Universidade Federal do Rio Grande do Norte

Programa de Pós-graduação em Engenharia Elétrica e de Computação

EEC1507 - Sistemas Robóticos Autônomos

RELATÓRIO

3º PROJETO DE SISTEMAS ROBÓTICOS AUTÔNOMOS - META 2

Alunos:
Edel Mary Quinn de Oliveira Figueiredo
Luís Gabriel Pereira Condados
Samigo Ricardo de Oliveira Silva
Professor orientador: Pablo Javier Alsina

Natal-RN
2020

Universidade Federal do Rio Grande do Norte

Programa de Pós-graduação em Engenharia Elétrica e de Computação

EEC1507 - Sistemas Robóticos Autônomos

RELATÓRIO

Relatório apresentado à disciplina de EEC1507- Sistemas Robóticos Autônomos, correspondente a 3º unidade do semestre 2020.2, sob orientação do **Prof. Pablo Javier Alsina**.

Alunos:
Edel Mary Quinn de Oliveira Figueiredo
Luís Gabriel Pereira Condados
Samigo Ricardo de Oliveira Silva

Natal-RN
2020

Sumário

Lista de ilustrações

1 INTRODUÇÃO

Este relatório se refere à segunda meta do terceiro projeto avaliativo da disciplina de Sistemas Robóticos Autônomos do Programa de Pós Graduação em Engenharia Elétrica e de Computação (PPGEEC) da UFRN, cujo objetivo foi construir uma grade de ocupação a partir de sensores de alcance em um ambiente simulado.

Visando atender esse objetivo, foi criado um ambiente simulado no CoppeliaSim (1), contendo um robô equipado com sensores de alcance e obstáculos (paredes e polígonos simples) e um programa cliente em C++ para controlar a simulação remotamente, realizar a leitura dos sensores e gerar a grade de ocupação, bem como exibi-la em tempo real por meio de gráfico criado com a API para C++ do Gnuplot (2). Gráficos resultantes da simulação podem serem vistos na seção 4.

A seção 2 apresenta as tecnologias básicas que foram utilizadas nesse trabalho e A seção 4 apresenta os resultados, bem como discussões e conclusões sobre eles.

2 REFERENCIAL TEÓRICO

O modelo cinemático do robô foi utilizado para realizar o controle manual do mesmo, com o objetivo de explorar o ambiente e gerar a grade de ocupação.

2.1 MODELO CINEMÁTICO DO ROBÔ

Para a construção da modelagem do robô com rodas de dois graus de liberdade, conhecida também na literatura como roda padrão, temos que levar em consideração a restrição de rolamento puro, onde todo o movimento da roda tem que ser acompanhado pela rotação correspondente da roda, enquanto que para a restrição de derrapagem lateral todo o movimento deve ser restrito ao plano da roda, ou seja, a roda não pode ser movimentada em direção ao eixo. Com a escolha do robô móvel com acionamento diferencial, o uso da relação entre a velocidade das rodas pode ser obtido pelo giro do robô com o raio r também pode ser obtido por análise simples a partir da Figura 1:

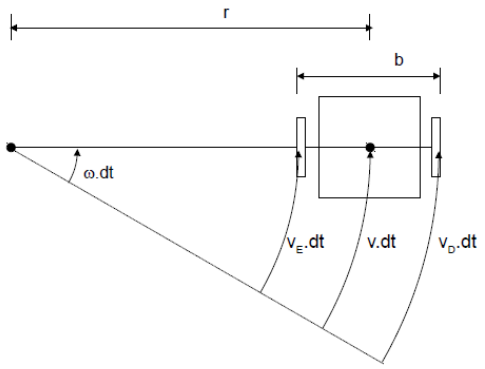


Figura 1 – Movimentos Infinitesimais para o modelo de robô utilizado.

Analisando a natureza do movimento circular na Figura 1 podemos observar as seguintes relações,

$$\omega\left(r - \frac{b}{2}\right) = v_e \tag{1}$$

$$\omega\left(r + \frac{b}{2}\right) = v_d \tag{2}$$

Somando v_e e v_d temos:

$$v = \frac{(\omega_d + \omega_e).r_w}{2} \tag{3}$$

E subtraindo v_e e v_d podemos chegar em:

$$\omega = \frac{(\omega_d - \omega_e).r_w}{b} \tag{4}$$

Podemos também relacionar as velocidades das rodas com o raio de giro instantâneo do robô

$$\frac{\omega_e}{\omega_d} = \frac{r - \frac{b}{2}}{r + \frac{b}{2}} \tag{5}$$

Por fim temos a **função de cinemática direta do robô**, considerando o espaço de configuração: $\begin{bmatrix} x & y & \theta \end{bmatrix}^T$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{(\omega_d+\omega_e).r_w}{2}.\cos\theta \\ \frac{(\omega_d+\omega_e).r_w}{2}.\sin\theta \\ \frac{(\omega_d-\omega_e).r_w}{b} \end{bmatrix} \tag{6}$$

Podemos também achar a matriz de transformação que relaciona as velocidades das rodas com a velocidade linear e angular, fazendo isso, podemos chegar ao seguinte resultado:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} (r_d/2) & (r_e/2) \\ (r_d/b) & -(r_e/2) \end{bmatrix} . \begin{bmatrix} \omega_d \\ \omega_e \end{bmatrix} \tag{7}$$

