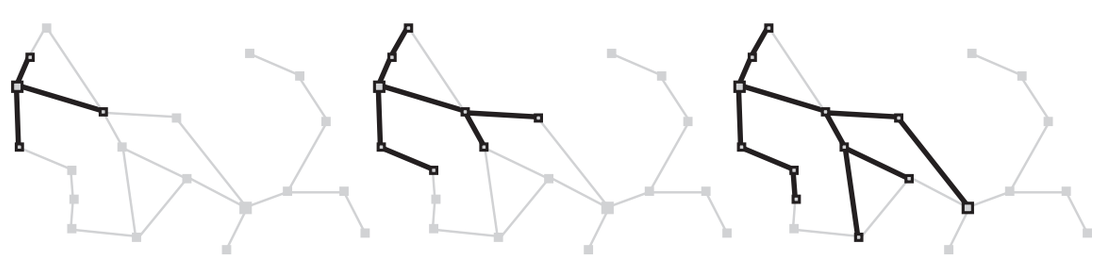
navegação

MAC318 - Introdução à Programação de Robôs Móveis

parte 2

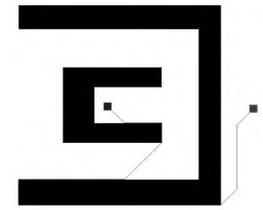
busca heurística

- Busca cega
  - usa apenas conhecimento sobre a especificação do problema
- Busca informada
  - usa conhecimento heurístico sobre domínio

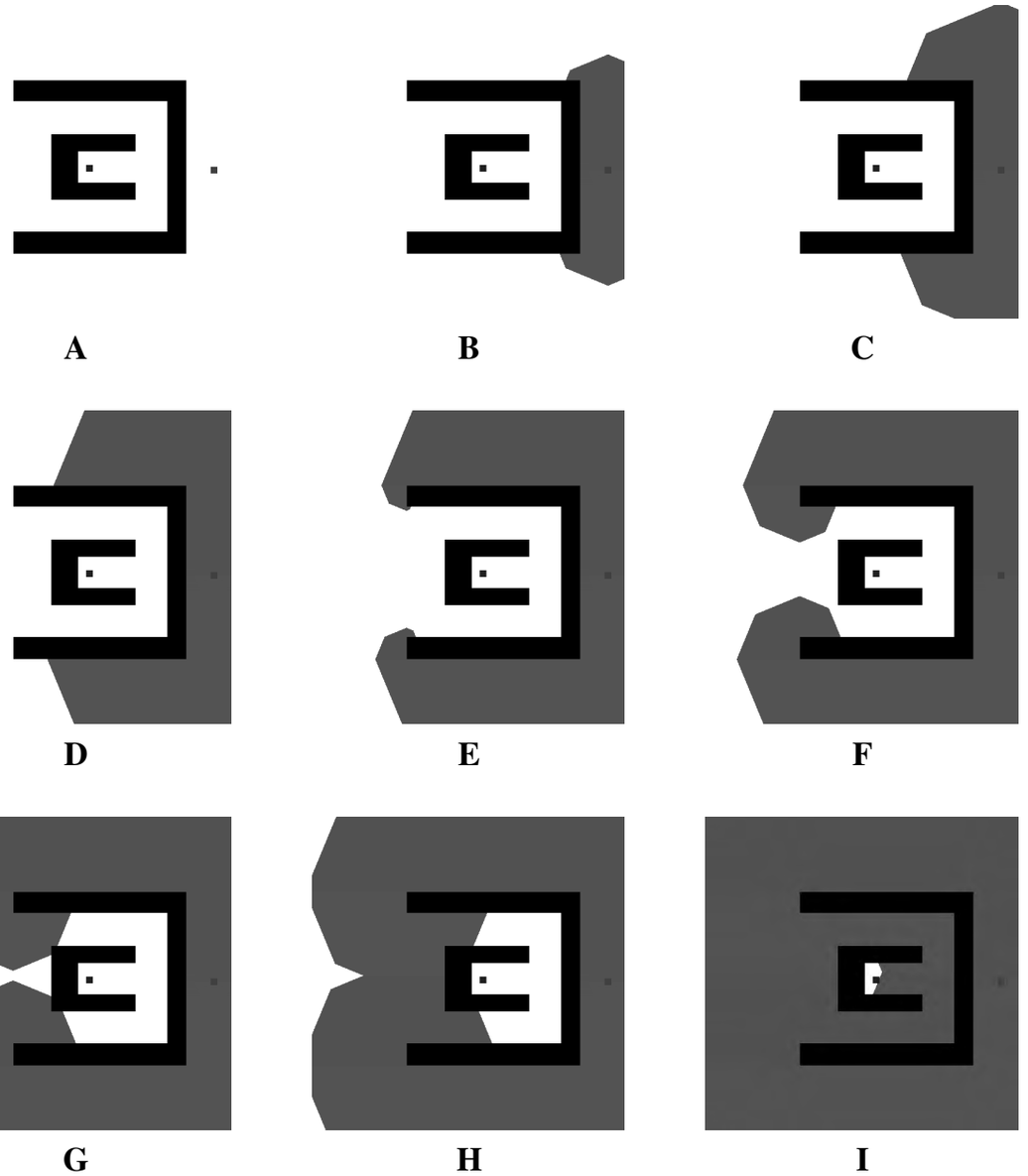


busca cega

busca cega



- Expande muitos nós em direções não interessantes, mesmo quando meta é alcançável a partir de ponto em linha reta



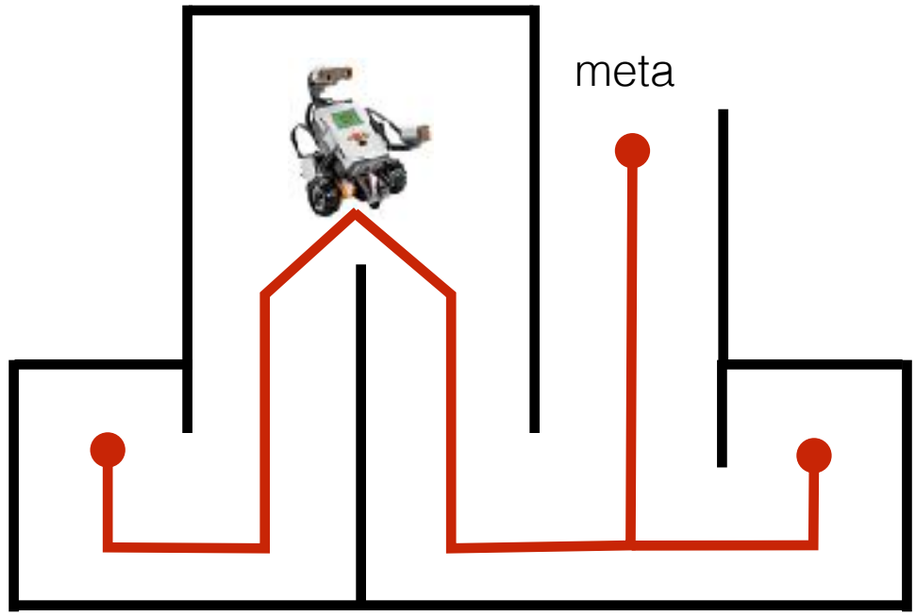
busca informada

- Utilizar conhecimento sobre domínio (“adivinhar futuro”)

