MESTRADO EM ENGENHARIA E DE COMPUTADORES

Relatório de Sistema Robóticos Autónomos

Modelação e controlo de movimento
da plataforma móvel de condução diferencial

Duarte Cruz, 2017264057
João Pedro Chaves Castilho, 2017263424

Universidade de Coimbra
12 de março de 2022

1 Introdução

Neste trabalho laboratorial tínhamos como objetivo simular o comportamento de uma plataforma de condução diferencial, nomeadamente três movimentos diferentes: movimento para um ponto, movimento segundo uma linha e movimento para uma pose arbitrária. Foi usado o MatLab para a programação dos algoritmos e o Gazebo para simular o comportamento de uma plataforma diferencial, nomeadamente o Turtlebot.

2 Movimento para um ponto

Neste movimento era-nos pedido, sendo (x, y, θ) a pose atual da plataforma, implementar um algoritmo que levasse a plataforma para um qualquer ponto (x^*, y^*). A implementação descrita nesta secção está presente no ficheiro *control_1.m*

Para realizar este movimento vamos implementar dois controladores, um controlador para a velocidade linear $v(k)$:

$$v(k) = k_v \cdot e(k) \tag{1}$$

sendo $e(k)$ o erro de posição definido por:

$$e(k) = \sqrt{(x^* - x(k))^2 + (y^* - y(k))^2} \tag{2}$$

O controlador para o velocidade angular, $w(k)$ é definido por:

$$w(k) = k_s \cdot (\phi^* \perp \theta(k)), \quad k_s > 0 \tag{3}$$

sendo ϕ^* o ângulo objectivo para orientar a plataforma relativamente ao ponto objectivo, definido por:

$$\phi^* = tan^{-1} \frac{y^* - y(k)}{x^* - x(k)} \tag{4}$$

O operador $\perp$ pode ser calculado através de:

$$\phi^* \perp \theta(k) = atan2(sin(\phi^* - \theta(k)), cos(\phi^* - \theta(k))) \tag{5}$$

Os ganhos de (1) e (3) têm de ser escolhidos criteriosamente de modo que a plataforma tenha um comportamento estável. Para encontrarmos estes valores corremos a nossa simulação com vários valores de k_v e k_s e guardamos o percurso resultante numa pasta.

2.1 Resultados

Alguns resultados da nossa implementação podem ser observador a seguir:

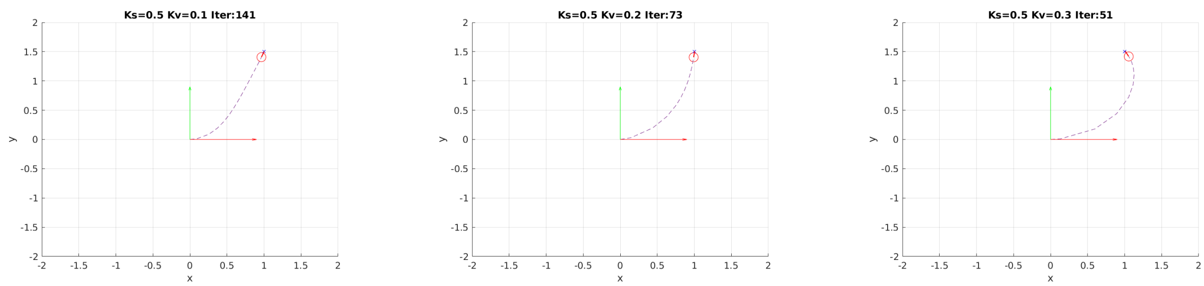

Figure 1: Efeito do k_v

Na Fig.1 podemos observar o efeito do ganho k_v. Este ganho tal como verificado na equação 1 é responsável pela velocidade linear da plataforma. Quanto maior este ganho, maior será a velocidade linear. Podemos verificar que quando este ganho é demasiado elevado, a plataforma tende a fazer movimentos mais curvilíneos, pois a velocidade angular da plataforma não consegue acompanhar a velocidade linear.

