Nome: Fabio Grassiotto

RA: 890441

Disciplina: IA368N, 2º S 2018

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

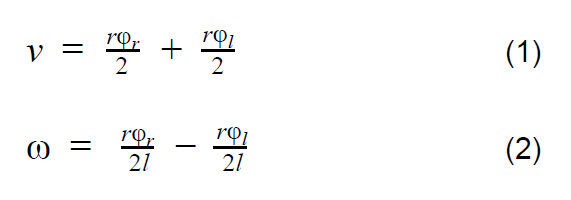
**Objetivo**

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.

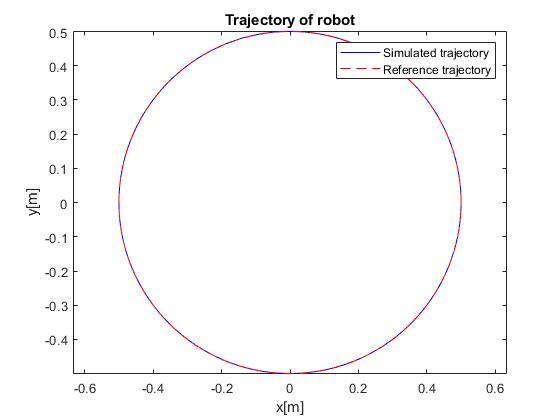
Dado que o modelo cinemático descreve as velocidades v e ω através do sistema de equações abaixo:



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

1. % Using system of 2 equations given by the problem we arrive at
2. LeftWheelVelocity = (vu - omega\*halfWheelbase)/wheelRadius;
3. 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.

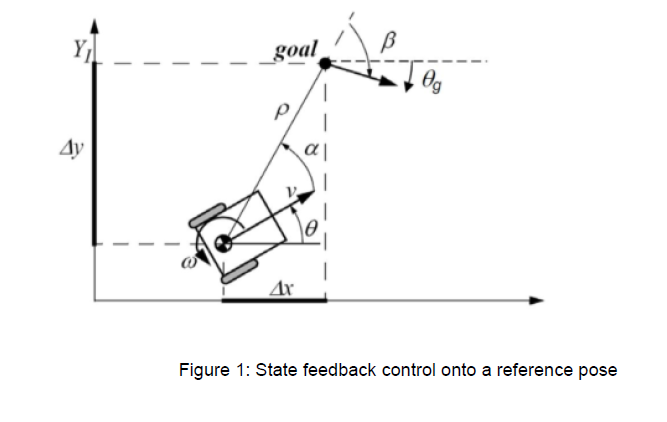
1. %% teleoperation program goes here
2. loop = 1;
3. keybuffer = 0; % to buffer the last key pressed and increase speed **if** necessary.
4. speed = 0.1; % initial speed, increased with repeated keypresses.
5. **while** loop
6. ch = getkeywaitchar(1); % gets keys every 1 second
8. % Sets speed
9. **if** (ch == keybuffer)
10. speed = speed + 0.1;
11. **else**
12. speed = 0.1; % resets speed
13. end
14. keybuffer = ch;
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.



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:

1. % calculate beta
2. beta = - theta - alpha;
4. % Now calculate velocity and omega.
5. vu = parameters.Krho\*rho;
6. omega = parameters.Kalpha\*alpha + parameters.Kbeta\*beta;

Resultados obtidos:

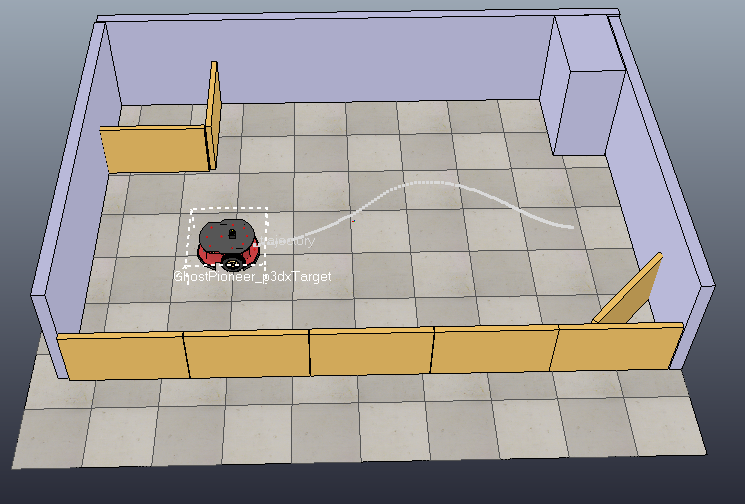
* Posicionamento inicial do robô e posição de objetivo:



* 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.

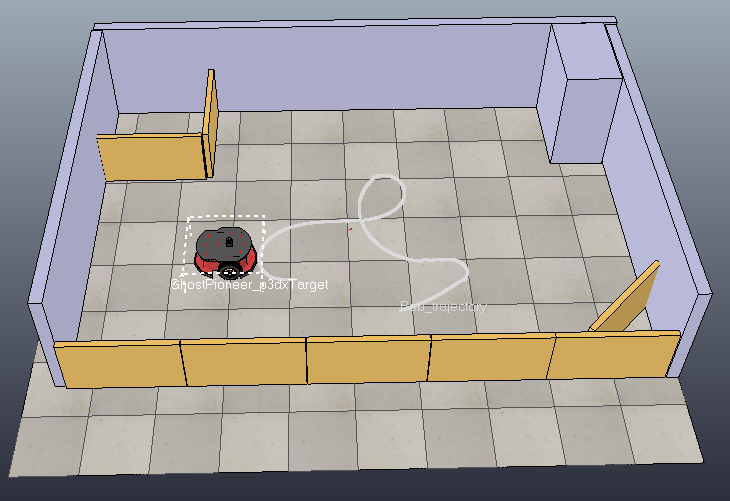
Para tanto, após se calcular o valor de beta, o parâmetro que permite que o robô se movimente para trás foi avaliado e foi criada uma variável para alterar o sinal da velocidade linear resultante, *vel\_factor.* Essa variável assume um valor negativo baseado no ângulo alpha qu emede a direção do objetivo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

1. % calculate beta
2. beta = - theta - alpha;
4. % the following paramerters should be used:
5. % Task 3:
6. % parameters.Kalpha, parameters.Kbeta, parameters.Krho: controller tuning parameters
7. % Task 4:
8. % parameters.backwardAllowed: This boolean variable should **switch** the between the two controllers
9. % parameters.useConstantSpeed: Turn on constant speed option
10. % parameters.constantSpeed: The speed used when constant speed option is on
12. % backwards velocities allowed:
13. **if** (parameters.backwardAllowed == **true**)
14. **if** ( alpha <= pi/2 && alpha >= -pi/2)
15. % Alpha is normalized between [-pi,pi].
16. % If the angle of the vector pointing from the robot to the goal in
17. % the robot frame is between -90 and 90 degrees, a positive velocity is a faster
18. % alternative **for** the controller. Otherwise, use a negative one.
19. vel\_factor = 1.0;
20. **else**
21. vel\_factor = -1.0;
22. end
23. 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.

1. % Now calculate velocity and omega.
2. **if** (parameters.useConstantSpeed == **false**)
3. % Constant speed setting is off.
4. vu = vel\_factor\*parameters.Krho\*rho;
5. omega = parameters.Kalpha\*alpha + parameters.Kbeta\*beta;
6. **else**
7. % Constant speed setting is on.
8. vu\_pre = parameters.Krho\*rho;
9. omega\_pre = parameters.Kalpha\*alpha + parameters.Kbeta\*beta;
11. % Now scale vu and omega to keep vu constant.
12. % Using the fact that the quotient between v/w is kept constant.
13. vu = vel\_factor\*parameters.constantSpeed;
14. omega = omega\_pre \* (vu/vu\_pre);
15. 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.