- *Ex.: se meta é “quase” alcançável a partir de ponto atual em linha reta*

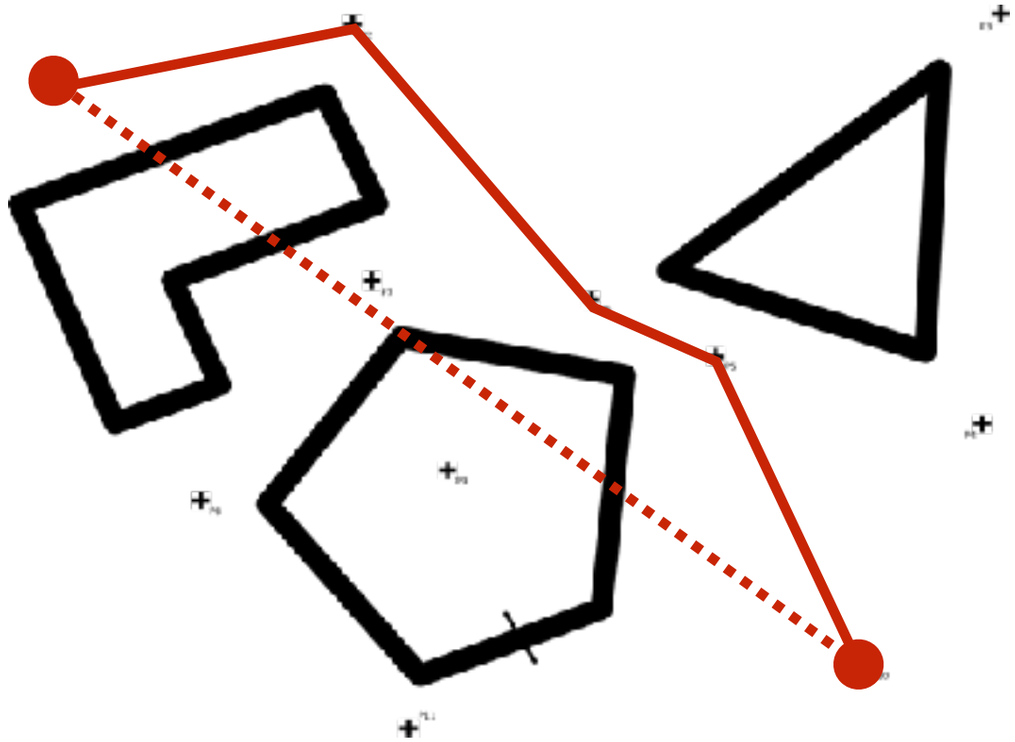
- heurística *h(n)*: função que subestima custo à meta a partir de nó *n*

- *h(n) ≥ 0, h(meta) = 0*



heurística

- Mapa métrico
  - Menor distância entre ponto e meta: distância euclidiana
- Mapa de ocupação
  - Menor distância sem obstáculos
    - Manhattan para 4-vizinhança



busca de melhor escolha

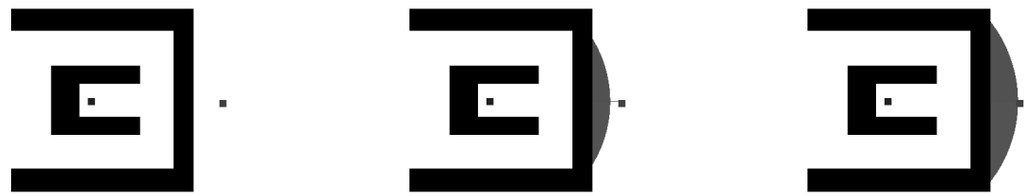
- Busca gulosa usando **heurística** como prioridade
- Não garante encontrar caminho mais curto
- Expande poucos nós em média

melhor escolha

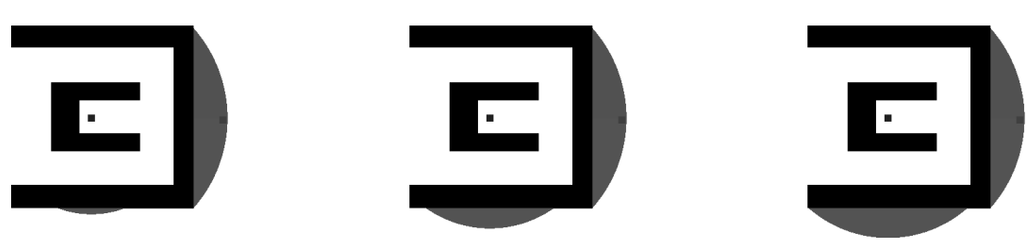
- Inicie *fila de prioridade* com nó inicial
- Repita:
  - selecione nó *n* não explorado com menor prioridade
  - acrescente *n* a conjunto *explorados* e seus sucessores à *fila* com prioridade *h(n)*
  - se algum sucessor de *n* for estado-meta, retorne solução correspondente

melhor escolha

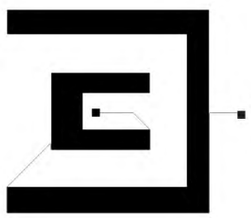
- rápida
- subótima

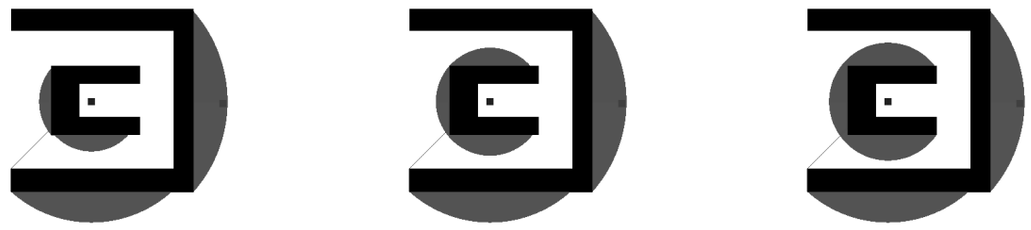


A B C



D E F



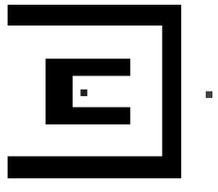


G H I

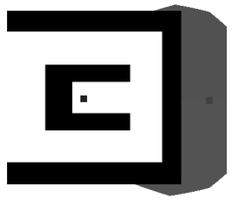
A*

- Busca de custo uniforme utilizando **custo + heurística** como prioridade
- Encontra caminho mais curto
- Expande menor número de nós

