UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

3º RELATÓRIO: Planejador de caminhos baseado na grade de ocupação construída com navegação utilizando mapa gerado com o LIDAR.

CoppeliaSim 4.1.0 (V-REP)

Natal - RN
2021

Iago Lucas Batista Galvão
Jhonat Heberson Avelino de Souza
Thiago de Araújo Brito

3º Relatório: Planejador de caminhos baseado na grade de ocupação construída com navegação utilizando mapa gerado com o LIDAR.

CoppeliaSim 4.1.0 (V-REP)

Última entrega de três do 3º Relatório da disciplina EGM0007 - Sistemas Robóticos Autônomos do Mestrado em Engenharia Mecatrônica da Universidade Federal do Rio Grande do Norte (UFRN), referente a nota parcial da terceira unidade.

Professor(a): Pablo Javier Alsina.

Natal - RN
2021

Sumário

1 INTRODUÇÃO

Com o aumento do poder computacional, houve um crescimento no número de pesquisas na área de robótica e, atualmente, o pesquisador é capaz de realizar simulações com qualidade em cenários 2D/3D com várias ferramentas revolucionárias. Os simuladores possibilitam a realização de experimentos, sem a necessidade de construir o *hardware* do robô para desempenhar vários tipos de testes. Como tal desenvolvimento custa caro, esta alternativa possibilita a pesquisa em robôs mais viável e difundida.

Existem muitos *softwares* de simulação robótica disponíveis como, por exemplo, Open HPR, Gazebo, Webots, V-REP etc (1). A plataforma utilizada nesse estudo foi a *Virtual Robot Experimentation Platform* (V-REP), pois atende muitos requisitos, como estrutura de simulação versátil e escalável, arquitetura de controle distribuída, controle por *Script*, *Plugins*, ou API cliente remota entre outras funcionalidades (1), ademais, esta é uma das mais empregadas nos estudos robóticos de todo o mundo.

A versatilidade do simulador contribui para diferentes plataformas robóticas e várias aplicações, não atentando-se somente ao fato de simular robôs com resultados iniciais. Outrossim, isenta o gasto e a necessidade da aquisição do *hardware* para resultados mais realísticos e adaptáveis.

Na Primeira etapa foi criado um modelo simples de robô móvel com a modelagem cinemática, como discutida antes, também criado controladores de posição, seguidor de caminho e trajetória.

Na Segunda etapa, de planejamento de mapa, aonde criamos nosso espaço de configuração (robô como um ponto), incluindo os obstáculos poligonais na planta baixa e detecção de colisões, e projetar caminho a ser percorrido com o robô.

Nesta etapa, foi colocado um sensor que consiga criar uma nuvens de pontos seja ele uma radar ou LIDAR, a partir dos dados consiga gerar um mapa, com técnica de mapa de grade ocupação, para o robô em tempo real consiga movimentar, e detectar os obstáculos, e assim a partir do método *Probabilistic road Map* (PRM) para seguir o caminho desejado.

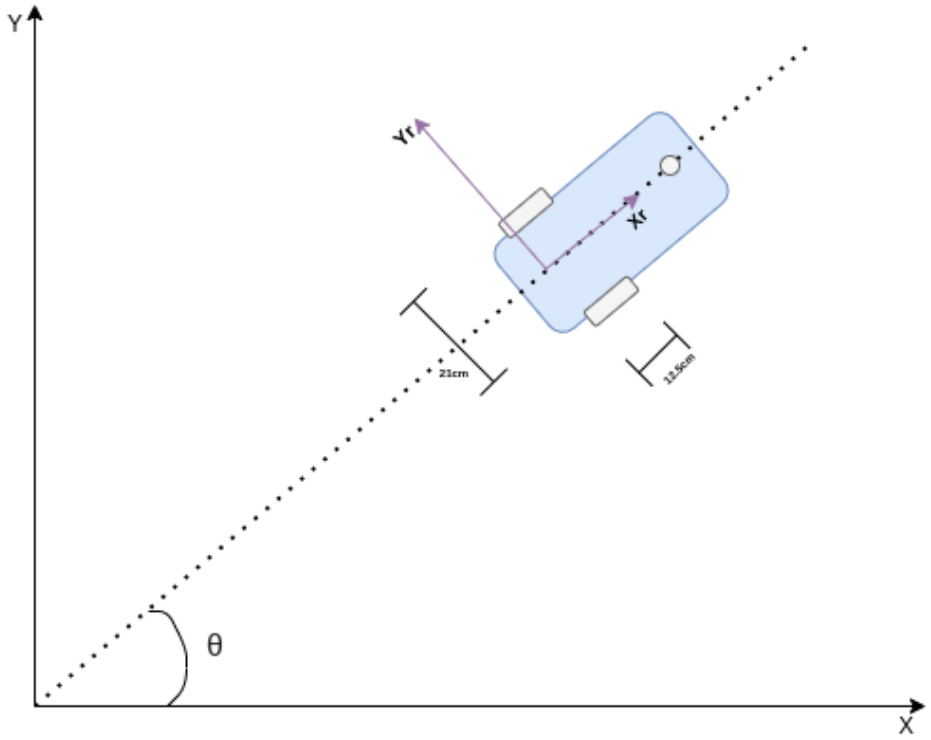
2 OBJETIVO

O propósito desta simulação é aplicar os conhecimentos teóricos aprendidos, desenvolvendo Planejadores de Caminhos para Robô Móvel que permitam ao mesmo executar movimentos especificados em espaço povoado de obstáculos, sem colidir com os mesmos. Nesta primeira entrega é desenvolvido um sistema navegação, baseado no mapeamento gerado via um sensor LIDAR.

3 METODOLOGIA

Primeiramente a uma modelagem não holonômica do robô para projetar o controlador e trajeto foi calculado com os parâmetros do robô. Dessa forma, empregou-se os conceitos aprendidos de modelagem cinemática e não holonômica de robôs, como demonstrada na Figura 1, logo abaixo.

Figura 1. *Modelagem do robô*



Fonte: Autores.

Roda Direita:

- θ_d = -90°
- $\alpha_d = 180°$
- $l_d = \frac{b}{2}$

Roda Esquerda:

- $\theta_e = 90°$
- $\alpha_e = 0°$
- $l_e = \frac{b}{2}$

Com base nessas informações podemos obter as equações derrapagem lateral e rolamento:

$$\left[\begin{array}{ccc} 1 & 0 & \frac{b}{2} \end{array}\right] * ({}^{i}R_{R}(\theta))^{T} * q_{\prime} = w_{d} * r_{d} \tag{1}$$

$$\left[\begin{array}{ccc} 1 & 0 & \frac{-b}{2} \end{array}\right] * ({}^{i}R_{R}(\theta))^{T} * q_{\prime} = w_{e} * r_{e} \tag{2}$$

Restrição de derrapagem lateral:

$$\left[\begin{array}{ccc} 0 & 1 & 0 \end{array}\right] * ({}^{i}R_{R}(\theta))^{T} * q_{\prime} = 0 \tag{3}$$

Ao agruparmos as Equações 1, 2 e 5 teremos nossa modelagem cinemático do robô:

$$\left[\begin{array}{ccc} 1 & 0 & \frac{b}{2} \\ 1 & 0 & \frac{-b}{2} \\ 0 & 1 & 0 \end{array}\right] * ({}^{i}R_{R}(\theta))^{T} * q_{\prime} = \left[\begin{array}{cc} r_{d} & 0 \\ 0 & r_{e} \end{array}\right] * \left[\begin{array}{c} w_{d} \\ w_{e} \end{array}\right] \tag{4}$$

para encontrar $q^{'}$:

$$\left[\begin{array}{c} X \\ Y \\ \theta \end{array}\right] = \left[\begin{array}{c} \frac{\cos(\theta) * (r_{d} * w_{d} + r_{e} * w_{e})}{2} \\ \frac{\sin(\theta) * (r_{d} * w_{d} + r_{e} * w_{e})}{2} \\ \frac{(r_{d} * w_{d} - r_{e} * w_{e})}{b} \end{array}\right] \tag{5}$$

onde $r_{d} = r_{e} = 6,25cm$ e $b = 21cm$.

