Universidade Federal do Rio Grande do Norte

Programa de Pós-graduação em Engenharia Elétrica e de Computação

EEC1507 - Sistemas Robóticos Autônomos

RELATÓRIO

1º 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 1º 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

Conteúdo

1 INTRODUÇÃO

O controle cinemático de um robô pode apresentar problemas complexos de resolução. Alguns robôs são naturalmente instáveis e podem possuir restrições no que diz respeito à velocidade do robô. Neste caso, os controladores podem ser aplicados de acordo com um objetivo específico: controladores estabilizantes, seguidores de trajetória e seguidores de caminho. Para adentrar este último, foi implementado neste trabalho um gerador de caminho baseado em polinômios interpoladores de 3º grau, baseado no trabalho de [5] para robô móvel com acionamento diferencial e restrição não-holonômica. O trabalho fez uso da linguagem de programação Python [1] e o ambiente de simulação CoppeliaSim [2] para realizar o teste e demostração do funcionamento do gerador de caminho.

Um gerador de caminhos consiste em, através de uma posição inicial e final de um robô, descobrir uma sequência de movimentos a ser percorrido para que ele chegue a essas posições.

2 REFERENCIAL TEÓRICO

Uma polinômio de terceiro grau parametrizado com relação a λ possuí as seguintes equações:

$$x(\lambda) = a_0 + a_1\lambda + a_2\lambda^2 + a_3\lambda^3 \tag{1}$$
$$y(\lambda) = b_0 + b_1\lambda + b_2\lambda^2 + b_3\lambda^3 \tag{2}$$
$$\theta(\lambda) = \tan^{-1}\left(\frac{dy/d\lambda}{dx/d\lambda}\right) = \tan^{-1}\frac{b_1 + 2b_2\lambda + 3b_3\lambda^2}{a_1 + 2a_2\lambda + 3a_3\lambda^2} \tag{3}$$

λ varia de 0 à 1, do ponto inicial até o ponto final, ou seja $\lambda = 0$ para (x_i,y_i,θ_i) e $\lambda = 1$ para (x_f,y_f,θ_f).

Condições de contorno:

$$(x(0), y(0), \theta(0)) = (x_i, y_i, \theta_i)$$
$$(x(1), y(1), \theta(1)) = (x_f, y_f, \theta_f)$$

Das restrições acima temos que,

$$\begin{cases} x(0) & = a_0 \\ y(0) & = b_0 \\ \alpha(0) & = \tan\theta(0) = \frac{b_1}{a1} \\ x(1) & = a_0 + a_1 + a_2 + a_3 \\ y(1) & = b_0 + b_1 + b_2 + b_3 \\ \alpha(1) & = \tan\theta(1) = \frac{b_1+2b_2+3b_3}{a_1+2a_2+3a_3} \end{cases}$$

O sistema acima pode ser simplificado para,

$$\begin{cases} \alpha_i a_1 - b_1 = 0 \\ a_1 + a_2 + a_3 = \Delta x \\ b_1 + b_2 + b_3 = \Delta y \\ \alpha_f a_1 + 2\alpha_f a_2 + 3\alpha_f a_3 - b_1 - 2b_2 - 3b_3 = 0 \end{cases}$$

onde $\alpha_i = \tan\theta(0)$, $\alpha_f = \tan\theta(1)$, $\Delta x = x_f - x_i$ e $\Delta y = y_f - y_i$.

Reescrevendo estas equações para o formato matricial e realizando um pivotamento parcial, chega-se a:

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 2 & 3d_f & 2d_f & d_f \\ 0 & 0 & 1 & 0 & 0 & -d_i \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_3 \\ b_2 \\ b_1 \\ a_3 \\ a_2 \\ a_1 \end{bmatrix} = \begin{bmatrix} \Delta y \\ 3\Delta y \\ 0 \\ \Delta x \end{bmatrix}$$

Devido ao sistema possuir 8 incógnitas e apenas 6 restrições,sendo assim o sistema têm 2 variáveis que podem assumir qualquer valor. Escolhe-se as variáveis a_1 e a_2 como variáveis arbitrárias, pois o calculo dos demais a partir destes não resulta em divisões, o que pode causar singularidade.

