UFMG
UNIVERSIDADE FEDERAL
DE MINAS GERAIS

Robótica Móvel

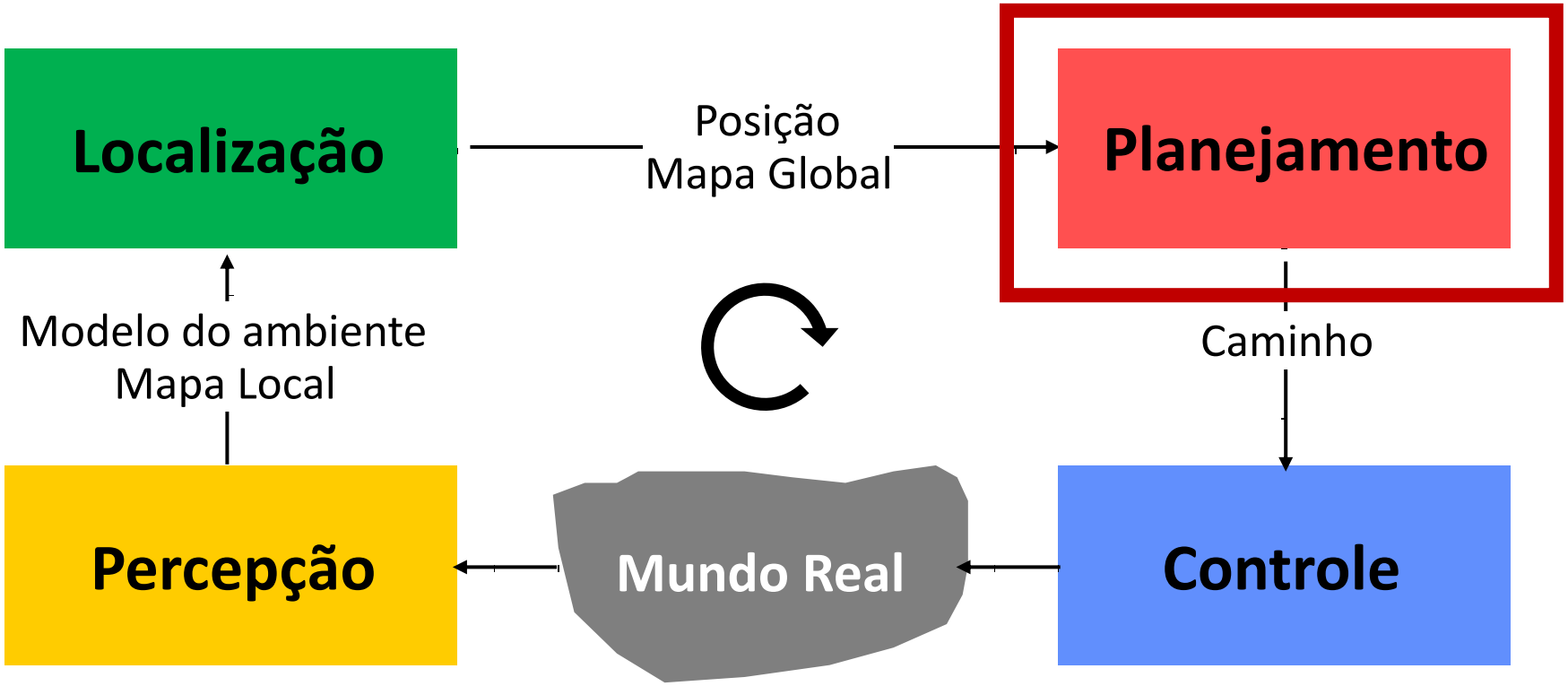
Planejamento de caminhos – Bug Algorithms

Prof. Douglas G. Macharet
douglas.macharet@dcc.ufmg.br

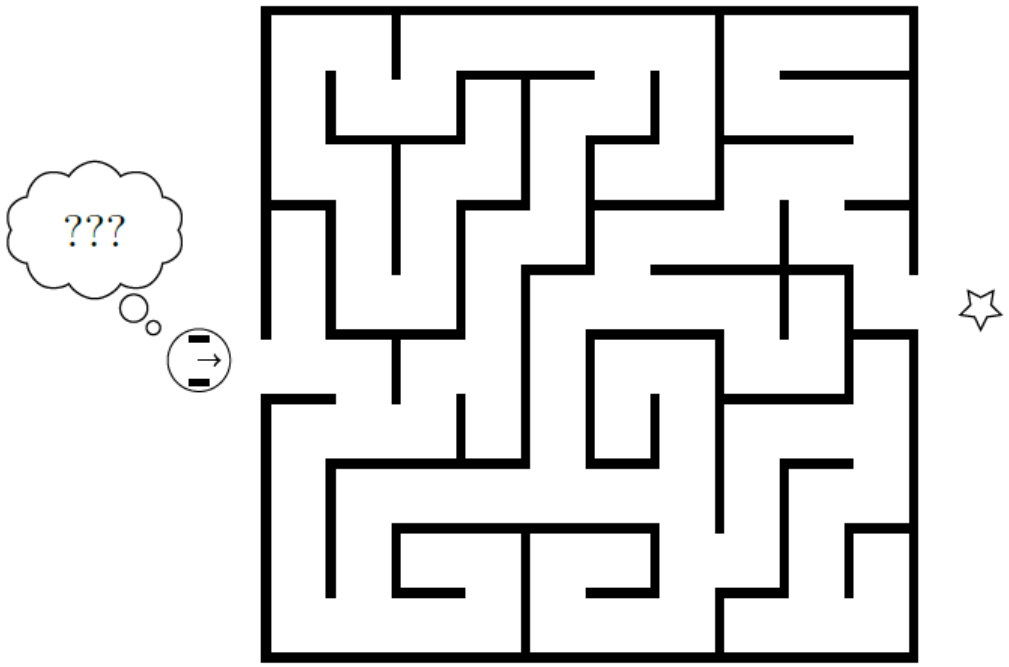
DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

VERLab
Visão e Robótica

Introdução


Localização
Posição
Mapa Global
Planejamento
Modelo do ambiente
Mapa Local
Caminho
Percepção
Mundo Real
Controle

