Faculdade de Engenharia da Universidade do Porto

Implementação de uma rotina para seguir um segmento de reta *(FollowLine)*

Sandro Augusto Costa Magalhães
Tiago José Ferreira Mendonça

VERSÃO FINAL

Trabalho realizada no âmbito da unidade curricular de
Sistemas Robóticos Autónomos

Docentes: Vitor Pinto e António Moreira

7 de Novembro de 2017

1 Introdução

No presente trabalho prático pretende-se que um robot móvel de tração diferencial seja capaz de seguir um segmento de reta, definido pelos seus pontos inicial e final, e que termine o seu movimento no ponto final estabelecido, com uma orientação equivalente à da reta. Para o efeito, em primeiro lugar, o objetivo consiste em aproximar o veículo ao ponto mais próximo da linha definida e, quando suficientemente próximo desta, iniciar o seu seguimento em sentido conveniente para atingir o ponto final. Posto isto, impõe-se a necessidade de implementação de dois controladores distintos: controlador para aproximação (*Controlador A*) e, posteriormente, comutação para controlador de seguimento (*Controlador B*). Estes controladores podem ser assumidos como dois estados distintos de uma máquina de estados, em que as transições são determinadas em função da distância ao ponto mais próximo da reta. A sua supervisão é conseguida com a implementação de um terceiro procedimento responsável pela comutação entre estados (*FollowLine*), estando, por isso, num nível hierarquicamente superior. Como premissa adicional do problema considera-se que o robot está sempre localizado entre os limites impostos pelos extremos do segmento.

2 *LineDistance* – Distância à linha

Esta função, implementada sob a forma de um procedimento, tem como objetivo calcular a distância ao ponto mais próximo do segmento de reta, tendo por bases os pontos extremos do mesmo e a posição atual do robot. Sendo assim, são necessários os seguintes passos:

- Definição de vetores entre a origem do referencial e a posição atual do robot ($\vec{P}$), a posição inicial da reta ($\vec{A}$) e a posição final ($\vec{F}$);
- Definição do versor unitário orientador da reta: $\hat{n} = \frac{\overrightarrow{AF}}{||\overrightarrow{AF}||}$, em que $\overrightarrow{AF} = \vec{F} - \vec{A}$;
- Definição do vetor com origem na posição atual do robot e que aponta para o extremo inicial da linha: $\overrightarrow{PA} = \vec{A} - \vec{P}$;
- Definição do vetor com origem na posição atual do robot e que aponta para o ponto mais próximo da reta: $\vec{D} = \overrightarrow{AP} - ((\overrightarrow{AP} \cdot \hat{n}) \cdot \hat{n})$;
- Cálculo do módulo do vetor anteriormente definido, retirando-se a distância do veículo à reta: $D = \left\|\overrightarrow{D}\right\|$.

3 *Controlador A* – Aproximação da Linha

O veículo para se aproximar com sucesso do ponto mais próximo da linha deve mover-se na direção e sentido definidos por $\vec{D}$, já calculado na função *LineDistance*. Desta forma, o robot deve orientar-se de acordo com este vetor, descrevendo um movimento de rotação cuja velocidade angular é proporcional ao erro entre o ângulo do robot e o ângulo de $\vec{D}$, cujo designamos por α. A constante de proporcionalidade é dada por k_α. Neste caso, não se considera pertinente que a velocidade angular seja função da distância do robot à linha, pois, devido ao valor significativo assumido por esta variável, a velocidade angular imposta no robot iria desestabilizar a sua trajetória de aproximação, descrevendo um movimento oscilatório, em virtude das perturbações oriundas do termo $k_d \cdot D$, que estariam permanentemente a ser corrigidas pelo termo $k_\alpha \cdot \alpha$. Adicionalmente, ao longo de todo o movimento a velocidade linear do robot é constante e de valor igual à variável *VNOM*.

