FACULDADE INDEPENDENTE DO NORDESTE – FAINOR
CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

Alisson Oliveira dos Anjos
Karine Ferreira Araújo
Tarcísio de Freitas Botelho

PROJETO UNIDADE 3

Vitória da Conquista, 2021

Alisson Oliveira dos Anjos

Karine Ferreira Araújo

Tarcísio de Freitas Botelho

ATIVIDADE AVALIATIVA - UNIDADE 3

Pesquisa apresentada como requisito para nota da III Unidade da componente curricular Robótica, do curso de Engenharia da Computação, na Faculdade Independente do Nordeste em 2021 Prof. Ms. Marcos Gomes Prado.

Vitória da Conquista, 2021

Sumário

1 INTRODUÇÃO

A evolução e inovação de máquinas inteligentes vêm sendo constante em nosso meio, pode-se ver claramente isso em fábricas, onde empregados são substituídos por máquinas inteligentes/robô industrial, quando é possível e/ou necessário. De acordo com Ceroni (2000), robô industrial é um sistema mecânico que pode ser programado para desempenhar uma grande variação de tarefas (locomoção, manipulação) sob o comando de um controle automático. Segundo Rosário (2005), o tipo de acionamento de um braço robótico pode ser elétrico, hidráulico e/ou pneumático.

O termo robô conforme indica a RIA(Associação Internacional de Robótica) possui uma definição aceita, abrangendo as características de um braço humanoide que é um manipulador reprogramável multifuncional, sendo capaz de movimentar peças, materiais, ferramentas e dispositivos em diversas direções. Além disso, apresenta estrutura física com geometria variada, corpo rígido e articulações entre os mesmos com a finalidade de sustentar, direcionar e orientar a ferramenta terminal.

É através de sua aplicabilidade ou atividade desenvolvida que o robô irá possuir uma especialidade na função em que vai assumir, no ambiente que será inserido (QUARESMA, 2012). Portanto, a definição mais simples de um robô, é um tipo de manipulador, que possui várias funções propositais, que poderá ser controlada manualmente através de controles e sensores inerciais ou automáticos (FERREIRA e ALVES, 2013).

O robô é constituído de um corpo rígido formado por juntas rotativas e ou prismáticas proporcionando uma certa cadeia cinemática, possuindo vários graus de liberdade dependendo da necessidade, que lhe for atribuída. Uma extremidade do braço robótico encontra-se fixado em uma base e a outra extremidade possui um efetuador, que se movimenta livremente pelo espaço onde foi locado para efetuar a tarefa, que lhe foi atribuída (QUARESMA, 2012).

A robótica é uma área que está em constante crescimento e evolução, causando um grande impacto nos diversos setores da sociedade, e também muito utilizada na realização de uma variedade de tarefas, dentre elas: reconhecimento de ambientes, agricultura, diversos setores da indústria (automotiva, alimentícia, farmacêutica, entre outras), instalações militares e ambientes que podem ser nocivos ao ser humano (WOLF et. al, 2009).

Robótica móvel é uma subárea da robótica que vem apresentando muitos avanços no seu campo de atuação. São robôs de menor tamanho e que podem conter sensores ou atuadores.

Segundo Vieira, um robô móvel é composto por uma arquitetura de hardware e software que é responsável por garantir a execução correta de sua navegação para uma determinada tarefa (VIEIRA, 2005). Assim, justifica-se sua utilização na sociedade no uso de aplicações domésticas, industriais, urbanas e militares.

O simulador de robô CoppeliaSim®, com ambiente de desenvolvimento integrado, é

baseado em uma arquitetura de controle distribuída: cada objeto/modelo pode ser controlado individualmente por meio de um script embutido, um plugin, ROS ou BlueZero, um cliente API remoto ou uma solução customizada. Isso torna o CoppeliaSim® muito versátil e ideal para aplicações multi-robôs. Os controladores podem ser escritos em C/C ++, Python, Java, Lua e etc.

