Universidade Federal do Rio Grande do Norte

Programa de Pós-graduação em Engenharia Elétrica e de Computação

EEC1507 - Sistemas Robóticos Autônomos

RELATÓRIO

2º PROJETO DE SISTEMAS ROBÓTICOS AUTÔNOMOS - META 1

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 2º 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 primeira etapa/meta do segundo projeto avaliativo da disciplina de Sistemas Robóticos Autônomos do Programa de Pós Graduação em Engenharia Elétrica e de Computação (PPGEEC) da UFRN. O Objetivo final desse segundo trabalho é desenvolver planejadores de caminhos para robôs móveis de forma a fazer com que estes robôs consigam ir de uma configuração inicial até uma configuração final em um ambiente populado de obstáculos sem que haja colisão com os mesmos. O projeto é divide em três etapas: Trabalhar com obstáculos poligonais no espaço de trabalho em um simulador e tratar o espaço de configuração correspondente; Implementar planejador baseado em grafos e por fim a última etapa é implementar planejadores baseadores em campos de potenciais.

A implementação ao qual este trabalho se refere é capaz de obter o espaço de configuração (x, y, θ), como apresentado em (1), para qualquer conjunto de obstáculos e robô em formato de polígono convexo e verificar/identificar colisão no espaço de configuração, dentre outras coisas. Para isso foi utilizado o simulador CoppeliaSim (2) na versão EDU, nele foi criado um cenário contendo obstáculos poligonais e um robô com acionamento diferencial, para o controle do robô, bem como um cliente em C++ para realizar o controle do robô de forma remota por meio da interface B0RemoteAPI provida para comunicação cliente/servidor com o CoppeliaSim; exibição em tempo real do espaço de trabalho; obtenção e exibição do espaço de configuração e verificação de colisão.

2 REFERENCIAL TEÓRICO

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:

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

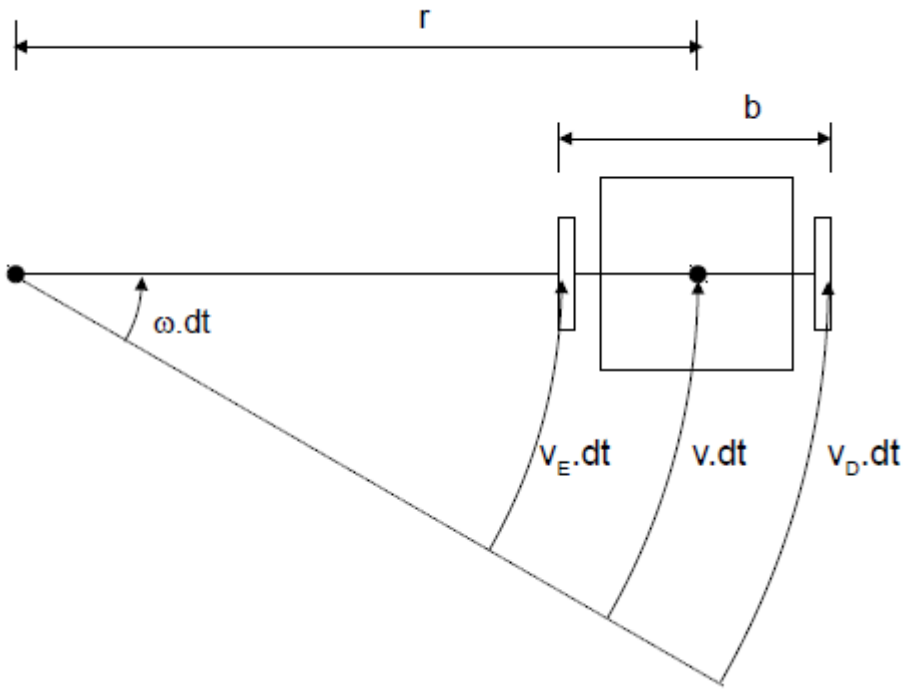


Figura 1 – Movimentos Infinitesimais para o modelo de robô utilizado.

