

José Iuri Barbosa de Brito, Lucas Tavares

# **Seguidor de Trajetória Arbitrária para Robôs Monociclo**

Campina Grande

23 de março de 2018



# **Lista de ilustrações**

Figura 1 – Diagrama de um robô do tipo monociclo. . . . .	5
Figura 2 – Diagrama de um robô do tipo monociclo seguindo uma trajetória. . . . .	6
Figura 3 – Posse de um robô monociclo (KANAYAMA et al., 1990). . . . .	8
Figura 4 – Dimensões do robô <i>Pioneer 3-DX</i> . . . . .	10
Figura 5 – Ambiente de Simulação . . . . .	25

# Lista de tabelas

# Sumário

<b>1</b>	<b>INTRODUÇÃO . . . . .</b>	<b>5</b>
<b>1.1</b>	<b>Fundamentação teórica . . . . .</b>	<b>5</b>
1.1.1	Cinemática de Robôs Móveis . . . . .	5
1.1.2	Seguidor de trajetória . . . . .	6
1.1.3	Seguidor de trajetória proporcional . . . . .	8
1.1.4	<i>Pioneer 3-DX</i> . . . . .	9
<b>1.2</b>	<b>Objetivos . . . . .</b>	<b>9</b>
<b>2</b>	<b>REALIZAÇÃO DO EXPERIMENTO . . . . .</b>	<b>11</b>
<b>2.1</b>	<b>Material Utilizado . . . . .</b>	<b>11</b>
<b>2.2</b>	<b>Procedimento Experimental . . . . .</b>	<b>11</b>
<b>3</b>	<b>RESULTADOS . . . . .</b>	<b>13</b>
<b>4</b>	<b>CONCLUSÃO . . . . .</b>	<b>17</b>
	<b>REFERÊNCIAS . . . . .</b>	<b>27</b>



# 1 Introdução

Este trabalho trata-se do relatório final da disciplina "Tópicos Especiais em Engenharia Elétrica Robótica, Teoria, Tecnologia e Programação" que tratará da aplicação de um algorítimo de controle em simulação de um robô móvel do tipo monociclo movendo-se no espaço cartesiano de duas dimensões que será capaz de seguir uma trajetória arbitrariamente definida pelo usuário com uma pequena taxa de erro.

## 1.1 Fundamentação teórica

### 1.1.1 Cinemática de Robôs Móveis

O robô utilizado neste trabalho possui a configuração representada pela Figura 1, composto basicamente de um par de rodas, cada um sendo rotacionada por um único motor impondo torque sobre a roda e uma roda boba ou castor que dá suporte a estrutura para que não caia e ao mesmo tempo permita a movimentação livre do veículo.

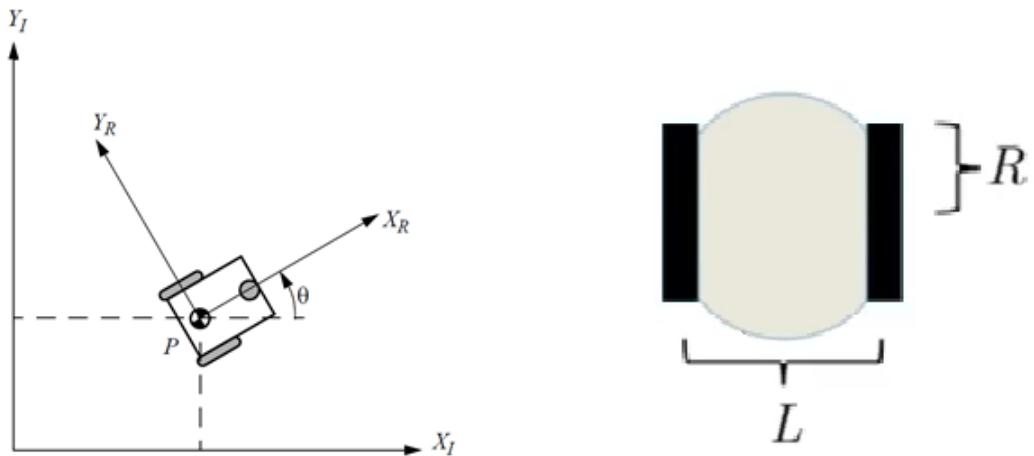


Figura 1 – Diagrama de um robô do tipo monociclo.

Com esse modelo pode-se simplificar as equações de cinemática com base na sua velocidade linear e angular da seguinte maneira:

$$\begin{aligned}\dot{x} &= V \cos(\theta) \\ \dot{y} &= V \sin(\theta) \\ \dot{\theta} &= \omega\end{aligned}\tag{1.1}$$

Onde  $V$  e  $\omega$  são a velocidade linear e angular, respectivamente.