E a relação inversa:

$$\begin{bmatrix} \omega_d \\ \omega_e \end{bmatrix} = \begin{bmatrix} (1/r_d) & (b/2r_d) \\ (1/r_e) & -(b/2r_e) \end{bmatrix} . \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{8}$$

2.2 GRADE DE OCUPAÇÃO

Esta subseção contém uma breve introdução sobre a grade de ocupação. Este método consiste na representação do espaço por um campo de variáveis aleatórias binárias, dispostas em uma grade discreta regular definida sobre o espaço contínuo, o valor para cada variável aleatória é uma medida da ocupação da célula da grade correspondente. O algoritmo de grade de ocupação é capaz de implementar uma estimação posterior das variáveis aleatórias de ocupação, neste método de mapeamento é permitido a partir de dados sensoriais ruidosos, partindo da premissa que a localização do robô é perfeitamente conhecida.

Os mapas mais comuns são os mapas $2D$ para este método, o que são suficientes para aplicações de navegação de robôs sobre superfícies planas, onde os sensores iram

captar uma fatia horizontal do espaço $3D$ associadas as plantas baixas de prédios, vale salientar que representações $3D$ são possíveis, contudo, o custo computacional pode se tornar proibitivo. O mapa m é representado por um conjunto de células m_i regularmente espaçadas.

$$m = \sum_i m_i$$

A célula m_i é associada a um valor binário de ocupação, onde 1 representará a célula ocupada e 0 representará a célula livre. A cada uma célula é associada uma probabilidade de ocupação $p(m_i)$ e o princípio chave do algoritmo é o cálculo da posterior sobre o mapa $p(m|z_{1:t}, q_{1:t})$, onde:

$z_{1:t}$ é o conjunto das medições desde o instante inicial (1)até o instante t;
$q_{1:t}$ é o caminho percorrido, sendo a sequência de poses assumidos pelo robô;

Devido a dimensionalidade deste problema, o cálculo de $p(m|z_{1:t}, q_{1:t})$ se torna intratável e baseado nesta situação a solução adotada é aproximar a posterior sobre o mapa pelo produto das marginais.

$$p(m|z_{1:t}, q_{1:t}) \cong p(m_i|z_{1:t}, q_{1:t})$$

O problema é dividido em um conjunto de problemas separado mais simples, para todas as células da grade, onde cada problema de estimação individual $p(m_i|z_{1:t}, q_{1:t})$ é um problema binário, com estado estático. Com o intuito de minimizar problemas de instabilidade numérica para probabilidades perto de 0 ou 1, no lugar da representação em probabilidade de ocupação, o algoritmo usa a sua representação em *log-odds*.

$$l_{t,i} = \log \frac{p(m_i|z_{1:t}, q_{1:t})}{1 - p(m_i|z_{1:t}, q_{1:t})}$$

As probabilidades de ocupação podem ser recuperadas facilmente pela função inversa.

$$p(m_i|z_{1:t}, q_{1:t}) = 1 - \frac{1}{1 + exp(l_{t,i})}$$

Onde, l_0 é o valor de ocupação conhecido a priori, em *log-odds*.

$$l_o = \log \frac{p(m_i = 1)}{p(m_i = 0)} = \log \frac{p(m_i = 1)}{1 - p(m_i = 1)}$$

A função do modelo de medição inverso do sensor implementa o modelo de medição inverso, que tem que ser transformado para a sua representação *log-odds*.

$$\textbf{modelo\_inverso\_sensor}(m_i, z_t, q_t) = p(m_i|z_t, q_t)$$

É considerado um modelo simples de sensor de alcance, para o robô localizado em $q = [x, y, \theta]^T$, o sensor retorna uma medida de alcance z_t dentro de um cone de abertura α, com um alcance máximo z_max e com um erro de medição e. Também pode ser facilmente percebido que a região ocupada pelo robô é livre e esta informação pode ser incorporada no algoritmo.

3 METODOLOGIA

Para a simulação presente neste trabalho foi necessário, primeiramente, criar um cenário no simulador CoppeliaSim. O cenário criado contém apenas algumas paredes e alguns obstáculos poligonais simples, além de um robô (modelo de um robô Pioneer) que foi equipado com quatro sensores de proximidade/alcance do tipo disco (ver Figura 2).

