Universidade Federal do Rio Grande do Norte

EGM0007 - SISTEMAS ROBÓTICOS AUTONOMOS

Seguidor de Caminho [SAMSON]

Técnicas de Controle Cinemático de Robôs Móveis

Discente:

Kaike Castro

Docente:
Pablo Javier

Natal - RN
2020

1 Introdução

Robôs Móveis são sistemas de Muitas Entradas e Muitas Saídas (MIMO), não lineares e, às vezes, subatuados como por exemplo os quadricópteros. Alguns robôs móveis possuem restrições não holonômicas adicionais como aqueles que possuem acionamento diferencial e outros são robôs móveis são naturalmente instáveis como os helicópteros autônomos. O problema de controle de robôs móveis não é trivial, portanto são necessárias técnicas de controle para controladores dinâmicos e cinemáticos.

2 Objetivo

Implementar um controlador do tipo Seguidor de caminho [Samson] para seguir um polinômio de terceiro grau.

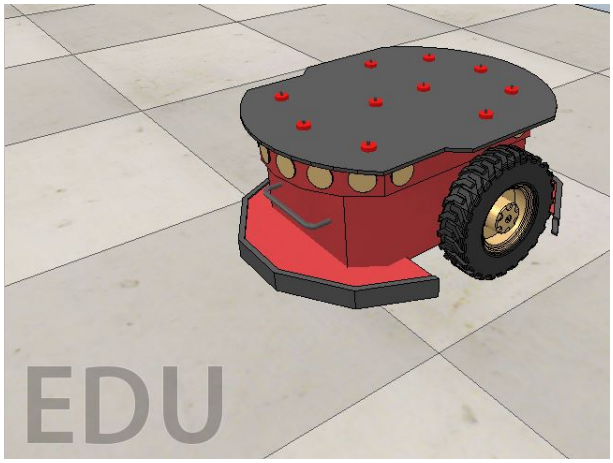
3 Metódos

O desenvolvimento do projeto seguiu os passos estabelecido na descrição do robô em relação ao caminho e a simulação foi realizada no V-REP. Segundo o artigo [1], o seguidor tem referencial Serret-Frenet que consiste de atribuir a cada ponto de uma curva espacial uma base ortonormal positiva no R3 que forneça propriedades geométricas da curva. O referencial S-F se move ao longo do caminho em que a posição do robô tem que convergir a um alvo virtual minimizando os erros até chegar ao ponto final do percurso.

3.1 Robô Móvel

O robô utilizado foi o Pioneer P3-DX, figura 1, esse possui duas rodas tipo padrão fixa com acionamento diferencial e uma roda castor. Os parâmetros a seguir descreve o modelo do pioneer.

Figura 1: Robô Pioneer



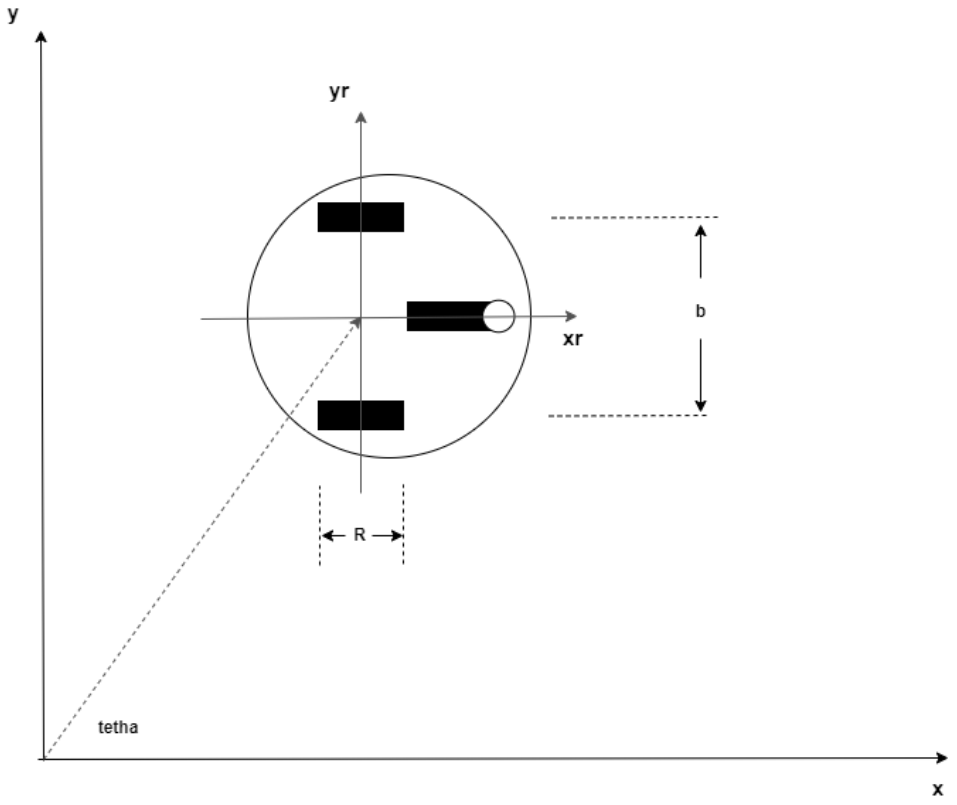
A figura 2 mostra as referências para o modelo cinemático em que b é a distância entre as rodas e R o raio das rodas padrão fixas.

