Nome: Fabio Grassiotto
RA: 890441
Disciplina: IA368N, 2º S 2018

Atividade 2 – Cinemática e controle de um veículo de acionamento diferencial

Objetivo

O objetivo desta atividade foi implementar um controlador de malha fechada para a movimentação de um robô de acionamento diferencial.

Tarefa 1

Para esta primeira tarefa, foram computadas as velocidades de rotação ϕr e ϕl baseadas nas velocidades v e ω em uma trajetória circular.

Dado que o modelo cinemático descreve as velocidades v e ω através do sistema de equações abaixo:

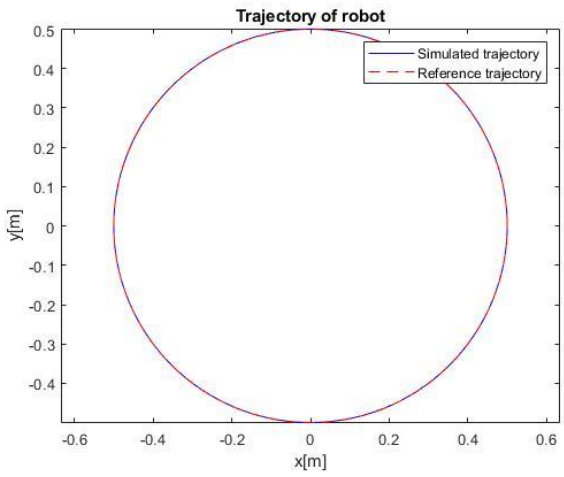
$$v = \frac{r\varphi_r}{2} + \frac{r\varphi_l}{2} \qquad (1)$$

$$\omega = \frac{r\varphi_r}{2l} - \frac{r\varphi_l}{2l} \qquad (2)$$

Isolando-se as variáveis ϕr e ϕl, foi alterado o arquivo calculateWheelSpeeds.m executando o cálculo das velocidades de rotação.

```
% Using system of 2 equations given by the problem we arrive at
LeftWheelVelocity = (vu - omega*halfWheelbase)/wheelRadius;
RightWheelVelocity = (vu + omega*halfWheelbase)/wheelRadius;
```

O resultado obtido pode ser verificado assim:



Tarefa 2

Nesta segunda tarefa, se propõe a teleoperação do robô utilizando o teclado.

O arquivo teleoperation.m foi alterado, adicionando um loop para detecção de teclas (através da chamada da função getkeywaitchar()) e execução de comandos de movimentação, utilizando as teclas com o seguinte mapeamento:

- Teclas d/D: o robô imprime uma linha a frente.
- Teclas a/A: o robô imprime uma linha para trás.
- Teclas w/W: o robô executa uma rotação no sentido anti-horário.
- Teclas s/S: o robô executa uma rotação no sentido horário.
- Teclas x/X: o robô finaliza a teleoperação.

```
1.  %% teleoperation program goes here
2.  loop = 1;
3.  keybuffer = 0; % to buffer the last key pressed and increase speed if necessary.
4.  speed = 0.1; % initial speed, increased with repeated keypresses.
5.  while loop
6.      ch = getkeywaitchar(1); % gets keys every 1 second
7.
8.      % Sets speed
9.      if (ch == keybuffer)
10.         speed = speed + 0.1;
11.     else
12.         speed = 0.1; % resets speed
13.     end
14.     keybuffer = ch;
15.
16.     if (ch == 44 || ch == 100) % D/d key prints a linear fwd speed
17.         Pioneer_p3dx_setWheelSpeeds(connection, speed, speed);
18.     elseif (ch == 65 || ch == 97) % A/a key prints a backwards fwd speed
19.         Pioneer_p3dx_setWheelSpeeds(connection, -speed, -speed);
20.     elseif (ch == 87 || ch == 119) % W/w key rotates counterclockwise
21.         Pioneer_p3dx_setWheelSpeeds(connection, 0.0, speed);
22.     elseif (ch == 83 || ch == 115) % S/s key rotates clockwise
23.         Pioneer_p3dx_setWheelSpeeds(connection, speed, 0.0);
24.     elseif (ch == 32) % stops robot
25.         Pioneer_p3dx_setWheelSpeeds(connection, 0.0, 0.0);
26.     elseif (ch == 88 || ch == 120) % X/x quits teleoperation
27.         loop = 0;
28.     end
29. end
```