Resolvendo o sistema com a_1 e a_2 arbitradas tem-se:

$$\begin{cases} b_3 &= 3\alpha_f\Delta x - 2\Delta y - \alpha_f a_2 - (2\alpha_f - \alpha_i)a_1 \\ b_2 &= 3\Delta y - 3\alpha_f\Delta x + \alpha_f a_2 - 2(\alpha_i - \alpha_f)a_1 \\ b_1 &= \alpha_i a_1 \\ a_3 &= \Delta x - a_2 - a_1 \end{cases} \tag{4}$$

As equações em 4 representam o caso geral para o polinômio de terceiro grau, embora respeite a restrição não-holonômica, possuí singularidades para $\theta = \pm\pi/2$, estes casos especiais são tratados como em [5], bem como as otimizações para escolher os coeficientes livres.

Para trabalhar com uma folga numérica em torno da singularidade define-se aqui uma variável δ para representar uma pequena variação.

2.1 Primeiro Caso Especial

Se $\theta_i \in [(\pi/2 - \delta), (\pi/2 + \delta)]$ e $\theta_f \in [(\pi/2 - \delta), (\pi/2 + \delta)]$

$$\begin{cases} b_1 &= \Delta y \text{ (coef. livre)} \\ b_2 &= 0 \text{ (coef. livre)} \\ a_0 &= x_i \\ a_1 &= 0 \\ a_2 &= 3\Delta x \\ a_3 &= -2\Delta x \\ b_0 &= y_i \\ b_3 &= \Delta y - b_1 - b_2 \end{cases} \tag{5}$$

2.2 Segundo Caso Especial

Apenas $\theta_i \in [(\pi/2 - \delta), (\pi/2 + \delta)]$

$$\begin{cases} a_3 &= -\Delta x/2 \text{ (coef. livre)} \\ b_3 &= \text{qualquer (coef. livre)} \\ a_0 &= x_i \\ a_1 &= 0 \\ a_2 &= \Delta x - a_3 \\ b_0 &= y_i \\ b_1 &= 2(\Delta y - \alpha_f \Delta x) - \alpha_f a_3 + b_3 \\ b_2 &= (2\alpha_f \Delta x - \Delta y) + \alpha_f a_3 - 2b_3 \end{cases} \tag{6}$$

2.3 Terceiro Caso Especial

Se $\theta_f \in [(\pi/2 - \delta), (\pi/2 + \delta)]$

$$\begin{cases} a_1 &= 3\Delta x/2 \text{ (coef. livre)} \\ b_2 &= \text{qualquer (coef. livre)} \\ a_0 &= x_i \\ a_2 &= 3\Delta x - 2a_1 \\ a_3 &= a_1 - 2\Delta x \\ b_0 &= y_i \\ b_1 &= \alpha_i a_1 \\ b_3 &= \Delta y - \alpha_i a_1 - b_2 \end{cases} \tag{7}$$

2.4 Caso Geral

$$\begin{cases} a_1 &= \Delta x \text{ (coef. livre)} \\ a_2 &= 0 \text{ (coef. livre)} \\ a_0 &= x_i \\ a_3 &= \Delta x - a_1 - a_2 \\ b_0 &= y_i \\ b_1 &= \alpha_i a_1 \\ b_2 &= 3(\Delta y - \alpha_f \Delta x) + 2(\alpha_f - \alpha_i) + \alpha_f a_2 \\ b_3 &= 3\alpha_f \Delta x - 2\Delta y - (2\alpha_f - \alpha_i) a_1 - \alpha_f a_2 \end{cases} \tag{8}$$

3 METODOLOGIA

A simulação foi realizada no ambiente *CoppeliaSim*[2] em configuração cliente/servidor, sendo o servidor responsável pela simulação da física em tempo real e o cliente em *Python*[1] responsável por calcular o polinômio de terceiro grau que servirá de caminho para o robô, a comunicação foi feita utilizando a biblioteca para Python disponibilizada pelo *CoppeliaSim* (Legacy remote API [2]), gráficos (gerados com o uso da biblioteca *matplotlib*[3]) também serão gerados e exibidos pelo cliente, em tempo de execução, um link para o código fonte bem como os demais recursos utilizados encontram-se em anexo 1.