- **Modelo Cinemático:**

$$\begin{bmatrix} x' \\ y' \\ \Theta' \end{bmatrix} = \begin{bmatrix} \frac{cos(\Theta).(r_d.w_d + r_e.w_e)}{2} \\ \frac{sin(\Theta).(r_d.w_d + r_e.w_e)}{2} \\ \frac{(r_d.w_d - r_e.w_e)}{b} \end{bmatrix} \tag{1}$$

- $\mathbf{b} = 0.1655$
- $\mathbf{R} = 0.1955/2$

Figura 2: Referência do robô



3.2 Geração de caminho por polinômio cúbico

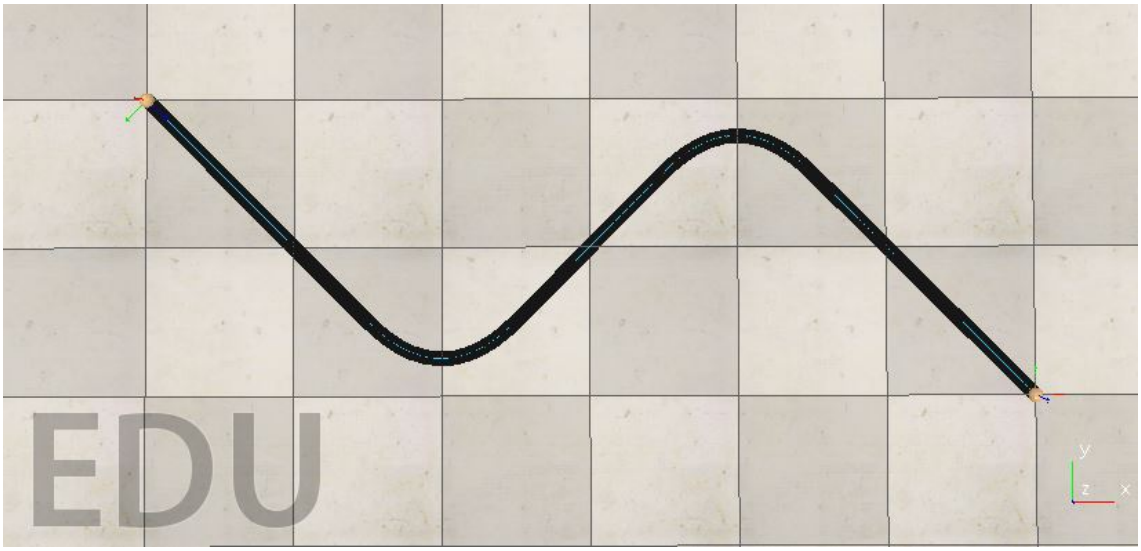
O caminho foi gerado a partir de quatro pontos no espaço de trabalho e feito uma interpolação polinomial entre eles. Desta forma, a equação (2) é resultado do polinômio parametrizado em s.
A equação (3) é o polinômio p(s) que foi usado para os cálculos.

$$\begin{cases} x = s \\ y = -\frac{2}{3}s^3 - s^2 + \frac{2}{3}s + \frac{1}{2} \end{cases} \tag{2}$$

$$p(s) = y = -\frac{2}{3}s^3 - s^2 + \frac{2}{3}s + \frac{1}{2} \tag{3}$$

A figura 3 mostra o caminho p(s) em que o robô seguirá no espaço de trabalho.

