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 3

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 terceira 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 divido 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 pôr fim a última etapa é implementar planejadores baseados em campos de potenciais.

Este trabalho se propõe a descrever a solução adotada pela equipe para a tarefa de planejar caminhos livros de obstáculos utilizando-se de campos de potenciais, para isso foi aproveitado todo o conhecido desenvolvido e trabalhado nas metas anteriores à essa, ou seja, faz-se uso do 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, porém é acrescentado neste trabalho a funcionalidade de planejamento de caminho livre de obstáculos. 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 , verificação de colisão bem como calcular e exibir o caminho livre de obstáculos.

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:

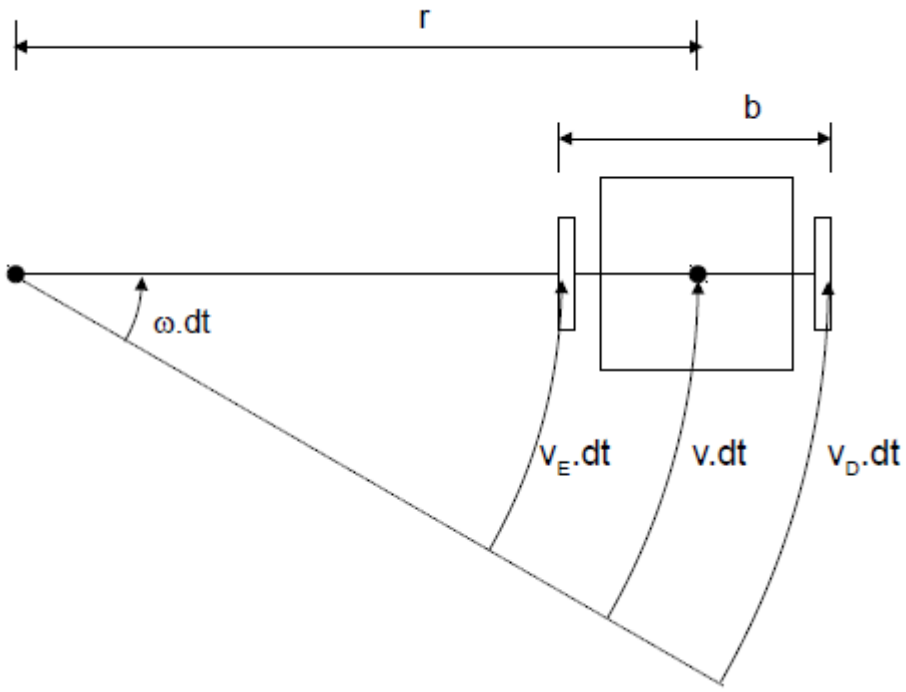

Figura 1 – Movimentos Infinitesimais para o modelo de robô utilizado.

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 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 transformar 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.

As Figuras 2 e 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.

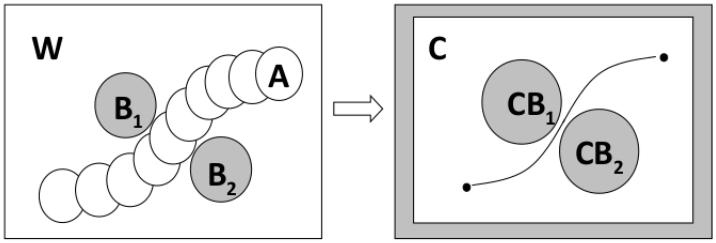

Figura 2 – Exemplo de passagem de espaço de trabalho para espaço de configuração para um robô circular.

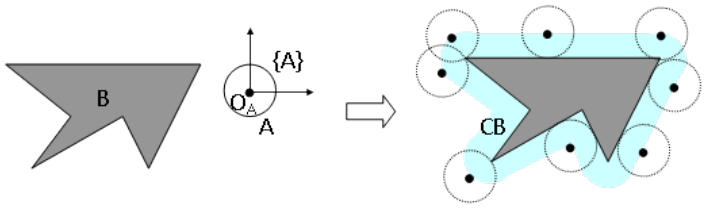

Figura 3 – Segundo exemplo de passagem de espaço de trabalho para espaço de configuração para um robô circular.

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.

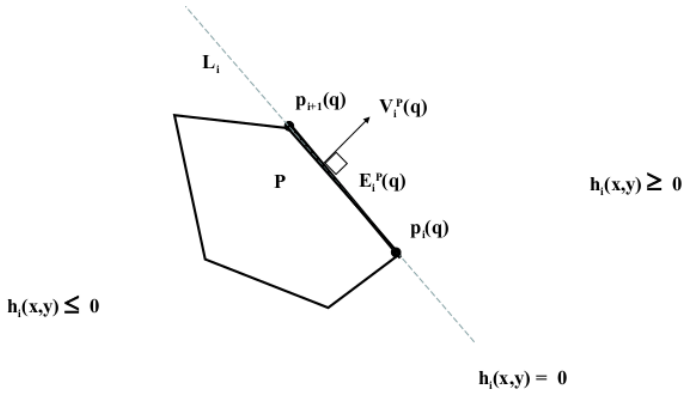

Figura 4 – ilustração de um polígono convexo orientado.

Como mencionado o algoritmo utilizado aqui para o cálculo dos CB's obstáculos restringem-se à obstáculos com formas poligonais convexas, ou seja, qualquer ligação entre pares de pontos na parte interna do objeto ainda pertencem 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 semiplano esquerdo, direito ou acima da reta que liga os vértices p_i e p_{i+1}, dessa forma, a união dos semiplanos esquerdo definem a região interna do polígono convexo, como ilustrado na Figura 5.

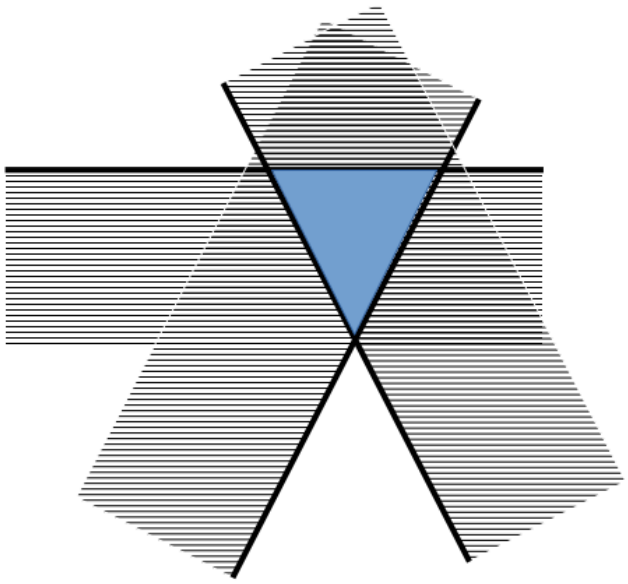

Figura 5 – Ilustração da união dos semiplanos definidos pelos lados do polígono.