Com base nisto, obtêm-se os seguintes parâmetros:

$$\omega = k_d \cdot D + k_\alpha \cdot \alpha = k_\alpha \cdot \alpha;\ \ V = VNOM$$

Através de várias simulações de caráter experimental calibraram-se os seguintes valores para as constantes indicadas:

$$k_\alpha = 50;\ VNOM = 10$$

4 *Controlador B* – Seguimento da Linha

Uma vez próximo da linha e tendo em conta que o robot se move numa direção perpendicular a esta, a topologia de controlo é comutada, entrando em vigor um controlador adequado. Neste caso, o robot deve orientar-se de acordo com a direção e sentido do segmento de reta, estabelecidos por $\hat{n}$, definido em *LineDistance*. Analogamente ao raciocínio anterior, a velocidade linear é considerada constante de valor *VNOM*, enquanto que a velocidade angular é função da distância *D* e do erro angular, α, entre a orientação atual do robot e o ângulo de $\hat{n}$. Para o cálculo do ângulo recorre-se à função *atan2* que retorna valores entre $-\pi\ e\ \pi$. Tendo em vista a normalização dos dados, ajustam-se os ângulos de forma a que a sua magnitude se localize entre $0\ e\ 2\pi$.

Um movimento de rotação no sentido dos ponteiros do relógio está associado a uma velocidade angular positiva, enquanto que no sentido contrário esta é negativa. Sendo α dado pela diferença entre o ângulo do robot e o ângulo de $\hat{n}$, sempre que este surgir avançado face ao primeiro, a velocidade angular será negativa e o robot aproxima-se da orientação da linha (fig.1) e, no caso contrário, igualmente. Contudo, sendo a escala entre $0\ e\ 2\pi$, na situação hipotética em que um dos ângulos esteja no primeiro quadrante e o outro no quarto, o valor de α será falacioso e de sinal contrário ao expectável (fig.2), forçando o robot a rodar em sentido contrário ao pretendido.