Figura 3: Curva polinomial de 3º grau



3.3 Controlador cinemático seguidor de caminho

O robô vai de encontro ao caminho p(s) procurando o ponto da curva mais próximo em relação ao seu sistema de coordenadas, e para isso a equação (4) foi a expressão utilizada para o minimizar o cálculo da distância.

A sua orientação para seguir a curva precisa se ajustada de acordo com a equação 5 que é a variação do ângulo tetha do robô menos o ângulo formado pelo vetor tangente a curva centrado no ponto mais próximo. A figura 4 mostrar as relações de posição e ângulo que o robô tem que seguir.

O vetor tangete a curva é calculado de acordo com a equação (6) em que a derivada da curva p(s) é calculada e depois retidada o módulo para assim efetuar a divisão para ter um vetor unitário tangente a curva. A curvatura de cada ponto próximo ao robô é obtida de acordo com a equação (7), sendo avaliada no próximo mais perto do pioneer tem-se um número menor que 1 isso para sastifazer o controle de Samson e evitar singularidade.

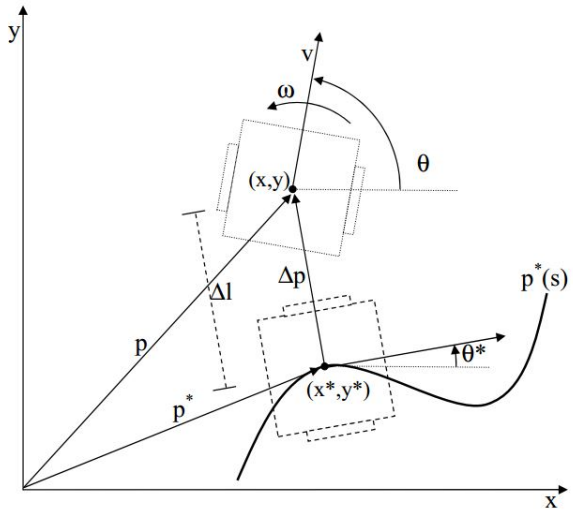
$$\Delta l = [(x - x^*)^2 + (y - y^*)^2]^{\frac{1}{2}} \tag{4}$$

$$\Delta\Theta = \Theta - \Theta^* \tag{5}$$

$$T(s) = \frac{p'(s)}{|p'(s)|} \tag{6}$$

$$k(s) = \frac{T'(s)}{p'(s)} \tag{7}$$

Figura 4: Referencial Serret-Frenet



- **Lei de controle cinemático de Samson:**

A seguir na equação (8) que pode ser encontrada no artigo [2] Controle de Sistemas encadeados - aplicação para seguir caminhos e variar no tempo estabilização pontual de robôs móveis. Primeiro se estabelece na lei de controle uma velocidade constante e depois criada um variável auxiliar u na equação (9) para servir de entrada na equação (10) cuja a saída é a velocidade angular w para o robô. Desta forma, as velocidades angulares nas rodas são obtidas de acordo com os passos das equações (11) e (12).

$$v = constate \tag{8}$$

$$u = -(k_{\Theta}.\Delta\Theta + \frac{k_l.\Delta l.v.sin(\Delta\Theta)}{\Delta\Theta}), k_l.k_{\Theta} > 0 \tag{9}$$

$$w = u + \frac{k(s).v.cos(\Delta\Theta)}{1 - k(s).\Delta l} \tag{10}$$

$$\begin{cases} v_d = v + \frac{b}{2}.w \\ v_e = v - \frac{b}{2}.w \end{cases} \tag{11}$$

$$\begin{cases} w_d = \frac{v_d}{R} \\ w_e = \frac{v_e}{R} \end{cases} \tag{12}$$