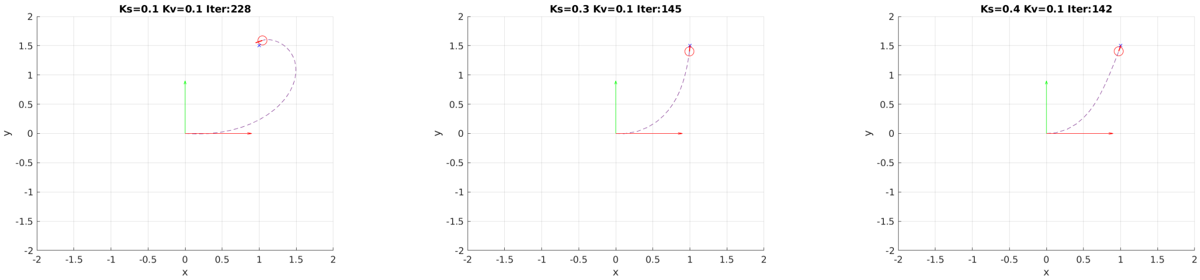

Figure 2: Efeito do k_s

Relativamente ao ganho k_s podemos observar na Fig.2 que quando aumentamos este ganho, para um k_v constante, a plataforma vai executar movimentos mais retilíneos.
Podemos então concluir que para esta estratégia de controlo de movimento para um ponto, para que a plataforma execute um bom movimento, é desejável que o k_s seja superior ao k_v.

Com a nossa implementação, também é possível que a plataforma efetue movimentos para uma sequência de pontos.

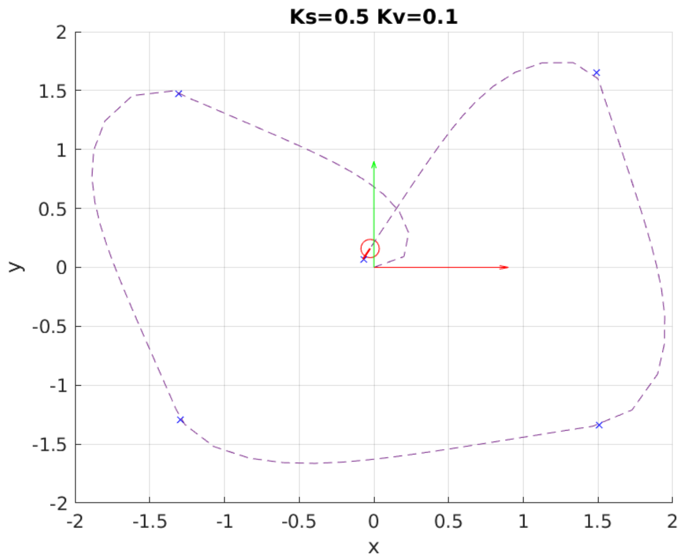

Figure 3: Movimento para vários pontos

3 Movimento segundo uma linha

O movimento descrito nesta secção está implementado na função *control_2.m*.
Neste movimento era-nos pedido que a plataforma efetuasse o seguimento de uma trajetória linear. Esta trajetória linear é definida no plano através de $ax + by + c = 0$. Para efetuar este movimento vamos precisar de implementar dois controladores: um para minimizar a distância da normal entre a plataforma e a linha, rodando a plataforma na direção da linha e outro controlador para ajustar o ângulo da plataforma de forma a esta ficar paralela à linha.

O primeiro controlador é definido por:

$$\alpha_d = -k_d d, \quad k_d > 0 \tag{6}$$

sendo d definido por:

$$d = \frac{(a, b, c) \cdot (x(k), y(k), 1)}{\sqrt{a^2 + b^2}} \tag{7}$$

O segundo controlador, é definido por:

$$\alpha_h = k_h(\phi^* \perp \theta(k)), \quad k_h > 0 \tag{8}$$

sendo ϕ^* o ângulo objetivo da plataforma, definido por:

$$\phi^* = tan^{-1}\frac{-a}{b} \tag{9}$$

Neste movimento temos apenas uma variável de controlo, a velocidade angular da plataforma. A velocidade linear é mantida constante ao longo de todo o movimento, sendo igual a v_{max}.
Por fim a lei de controlo combinada fica então:

$$\omega(k) = \alpha_d + \alpha_h \tag{10}$$

3.1 Resultados

Para que o nosso simulador seja mais interativo, damos a opção ao utilizador de escolher a reta que quer que a plataforma siga, escolhendo dois pontos presentes dessa reta.