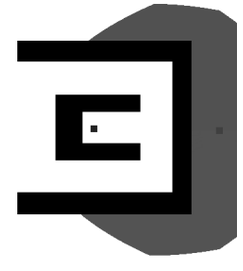
A*



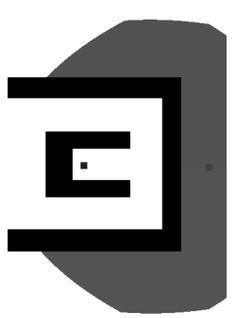
A



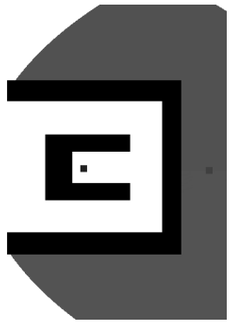
B



C

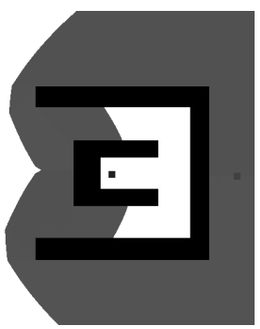


D

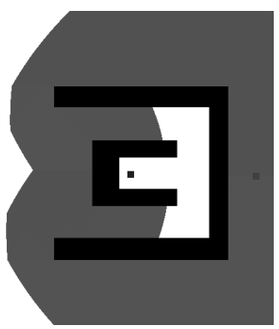


E

F

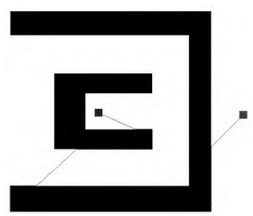


G



H

I



A*

- Inicie *fila de prioridade* com nó inicial
- Repita:
  - selecione nó *n* não explorado com menor prioridade
  - acrescente *n* a conjunto *explorados* e seus sucessores à *fila* com prioridade *c(n) + h(n)*
  - se algum sucessor de *n* for estado-meta, retorne solução correspondente