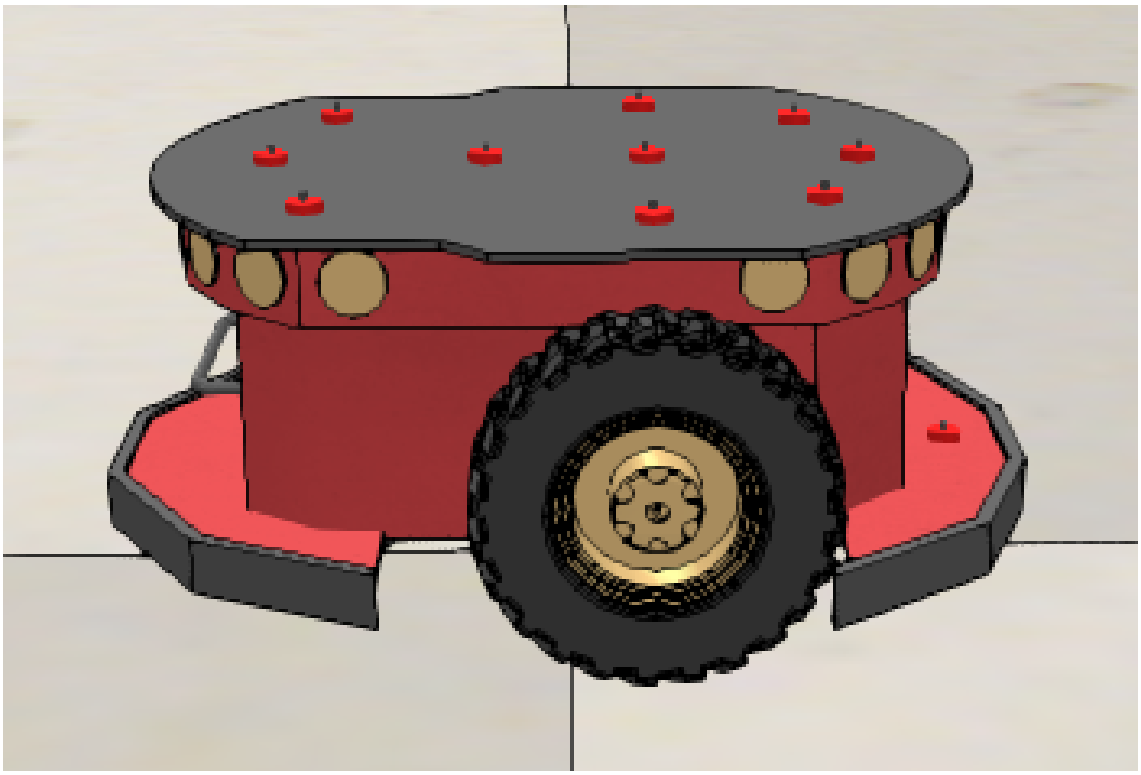

Figura 1: Modelo simulado do robô Pioneer.

O robô utilizado possui rodas com rádio igual a $0.09751m$ (r_w) e a distância entre os centros das mesmas é de $0.331m(b)$, podemos ver na figura 1 o modelo do robô no ambiente simulado.