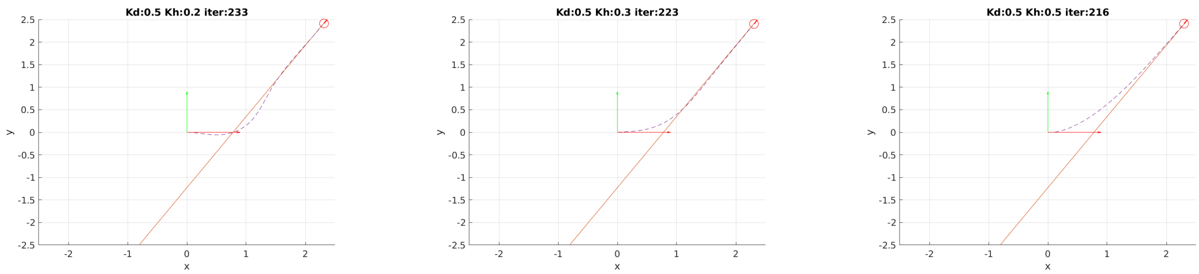

Figure 4: Efeito do k_h

Na Fig.4 podemos observar o efeito de k_h para um valor constante de $k_d = 0.5$. Verificamos que para um valor de k_h baixo, relativamente a k_d faz com que a plataforma tenha algum overshoot no seguimento da reta, no entanto, a plataforma diminui a distância para a reta mais rapidamente. Ao aumentarmos o valor de k_h o overshoot é diminuído e a plataforma demora mais tempo a anular o erro da distância à reta.

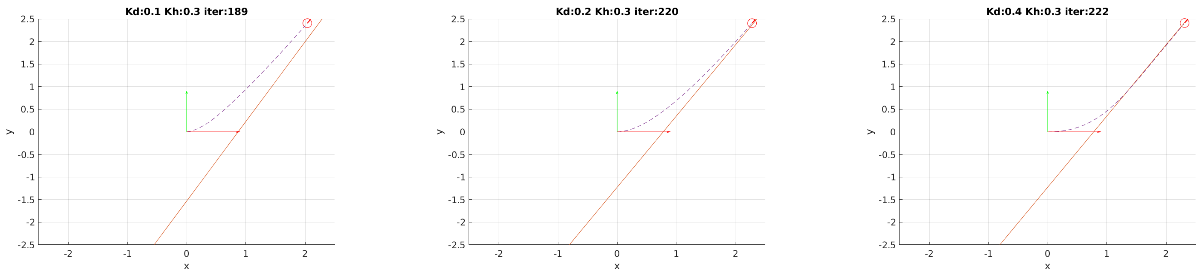

Figure 5: Efeito do k_d

No que diz respeito ao k_d, observamos um comportamento semelhante. Ao aumentarmos k_d obtemos uma melhor resposta. Podemos concluir, com os resultados presentes nas Fig.4 e 5 que para uma resposta adequado para o seguimento da linha, queremos que o k_d seja maior que o k_h. No entanto, esta diferença não pode ser muito grande, pois pode criar overshoot no seguimento da reta.

4 Movimento para uma pose arbitrária

Para este movimento era-nos pedido que para movimentar a plataforma da sua pose arbitrária, (x, y, θ), para uma pose objectivo (x^*, y^*, θ^*), sendo ambas as poses escolhidas pelo utilizador.

