Trabalho Prático 1

Arthur da Costa Vangasse * **Gabriel Gomes** **
João Felipe Ribeiro Baião ***

* *Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, MG, (e-mail: vangasse@ufmg.br).*
** *Departamento de Engenharia Elétrica, Universidade Federal de Minas Gerais, MG, (e-mail: ggomes1015@ufmg.br).*
*** *Departamento de Mecânica, Universidade Federal de Minas Gerais, MG, (e-mail: baiaojfr@ufmg.br).*

Resumo:
Este trabalho apresenta o desenvolvimento do primeiro trabalho prático da disciplina de Planejamento de Movimento de Robôs, do Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal de Minas Gerais ministrada no semestre 2022.2. Neste, são utilizados os pacotes desenvolvidos para o robô móvel Turtlebot3 em Ros2 para simulação física e validação dos algoritmos implementados. Os algoritmos compreendidos foram escritos em Python, sendo eles: Tangent Bug, Seguimento de Curva por Robô Diferencial, Campos Potenciais e Wave-Front.

1. TANGENT BUG

O algoritmo Tangent Bug foi implementado conforme indica o diagrama da Figura 1, onde as setas verdes indicam que a condição está sendo atendida e as setas vermelhas o contrário. A cada iteração da simulação, avalia-se o estado do robô de acordo com os dados de posição e de detecção do laser.

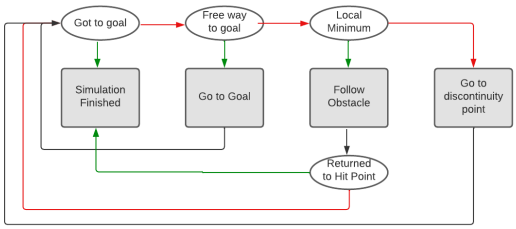

Figura 1. Esquema de Tomada de Decisão do Tangent Bug

A condição de ”objetivo alcançado”é verdadeira quando a distância calculada entre o objetivo e a posição atual do robô é menor que um raio pré-estabelecido. Nessa situação, a simulação é encerrada com aviso que o objetivo foi alcançado.

A segunda condição, ”caminho livre”, avalia o feixe de 12 lasers em torno da direção do objetivo, no referencial do robô. Caso todos esses lasers indicarem uma distância maior que a distância do robô ao obstáculo acrescida do raio do robô, considera-se que a condição foi cumprida. Isso permite que, quando for possível, o robô vá em direção ao objetivo mesmo que esse se encontre entre o robô e um obstáculo.

Caso as condições anteriores não sejam cumpridas, toda a lista de detecção do laser é avaliada para presença de mudanças bruscas de distância percebida entre lasers vizinhos. As mudanças bruscas observadas são armazenadas como pontos de descontinuidade. Esses pontos são obtidos a partir dos dados dos lasers e da transformação homogênea do referencial do robô em relação ao mundo.

A Figura 2 indica com círculos todos os pontos de descontinuidades armazenados em determinado momento de simulação, bem como, o objetivo como uma cruz verde. Esses pontos são julgados quanto proximidade do objetivo segundo a seguinte heurística: somatório da distância entre o robô e o ponto de descontinuidade e desse ponto ao objetivo. O ponto que apresentar menor distância é selecionado (no caso da Figura 2, trata-se do círculo verde).

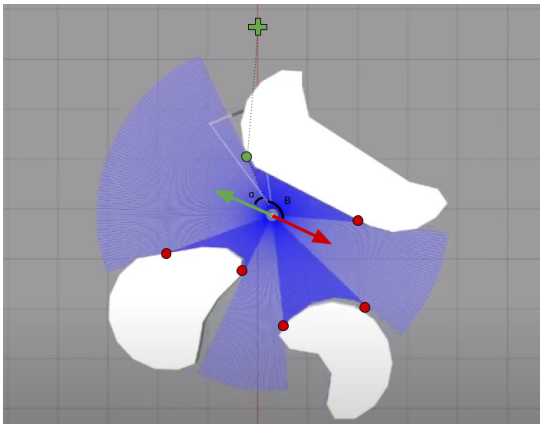

Figura 2. Exemplo de Estado do Robô na Simulação

O robô seguirá o ponto de interesse selecionado de forma análoga ao objetivo, a não ser que esteja ativa a terceira condição: ”mínimo local”. Portanto, ao seguir o ponto de interesse, avalia-se a detecção de um obstáculo a uma distância menor que $\epsilon + \delta$ em um feixe de lasers frontal ao robô. Nesse caso, há iminência de colisão e é ativado o estado de contorno de obstáculo. Isso identifica mínimos

⋆ Reconhecimento do suporte financeiro deve vir nesta nota de rodapé.

locais, pois, nesses casos, o robô aproxima-se indefinidamente de um obstáculo.

