& & & & **ECAI21.2 - Robótica Móvel (Prática) - Turma 01**

& & & & **Prof. André Chaves Magalhães**
& & & & **Dezembro/2023**
& & & & **Universidade Federal de Itajubá - *Campus* de Itabira**

RELATÓRIO - PROJETO 6

Emilly Lara Deiró*, Rodrigo*, Saulo Labiapari*, Vinicius de Sousa David*, Wilgner Luís Pereira Barreto*

* *Universidade Federal de Itajubá - Campus de Itabira*
Rua Irmã Ivone Drumond, 200 - Distrito Industrial II - 35903-087
Itabira, Minas Gerais, Brasil

E-mails: emillydeiro@unifei.edu.br, @unifei.edu.br, saulolabiapari@unifei.edu.br, d2019005631@unifei.edu.br, wilgner.v9@gmail.com

Resumo— Este trabalho propõe a implementação de uma estratégia simples de função de potencial (Potencial Atrativo + Potencial Repulsivo) para navegar o mesmo robô criado no primeiro trabalho, dentro do StageROS entre duas posições quaisquer num ambiente com obstáculos. Utilizando o Laser para o cálculo das distâncias em relação aos obstáculos e também para encontrar a direção do gradiente.

Palavras-chave— Controlador, Robô, Funções de Potencial, StageROS

1 Introdução

A presente pesquisa tem como propósito implementar uma estratégia simples de função de potencial (Potencial Atrativo + Potencial Repulsivo), a fim de navegar o robô no StageROS entre duas posições quaisquer num ambiente com obstáculos. Para isto, foi utilizado um laser no cálculo das distâncias em relação aos obstáculos e também para encontrar a direção do gradiente.

A função potencial é definida por um campo de potencial artificial U, como na equação X o qual retrata a soma de potenciais repulsivos e atrativos. Onde o potencial atrativo empurra o robô em direção ao alvo, já o repulsivo afasta o robô dos obstáculos. Este método enfatiza a eficiência em tempo real em detrimento da garantia de alcançar o alvo. Entretanto, este pode apresentar falhas durante a busca do caminho.

$$U(q) = U\_atr(q) + U\_rep(q) \tag{1}$$

Em que,

- $U\_atr(q)$ é o potencial atrativo;
- $U\_rep(q)$ é o potencial repulsivo;

2 Objetivos

Este estudo tem como objetivo principal i

3 Ferramentas

Para a realização deste projeto foram utilizados o Sistema Operacional de Robôs (ROS) e o ambiente de simulação StageROS. O ROS é um framework de software livre amplamente utilizado em robótica, que fornece uma plataforma para desenvolvimento, integração e execução de software em robôs. Ele oferece uma ampla variedade de bibliotecas, ferramentas e pacotes para programação de robôs, incluindo controle, percepção, planejamento e comunicação. O ROS foi utilizado neste projeto para implementar

e para visualizar os dados em tempo real no ambiente de simulação.

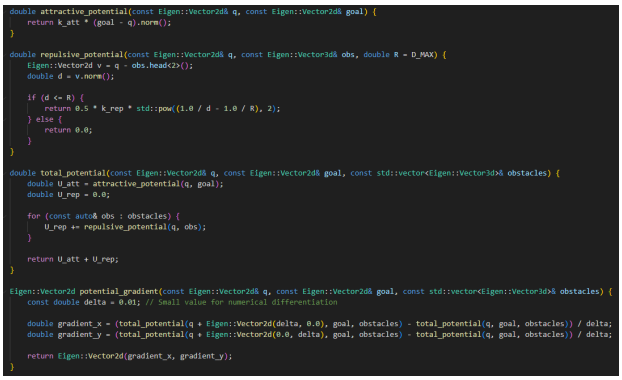
O StageROS é um ambiente de simulação 2D para robôs móveis que permite a criação de cenários virtuais para testar e validar algoritmos de controle e navegação. Ele é baseado no simulador Stage, que é um simulador de robôs de código aberto e multiplataforma. O StageROS foi utilizado neste projeto para simular o robô de acionamento diferencial e para testar

o controlador desenvolvido pelos alunos. Ele permitiu que os alunos avaliassem o desempenho

em diferentes cenários e ajustassem os parâmetros do controlador para melhorar a precisão e a estabilidade do robô.