Usando a seguinte lei de controlo linear:

$$v = k_\rho \rho \tag{11}$$

$$w = k_\alpha \alpha + k_\beta \beta \tag{12}$$

O sistema de controlo em malha fechada vai ser:

$$\begin{bmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} -k_\rho.\rho.\cos\alpha \\ k_\rho.\sin\alpha - k_\alpha.\alpha - k_\beta.\beta \\ -k_\rho.\sin\alpha \end{bmatrix} \tag{13}$$

Isto conduz a plataforma para $(\rho, \alpha, \beta) = (0, 0, 0)$. Para o sistema ser estável têm de se cumprir as condições abaixo:

$$k_\rho > 0 \quad ; \quad k_\beta < 0 \quad ; \quad k_\alpha - k_\rho > 0 \tag{14}$$

É preciso ainda ter em atenção que as fórmulas anteriores consideram a pose objetivo como a origem, por isso foi necessário fazer a seguinte transformação de coordenadas para que a pose objetivo passe a ser a pose arbitrária, (x^*, y^*, θ^*), definida pelo utilizador.

$$x' = x - x^* \quad ; \quad y' = y - y^* \quad ; \quad \beta' = \beta + \phi^* \tag{15}$$

Para calcular os parâmetros de ρ, α e β para os instantes seguintes, utilizamos uma função do *Matlab* desenvolvida por nós chamada *update_parameters*. Esta função vai calcular as derivadas dos parâmetros através da equação (13) e depois vai calcular o valor desses parâmetros para o instante seguinte através da seguinte fórmula:

$$\begin{bmatrix} \rho(t+\Delta t) \\ \alpha(t+\Delta t) \\ \beta(t+\Delta t) \end{bmatrix} = \begin{bmatrix} \Delta t.\dot{\rho} + \rho(t) \\ \Delta t.\dot{\alpha} + \alpha(t) \\ \Delta t.\dot{\beta} + \beta(t) \end{bmatrix} \tag{16}$$

Para assim podermos estimar as velocidades (11) e (12) para os próximos instantes.

4.1 Resultados

Os resultados desta implementação podem ser visualizados em baixo.

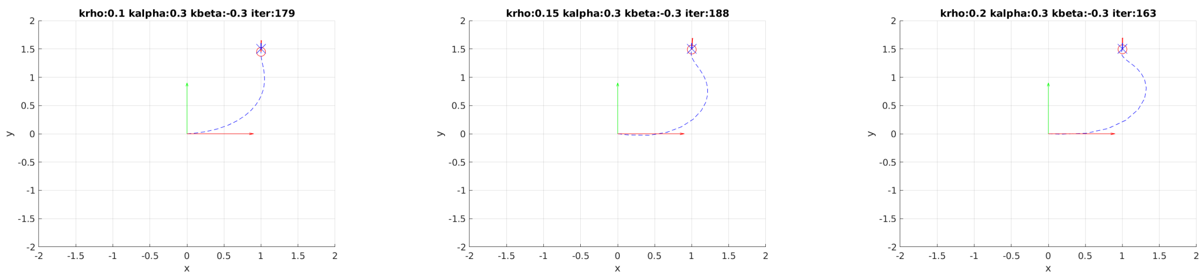

Figure 6: Efeito do k_ρ

Na Fig.6 podemos observar o efeito do ganho k_ρ no movimento da plataforma. Este ganho é responsável pela velocidade linear da plataforma, e quanto maior for este ganho, maior será a velocidade linear da plataforma. Podemos observar que quando aumentamos o k_ρ e mantemos os restantes ganhos, a plataforma vai realizar movimentos mais curvilíneos, pois a velocidade angular não consegue "acompanhar" a velocidade linear.

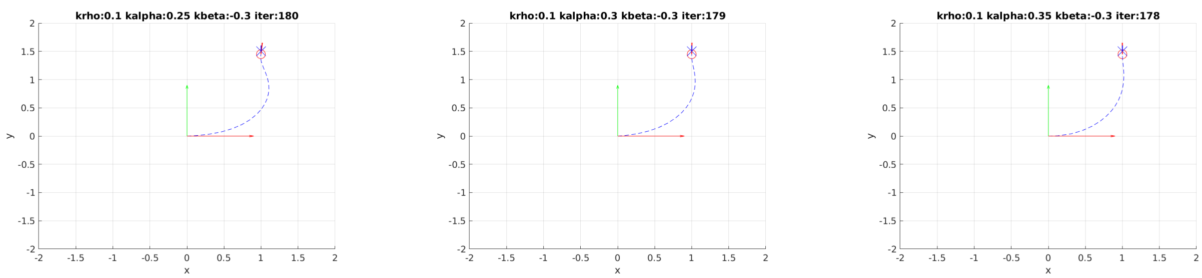

Figure 7: Efeito do k_α

Quando mantemos k_ρ e variamos os restantes ganhos, verificamos que ao aumentar k_α, responsável pela orientação da plataforma, vemos que a plataforma consegue ajustar a sua orientação para alinhar com a orientação desejada, mais rapidamente. O mesmo pode ser observado para o ganho k_β.

