Trabalho Prático 1 - Implementação de um controlador para seguir uma curva em forma de oito e do algoritmo *Tangent Bug*

Rafael Fernandes Gonçalves da Silva
Departamento de Engenharia Elétrica
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
rafaelfgs@ufmg.br

I. Motivação

O presente trabalho apresenta métodos de implementação para o controle de robôs em duas aplicações.

Primeiramente, foi determinada a tarefa para desenvolver um controlador utilizando uma trajetória em forma de oito. Para o seguimento da trajetória, deve-se utilizar um robô com acionamento diferencial, o qual deve convergir para a curva e ser capaz de seguir eternamente a mesma. Os parâmetros da curva devem ser entradas do programa, o qual deve funcionar no simulador *StageROS*.

A segunda aplicação envolveu a implementação do método *tangent bug*, também utilizando um robô com acionamento diferencial.O robô deve estar equipado com um sensor laser do tipo *Hokuyo*, de forma a identificar os obstáculos no mapa. O processo deve funcionar no simulador *StageROS*, com os parâmetros da posição final como entradas do programa.

II. Metodologia

Conforme explicado na seção anterior, foi utlizado o simulador *StageROS* para a resolução dos problemas propostos. As implementações de cada parte do problema estão descritas a seguir.

A. Seguidor de Curva

O problema presenta consiste em uma implementação de um controlador que faça um robô seguir uma trajetória em forma de oito. Para isso, foi desenvolvido uma sequência de comandos, mostrada no pseudo-código a seguir.

BEGIN
INITIALISE current node, subscribers and publishers;
EVALUATE current time;
REPEAT UNTIL (forever) DO
1 EVALUATE current time;
2 SET curve coordinates and velocities;
3 EVALUATE control signal;
4 EVALUATE linear and angular velocities;
5 PUBLISH velocities;
OD
END

As equações da curva utilizadas no passo 2, foram definidas da seguinte forma:

$$\begin{aligned} x_d &= r_x.\sin(\omega.t) + c_x \\ y_d &= r_y.\sin(2.\omega.t) + c_y \\ \dot{x_d} &= r_x.\omega.\cos(\omega.t) \\ \dot{y_d} &= 2.r_y.\omega.\cos(2.\omega.t) \end{aligned} \tag{1}$$

onde ω é calculado como $2.\pi.freq$, sendo $freq$ a frequência de giro da curva (inverso do período), e as constantes c_y, c_x, r_x e r_y, são parâmetros de entrada representando o centro e o raio em x e y da curva.

O sinal de controle utilizado foi obtido com as equacões:

$$\begin{aligned} U_x &= k.(x_d - x) + \dot{x_d} \\ U_y &= k.(y_d - y) + \dot{y_d} \end{aligned} \tag{2}$$

onde x e y é a posição atual do robô e k é o parâmetro de convergência, informado no início do programa.

Os valores das velocidades linear e angular foram obtidos através do *feedback linearization*, dado pela equação:

$$\begin{bmatrix} V_x \\ W_z \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \frac{-\sin(\theta)}{d} & \frac{\cos(\theta)}{d} \end{bmatrix} \cdot \begin{bmatrix} U_x \\ U_y \end{bmatrix} \tag{3}$$

onde θ é a orientação atual do robô e d é o deslocamento do ponto de controle, informado no início do programa.

B. Tangent Bug

Para o problema de desvio de obstáculos proposto, foi implementado um algoritmo de *tangent bug* da seguinte forma:

BEGIN
INITIALISE current node, subscribers and publishers;
REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
1 SET a virtual position;
2 EVALUATE control signal;
3 EVALUATE linear and angular velocities;
4 PUBLISH velocities;
OD
END

O primero passo dentro da repetição pode ser dividido em três diferentes modos: ir para o objetivo final; ir para o ponto de descontinuidade; e contorno do obstáculo. O sinal de controle e o *feedback linearization* aqui utilizados foram semelhantes ao da subseção anterior, dados pelas equações (2) e (3). O critério de parada utilizado foi dado por uma variável de modo de operação secundário, a qual recebia valores específicos caso o robô chegue ao objetivo ou caso ele contorne todo o obstáculo sem avistar o alvo.