4 Desenvolvimento

O potencial atrativo e repulsivo são conceitos fundamentais em algoritmos de navegação baseados em campos potenciais. Eles são usados para orientar um



```
double attractive_potential(const Eigen::Vector2d& q, const Eigen::Vector2d& goal) {
    return k_att * (goal - q).norm();
}

double repulsive_potential(const Eigen::Vector2d& q, const Eigen::Vector3d& obs, double R = D_MAX) {
    Eigen::Vector2d v = q - obs.head<2>();
    double d = v.norm();

    if (d <= R) {
        return 0.5 * k_rep * std::pow((1.0 / d - 1.0 / R), 2);
    } else {
        return 0.0;
    }
}

double total_potential(const Eigen::Vector2d& q, const Eigen::Vector2d& goal, const std::vector<Eigen::Vector3d>& obstacles) {
    double U_att = attractive_potential(q, goal);
    double U_rep = 0.0;

    for (const auto& obs : obstacles) {
        U_rep += repulsive_potential(q, obs);
    }

    return U_att + U_rep;
}

Eigen::Vector2d potential_gradient(const Eigen::Vector2d& q, const Eigen::Vector2d& goal, const std::vector<Eigen::Vector3d>& obstacles) {
    const double delta = 0.01; // Small value for numerical differentiation

    double gradient_x = (total_potential(q + Eigen::Vector2d(delta, 0.0), goal, obstacles) - total_potential(q, goal, obstacles)) / delta;
    double gradient_y = (total_potential(q + Eigen::Vector2d(0.0, delta), goal, obstacles) - total_potential(q, goal, obstacles)) / delta;

    return Eigen::Vector2d(gradient_x, gradient_y);
}
```

Figura 1: Calculo do campo

robô móvel em direção a um objetivo desejado (potencial atrativo) e para evitar colisões com obstáculos no ambiente (potencial repulsivo).

1. Potencial Atrativo: - O potencial atrativo é calculado com base na distância entre a posição atual do robô e o objetivo desejado. - Quanto mais distante o robô estiver do objetivo, maior será o potencial atrativo. - O potencial atrativo é projetado para atrair o robô em direção ao objetivo, incentivando-o a se mover na direção correta.

2 Potencial Repulsivo: - O potencial repulsivo é calculado com base na presença de obstáculos no ambiente. - Quanto mais próximos os obstáculos estiverem do robô, maior será o potencial repulsivo. - O potencial repulsivo é projetado para evitar colisões, desviando o robô dos obstáculos e mantendo uma distância segura.

3. Combinação dos Potenciais: - O potencial total é obtido combinando os potenciais atrativo e repulsivo. - Em um ambiente sem obstáculos, o potencial atrativo dominará, direcionando o robô em linha reta em direção ao objetivo. - Quando obstáculos estão presentes, o potencial repulsivo se torna dominante, desviando o robô dos obstáculos.

4. Gradiente de Potencial: - O gradiente de potencial é calculado a partir do potencial total para determinar a direção na qual o robô deve se mover. - Ele fornece informações sobre a direção e a magnitude do movimento necessário para alcançar o objetivo, evitando obstáculos.

No código acima, as funções ‘(double attractive_potential(const Eigen::Vector2d& q, const Eigen::Vector2d& goal), (double repulsive_potential(const Eigen::Vector2d& q, const Eigen::Vector3d& obs, double R = D_MAX)) calcula o potencial atrativo e repulsivo entre um ponto (q) e um objetivo (goal), respectivamente, com um raio (R) padrão de (D_MAX).

A função (double total_potential(const Eigen::Vector2d& q, const Eigen::Vector2d& goal, const std::vector<Eigen::Vector3d>& obstacles)) calcula o potencial total em um ponto (q) com relação a um objetivo (goal) e a uma lista de obstáculos. O potencial total é a soma do potencial atrativo e do potencial repulsivo. Onde a variável (k_att) é uma constante que multiplica o potencial atrativo, enquanto (k_rep) é uma constante que multiplica o potencial repulsivo. Ademais, tem-se a função (laserMessageReceived(const sensor_msgs::LaserScan& laserScan)) que é chamada sempre que uma mensagem do tipo (sensor_msgs::LaserScan) é recebida. A função processa os dados do scan do laser para obter as posições dos obstáculos. Com isto, a função (poseMessageReceived(const nav_msgs::Odometry::ConstPtr& msg)) é chamada sempre que uma mensagem do tipo (nav_msgs::Odometry) é recebida, esta função obtém a posição e orientação atuais do robô.

A geração de comandos de velocidade é uma etapa crucial em algoritmos de navegação baseados em campos potenciais. Uma vez calculado o gradiente de potencial, os comandos de velocidade são gerados para orientar o robô móvel em direção ao objetivo desejado, ao mesmo tempo em que evita colisões com obstáculos no ambiente.