É modelos personalizados podem ser criados usando uma grande variedade de atuadores e grande variedade de atuadores e sensores embutidos atuadores e sensores embutidos, articulações, formas e malhas, scripts, etc. O V-REP está em um estágio avançado e contínuo de desenvolvimento e tem um fórum de apoio muito ativo.

CoppeliaSim® é usado para desenvolvimento rápido de algoritmos, simulações de automação de fábrica, prototipagem e verificação rápida, educação relacionada à robótica, monitoramento remoto, verificação dupla de segurança, como Digital Twin® e outros. O V-REP é dividido em três versões, as quais podem ser citadas com as seguintes particularidades:

• V-REP PRO EDU - pode ser utilizado por estudantes, professores, escolas e universidades de forma gratuita e sem limitações (totalmente funcional), não necessitando de registro. Entretanto, não pode ser empregado em aplicações comerciais, instituições de pesquisa e organizações sem fins lucrativos;

• V-REP PRO - pode ser empregado sem limitações (totalmente funcional) em empresas, instituições de pesquisa e organizações sem fins lucrativos, tendo um custo/valor associado. Além disso, pode ser usado em aplicações comerciais;

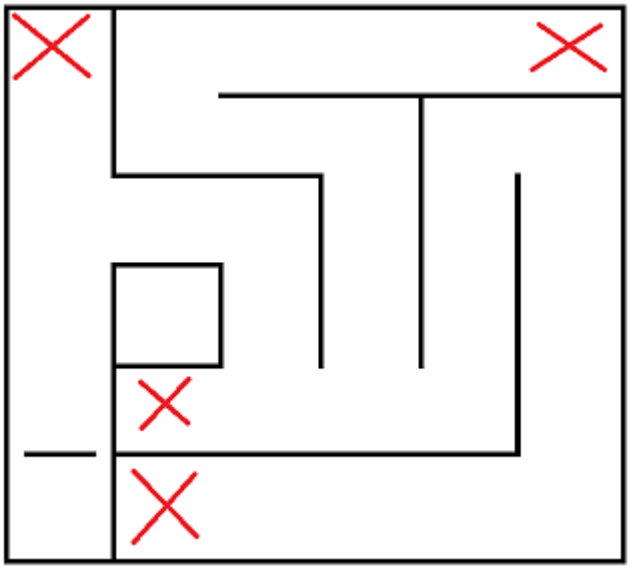
• V-REP PLAYER - pode ser usado por qualquer pessoa de forma gratuita para qualquer aplicação. No entanto, com capacidade limitada de edição, em que a opção "salvar" está desabilitada

Por ser um software de simulação multi plataforma desenvolvido pela Coppelia Robotics GmbH. Ela permite personalização completa da simulação através de diferentes abordagens (add-ons, plugins, scripts, etc.) e plugins, scripts, etc.) e suporta quatro diferentes dinâmicas/físicas motores dinâmicos/físicos: Bullet, ODE, Newton e Vortex.

2 OBJETIVO

Esse trabalho teve por objetivo programar um robô móvel autônomo no ambiente de simulação V-rep, capaz de alcançar quatro posições em um labirinto partindo do canto inferior esquerdo. Como mostra a seguinte figura:

Figura 1 - Labirinto proposto



Fonte: dos autores, 2021.

3 DESENVOLVIMENTO

3.1 Configuração da api remota

Inicialmente foi configurada uma api remota para a utilização da linguagem Python no *software* de simulação V-rep para isso é preciso alterar um script da linguagem lua localizado na seguinte pasta: C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\lua. Após isso é necessário apenas abrir o arquivo defaultMainScript.lua em algum editor de texto e adicionar o comando "simRemoteApi.start(19999)" na função function sysCall_init() conforme mostra o código a seguir:

Quadro 1 - Código para o arquivo

1	function sysCall_init()

2	sim.handleSimulationStart()
3	sim.openModule(sim.handle_all)
4	backCompatibility.handle(0)
5	simRemoteApi.start(19999)
6	end

Fonte: dos autores, 2021.