A*

Dijkstra

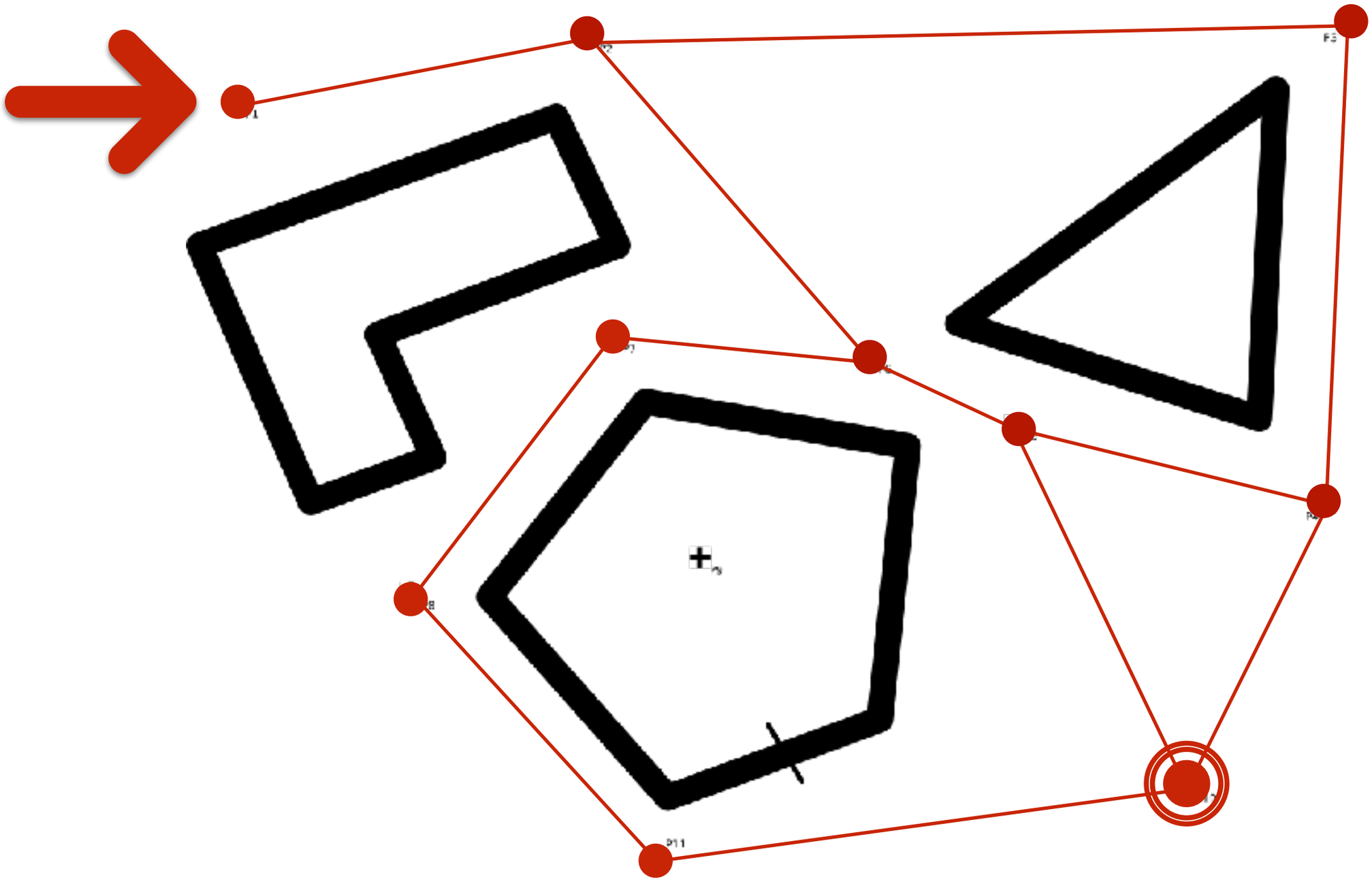
A*

BFS/Dijkstra

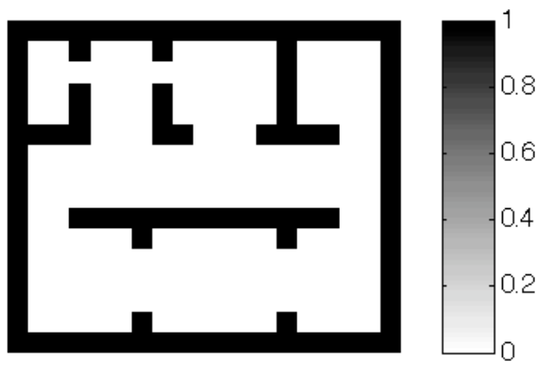
Projeto 6: Parte A

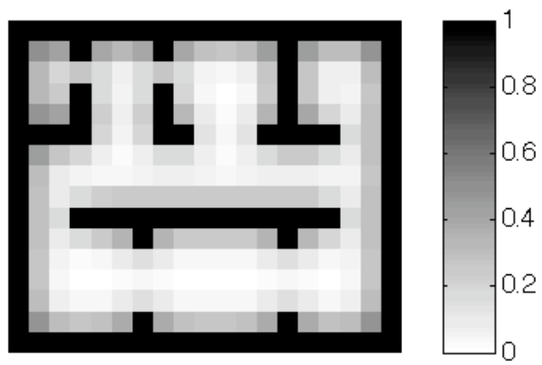
- Implemente algoritmos de melhor escolha e A*
- Utilize os algoritmos para encontrar trajetória de menor comprimento (custo) ligando P1 a P8 e P1 a P10
  - use mapa topológico
- Faça robô executar trajetória (meça distância à meta para diferentes representações)