Alguns testes foram também feitos para um percurso com várias "poses".

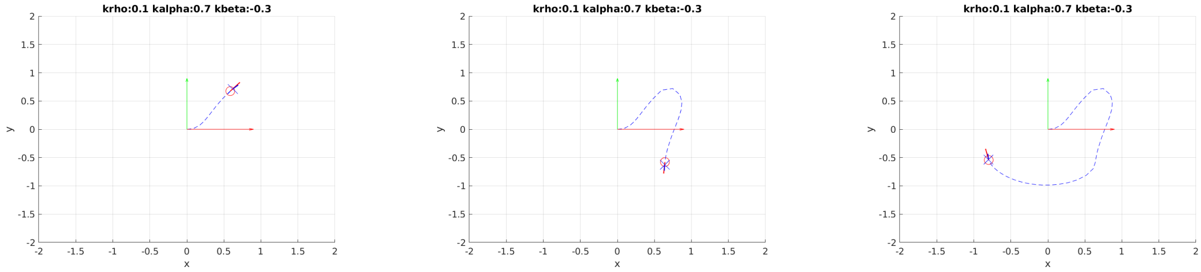

Figure 8: Sequência de posições

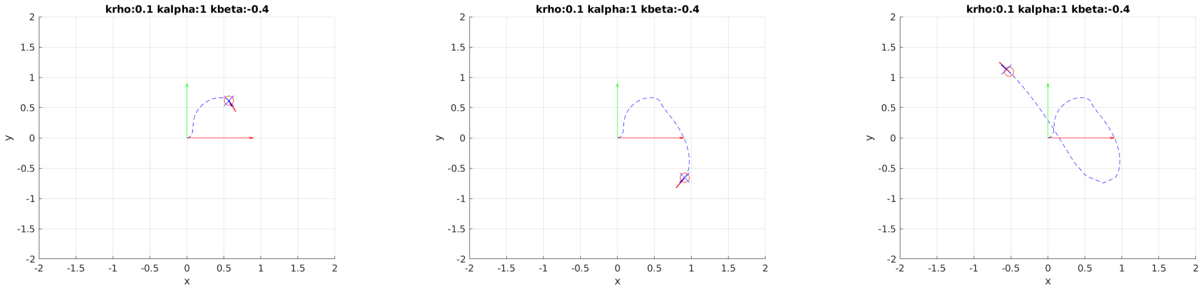

Figure 9: Sequência de posições

Podemos verificar, observando as Fig.8 e 9 que a nossa plataforma é capaz de atingir qualquer posição escolhida.

Noutra simulação que fizemos podemos observar a evolução das variáveis de controlo, e também a velocidade linear e angular da plataforma.