Do ponto de vista real a utilização de robôs em conjunto com atuadores requerem uma velocidade de atuação em cada dispositivo, logo pode-se também escrever a pose do robô monociclo na equação 1.4 tendo como entrada as velocidades de atuação nas rodas. Como demonstrado abaixo:

$$\begin{aligned}\dot{x} &= \frac{R}{2}(v_r + v_l)\cos(\theta) \\ \dot{y} &= \frac{R}{2}(v_r + v_l)\sin(\theta) \\ \dot{\theta} &= \frac{R}{L}(v_r - v_l)\end{aligned}\quad (1.2)$$

Dessa forma é possível atuar no robô diretamente nos motores que movem as rodas.

De 1.4 em 1.2 tem-se as seguintes expressões:

$$\begin{aligned}v_r &= \frac{2V + \omega L}{2R} \\ v_l &= \frac{2V - \omega L}{2R}\end{aligned}\quad (1.3)$$

### 1.1.2 Seguidor de trajetória

O seguidor de trajetória utilizado nesse experimento foi proposto por (MORRO; SGORBISSA; ZACCARIA, 2011) e apresenta vantagens em relação aos seguidores de trajetória desenvolvidos anteriormente, já que propõe uma nova abordagem baseada na equação implícita da curva,  $f(x, y) = 0$ , e que garante a estabilidade assíntotica para qualquer configuração inicial do robô. Primeiramente devemos considerar o diagrama do modelo monociclo como definido abaixo.

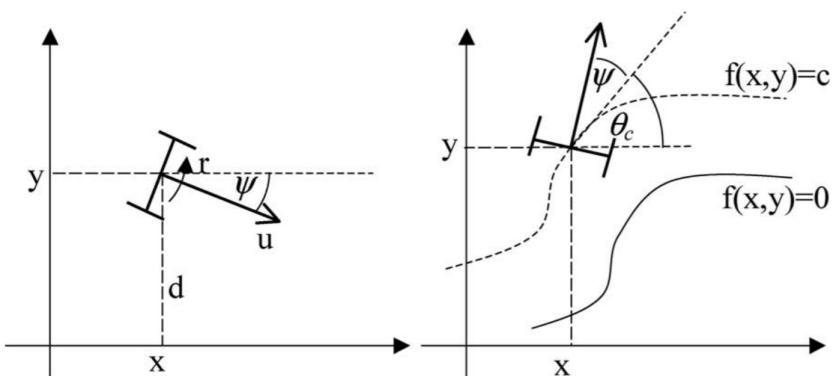


Figura 2 – Diagrama de um robô do tipo monociclo seguindo uma trajetória.

Considerando as equações 1.4 dispostas da seguinte forma:

$$\begin{aligned}\dot{x} &= u \cos(\theta) \\ \dot{y} &= u \sin(\theta) \\ \dot{\theta} &= r\end{aligned}\tag{1.4}$$

Pode-se considerar a curva  $f(x, y) = z$  como sendo a trajetória desejada, e considerando o plano  $z = 0$  temos que  $f(x, y) = 0$  fornece geometricamente a curva de superfície em que há a trajetória a ser seguida. Logo tomando as propriedades dessa equação implícita tem-se as seguintes acertivas.

1.  $f$  é diferenciável e  $\|\nabla f\|^2 = f_x^2 + f_y^2 > 0$  em  $\mathbb{R}^2$ ;
2.  $\lim_{t \rightarrow \infty} u \neq 0$ ;
3.  $f$  é duplamente diferenciável e  $f_x, f_y, f_{xx}, f_{xy}$  e  $f_{yy}$  são limitadas em qualquer domínio fechado  $D \subset \mathbb{R}^2$  onde  $f$  é limitado;
4.  $u$  e  $\dot{u}$  são limitados.

Seja  $f_x, f_y, f_{xx}, f_{xy}$  e  $f_{yy}$  as derivadas parciais de  $f(x, y)$  de primeira e segunda ordem. E considerarmos  $\psi$  o ângulo entre a inclinação do veículo e a inclinação desejada e o vetor  $(f_y, -f_x)$  é a tangente da curva de nível como demonstrado na Figura 2. Para atender os quesitos 1) à 4) tem-se que segundo (MORRO; SGORBISSA; ZACCARIA, 2011):

$$\begin{aligned}u &= u(t) \\ r &= K_1(-\|\nabla f\|uS(f) - f_x|u|\cos\theta - f_y|u|\sin\theta) + \dot{\theta}_c\end{aligned}\tag{1.5}$$

Onde  $K_1 > 0$  e  $S(f)$  é a  $C^n$  função sigmoide:

$$S(f) = \frac{K_2 f}{\sqrt{1 + f^2}} \quad 0 > K_2 > 1\tag{1.6}$$

O termo  $\theta_c = \theta - \psi$  é definido como o ângulo entre o eixo x e o vetor  $(f_y, -f_x)$  normal a  $\|\nabla f\|$  em  $(x, y)$ . Logo pode-se definir o ângulo como sendo:

$$\theta_c = -\cot^{-1} \frac{f_y}{f_x}\tag{1.7}$$

Daí tem-se que:

$$\dot{\theta}_c = \frac{(f_x f_{xy} - f_y f_{xx})u \cos\theta + (f_x f_{xy} - f_y f_{xx})u \sin\theta}{\|\nabla f\|^2}\tag{1.8}$$

Observa-se que as equações 1.5 podem ser escritas como equações diferenciais ordinárias com solução analítica e dependente das condições iniciais:

$$\begin{aligned}\dot{x} &= u \cos(\theta) \\ \dot{y} &= u \sin(\theta) \\ \dot{\theta} &= r\end{aligned}\tag{1.9}$$

Ao resolver as equações 1.9 tem-se o conjunto de pontos da pose da trajetória do robô onde ele deverá seguir a função de referência  $f(x, y)$ .