Introdução



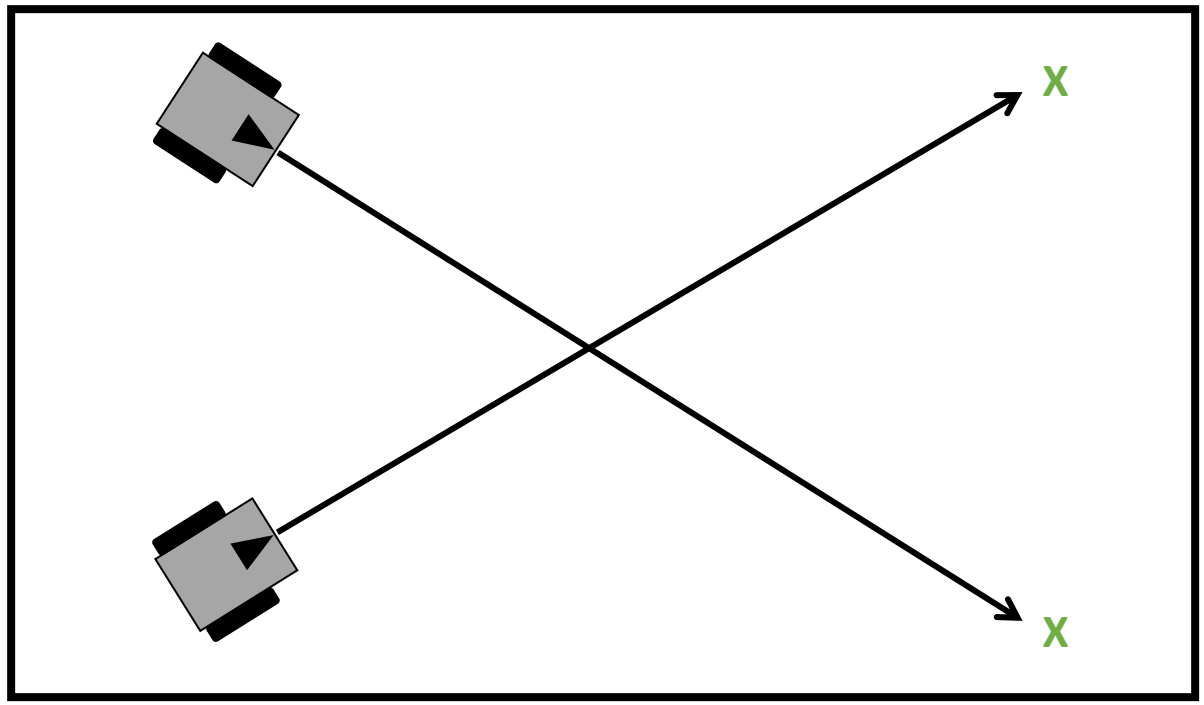
Planejamento de movimento: uso de métodos computacionais para gerar o movimento de um robô para realizar uma tarefa específica.

Introdução

- **Caminho**
  - Conjunto de configurações em uma ordem específica sem considerar o tempo em que as configurações serão alcançadas

- **Trajetória**
  - Conjunto de configurações em ordem específica considerando o momento (tempo) em que cada uma deveria ser atingida

Introdução



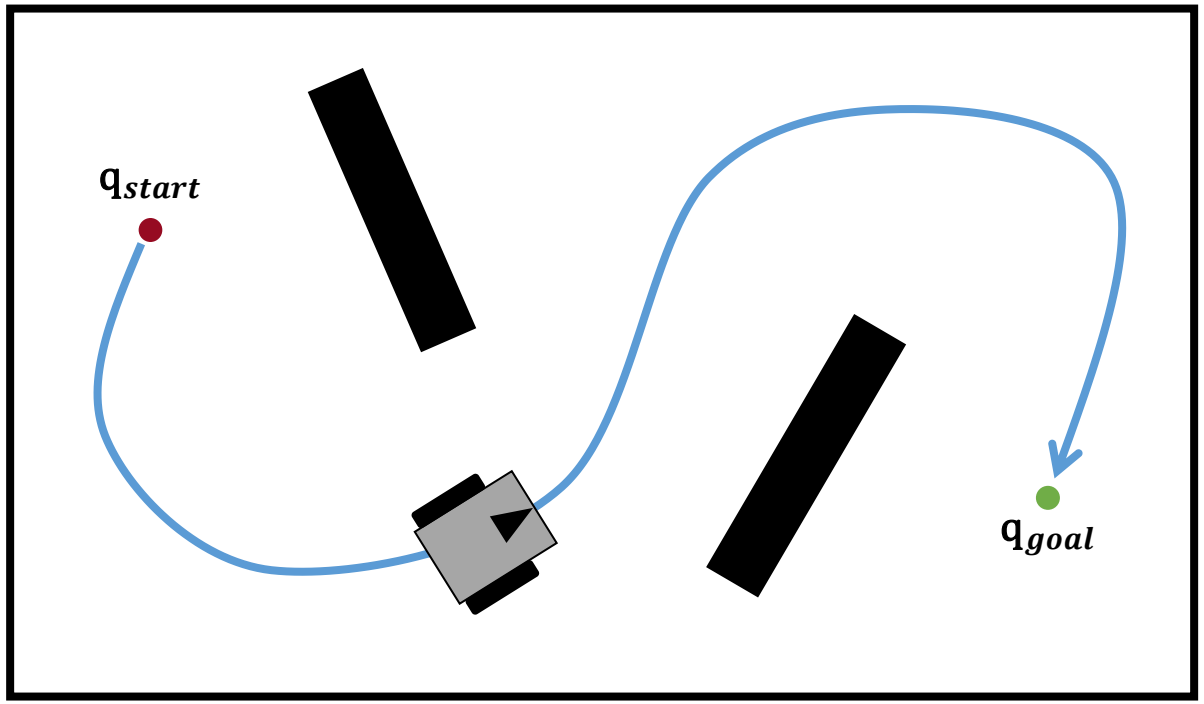
- São caminhos válidos?
  - SIM!
- São trajetórias válidas?
  - Depende

Introdução

- **Planejamento de caminhos**
  - Determinar um caminho contínuo (sequência válida) de uma configuração inicial (q_{start}) até uma configuração final (q_{goal})

- Objetivos
  - Realizável
  - Seguro
  - Eficiente
    - Tempo, comprimento, energia, segurança, …

Introdução



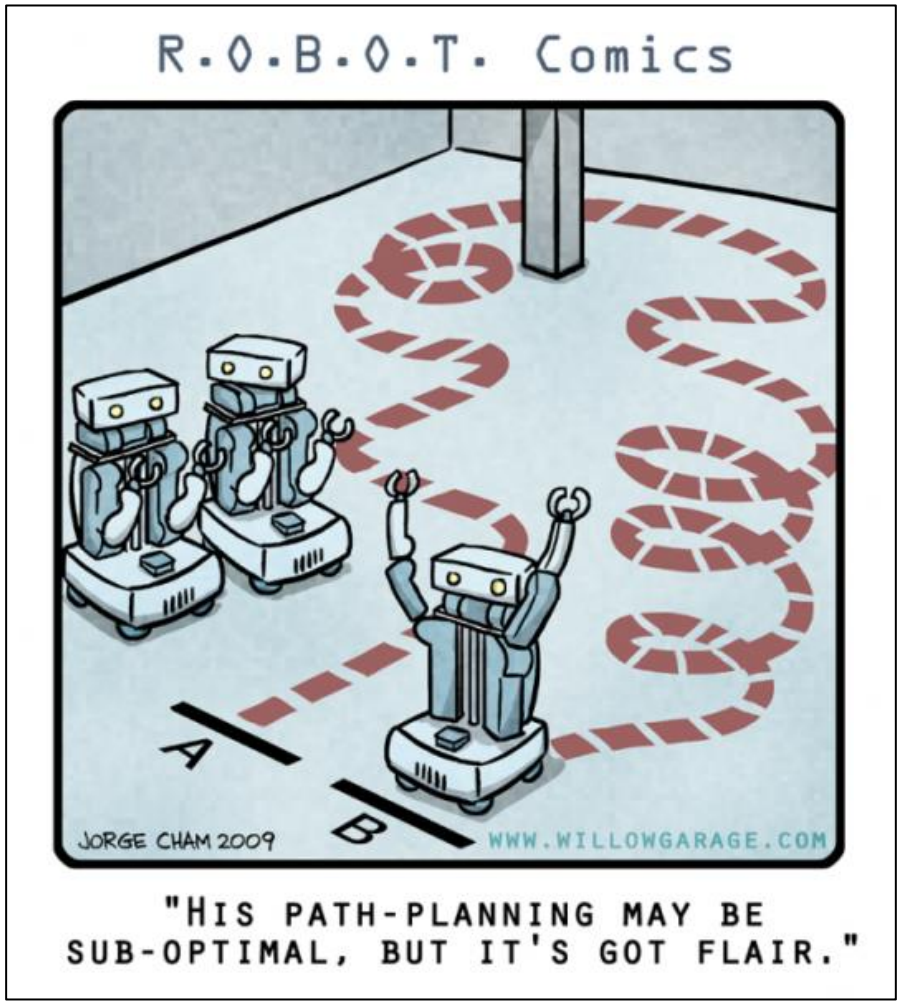
$\gamma : [0,1] \rightarrow \mathcal{C}_{free}$, onde $\gamma(0) = q_s$ e $\gamma(1) = q_g$