Além da posição e orientação, necessita-se do controle das velocidades linear e angular para concretizar o controlador da trajetória a ser seguida. Dessa forma podemos iniciar a modelagem dos controladores, os quais serão discutidos nos próximos tópicos.

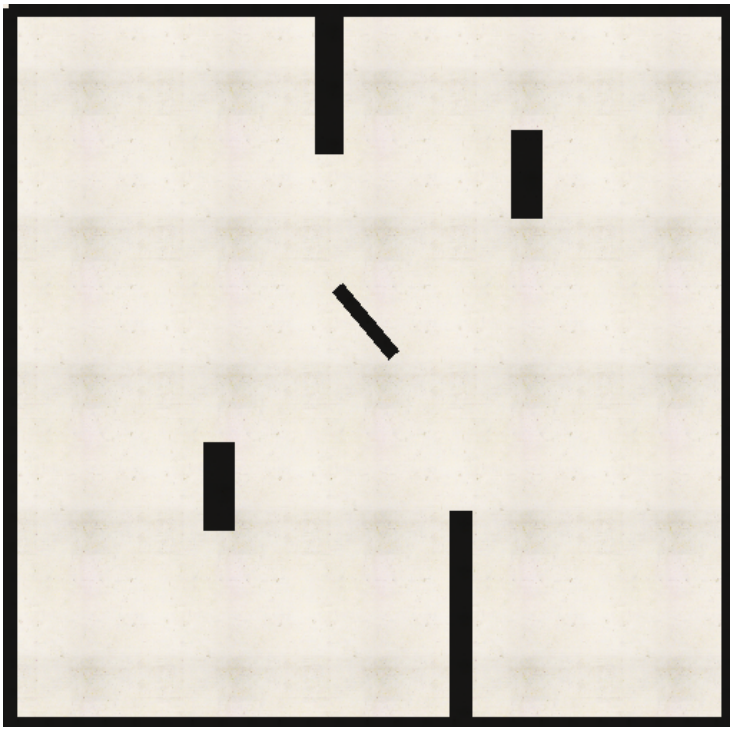
Inicialmente utilizamos a API do V-REp com Python para projetar os controladores, porém para essa segunda etapa que é para planejamento do mapa, teríamos que implementar funções do zero, a qual não é o principal intuito dessa disciplina, por isso optamos com mudar nossa principal ferramenta de implementação para MATLAB (MATrix LABoratory) trata-se de um software interativo de alta performance voltado para o cálculo numérico. O MATLAB integra análise numérica, cálculo com matrizes, processamento de sinais e construção de gráficos em ambiente fácil de usar onde problemas e soluções são expressos somente como eles são escritos matematicamente, ao contrário da programação tradicional, onde tem funções já implementadas a partir de tools box como *Robotics System Toolbox*.

Na primeira etapa do projeto foi dividido em três subetapas: a primeira foi a simulação do modelo cinemático, a segunda etapa foi gerado o caminho baseado em um polinômio de 3° aplicado pelo usuário e, a terceira, implementa os controladores cinemáticos do robô móvel.

Na segunda etapa do projeto, objetivou-se o planejamento do mapa. Para isso, foram inseridos obstáculos no espaço de trabalho, os quais devem ser reconhecidos pela câmera e considerados pelo robô móvel ao planejar o caminho. A planta baixa do nosso espaço junto com os obstáculos

é ilustrada na Figura 2, onde em preto são os objetos a serem evitados.

Figura 2. *Planta baixa.*



Fonte: Autores.

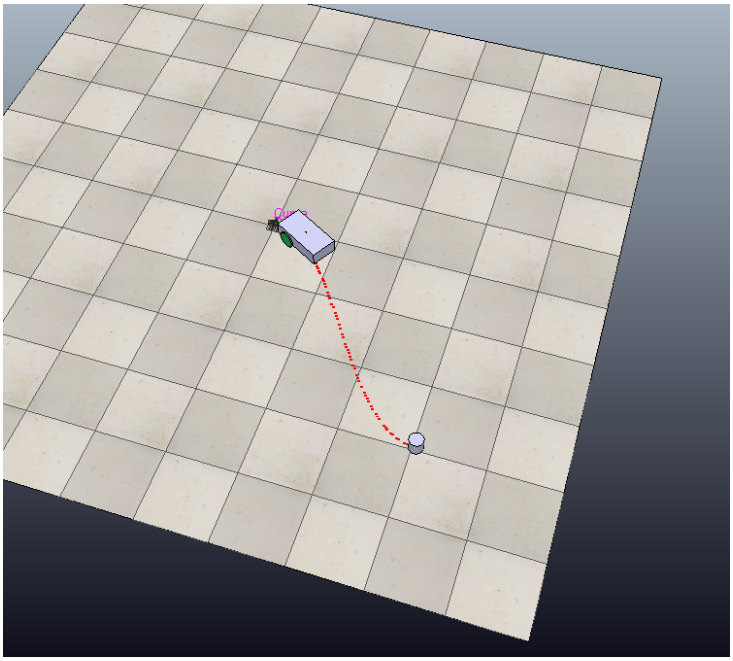
Nesta terceira etapa, vamos supor que não temos um mapa, dessa forma vamos utilizar de um sensor gerador de nuvens de pontos para construir nosso mapa e identificar obstáculos, para isso utilizamos o sensor LIDAR (*Light Detection And Ranging*, em português, Detecção e Alcance da Luz), ondes os dados são processados para gerar a nuvens de pontos e utilizamos o algoritmo de mapa grade de ocupação para tratar estas nuvens e identificar os objetos.

4 GERADOR DE CAMINHO E CONTROLADORES

No primeiro experimento foi aplicado a função seguidor de caminho com a curva interpolada. Para tal, o robô recebe os pontos gerados pela interpolação do polinômio de 3º grau fornecido e com a função *"Path planning"*, este segue ponto-a-ponto o caminho traçado com cada posição e orientação predefinidas (x, y e θ).

A interpolação do polinômio fornecido foi plotada ponto-a-ponto da origem até o destino. Na Figura 3 ilustra em vermelho o caminho interpolado a ser seguido.

Figura 3. *Interpolação do polinômio*



Fonte: Autores.

4.1 SEGUIDOR DE CAMINHO

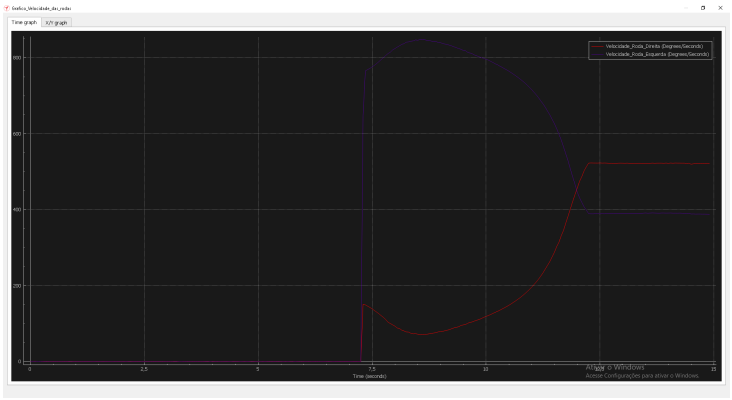
Como mencionado anteriormente de forma indireta, para o carro seguir a trajetória, necessita-se do controle de posição e orientação. Este controle é ajustado pela variação do tamanho da reta e do raio de curvatura do ângulo, ambos em relação ao ponto/destino a ser alcançado.

