Universidade Federal do Rio Grande do Sul
Instituto de Informática
INF01034 - Robótica II
Prof. Renan Maffei

Trabalho Prático 3 - Exploração Campos Potenciais

Prazo: 01/11/2020

Considerações gerais

O trabalho deverá ser feito preferencialmente em DUPLA (senão INDIVIDUALMENTE). A avaliação do trabalho será feita através da análise do funcionamento das implementações no simulador MobileSim. Entregue via Moodle um arquivo compactado com os códigos desenvolvidos, e posteriormente o trabalho deverá ser apresentado ao professor em horário a combinar.

Instruções sobre implementação do algoritmo de exploração no framework

O framework disponibilizado neste trabalho contém uma classe chamada `Grid` descrita no arquivo `Grid.cpp`, que dentre outras coisas define funções para desenhar o mapa na tela. Contém uma classe chamada `Robot` descrita no arquivo `Robot.cpp`, responsável pelo controle da movimentação do robô e mapeamento dos obstáculos. E contém também uma classe chamada `Planning` descrita no arquivo `Planning.cpp`, responsável por realizar a atualização do planejamento de caminhos do robô.
A versão atual do framework contém 6 modos de visualização do mapa, que podem ser alternados pressionando a tecla 'v' ou 'b'.

0. Mapa com LOG-ODDS usando LASER
1. Mapa com HIMM usando LASER
2. Mapa com classificação das células *(default)*
3. Mapa com Campo Potencial Harmônico
4. Mapa com Campo Potencial com Preferências
5. Mapa com Campo Potencial com Objetivos Dinâmicos próximos à paredes

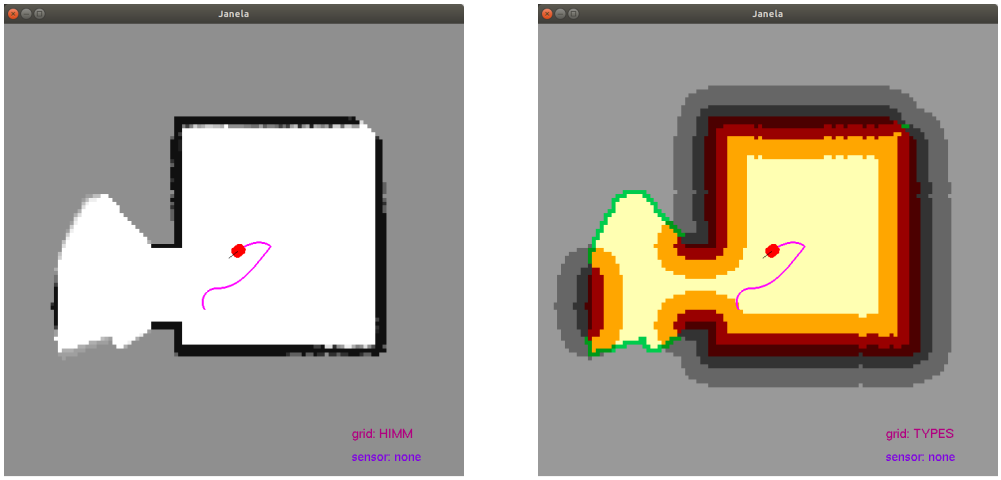
OBS: por enquanto todos os mapas estão vazios, pois as implementações de mapeamento e campos potenciais não estão inclusas.

Este trabalho precisa que ao menos uma técnica de mapeamento de ambientes (por exemplo, HIMM) e a classificação de células, pedidos no trabalho anterior, estejam funcionando adequadamente (Figura 1). Relembrando, as células devem estar classificadas quanto ao tipo de ocupação (`occType`), que pode ser:

- **não-explorada**: `c->occType = UNEXPLORED` (mostrada em tons de cinza)
- **livre**: `c->occType = FREE` (mostrada em amarelo bem claro)
- **obstáculo**: `c->occType = OCCUPIED` (mostrada em marrom escuro)

E também devem estar classificadas quanto ao tipo de função no planejamento (`planType`). Originalmente tinham sido definidos 3 tipos de células, agora foram incluídos mais 2:

- **fronteira**: `c->planType = FRONTIER` (mostrada em verde)