A forma com que o *tangent bug* foi aqui implementado depende exclusivamente do escolha do ponto virtual, o qual é usado para controlar o robô de acordo com os três modos citados no parágrafo anterior, descritos a seguir.

1) Ir para o Objetivo Final: O primeiro modo de operação ocorre quando não há obstáculos em torno da linha entre o alvo e a posição atual do robô. Este é o modo mais simples de operação, visto que o ponto virtual é o próprio alvo e o robô é controlado até esse ponto. Nesse ponto, é também verificado se o robô não está no terceiro modo de operação, porém isso será melhor explicado quando necessário.

Foram utilizados dois parâmetros, d_{max} e ϕ_0, sendo o primeiro a distância mínima que o robô observa como obstáculo (foi utilizado como padrão um valor de $2.d_2$, sendo esta uma contante explicada no próximo modo) e o segundo o ângulo que determina a faixa de varredura (para evitar trajetórias muito próximas aos obstáculos). Esse modo está demonstrado na Figura 1.

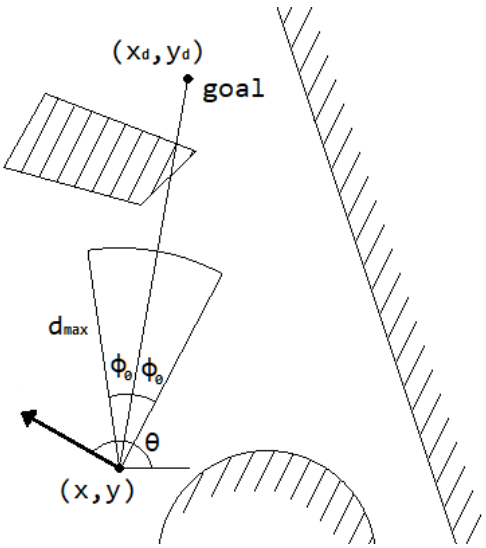

Figura 1. Modo de ir ao objetivo final.

2) Ir para o Ponto de Descontinuidade: Caso haja algum obstáculo entre o robô e o alvo, é realizada uma rotina para verificar qual ponto do obstáculo possui a menor distância até o alvo. Caso essa distância diminua, o modo de operação passa a ser ir para o ponto de descontinuidade, com o ponto virtual próximo ao ponto de descontinuidade, conforme mostrado nas Figuras 2 e 3.

O ponto virtual utilizado está localizado a uma distância d_2 perpendicular a reta entre o robô e o ponto do obstáculo de menor distância até o alvo. Com os valores de θ, ϕ e d_{obst} conhecidos e d_2 determinado, o ponto de menor distância e o ponto virtual podem ser encontrados por:

$$\begin{aligned} x_{min} &= x + d_{obst}.\cos(\theta + \phi) \\ y_{min} &= y + d_{obst}.\sin(\theta + \phi) \\ x_d &= x_{min} + d_2.\cos(\theta + \phi \pm \tfrac{\pi}{2}) \\ y_d &= y_{min} + d_2.\sin(\theta + \phi \pm \tfrac{\pi}{2}) \end{aligned} \tag{4}$$

onde o sinal $\pm$ dos cálculos do ponto virtual depende exclusivamente se o obstáculo está à direita ou à esquerda do robô. Este problema pode ser contornado através de uma estrutura condicional, verificando o sinal do ângulo ϕ, dado pelo laser do robô.

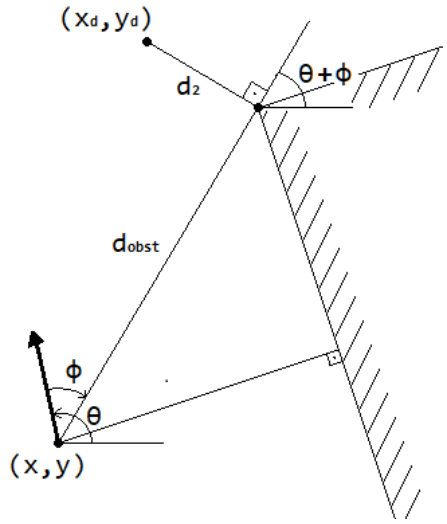

Figura 2. Modo de ir para o ponto de descontinuidade, com o obstáculo à direita.