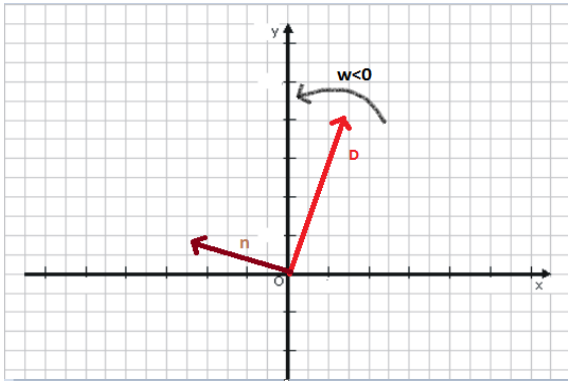


Figura 4.2 - ***Velocidade angular correta***

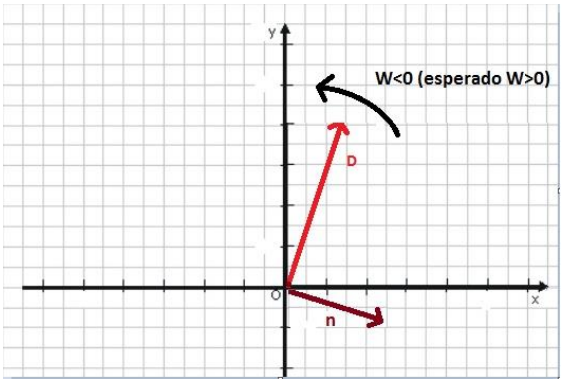


Figura 4.1 - ***Velocidade angular falaciosa (1º e 4º quadrantes)***

Neste caso, a utilização da gama de valores entre $-\pi$ *e* π permite colmatar este efeito e obter o valor desejado para a velocidade angular (pois o ângulo do robot já será superior ao de $\hat{n}$ e, como tal, $\alpha > 0$) e, consequentemente, o sentido de rotação correto do robot. Sendo assim, é necessário fazer esta distinção no momento em que se calcula o desfasamento angular, tendo-se implementado o seguinte trecho de código para satisfazer esse propósito:

```
if (((GetRobotTheta(0)<0) AND (GetRobotTheta(0)>(-pi/2)) AND (ATan2(Mgetv(N,
1, 0),Mgetv(N,0,0))>0) AND (ATan2(Mgetv(N, 1, 0),Mgetv(N,0,0))<(pi/2))) OR
((GetRobotTheta(0)>0) AND (GetRobotTheta(0)<(pi/2)) AND (ATan2(Mgetv(N, 1,
0),Mgetv(N,0,0))<0) AND (ATan2(Mgetv(N, 1, 0),Mgetv(N,0,0))>(-pi/2)))) then
begin
    a := GetRobotTheta(0) - ATan2(Mgetv(N, 1, 0),Mgetv(N,0,0));
   end
  else
    a := GetRobotTheta(0)+pi*(1-sign(GetRobotTheta(0))) - ATan2(Mgetv(N, 1,
0),Mgetv(N,0,0)) - pi*(1-sign(ATan2(Mgetv(N, 1, 0),Mgetv(N,0,0))));
```

5 *FollowLine* – Procedimento de Supervisão

Para garantir a comutação entre os diferentes controladores é necessário implementar um procedimento de supervisão. Esta rotina monitoriza a distância a que o robot se encontra da linha e dita a transição entre os controladores no momento adequado. Assim, quando o robot se desloca em direção à linha sob ação do *controlador A* e atinge a distância *DMAX* comuta para o *controlador B*, iniciando o seguimento da linha. Para voltar a transitar para o estado anterior a distância deve ser superior a *DMAX+DHEST*. Este parâmetro adicional tem como propósito definir uma janela de histerese para evitar a oscilação entre os dois estados no limiar de transição. Por fim, quando o robot atinge a posição final com um erro inferior a *TolPosFinal* o seu movimento cessa.