O código inserido no arquivo do V-rep encontra-se na linha 5 do quadro acima. É importante salientar que este processo pode mudar em versões diferentes do V-rep, a versão utilizada para este trabalho foi a 4.2.0, Windows 64 bit.

Após isso foi criada a pasta do projeto com os seguintes arquivos:

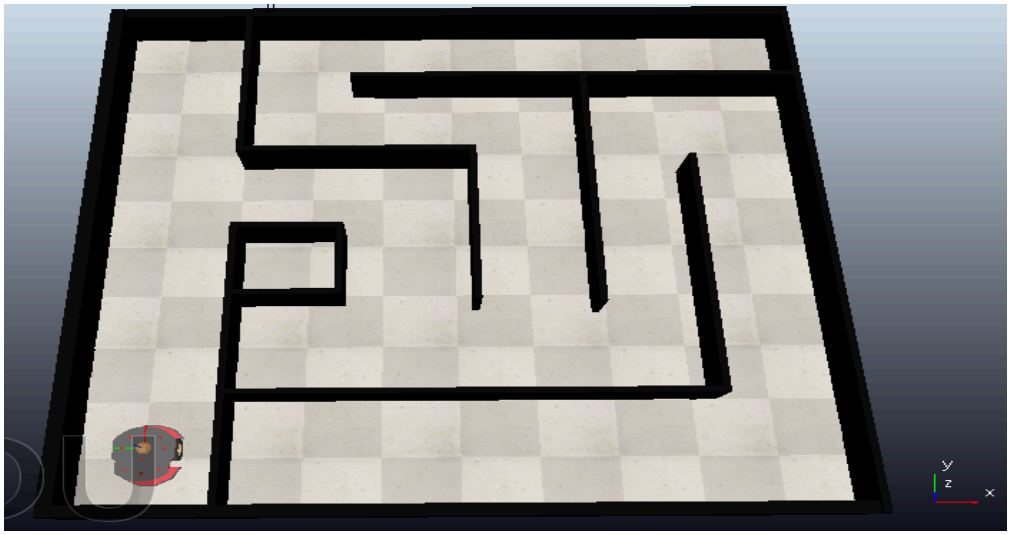
- remoteApi.dll - arquivo copiado da pasta "C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\lib\lib\Windows";
- sim.py; simConst.py - os dois copiados da pasta "C:\Program Files\CoppeliaRobotics\CoppeliaSimEdu\programming\remoteApiBindings\python\python".

Em seguida na mesma pasta foi criado o arquivo floodFill.py para a codificação do robô móvel autônomo.

3.2 Configuração do ambiente de simulação

Nesta etapa foi criado no V-rep um labirinto conforme o proposto pelo professor, utilizando formas primitivas cubóides fornecidas pelo simulador para criar as paredes do labirinto, em seguida foi escolhido como robô móvel o modelo Pioneer_p3dx. Conforme mostra a figura a seguir:

Figura 2 - Ambiente de simulação



Fonte: dos autores, 2021.

3.3 Codificação

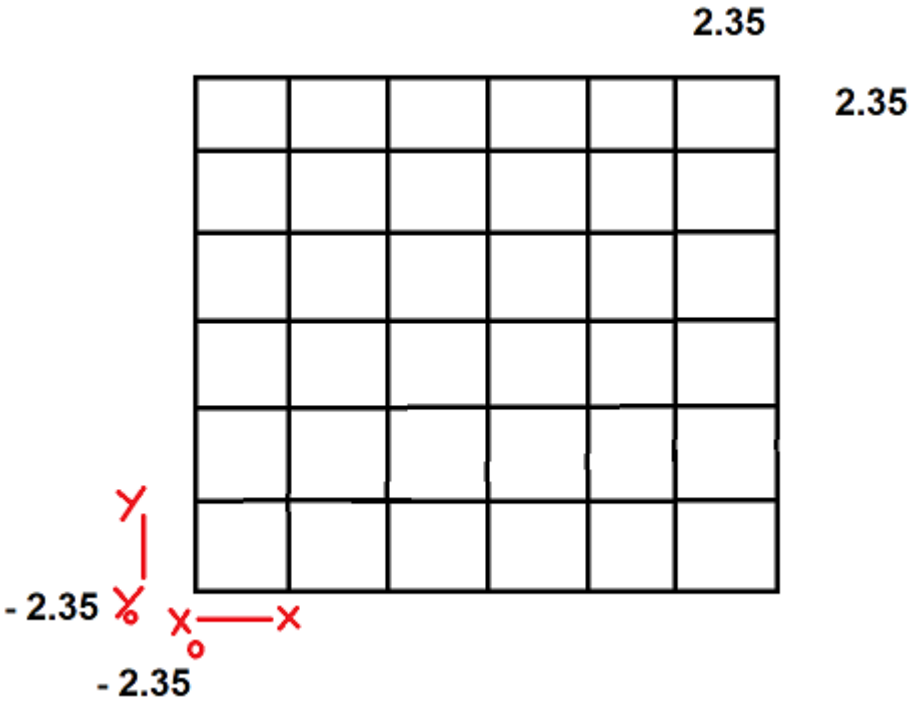
Esta etapa foi dividida em três fases de codificação, foram elas:

