UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMATICA
SISTEMA INTELIGENTES
2017.1

BUSCA DE CAMINHO COM ROBORES.

EQUIPE:
JOAO PEDRO CAVALCANTI UCHOA – JPC2

Projeto

Desafio

Tem como objetivo construir um algoritmo que mova um robô em um mapa de um ponto inicial ao seu objetivo no menor tempo. Disponibiliza-se para isso de um robô com rodas, motor, e um sonar como um dispositivo de navegação.
Os obstáculos do espaço em que o robô está situado é desconhecido a ele, logo não é possível obter uma solução antes de começar o movimento.

Teoria de Implementação

Uma solução possível e ótima para problemas de distancias deste tipo é o algoritmo A*, no qual é usado na a busca em nós. Antes de se aplicar A* se transforma o espaço continuo em pedaços (tiles), idealmente um tile terá o tamanho do robô, pois este não pode passar por lugares menores que seu tamanho, porem o tamanho da tile pode ser expandida para ganho de performance com pouco custo em velocidade de solução.

Não se sabe como são as características físicas do ambiente aonde o robô esta situado em relação a obstáculos, logo é necessário fazer reconhecimento da área e memorizar quais são e aonde estão os obstáculos, para isso se utiliza uma matriz para armazenar a localização das paredes da área. Esta matriz é inicializada com todos valores 0, e quando o robô descobre o obstáculo, ele marca 1 na posição, isto será utilizado no futuro para se transformar os espaços vazios em nós.

Será implementado então um algoritmo A* dinâmico calculando a melhor rota contando com os obstáculos já obtidos. A cada tile percorrida será recalculada a rota, pois se obtém mais informações de novos obstáculos quando o robô se movimenta.

Implementação

Como a implementação de A* ser mais complexa em C++ (devida a necessidade de atenção com garbage collection), o projeto utilizará a biblioteca AriaPy sendo desenvolvido em Python.

A matriz representará o espaço real, sendo ela discreta, utilizando um tamanho de tile de 51cm (um pouco a mais do comprimento do robô), será utilizado um offset de metade do tamanho da matriz, para armazenar coordenadas negativas.

$$Xmap = \frac{Xreal}{Tsize} + Offset \qquad Xreal = (Xmap - Offset) \times Tilesize$$

FIG 1 - Equação de conversão entra coordenadas reais e coordenadas da matriz.

Para testar se os caminhos estavam sendo propriamente mapeados foi inserido o código de mapeamento em um exemplo da biblioteca Aria e a partir disso obtido dados na matriz. Transformando a matriz em string, feita substituição via regex de 0's e 1's para uma melhor visualização se obteve-se o resultado da figura 2.

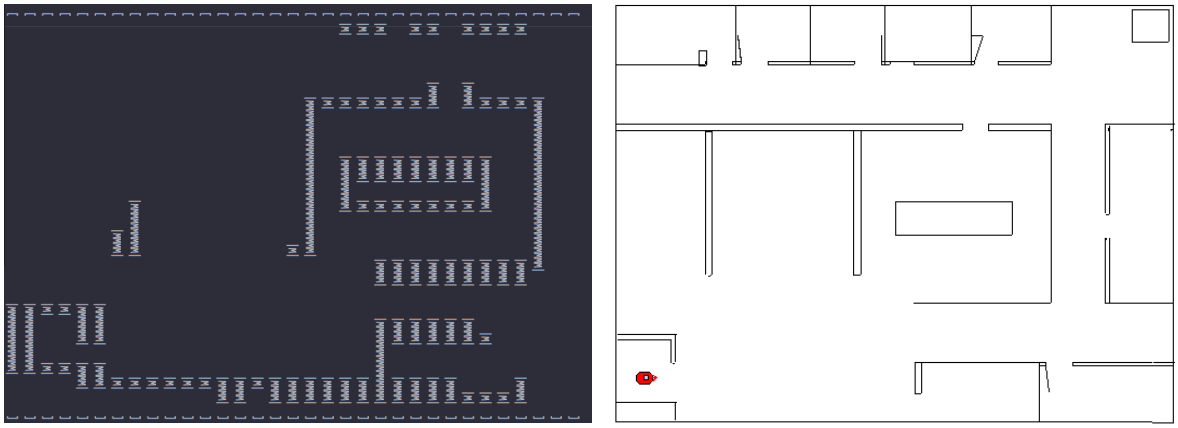

FIG 2 – Mapa parcial armazenado em matriz (esquerda) e mapa original (direita).