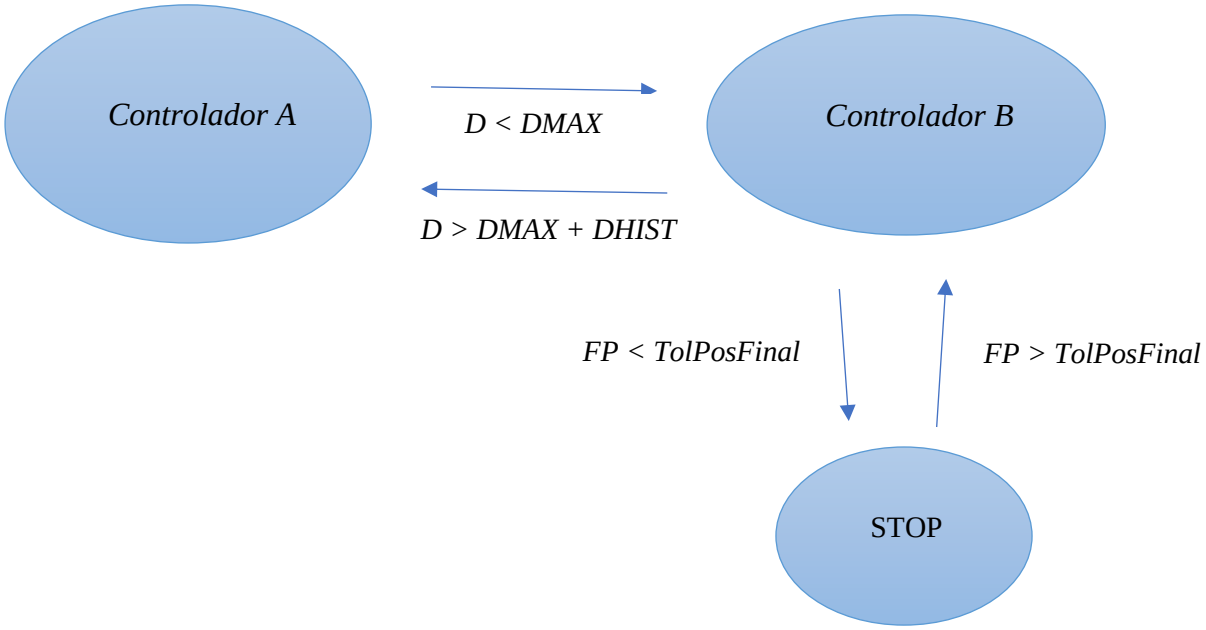


Figura 3 – *Diagrama de Estados*

Após vários ensaios experimentais, sintonizaram-se os seguintes parâmetros:

$$DMAX = 0.1;\ DHIST = 0.1;\ \ TolPosFinal = 0.1$$

6 Demonstração de resultados

Para a demonstração dos resultados obtidos em relação a este procedimento experimental, teve-se em consideração um caso que se torna crítico para o robot, que depende essencialmente do lado que está o robot em relação à reta definida por $\hat{n}$ e saber para que lado é que deve rodar, por forma a chegar ao ponto final desejado. Portanto, para efeitos de simulação, teste e demonstração de resultados, considerou-se os seguintes pontos.

$$p_{i1} = (1,0)\,; p_{f1} = (0,1)$$

$$p_{i2} = (0,-1)\,; p_{f2} = (-1,0)$$

Para outros testes também efetuados, revelou-se que é insignificante demonstrar a experiência com mais pontos, uma vez que o comportamento seria semelhante.

Começou-se, então, por analisar o primeiro caso de estudo com os pontos inicial e final, respetivamente, em (1,0) e (0,1).

Da figura 6.1, observa-se que o robot é capaz de concluir a missão para a qual cuja máquina de estados foi desenhada.

No entanto, observando atentamente a figura 6.1 b e a figura 6.3 a, verifica-se que o robot não se aproxima da reta perpendicularmente tal como era desejado (figura 6.2). Esta particularidade resulta da distância inicial do robô à reta e da forma como foi projetado o algoritmo de aproximação a esta, que o faz de uma forma dinâmica, onde a aproximação por um ângulo de

noventa graus não é feita no imediato (figura 6.3), sendo apenas visível em percursos de aproximação mais longos.