- Um caminho realizável é uma curva contínua no espaço de configurações e não viola nenhuma restrição
- Uma solução aceitável é um caminho viável começando na configuração inicial e terminando na configuração objetivo
- Se um critério de otimização for fornecido, a solução deve ser ótima (minimizar uma função de custo) ou o mais próximo possível do ideal

Introdução

- Características/propriedades dos algoritmos
  - Completude
    - Garantia de encontrar solução em tempo finito se existir (ou falha)
  - Eficácia
    - Qualidade do caminho encontrado dado um critério de otimização
  - Complexidade
    - Eficiência (uso de recursos) para se calcular a solução

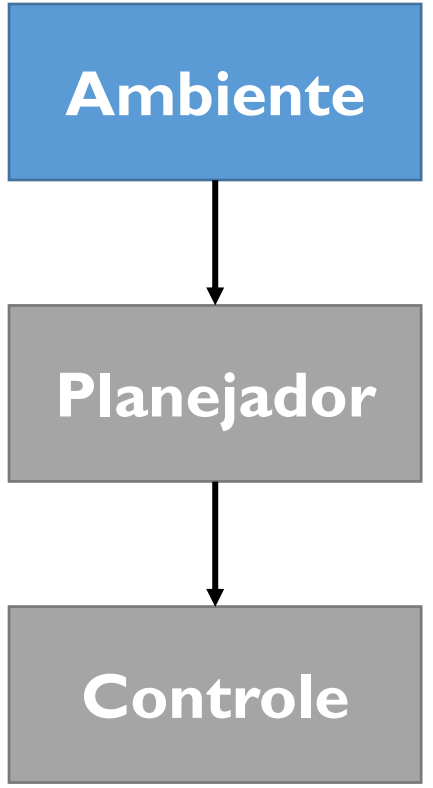
Introdução


R.O.B.O.T. Comics
A
B
JORGE CHAM 2009
WWW.WILLOWGARAGE.COM
"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Introdução
Ambiente

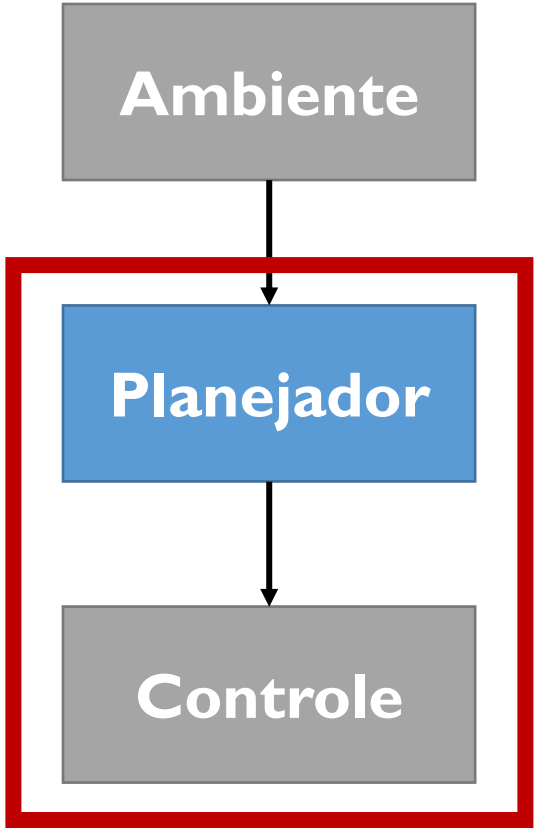
- Observável
  - Totalmente x Parcialmente
  - Grau de acesso às informações do ambiente
  - Relação direta com os sensores utilizados

- Dinâmico x Estático
  - O ambiente não muda durante o planejamento
  - E se houverem outros robôs no ambiente?



Introdução
Estratégias

- Reativa
  - Navegação → Sensor/Controle
  - Modelo simplificado do mundo (ou nenhum)
  - Decisões locais
- Deliberativa
  - Planejamento → Busca
  - Modelo mais completo do mundo
  - Decisões mais complexas



Bug Algorithms

- Estratégias reativas simples para navegação
  - Inspiração no *right-hand rule* para labirintos
- Informação necessária
  - Conhecimento global do goal (localização do robô perfeita)
  - Conhecimento local do ambiente (sensor de toque/proximidade)
- Assume-se um robô puntual

https://en.wikipedia.org/wiki/Maze_solving_algorithm

Bug Algorithms

- Dois comportamentos básicos
  1. Mover em direção ao goal (objetivo)
  2. Seguir parede (circunavegar obstáculos)

- Problemas
  - Caminhos não são os mais eficientes
  - Necessita de informação sensorial instantânea

Bug Algorithms

- Notação
  - q_{start} e q_{goal}
  - "Hit point": q_i^H
  - "Leave point": q_i^L

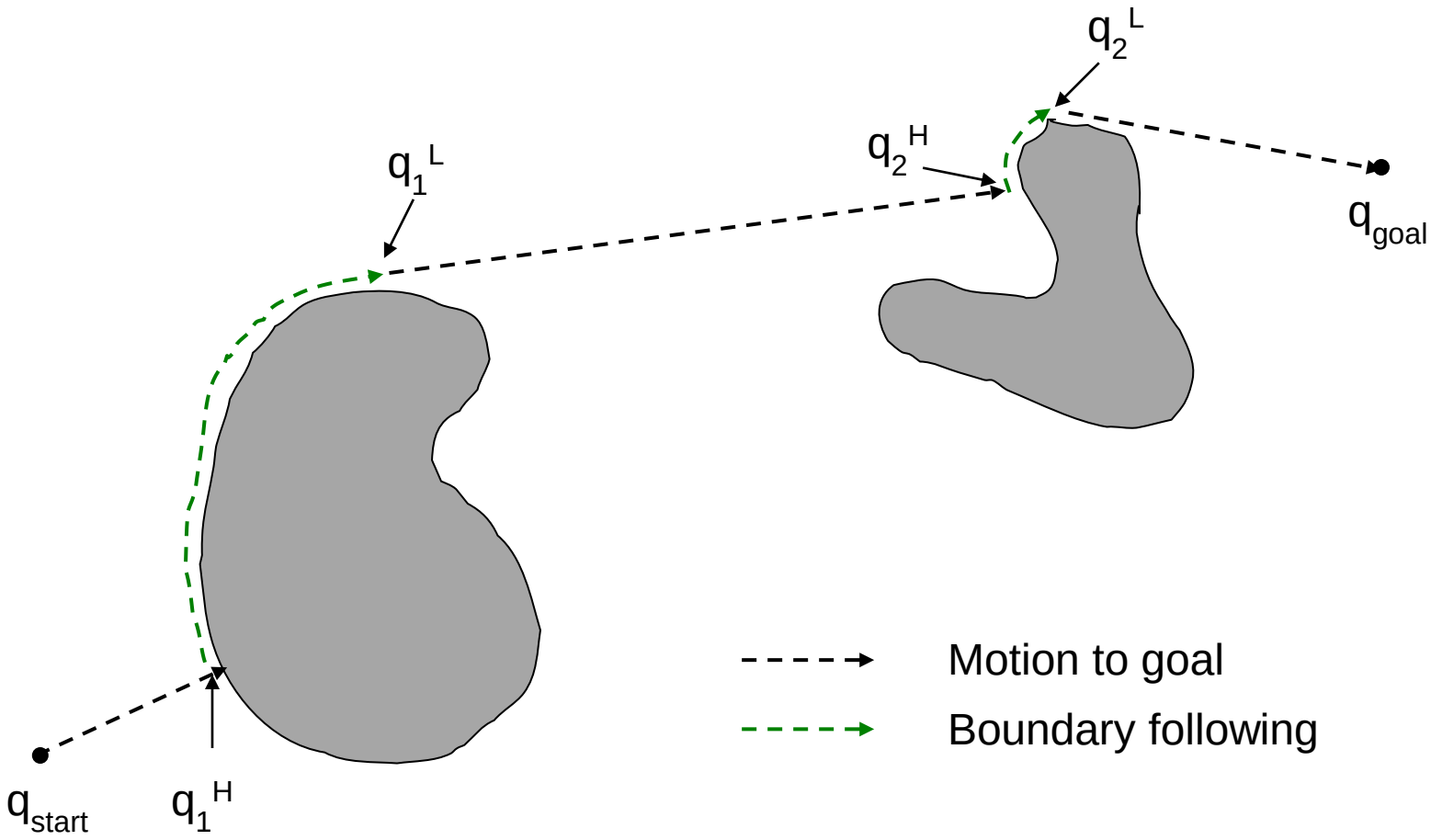
- O caminho é formado por uma sequência de pares q_i^H / q_i^L, delimitados nos extremos por q_{start} e q_{goal}