- Movimentação básica do robô - Código de movimentação e rotação do robô nas células da matriz do labirinto ;
- Evitar colisões - Utilização de sensores ultrassônicos para de detectar as paredes do labirinto;
- Algoritmo de exploração - Código para realizar a exploração do labirinto em busca das posições propostas como objetivos.

3.3.1 Movimentação básica do robô

A movimentação do robô consistiu em realizar dois movimentos, uma rotação de 90 graus para à direita ou esquerda e um “passo para frente”, que significa o deslocamento linear do robô no ambiente, limitado pela distância de uma célula da matriz de movimentação. A matriz de movimentação foi criada utilizando valores fornecidos pelo V-rep que atuam na forma de gps, o ambiente então foi dividido em 36 partes para obter a localização do robô no labirinto utilizando uma matriz 6 por 6 contendo os limites x e y de cada célula. Como mostra a seguinte figura:

Figura 3 - Criação de matriz de movimentação

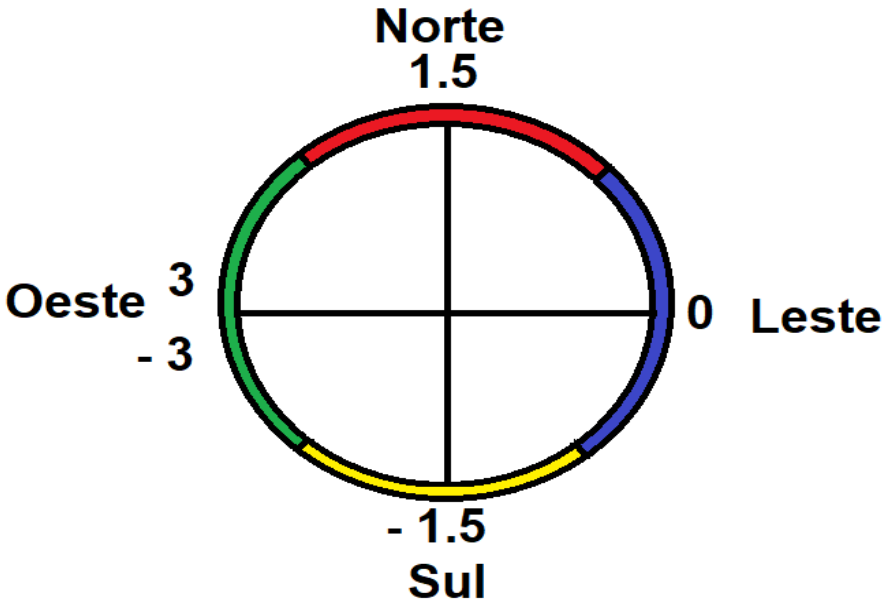


Fonte: dos autores, 2021.

A posição de origem do robô (linha 0 coluna 0 no canto inferior esquerdo), por exemplo, é delimitada pelos valores -2,35 e -1,55 no eixo x, e por -2,35 e -1,55 no eixo y. O simulador ainda permite obter a posição absoluta do robô no ambiente e assim, verificar sua posição na matriz de movimentação.