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 $\theta = \theta_0$.

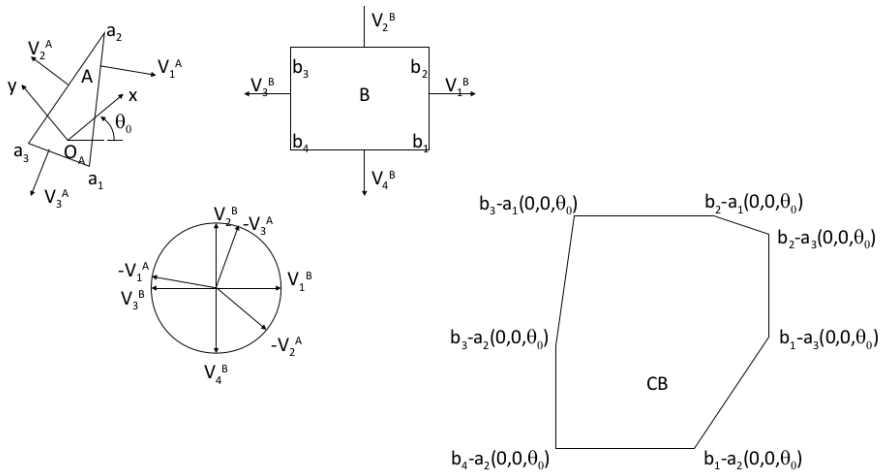

Figura 6 – Exemplo do procedimento para obtenção do CB obstáculo.

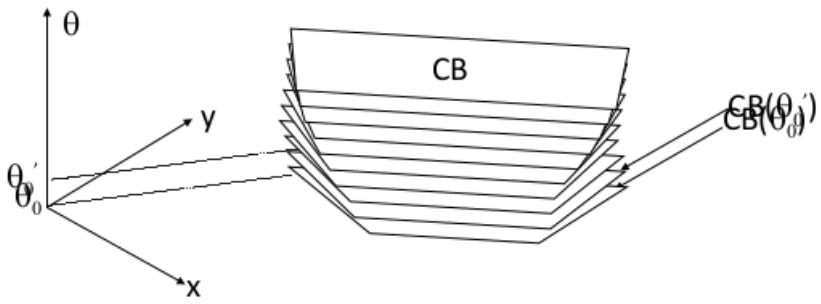

Figura 7 – Exemplo de um CB obstaculo no espaço de configuração (x, y, θ) para θ discretizado de $[0, 2\pi]$.

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 a forma discreta do objeto CB no espaço de configuração para diferentes orientações, como ilustrado na Figura 7.

2.3 CONTROLE DE CAMINHO

Baseado em um referencial Serret-Frenet(**S-F**), onde é localizado no ponto de projeção ortonormal, o vetor tangente, o vetor normal e o vetor binomial, do robô sobre o caminho onde o referencial **S-F** move-se, ou seja, a posição do robô tem que convergir para o alvo virtual $p^*(s)$, que está se movendo junto com o referencial **S-F**. Para Minimizar os erros o controlador usa as seguintes equações:

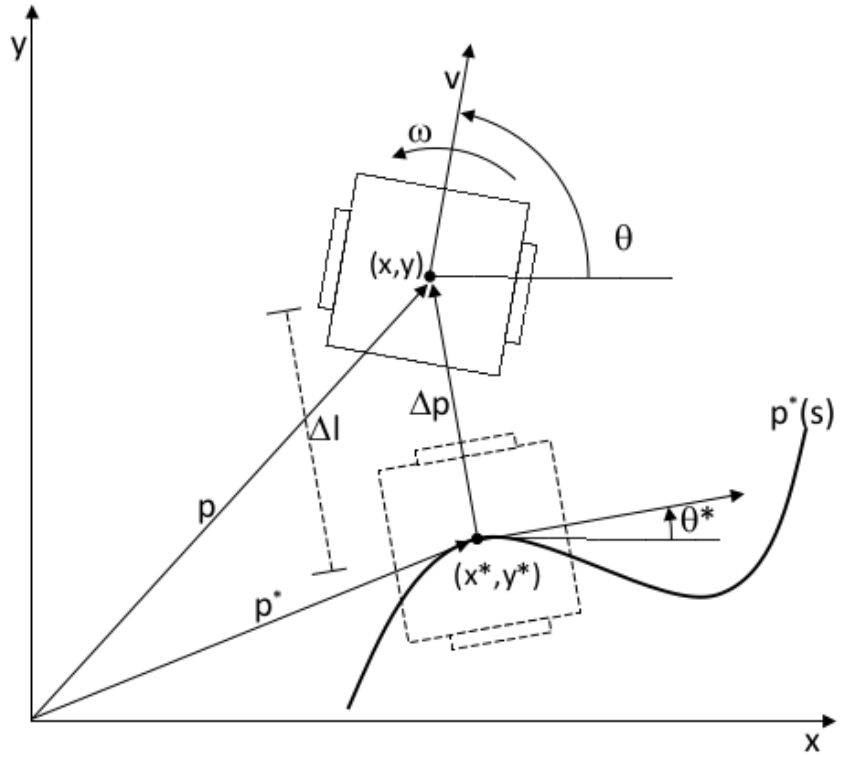

Figura 8 – Descrição do robô em relação ao caminho: baseada em um referencial Serret-Frenet.