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 OBSTÁCULOS EM ESPAÇO DE CONFIGURAÇÃO

Como já mencionado, neste trabalho foi utilizado como abordagem o problema de planejamento de caminho de forma a evitar obstáculos, a abordagem por espaço de configuração, como apresentado em (1). O método consiste em transforma os objetos presente no espaço de trabalho ($\mathbb{R}^2$ ou $\mathbb{R}^3$ por exemplo) para seus correspondentes em espaço de configuração $(x, y, z, row, pitch, yaw)\, ou\, (x, y, \theta)$ por exemplo, desta forma o robô móvel passar a ser representado como um ponto, facilitando assim o trabalho de planejamento de caminho.

Algumas definições que estão sendo adotadas neste trabalho:

- $\{A\}$: Referencial fixo em um robô A;
- $\{W\}$:Referencial fixo no espaço de trabalho W;
- $\{C\}$:Espaço de todas as configurações possíveis;
- $A(q)$:Subconjunto de W ocupado por A em q;
- CB: Conjunto de configurações em que o robô se superpõe a região de obstáculos B.

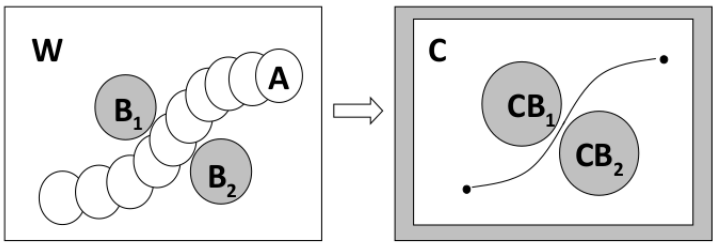


Figura 2 – Exemplo de passagem de espaço de trabalho para espaço de configuração para um robô circular.

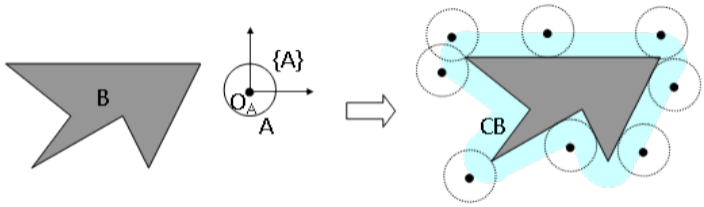


Figura 3 – Segundo exemplo de passagem de espaço de trabalho para espaço de configuração para um robô circular.

As figuras 2 3 ilustram um caso simples, apenas translacional, da transformação dos objetos para o espaço de configuração e como o robô passa a poder ser considerado um ponto. Para ambos os casos ilustrados nas imagens acima, o robô possui um formato circular, fazendo com que o processo de transformação dos objetos seja simplificado, sendo necessário apenas aumentar seus limites conforme o raio do robô.

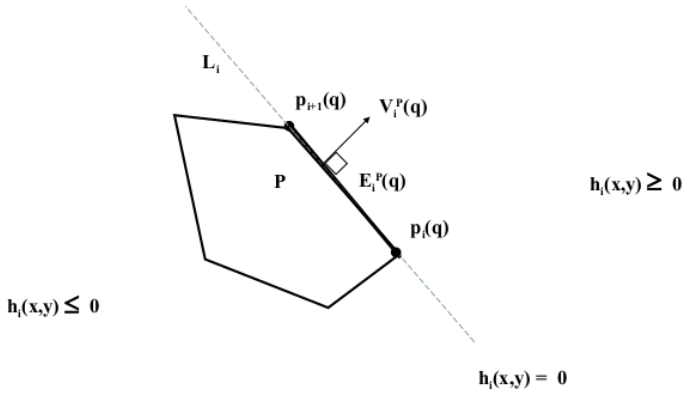


Figura 4 – ilustração de um polígono convexo orientado.

Como mencionado o algoritmo utilizado aqui para o calculo dos CB's obstáculos restringe-se à obstáculos com formas poligonais convexas, ou seja, qualquer ligação entre

pares de pontos na parte interna do objeto ainda pertence a essa região interna, além dessa propriedade os vértices desses polígonos devem ser ordenados no sentido anti-horário, isso faz-se necessário para facilitar a identificação da parte interna do objeto.