Para contornar o obstáculo, é necessário definir o sentido (horário ou anti-horário) desejado. Para isso, utiliza-se os ângulos α e β segundo a Figura 2. Esses ângulos comparam o vetor direção do ponto de interesse no referencial do robô e os vetores tangentes ao obstáculo no sentido horário e anti-horário respectivamente. Caso o ângulo α seja menor que o β, o sentido horário é selecionado e vice-versa.

Para os estados de ir para o objetivo ou ir para o ponto de interesse, é aplicado um ganho proporcional na distância atual do robô e o ponto desejado, obtendo um vetor velocidade que é aplicado por feedback linearization no robô. É estabelecida uma velocidade linear mínima de modo que ao se aproximar do ponto desejado a velocidade não fique muito reduzida (por se tratar de um ganho somente proporcional).

Para o estado de contornar obstáculo, adapta-se o método de campos vetoriais de Rezende et al. (2021). Esse método é descrito na Seção 2. Para a aplicação de contornar obstáculos, a curva plana é uma curva que compreende o obstáculo expandido de um raio ϵ conforme a Figura 3.

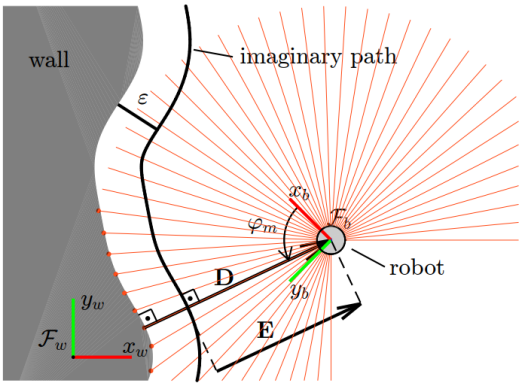

Figura 3. Configuração de Campo Vetorial para Contornar Obstáculo

Ao entrar no estado de contornar obstáculo, a quarta condição, "retorno ao ponto de encontro", passa a ser medida. No momento em que o estado é ativado, armazena-se a posição do robô como ponto de encontro. A todo momento, é avaliado se o robô continua seguindo o mesmo obstáculo, isso é feito por meio de um acumulador que zera caso o robô se afaste do obstáculo ou caso o obstáculo mais próximo altere-se bruscamente. A cada iteração no estado de seguir obstáculo o acumulador é acrescido de um. Depois de passadas várias iterações ainda no estado de contornar obstáculo, passa-se a avaliar se a posição atual é a mesma do "ponto de encontro", caso positivo, a simulação é encerrada com aviso de que o objetivo não pode ser atingido. O sentido de seguimento do obstáculo (horário ou anti-horário) também é decidido apenas caso o acumulador seja zerado.

2. SEGUIMENTO DE CURVA PLANA

Em Rezende et al. (2021) propõe-se uma estratégia para gerar campos vetoriais em dimensão-n que convergem para uma curva desejada, que pode variar no tempo. O campo **F** proposto, para o caso em que a curva $C(s)$ é invariante no tempo, pode ser descrito como:

$$\mathbf{F} = \eta(\boldsymbol{p}) \cdot \left[-\hat{G}(\boldsymbol{p}) \frac{\mathbf{D}(\boldsymbol{p})}{D(\boldsymbol{p})} + \hat{H}(\boldsymbol{p}) \mathbf{T}(\boldsymbol{p}) \right], \tag{1}$$

em que $\eta(\boldsymbol{p})$ é definido de maneira que a norma de **F** seja igual ao valor de uma velocidade desejada para seguir o campo vetorial. $D(\boldsymbol{p})$ é a menor distância entre a posição do robô ($\boldsymbol{p}$) e a curva $C(s)$. De maneira que, $\mathbf{D} = \boldsymbol{p} - \mathbf{c_{min}}$ representa o vetor distância entre o robô e o ponto da curva mais próximo a ele (c_{min}). Conforme Rezende et al. (2021) sugere-se, $\hat{G} = 2\pi \cdot \arctan(k_f D(\boldsymbol{p}))$ e $\hat{H} = \sqrt{1 - \hat{G}^2}$ garantem convergência assintótica à curva, sendo $k_f > 0$.