O cenário criado no ambiente é bem simples, contem apenas 2 objetos que não vêm no cenário padrão do simulador, o robô e um objeto circular na cor azul, o robô embora não seja acionado ele está servindo como configuração

inicial (posição e orientação) e a marcação em azul é o ponto final, as configurações destes dois objetos no mundo servirá como entrada para o gerador de caminho que será executado em Python, o programa cliente coleta a informação de posição e orientação destes dois objetos e utiliza como entrada para a função que calculará os coeficientes da curva. A figura 2 mostra o cenário descrito acima.

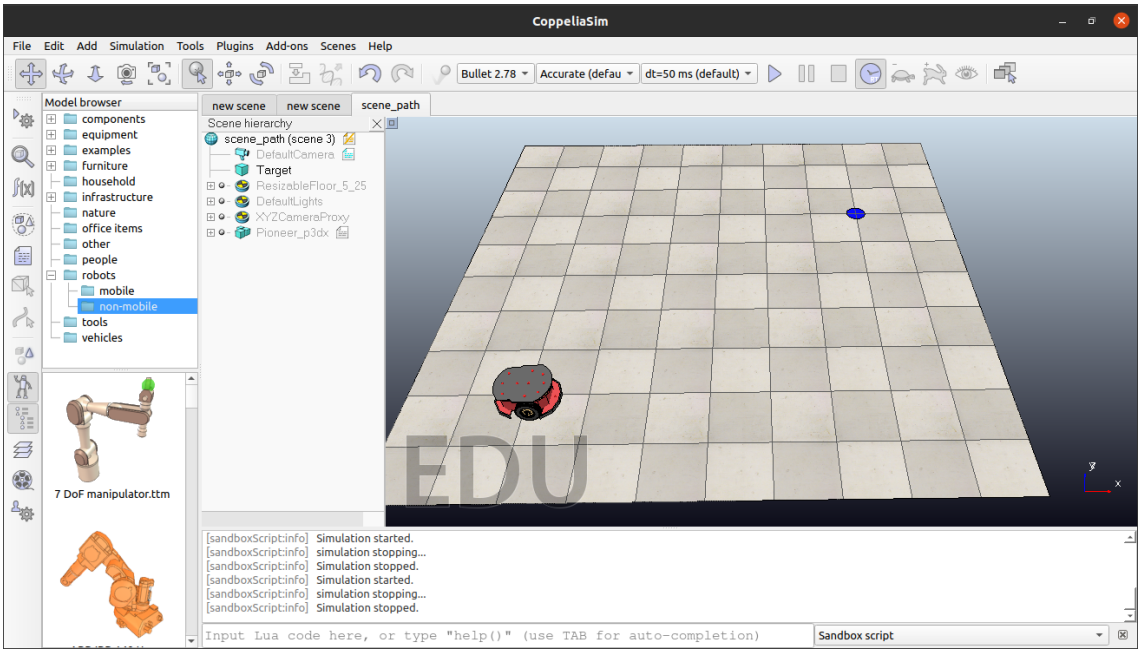

Figura 2: Cenário no simulador CoppeliaSim.

O programa cliente contem a implementação para calcular os coeficientes do polinômio cúbico, como explicado na seção de referencial teórico, uma vez conectado com o simulador (por meio da biblioteca remoteAPI legacy) o script Python obtém as configurações de posição e orientação inicial e finais, por meio da leitura destas configurações provenientes do simulador, computa o caminho e envia para o simulador os pontos do caminho para ser possível visualizar o mesmo no ambiente de simulação (isto será explicado com mais detalhes adiante), por fim o script mostra um gráfico do caminho gerado, como será visto na seção de resultados.

O simulador contem um script simples, do tipo childThread (que está associado com o objeto câmera padrão, mas independe disso) isso é uma rotina em Lua [6] que fica responsável por ler sinais do tipo string (por meio da função getStringSignal [2]) que é por onde o cliente enviará os pontos pertencentes ao caminho calculado e que deve ser exibido no simulador. Uma

vez que estes pontos chegam ao simulador o mesmo é desenhado em tempo real com as funções addDrawingObject e addDrawingObjectItem [2].