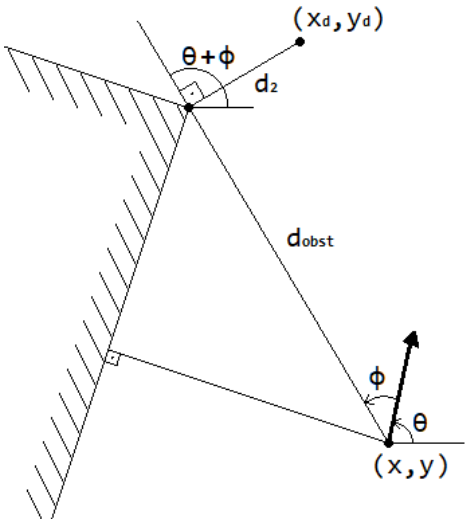

Figura 3. Modo de ir para o ponto de descontinuidade, com o obstáculo à esquerda.

3) Contorno do Obstáculo: Caso não haja atualização do ponto de mínimo (obviamente para menor), o modo de operação se altera para contornar o obstáculo. De uma forma bastante similar ao modo anterior, o ponto virtual é criado estando a uma distância d_2 do obstáculo e a uma distância d_1 do robô, de forma que haja um desvio tangente ao obstáculo. O esquema desse modo está representado pelas Figuras 4 e 5.

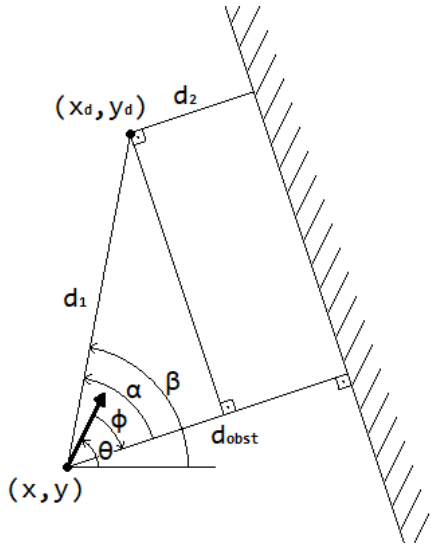

Figura 4. Modo de contorno simples do obstáculo, com este à direita.

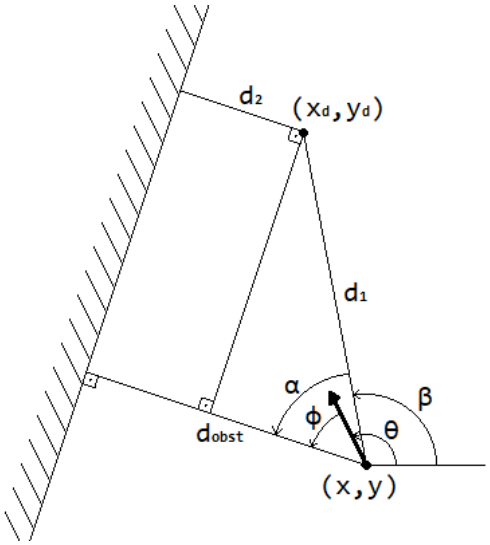

Figura 5. Modo de contorno simples do obstáculo, com este à esquerda.

Com os valores de θ, ϕ e d_{obst} conhecidos e d_1 e d_2 determinados, os valores de α, β e do ponto virtual podem ser encontrados por:

$$\begin{aligned} \alpha &= \arccos\left(\tfrac{d_{obst}-d_2}{d_1}\right) \\ \beta &= \theta + \phi \pm \alpha \\ x_d &= x + d_1.\cos(\beta) \\ y_d &= y + d_1.\sin(\beta) \end{aligned} \tag{5}$$

Assim como no caso anterior, o sinal $\pm$ do cálculo do α depende se o obstáculo está à direita ou à esquerda do robô e também é determinado através do sinal do ângulo ϕ.