Dessa maneira, os erros de posição e orientação do robô são minimizados com o aumento ou redução da velocidade na roda. O direcionamento deste é realizado pelas aplicações de velocidades distintas em cada motor, onde a parte frontal é inclinada de forma a reduzir o ângulo do eixo central do robô ao ponto a ser seguido.

A Figura 4 representa o gráfico da velocidade de cada do robô por toda trajetória percorrida.

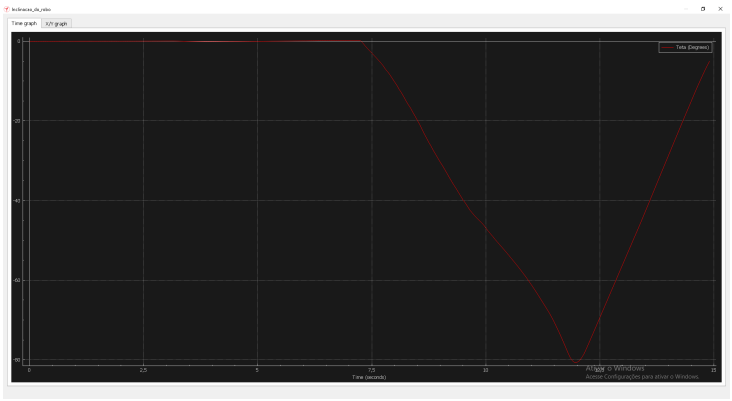
Por fim, as Figuras 5 e 6 representam os gráficos de inclinação do carro em relação ao ponto de destino e a configuração do robô para o volume *XYZ*, respectivamente. Onde o controlador deve ajustar cada velocidade com a posição e orientação atual do carro até o destino final.

Figura 4. *Velocidades das rodas*



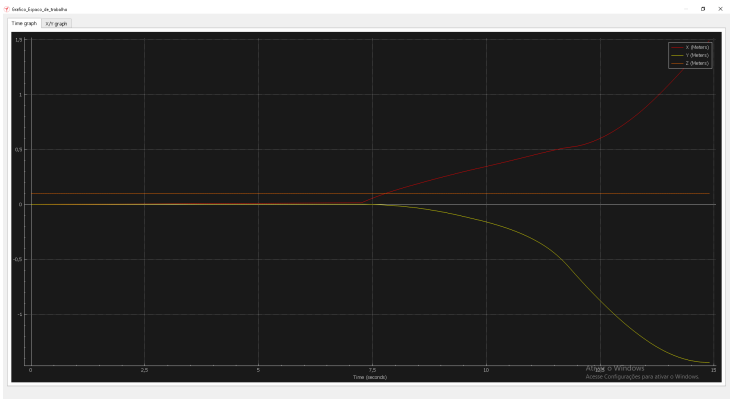
Fonte: Autores.

Figura 5. *Inclinação θ do robô móvel*



Fonte: Autores.

Figura 6. *Configuração XYZ do robô móvel*



Fonte: Autores.

A curva (em rosa) gerada pelo caminho seguido, pode ser ajustada pelos ganhos linear e angular. Onde tal ajuste influência na precisão da trajetória seguida, a qual deve ser a mais próxima e suave da curva interpolada possível.

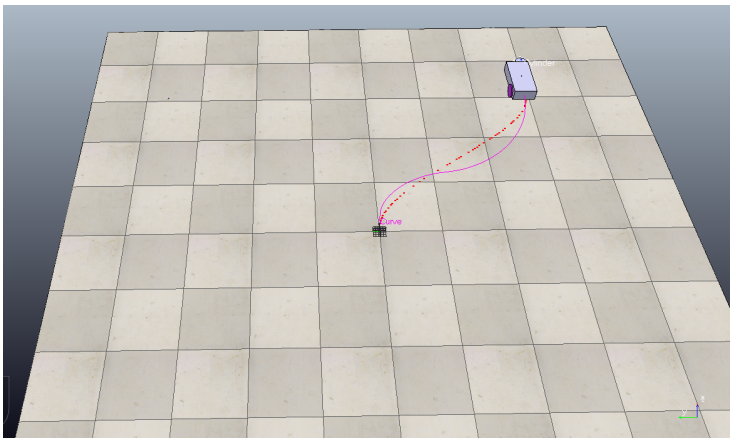
4.2 CONTROLADOR DE POSIÇÃO

Nesta seção, são discutidos a implementação de cada controlador do robô móvel. Após gerar o caminho, temos a curva interpolada que é passada para a função seguidor de caminho. Para definir a trajetória a ser seguida, o controle do caminho é limitado no tempo e o erro da posição e orientação deste é gerado ponto-a-ponto, onde o controle de velocidade em cada roda é ajustado para correção.

A partir dos ganhos proporcionais em cada motor, as velocidades linear e angular são ajustadas de acordo com o tamanho da reta ou raio da curva, respectivamente. No tópico anterior foi gerado o caminho e traçado os pontos a serem seguidos. Para seguir o caminho, o robô deve retornar sua posição atual, para ajustar os erros de posição e orientação em relação ao próximo ponto até que atinja uma distância do destino menor que *0.1cm*.

A trajetória é baseada no tempo, assim, as velocidades das rodas podem ser definidas baseadas nos erros. Em outras palavras, a aceleração é definida da relação do ponto atual ao ponto a ser seguido. A Figura 7 ilustra o *frame* da simulação ao atingir o destino desejado.

Figura 7. *Trajetória percorrida pelo robô móvel*

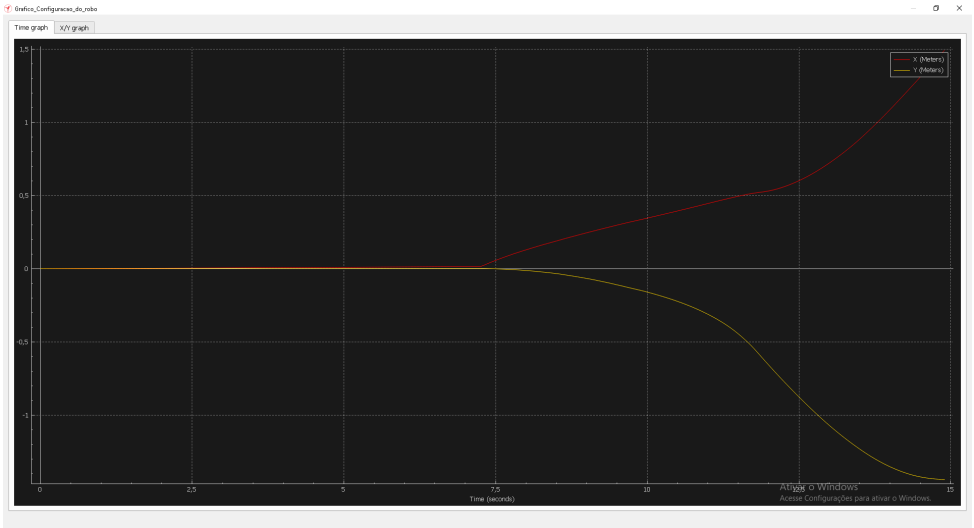


Fonte: Autores.

Os gráficos abaixo foram gerados para analisar o comportamento da função e uma melhor compreensão da rotina de simulação e ajustes dos ganhos. A configuração do robô para X e Y é demonstrada na Figura 8.