4 RESULTADOS E CONCLUSÕES

Aqui podemos visualizar os resultados para uma simulação com o robô na configuração ($x \approx -1.4$,$y \approx -1.4$,$\theta = 0$) e o target(destino final) na configuração ($x = 2.5$,$y = 2.5$,$\theta = 0$), ou seja as configurações inicial e final respectivamente. Nessa configuração o algoritmo resultou nos seguintes parâmetros para o polinômio de terceiro grau que interpola estas configurações:

$$\begin{cases} a_1 = 2.97 \\ a_2 = 0 \\ a_0 = -1.47 \\ a_3 = 0.0 \\ b_0 = -1.57 \\ b_1 = 0.0 \\ b_2 = 7.72 \\ b_3 = -5.15 \end{cases}$$

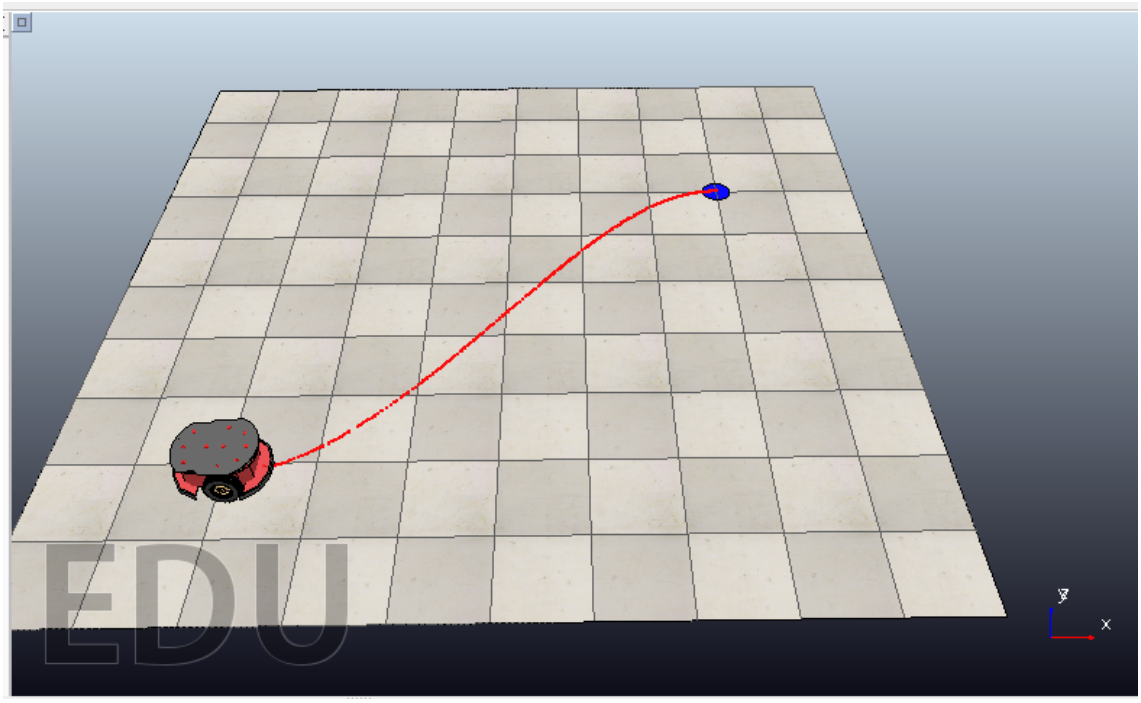

Figura 3: Caminho gerado e inserido no ambiente de simulação.