4 Resultados

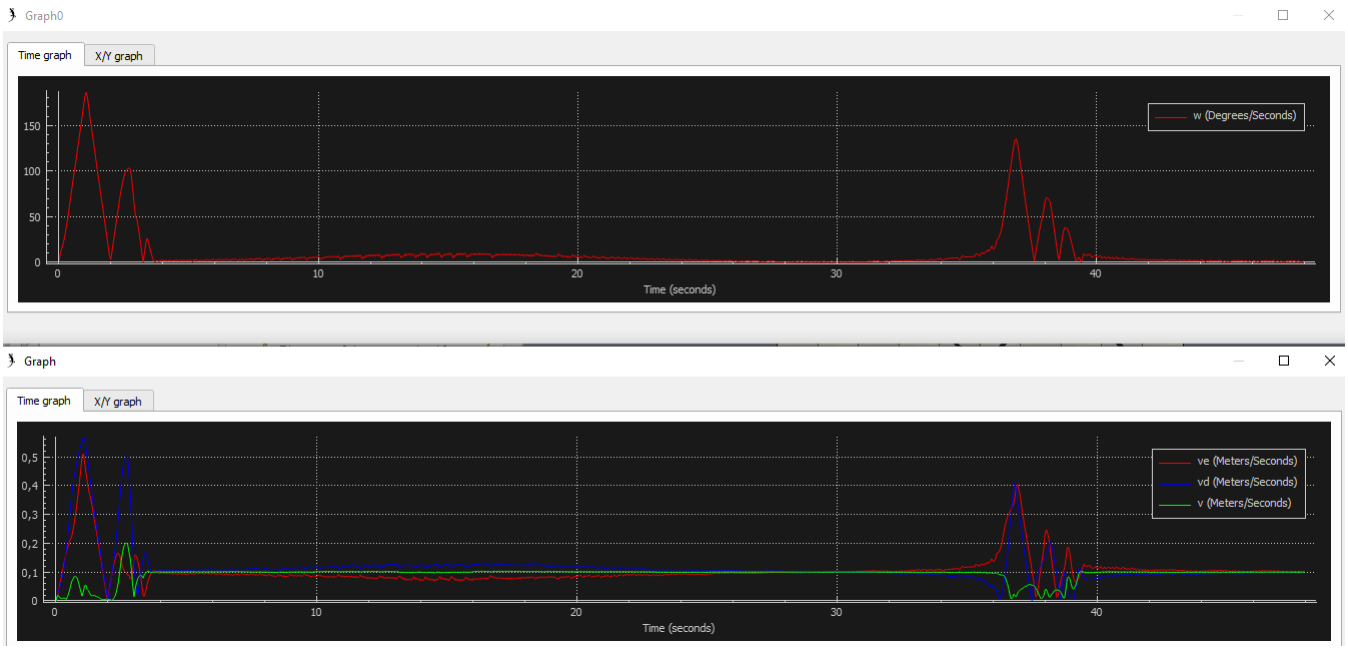
O controlador de caminho de Samson foi implementado atendendo os seus requisitos. Houve um problema o qual não foi possível entender que a curva que o robô seguiu estava defasada e ampliada significamente em relação a curva p(s). Entretanto, o robô conseguir percorrer o caminho de um polinomio de terceiro grau.

Na equação (9) do controle de Samson teve que ajustar duas constantes kl e kTheta e foi encontrado os valores 0.1 e 0.3, respectivamente que melhor atendeu o controle. Porém, é possível achar melhores valores.

A figura 5 mostra no primeiro gráfico a velocidade angular do robô sendo possível ver os momentos que as oscilações do controle precisariam de mais ajustes sobretudo as constantes kl e kTheta.

As velocidades lineares das rodas e do robô estão no segundo gráfico da figura 5,a curva em vermelho representa a velocidade na roda esquerda, azul na roda direita e em verde a velocidade do robô.

Figura 5: Referencial Serret-Frenet



5 Conclusão

O desenvolvimento do trabalho buscou-se seguir os passos para o controle de Samson e as orientações para elaborar o projeto.

Referências

[1] Conrado Damato de Lacerda. “O Referencial de Frenet-Serret e Grupos de Lie”. Em: ().