### 1.1.3 Seguidor de trajetória proporcional

Um dos problemas enfrentados por robôs móveis com rodas é o fenômeno do escorregamento que é o fato do coeficiente de atrito estático não ser alto suficiente para que ocorra o impulso do chão nas rodas e empurre o veículo na direção desejada. Devido a esse fenômeno a utilização de uma trajetória de referência fica comprometida, já que ao longo do tempo a referência não será mais seguida e o torque aplicado nos motores do robô não será suficiente para gerar o movimento. Para contornar essa situação em situações de escorregamento é necessário a utilização de um controlador proporcional para compensar o efeito indesejado.

Uma das maneiras mais simples de utilizar um controlador proporcional foi proposta por (KANAYAMA et al., 1990) e propõe a utilização de um controlador com realimentação de erro como segue.

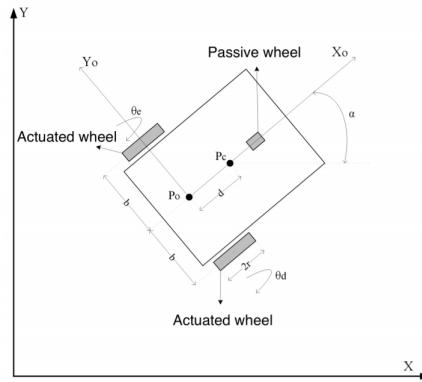


Figura 3 – Posse de um robô monociclo (KANAYAMA et al., 1990).

Onde as poses atual e de referência dos robôs podem ser definidas como

$$q_r = [x_c \ y_c \ \alpha_c]^T\tag{1.10}$$

$$q_r = \begin{bmatrix} x_r & y_r & \alpha_r \end{bmatrix}^T \quad (1.11)$$

Dessa forma o Erro pode ser definido como:

$$q_e = \begin{bmatrix} x_e \\ y_e \\ \alpha_e \end{bmatrix} = \begin{bmatrix} \cos \alpha(x_r - x_c) + \sin \alpha(y_r - y_c) \\ -\sin \alpha(x_r - x_c) + \cos \alpha(y_r - y_c) \\ \alpha_r - \alpha_c \end{bmatrix} \quad (1.12)$$

As velocidades de referência podem ser obtidas de maneira simples com base na Figura 4 onde:

$$\begin{aligned} v_r &= \sqrt{(\dot{x}_r)^2 + (\dot{y}_r)^2} \\ \omega_r &= \dot{\alpha}_r \end{aligned} \quad (1.13)$$

As velocidades desejadas  $v_d$  e  $\omega_d$  podem então ser derivadas pelo controlador proposto:

$$\begin{aligned} v_d &= v_r \cos(\alpha_e) + K_x x_e \\ \omega_d &= \omega_r + v_r(K_y y_e + K_\alpha \sin(\alpha_e)), \end{aligned} \quad (1.14)$$

onde  $K_x$ ,  $K_y$  e  $K_\alpha$  são os ganhos do controlador. As velocidades de cada roda podem ser obtidas pela equação 1.3.

#### 1.1.4 Pioneer 3-DX

O robô utilizado nesse experimento foi o *Pioneer 3-DX* que foi desenvolvido pela OMRON ADEPT MOBILEROBOTS e tem características bastante úteis. O Pioneer 3 DX é um robô móvel de acionamento diferencial compacto com controlador embarcado, motores com codificadores de 500 ticks, rodas de 19cm, corpo de alumínio resistente, 8 sensores ultrassônicos voltados para a frente, 8 sonares opcionais voltados para trás, 1, 2 ou 3 baterias *hot-swap* podendo também ser conectado diretamente com o computador.

## 1.2 Objetivos

Fazer com que o robô *pioneer 3-DX* percorra uma trajetória arbitrariamente definida e comparar o resultado do método com o valor simulado utilizando o *software V-Rep*.