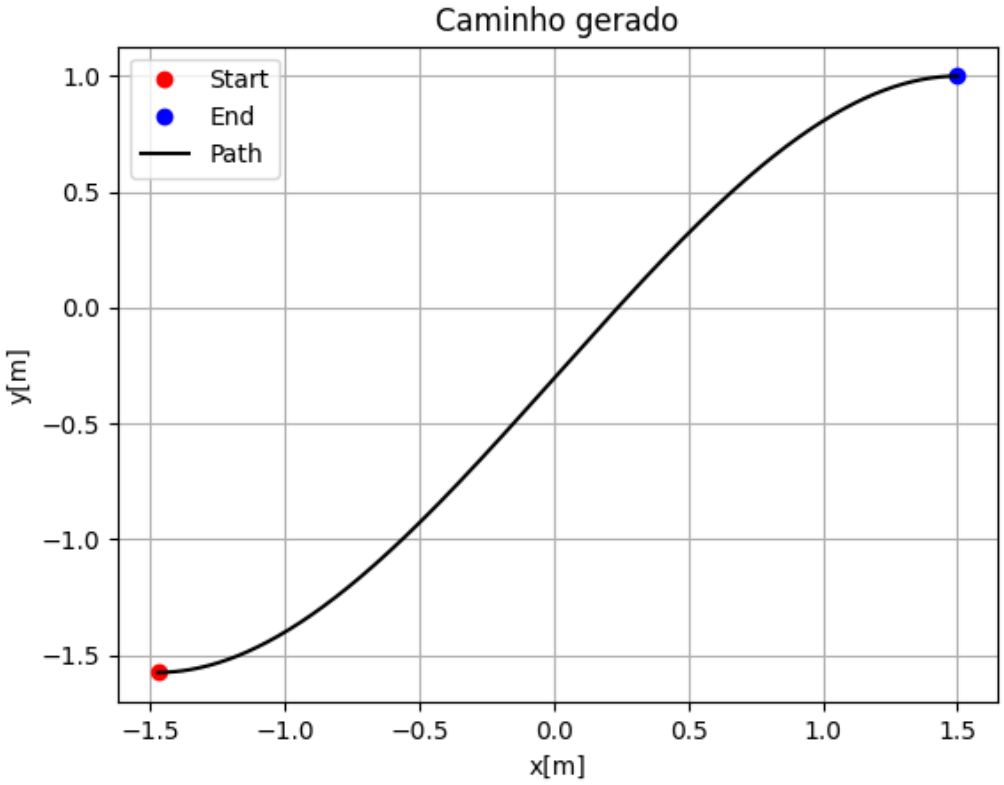

Figura 4: Gráfico do caminho gerado.

Na figura 3 podemos ver o caminho desenhado no simulador, ele parte do Pioneer ($\lambda = 0$), na configuração do mesmo e vai até o target em azul ($\lambda = 1$). Também é possivel visualiza-lo na saída do script em Python, ver figura 4, onde o ponto em vermelho representa a configuração inicial (robô), o ponto em azul o final (circulo azul) e em preto temos o polinômio de terceiro grau ligando ambos.

5 REFERÊNCIAS

Referências

[1] Python Org. Disponível em: <https://www.python.org/. Acesso em: 22 set. 2020.

[2] E. Rohmer, S. P. N. Singh, M. Freese, "CoppeliaSim (formerly V-REP): a Versatile and Scalable Robot Simulation Framework", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2013. www.coppeliarobotics.com

[3] Matplotlib: Visualization with Python. Disponível em: <https://matplotlib.org/. Acesso em: 22 set. 2020.

[4] The NumPy Org. Disponível em: <https://numpy.org/. Acesso em: 22 set. 2020.

[5] PEDROSA, Diogo Pinheiro Fernandes. Sistema de navegação para robôs móveis autônomos. 2001. 85 f. Dissertação (Mestrado em Automação e Sistemas; Engenharia de Computação; Telecomunicações) - Universidade Federal do Rio Grande do Norte, Natal, 2001. Disponível em: https://repositorio.ufrn.br/handle/123456789/15417

[6] The programming language Lua. Disponível em: https://www.lua.org/. Acesso em: 29 set. 2020.

6 ANEXO 1

Código fonte: `https://github.com/Gabriellgpc/Sistemas_Roboticos_Autonomos_pos.git`

Vídeo de demostração: `https://www.youtube.com/watch?v=RMSFS0rfCHw`