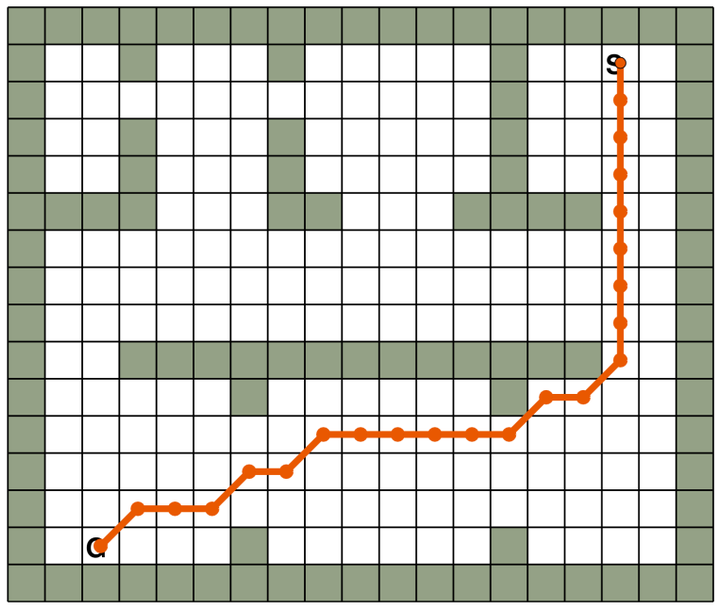
mapa topológico

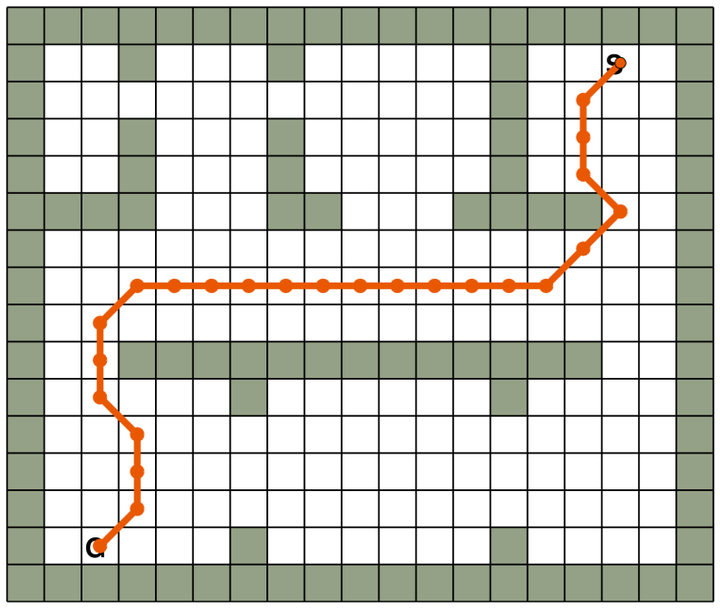


mapa de ocupação probabilístico







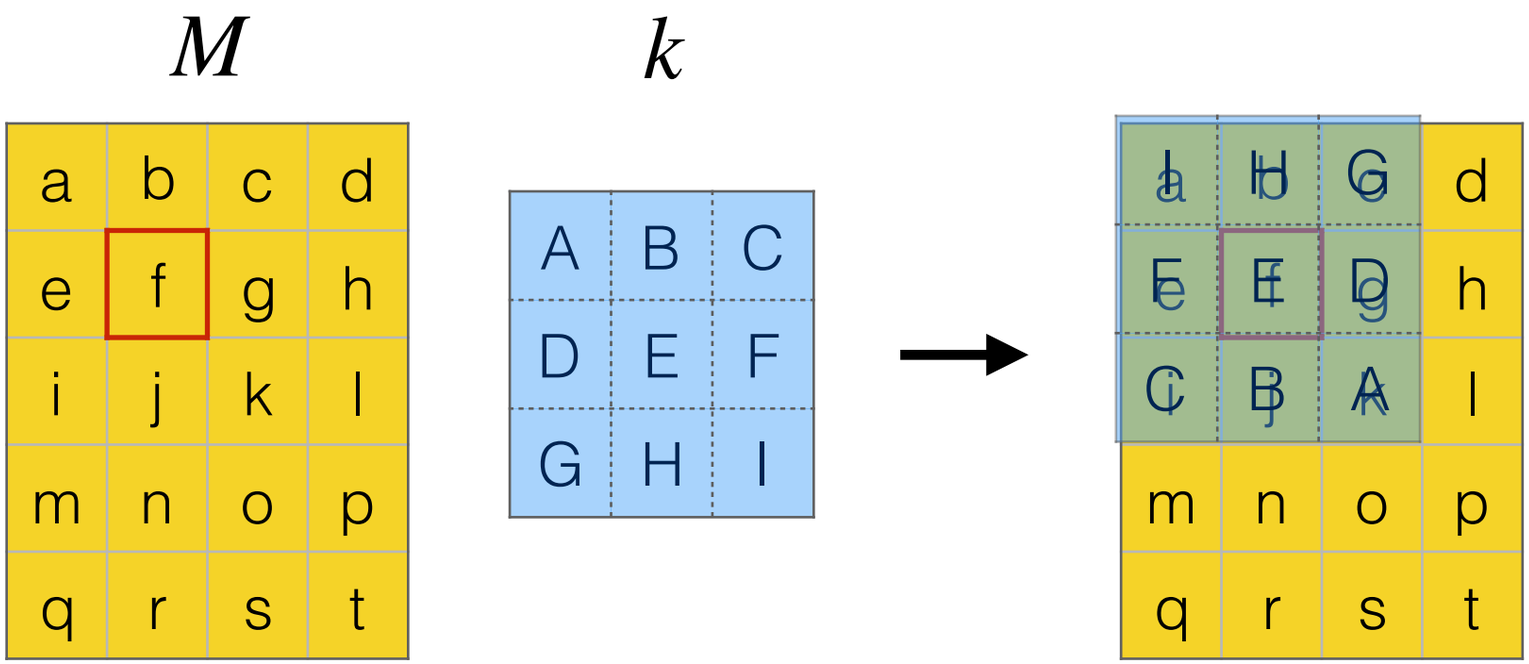


convolução

- Mapa M, matriz de convolução (máscara) k

$$(M \circledast k)(x,y) = \sum_{i,j} M(i,j)k(x-i,y-j)$$

- *Ex.: Operação de "desfocar" imagem (gaussian blur)*



$$(M \circledast k)(2,2) =$$

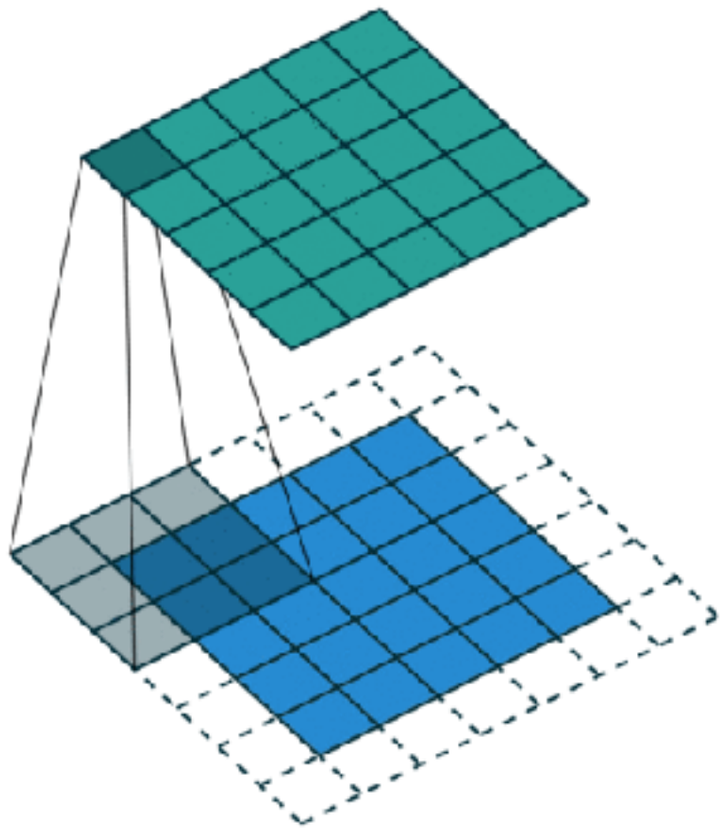
aI + bH + cG
+ eF + fE + gD
+ iC + jB + kA

https://en.wikipedia.org/wiki/Kernel_(image_processing)

convolução

- Mapa M, matriz de convolução (máscara) k

$$(M \circledast k)(x, y) = \sum_{i,j} M(i, j)k(x - i, y - j)$$



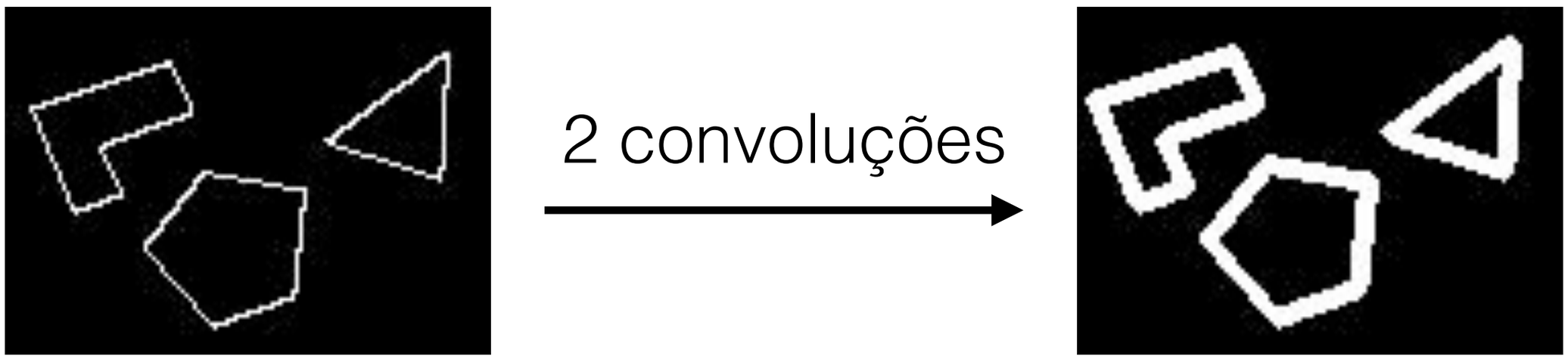
https://en.wikipedia.org/wiki/Kernel_(image_processing)

convolução

- Ex.: matriz de convolução para “engrossar linhas”