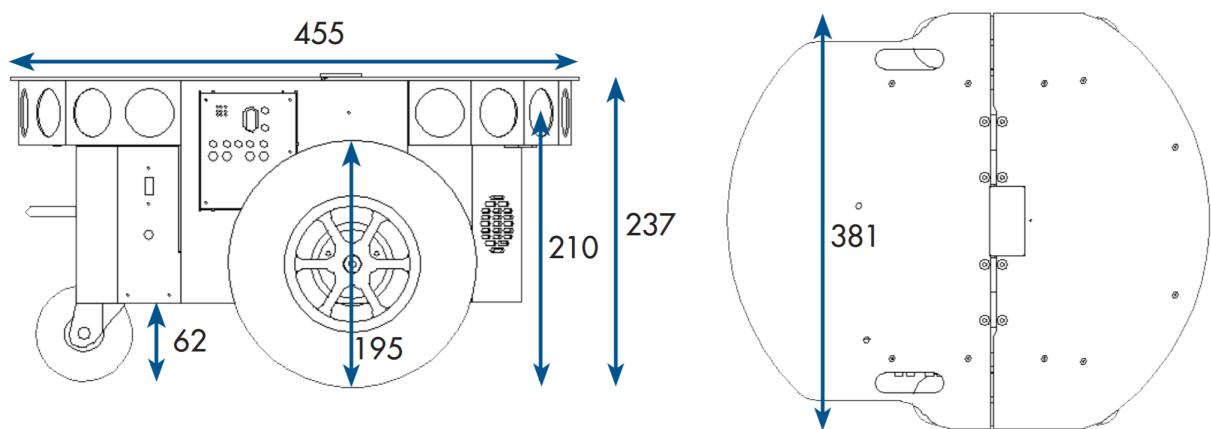


Figura 4 – Dimensões do robô *Pioneer 3-DX*

## 2 Realização do Experimento

### 2.1 Material Utilizado

Nesse experimento foram utilizados os *softwares* MATLAB® fornecido pela MathWorks e o simulador V-Rep da Coppelia Robotics.

### 2.2 Procedimento Experimental

Primeiramente escreve-se os códigos referentes ao método de seguidor de trajetório proposto por (KANAYAMA et al., 1990) onde é possível definir a função  $f(x, y)$  a ser seguida e depois realizamos a conexão entre o MATLAB e o V-Rep para realizar a simulação com o robô que está disponível na biblioteca *mobile* de robôs. É importante expandir o chão para uma área de 50 m por 50 m para evitar problemas de queda durante as simulações. Foi alterada as trajetórias e os gráficos foram gerados para cada uma comparando com a referência imposta ao movimento do robô. Os códigos utilizados estão descritos na seção de Anexos. As funções utilizadas e as posições inciais foram as seguintes:

- Seno:

$$\begin{aligned} f(x, y) &= y - 3 \sin \frac{\pi x}{9} \\ f_x &= -\frac{\pi}{3} \cos \frac{\pi x}{9} \\ f_{xx} &= \frac{\pi^2}{27} \sin \frac{\pi x}{9} \\ f_{xy} &= 0 \\ f_y &= 1 \\ f_{yy} &= 0 \end{aligned} \tag{2.1}$$

$$q_0 = [0 \ 5 \ 0]^T \tag{2.2}$$

- Círculo:

$$\begin{aligned}
 f(x, y) &= y^2 + x^2 - 100 \\
 f_x &= 2x \\
 f_{xx} &= 2 \\
 f_{xy} &= 0 \\
 f_y &= 2y \\
 f_{yy} &= 2
 \end{aligned} \tag{2.3}$$

$$q_0 = [0 \ 5 \ 0]^T \tag{2.4}$$

- Parábola:

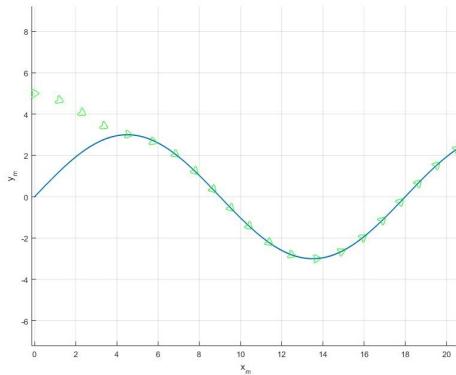
$$\begin{aligned}
 f(x, y) &= (x + 2)^2 - y \\
 f_x &= 2(x + 2) \\
 f_{xx} &= 2 \\
 f_{xy} &= 0 \\
 f_y &= -1 \\
 f_{yy} &= 0
 \end{aligned} \tag{2.5}$$

$$q_0 = [3 \ 5 \ 0]^T \tag{2.6}$$

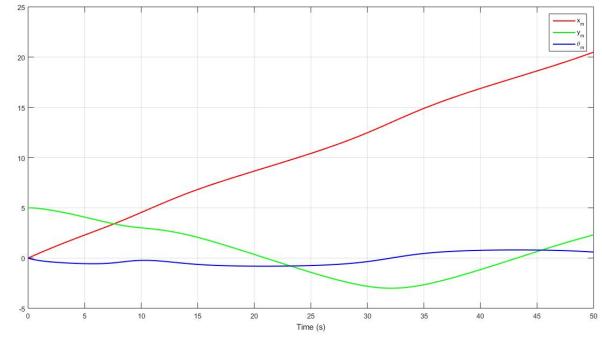
### 3 Resultados

Utilizando as equações de curvas utilizadas tem-se, para as constantes definidas como  $K_x = 1.5$ ,  $K_y = 8$  e  $K_\alpha = 180$  tem-se as curvas abaixo, onde para as curvas teóricas (azul) foi comparada com as posições do veículo (verde), ao lado está a pose do veículo ao longo do tempo. Após esses resultados, são mostrados os resultados simulados com os parâmetros já listados, tem-se a curva de referência (azul) e a realizada pelo veículo (laranja) e ao lado a evolução da pose do robô.