Foram escolhidos $k_f = 3$ e $\eta(\boldsymbol{p}) = 0.7$. Considerando a orientação da curva para a computação de $\boldsymbol{T}$, interseções da curva paramétrica não ocasionaram comportamentos impeditivos.

3. CAMPO POTENCIAL

Uma forma usada para realizar a exploração do espaço livre de maneira incremental é através de funções potenciais. Nesse método, o robô é tratado como uma partícula que se move através de um campo vetorial gradiente de uma posição inicial até chegar em uma posição final, enquanto desvia de obstáculos. A posição final atrai o robô enquanto os obstáculos o repelem, a função potencial U é então construída como a soma do potencial atrativo e repulsivo, Equação 2. O gradiente ∇U aponta na direção em que U aumenta localmente, dessa forma, o gradiente descendente $-\nabla U$ pode ser usado para navegar pelo espaço até uma determinada posição Choset et al. (2005).

$$U = U_{att} + U_{rep} \tag{2}$$

O deslocamento do robô de um ponto inicial até um determinado ponto é dado pelo potencial atrativo que é imposto pela posição que se deseja chegar, o robô utiliza o gradiente atrativo ∇U_{att} para navegar até esse objetivo. Neste projeto, o gradiente atrativo foi implementado de três formas diferentes: cônico, quadrático e uma associação dos dois. O gradiente cônico possui uma descontinuidade quando o robô atinge o objetivo, a forma encontrada para resolver esse problema é considerando a distância até o objetivo no lugar da norma do gradiente.

A detecção de obstáculos é realizada com o auxílio do laser, que é capaz de identificar obstáculos em volta do robô. Na implementação, o robô identifica apenas obstáculos que estejam situados na sua parte frontal até uma determinada distância. O processo de reconhecimento de um obstáculo considera apenas um ponto que equivale a menor distância entre ele e o robô. A fim de aumentar o número de obstáculos que podem ser reconhecidos de uma vez, a parte frontal foi decomposta em cinco regiões, Fig. 4, que são usadas para identificação simultânea de obstáculos.

O gradiente repulsivo ∇U_{rep} que cada obstáculo impõe no robô é calculado através do ponto do obstáculo que indica a menor distância entre eles. As coordenadas desse ponto não podem ser identificadas apenas com os dados fornecidos pelo laser, distância e ângulo, porque estão relacionados apenas ao sistema de coordenados do robô

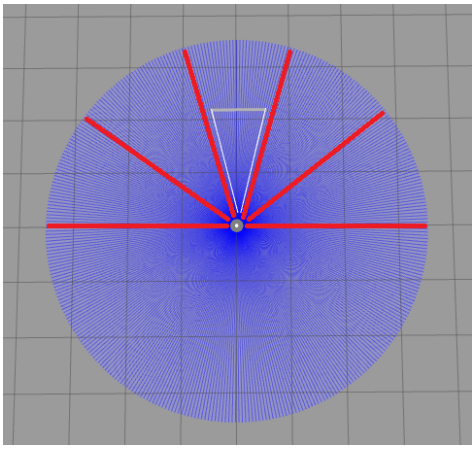

Figura 4. Divisão da parte frontal do laser em 5 regiões.

e não ao sistema de coordenadas inercial. Por este motivo, uma transformada homogênea, Equação 3, foi utilizada para encontrar as coordenadas desse ponto com base no referencial inercial. A posição e a orientação do robô, $p = (p_x, p_y, p_z)^T$ e ψ, foram obtidas pela leitura do tópico /odom. Desse modo, o ponto p_{ro} de um obstáculo determinado pelo laser do robô pode ser representado por um ponto p_{io} no referencial inercial e, dessa forma, ser usado para o cálculo do gradiente repulsivo.