0	1	0
1	1	1
0	1	0

- número de convoluções representa grossura

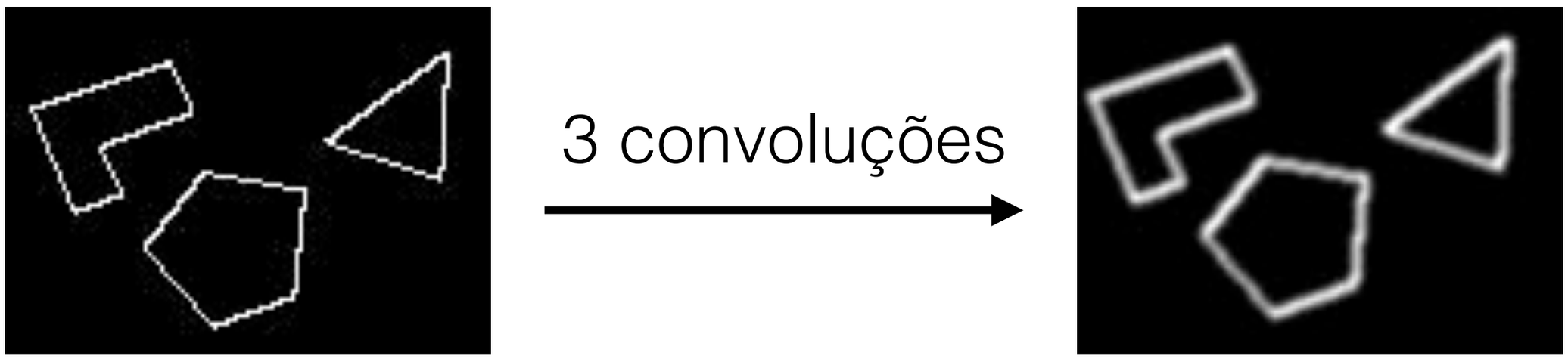


convolução gaussiana

- Célula ocupada é ocupada com probabilidade proporcional a número de vizinhos ocupados ponderados pela distância
- Matriz de convolução:

0.05	0.1	0.05
0.1	0.4	0.1
0.05	0.1	0.05

- Valor de mapa convolucionado pode ser lido como **probabilidade de célula estar ocupada**



mapa de ocupação probabilístico

- Probabilidade de robô colidir ao passar por célula (x, y) é $M(x,y)$
- Probabilidade de robô colidir em trajetória π:

$$p(\pi) = 1 - \prod_{x,y \in \pi} [1 - M(x, y)]$$

- Custo de trajetória π sem colisão:

$$\sum_{t=2}^{|\pi|} c(x_{t-1}, y_{t-1} \rightarrow x_t, y_t)$$

A* com mapa de ocupação probabilístico

- Otimização bi-objetivo: minimizar probabilidade de colisão e custo
  - heurística: distância em linha reta com probabilidade zero de colisão
  - como combinar objetivos em um só
- Função de avaliação multiplicativa (busca cega quando $M(x,y) = 0$):

$$c(n) \cdot p(n) + h(n)$$

n: nó de busca, representa trajetória de posição inicial até posição atual na busca
c(*n*): custo de trajetória representada por n
p(*n*): probabilidade de colisão na trajetória representada por n
h(*n*): heurística de posição n até meta

- Função de avaliação aditiva:

$$\alpha \cdot c(n) + (1 - \alpha)p(n) + h(n)$$

α: peso entre custo e probabilidade; quando α=0, encontra caminho mais seguro, quando α=1, encontra caminho menos custoso

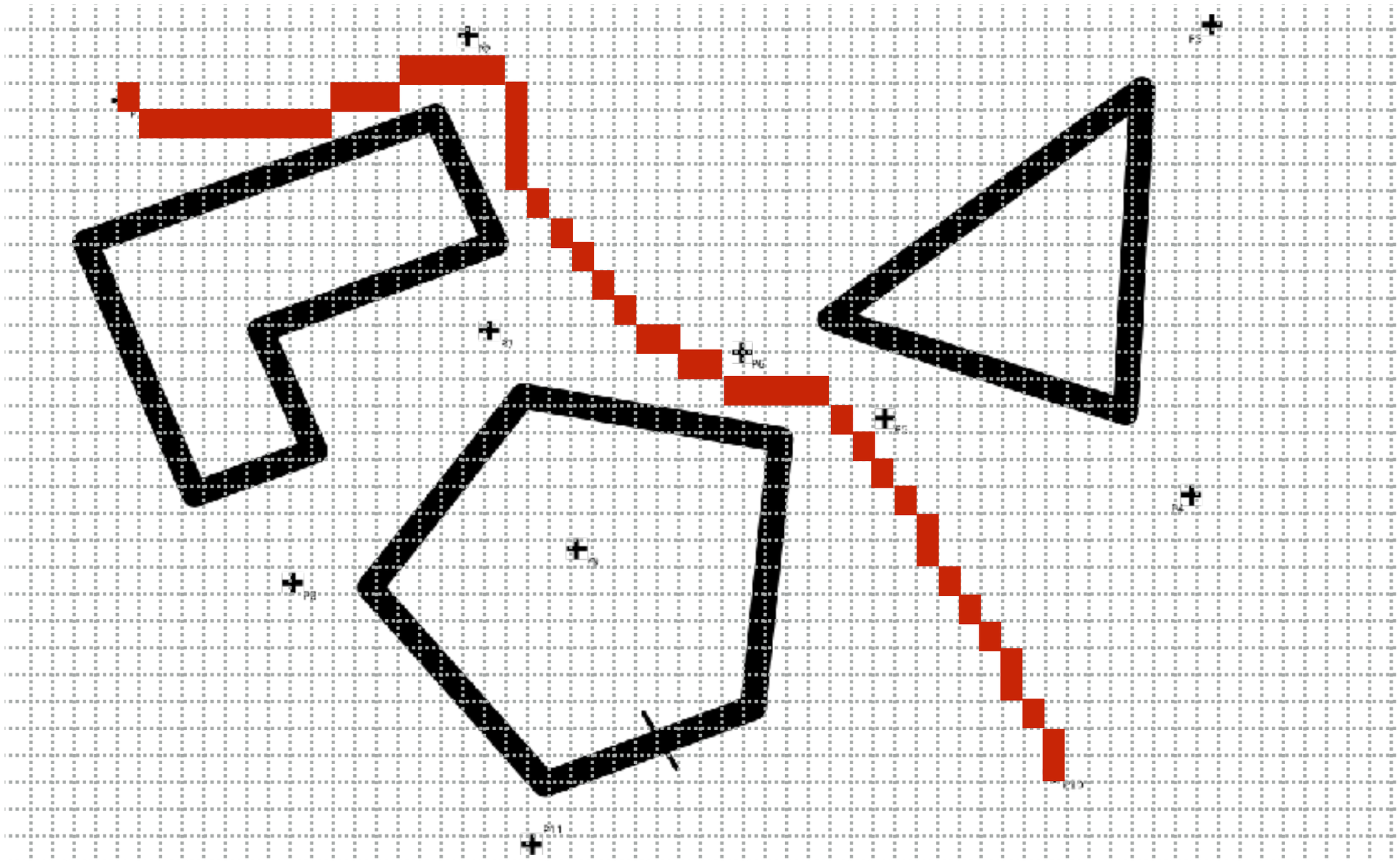
Projeto 6: Parte B

- Implemente o algoritmo A* com mapa de ocupação probabilístico e função de avaliação aditiva