Outra informação necessária para a movimentação do robô é a sua orientação, pois definir direções como direita ou esquerda, para trás ou para frente numa matriz, depende diretamente da orientação do robô (Norte, Sul, Leste, Oeste). Para isso foi utilizada uma função do V-rep que fornece a orientação do objeto mas em forma de números, variando de -3 a 3. As orientações poderiam ser interpretadas por valores únicos como por exemplo 1,5 para Norte e -1,5 para Sul, mas essa abordagem permite erros em situações onde o robô rotaciona mais ou menos que o desejado, devido a atrasos de comunicação entre a api Python e o V-rep, para evitar isso foram definidas faixas de valores que representam as orientações. Como é ilustrado na figura a seguir:

Figura 4 - Obtenção da orientação do robô



Fonte: dos autores, 2021.

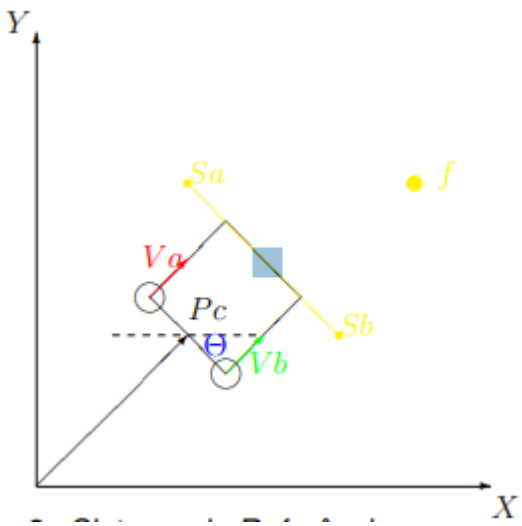
A partir dessas informações obtidas, foram criadas quatro funções para realizar a movimentação, são elas:

- axialRotation (direction, orientation): função de rotação do robô que recebe como entrada ou parâmetro a direção escolhida em forma de uma *String* e a orientação fornecida pelo V-rep, a função interpreta essa orientação e de acordo com a direção escolhida, liga os motores das rodas (uma delas de forma inversa) até que a orientação mude;
- currentCell(objectAbsolutePosition): função que recebe como entrada a posição absoluta fornecida pelo gps do V-rep e retorna a posição xy do robô referente à matriz de movimentação;
- nextCell(x, y, orientation): função que utiliza a posição na matriz de movimentação e a orientação do robô e retorna a célula da matriz de movimentação localizada logo à frente do robô;
- fwdStep(objectAbsolutePosition, orientation): função que recebe como entrada a posição do robô fornecida pelo gps e sua orientação, a partir dessas informações os motores das rodas são acionados até que a posição do robô esteja entre os limites da célula localizada à sua frente, limites esses fornecidos pela função anterior.

3.3.2 Evitar colisões

Os Veículos de Braitenberg são uma série de experimentos mentais nos quais veículos cada vez mais complexos são construídos a partir de versões mais simples. Suas estruturas são relativamente simples, mas dão origem a comportamentos complexos que podem ser identificados como reações de medo, valores, agressividade e etc.

Um sistema de referência apresentado na figura a seguir, representa um veículo em seu ambiente simulado, permitindo uma interação com fontes de estímulo.



Algoritmo simplificado

1) Calculo da posição de cada componente do carro e das fontes de luz

2) Desenho do ambiente

3) Calculo das intensidades medidas pelos sensores

4) Excitação/Inibição dos motores (cálculo das velocidades)

5) Calculo da cinemática do sistema (novas posições e ângulos)