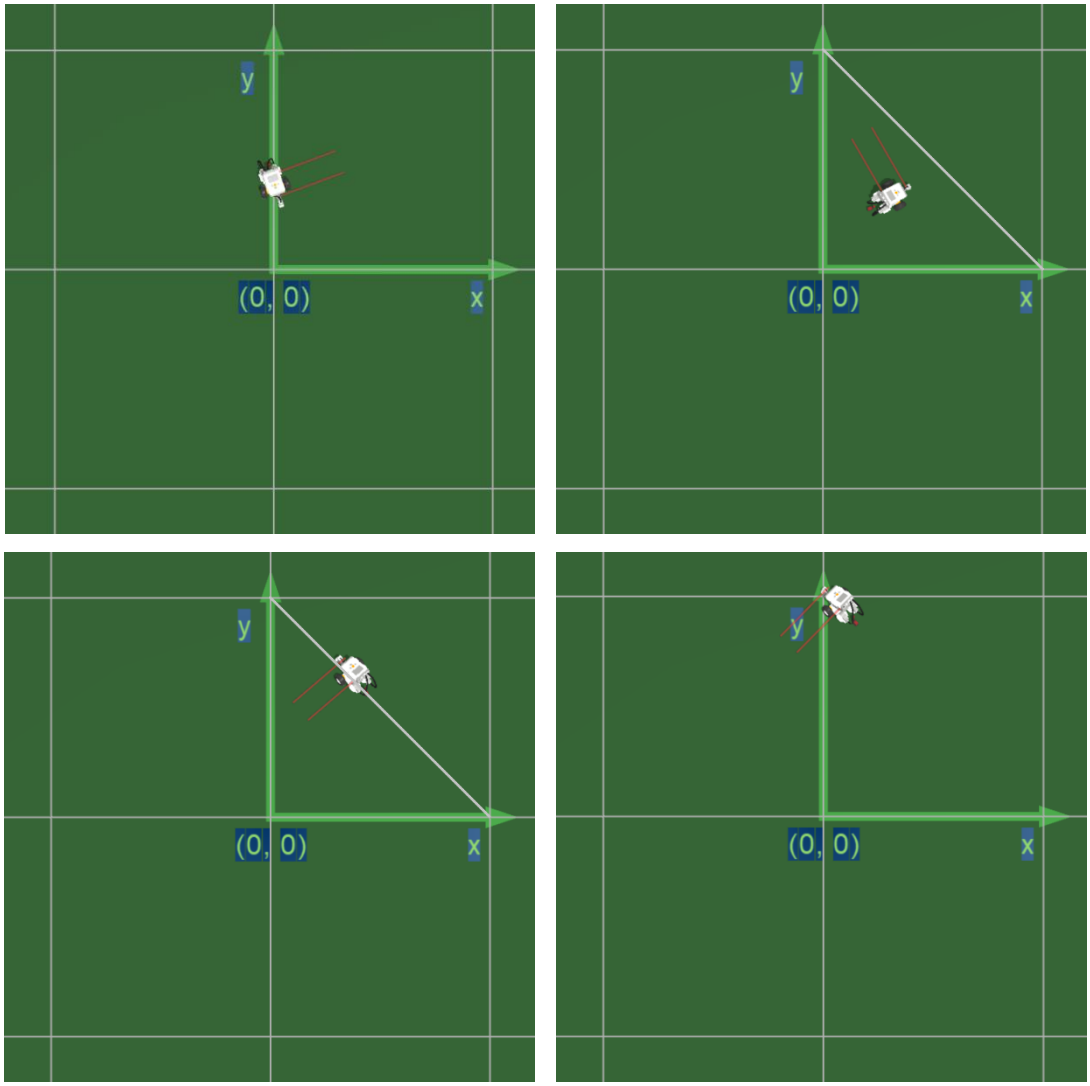


Figura 6.1 Trajetória do robot desde a posição inicial até à posição final. (a) Robot na posição inicial. (b) Robot sob o efeito do controlador A, responsável por o aproximar da reta que se pretende seguir. (c) Robot sob o efeito do controlador B, que o faz seguir a reta definida pelo vetor $\hat{n} = (-1, 1)$. (d) Robot na posição final.

(Considere-se as imagens numeradas da direita para a esquerda, de cima para baixo.)