Executando o programa, pode-se verificar que o comando atua corretamente, aumentando de forma gradual a velocidade do tracejado das linhas ao se pressionar uma mesma tecla repetidamente.

Tarefas 3 e 4

Para as duas tarefas finais do exercício, a lei de controle de feedback de estado linear descrita no livro Introdução aos Robôs Móveis Autônomos que acompanham esta tarefa pôde ser implementada.

A lei de controle pode ser sumarizada e descrita na figura abaixo.

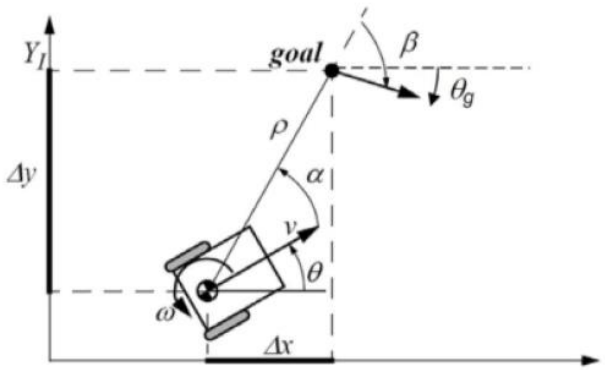


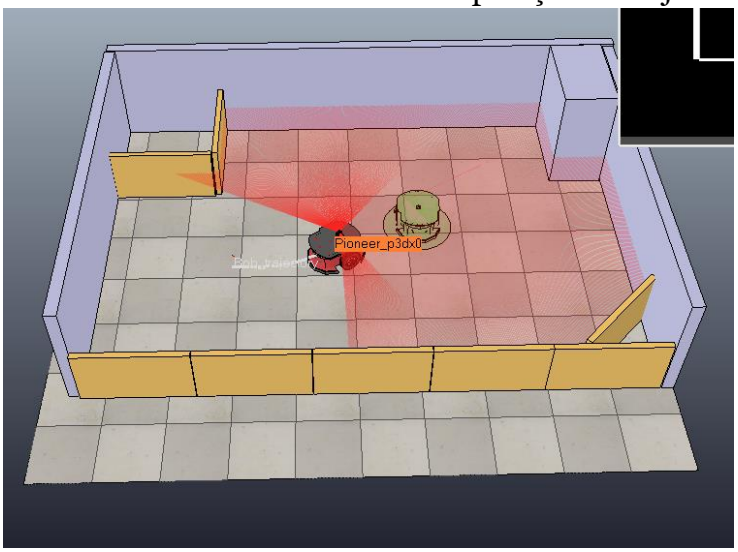
Figure 1: State feedback control onto a reference pose

Para tarefa 3, foi implementado um controlador de malha fechada simples no arquivo CalculateControlOutput.m, inicialmente calculando o valor do ângulo beta a partir dos ângulos theta e alpha, e posteriormente calculando as velocidades v e ω:

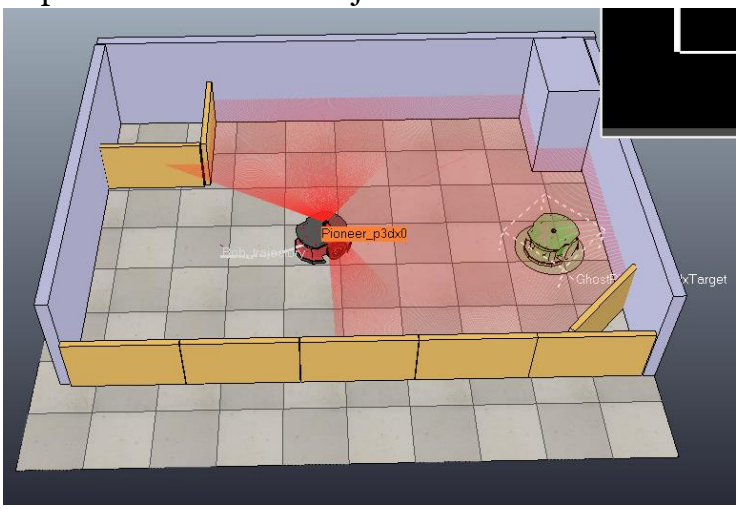
```
% calculate beta
beta = - theta - alpha;

% Now calculate velocity and omega.
vu = parameters.Krho*rho;
omega = parameters.Kalpha*alpha + parameters.Kbeta*beta;
```

Resultados obtidos:

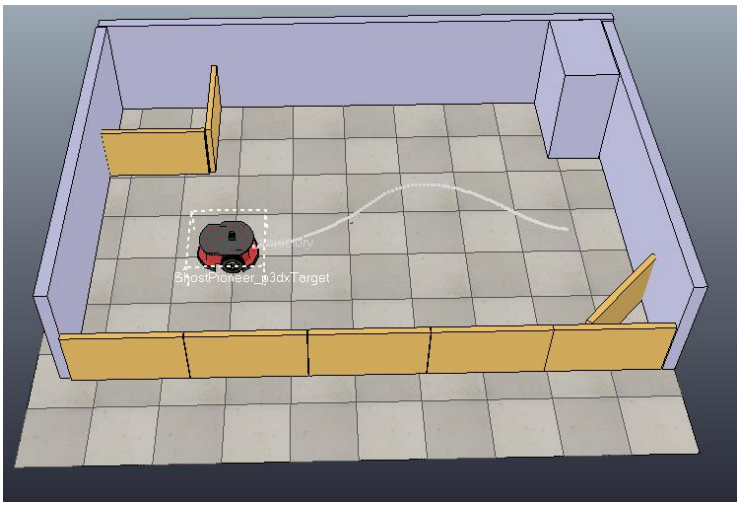
- Posicionamento inicial do robô e posição de objetivo:



- Reposicionamento do objetivo:



- Rota resultante do robô:



Após obter os resultados acima, o objetivo da Tarefa 4 foi melhorar o loop de controle fechado.

Buscou-se implementar as seguintes melhorias no controle do robô:

- Movimentação com velocidade constante.
- Movimentação do robô em duas direções de acordo com o posicionamento do objetivo.

O controlador de malha fechada foi alterado para executar a leitura de parâmetros de entrada e implementar as funcionalidades definidas acima.

Para tanto, após se calcular o valor de beta, o parâmetro que permite que o robô se movimente para trás foi avaliado e foi criada uma variável para alterar o sinal da velocidade linear resultante, *vel_factor*. Essa variável assume um valor negativo baseado no ângulo alpha qu emede a direção do objetivo em relação ao robô.