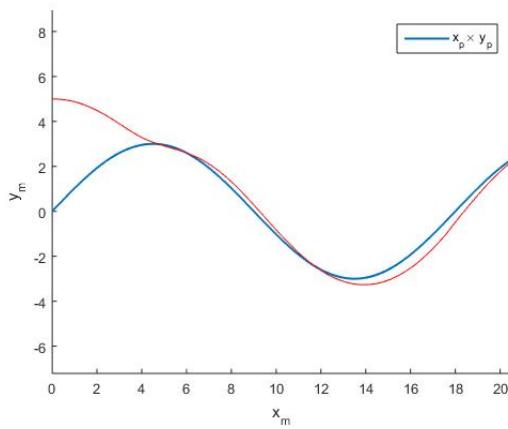
Seno



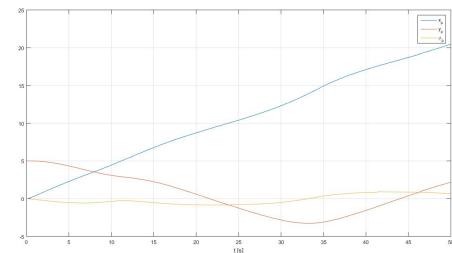
(a) Curva de superfície teórica



(b) Pose do robô ao longo do tempo

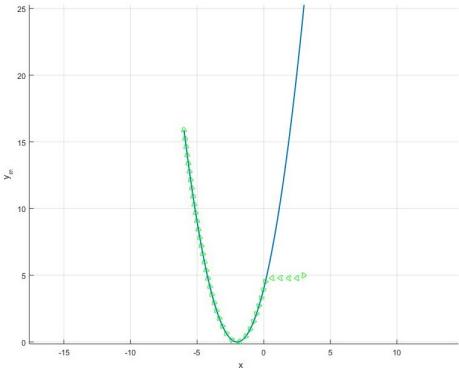


(c) Curva de superfície simulada

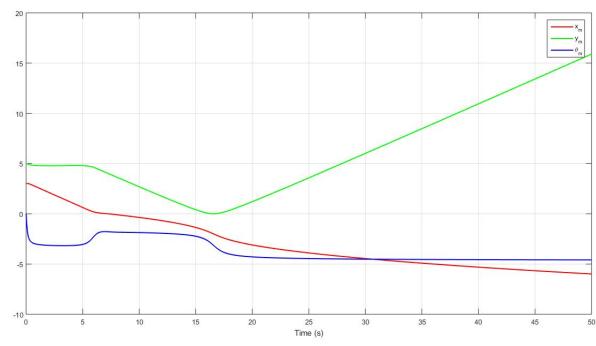


(d) Pose do robô simulado ao longo do tempo

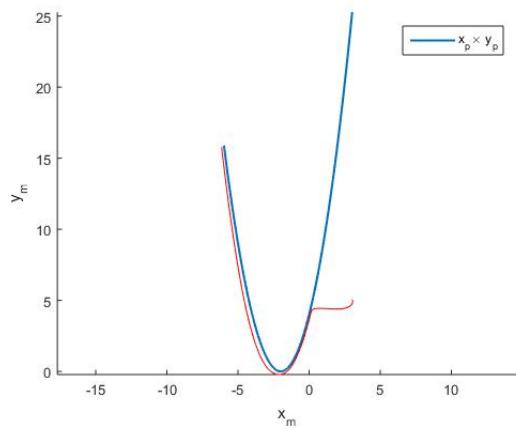
### Parábola



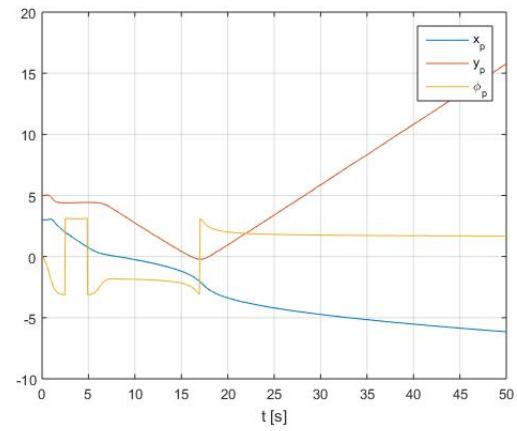
(a) Curva de superfície teórica



(b) Pose do robô ao longo do tempo

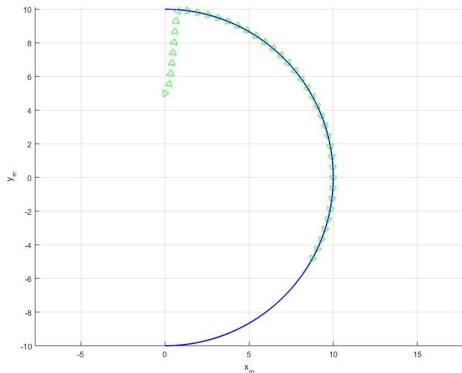


(c) Curva de superfície simulada

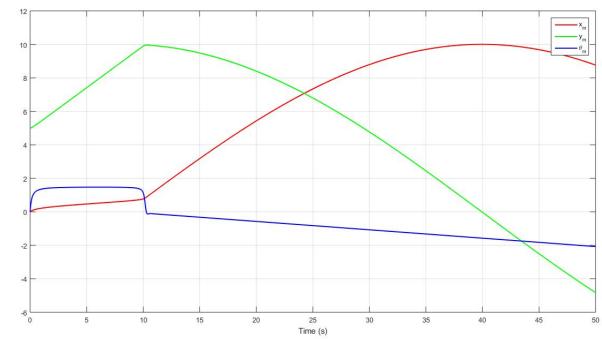


(d) Pose do robô simulado ao longo do tempo

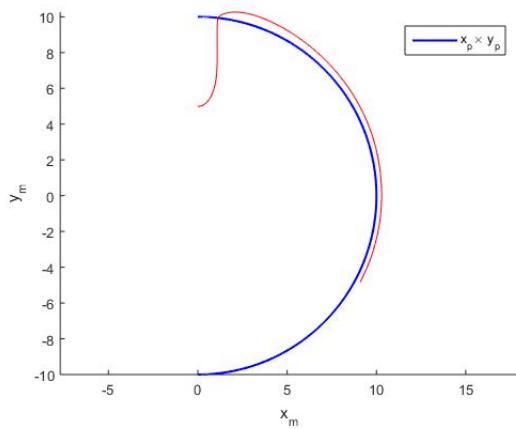
### Semicírculo



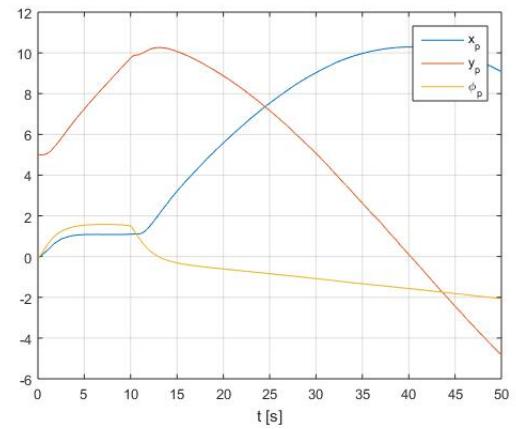
(a) Curva de superfície teórica



(b) Pose do robô ao longo do tempo



(c) Curva de superfície simulada



(d) Pose do robô simulado ao longo do tempo