Com um polígono com seus vértices ordenados no sentido anti-horário, como na figura 4, é possível determinar se um ponto encontra-se no semi-plano esquerdo, direito ou acima da reta que liga os vértices p_i e p_{i+1}, dessa forma, a união dos semi planos esquerdo define a região interna do polígono convexo, como ilustrado na figura 5.

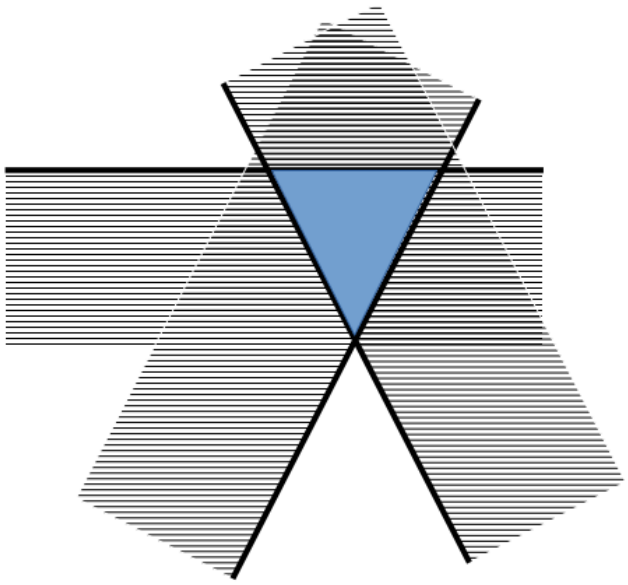


Figura 5 – Ilustração da união dos semi planos definidos pelos lados do polígono.

Uma vez que tenha-se o espaço de trabalho composto esses polígonos o algoritmo para obter os respectivos obstáculos em espaço de configuração para um θ_0 fixo será:

1. Fixar as normais $-V_i^A\,(i = 1, ..., n_A)$ e $V_j^B\,(j = 1, ..., n_B)$ no círculo unitário (S^1).
2. Varrer S^1 em sentido anti-horário
3. De acordo com o tipo de vértice (V^A ou V^B), criar os novos vértices:
   - Se $-V_i^A$ está entre V_{j-1}^B e V_j^B, criar vértice: $b_j - a_i(q_0)$;
   - Se V_j^B está entre $-V_{i-1}^A$ e V_i^A, criar vértice: $b_j - a_i(q_0)$;

A figura 4 ilustra esse procedimento para a criação de um CB obstáculo para um robô com $\theta = \theta_0$.

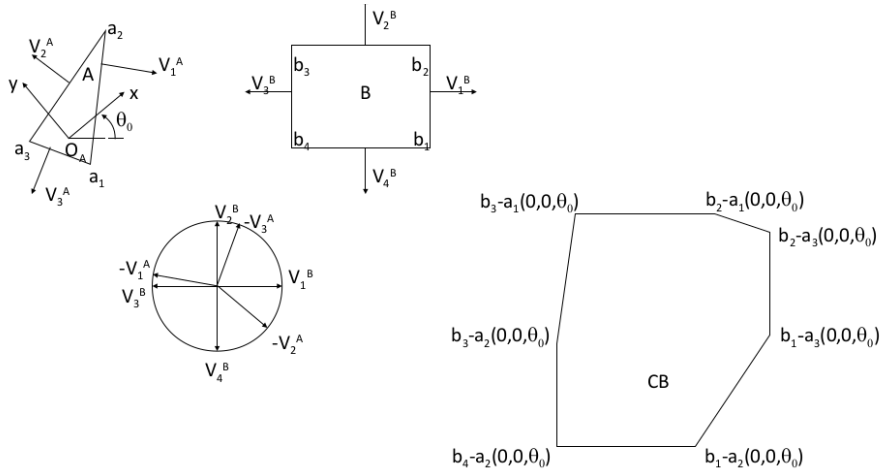


Figura 6 – Exemplo do procedimento para obtenção do CB obstáculo.