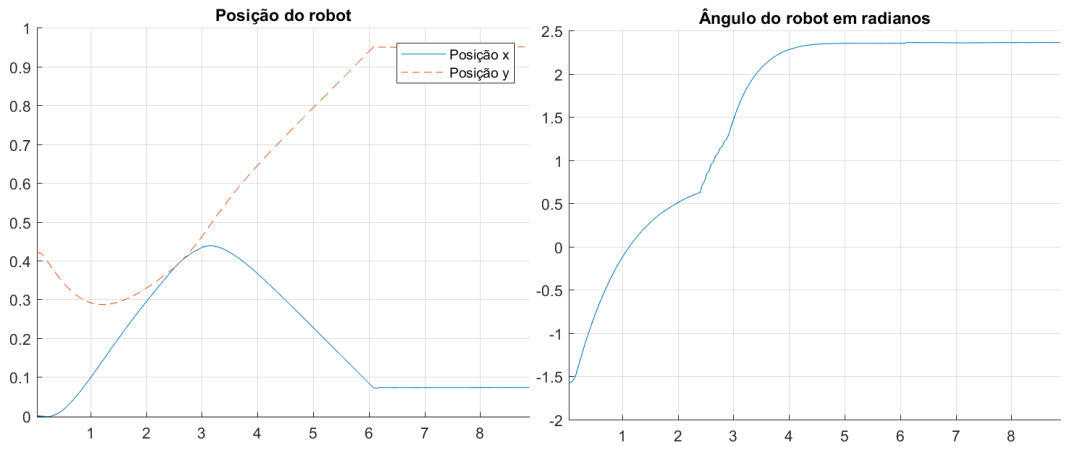


Figura 6.2 Posição (a) e ângulo (b) do robot ao longo do tempo

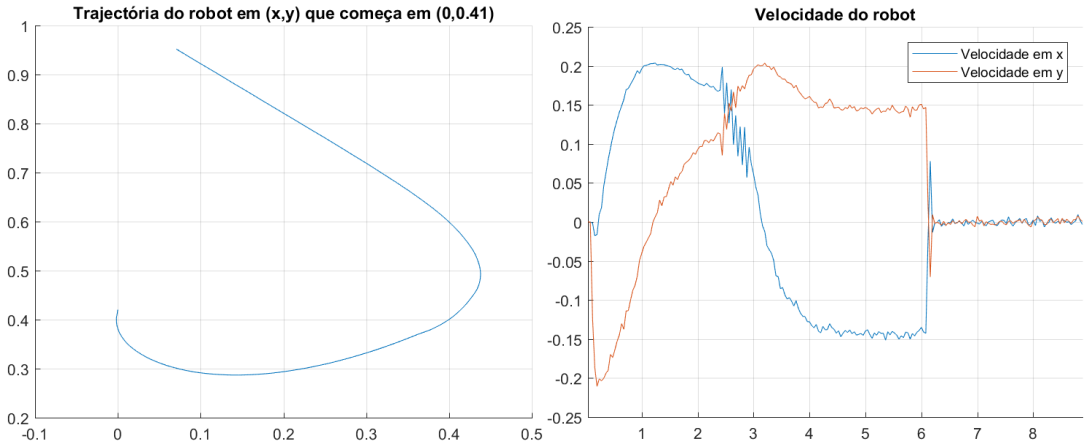


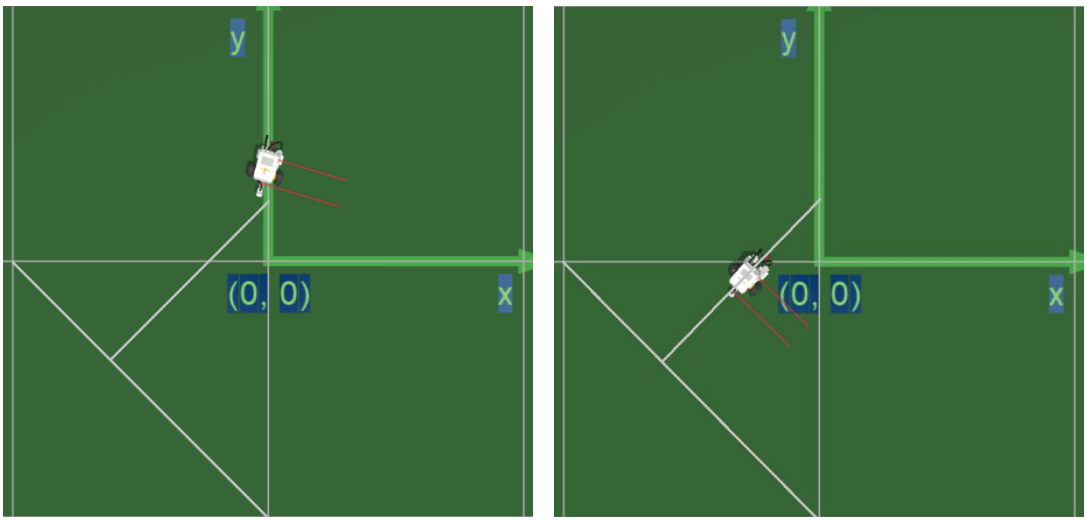
Figura 6.3 Trajetória (a) e Velocidade (b) do robot

Da observação da figura 6.3, conseguimos concluir que não se obteve uma máquina de estados perfeitamente calibrada, dado que a transição entre o controlador A e o controlador B é feita de uma forma ruidosa (intervalos de tempo entre os 2 e 3 segundos).