A geração de comandos de velocidade envolve as seguintes etapas:

1. Feedback Linearization: - O gradiente de potencial fornece informações sobre a direção na qual o robô deve se mover. - Para converter o gradiente em comandos de velocidade, é comum aplicar técnicas de feedback linearization. - A técnica de feedback linearization é usada para transformar o sistema não linear (representado pelo gradiente) em um sistema linearizado, facilitando o controle.

2. Limitação da Velocidade: - Antes de aplicar os comandos de velocidade, é importante considerar limites de velocidade para o robô. - Os comandos de velocidade gerados podem ser ajustados para garantir que estejam dentro dos limites de velocidade do robô.

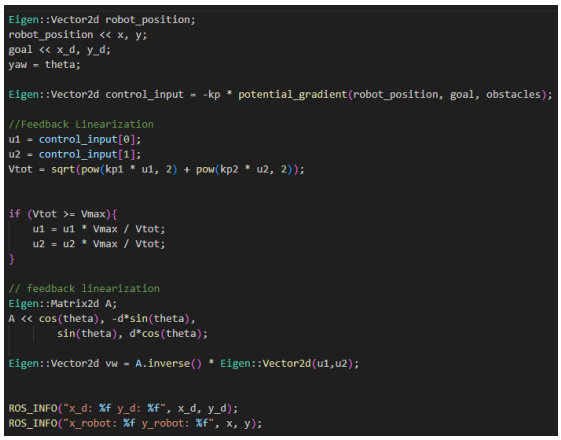
3. Aplicação de Controladores: - Com base no gradiente de potencial e na técnica de feedback linearization, os comandos de velocidade linear e angular são calculados. - Controladores proporcionais, derivativos e integrativos (PID) ou outros métodos de controle podem ser aplicados para ajustar os comandos de velocidade com base nos erros de posição e orientação.

4. Geração de Mensagens de Velocidade: - Os comandos de velocidade resultantes são usados para preencher mensagens de velocidade, geralmente no formato de mensagens ROS (Robot Operating System) ou em outros formatos de comunicação de robôs. - As mensagens de velocidade contêm informações sobre a velocidade linear e angular que o robô deve seguir.

5. Publicação dos Comandos de Velocidade: - As mensagens de velocidade são publicadas no sistema de controle do robô, permitindo que o robô interprete e execute os comandos de movimento.

5 Conclusões

O uso de campos potenciais é uma técnica popular para a navegação autônoma de robôs móveis em ambientes dinâmicos. Este trabalho apresentou uma implementação de um controlador de campo poten-



```
Eigen::Vector2d robot_position;
robot_position << x, y;
goal << x_d, y_d;
yaw = theta;

Eigen::Vector2d control_input = -kp * potential_gradient(robot_position, goal, obstacles);

//Feedback Linearization
u1 = control_input[0];
u2 = control_input[1];
Vtot = sqrt(pow(kp1 * u1, 2) + pow(kp2 * u2, 2));


if (Vtot >= Vmax){
    u1 = u1 * Vmax / Vtot;
    u2 = u2 * Vmax / Vtot;
}

// feedback linearization
Eigen::Matrix2d A;
A << cos(theta), -d*sin(theta),
     sin(theta), d*cos(theta);

Eigen::Vector2d vw = A.inverse() * Eigen::Vector2d(u1,u2);


ROS_INFO("x_d: %f y_d: %f", x_d, y_d);
ROS_INFO("x_robot: %f y_robot: %f", x, y);
```

Figura 2: FeedBack Linearization

cial para um robô móvel, que utiliza um potencial atrativo para guiar o robô em direção a um objetivo desejado e um potencial repulsivo para evitar colisões com obstáculos no ambiente.

O controlador de campo potencial apresentado neste trabalho utiliza o gradiente de potencial para gerar comandos de velocidade que orientam o robô em direção ao objetivo desejado, ao mesmo tempo em que evita colisões com obstáculos. A técnica de feedback linearization é usada para converter o gradiente de potencial em comandos de velocidade linear e angular, que são ajustados com base nos erros de posição e orientação.

A implementação do controlador de campo potencial foi testada em um ambiente simulado, onde o robô móvel foi capaz de navegar com sucesso em direção ao objetivo desejado, evitando obstáculos no ambiente. Os resultados mostraram que o controlador de campo potencial é uma técnica eficaz para a navegação autônoma de robôs móveis em ambientes dinâmicos.

Referências