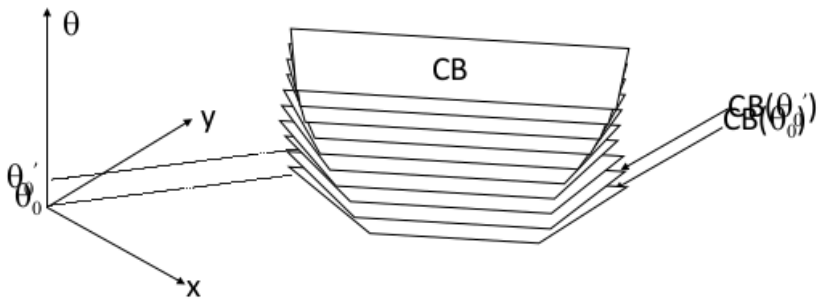


Figura 7 – Exemplo de um CB obstaculo no espaço de configuração (x, y, θ) para θ discretizado de $[0, 2\pi]$.

O procedimento para o cálculo do CB no caso translacional, ou seja para um determinado θ, pode ser repetido para diferentes orientações de forma a se obter de forma discreta o objeto CB no espaço de configuração para diferentes orientações, como ilustrado na figura 7.

3 METODOLOGIA

No simulador, como já mencionado, foi utilizado o CoppeliaSim(2), foi criado um cenário contendo obstáculos com formatos poligonais(ver figura 8), a tabela 1 apresenta de forma mais detalha sobre o posicionamento e o formato dos objetos contidos neste cenário. Uma vez que o cenário foi criado, foi posicionado um robô de acionamento diferencial (robô Pioneer, ver figura 9) com uma configuração inicial qualquer e habilitado o modo servidor do B0RemoteAPi(Api responsável por fornecer um canal para comunicação/controle remoto

do ambiente simulado), com isso encerra o trabalho realizado no lado do servidor/simulador.

O programa cliente foi desenvolvido em C++, foi criados diversas classes auxiliares e uma classe para tratar de diversas operações com polígonos no espaço 2D, além da implementação do algoritmo para computar os CB obstáculos, conforme apresentado na seção de referencial teórico.

Uma vez que essas classes auxiliares foram criadas, programou-se o cliente propriamente dito, o cliente inicia recriando o cenário/espaço de trabalho conforme no simulador em sua estrutura de dados que irá encapsula todos os obstáculos presentes no espaço de trabalho, neste instante também é criado o representante correspondente geométrico do robô, essa geometria pode ser qualquer polígono convexo, mas como melhor aproximação para o robô utilizado, é feito aproximação para um polígono circular (ver figura 10), essa aproximação do polígono para o circulo é ajustável, ou seja quanto mais vértices esse polígono possuir mais próximo será de um circulo perfeito.

Com a estrutura de dados do cliente contendo o correspondente do espaço de trabalho e o robô, é realizado a transformação para o espaço de configuração, esse procedimento é feito antes do loop principal, pois como os obstáculos são estáticos não há a necessidade de re-computar o espaço de configuração mais de uma vez, o programa irá criar os CBs obstáculos ao longo do eixo θ(orientação) de forma discreta, a quantidade de amostras no eixo θ é um parâmetro passado por linha de comando para o programa, assim como o número de vértices utilizados na aproximação para o formato circular do robô, essa ideai da discretização na orientação foi ilustrado na figura 7.

A última etapa da inicialização é lançar uma thread que será responsável pelo plot (plot realizado usando-se Gnuplot(3) e sua interface para C++) em tempo real, dois gráficos são exibidos e atualizados em tempo de execução, são eles: o espaço de trabalho e o espaço de configuração para a orientação atual, ou seja, é exibido apenas a "fatia"C espaço que corresponde a orientação atual, isso facilita a visualização, essas gráficos podem ser vistos nas figuras 11b 11a. Após lançar a thread responsável pelo plot, a thread principal (main) inicia o loop principal, onde é feito a comunicação com o servidor para obter constantemente a configuração atual do robô e para enviar as velocidades que devem ser aplicadas nos motores direito e esquerdo do robô, o loop permanece ativo até que se tenha passado um tempo pré-determinado no programa ou caso seja identificado colisão da configuração atual com algum CB obstáculo.