Mapa gerado com HIMM | Mapa com células classificadas

Figura 1. Exemplo de mapas

- **perigo**: `c->planType = DANGER` (mostrada em marrom)
- (NOVO) **próximas a obstáculos**: `c->planType = NEAR_WALLS` (mostrada em laranja)
- (NOVO) **fronteira próxima a obstáculo**: `c->planType = FRONTIER_NEAR_WALL` (mostrada em verde escuro)
- **regular**: `c->planType = REGULAR` (que não se enquadram nos tipos anteriores)

Será preciso adaptar (sem maior dificuldade) a função `updateCellsTypes` da classe `Planning`, que foi desenvolvida no trabalho anterior. As células próximas a obstáculos serão encontradas da mesma forma que se classificou as células de perigo, apenas deve-se aumentar a distância de verificação. Por exemplo, células com distância de 3 (células) podem ser classificadas como `DANGER`, enquanto que com distância de 4 a 8 podem ser classificadas `NEAR_WALLS`. Já na classificação de fronteiras, se a célula se encaixar na categoria de `NEAR_WALLS` e `FRONTIER`, deverá ser marcada como `FRONTIER_NEAR_WALL`.

A - Harmônico | B - Usando preferências | C - Objetivos dinâmicos

Figura 2. Estratégias de Campos Potenciais

Campos potenciais a serem implementados

Três estratégias de campos potenciais baseados no problema de valor de contorno (mostradas na Figura 2) serão implementadas:

A. Campo Potencial Harmônico

Condições de contorno (potencial fixo): **fronteiras** entre área desconhecida e espaço livre - potencial atrator; **obstáculos** - potencial repulsivo.
Movimento do robô: deve seguir o gradiente descendente do campo potencial. Robô para de se mover quando gradiente vira nulo (exploração completa).
Atualização: Potencial no espaço livre definido pela Equação de Laplace. É computado repetidamente usando o método de Gauss-Seidel enquanto o robô navega e as fronteiras mudam.

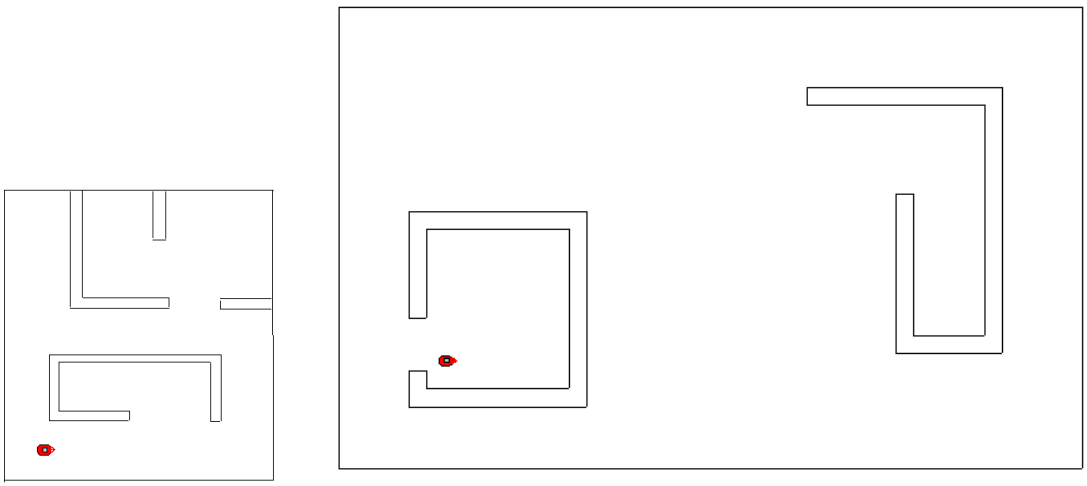
B. Campo Potencial com Preferências

Condições de contorno (potencial fixo): idem ao método A.
Movimento do robô: idem ao método A.
Atualização: Potencial no espaço livre considera regiões com diferentes preferências. Neste trabalho, regiões próximas a paredes terão preferência alta, enquanto que as demais regiões terão preferência baixa.

C. Campo Potencial com Objetivos Dinâmicos próximos à paredes

Condições de contorno (potencial fixo): fronteiras entre área desconhecida e espaço livre que estejam **próximas a paredes** - potencial atrator; **demais fronteiras e obstáculos** - potencial repulsivo.
Movimento do robô: idem ao método A.
Atualização: idem ao método A.

Dois ambientes estão disponíveis para realizar testes, conforme mostrados na Figura 3. O primeiro é um ambiente pequeno onde a exploração deverá ocorrer de forma rápida. Já o segundo é um ambiente esparso grande, onde será possível examinar com maior clareza as diferenças dos 3 métodos.