Para um segundo caso de estudo, considera-se agora os pontos inicial e final do segmento de reta, respetivamente, (0, -1) e (-1, 0), procurando, assim comprovar que o robot é capaz de distinguir, perfeitamente, os lados para onde deve rodar, no sentido de atingir os pontos finais desejados (figura 6.4).

Contrariamente ao caso anterior, verifica-se agora que o robot se direciona para a reta que pretende seguir numa trajetória perpendicular (figura 6.4 b e figura 6.6 a), pois a distância do ponto inicial do robot à reta é suficientemente grande. No entanto, todas as outras propriedades se mantêm, apresentando-se alguma robustez na solução encontrada.

Na figura 6.5 b observa-se um salto no ângulo do robot que é referente à limitação do simulador que representa todos os ângulos entre $-\pi$ e π. A solução encontrada para ultrapassar este problema foi explicitada anteriormente na secção 4 referente ao controlador responsável por fazer o robot seguir a linha.



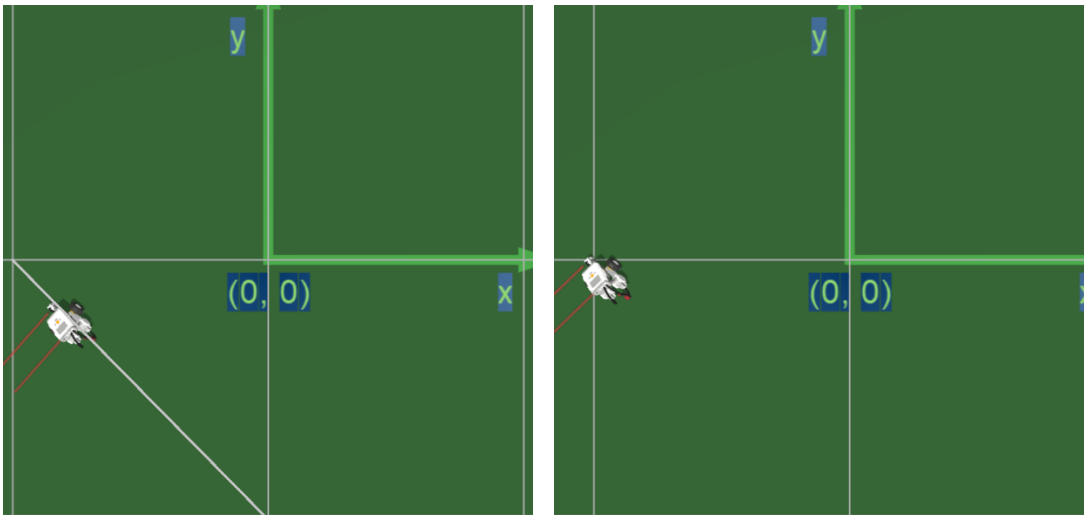


Figura 6.4 Trajetória do robot desde a posição inicial até à posição final. (a) Robot na posição inicial. (b) Robot sob o efeito do controlador A, responsável por o aproximar da reta que se pretende seguir. (c) Robot sob o efeito do controlador B, que o faz seguir a reta definida pelo vetor $\hat{n} = (-1, 1)$. (d) Robot na posição final.

(Considere-se as imagens numeradas da direita para a esquerda, de cima para baixo.)