Bug Algorithm 0

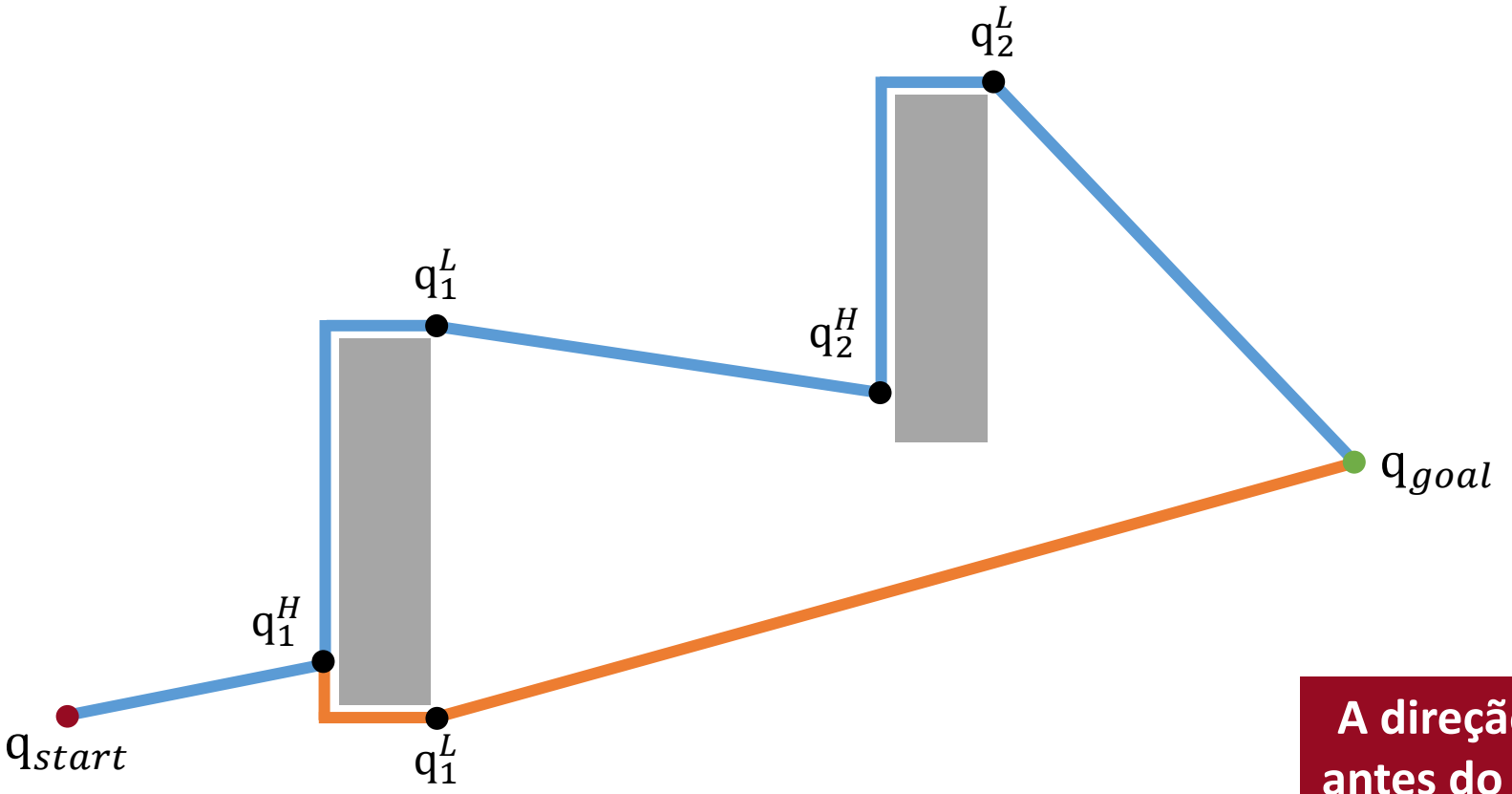
- Algoritmo

```
1. Siga em direção à posição goal
2. Caso um obstáculo seja encontrado
   2.1. Contorne o obstáculo até que se tenha
   novamente uma visão livre da posição goal
3. Retorne ao passo inicial
```

Bug Algorithm 0



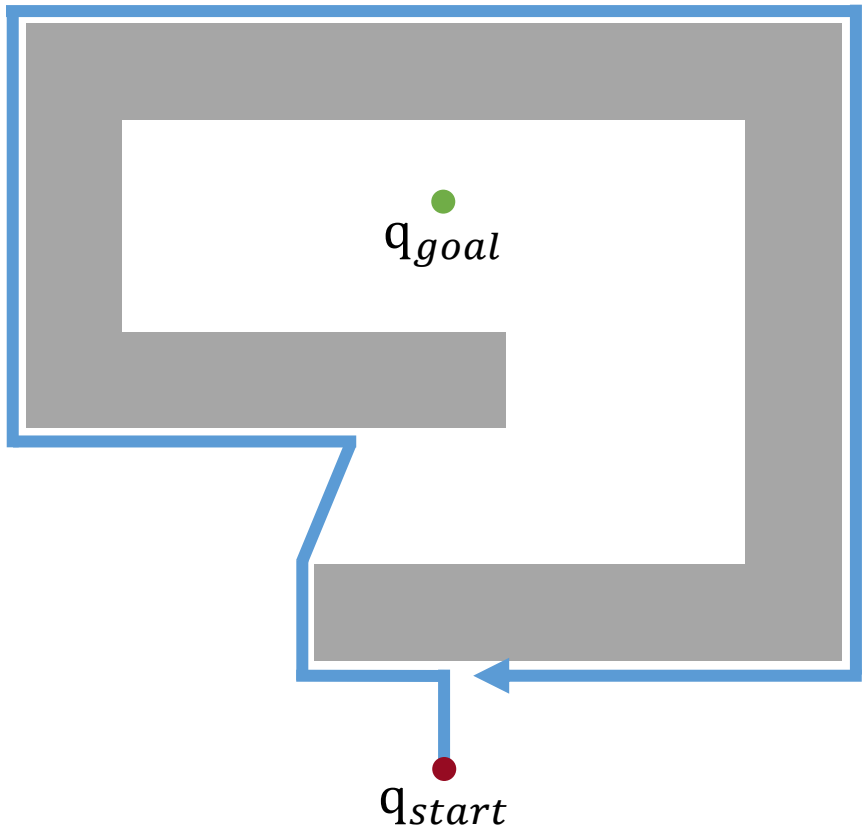
Bug Algorithm 0



A direção de giro é definida antes do início da navegação!