Mapa denso Mapa esparso

Figura 3. Exemplo de mapas

1. Implementação do método de inicialização dos campos potenciais

Complete a função `initializePotentials` da classe `Planning` para definir as condições de contorno dos campos potenciais. Essa função deve verificar todas as células dentro da área conhecida do grid (definida por gridLimits). Os três valores de potencial de cada célula estão definidos no vetor `c->pot`, onde `c->pot[0]` é o potencial harmônico (método A), `c->pot[1]` é o potencial com preferências (método B) e `c->pot[2]` o potencial com objetivos dinâmicos (método C).

Independente do método, células OCCUPIED e DANGER deverão sempre receber um **potencial repulsivo**, igual a 1.0.

Nos métodos A e B, células de fronteira (FRONTIER e FRONTIER_NEAR_WALL) deverão receber um **potencial atrator**, igual a 0.0. Além disso, por padrão nestes dois métodos, todas as células iniciam com potencial baixo, `pot[0]=pot[1]=0` (isso já está definido na classe Grid e não deve ser alterado).

Em contrapartida, no método C somente as células de fronteira próximas a paredes (FRONTIER_NEAR_WALL) recebem **potencial atrator** fixo (0.0) . As demais células de fronteira (FRONTIER), são fixadas com **potencial repulsivo** (1.0). Note que diferentemente dos métodos A e B, todas as células iniciam com potencial alto (`pot[2]=1`). Isto é feito para garantir que somente os "cantos" das fronteiras sejam atratores.

Os potenciais das demais células (que não se enquadram nessas classificações) não devem ser inicializados, pois eles serão computados iterativamente a partir dos valores anteriores. Se os valores forem erroneamente zerados, os campos potenciais provavelmente não convergirão de forma adequada.

Por fim, é nesta função que se deve **inicializar as preferências** das células (`c->pref`) que serão utilizadas no método B. Você deve escolher um valor de preferência **fixo positivo** para as células NEAR_WALLS e um valor **fixo negativo** para as demais células livres. Escolha valores simétricos entre 1 e -1 (isto é, $\pm$ 0.1; $\pm$ 0.2; $\pm$ 0.3, ...). Valores adequados devem ser encontrados empiricamente.

2. Implementação do método de atualização dos campos potenciais

Complete a função `iteratePotentials` da classe `Planning`. Note que cada chamada dessa função executa UMA ITERAÇÃO de atualização do campo potencial, ou seja, atualiza cada célula uma vez só. Porém, como essa função é chamada repetidamente dentro da função `run`, o campo potencial deverá convergir ao longo do tempo.

Após terem sido definidas as condições de contorno do campo potencial, basta atualizar o potencial das células FREE. Para isso varra todas as células dentro da área conhecida do mapa (limitada por `gridLimits`). Células que forem FREE devem ser atualizadas em função das células vizinhas usando diferenças finitas.

Os métodos A e C usam a definição original do potencial harmônico descrita no Algoritmo 1 da Aula 15.

$$p(\boldsymbol{c}_{i,j}) \leftarrow \frac{p(\boldsymbol{c}_{i,j+1}) + p(\boldsymbol{c}_{i,j-1}) + p(\boldsymbol{c}_{i+1,j}) + p(\boldsymbol{c}_{i-1,j})}{4} \tag{1}$$

Já o método B usa a equação atualizada definida no Algoritmo 2 da Aula 15, que

permite o uso de preferências no cálculo do potencial. A preferência da célula é dada por `c->pref`.

$$
\begin{aligned}
h(\boldsymbol{c}_{i,j}) &\leftarrow \frac{p(\boldsymbol{c}_{i,j+1}) + p(\boldsymbol{c}_{i,j-1}) + p(\boldsymbol{c}_{i+1,j}) + p(\boldsymbol{c}_{i-1,j})}{4} \\
d(\boldsymbol{c}_{i,j}) &\leftarrow \left(\left| \frac{p(\boldsymbol{c}_{i,j+1}) - p(\boldsymbol{c}_{i,j-1})}{2} \right| + \left| \frac{p(\boldsymbol{c}_{i+1,j}) - p(\boldsymbol{c}_{i-1,j})}{2} \right| \right) \\
p(\boldsymbol{c}_{i,j}) &\leftarrow h(\boldsymbol{c}_{i,j}) - \frac{pref(\boldsymbol{c}_{i,j})}{4} d(\boldsymbol{c}_{i,j})
\end{aligned} \tag{2}
$$

ATENÇÃO: Não se deve misturar potenciais diferentes durante o cálculo. Por exemplo, caso se esteja calculando o potencial A de uma célula, deve-se considerar os potenciais A das células vizinhas, e não B ou C. O mesmo se aplica aos demais potenciais.