A determinação do sinal de α através do ângulo do obstáculo pode gerar um problema caso o robô chegue há uma quina fechada com um obstáculo à frente, conforme mostrado nas Figuras 6 e 7. Nesse modo é ainda verificado se há um obstáculo à frente do robô e, caso haja, o ponto virtual é dado para a posição onde não há obstáculo.

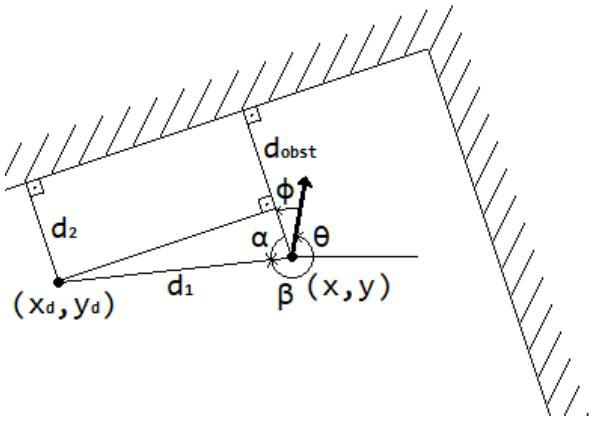

Figura 6. Modo de contorno com um obstáculo à direita e à frente.

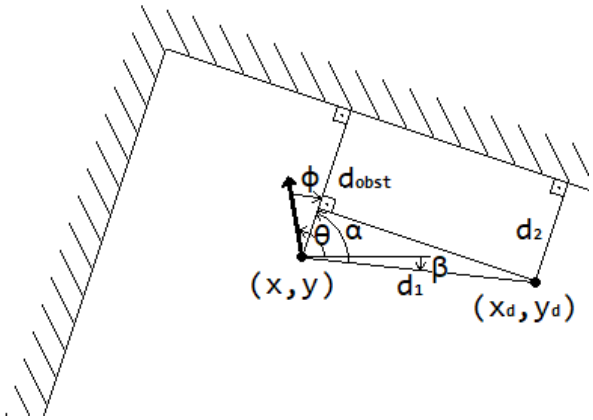

Figura 7. Modo de contorno com um obstáculo à esquerda e à frente.

As equações para esse esquema é similar às obtidos no caso anterior, sendo que nesse caso o sinal de α é positivo para um obstáculo à direita e negativo à esquerda.

III. Resultados e Discussão

Foram feitas simulações do código implementado no *StageROS*. Os resultados obtidos com o seguidor de curva e o *tangente bug* estão mostrados a seguir. Por padrão os valores das contantes foram fixadas da seguinte forma: $d = 0.1$, $k = 1.0$, $d_1 = 1.0$, $d_2 = 3.0$ e $\phi_0 = \frac{\pi}{4}$.

A. *Seguidor de Curva*

No seguidor de curva em forma de oito, os parâmetros de entrada podem ser dados pelo usuário. Para o caso da curva centrada na origem, com raio igual à 8 em x e 4 em y e um tempo de 100 segundos, foi obtido o seguinte resultado.

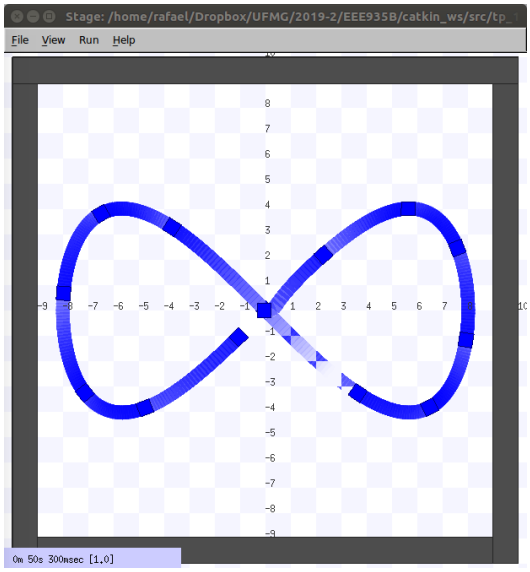

Figura 8. Trajetória observada do robô seguindo a Leminiscata.