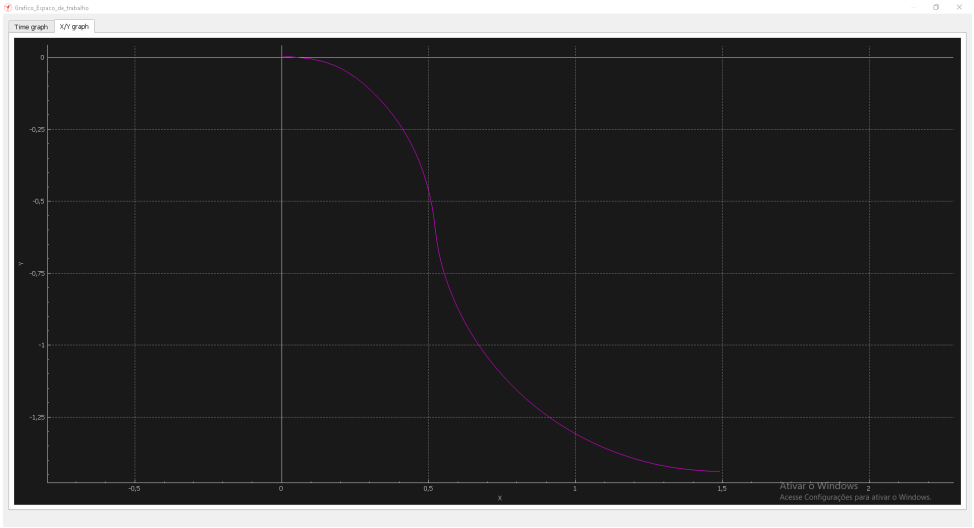
O caminho seguido, aproxima-se da curva interpolada, onde este está demonstrado na Figura 9 à seguir.

Figura 8. *Configuração do robô móvel para o plano XY*



Fonte: Autores.

Figura 9. *Trajetória percorrida no plano XY*



Fonte: Autores.

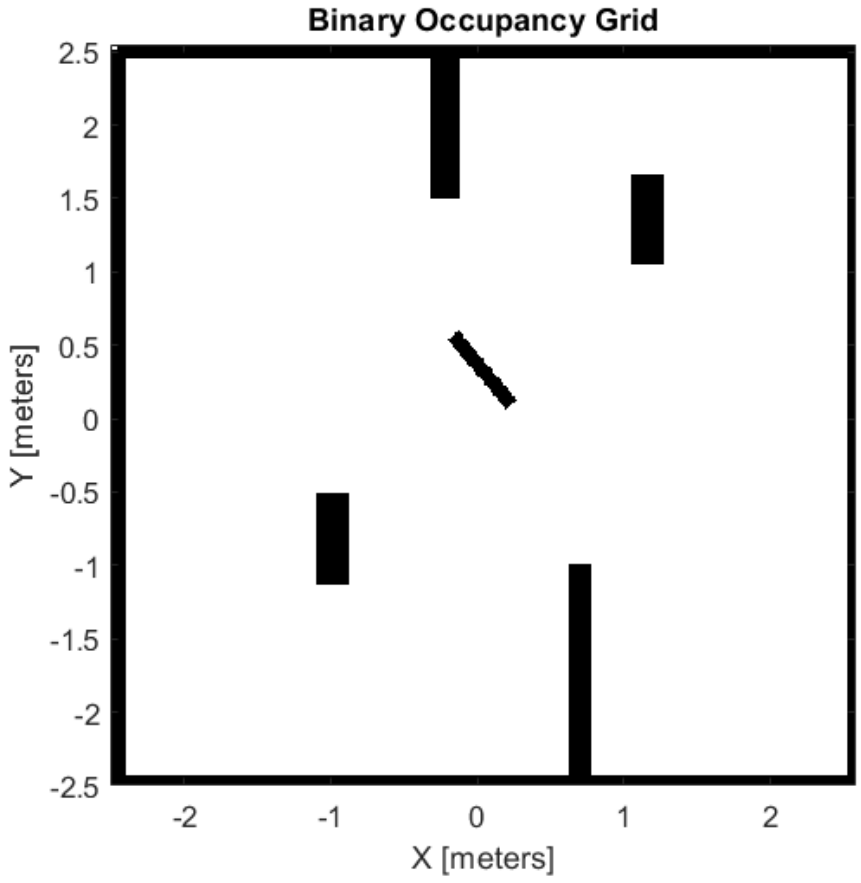
5 PLANEJADOR DE CAMINHO

Com as informações do modelo cinemático, podemos implementar esse robô e modelar no **V-REP**, aplicando as dimensões da modelagem para testar a simulação e gerar os gráficos necessários, antes de implementar o controle das velocidades via *script*. Utilizando uma câmera no simulador é possível gerar uma imagem da cena com os obstáculos predefinidos, possibilitando a criação do mapa para planejamento do melhor caminho, o qual considera a menor distância do ponto A ao B sem colidir em nenhum objeto.

5.1 MAPA E ESPAÇO DE CONFIGURAÇÃO

Neste experimento, após a captura da imagem do espaço de trabalho, esta foi convertida em outra imagem de mesmas dimensões. A nova imagem gerada, contém apenas *pixels* brancos e pretos, os quais representam o espaço de trabalho e os obstáculos, respectivamente. Esta é representada na Figura 10 binária de 5x5 metros, onde foi considerada a origem (0,0) no centro da imagem.

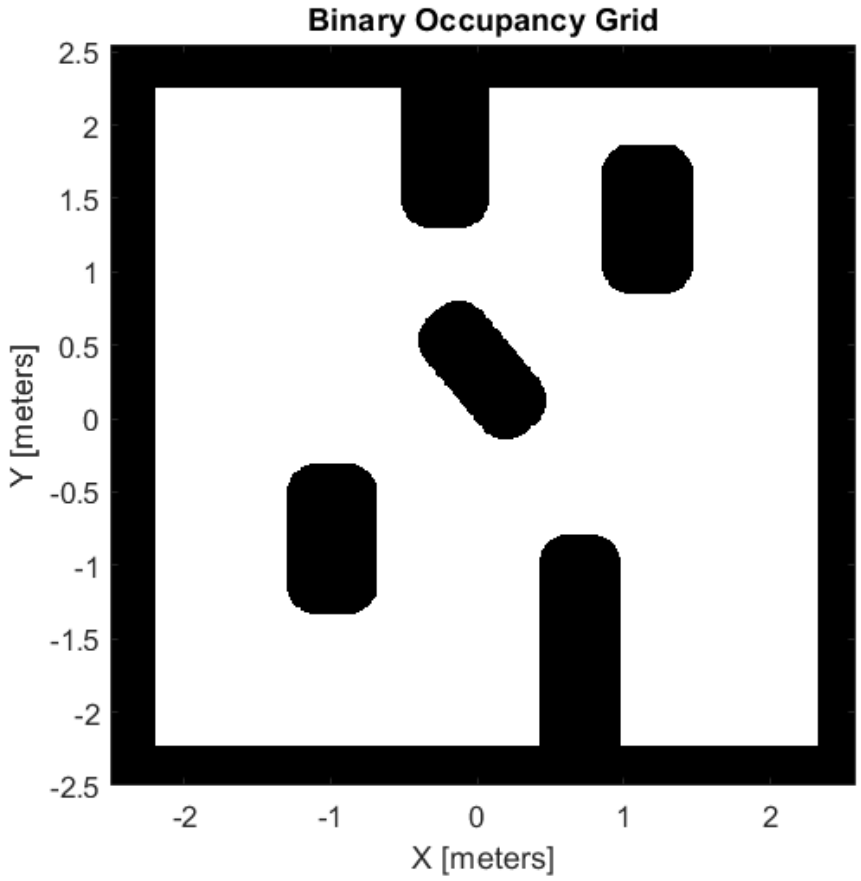
Figura 10. *Mapa Binário*



Fonte: Autores.