$$\Delta l = \sqrt{(x - x^*)^2 + (y - y^*)^2} \tag{9}$$

$$\Delta\theta = \theta - \theta^* \tag{10}$$

Onde o caminho tem como característica uma curvatura inversa ao raio de giro, $\kappa(s)$. Logo θ^* tem que satisfazer a seguinte equação:

$$\frac{d\theta^*}{dt} = \kappa(s).\frac{ds}{dt} \tag{11}$$

O alvo virtual só existe se k(s) possuir um limite superior a p*(s), não podendo ter curvas abruptas e o robô não pode se afastar muito do caminho, pois a parametrização Serret-Frenet é local, logo o modelo cinemático em relação ao referencial **S-F** é:

$$\begin{aligned}
\frac{d(\Delta l)}{dt} &= v\sin(\Delta\theta) \\
\frac{d(s)}{dt} &= \frac{v\cos(\Delta\theta)}{1 - \kappa(s)\Delta l} \\
\frac{d(\Delta\theta)}{dt} &= \omega - \frac{\kappa(s)v\cos(\Delta\theta)}{1 - \kappa(s)\Delta l} = u
\end{aligned}$$

Para a lei de controle de Samson a expressão dos erros de seguimento de caminho são:

$$\frac{\Delta l}{dt} = v \sin(\Delta\theta)$$
$$\frac{\Delta\theta}{dt} = u$$

Onde u é uma nova estrada de controle para o erro angular. A lei de controle cinemático de **Samson** para o seguimento de caminho são:

$$v = \text{constante} \tag{12}$$
$$u = -\left(K_\theta \Delta\theta + \frac{K_l.\Delta l.v.\sin(\Delta\theta)}{\Delta\theta}\right) \tag{13}$$
$$\omega = u + \frac{\kappa(s).v.\cos(\Delta\theta)}{1-\kappa(s)\Delta l} \tag{14}$$

Onde K_θ e $K_l > 0$.

Com está lei de controle acaba-se minimizado a função de Lyapunov para:

$$v(\Delta l, \Delta\theta) = \frac{K_l.\Delta l^2 + \Delta\theta^2}{2}$$

2.4 PLANEJAMENTO DE CAMINHOS POR CAMPOS POTENCIAIS

O método baseado em Campos Potenciais, o robô é considerado uma partícula e move-se sob a influência de um campo de potencial artificial, de tal modo que os obstáculos geram um campo potencial de repulsão contra o robô e a configuração final o robô, influenciando nas assim o espaço livre. A partir deste pressuposto, a cada configuração, a direção do movimento do robô é determinada pela força resultante proveniente do campo potencial. Este método foi desenvolvido originalmente para contornar obstáculos, sendo aplicável onde não se dispõe de um modelo destes obstáculos inicialmente e são recebidos de forma on-line, tem um eficiente resultado em tempo real, mas tem uma perda garantida no alcance do alvo, possuindo uma implementação simplificada e geralmente rápida.
É caracterizado pelo uso de um método "local" e otimizações baseadas no gradiente, o que pode trazer problemas com os mínimos locais, no qual necessita de algumas soluções, como incluir técnicas para escapar de mínimos locais e definir funções de potencial, de tal modo que haja um ou poucos mínimos locais, o que o torna um método global. É considerado um método incompleto, pois pode falhar na busca mesmo que o caminho exista.

O planejamento baseado em campos potenciais possuí como base as seguintes características básicas:

- Robô é interpretado como uma partícula imersa em um campo de potencial artificial **U** que representa de forma indita o espaço livre;

- Obstáculos geram potenciais repulsivos;

- Alvo/Configuração destino possuí potencial atrativo;

- O caminho livre de obstáculos é gerado iterativamente/incrementalmente, seguindo a direção do negativo do gradiente do campo potencial artificial **U**;

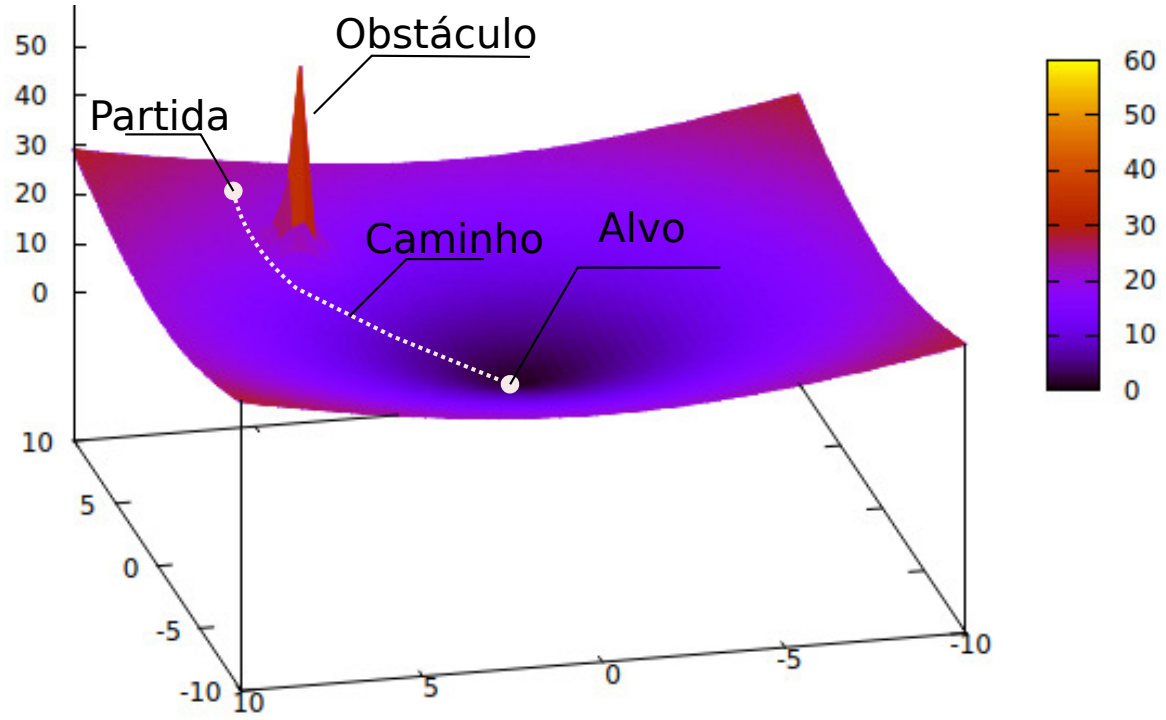

Figura 9 – Ilustração de um planejamento de caminho usando função de potencial artificial.

O planejamento do caminho é realizado de forma incremental, onde segue a direção da força artificial induzida em direção ao negativo do gradiente da função potencial.

$$F(q) = -\nabla U(q) \tag{15}$$

Induzido por **U** na configuração **q** é definindo a direção do movimento mais promissor e incremento da posição.
No campo de potencial para o caso translacional

$$W = \mathbb{R}^n$$

$$C = \mathbb{R}^n$$

Onde, **n = 2 ou 3**.