3. Implementação do método de atualização do gradiente dos campos potenciais

Complete a função `updateGradient` da classe `Planning`.

Nesta etapa é preciso computar o vetor gradiente descendente ($-\nabla p$) de cada um dos três potenciais das células FREE, através de uma varredura igual a feita no exercício anterior. Cada célula (`Cell* c`) do grid tem os vetores gradiente descritos pelas componentes horizontais e verticais: `c->dirX[i]` e `c->dirY[i]`, onde `i=0,1,2` indica a qual potencial se refere.

$$
-\nabla p(\boldsymbol{c}_{i,j}) \leftarrow \left(\underbrace{-\frac{p(\boldsymbol{c}_{i+1,j}) - p(\boldsymbol{c}_{i-1,j})}{2}}_{dirX}, \underbrace{-\frac{p(\boldsymbol{c}_{i,j+1}) - p(\boldsymbol{c}_{i,j-1})}{2}}_{dirY} \right)
$$

Lembre de normalizar o vetor gradiente pois para a navegação só nós interessa saber a direção do gradiente, não sua intensidade. Ou seja, divida as componentes horizontais e verticais pela norma do gradiente $|\nabla p|$ (desde que $|\nabla p| \neq 0$),

$$
|\nabla p| \leftarrow \sqrt{(dirX)^2 + (dirY)^2}
$$

Por fim, não é preciso computar o gradiente nas células diferentes de FREE pois elas não são navegáveis, então basta colocar `c->dirX[i]` e `c->dirY[i]` iguais a 0.

Para visualizar o gradiente, aperte a tecla ‘f’. O gradiente correspondente aparecerá em cada um dos campos potenciais sendo mostrado.

4. Implementação do método de navegação seguindo o gradiente do campo potencial

Nesta nova versão do framework há 3 novos modos de navegação, POTFIELD_A, POTFIELD_B e POTFIELD_C, que podem ser ativados pressionando as teclas ‘5’, ‘6’ e ‘7’,

respectivamente [1]. Nesses modos cabe ao robô seguir o gradiente descente do campo potencial escolhido na célula que ele se encontra sobre.

Isso deverá ser feito completando a função `followPotentialField` da classe `Robot`. A ideia básica é computar a diferença de orientação do gradiente (usando $atan2$) em relação à orientação do robô (θ) e aplicar uma técnica de controle sobre a velocidade angular do robô para mover o mesmo de acordo com o campo potencial.

Primeiramente compute essa diferença de orientação ϕ (e lembre-se de normalizá-la[2] posteriormente entre $-180°$ e $180°$):

$$\phi = RAD2DEG(atan2(dirY[i], dirX[i])) - \theta$$

Conhecendo o ângulo ϕ use um controlador estudado em aula para variar a velocidade angular ω do robô, mantendo fixa a velocidade linear. Um controlador do tipo P já é bom o suficiente (mas se quiser pode usar outro):

$$\omega = \tau_p \cdot \phi$$

Dica 1: Quando o ângulo ϕ for muito grande, i.e. quando o gradiente apontar em uma direção oposta à orientação do robô (o que geralmente acontece quando o robô está indo em direção a um obstáculo), o controlador com velocidade linear fixa pode não ser suficientemente rápido para evitar colisões. Nesse caso, é bom verificar manualmente se $|\phi|$ é muito grande (maior que um dado limiar α), e rotacionar o robô sobre o próprio eixo.

Dica 2: Às vezes, a direção do gradiente pode mudar de forma brusca, especialmente no método B (com preferências). Mudanças bruscas são ruins para a movimentação do robô. Você pode minimizar esse efeito pegando a média das componentes dirX[i] e dirY[i] de várias células em uma janela (pequena) ao redor da célula do robô, ao invés de pegar as componentes de uma única célula.

[1] As teclas de 1 a 4 ativam os modos de navegação implementados no primeiro trabalho.
[2] Dica: use a mesma função de normalização do trabalho anterior