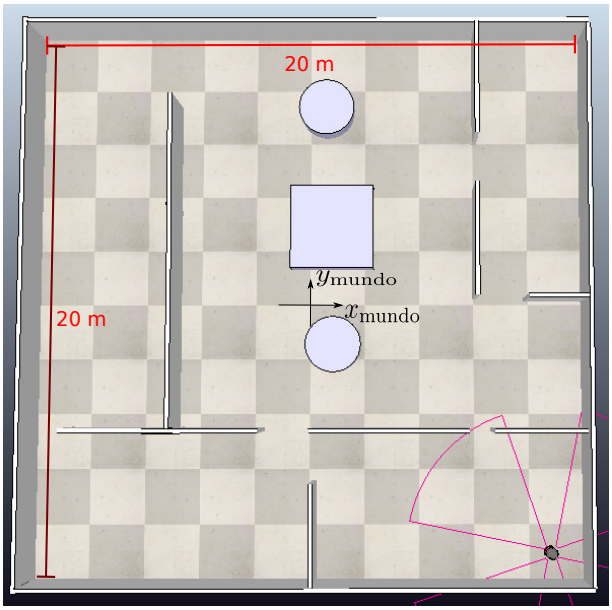
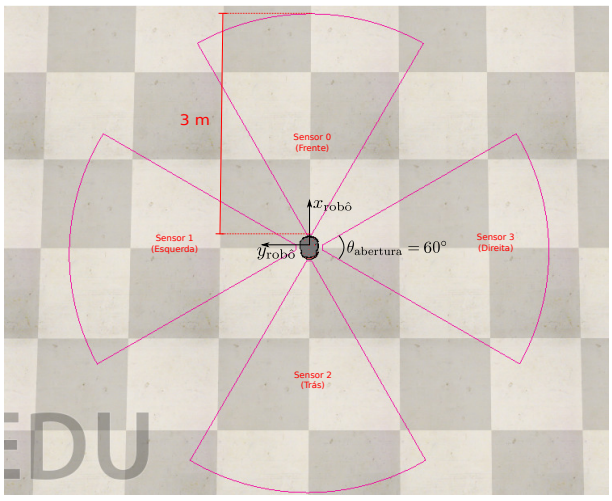
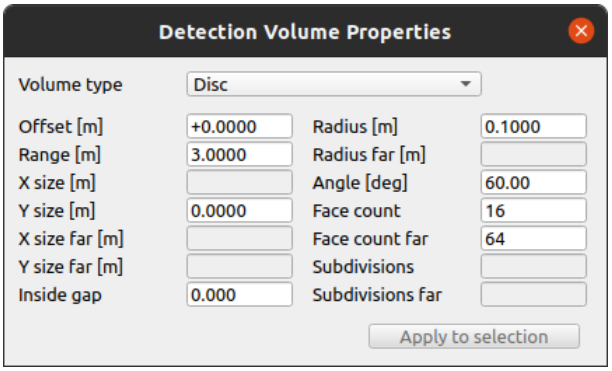


Figura 2 – Cenário utilizado na simulação.



(a) Robô equipado com quatro sensores de proximidade.



(b) Configurações dos sensores.

Figura 3 – Robô equipado com sensores de alcance.

A Figura 3 mostra com mais detalhes o Pioneer equipado com os sensores de proximidade, foram utilizados quatro sensores espaçados em 90° um do outro, cada sensor foi

configurado para possuir um alcance de 3 m e um ângulo de abertura de 60°, a Figura 3b mostra a janela de configurações de um desses sensores (todos possuem essa mesma configuração).

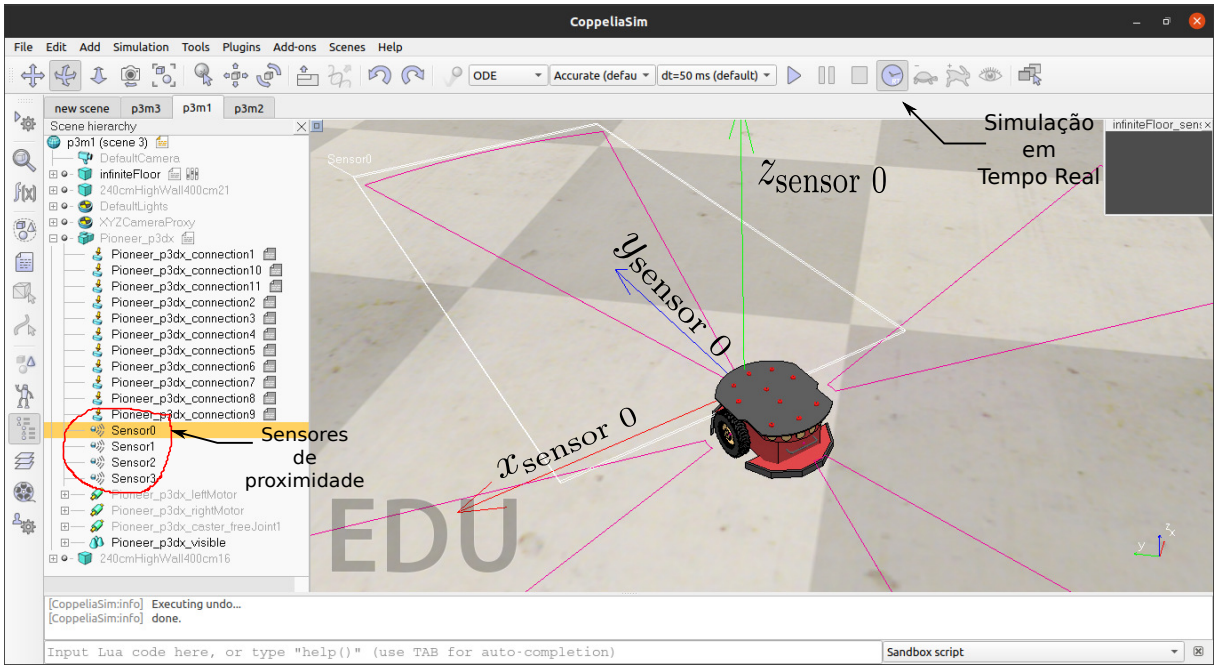


Figura 4 – Detalhes da simulação com Pioneer equipado com sensores de proximidade.