A função potencial

$$U(q) = U_{atr}(q) + U_{rep}(p) \tag{16}$$

Onde,

$U_{atr}(q)$ é a função de potencial atrativo, associado ao alvo e independente da região de C-obstáculos.

$U_{rep}(q)$ é a função de potencial repulsivo, associado a região de C-obstáculos e independente do alvo.

A força artificial

$$F(q) = -\nabla U(q) = F_{atr}(q) + F_{rep}(q) \tag{17}$$

$F_{atr}(q) = -\nabla U_{atr}(q)$ é a força atrativa que empurra em direção ao alvo.

$F_{rep}(q) = -\nabla U_{rep}(q)$ é a força repulsiva que afasta o robô dos obstáculos.

O potencial atrativo baseado em potencial paraboloide

$$U_{atr}(q) = \xi.[\rho_{fin}(q)]^2/2$$

Onde,

$\rho_{fin}(q) = ||q - q_{fin}||$ é a distância euclidiana ao alvo e ξ é fator de ganho positivo.

$U_{atr}(q) \geq 0$, com mínimo em q_{fin} e $\rho_{fin}(q)$ diferenciável em $\forall q \in C$.

A força atrativa fica então

$$F_{atr}(q) = -\nabla U_{atr}(q) = -\xi.[\rho_{fin}(q)].\nabla \rho_{fin}(q) = -\xi.(q - q_{fin})$$

Algumas observações devem ser analisadas como, $F_{atr}(q)$ converge linearmente para zero quando se aproxima da configuração alvo, no qual a $F_{atr}(q)$ tende a infinito para $\rho_{fin} \to \infty$ e a estabilidade assintótica pode ser alcançada quando se adiciona uma força dissipativa que é proporcional à velocidade dq/dt.

Para o potencial atrativo baseado em potencial cônico

$$U_{atr}(q) = \xi.\rho_{fin}(q)$$

$U_{atr}(q) \geq 0$, com mínimo em q_{fin}. $\rho_{fin}(q)$ é diferenciável em $\forall q \in C$, exceto em $q = q_{fin}$.

Para este modelo a força atrativa será então

$$F_{atr}(q) = -\xi.\nabla \rho_{fin}(q) = -\xi.(q - q_{fin})/||q - q_{fin}|| \tag{18}$$

Pode ser observado que $||F_{atr}(q)||$ é constante em $\forall q \in C$, exceto quando $q = q_{fin}$, a $F_{atr}(q)$ não vai possuir características estabilizantes, visto que não tenderá a zero quando $q \to q_{fin}$.Logo uma alternativa é mesclar o perfil cônico, longe de q_{fin} com perfil parabólico, nas vizinhanças de q_{fin}.

Para o potencial repulsivo, no qual é uma barreira de potencial entorno do CB, onde é desejável que $U_{rep}(q)$ não afete o movimento do robô quando este estiver suficientemente

longe dos obstáculos.
Se $\rho(q) \leq \rho_0$

$$U_{rep}(q) = (\eta/2).[(1/\rho(q)) - (1/\rho_0)]^2$$

Se $\rho(q) > \rho_0$

$$U_{rep}(q) = 0$$

onde, η é o fator de ganho positivo, o $\rho(q) = min_{q' \in CB}||q - q'||=$ distância de q a CB e ρ_0 é a Distância de influência com parâmetro positivo. Podendo observar que $U_{rep}(q)$ é positivo para $\rho(q) < \rho_0$, tendendo a infinito quando o robô se aproxima do obstáculo e $U_{rep}(q)$ é nulo quando o robô se afasta de CB mais longe do que ρ_0. Para a força repulsiva em obstáculos convexos, a região convexa CB, com limites diferenciáveis por partes, então $\rho(q)$ é diferençável pra $\forall q \in C_L$. Onde a força repulsiva

$$F_{rep}(q) = -\nabla U_{rep}(q)$$

Se$\rho(q) \leq \rho_0$

$$F_{rep}(q) = \eta.[(1/\rho(q)) - (1/\rho_0)].(1/\rho(q))^2.\nabla\rho(q)$$

Se$\rho(q) > \rho_0$

$$F_{rep}(q) = 0$$

Enquanto que para a força repulsiva para objetos não convexos a solução é decompor CB em componentes convexas $CB_k, k = 1, ..., r$, com uma função potencial seja associada em $U_k(q)$. Onde o potencial repulsivo total será a soma dos potenciais gerados por cada componente convexa

$$U_{rep}(q) = \sum U_k(q)$$

Se$\rho_k(q) \leq \rho_0$

$$U_k(q) = (\eta/2).[(1/\rho_k(q) - (1/\rho_0(q)]^2$$

Se$\rho_k(q) > \rho_0$

$$U_k(q) = 0$$

Onde $\rho_k(q)$ é a distância de q a CB_k.
Logo a força repulsiva correspondente a $U_{rep}(q)$ é dada por

$$F_{rep} = \sum F_k(q)$$

$$F_k = -\nabla U_k(q)$$

O Problema para uma dada métrica d geralmente não existe método eficiente e simples para computar $\rho(q)$ e seu gradiente $\nabla\rho(q)$, mas pode ser solucionado definindo os potenciais atrativos e repulsivos em W e combinar seus efeitos em vários pontos de A.

3 METODOLOGIA

No simulador, como já mencionado, foi utilizado o CoppeliaSim(2), foi criado um cenário contendo obstáculos com formatos poligonais(ver Figura 10). Uma vez que o cenário foi criado, foi posicionado um robô de acionamento diferencial (robô Pioneer, ver Figura 11) 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++, foram 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 e as funções de potencial artificial, conforme apresentado na seção do 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 12), essa aproximação do polígono para um círculo ajustável, ou seja, quanto mais vértices esse polígono possuir mais próximo será de um círculo 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 CB's 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 ideia da discretização na orientação foi ilustrado na Figura 7, com o espaço de espaço de trabalho abstraído e o espaço de configuração computado é realizado em seguida o planejamento do caminho propriamente dito, onde é criado a função de campo potencial artificial tendo os obstáculos uma contribuição repulsiva, em que a intensidade é igual ao quadrado da área do obstáculo e a distância de interação com essa força repulsiva é igual a três(3) vezes o raio do obstáculo e o ponto central do campo de repulsão é o centroide do mesmo. A configuração destino também é passada como parâmetro para a função de planejamento de caminho, sendo essa a única força atrativa no campo, como explicado nas seções anteriores.