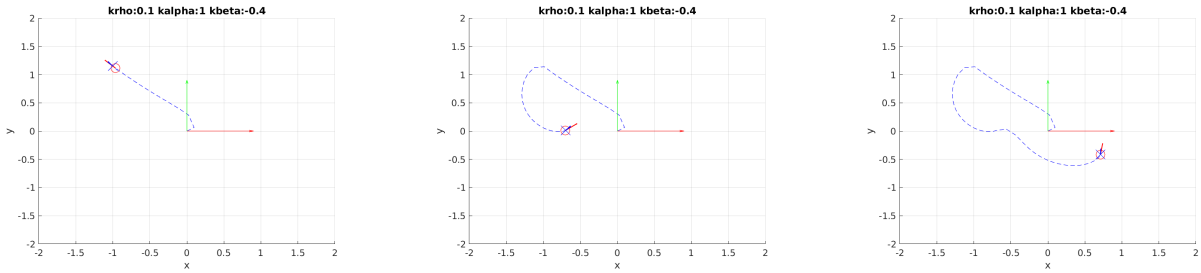

Figure 10: Sequência de posições

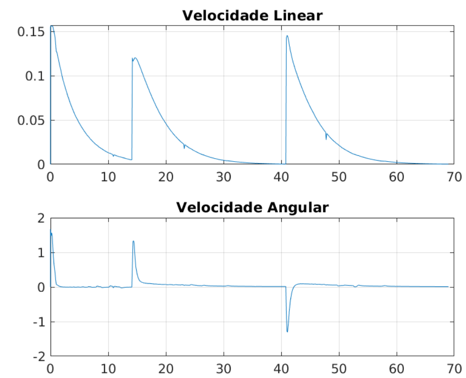

Figure 11: Evolução das velocidades

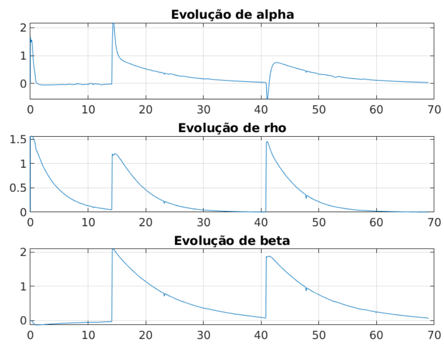

Figure 12: Evolução de variáveis de controlo

5 Apendix

5.1 Loop de controlo para movimento para o ponto

```
while (dist > 0.1)
    % read TurtleBot pose
    [x,y,theta] = tbot.readPose();

    dist = sqrt((goal_pose(1)-x)^2+(goal_pose(2)-y)^2);
    v = kv(j)*dist;
    phi = atan2((goal_pose(2)-y),(goal_pose(1)-x));
    w = ks(i)*atan2(sin(phi-theta),cos(phi-theta));
    tbot.setVelocity(v,w);
end
```

5.2 Loop de controlo para movimento linha

```
reta = [-4.03 2.55 3.14]; %ax+by+c=0

while (1)
    [x,y,theta] = tbot.readPose();

    d = dot(reta,[x,y,1])/sqrt(reta(1)^2+reta(2)^2);
    phi = atan2(-reta(1),reta(2));

    alpha_d = -kd(i)*d;
    alpha_h = kh(j)*atan2(sin(phi-theta),cos(phi-theta));
    w = alpha_d+alpha_h;

    tbot.setVelocity(0.08,w);
end
```

5.3 Loop de controlo para movimento para "pose"

```
while (rho_>0.05 || abs(beta_)>0.07 || abs(alpha_)>0.07)
    [x,y,theta]=tbot.readPose();
    dx = goal_pose(1) - x;
    dy = goal_pose(2) - y;
    rho_ = sqrt(dx^2 + dy^2);

    alpha_ = -theta + atan2(dy,dx);
    alpha_ = atan2(sin(alpha_),cos(alpha_));

    beta_ = -theta - alpha_ + goal_pose(3);
    beta_ = atan2(sin(beta_),cos(beta_));

    [rho_,alpha_,beta_] = update_parameters(rho_,alpha_,beta_,k_rho,k_alpha,k_beta,last_update);
    last_update = tic;
    v = k_rho*rho_;
    w = k_alpha*alpha_+k_beta*beta_;

    tbot.setVelocity(v,w);
end

function [rho_new, alpha_new, beta_new] = update_parameters(rho, alpha, beta, ...
                                        k_rho, k_alpha, k_beta,T)
    dRho = -k_rho*rho*cos(alpha);
```

```
    dAlpha = k_rho*sin(alpha)-k_alpha*alpha-k_beta*beta;
    dBeta = -k_rho*sin(alpha);

    rho_new = dRho*toc(T)+rho;
    alpha_new = dAlpha*toc(T)+alpha;
    alpha_new = atan2(sin(alpha_new),cos(alpha_new));
    beta_new = dBeta*toc(T)+beta;
    beta_new = atan2(sin(beta_new),cos(beta_new));
end
```