## 4 Conclusão

Neste experimento foi possível obter um modelo de controle de trajetória para veículos monociclos em que é possível definir uma trajetória arbitrária e uma posição inicial arbitrária para seguir. Os resultados obtidos foram bastante satisfatório e os algorítimos desenvolvidos mostraram-se válidos para realizar a atividade proposta. Uma vez definido o algorítimo para seguir a trajetória é possível desenvolver outras áreas da navegação robótica, como o planejamento de trajetória e algorímos de SLAM (*Self Localization and Mapping*).



# Anexos

Função para cálculo das equações diferenciais:

```

1 function zp = sgorbissa(t,z)
2 %state
3 x=z(1,1);
4 y=z(2,1);
5 q=z(3,1);
6 qc=z(4,1);
7 %
8 f=(x+2)^2 - y;
9 fx= 2*(x+2);
10 fxx=2;
11 fxy=0;
12 fy=-1;
13 fyy=0;
14 %
15 Vf=sqrt(fx^2+fy^2);
16 K2=1.0;
17 S=K2*f/sqrt(1+f^2);
18 %
19 u=0.5;
20 dqc=((fx*fxy-fy*fxx)*u*cos(q)+(fx*fyy-fy*fxy)*u*sin(q))/Vf/Vf;
21 K1=2;
22 r=K1*(-Vf*u*S-fx*abs(u)*cos(q)-fy*abs(u)*sin(q))+dqc;
23 %
24 zp(1,1)=u*cos(q);
25 zp(2,1)=u*sin(q);
26 zp(3,1)=r;
27 zp(4,1)=dqc;
28 end