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, esses gráficos podem ser vistos na Figura 15. 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, sendo essas velocidades saídas do controle seguidor de caminho que está atuando para garantir que o robô siga o caminho planejado na etapa inicial. O *loop* permanece ativo até que: se tenha passado um tempo pré-determinado no programa; seja identificado colisão da configuração atual com algum CB obstáculo ou o robô tenha completado a trajetória, ou seja, chegado a uma distância consideravelmente perto da configuração alvo.

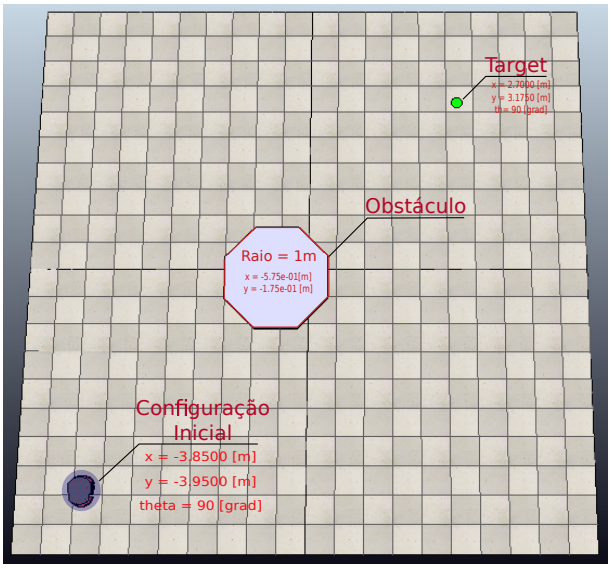

Figura 10 – Cenário criado no ambiente CoppeliaSim utilizado para este trabalho.

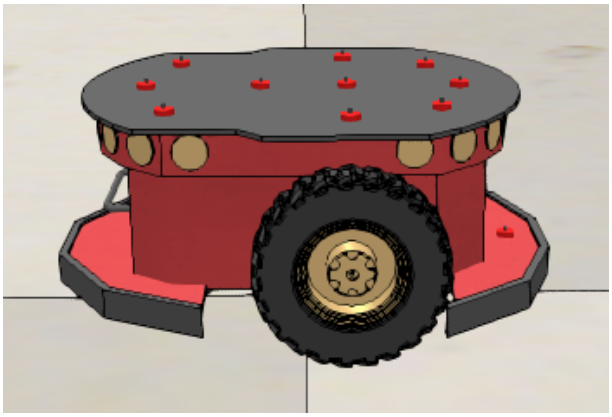

Figura 11 – Modelo simulado do robô Pioneer no CoppeliaSim.

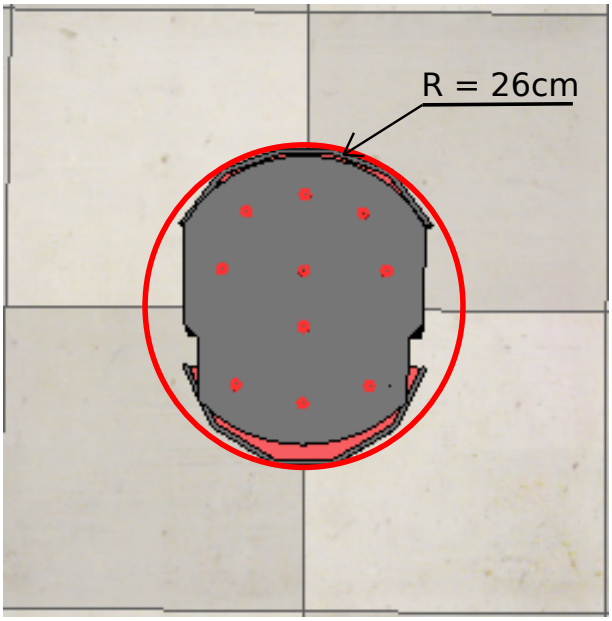

Figura 12 – Aproximação considerada no formato do robô Pioneer para um circulo de radio $26cm$.

A Figura 12 mostra a aproximação do formato do *Pioneer* para um circulo com raio de 26cm.

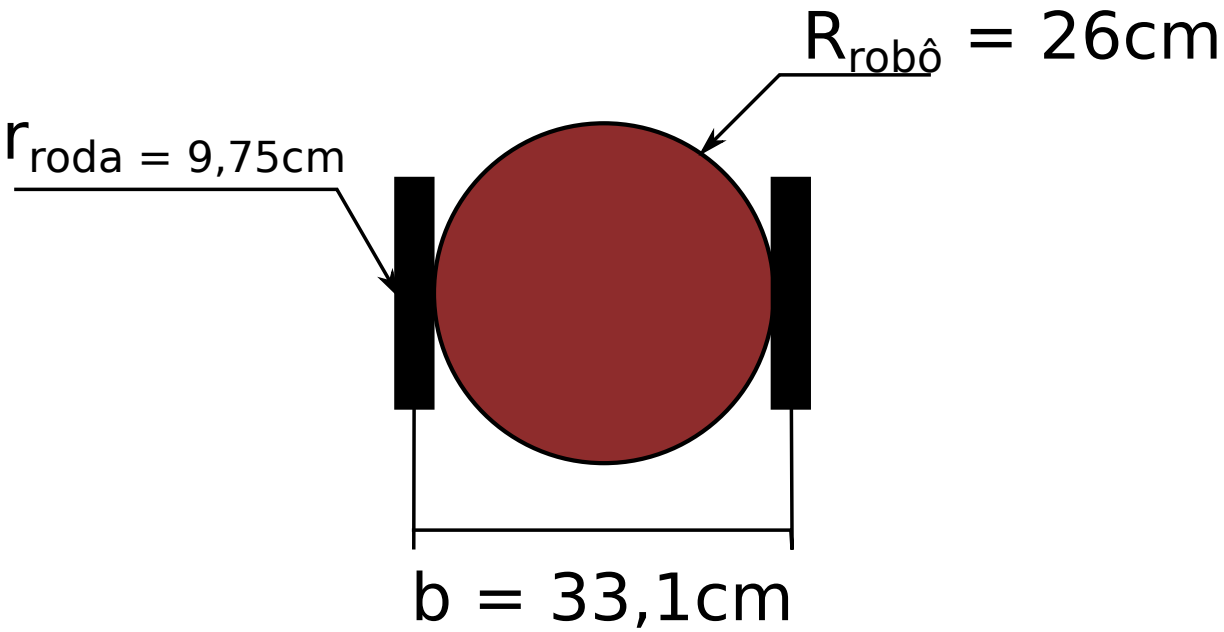

Figura 13 – Ilustração do modelo aproximação usada sobre o *Pioneer* com as dimensões do modelo real.