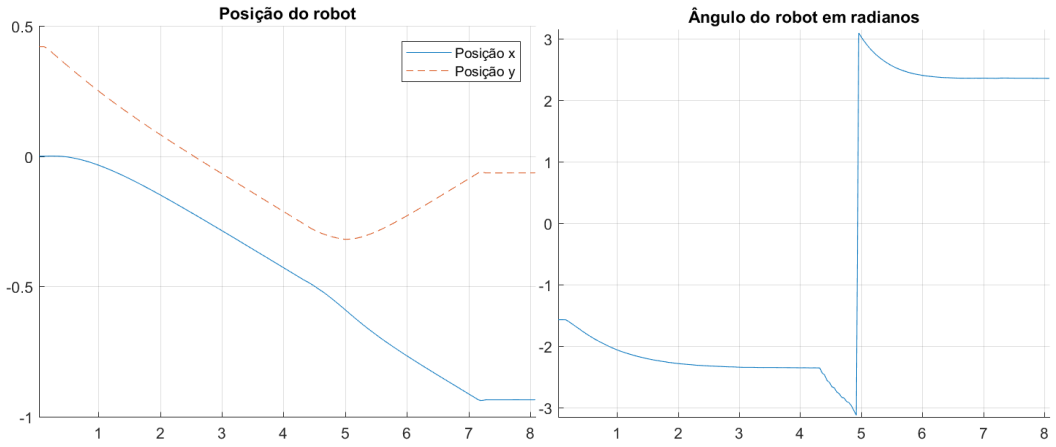


Figura 6.5 Posição (a) e ângulo (b) do robot ao longo do tempo

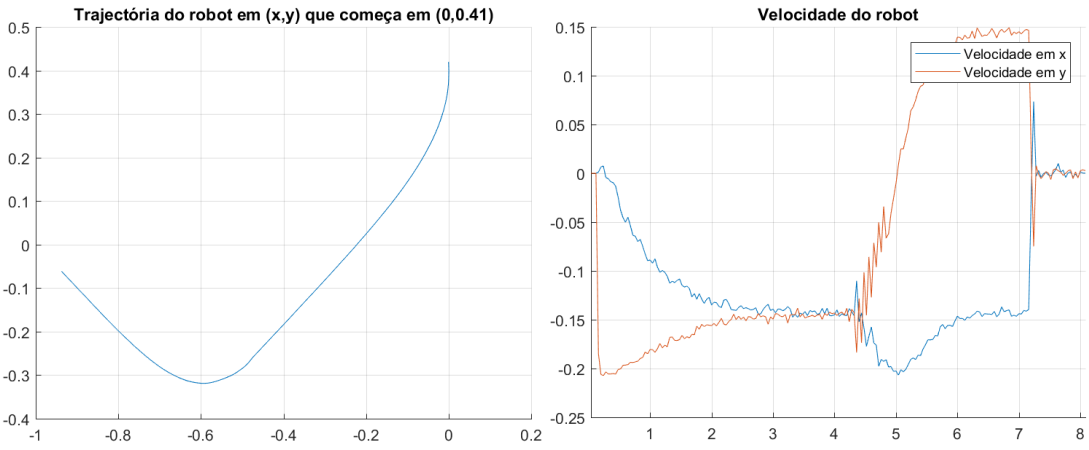


Figura 6.6 Trajetória (a) e Velocidade (b) do robot

7 Conclusão

Verifica-se que o algoritmo seguidor de linha é um algoritmo simples de implementar, mas que apresenta alguma complexidade quando pretendemos calibrar as velocidades nominais e os ganhos de compensação dos erros que o robot vai tendo ao longo das trajetórias.

Este algoritmo é, por vezes, bastante útil quando o principal objectivo é desenvolver um robot que seja capaz de seguir linhas que são inexistentes baseando-se na sua visão pelo mundo. Como esta visão é falaciosa, este também fornece ao robot a capacidade de se recuperar quando encontra marcos que o ajudam a relocalizar-se.