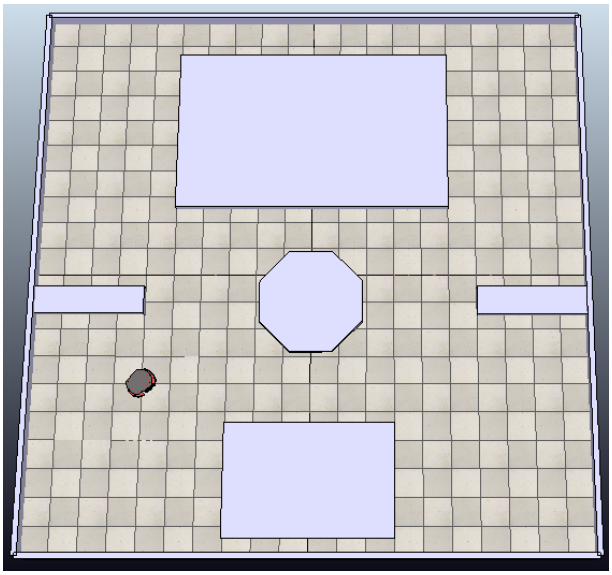


Figura 8 – Cenário criado no ambiente CoppeliaSim para este trabalho utilizado neste trabalho.

Id	**Descrição**	**Formato**	**Dimensões[m]**	**Posição** $(x[m], y[m])$
0	Parede inferior	Retangular	Largura: 10 Altura: 0.1	$(0.0, -5.0)$
1	Parede direita	Retangular	Largura: 0.1 Altura: 10	$(-5.0, 0)$
2	Parede superior	Retangular	Largura: 10 Altura: 0.1	$(0, 5.0)$
3	Parede esquerda	Retangular	Largura: 0.1 Altura: 10	$(-5.0, 0)$
4	Retângulo esquerdo	Retangular	Largura: 2 Altura: 0.5	$(-4.0, -0.5)$
5	Retângulo direito	Retangular	Largura: 2 Altura: 0.5	$(4.0, -0.5)$
6	Retângulo inferior	Retangular	Largura: 3.0 Altura: 2.0	$(0, -3.7)$
7	Retângulo superior	Retangular	Largura: 5.0 Altura: 3.0	$(0, 2.75)$
8	Octágono central	Octogonal	Raio:1	$(0, 0)$

Tabela 1 – Descrição dos polígonos no cenário.

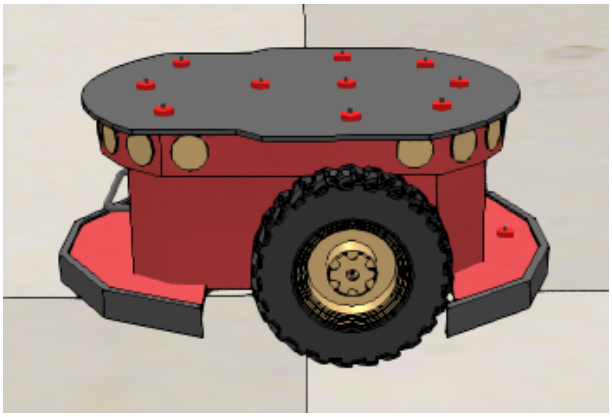


Figura 9 – Modelo simulado do robô Pioneer no CoppeliaSim.