O robô móvel possui um raio de giro de *20cm*; desse modo, como este é representado como um ponto, aumentou-se *0.2m* dos *pixels* pretos na imagem binária para compensar a distância real do robô no planejamento. Dessa maneira, o mapeamento considera apenas a parte branca restante, onde os cálculos dos pontos podem ser realizados com a margem real do robô móvel para evitar colisões. O novo espaço de configuração inflado é ilustrado na Figura 11.

Figura 11. *Mapa Binário Aumentado com Raio de Giro*



Fonte: Autores.

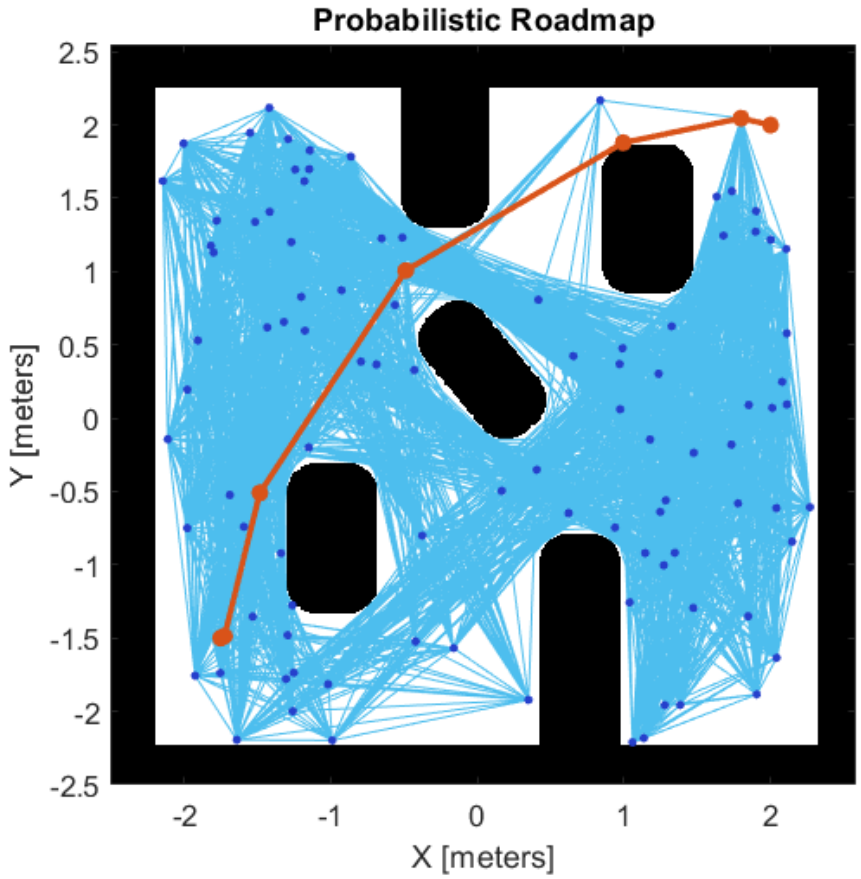
Como mencionado, também é possível fornecer as entradas via *Script*. Assim, o código fornece as mesmas velocidades, onde são definidos os valores para cada motor, atribuindo a variável a roda específica com os respectivos parâmetros, resultando em uma simulação idêntica. Este conecta-se remotamente, via API, o MatLab com o V-REP para troca de informações; neste caso, a imagem a ser tratada e o planejamento de caminho caulculado.

5.2 MAPA PROBABILÍSTICO DE ROTAS

Posteriormente, foi calculado uma função para gerar os 100 pontos (limite definido pelo usuário, pode ser mais ou menos pontos, a depender da precisão necessária) no mapa de rota probabilístico. Este define todas as rotas possíveis para trajetória do robô móvel no espaço de configuração adequado (parte branca da imagem binária aumentada).

Conhecendo todos os caminhos possíveis para locomoção, a menor trajetória a ser seguida pelo carro é traçada. Na Figura 12, está demonstrado em azul, as possíveis rotas as quais evitam os obstáculos e, em vermelho, o caminho a ser seguido do ponto inicial (-1.7500, -1.5000) ao ponto final (2.0000, 2.0000).

Figura 12. *Mapa com os Pontos de Locomoção e Trajetória*



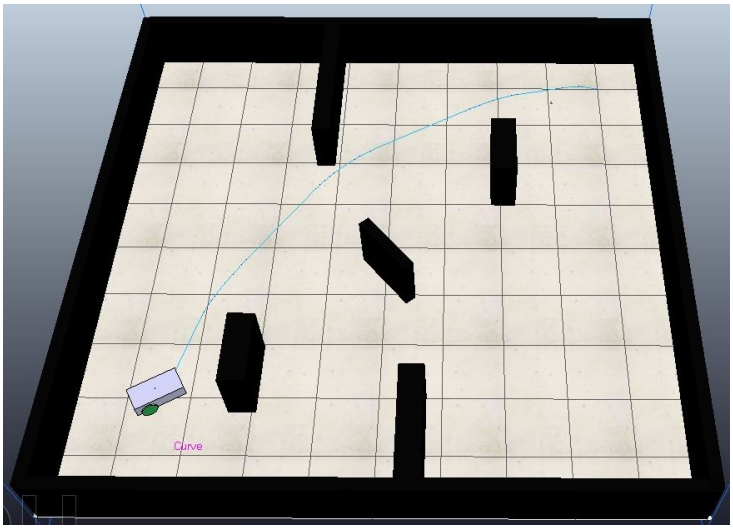
Fonte: Autores.

Os sete pontos definidos para o menor caminho são, respectivamente do ponto inicial ao final: (-1.7500, -1.5000), (-1.6676, -1.7067), (-1.3792, -0.5678), (-0.9062, 0.2122), (-0.1379, 1.2376), (1.3846, 2.1755), (2.0000, 2.0000).

Finalmente, usufruindo do mapa probabilístico e o caminho a ser seguido, é possível aplicar os controladores de velocidades linear e angular para seguir o trajeto planejado e evitar quaisquer colisões. A Figura 13 ilustra a cena do robô móvel na sua posição e orientação inicial, bem como,

a trajetória a ser percorrida, onde está evita os obstáculos no simulador *V-REP*.

Figura 13. *Curva Plotada no Simulador*



Fonte: Autores.

6 MAPEAMENTO POR NUVENS DE PONTOS

Para o terceiro e último experimento, foi proposto o desenvolvimento do mapa a partir dos dados resultantes pelos escaneamentos com o sensor *LIDAR*. Primeiro, realizou-se a movimentação por todo espaço de trabalho para definição dos obstáculos. Seguidamente, aplicou-se o mapa desenvolvido no gerador de caminho para execução do trajeto desejado do ponto inicial ao final; se colidir com os objetos mapeados e na posição/orientação desejadas.

6.1 MAPA MÉTRICO POR GRADE DE OCUPAÇÃO

Dentre as categorias de mapas métricos, o escolhido para aplicação nesse trabalho foi o Grade de Ocupação. O mapa resultante desta metodlogia, descreve o ambiente com um baixo nível de abstração e representa o espaço como uma união de células de formato regular, onde geralmente possuem o mesmo tamanho. Para cada uma dessas células pode-se definir um valor de ocupação que pode ser determinístico ou probabilístico.