Bug Algorithm 0

- Esse algoritmo é completo?
  - Não
  - Qual cenário de falha?
- Como é possível melhorá-lo?
  - Adicionar memória!

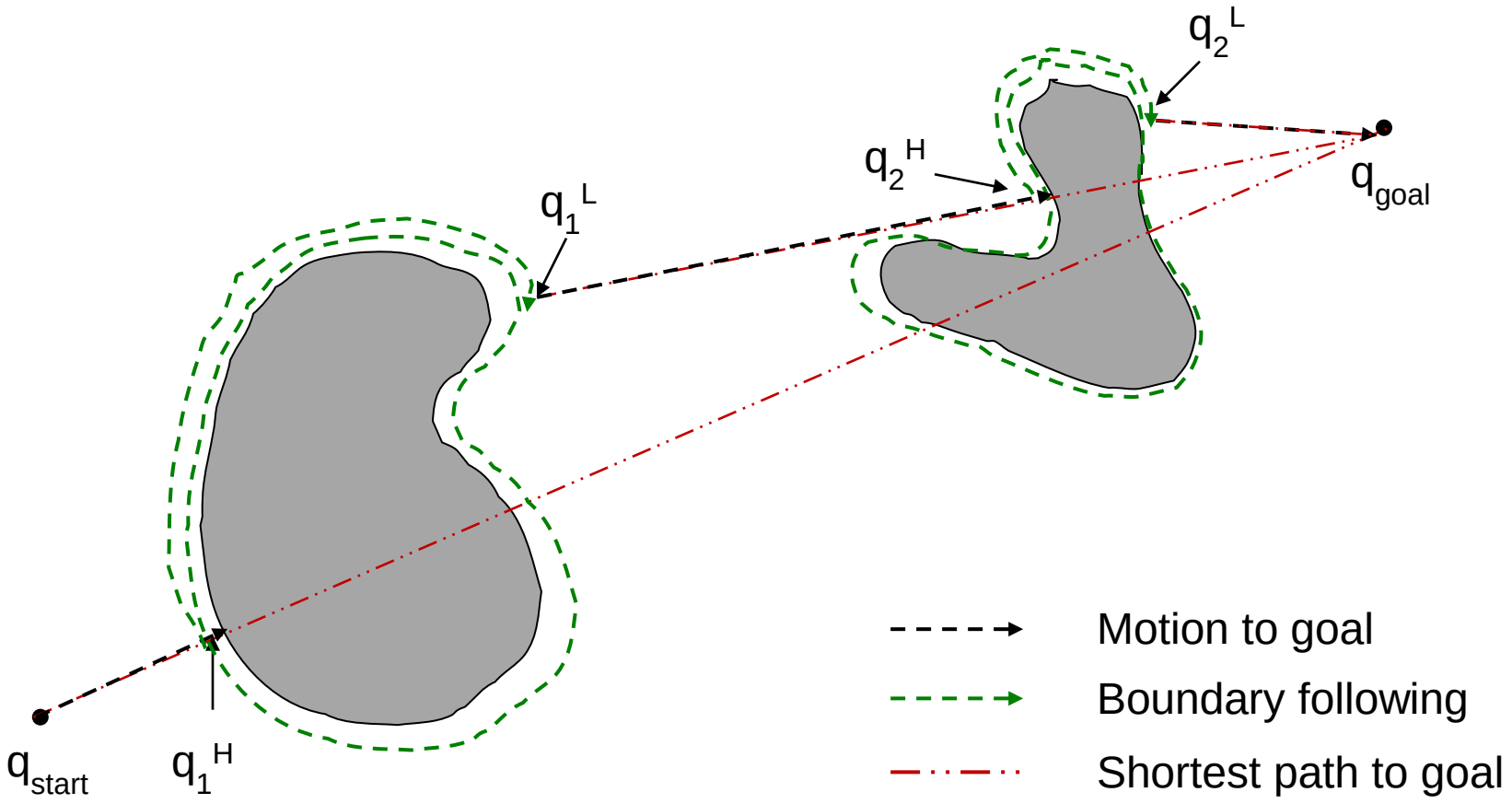


Bug Algorithm 1

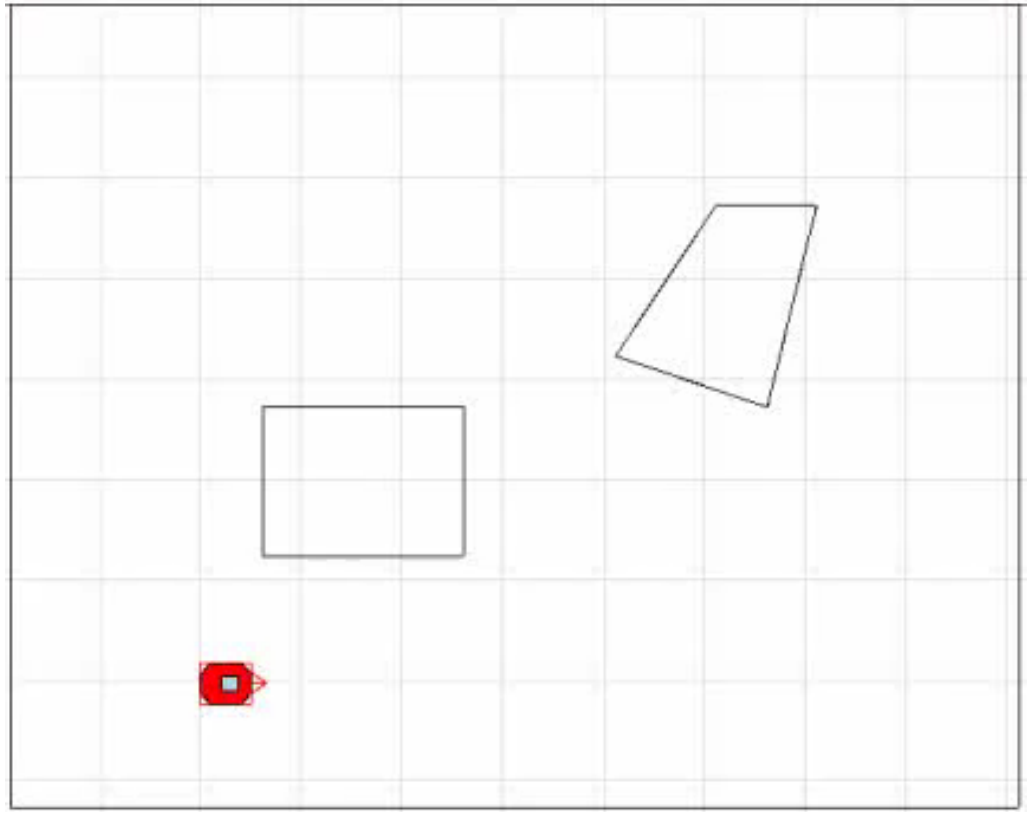
- Algoritmo

```
1. Siga em direção à posição goal
2. Caso um obstáculo seja encontrado
   2.1. Contorne o obstáculo por completo
   2.2. Lembre o ponto mais próximo do goal
3. Retorne usando a menor rota ao ponto mais
próximo guardado e continue em direção ao goal
```

