	**Desvio padrão**	**Média**	**Desvio padrão**
Dijkstra	0.03742	0.02104	79.83	38.57
Greedy Search	0.00130	0.00030	103.12	58.79
A*	0.00732	0.00648	79.83	38.57

Tabela 1: Tabela de comparação entre os algoritmos de planejamento de caminho.