[2] Claude Samson. “Control of chained systems application to path following and time-varying point-stabilization of mobile robots”. Em: *IEEE transactions on Automatic Control* 40.1 (1995), pp. 64–77.

6 Anexo

```
function sysCall_init() -- Funcao de Inicializacao

    local vref = 0.0
    motorLeft=sim.getObjectHandle("Pioneer_p3dx_leftMotor")
    motorRight=sim.getObjectHandle("Pioneer_p3dx_rightMotor")
    path = sim.getObjectHandle('Path')
    sim.setPathTargetNominalVelocity(path,vref)
    ref_point=sim.getObjectHandle('ref_point')
    ref_coord=sim.getObjectHandle('ref_coord')
    ref_end = sim.getObjectHandle('end')
    robot_pose=sim.getObjectHandle('robot_pose')

    pose = updateRobotPose()
    refP = updateRefPose()

end

function updateRobotPose()

    local pose
    position=sim.getObjectPosition(robot_pose,-1)
    orientation=sim.getObjectOrientation(robot_pose,-1)
    pose={position[1],position[2],orientation[3]}
    return pose

end

function curvatura(c) -- Calcula curvatura de um ponto da curva p(s)

rls = (-(2*c*c)-(2*c) + 2/3) -- derivada no ponto c

local msT = math.sqrt((rls*rls)+1)

local T = {1 / msT ,rls / msT}
moduleT = math.sqrt((T[1]*T[1]) + (T[2]*T[2]))

k = math.abs(moduleT / msT)

return k
end

function controleSamson(v, ks, deltaTetha,dL) -- Controle de Samson

kT = 0.3 -- constante angular
kL = 0.1 -- 0.3 -- constante L
```

```
u = -(kT*deltaTetha + (kL*dL * v * math.sin(deltaTetha))/deltaTetha)
w = u + (ks*v*math.cos(deltaTetha)) / (1-ks*dL)

return w

end

function updateRefPose() -- Retorna as x,y e tetha do robo em relacao ao referencial

    local refP
    reFposition=sim.getObjectPosition(ref_point,-1)
    reForientation=sim.getObjectOrientation(ref_point,-1)
    refP={reFposition[1],reFposition[2],reForientation[3]}
    return refP

end

function getDistance(a,b) -- Calcula a distancia entre dois pontos

    local x,y = (a[1]- 0.0445) - b[1], a[2] - b[2]
    return math.sqrt(x*x+y*y)

end

function deltaTetha() -- Calcula a variação de Theta

p = updateRobotPose()
local a = distanceMin() -- x,y, deltaL

local yT = -(2*a[1]*a[1])-(2*a[1]) + 2/3

local mT = math.sqrt((yT*yT)+1)

local T = {1 / mT ,yT / mT}
moduleT = math.sqrt((T[1]*T[1]) + (T[2]*T[2]))

deltaP= {p[1] - a[1],p[2] - a[2]}

tethaP = math.deg(math.atan(T[2]/T[1]))
tethaR = math.deg(p[3])
deltaT = tethaR - tethaP
return deltaT

end


function distanceMin() -- Calcula a menor distancia entre o robo e o ponto mais próximo
```

```
local deltaL = 10000
local d = 0

while d < deltaL do
for s=-5,3,0.005 do

xp = s
yp = ((-2/3)*s*s*s)-(s*s) + (2/3)*s + (1/2)

d = getDistance({xp,yp}, updateRobotPose())

if d < deltaL then

 deltaL = d
 x = xp
 y = yp
end
end

end
 return {x,y,deltaL}

end
function sysCall_actuation() -- Função Loop

      v = 0.1
      b = 0.1655
      R = 0.1955/2 -- raio das rodas
      rd = R
      re = R

     local pos = distanceMin()  -- x, y, deltaL
     sim.setObjectPosition(ref_coord,-1,{pos[1],pos[2],0})

    W = controleSamson(v, curvatura(pos[1]), deltaTetha(),pos[3]) -- v, ks, deltaTetha, del

    vd = v + b/2 * W
    ve = v - b/2 * W

    wd = vd / rd -- rd raio da roda direita
    we = ve / re -- re raio da roda esquerda

    sim.setJointTargetVelocity(motorLeft,we)
    sim.setJointTargetVelocity(motorRight,wd)
end
```