A Figura 4 apresenta uma imagem do ambiente simulado com algumas informações importantes sendo destacadas:

- A simulação foi configurada para rodar em tempo real, ou seja, sem ser acelerada.
- A nomenclatura dos sensores. Essa informação é importante para poder realizar a leitura desses sensores por meio da API de comunicação.
- A referência dos sensores. É importante saber como são referenciados, pois suas leituras são de acordo com esse referencial.

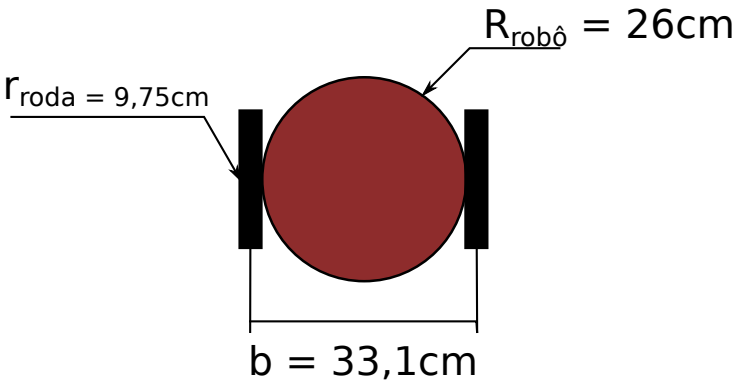


Figura 5 – Ilustração do modelo aproximação usada sobre o *Pioneer* com as dimensões do modelo real.

Especificações do Pioneer (parâmetros utilizados para a função de cinemática inversa):

- Raio de cada roda: $0,09751$ m (r_w)
- Distância entre os centros das rodas $0,331$ m(b)

Com o objetivo de explorar o cenário com o robô, foi utilizado um programa/*script*, na linguagem de programação Lua (3), esse programa foi colocado no modelo do robô diretamente na simulação, uma descrição do funcionamento desse *script* é apresentado a seguir:

O programa monitora o pressionar das teclas do teclado e executa as seguintes operações/modos de acordo:

- **Automático:** Ao se pressionar a tecla **A** o robô passa a utilizar um controle reativo, com uma velocidade linear fixa e a velocidade angular sendo ajustada de acordo com as leituras dos 16 sensores de proximidades presentes ao redor do modelo do Pioneer, o objetivo disso é de evitar colisões. Esse controle reativo vem por padrão no modelo do Pioneer do CoppeliaSim.
- **Manual:** Pressionando a tecla **A** do teclado, caso o modo automático esteja ligado ele será desligado e passa para o controle manual, onde as setas do teclado passam a incrementar ou decrementar as velocidades linear e angular do robô. As setas para cima e para baixo são responsáveis pela velocidade linear, servindo para aumentar ou diminuir, respectivamente. As setas para a esquerda e para a direita controlam a velocidade angular, sendo a primeira para aumentar a velocidade angular (favorecendo o movimento no sentido anti-horário) e a segunda para diminuir (sentido horário).
- **Parar:** Ao pressionar a tecla **S** as rodas do robô param.
- **Rotacionar:** A tecla **R** faz com que o robô rotaciona em torno do seu próprio eixo com uma velocidade constante no sentido anti-horário.

Para realizar a leitura dos sensores e controlar a simulação (*play*/*pause*) foi feito um programa em C++, o programa possui a seguinte estrutura:

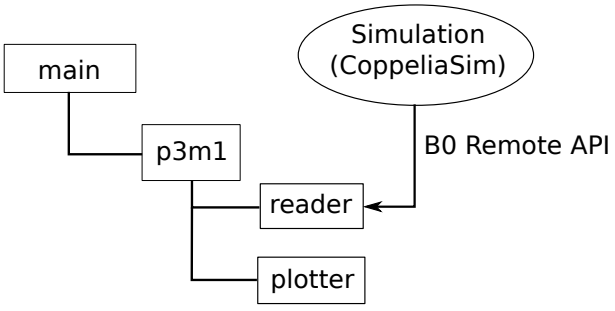


Figura 6 – Ilustração da arquitetura do programa em C++.

A Figura 6 ilustra de forma simplificada como foi organizado o cliente em C++, sendo:

- **main** o programa/*thread* principal que é responsável por iniciar o *p3m1*, após lançar a execução da *p3m1* o *main* fica aguardando uma entrada de teclado para encerrar a simulação.

- **p3m1** encapsula as rotinas que são executadas em paralelo: **reader** e **plotter**, além de iniciar (*play*) o ambiente de simulação (CoppeliaSim deve está com o cenário aberto e com o **Add-on B0RemoteApiServer** ligado).

- **reader** é a rotina responsável por realizar a leitura periódica dos sensores de proximidade por meio da *B0 Remote API* (4) para C++, bem como a configuração atual do robô (posição e orientação nas coordenadas de mundo, esse referencial é ilustrado na Figura 2).

- **plotter** cria o gráfico (por meio da API Gnuplot (2) para C++) que representa o espaço de trabalho, o robô (que é representado por um retângulo com as dimensões correspondentes às do Pioneer), sua trajetória (linha em vermelho) e cada leitura dos sensores é marcada no gráfico com uma marcação em azul. Esses gráficos podem ser visto na seção 4

A grade de ocupação é atualizada a medida que as leituras dos sensores são feitas, sua atualização ocorre de acordo com o apresentado na subseção 2.2. Os parâmetros utilizados para gerar a grade de ocupação nesse trabalho foram:

- Erro dos sensores:0,1 m;
- l_0:0;
- l_L:-0,5;
- l_{oc}:0,8.