Bug Algorithm 1



Bug Algorithm 1



https://www.youtube.com/watch?v=Vm37TRRJAjc

Bug Algorithm 1

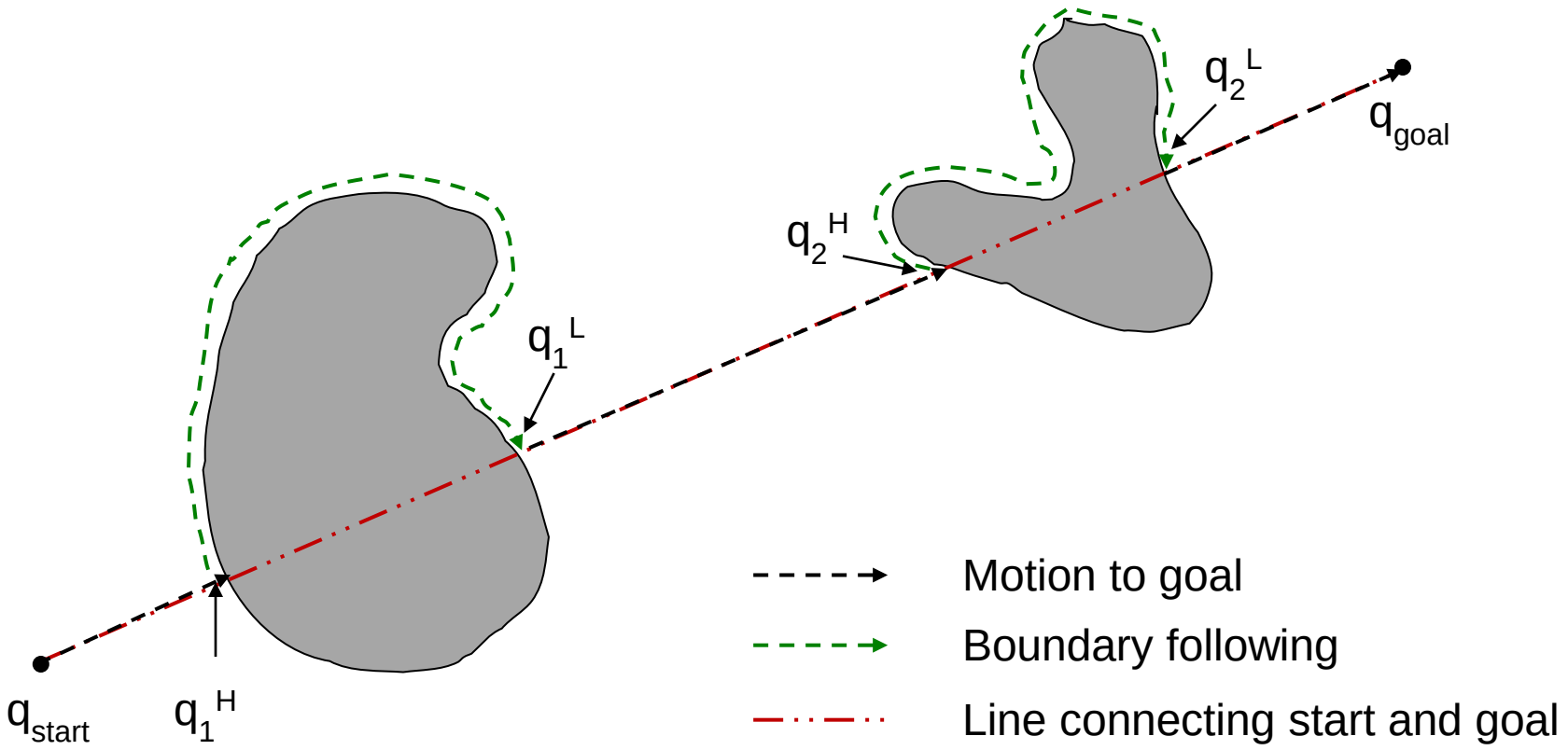
- Algoritmo completo
  - Encontra um caminho se esse existir
  - Caso contrário, informa execução com falha
- O robô nunca retorna à um obstáculo
- Caminho pode ser muito ineficiente
  - Depende do perímetro do obstáculo
- Alguma ideia de como melhorar?

Bug Algorithm 2

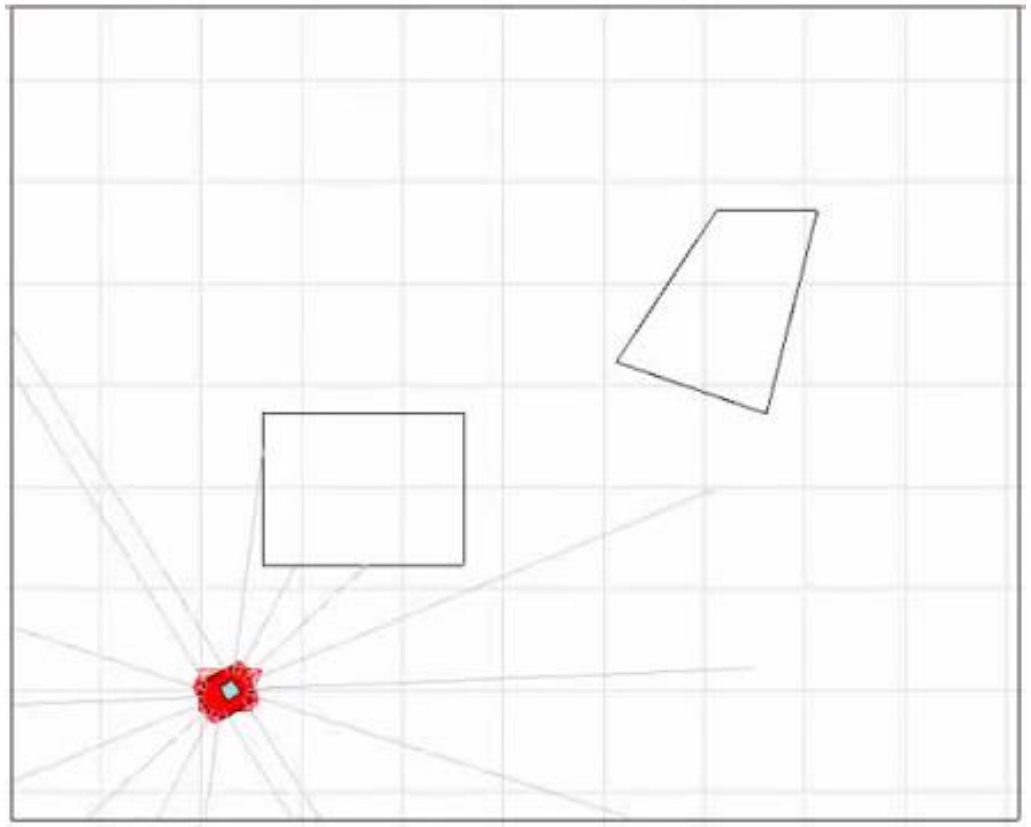
- Algoritmo

```
1. Siga em direção à posição goal
2. Caso um obstáculo seja encontrado
   2.1. Contorne o obstáculo até que o segmento de reta
   que liga os pontos inicial e final (m-line) seja
   novamente encontrado em um ponto mais próximo ao goal
3. Retorne ao passo inicial
```