B. *Tangent Bug*

O algoritmo do *tangent bug* foi testado em diferentes mapas, mostrados a seguir.

1) Mapa com Obstáculos: Utilizando o mapa disponibilizado no exemplo e iniciando na posição $x = 27$ e $y = -27$ com o alvo em $x_{goal} = -24$ e $y_{goal} = 6$, foi obtido o seguinte resultado.

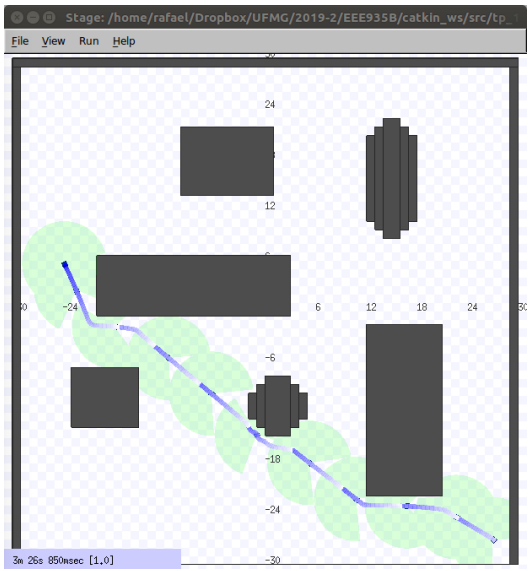

Figura 9. Trajetória observada do robô no mapa com obstáculos.

2) Mapa de uma Sala: Para o mapa de simulação de uma sala, criado para teste, o robô foi posicionado dentro da sala, no ponto $x = 0$ e y=0 com o alvo em $x = 12$ e $y = 0$. O resultado obtido foi o seguinte.

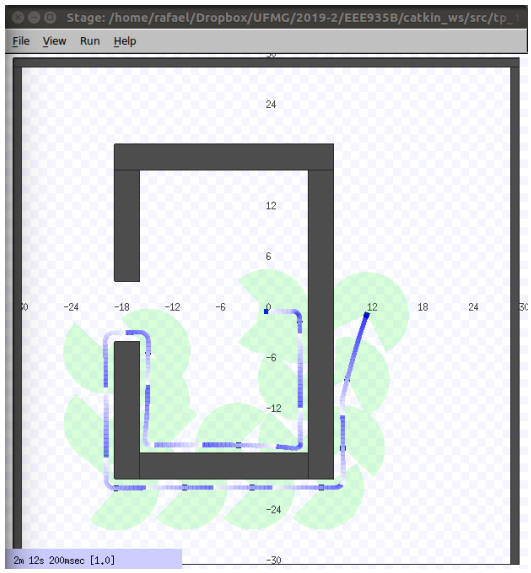

Figura 10. Trajetória observada do robô no mapa de uma sala.

3) Mapa de um Labirinto: Foi tambem criado um mapa mais complexo para uma verificação melhor do comportamento do algoritmo nos pontos de quina. Partindo do ponto $x = -25$ e $y = 25$ em direção ao alvo em $x = 25$ e $y = -25$, foi obtido o seguinte resultado.