Especificações do Pioneer:

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

Os seguintes ganhos para o controle seguidor de caminho:

$$\begin{cases} K_{angular} & = 1.55 \\ K_{linear} & = 0.5 \end{cases}$$

Ambos os experimentos foram feitos passando-se como parâmetro ao programa, o correspondente 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. Já para o planejamento do caminho por campos potenciais, foram considerados como raio de ação mínima de repulsão dos obstáculos (rho_0) igual a três vezes o raio do mesmo e a intensidade/amplitude dessa força foi fixada para ser o quadrado da área do polígono. Mais detalhes podem ser visto na Figura 14.

4 RESULTADOS E CONCLUSÕES

Os resultados apresentados nesta seção foram obtidos no cenário simulado, como na Figura 10. O objetivo proposto inicialmente era gerar um caminho (utilizando campos potenciais), fazendo com que o robô evitasse colidir com os obstáculos. A Figura 14 mostra o caminho planejado para o robô seguir e a trajetória feita pelo robô. Como observado, os dois caminhos possuem um pequeno percentual de erro porém, ao final da trajetória atingem o alvo requerido.

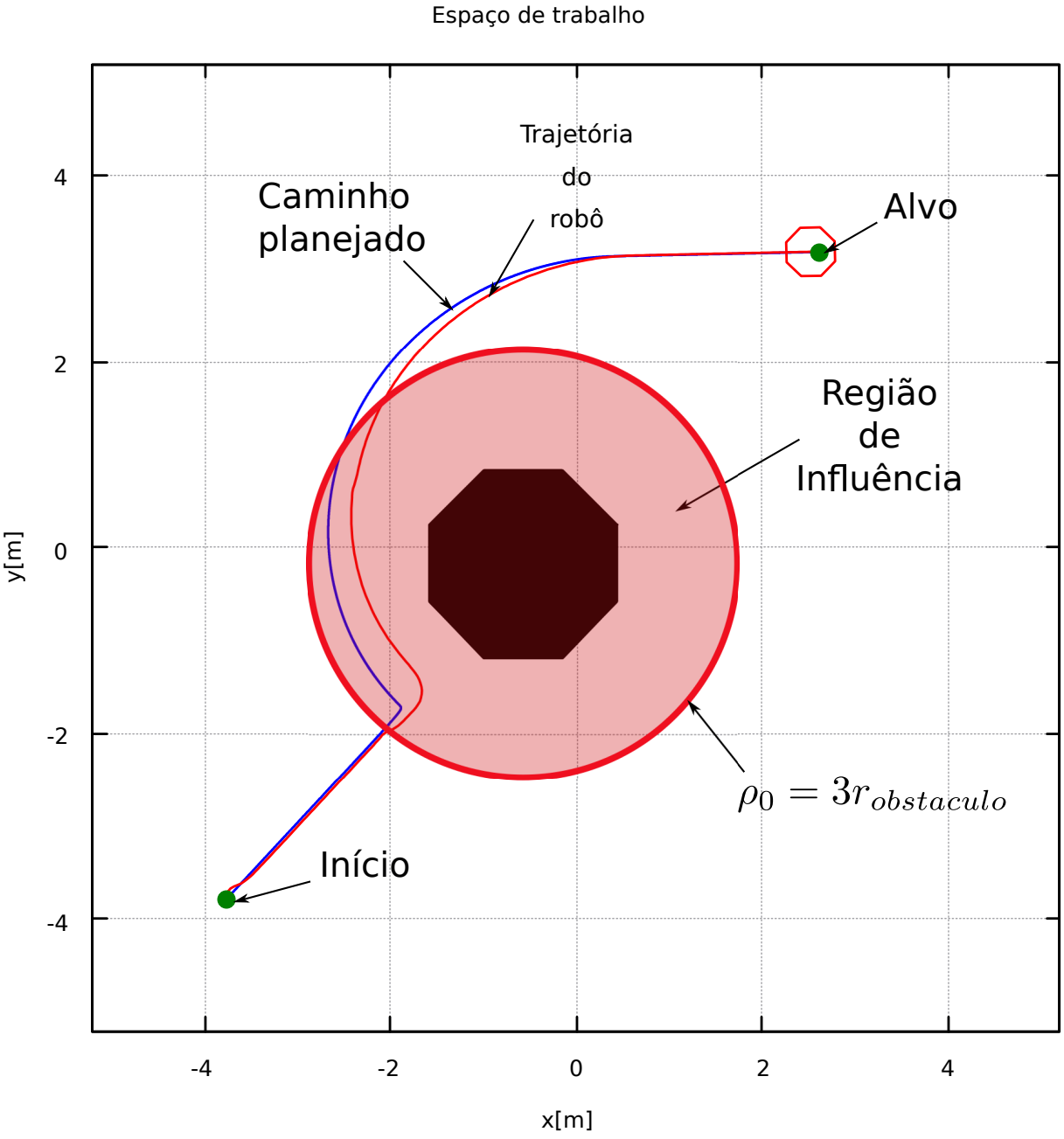

Figura 14 – Resultado com análises para o teste no cenário 1.

A Figura 15 mostra a geração de caminho no espaço de configuração.