Bug Algorithm 2

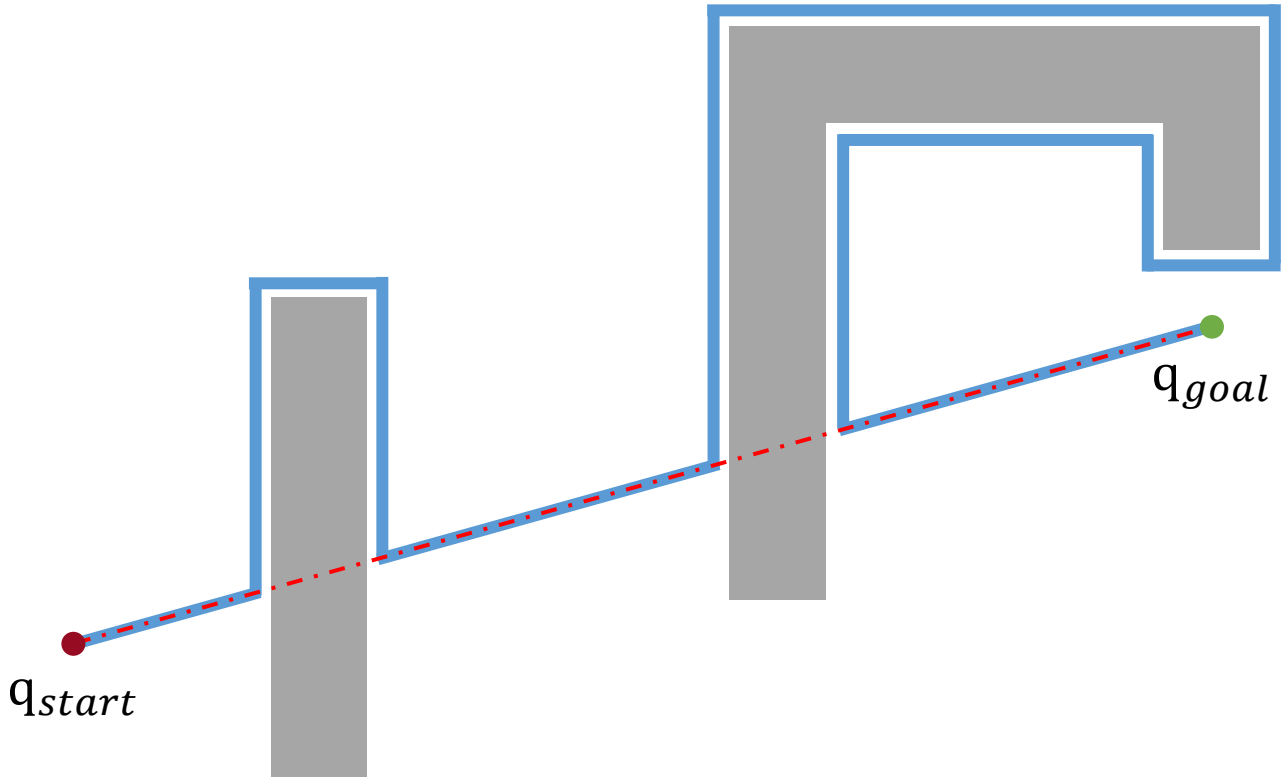


Bug Algorithm 2

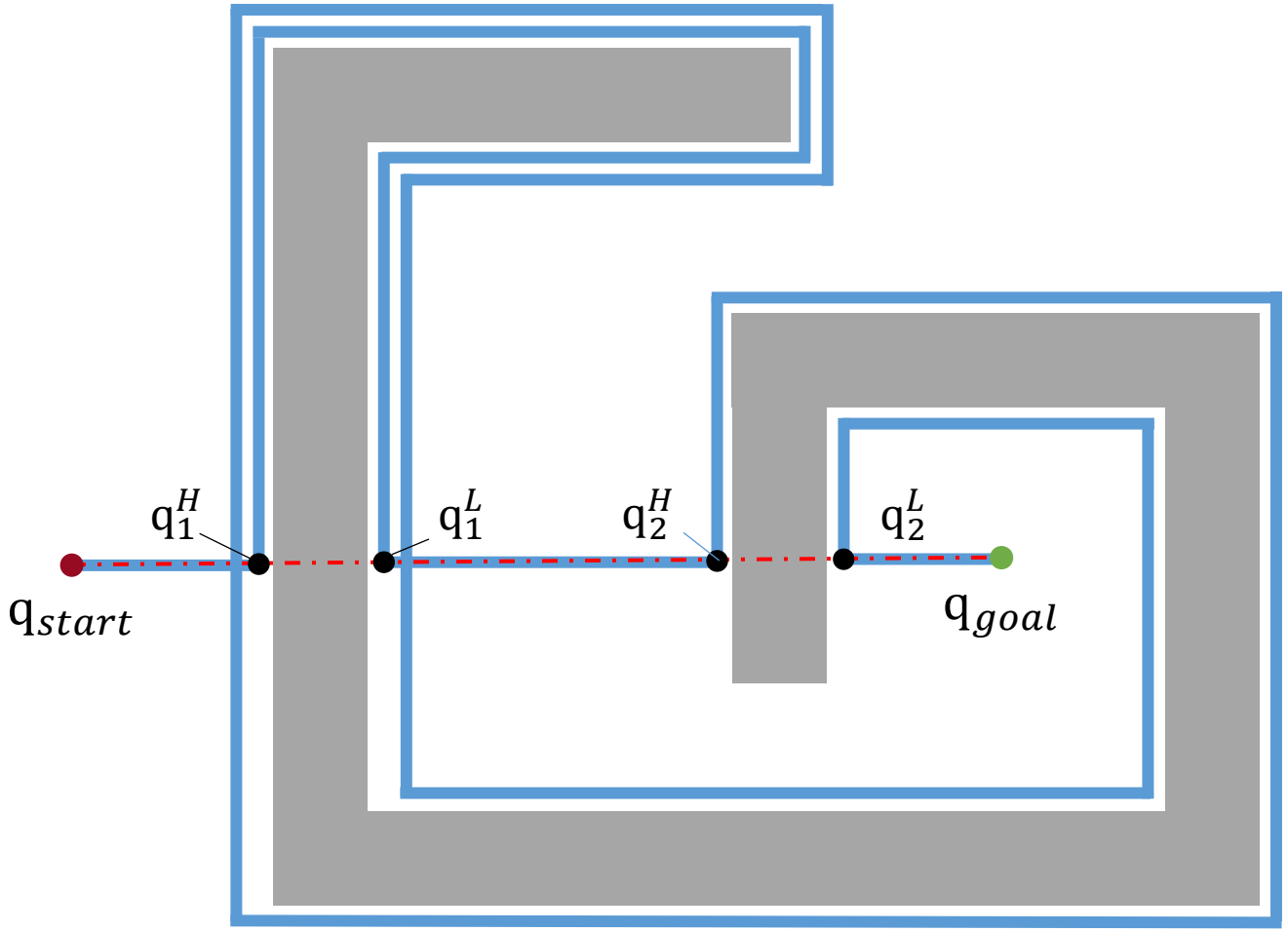


https://www.youtube.com/watch?v=jW10e99AeaI

Bug Algorithm 2



Bug Algorithm 2



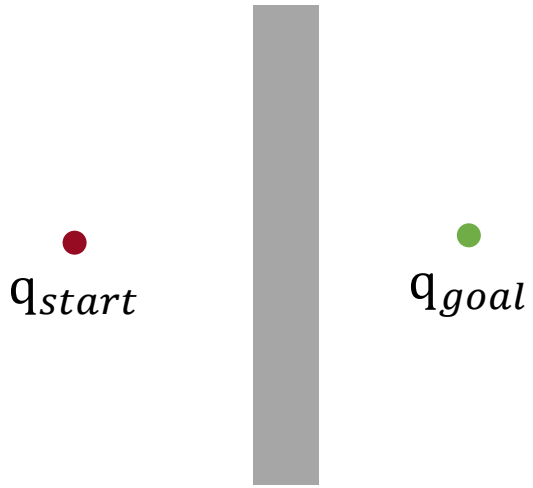
Bug Algorithm 2

- Algoritmo completo (ou quase)
  - Encontra um caminho se esse existir
  - Informa execução com falha → identificar ciclos
- O robô nunca retorna à um obstáculo
- Apenas irá (parcialmente) circunavegar aqueles obstáculos que interceptam a linha entre q_{start} e q_{goal}

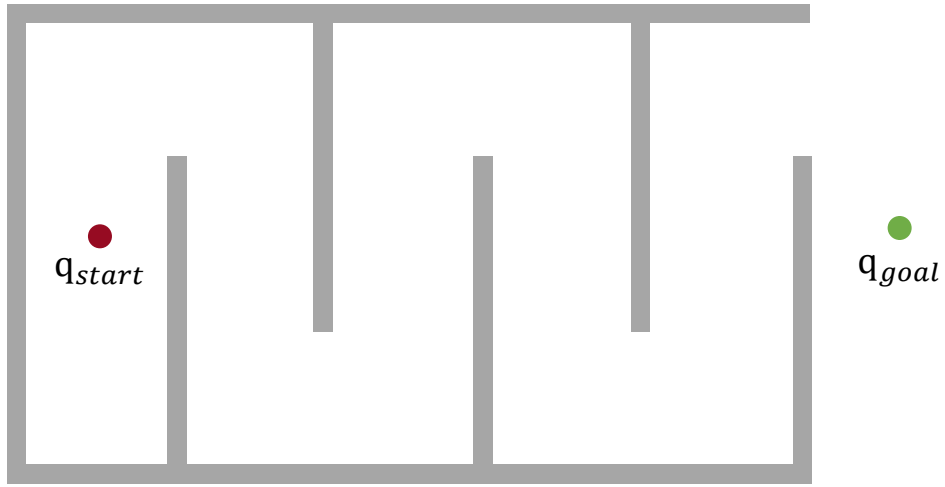
Bug 1 vs. Bug 2

- Bug 1
  - Algoritmo de busca exaustiva
  - Verifica todas as opções para tomar decisão
  - Comportamento mais previsível
- Bug 2
  - Algoritmo guloso
  - Escolhe a primeira opção que parece melhor
  - Desempenho melhor na maioria dos casos

Bug 1 vs. Bug 2

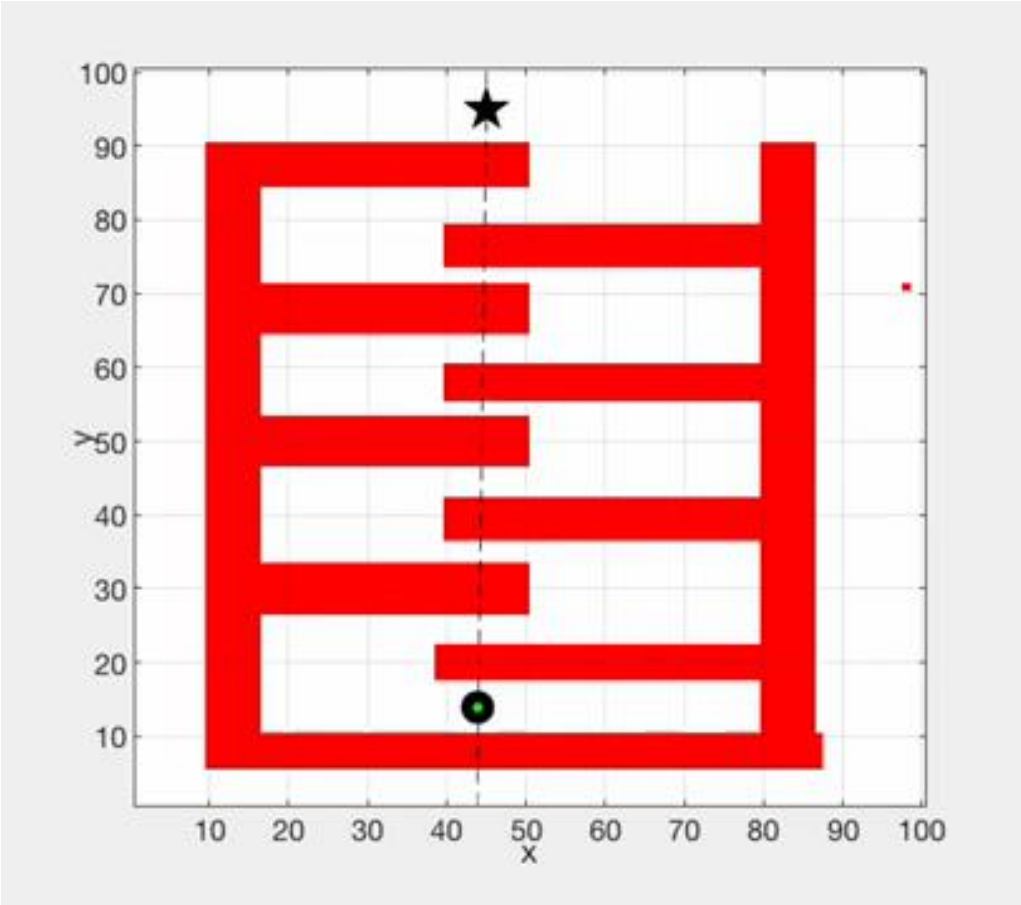


Bug 2 é melhor.



Bug 1 é melhor.

Bug 1 vs. Bug 2



https://www.youtube.com/watch?v=hXmlwwjd9aY

Considerações finais

- Seja D a distância em linha reta do start ao goal, e P_i o perímetro do i-ésimo obstáculo presente no ambiente

	Bug 1	**Bug 2**
Lower bound: a <u>menor</u> distância do caminho	D	D
Upper bound: a <u>maior</u> distância do caminho	$D + 1.5 \sum_i P_i$ (1.5 → 2)	$D + \sum_i n_i \frac{P_i}{2}$ (n_i: Interseções)

Fonte: *Lectures on Robotic Planning and Kinematics*

Considerações finais

- Algoritmos reativos com informação local
- Bug 0
  - Algoritmo simples, não é completo
- Bug 1
  - Algoritmo completo, seguro e confiável
- Bug 2
  - Algoritmo completo, pode ser melhor

Considerações finais

- Como é possível melhorar essa navegação?
- Ter algum tipo de planejamento prévio
  - Conhecimento do ambiente → Mapa
  - Determinar um caminho a priori já sabendo todos os detalhes/problemas do ambiente