6) Retorne ao passo 1

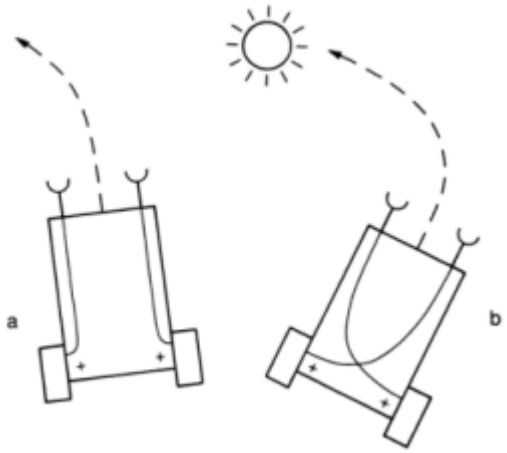
Um veículo simulado possui um sensor ligado a um motor. Quanto maior for a intensidade medida pelo sensor, maior será a velocidade do motor que o sensor alimenta. Se o sensor for

sensível a calor e houver uma fonte de calor por perto, o carro irá se movimentar tão rápido quanto mais perto o sensor estiver da fonte. Ele irá desacelerar quando estiver longe da fonte de calor.

Veículo 1

Para um observador que não sabe do funcionamento interno desse veículo, o comportamento apresentado passa a ideia de que se trata de uma criatura viva, que não gosta de regiões muito quentes.

Outro tipo de veículo, possui dois sensores e dois motores. Considerando o lado esquerdo do veículo como sendo o lado A e o direito como sendo o B, é possível ligar o sensor B no motor B e o sensor A no motor A ou o sensor B no motor A e o sensor A no motor B. É possível ligar também o sensor A nos motores A e B e o sensor B nos motores A e B, o que produziria o mesmo efeito de um veículo do tipo apresentado anteriormente.



Veículo 2

O efeito de Veículo 2a é que o veículo tende a acelerar sempre que se encontra perto de algo que ative seus sensores. Se existe uma diferença nas intensidades medidas, como no caso de o veículo estar inclinado mais para um lado, o motor do lado cujo sensor está mais próximo da fonte tende a acelerar mais que o do lado oposto. O resultado é que o veículo tende a se afastar da fonte.

O funcionamento do veículo depende diretamente da diferença entre as intensidades medidas pelos dois sensores posicionados nas laterais do mesmo. Assim, é possível verificar que uma alteração da distância entre os sensores acaba produzindo uma alteração no comportamento do veículo.

Já o Veículo 2b tem as suas ligações cruzadas. O sensor A alimenta o motor B e o sensor B alimenta o motor A. O resultado é que, no caso de o sensor A estar mais próximo da fonte que o sensor B, ocorre que o motor B trabalha mais que o A, fazendo com que o veículo se dirija em direção à fonte com velocidade cada vez maior. A mudança na distância entre os sensores teve um impacto profundo no comportamento do veículo. Caso a fonte em questão emitisse uma determinada intensidade luminosa, poder-se-ia dizer que ambos os veículos não gostam de fontes de luz, sendo que o Veículo 2a é do tipo covarde que foge das fontes, enquanto que o Veículo 2b é agressivo, e tenta destruir as fontes de luz com violência.

Uma solução para o temperamento ruim dos veículos anteriores seria introduzir uma relação de inibição entre os sensores e os motores. Assim, quando os sensores fossem excitados haveria uma redução da velocidade dos motores. Neste cenário, os veículos terão a tendência de se manterem perto das fontes.

Outro fenômeno importante é que há uma inversão nos comportamentos dos dois tipos possíveis de veículos. Neste caso, o de ligações diretas se aproxima da fonte de luz e lá permanece para sempre, como se o mesmo tivesse “apaixonado” pela fonte. Enquanto o de ligações cruzadas tende a se manter um tempo perto da fonte mas, eventualmente se afasta da mesma, o que caracteriza um comportamento exploratório.