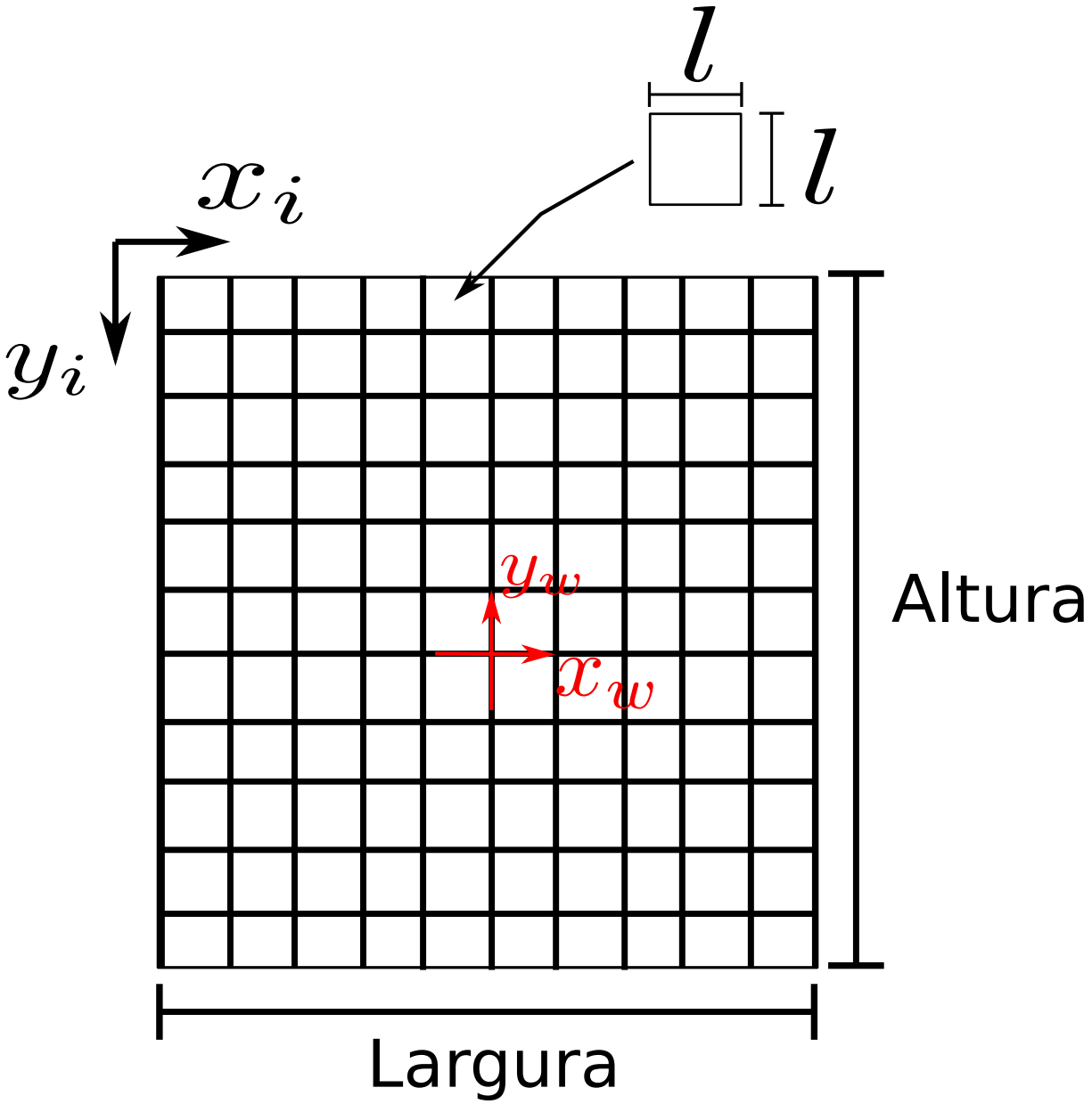


Figura 7 – Ilustração do modelo de uma grade regular conforme utilizado nesse trabalho.

Uma ilustração do modelo da grade regular utilizada é apresentada na Figura 7, sendo x_i e y_i os eixos coordenados discretos que foram considerados para tratar a grade como é feito em processamento digital de imagens. A grade é composta por quadrados com lado l e seu centro corresponde ao centro do cenário (que também é o centro do referencial de "mundo"/*World*, representados pelos eixos coordenados x_w e y_w). A **Largura** e a **Altura** equivalem as dimensões do cenário na simulação, ou seja, ambas correspondem à 20 m ou 200 células, pois no trabalho foi utilizado l igual a $0,1$ m (por isso 20 m , já que números de células $= 20/0,1 = 200$ células). Outro ponto que é importante destacar é que a grade é armazenada em um vetor unidimensional, por isso ela é percorrida com apenas um índice de posição (i), porém para facilitar a compreensão é convertida para x_i e y_i em alguns momentos no programa.