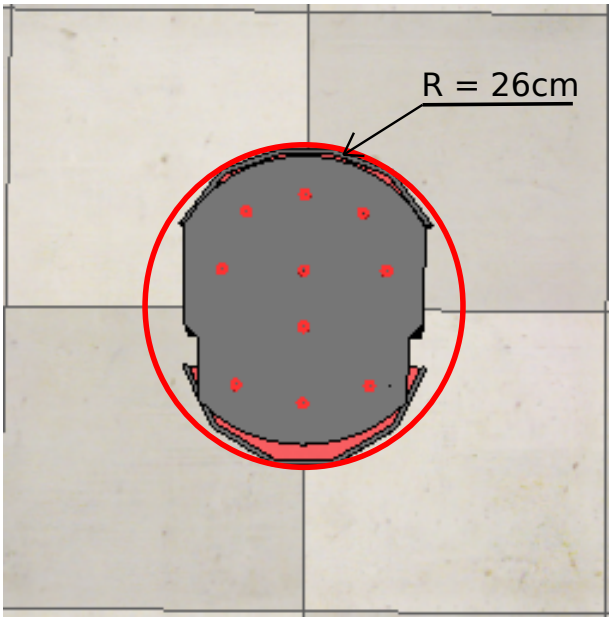


Figura 10 – Aproximação considerada do formato do robô Pioneer para um circulo de radio $26cm$.

A figura 10 mostra a aproximação do formato do Pioneer para um circulo com raio de 26cm.

Especificações do Pioneer:

- Raio de cada roda: $0.09751m$ (r_w)
- Distância entre os centros das rodas $0.331m(b)$

4 RESULTADOS E CONCLUSÕES

Os resultados apresentados nesta seção são provenientes de dois experimentos, duas simulações diferentes, ambos iniciam na configuração $(x = -3.0, y = -2.0, \theta = 45^o)$ e realizam dois tipos de movimentos, sendo eles:

Para $t \leq 30.0s$:

$$\begin{cases} V & = 0.0m/s \\ \omega & = 2.0\pi/30.0\text{rad/s} \end{cases}$$

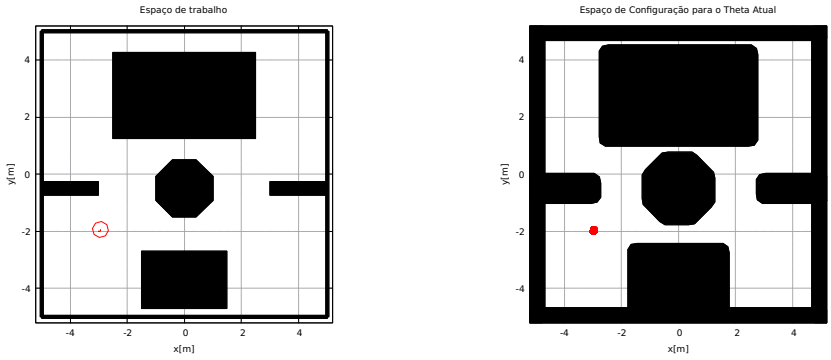
O que faz com que o robô realize um movimento circular em torno do próprio eixo no sentido anti-horário, levando $30s$ para realizar uma volta completa. Esse movimento é interessante, pois assim é visualizar a mudança dos CBs obstáculos conforme passa-se por diferentes orientações. A observação da mudança do formato dos CBs obstáculos só é possível se o robô possuir um formato suficientemente diferente de um circulo, pois caso contrário os obstáculos nesse espaço não irão alterar com a mudança da orientação, devido a simetria circular.

Para $t > 30.0s$:

$$\begin{cases} V & = 0.2\text{m/s} \\ \omega & = 0.1\text{rad/s} \end{cases}$$

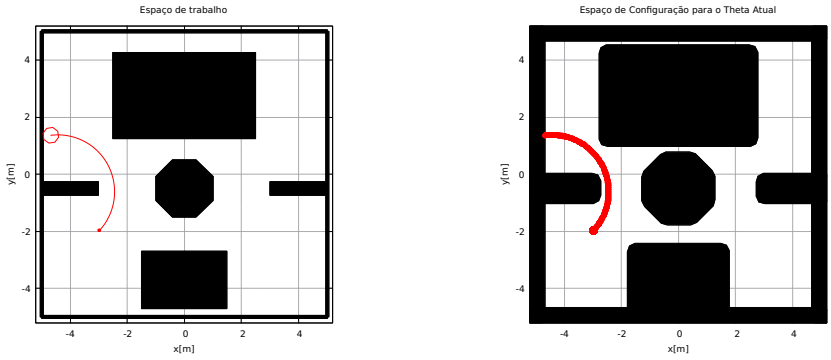
Esse segundo padrão de movimento é apenas para acionar a condição de para por colisão, dessa forma o robô realizara um movimento com velocidade linear diferente de zero, fazendo com que ocasionalmente ele colida com algum obstaculo, nesse caso ele irá colidir com o objeto 3 (parede esquerda).

