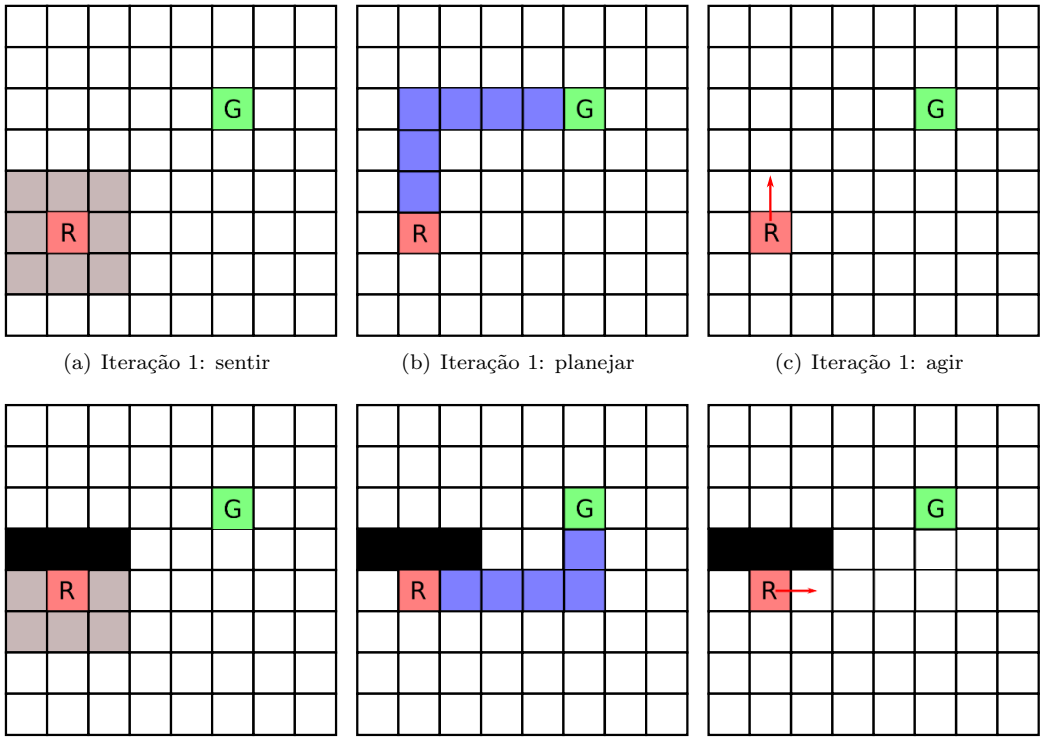
Programação robótica - UFPE 2022.1
Projeto 1: A linguagem Python

Descrição do trabalho

Este projeto tem como objetivo simular um robô omnidirecional navegando em um ambiente contendo obstáculos. Desde sua posição inicial, o robô precisa atingir um determinado objetivo, evitando obstáculos. Para detectar os obstáculos, o robô está equipado com um laser.

- O ambiente é representado por uma grade. Se uma célula estiver vazia, seu valor será 0. Se houver um obstáculo, estático ou móvel, seu valor será -1.
- O robô tem um sensor que pode perceber até n_l células. n_l é o campo do sensor.
- Inicialmente o robô não conhece o ambiente. Ele pode atualizar sua mapa cada vez que ele deteta um obstaculo.
- A implementação da tarefa de navegação seguirá a metodologia de controle **SPA**:

Sense → Plan → Act
Sentir → Planejar → Agir



(a) Iteração 1: sentir (b) Iteração 1: planejar (c) Iteração 1: agir

(d) Iteração 2: sentir (e) Iteração 2: planejar (f) Iteração 2: agir

Figura 1: Exemplo de SPA sobre duas iterações e o campo do sensor $n_l = 1$

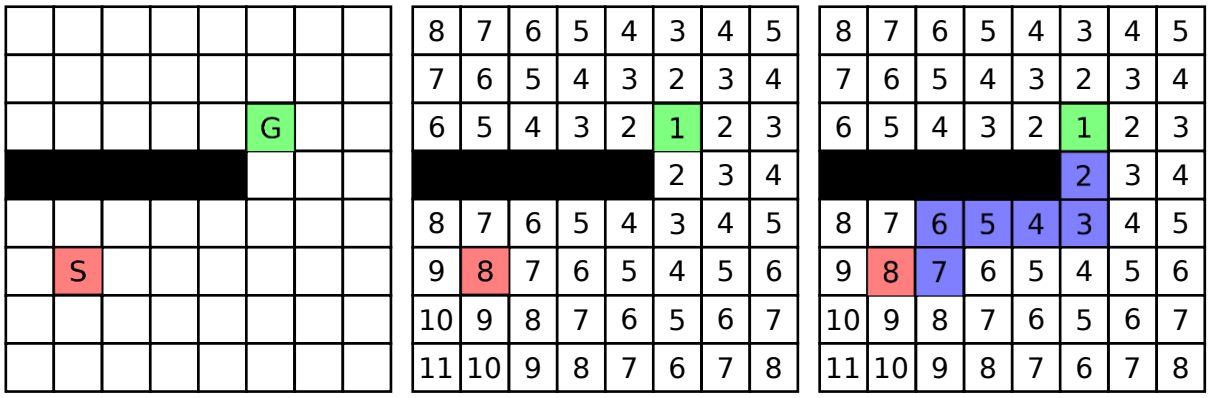
Descrição das tarefas

- Criar uma classe **ambiente**.
  - Atributos: dimensões da grade (número de linhas e de colunas).
  - Atributos: grade representando o ambiente.
  - Método: adicionar um obstáculo no ambiente.
  - Método: remover um obstáculo no ambiente.
- Criar uma classe **robô**.
  - Atributos: dimensões da grade (número de linhas e de colunas).
  - Atributos: posição (número da linha e da coluna).
  - Atributos: posição desejada (número da linha e da coluna).
  - Atributos: o campo do sensor.
  - Atributos: grade representando o ambiente conhecido pelo robô.
  - Atributos: a caminho planejado para atingir a posição desejada.
  - Método: definir a posição desejada.
  - Método: sentir os obstáculos no campo do sensor e atualizar a mapa.
  - Método: planejar o caminho para chegar na posição desejada (o planejamento pode falhar se não houver caminho).
  - Método: mudar o robô para a próxima posição.
  - Método: detectar colisões com os obstáculos.
- Crie uma classe **obstaculo móvel** e incluir pelo menos dois obstáculo móveis no cenário anterior.
  - Atributos: a ser definido pelos alunos.
  - Método: a ser definido pelos alunos.
  - Um obstaculo move aleatoriamente (ver NumPy/rand).
- Comentários:
  - Usar um arquivo para descrever a dimensão do ambiente, as células inicial e desejada. Escrever cada caminho planejado num arquivo único.
  - Para cada iteração, o ambiente conhecido pelo robô e o caminho planejado deve ser representado graficamente (ver a biblioteca plotLib.py)

Planejamento de caminho

Nesse projeto, é proposta usar o método do campo de potenciais para planejar o camhino.

- Uma grade representa o ambiente.
- A célula vermelha é a situação inicial e a verde é a desejada.
- As células pretas representam os obstáculos.
- Para calcular o campo de de potenciais, a célula verde recebe o valor 1.
- Para a célula com o valor 1: cada célula livre à direita, à esquerda, na parte superior e na parte inferior recebe o valor 2.
- Para cada célula com o valor 2: cada célula livre à direita, à esquerda, na parte superior e na parte inferior recebe o valor 3.
- Continue até encher a grade.
- Para planejar o caminho, começar com o valor da célula inicial.
- O primeiro passo do caminho é uma célula à direita, à esquerda, na parte superior ou na parte inferior que tem um valor menor.
- Repetir até atingir a célula desejada.



(a) Grade representando o ambiente (b) Cálculo do campo de potenciais (c) Planejamento do caminho

Figura 2: Respostas de um sistema de segunda ordem ao degrau unitário

Etapas que devem ser validadas pelo professor

- Implementação da classe **ambiente** e a visualização via a função `plotRealEnv`
- Implementação dos métodos `mudar` e `sentir` da classe **robô** e a visualição via a função `plotRobEnv`
- Implementação do método `palnejar` da classe **robô** e a visualização via a função `plotPlanningMap`
- O robô navegando em um ambiente com obstáculos estáticos
- O robô navegando em um ambiente com obstáculos dinâmicos

Recursos

- Thonny: um software para programar usando Windows/Linux/Mac link
- Instalar Matplotlib e Numpy: Thonny → /Tools/Manage → digitar Numpy ou Matplotlib → Search → Install
- Matplolib/matshow: traçar uma grade clique para ver um exemplo
- NumPy/array: criar e trabalhar com um `array` clique para ver um exemplo
- NumPy/rand: Como gerar números aleatórios clique para ver um exemplo