- Utilize o algoritmo para encontrar trajetória ligando P1 a P10, e P1 a P9 (sem base do pentágono)
  - use mapa de células de tamanho 5x5cm e “linearização” de trajetória
  - teste configurações distintas para α e para número de convocações e discuta resultados

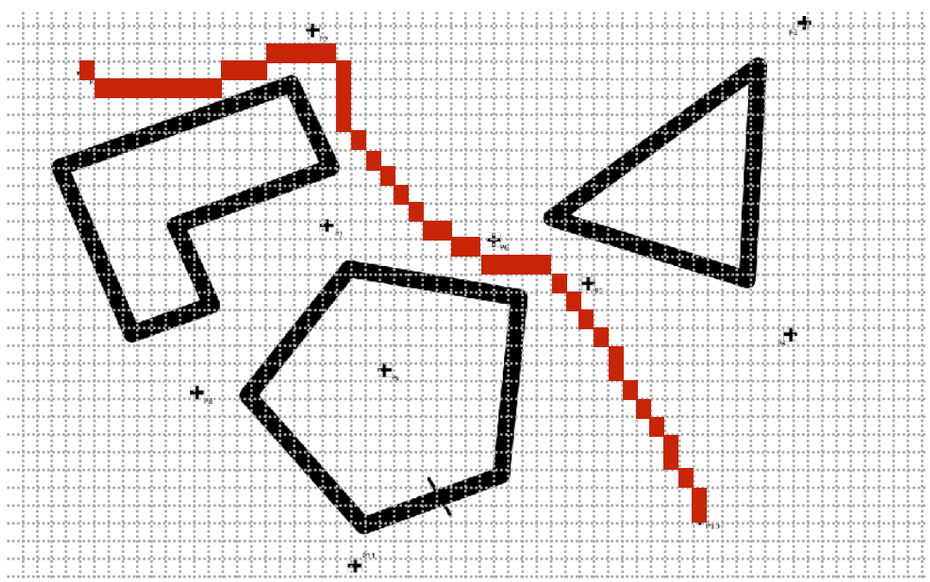
- Faça robô executar trajetória (meça distância à meta para diferentes representações)

trajetórias em células



representação por pose

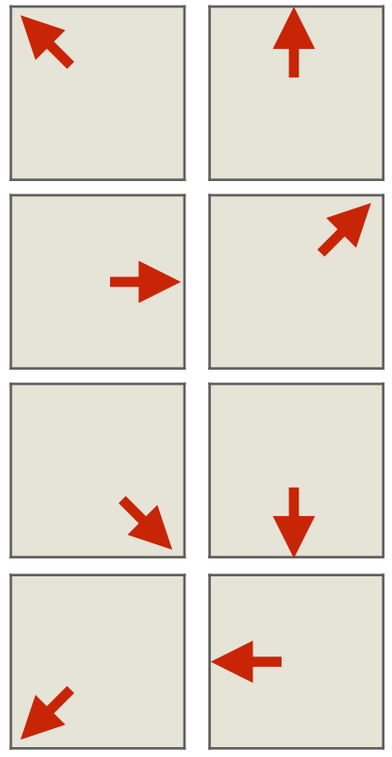
- Linearização de trajetórias foi obtida como pós-processamento
  - custo de trajetória linearizada pode ser muito maior que custo original
  - busca perde tempo com estados descartados posteriormente



representação por pose

- Linearização por representação de pose
  - ações de rotacionar a 0, 45, 90, 180, 270 graus
  - ações de mover-se a célula a frente (de acordo com orientação)
  - custo de rotacionar 15 graus > custo de rotacionar 45 graus
  - custo de rotacionar > custo de mover-se a frente