```

Código para realização do modelo teórico:

```
1 clear all
2 close all
3 clc
4 %Experiment 1
5 y0 = [3,5,0,0]';
6 y = y0(:);
7 %
8 ti = 0;
9 tf = 50;
10 tspan = ti:0.05:tf;
11
12 %
13 options = odeset('RelTol',1e-4,'AbsTol',1e-4*ones(size(y0)));
14 tic
15 [T,Y] = ode45(@(t,y)sgorbissa(t,y),tspan,y0,options);
16 toc
17 %
18 Xm = Y(:,1);
19 Ym = Y(:,2);
20 Qm = Y(:,3);
21 %
22 figure(1)
23 h = plot(T,Xm,'r',T,Ym,'g',T,Qm,'b');
24 set(h,'linewidth',1.5);
25 xlabel('Time (s)');
26 legend('x_{m}', 'y_{m}', '\theta_{m}')
27 grid on
28 %
29 figure(2)
30 if ~ishold,
31     hold on
32 end
33 axis('equal')
34 h=plot(Xm, (Xm+2).^2);
35 set(h,'linewidth',1.5);
36 xlabel('x_{m}');
37 ylabel('y_{m}');
38 grid on
39 for i=1:25:length(Xm),
40     vehicle(Xm(i),Ym(i),Qm(i),0.2);
41 end
42
43 Xp = [diff(Xm); 0];
44 Yp = [diff(Ym); 0];
```

```
45 Qp = [diff(Qm); 0];
```

Código para plotaegm do veículo

```

1 function []=vehicle(x,y,q,s)
2 p=[ 1           1/7
3      -3/7        1
4      -5/7        6/7
5      -5/7        5/7
6      -3/7        2/7
7      -3/7        0
8      -3/7        -2/7
9      -5/7        -5/7
10     -5/7        -6/7
11     -3/7        -1
12     1           -1/7
13     1           1/7 ];
14 %
15 p=s*p;
16 p=[p,ones(length(p),1)];
17 r=[cos(q),sin(q);-sin(q),cos(q);x,y];
18 p=p*r;
19 %
20 X=p(:,1);
21 Y=p(:,2);
22 %h = fill(X,Y,'r.');
23 plot(X,Y,'g-')
24 end
```

Código para controle da simulação no Vrep

```

1 %% P3DX dimensions
2 D = 0.195;
3 R = D/2;
4 L = 0.038;
5 Vmax = 1.2;
6 %% Running V-REP and loading test scene
7 %%winopen('cena.ttt');
8 %% Loading V-REP remote interface – client side
9 vrep = remApi('remoteApi');
10 %% Closing any previously opened connections
11 vrep.simxFinish(-1);
12 %% Connecting to remote V-REP API server
13 retCod = 0;
14 connectionAddress = '127.0.0.1';
```

```
15 connectionPort = 19997;
16 waitUntilConnected = true;
17 doNotReconnectOnceDisconnected = true;
18 timeOutInMs = 5000;
19 commThreadCycleInMs = 5;
20 while(retCod == 0)
21     [clientID]=vrep.simxStart(connectionAddress,connectionPort, ...
22         waitUntilConnected,doNotReconnectOnceDisconnected,timeOutInMs, ...
23         commThreadCycleInMs);
24     if(clientID > -1),
25         fprintf('Starting\n');
26     retCod = 1;
27     else
28         fprintf ('Waiting\n');
29     end
30 %% Getting robot handles
31 [retCod,rob1] = vrep.simxGetObjectHandle(clientID,'Pioneer_p3dx',...
32     vrep.simx_opmode_oneshot_wait);
33 [retCod,rob1LM] = vrep.simxGetObjectHandle(clientID,'...
34     Pioneer_p3dx_leftMotor',vrep.simx_opmode_oneshot_wait);
35 [retCod,rob1RM] = vrep.simxGetObjectHandle(clientID,'...
36     Pioneer_p3dx_rightMotor',vrep.simx_opmode_oneshot_wait);
37 %% Defining robot initial position
38 % Position
39 xp0 = y(1);
40 yp0 = y(2);
41 zp0 = R + 0.1;
42
43 % Orientation
44 ap0 = 0;
45 bp0 = 0;
46 cp0 = y(3);
47 [retCod] = vrep.simxSetObjectPosition(clientID,rob1,-1,[xp0,yp0,zp0],...
48     vrep.simx_opmode_oneshot);
49 [retCod] = vrep.simxSetObjectOrientation(clientID,rob1,-1,[ap0,bp0, ...
50     cp0],vrep.simx_opmode_oneshot);
51 %% Starting V-REP simulation
52 [retCod] = vrep.simxStartSimulation(clientID,vrep.simx_opmode_oneshot...
53     );
54 %% Defining V-REP client side controller parameters
55 np = length(T);
56 hd = 50e-3;
57 %tf = 50;
58 tf = (np-1)*hd;
59 tc = 0;
60 td = 0;
```

```

54 %
55 Ups = Vmax/4;
56 OmeMax = 1.00;
57 fd = 1/2;
58 %
59 t = zeros(np,1);
60 xp = zeros(np,1);
61 yp = zeros(np,1);
62 fp = zeros(np,1);
63 vp = zeros(np,1);
64 wp = zeros(np,1);
65
66 %
67 id = 1;
68
69 %
70
71 Kx = 1.5;
72 Ky = 8;
73 Ka = 140;
74
75 %% Main control loop - V-REP client side
76 t0 = cputime
77 while tc < tf,
78     tc = cputime - t0;
79     %% Current sampling instant
80     if tc > td,
81         t(id) = tc;
82         %% Measuring and saving
83         [retCod,rob1Pos] = vrep.simxGetObjectPosition(clientID,rob1...
84             ,-1,vrep.simx_opmode_oneshot_wait);
85         [retCod,rob1Ori] = vrep.simxGetObjectOrientation(clientID, ...
86             rob1,-1,vrep.simx_opmode_oneshot_wait);
87         % Robot pose
88         xa = rob1Pos(1,1);
89         ya = rob1Pos(1,2);
90         fa = rob1Ori(1,3);
91         xp(id) = xa;
92         yp(id) = ya;
93         fp(id) = fa;
94
95         %% Controlling
96         ex = (cos(fa)*(Xm(id)-xa)+sin(fa)*(Ym(id)-ya));
97         ey = (-sin(fa)*(Xm(id)-xa)+cos(fa)*(Ym(id)-ya));
98         ea = Qm(id)-fa;