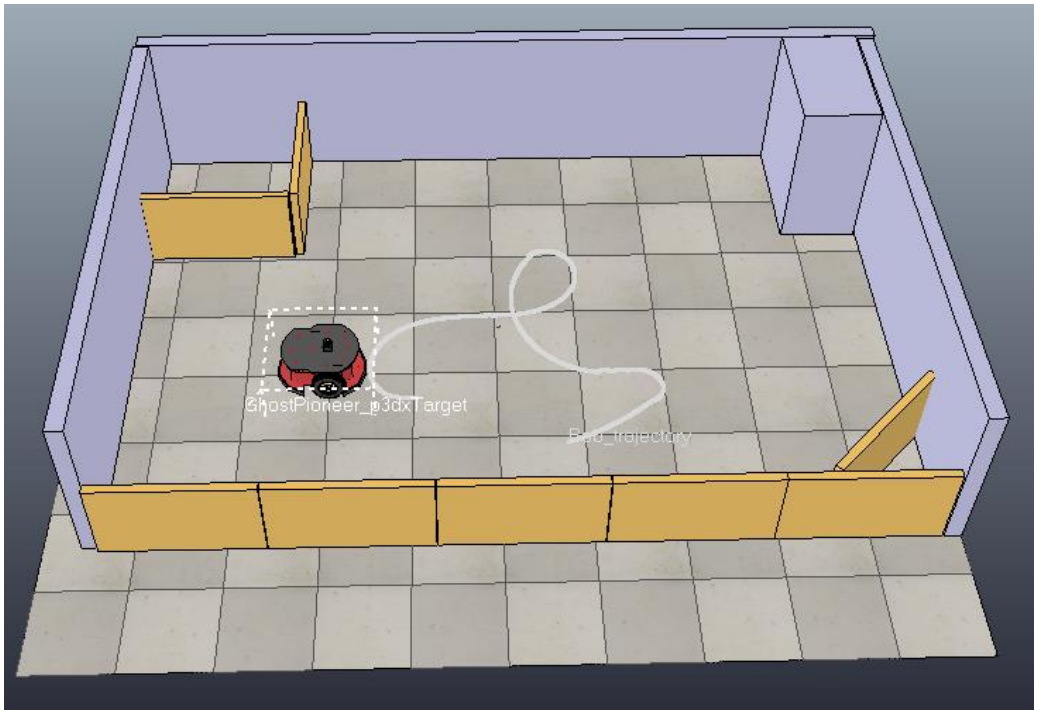
```
1.  % calculate beta
```

```
beta = - theta - alpha;

% the following paramerters should be used:
% Task 3:
% parameters.Kalpha, parameters.Kbeta, parameters.Krho: controller tuning parameters
% Task 4:
% parameters.backwardAllowed: This boolean variable should switch the between the two controllers
% parameters.useConstantSpeed: Turn on constant speed option
% parameters.constantSpeed: The speed used when constant speed option is on

% backwards velocities allowed:
if (parameters.backwardAllowed == true)
    if ( alpha <= pi/2 && alpha >= -pi/2)
        % Alpha is normalized between [-pi,pi].
        % If the angle of the vector pointing from the robot to the goal in
        % the robot frame is between -90 and 90 degrees, a positive velocity is a faster
        % alternative for the controller. Otherwise, use a negative one.
        vel_factor = 1.0;
    else
        vel_factor = -1.0;
    end
end

```

Após se determinar o fator de inversão de velocidades, o parâmetro que determina o uso de velocidade constante foi medido e, quando ativado, foi executado o escalonamento da velocidade angular ω conforme abaixo.

```
% Now calculate velocity and omega.
if (parameters.useConstantSpeed == false)
    % Constant speed setting is off.
    vu = vel_factor*parameters.Krho*rho;
    omega = parameters.Kalpha*alpha + parameters.Kbeta*beta;
else
    % Constant speed setting is on.
    vu_pre = parameters.Krho*rho;
    omega_pre = parameters.Kalpha*alpha + parameters.Kbeta*beta;

    % Now scale vu and omega to keep vu constant.
    % Using the fact that the quotient between v/w is kept constant.
    vu = vel_factor*parameters.constantSpeed;
    omega = omega_pre * (vu/vu_pre);
end
```

Com as melhorias implementadas, nota-se que o robô se tornou capaz de manobrar em menores espaços conforme pode ser verificado na figura abaixo:



Conclusão

Os exercícios propostos permitiram exercitar os passos necessários para o controle de um veículo com acionamento diferencial.

O ambiente de simulação V-Rep foi utilizado juntamente com o Matlab para verificar a performance do controlador de malha fechada em velocidades variável e constante.