$$p_{io} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & p_x \\ \sin\psi & \cos\psi & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} p_{ro} \qquad (3)$$

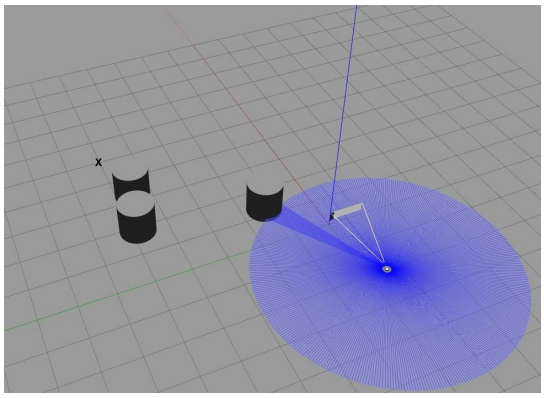

Figura 5. Navegação através de obstáculos utilizando o campo potencial (o objetivo é indicado por "X").

O robô possui duas formas de parada, quando chega no objetivo ou quando atinge um mínimo local. O modo encontrado para avaliar se o robô chegou no seu destino foi através da norma do gradiente atrativo, $\|\nabla U_{att}\|$, que tende a zero quando o robô se aproxima de seu objetivo. No caso do mínimo local, a verificação é realiza pela norma da soma do gradiente atrativo e repulsivo, $\|\nabla U_{att} + \nabla U_{rep}\|$, que tende a zero quando o caminho é obstruído por algum obstáculo. É difícil fazer fazer a verificação $\|\nabla U\| = 0$, por este motivo é considerado $\|\nabla U\| < \epsilon$, sendo ϵ um parâmetro pequeno que é designado de acordo com os requisitos da tarefa.

O processo de identificação do gradiente do potencial atrativo foi implementado de três formas diferentes, a maneira encontrada para comparar o desempenho de cada uma dessas soluções foi analisando o desempenho do robô em atingir o seu objetivo, quando submetido a navegar entre obstáculos, Fig. 5. Desse modo, as informações da norma do gradiente no decorrer da execução foram coletadas junto com o tempo de execução, o gráfico pode ser visto na Fig. 6. É possível constatar que em determinados instantes a norma do gradiente sofre decaimentos, isso ocorre quando o robô encontra um obstáculo no seu caminho. Os dados da funções potenciais referentes a cada gradiente também foram coletadas e plotadas em relação ao tempo de execução, Fig. 7, é observado que o valor de cada função decresce até atingir zero quando o robô chega em seu objetivo. Comparando as três funções potencias é averiguado que a função potencial cônico tem o pior tempo de execução entre as três.

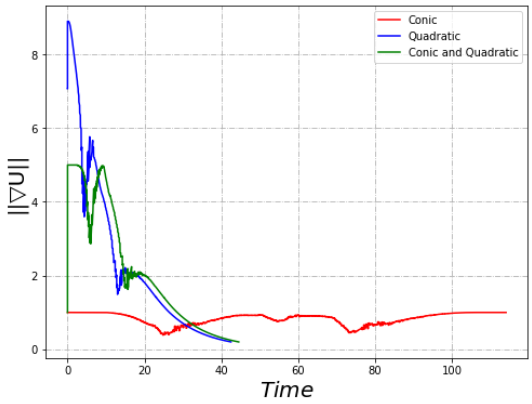

Figura 6. Gradientes das funções potenciais no tempo de execução.

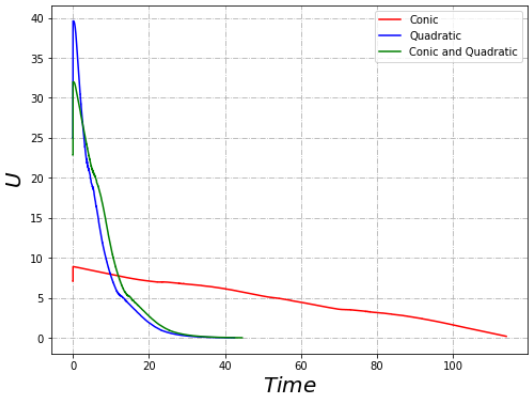

Figura 7. Funções potenciais no tempo de execução.