3.3.3 Algoritmo de exploração

Para realizar a exploração do labirinto em busca dos objetivos propostos foi escolhido o algoritmo *floodfill*, que a grosso modo, consiste em preencher uma matriz que representa o labirinto com valores de custo de movimentação, para este trabalho porém, foi desenvolvida uma forma simplificada do algoritmo, onde os valores das células não são atualizados e portanto não é

encontrado o menor caminho entre o robô e o objetivo, nessa abordagem um pequeno cálculo é feito para obter o custo que cada célula possui, o cálculo consiste na seguinte fórmula:

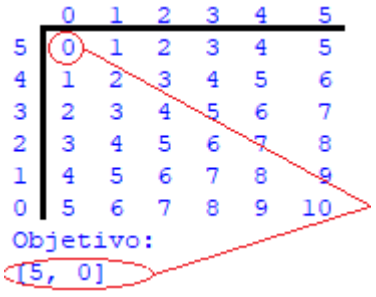
$$|x\,i\ -\ x\,obj|\ +\ |y\,i\ -\ y\,obj|\ =\ custo$$

Onde *xi* e *yi* correspondem aos valores de x e y de uma determinada célula, e os valores de *x obj* e *y obj* correspondem à célula escolhida como objetivo. Foi então criada uma função para fornecer essa matriz de custos, como é descrito a seguir:

- floodFill(goal): função que recebe como entrada uma matriz de duas posições que representa os valores de x e y do objetivo e gera uma matriz de custos de tamanho 6 por 6.

A figura a seguir ilustra no terminal a matriz gerada a partir do objetivo [5,0] (linha 5 coluna 0):

Figura 5 - Matriz de custos do primeiro objetivo



Fonte: dos autores, 2021.

Para realizar a exploração o robô avalia se existe um caminho livre à sua direita, esquerda e frente, dentre as opções disponíveis é escolhida a direção com o menor custo. Para obter os custos das células que estão dos lados e na frente do robô foi criada a seguinte função:

- getFloodFillValue(x, y, direction, orientation): função que recebe como entrada a posição do robô na matriz de movimentação, a direção que pode ser direita, esquerda ou frente, a orientação fornecida pelo V-rep e retorna o custo da célula requisitada.

Como foi mencionado, antes de avaliar o custo de uma célula nas direções citadas, o robô verifica se existe caminho livre, isto significa observar se existe uma parede naquela direção específica. para isso foram criadas as funções a seguir:

- isPathLeft();
- isPathRight();
- isPathFwd().

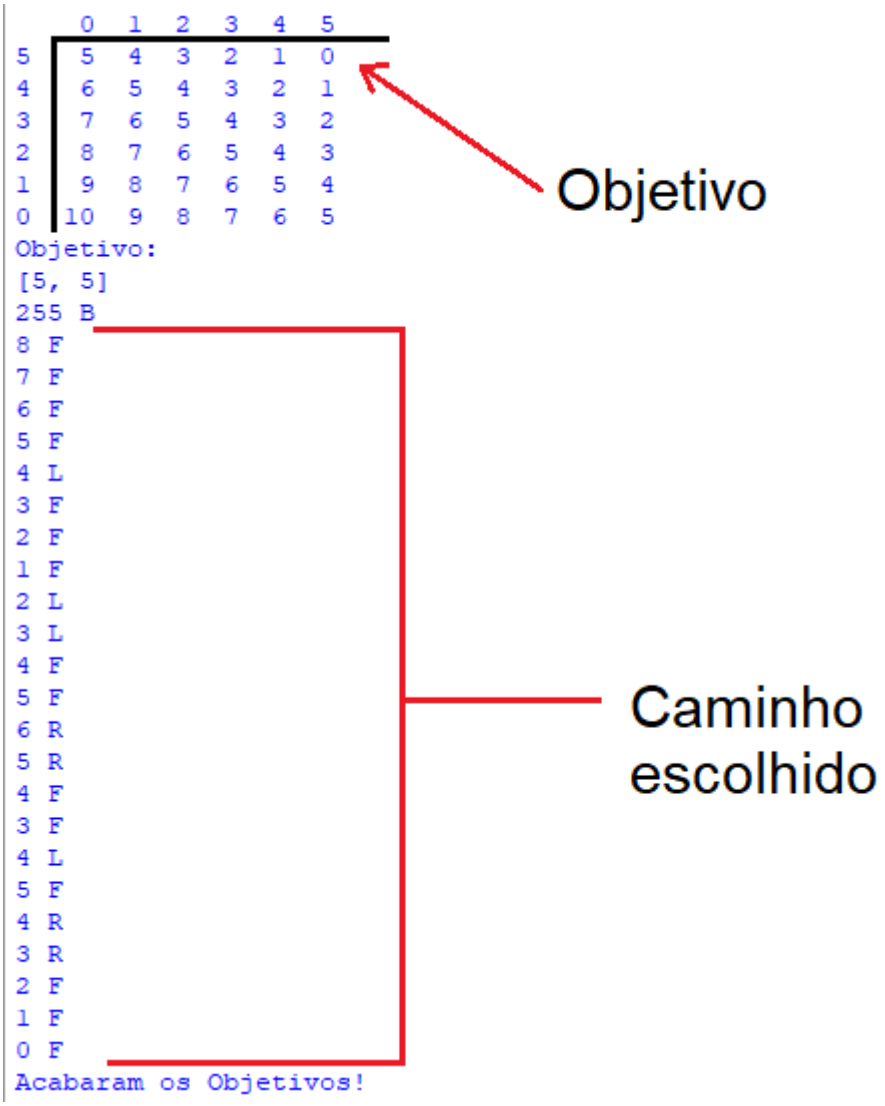
Essas funções não recebem parâmetros mas requisitam do V-rep informações de 6 sensores ultrassônicos e determina a existência de um caminho livre, são dois sensores para cada direção, os sensores 9 e 8 para a esquerda, 1 e 16 para a direita e por fim 4 e 5 para a frente.