Quando as medições obtidas para construção da grade de ocupação são ruidosas, utiliza-se a abordagem probabilística para a ocupação de cada célula. Dessarte, individualmente, estas recebem um valor de probabilidade referente a ocupância aferida, dependendo da confiabilidade das informações recebidas pelo sensor utilizado. Para a abordagem probabilística, geralmente definem-se os seguintes valores iniciais:

- Célula cheia: probabilidade de ocupação 100%;
- Célula vazia: probabilidade de ocupação 0%;
- Células sem informações disponíveis: probabilidade de ocupação 50%.

Dessa forma, é possível gerar um mapa por grade de ocupação a partir de dados sensoriais ruidosos, desde que, seja conhecida a posição do robô de forma precisa.

6.2 ALGORITMO DE MAPEAMENTO POR GRADE DE OCUPAÇÃO

O mapa m pode ser representado por um conjunto finito de células m_i regularmente espaçados:

$$m = \sum_i m_i \tag{6}$$

Para cada uma das células é associado um valor de probabilidade de ocupação $p(m_i)$.

A varredura do mapa e verificação das probabilidades de ocupação de cada célula foi desenvolvida pelo seguinte algoritmo ilustrado na Figura 14:

E a função de modelo inverso do sensor de alcance dada pelo seguinte código demonsrado na Figura 15:

Figura 14. *Algoritmo de Varredura.*

	GO($\{l_{t-1,i}\}$,z_t,q_t)
1	**para** todas as células m_i **faça**
2	**se** m_i está no campo perceptual de z_t **então**
3	$l_{t,i} = l_{t-1,i}$ + **modelo_inverso_sensor**(m_i,z_t,q_t) – l_0
4	**senão**
5	$l_{t,i} = l_{t-1,i}$
6	**fim se**
7	**fim para**
8	**retornar** $\{l_{t,i}\}$

Fonte: Pablo Javier.

Figura 15. *Algoritmo modelo inverso do sensor de alcance.*

	modelo_inverso_sensor(m_i,z_t,q_t)
1	(x_i,y_i) centro da célula m_i , θ_s = orientação do sensor no referencial do robô.
2	$r = [(x_i\text{-}x)^2 + (y_i\text{-}y)^2]^{1/2}$
3	$\varphi = \text{atan2}(y_i\text{-}y, x_i\text{-}x) - (\theta+\theta_s)$
4	**se** $r > \min(z_{max}, z_t+e/2)$ **ou** $\lvert\varphi\rvert > \alpha/2$ **então**
5	**retornar** l_0
6	**se** $z_t < z_{max}$ **e** $\lvert r - z_t\rvert < e/2$ **e** $\lvert\varphi\rvert < \alpha/2$ **então**
7	**retornar** l_{oc}
8	**se** $r \leq z_t - e/2$ **e** $\lvert\varphi\rvert < \alpha/2$ **então**
9	**retornar** l_L
10	**fim se**

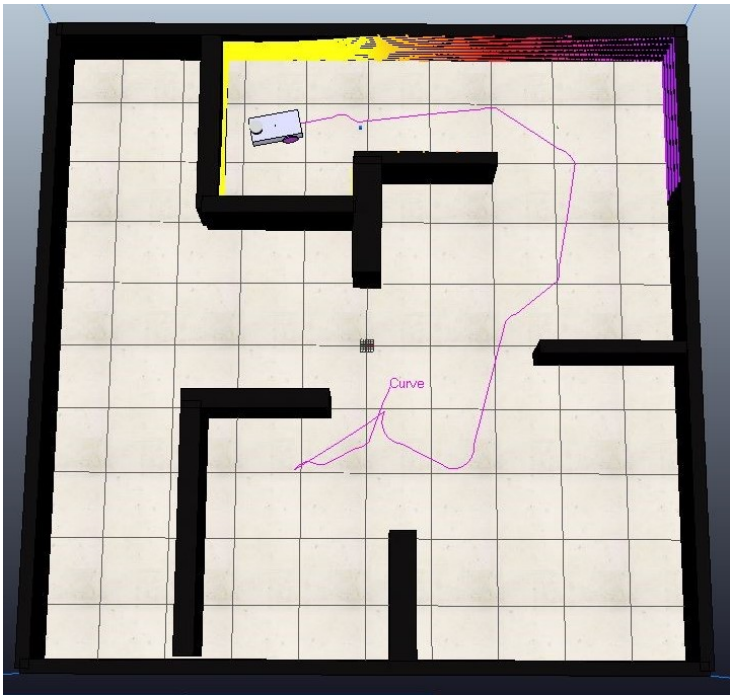
Fonte: Pablo Javier.

Dessa forma, no trabalho aqui descrito, foi realizada a varredura de todo o mapa e definidos os seguintes valores para a grade de ocupação:

• Célula cheia: probabilidade superior a 80%;
• Célula possivelmente cheia: probabilidade superior a 60%;
• Célula vazia: probabilidade inferior a 60%.

Para o terceiro projeto, alterou-se o ambiente localizado para uma melhor representação dos resultados. Ao iniciar o programa, o robô foi movimentado manualmente por todo o espaço, para mapear os obstáculos e desenvolver o mapa. Abaixo segue imagem do ambiente utilizado para o mapeamento para comparação com as medições obtidas:

Figura 16. *Ambiente utilizado para mapeamento.*



Fonte: Autores.

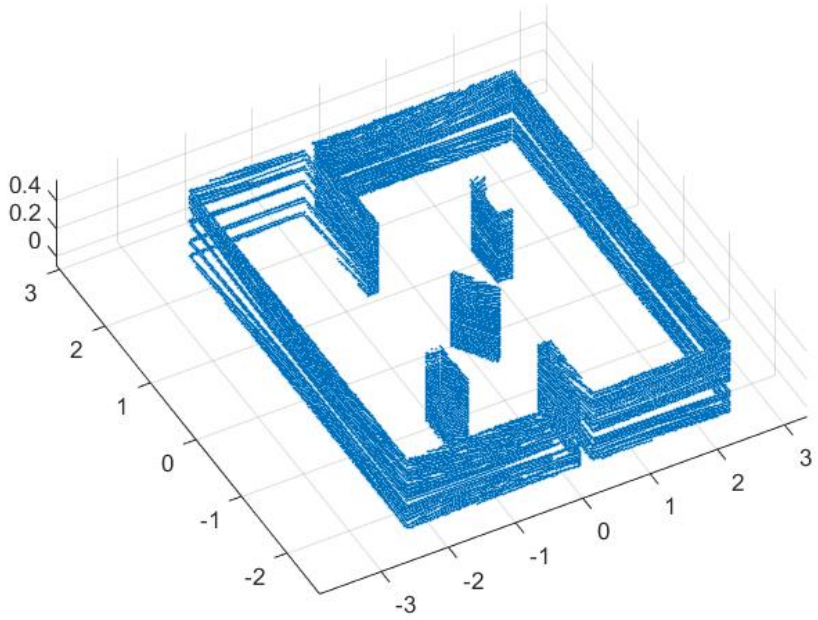
6.2.1 MAPEAMENTO DETERMINÍSTICO

Baseando-se no posicionamento do carro e no mapeamento manual, o mapa com a grade de ocupação em 3D, sensoriada pelo *LIDAR* está expresso na Figura 17.

O mapa foi convertido para 2D e as células que obtiveram uma probabilidade de ocupação superior a 60% na varredura são mostradas na Figura 18.