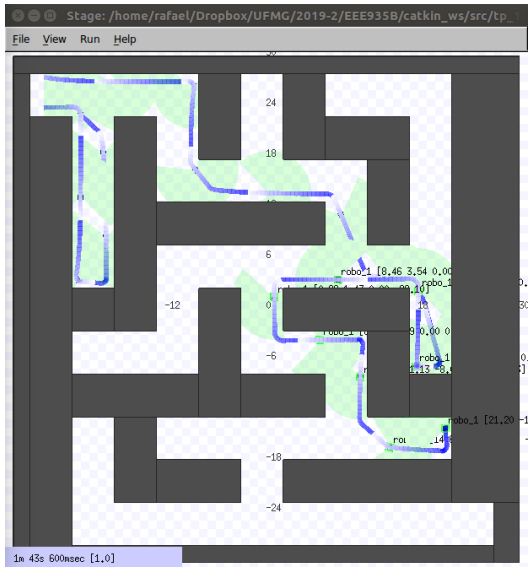

Figura 11. Trajetória observada do robô no labirinto.

Nesse caso a simulação não ocorreu até o encontro o ponto alvo. Foi observado que o robô iria fazer o contorno de todo o labirinto, o que demandaria tempo e iria atrapalhar a visualização do resultado, no entanto esse comportamento foi natural do algoritmo para as condições dadas.

4) Mapa com Osbtáculos sem Solução: Por fim, foi realizada uma simulação sem solução para o algoritmo, colocando o alvo no interior de um obstáculo. Nesse caso o programa deveria se encerrar e avisa que não houve solução. Partindo do ponto $x = 27$ e $y = -27$ com o alvo em $x_{goal} = -12$ e $y_{goal} = 0$, o robô percorreu a seguinte trajetória.

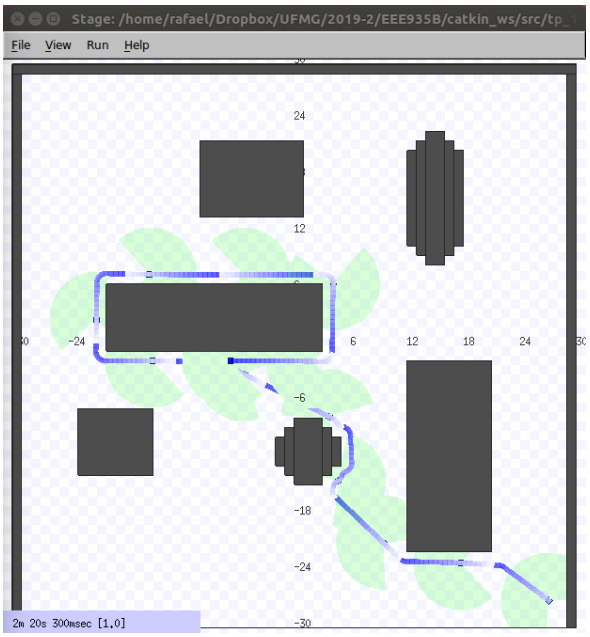

Figura 12. Trajetória observada do robô no mapa com obstáculos sem solução.

IV. Conclusão

O presente trabalho apresentou implementações para um método de seguidor de curva e um outro da forma *tangent bug*. Foram utilizados métodos analítcos para a resolução dos problemas, onde o ponto virtual criado para o controle do robô foi obtido exclusivamente de forma geométrica.

Os resultados apresentados foram bem satisfatórios, apresentando problemas somente em situações muito específicas. Uma dessas situações foi para o caso de o robô chegar em um ponto com obstáculos à frente, à esquerda e à direita à uma distância inferior a d_1, onde o robô apresentou um comportamento anormal e acabou por colidir com o obstáculo.

Outro ponto a ser observado é o problema relacionado à discretização dos ângulos do laser. Como a distância dos obstáculos é obtida por feixes com ângulos discretizados, uma alteração na orientação do robô (sem alterar a posição) é capaz de alterar a distância do obstáculo, mesmo que de forma milimétrica. Para contornar esse problema, foi utilizado um arredondamento dessa distância, para evitar atualizações milimétricas na função de cálculo da menor distância entre o alvo e os pontos do obstáculo. Porém isso resultou em um alto chaveamento entre os modos de descontinuidade e de contorno, já que em alguns *frames* o ponto de mínimo não conseguia se atualizar. No entanto isso pode ser percebido na trajatória do robô, já que em ambos os modos, o cálculo do ponto virtual são obtidos de forma bastante similar.