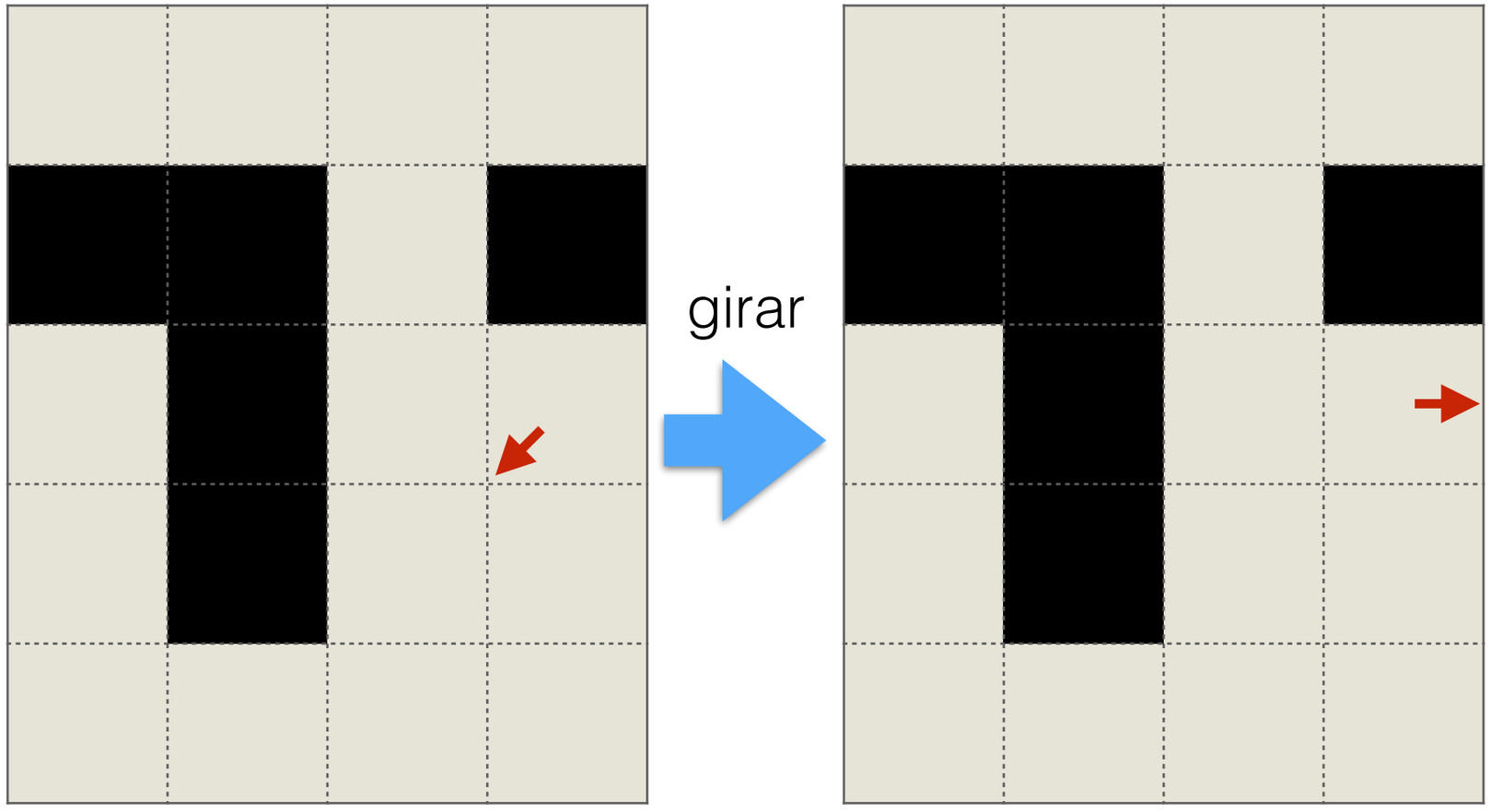
8 poses por localização



representação por pose

vizinhança:

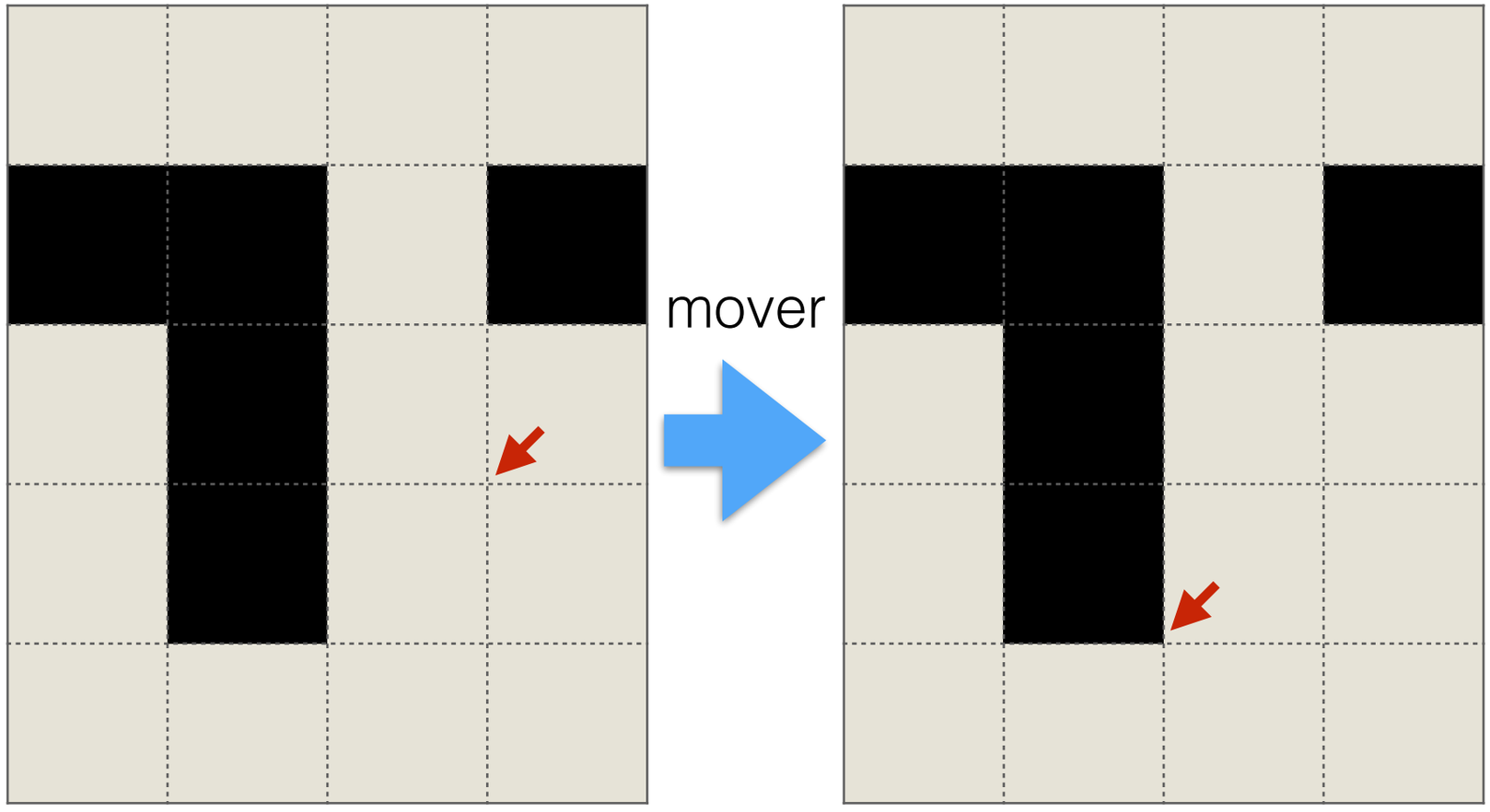
- mesma localização, poses distintas
- localizações adjacentes, mesma pose



representação por pose

vizinhança:

- mesma localização, poses distintas
- mesma pose, localizações “compatíveis”



Projeto 6: Parte C

- Implemente o algoritmo A* com representação por pose
- Utilize o algoritmo para encontrar trajetória de P1 a P9 (sem base do polígono)
  - mapa de células de tamanho 5x5cm e “linearização” de trajetória
  - teste diferentes custos para rotacionar
  - custo em linha reta é distância para percorrer
- Faça robô executar trajetória (meça distância à meta para diferentes representações)