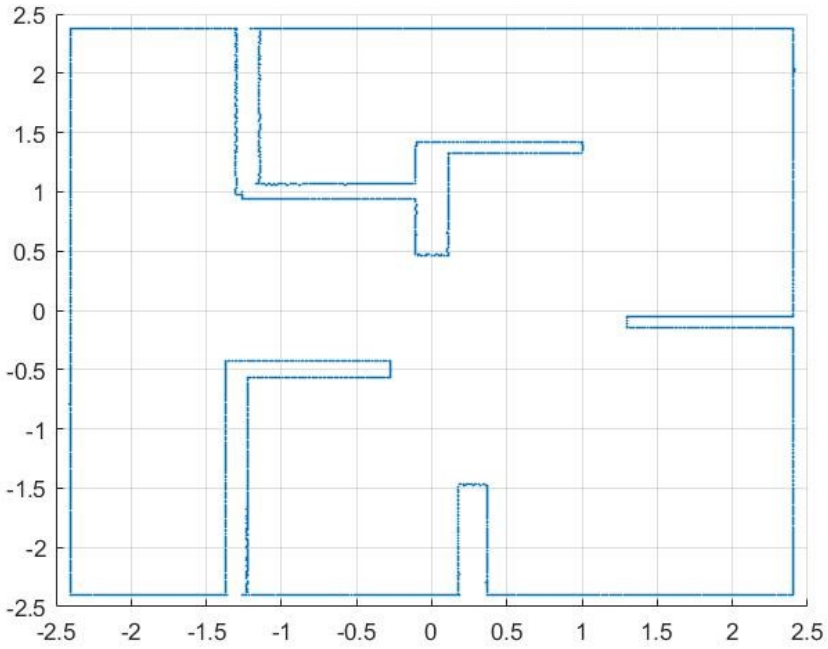
As demais células com probabilidade inferior a 60% foram definidas como células vazias, sendo então um espaço livre para deslocamento do robô. De tal forma, as células com probabilidade superior a 80% foram deliberadas como ocupadas na grade de ocupação do mapa, conforme exibido na Figura 19 abaixo.

Figura 17. *Scan 3D LIDAR.*



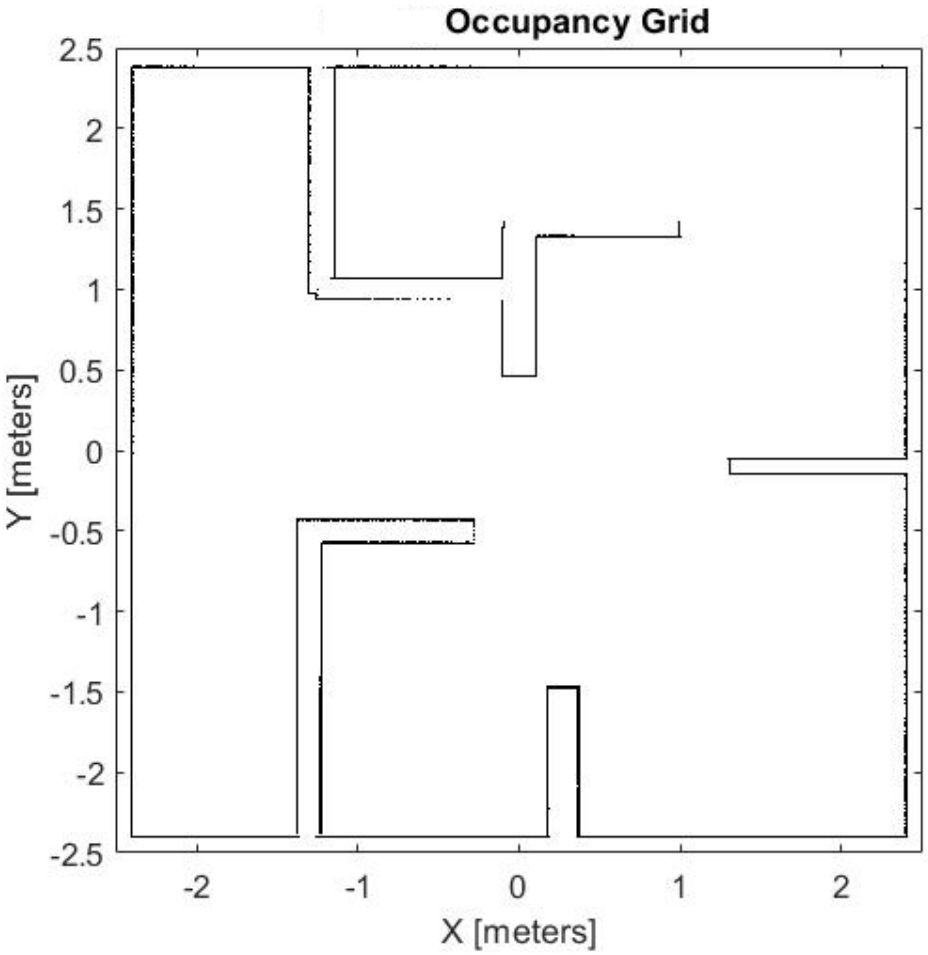
Fonte: Autores.

Figura 18. *Células com probabilidade superior a 60% de estarem ocupadas.*



Fonte: Autores.

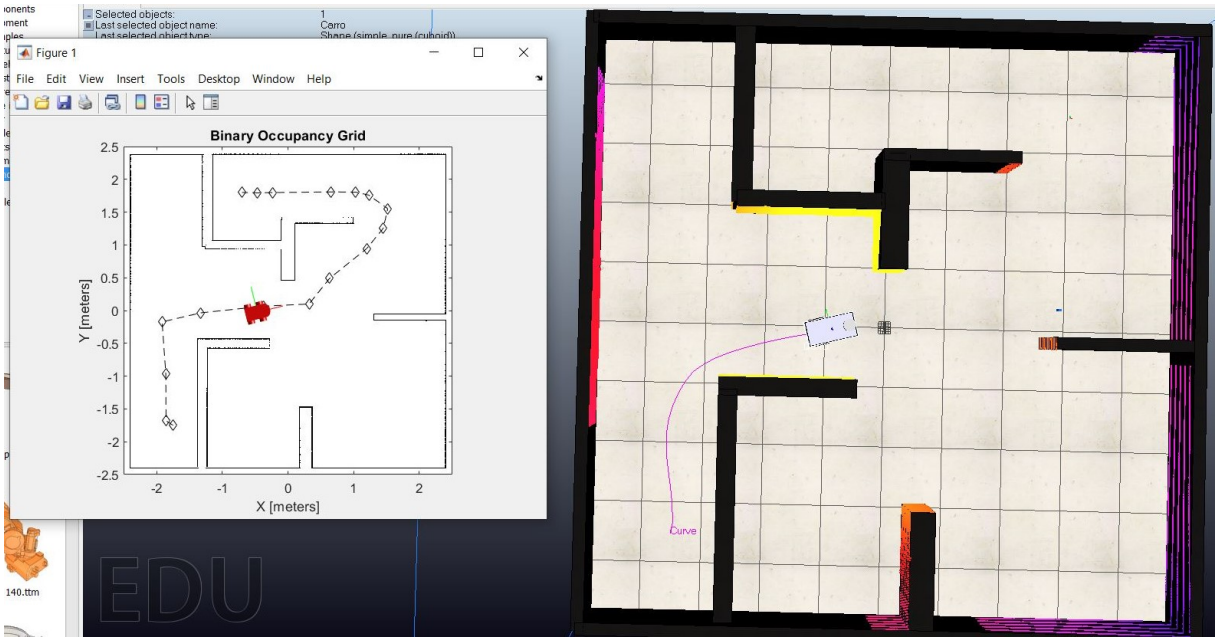
Figura 19. *Células com probabilidade superior a 80% de estarem ocupadas.*



Fonte: Autores.