A partir deste experimento fica visto que o sensor não é omnidirecional e contem variância relativamente alta. Logo é importante balancear o valor das tiles, pois uma tile muito pequena o dispositivo conseguirá passar, e muito grande ele fechará caminhos que o dispositivo poderia passar.

Com o auxilio da biblioteca "heapq" para ordenar a prioridade baseada em heurística, foi implementado o algoritmo A Star, do resultado foi tirado então a próxima tile a se mover e utilizado o comando `setGoal()` do Aria para essa finalidade.

Resultado

Foi terminada a implementação do código e obtido êxito em achar o caminho final para o robô caso fosse possível achar um, devido a variações do sensor, o programa ocasionalmente acusa lugares vazios perto de obstáculos de obstáculo, o problema é mitigado se diminuir o tamanho da tile, mais devido ao tamanho do robô, e a simplicidade do algoritmo “setGoal”, este faz com que o robô bata nas paredes em curvas muito abertas.

Algo que se limita a solução de ser melhor é devido ao sensor ter um alcance bastante limitado, e apontar para poucas direções o que diminui a eficácia de busca: o robô entra em salas vazias e percorre ela, até voltar e procurar outro caminho, este problema é resolvido com a implementação de laser.

A figura 3 demostra o mapa final dos obstáculos reconhecidos pelo robô e o caminho percorrido por ele ao objetivo final, o robô se movimenta diretamente em direção ao objetivo (marcado com D), e logo se depara com a parede, pois o sensor só detecta paredes se estiver em movimento, ele então faz o retorno para a abertura e vai diretamente em linha reta novamente ao objetivo, ele então entra na sala e quando as paredes chegam ao seu alcance faz a volta novamente, desta vez seguindo pelo caminho quase ideal a solução ótima.

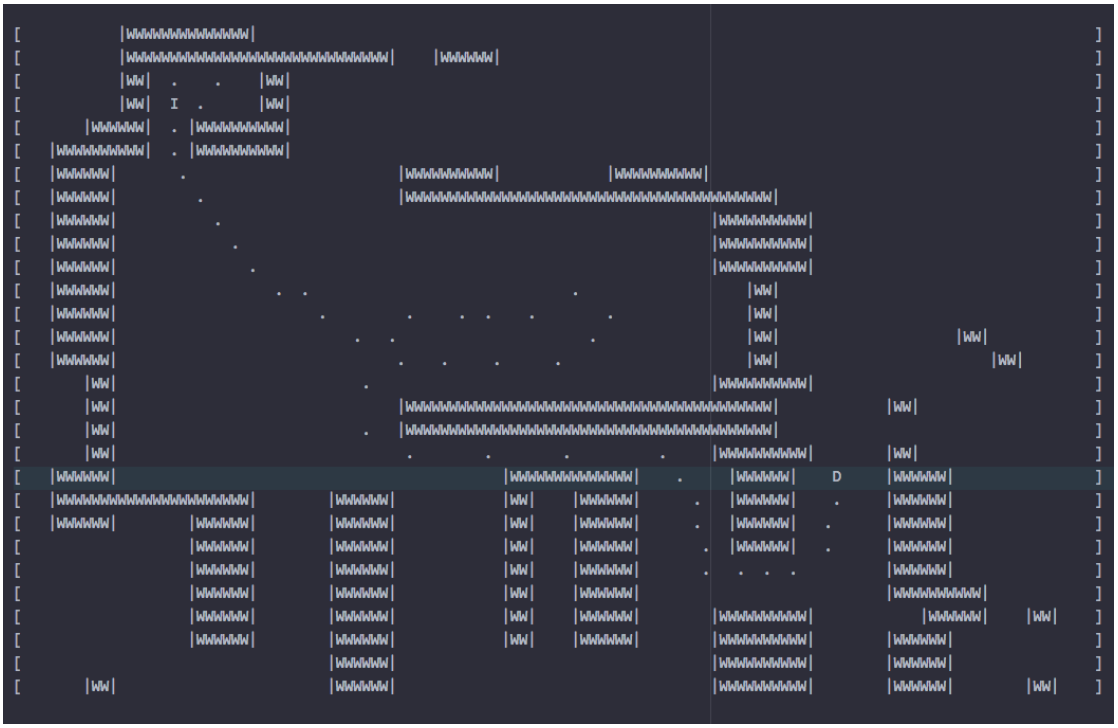

FIG 3 – Caminho percorrido pelo robô, em situações ideais do sensor.