```

```
99         Vr = sqrt((Xp(id)^2)+(Yp(id)^2));
100        Wr = Qp(id);
101
102        %Vd = Vr;
103        Vd = Vr*cos(ea)+Kx*ex;
104        %Wd = Qp(id);
105        Wd = Wr + Vr*(Ky*ey+Ka*sin(ea));
106
107        % Differential velocities
108        leftVel=(Vd - L*Wd)/R;
109        rightVel=(Vd + L*Wd)/R;
110
111        %% Actuating
112        [retCod] = vrep.simxSetJointTargetVelocity(clientID,rob1LM, ...
113            leftVel,vrep.simx_opmode_oneshot);
114        [retCod] = vrep.simxSetJointTargetVelocity(clientID,rob1RM, ...
115            rightVel,vrep.simx_opmode_oneshot);
116
117        %% Next sampling instant
118
119        td = td + hd;
120        id = id + 1;
121        %td = T(id);
122    end
123
124    %% Stoping V-REP simulation
125    vrep.simxStopSimulation(clientID,vrep.simx_opmode_oneshot_wait);
126    fprintf('Ending\n');
127
128    %% Plotting results
129    figure(3)
130    plot(t,xp,t,yp,t,fp),grid
131    legend('x_p','y_p','\phi_p')
132    xlabel('t [s]')
133
134    figure(4)
135    if ~ishold,
136        hold on
137    axis('equal')
138    h=plot(Xm,(Xm+2).^2);
139    set(h,'linewidth',1.5);
140    xlabel('x_{m}');
141    ylabel('y_{m}');
142    grid on
143    plot(xp,yp,'r-'),grid
```

```
144 legend('x_p\times y_p')  
145 %% Stopping V-REP  
146 %dos('taskkill /F /IM vrep.exe');
```

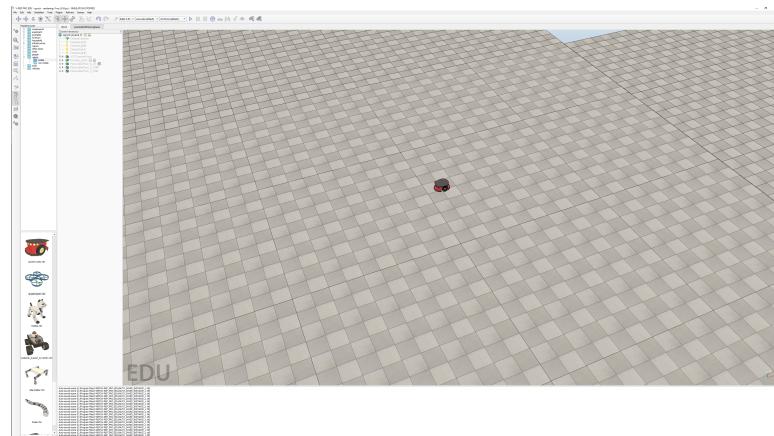


Figura 5 – Ambiente de Simulação



## Referências

KANAYAMA, Y. et al. A stable tracking control method for an autonomous mobile robot. In: *Proceedings., IEEE International Conference on Robotics and Automation*. [S.l.: s.n.], 1990. p. 384–389 vol.1. Citado 3 vezes nas páginas 1, 8 e 11.

MORRO, A.; SGORBISSA, A.; ZACCARIA, R. Path following for unicycle robots with an arbitrary path curvature. *IEEE Transactions on Robotics*, v. 27, n. 5, p. 1016–1023, Outubro 2011. ISSN 1552-3098. Citado 2 vezes nas páginas 6 e 7.