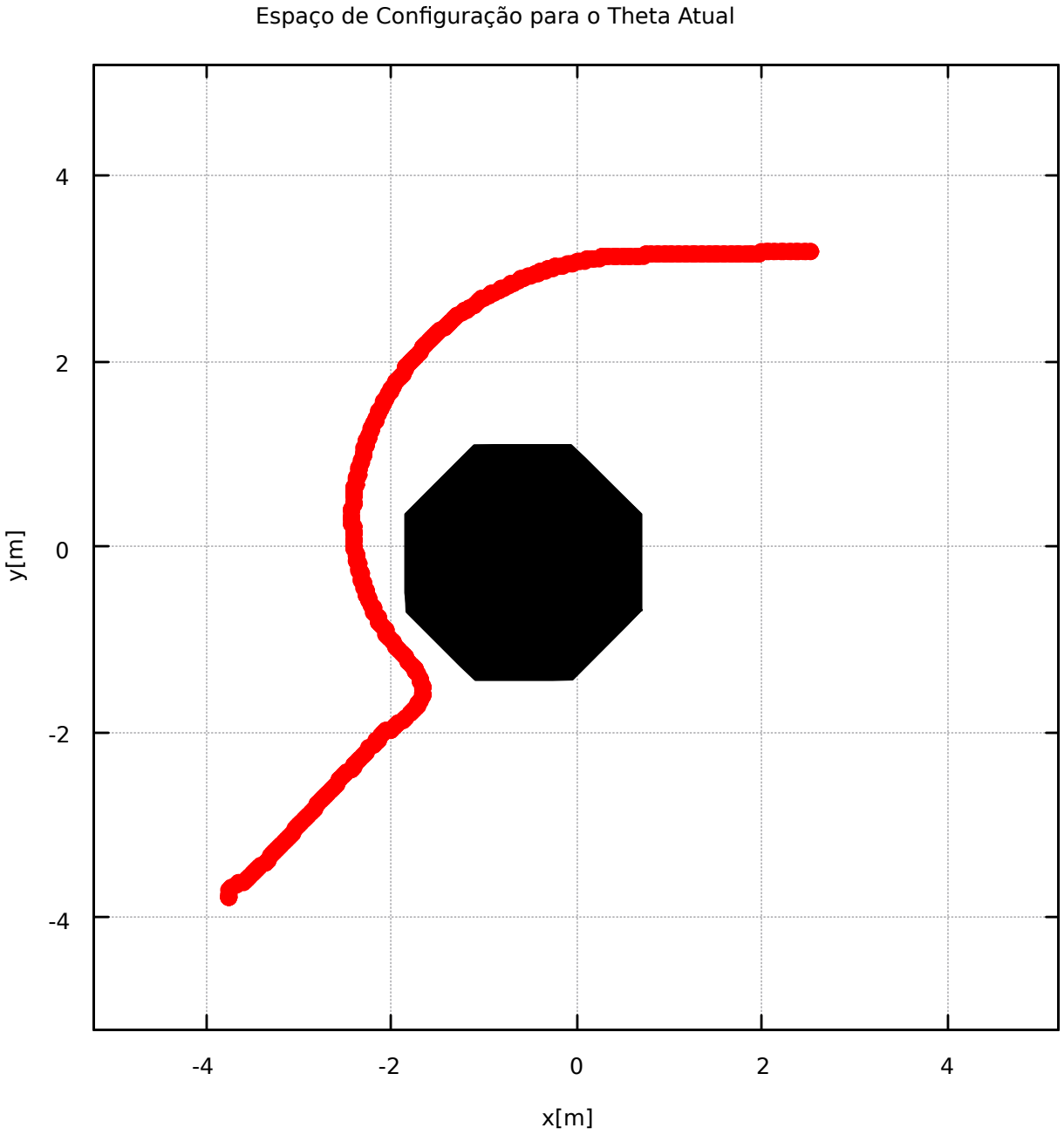

Figura 15 – Resultado para o teste no cenário 1. Visualização em espaço de configuração para o θ_{target}.

A Figura 16 mostra o robô percorrendo a trajetória. No entanto há a presença de um mínimo local, o qual faz o robô parar e permanecer no local até o início de um novo teste.