A partir dessas funcionalidades foi criado o algoritmo de exploração que consiste em um laço de repetição que avalia as opções de movimentação do robô, escolhendo dentre elas a de menor custo, até que a posição do robô na matriz de movimentação seja igual a posição definida como objetivo.

4 RESULTADOS E DISCUSSÃO

Como resultado, o robô atingiu com êxito os objetivos definidos, para uma visualização mais detalhada do processo de tomada de decisão na exploração do labirinto foi utilizada a biblioteca Pandas para mostrar no terminal a matriz de custos e em seguida vem as informações de qual é o objetivo atual, a direção escolhida e o respectivo custo dela. A figura a seguir mostra o resultado obtido no último objetivo.

Figura 6 - Caminho escolhido pelo algoritmo no último objetivo



Fonte: dos autores, 2021.

As letras F, L, e R representam frente, esquerda e direita na devida ordem. Ao completar todos os objetivos é enviada uma mensagem ao terminal e o programa é encerrado. Os objetivos foram alcançados na seguinte ordem: [5,0] →[1,1] →[0,1] →[5,5].

5 CONCLUSÃO

Durante o desenvolvimento deste trabalho, tornou-se evidente o fato de que o robô móvel autônomo, embora pareça simples à primeira vista, trata-se de um sistema de considerável complexidade que dialoga simultaneamente com diversas áreas das ciências exatas. Por isto, se fez necessária uma extensa abordagem teórica a fim de consolidar o pleno entendimento de todo o funcionamento do robô.

O robô conseguiu realizar as trajetórias propostas. Além de o projeto ser extremamente interessante, foi uma experiência gratificante e de grande valia desenvolver este projeto.

REFERÊNCIAS

SMIDT, A. C. G. **Implementação de uma plataforma robótica controlada remotamente utilizando o Arduino**, São Carlos, Junho 2013.

VAGALLIS, Nikos. Gazebo Robotics Simulator. Artigo escrito em 2016 - Disponível em: https://www.i-programmer.info/news/169-robotics/9439-gazebo-robot-simulator.html VANÇAN, Nícolas B. Montanha. PROTOBOT - Protótipo de Braço

PAZOS, Fernando, **Automação de Sistemas & Robótica**, Rio de Janeiro, Axcel Books, 2002.

ORLANDINI, Guilherme. **Desenvolvimento de Aplicativos Baseados em Técnicas de Visão Computacional para Robô Móvel Autônomo**, Mestrado em Ciência da Computação; UMP; Piracicaba, 2012.