Algumas relações que foram importantes para este trabalho são apresentadas a seguir:

$$x_w = x_i * l + \text{Largura}/2$$
$$y_w = \text{Altura}/2 - y_i * l$$

onde Largura e Altura estão em **metros**.

$$x_i = i \mod Largura$$
$$y_i = \lfloor i/Largura \rfloor$$

onde Largura e Altura representam quantidade de células.

Para validar o trabalho, foi salvo o gráfico contendo a grade de ocupação em diferentes instantes de uma simulação, os resultados são apresentados na seção seguinte (seção 4). Um link para o vídeo que mostra a simulação sendo executada pode ser encontrado na seção 5.

4 RESULTADOS E CONCLUSÕES

Nesta seção serão apresentados os resultados de algumas simulações que foram feitas utilizando a metodologia descrita na 3.

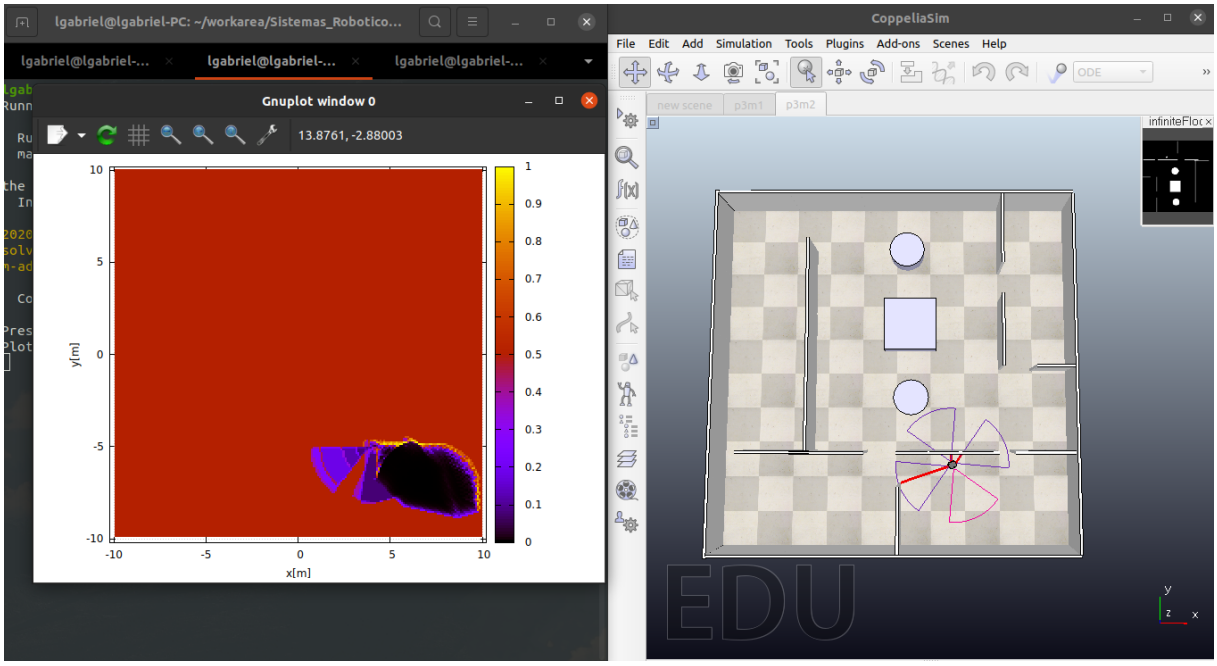
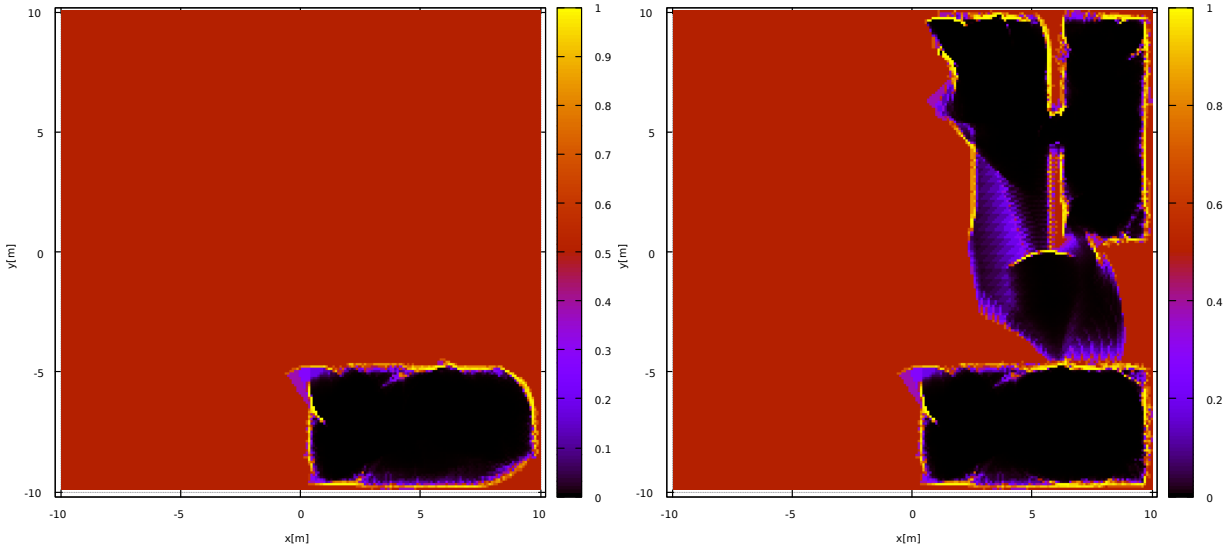