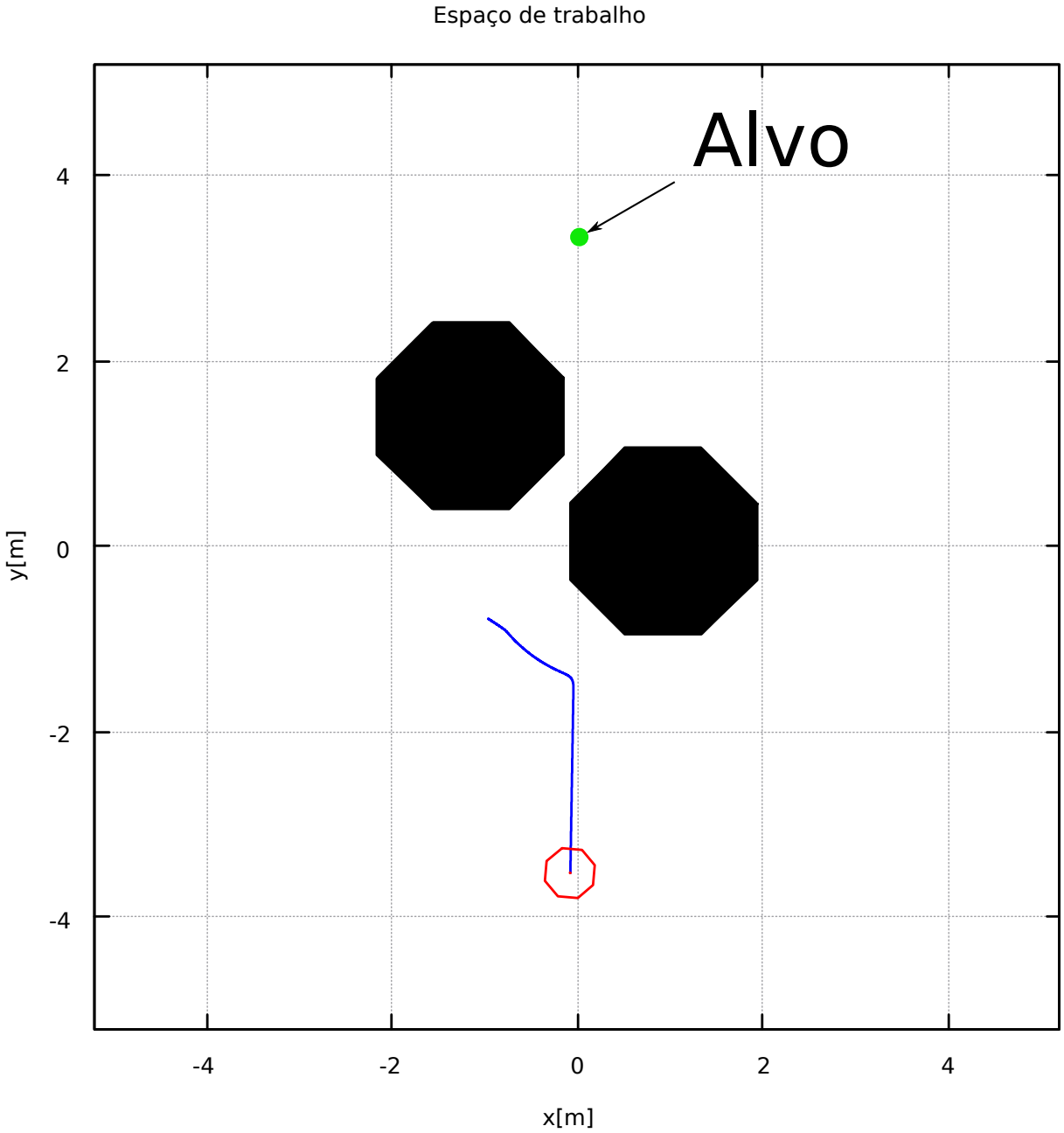

Figura 16 – Resultado para segundo teste. Situação de mínimo local.

A Figura 17 mostra a trajetória do robô desde seu ponto inicial até o final da trajetória (alvo), mostrando que o objetivo proposto foi alcançado no teste 3.

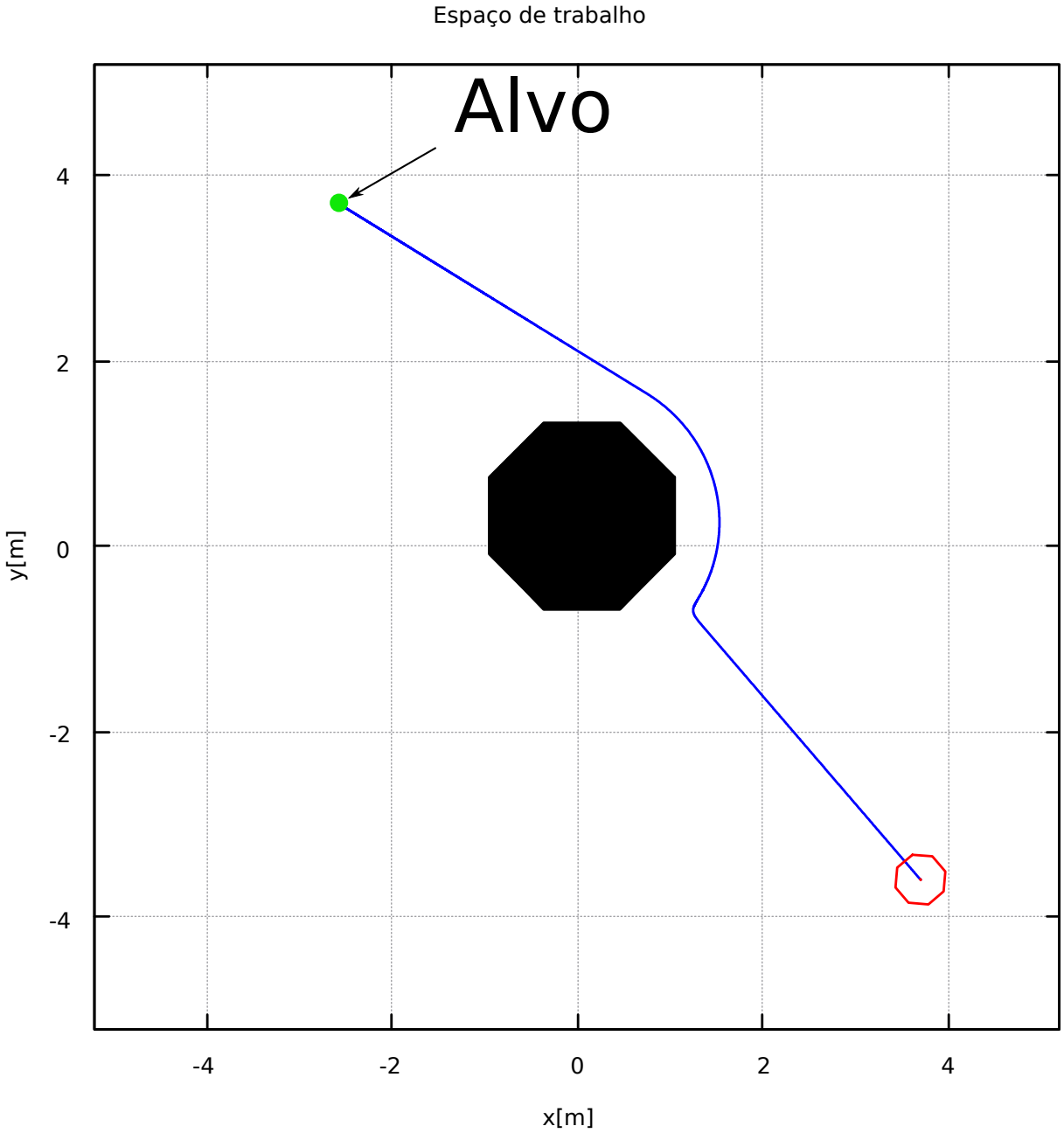

Figura 17 – Terceiro teste.

Pelos resultados apresentados acima, pode-se observar que a abordagem de planejamento de caminho por campos potenciais funcionou como o esperado, ou seja, o caminho gerado consegue ligar a configuração inicial com a final sem passar por uma região de obstáculo. Alguns problemas também são observáveis nos experimentos, um deles é a diferença notável entre o caminho percorrido pelo robô e o caminho planejado, isso pode ser corrigido ajustando-se os ganhos do controlador seguidor de caminho. Outro problema foi a falha do planejador em situação de mínimo local, muito embora não haja uma solução definida para esse problema, pode-se aplicar algumas técnicas para se contornar isso, como por exemplo, ao se identificar a situação de mínimo local (gradiente zero em uma configuração que não seja a configuração final): desabilitar o planejamento por campos potenciais e realizar um movimento diferente para sair dessa região de estagnação.

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, 12

3 WILLIAMS, T.; KELLEY, C.; many others. *Gnuplot 4.6: an interactive plotting program*. 2013. <http://gnuplot.sourceforge.net/>. 12

ANEXO 1

Código fonte: <https://github.com/Gabriellgpc/Sistemas_Roboticos/tree/master/program/p2m3>