4. WAVE-FRONT

O planejador wave-front propõe uma das mais simples soluções para o problema de mínimo local, porém, requer um mapa discretizado do ambiente considerando espaços livres e ocupados e ainda, as coordenadas atuais do robô e do objetivo.

Para construção do mapa, foram extraídas do modelo 3D imagens da vista de cima compreendendo o plano

xy, como exemplificado na Fig. 8 (Original). As mesmas foram tratadas para que fossem alinhados os eixos das imagens aos eixos dos ambientes de simulação. Em seguida, um filtro foi utilizado para segmentar as imagens tirando proveito da intensidade dos tons de cinza que compunham os modelos, deste modo as imagens se tornaram matrizes de 1's e 0's, identificando caminhos livres e ocupados respectivamente. O último tratamento se deu pela erosão do mapa, crescendo os obstáculos na proporção do raio do robô. O eé exemplificado na Fig. 8 (Treated Map), onde os caminhos livres são coloridos em branco e obstruidos em preto.

Na implementação do algoritmo, definido o objetivo, foi considerada a distância de Manhattan, em que a partir do ponto em questão, os vizinhos imediatamente ao Norte, Sul, Leste e Oeste são visitados. Caso o valor de qualquer vizinho seja zero, este é desconsiderado. Caso seja 1, não sendo o próprio objetivo, recebe o valor do pixel atual acrescido de 1. Ainda, caso não seja 0 nem 1, é verificado se seu novo valor é menor que o atual, caso o seja, é alterado. Ao sofrer qualquer alteração, os vizinhos são inseridos em uma fila, sobre a qual o processo vem a se repetir até que a fila seja esvaziada. O resultado é um mapa como na Fig. 8 (Wave-Front), em que o degradé representa o decaimento dos valores dos pixels, propondo o deslocamento das áreas mais claras às mais escuras, porém evitando as de valor 0.

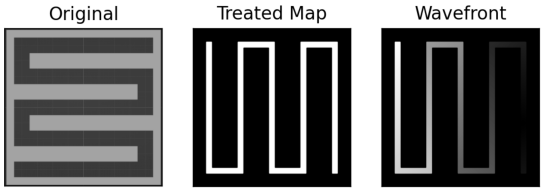

Figura 8. Etapas de tratamento e produção do mapa wave-front.

Durante a simulação, foi assumido que a granularidade é alta de modo a permitir a associação de cada pixel da imagem a um único ponto correspondente no plano xy. Devido a esta mesma propriedade, o ponto seguinte, indicado pelo passo do wave-front, está sempre muito próximo à localização do robô, o que provoca uma lenta convergência do controle proporcional. Por esse motivo, foi utilizado sempre o que seria o décimo passo adiante computado pelo algoritmo. Ainda, devido a ruídos implementados pela simulação física, a dinâmicas não tratadas pela lei de controle e a inexatidão na modelagem da solução, por vezes, o robô chega se posicionar sobre um espaço ocupado do mapa. Para tratar esse problema, a margem do obstáculo foi levemente estendida e diante da ocorrência, recorre-se ao último objetivo determinado pelo algoritmo, antes do robô se posicionar sobre um obstáculo. Ressalta-se que a cada atualização odométrica do robô, um novo caminho é computado com base no mapa e localização corrente. O resultado da primeira computação é ilustrado na Fig. 9, em que o traçado vermelho leva o robô ao objetivo.

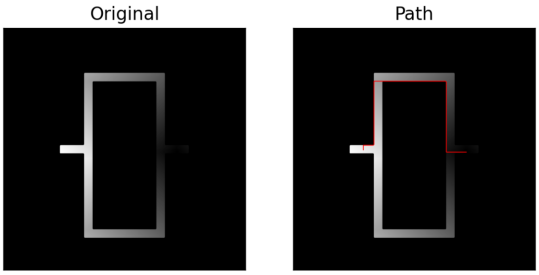

Figura 9. Caminho computado na primeira iteração do algoritmo wave-front.

REFERÊNCIAS

Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G.A., Burgard, W., Kavraki, L.E., and Thrun, S. (2005). *Potential Functions*, 77–106.

Rezende, A.M., Goncalves, V.M., and Pimenta, L.C. (2021). Constructive time-varying vector fields for robot navigation. *IEEE Transactions on Robotics*, 38(2), 852–867.