Ambos os experimentos foram feitos passando-se como parâmetro para o programa, o corresponde ao número de amostras no eixo da orientação, igual a 200, ou seja, foi realizado uma amostragem de 200 ângulos entre $[0, 2\pi)$ e no experimento 1 foi utilizado 8 vértices na aproximação do circulo e no experimento 2 é utilizado 32. Os resultados destes experimentos podem ser visto nas figuras abaixo.



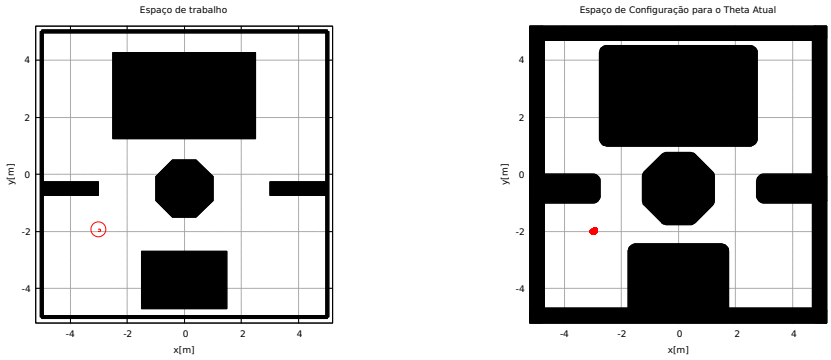
(a) Espaço de trabalho (b) Espaço de configuração.

Figura 11 – Experimento 1, Configuração inicial.



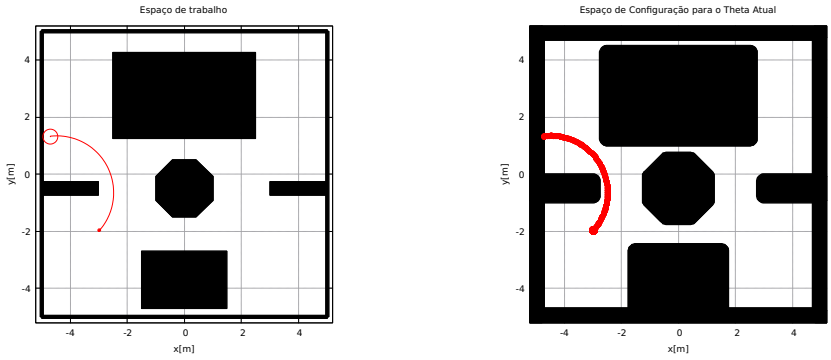
(a) Espaço de trabalho (b) Espaço de configuração.

Figura 12 – Experimento 1, Configuração final.



(a) Espaço de trabalho (b) Espaço de configuração.

Figura 13 – Experimento 2, Configuração inicial.



(a) Espaço de trabalho (b) Espaço de configuração.

Figura 14 – Experimento 2, Configuração final.

Referências

1 Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32, n. 2, p. 108–120, 1983. 1, 3

2 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, 6

3 WILLIAMS, T.; KELLEY, C.; many others. *Gnuplot 4.6: an interactive plotting program*. 2013. <http://gnuplot.sourceforge.net/>. 7

5 ANEXO 1

Código fonte: <https://github.com/Gabriellgpc/Sistemas_Roboticos/tree/master/program/p2m1>

Vídeo de demostração: <https://www.youtube.com/watch?v=KIsikoNiXtg&t=6s>