Recorrendo ao método determinístico, a Figura 20, ilustra um *frame* do simulador, com o ambiente de trabalho e o controle do *Matlab* em tempo real. O robô percorre o caminho definido (janela à esquerda) e a curva percorrida, em rosa, é atualizada no simulador (janela à direita).

Figura 20. *Percorrendo caminho com mapeamento via LIDAR.*



Fonte: Autores.

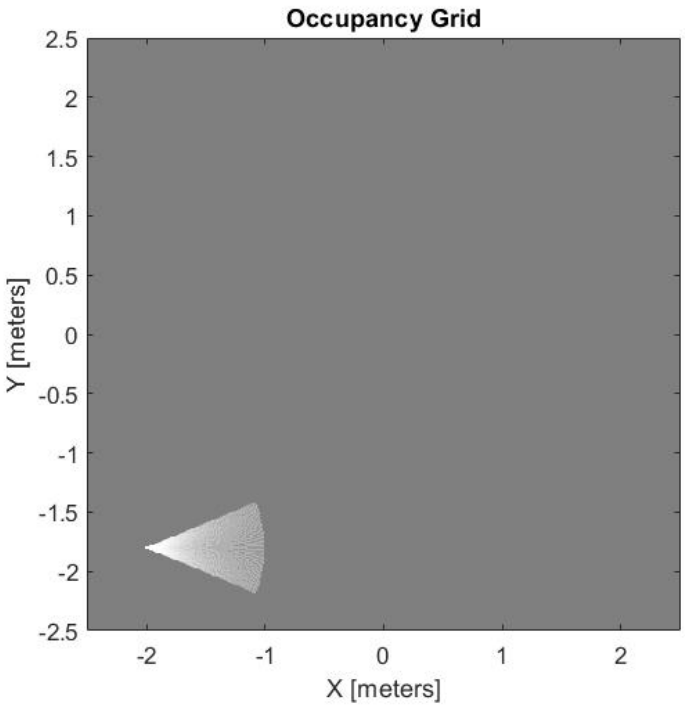
6.2.2 MAPEAMENTO PROBABILÍSTICO

O experimento com o método probabilístico, aderiu o mesmo espaço de trabalho para realizar o mapeamento. Com o mesmo sensor, procedeu-se a leitura, onde os dados de menor distância, resultaram em uma maior probabilidade de não existir um obstáculo a sua frente e, pela precisão do sensor, para maiores distâncias torna-se mesmo provável.

Como mencionado, as células vazias são inferiores a 60%, as quais estão ilustradas em branco na Figura 21 (os pontos em cinza representam obstáculos ou a não leitura da área). Nota-se, que o formato cônico referente a uma leitura, possui uma graduação do mais puro branco (probabilidade quase nula de obstáculo) até o mais mesclado ao cinza (maior incerteza, menor probabilidade de área livre).

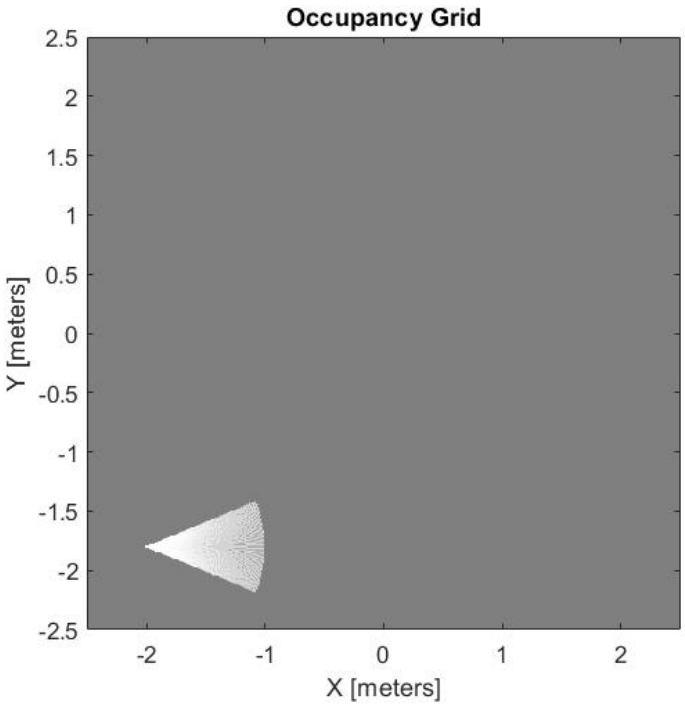
Na segunda leitura do robô, mantendo-se o posicionamento, a probabilidade de objeto à sua frente diminuiu. Tanto para os dados mais próximos, como também, os mais afastados. É possível comparar a diferença de coloração pela Figura 22, para tanto, demonstrando que com duas leituras, o sensor já consegue determinar a área livre a sua frente (ou objetos) e prosseguir por toda a área não mapeada do ambiente de trabalho.

Figura 21. *Probabilidade de existir um obstáculo de 22,85%.*



Fonte: Autores.

Figura 22. *Probabilidade de existir um obstáculo de 16,49%.*



Fonte: Autores.

7 CONCLUSÃO

Com a inovação de simuladores como este, pesquisadores de todos os níveis podem realizar suas pesquisas sem a necessidade física do dispositivo robótico. Assim, reduzindo ao máximo os possíveis erros de programação e custos, antes da aplicação no robô físico.

A partir desta, também, facilitou a compreensão da teoria mostrada em aula, principalmente, nos casos de escassez ou ausência dos matérias necessários para tal finalidade. Ademais, destaca-se a importância dessas ferramentas simulatórias, para o caso atual de pandemia, onde as aulas são remotas.

O controle segui preciso e suave, proporcionando uma boa estabilidade e condizente com a curva planejada. Ademais, o posicionamento e orientação deste, obteve erro ínfimo, o que condiz com as condições de erro induzidas pelo simulador.

O planejamento de caminho foi implementado com sucesso e a precisão é dada pelo limite de pontos escolhidos para tracejo das rotas sem colisões. Para tanto, dependendo da aplicação, a precisão pode ser maior, porém demanda proporcionalmente o aumento do gasto computacional. Como o ajuste da distância do raio de giro foi exatamente o tamanho do robô móvel, ressalta-se que deve-se considerar erros de dimensionamento para aplicações reais, onde deve ser ajustado o aumento dessa distância para garantir a não colisão deste.

O mapeamento probabilístico também obteve sucesso, tanto para o método determinístico, como para o probabilístico. A definição das células, logo os obstáculos, resultou em um mapa binário (preto e branco), fiel ao espaço de trabalho, com locomoção precisa do robô.

Finalmente, o experimento resultou nas proposições desejadas e este serve de base para os próximos, onde será otimizado o mapeamento por pontos e aplicado do sensor *sonar* paras leituras, o qual é menos preciso, mais ruidoso e com mais peculiaridades. Este servirá para aproximar-se dos erros reais e dificultar a simulação, induzindo ao aprendizado.

REFERÊNCIAS

[1] ROHMER, E.; SINGH, S. P. N.; FREESE, M. V-rep: A versatile and scalable robot simulation framework. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2013. p. 1321–1326. 4

[2] NASCIMENTO, L. B. P. et al. Introdução ao v-rep: Uma plataforma virtual para simulação de robôs. In: *Alex Oliveira Barradas Filho; Pedro Porfirio Muniz Farias; Ricardo de Andrade Lira Rabêlo. (Org.). Minicursos da ERCEMAPI e EAComp 2019. 1ed.Porto Alegre: Sociedade Brasileira de Computação*. [S.l.: s.n.], 2019. p. 49–68.

[3] OCCUPANCY Map. `https://www.mathworks.com/help/nav/ref/occupancymap.html`. Acessado em 15/08/2021.