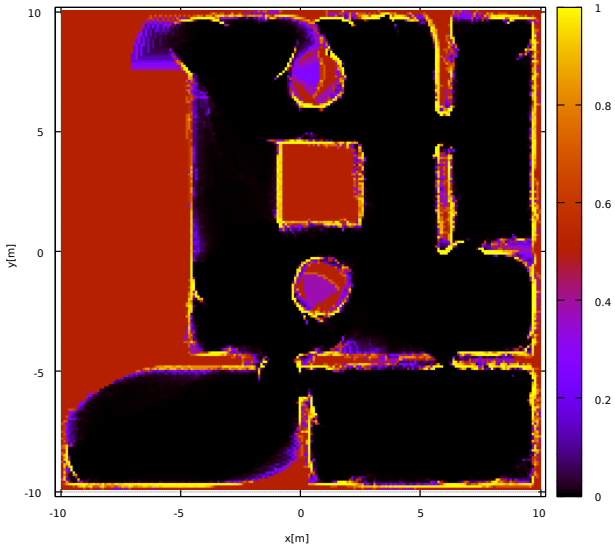
Figura 8 – Gráfico e simulação em execução.

A Figura 8 é uma captura de tela que contem o gráfico/grade de ocupação gerada e o ambiente do CoppeliaSim em execução. A seguir é apresentado uma sequencia de imagens que mostram a evolução na construção da grade de ocupação (Figura 9).

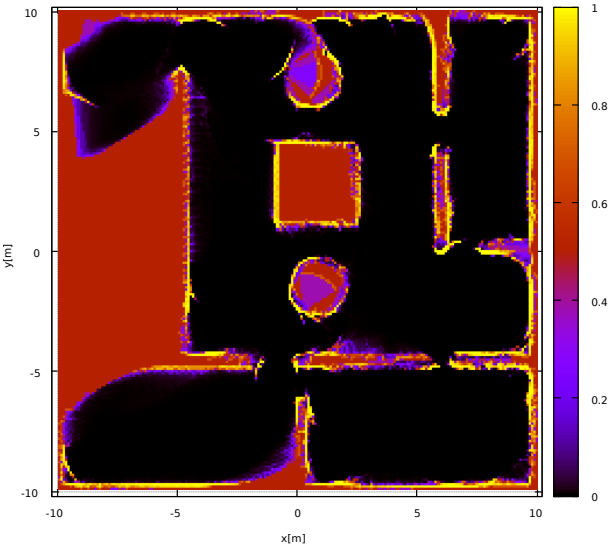


(a) Grade de ocupação. Instantâneo 1.

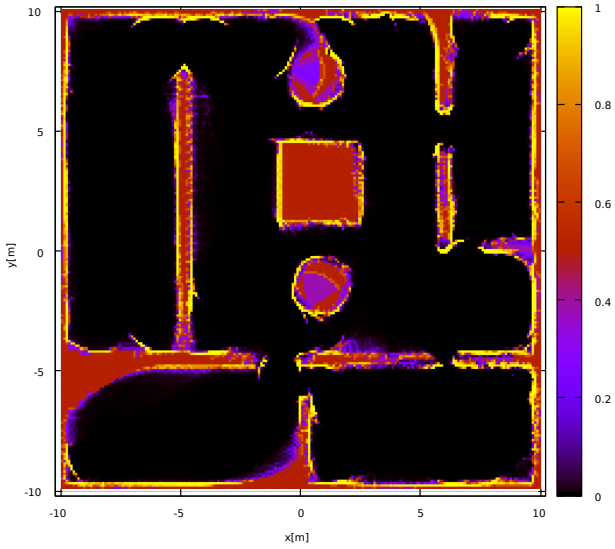
(b) Grade de ocupação. Instantâneo 2.



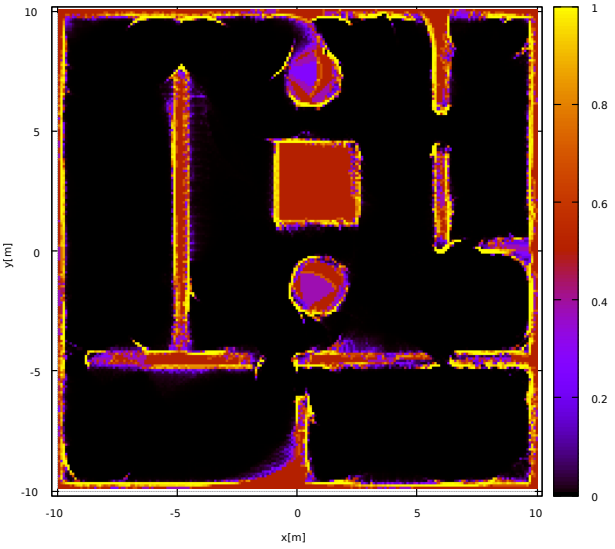
(c) Grade de ocupação. Instantâneo 3.



(d) Grade de ocupação. Instantâneo 4.



(e) Grade de ocupação. Instantâneo 5.



(f) Grade de ocupação. Instantâneo 6.

Figura 9 – Evolução da construção da grade de ocupação.

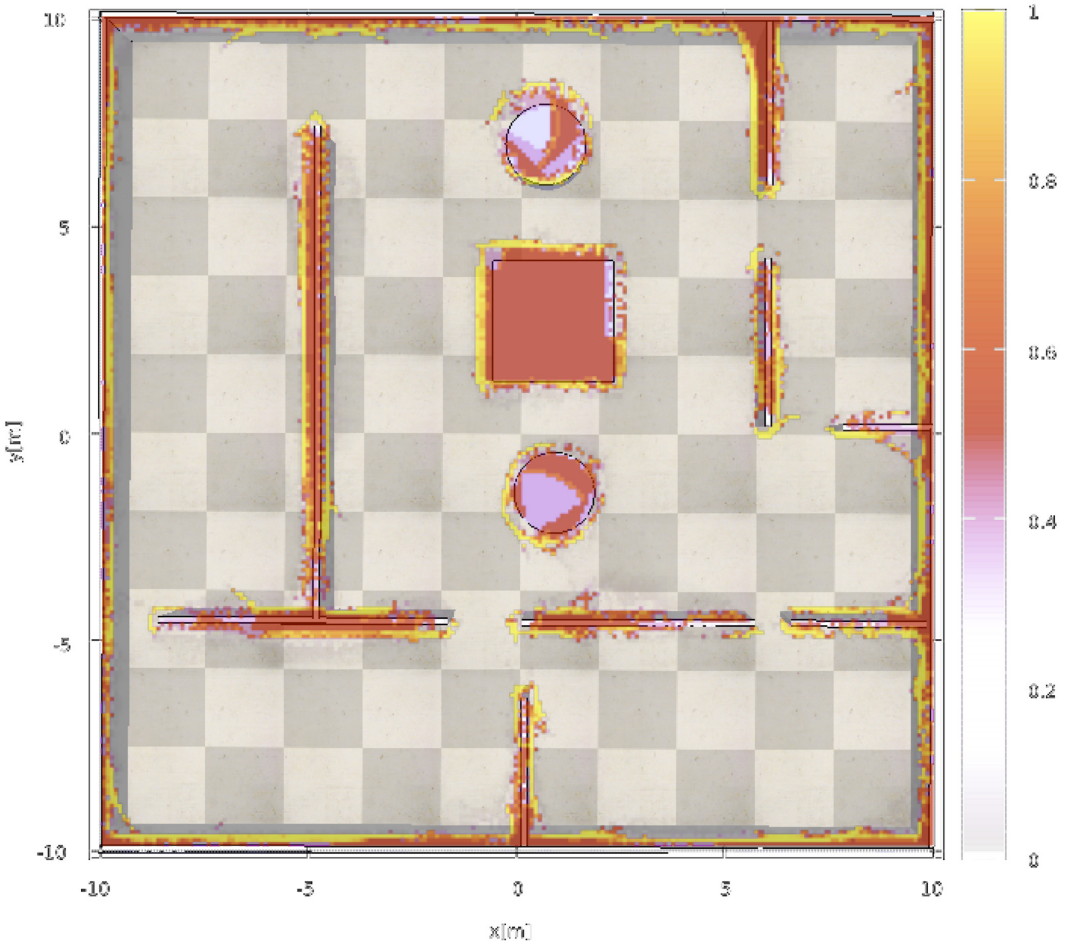


Figura 10 – Imagem da grade de ocupação superposta ao cenário.

A Figura 10 é uma imagem que contém o superposição entre a grade de ocupação gerada na simulação e o cenário. Por meio dessa imagem e da sequencia apresentada na Figura 9, podemos observar que a praticamente toda a região livre foi reconhecida, porém houveram alguns ruídos em algumas regiões, ou seja, nem todas regiões ocupadas estão com 100% de certeza (amarelo) e a região interna de alguns obstáculos foram considerados como região livre, sendo que deveriam continuar sendo regiões incertezas (vermelha/50% de estarem ocupadas).

Apesar da qualidade de alguns resultados apresentados, podemos concluímos, pelos resultados a cima, que o objetivo de gerar uma grade de ocupação no ambiente simulado fazendo uso de sensores de alcance foi atingido.

Referências

1 ROHMER, E.; SINGH, S. P. N.; FREESE, M. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*. [S.l.: s.n.], 2013. Www.coppeliarobotics.com. 1

2 WILLIAMS, T.; KELLEY, C.; many others. *Gnuplot 4.6: an interactive plotting program*. 2013. <http://gnuplot.sourceforge.net/>. 1, 9

3 ORG, L. *Lua the programming language*. <https://www.lua.org/>. Acessado em: 21 Dez. 2020. 8

4 ROBOTICS, C. *B0 Remote API*. <https://www.coppeliarobotics.com/helpFiles/en/b0RemoteApiOverview.htm>. Acesso em: 20 Dez. 2020. 9

5 ANEXO 1

Código fonte: <https://github.com/Gabriellgpc/Sistemas_Roboticos/tree/master/program/p3m2>

Vídeo Demostrativo: <https://youtu.be/5lMkQWt_dUw>