Dissertação de Mestrado

Framework para teleoperação de robôs móveis

por Andressa Martins Oliveira

orientada por
Prof. Dr. Ícaro Bezerra Queiroz de Araújo

Universidade Federal de Alagoas
Instituto de computação
Maceió, Alagoas
Julho 19, 2023

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de computação

FRAMEWORK PARA TELEOPERAÇÃO DE ROBÔS MÓVEIS

Dissertação de Mestrado submetida pelo Instituto de computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do diploma em Mestre em informática

Andressa Martins Oliveira

Orientador: Prof. Dr. Ícaro Bezerra Queiroz de Araújo

Banca Examinadora:

Tiago Figueiredo Vieira	Prof. Dr., UFAL
Allan de Medeiros Martins	Prof. Dr., UFRN

Maceió, Alagoas
Julho 19, 2023

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

O48f Oliveira, Andressa Martins.
Framework para teleoperação de robôs móveis / Andressa Martins Oliveira. – 2023.
56 f. : il.

Orientador: Icaro Bezerra Queiroz de Araújo.
Dissertação (mestrado em informática) - Universidade Federal de Alagoas. Instituto de Computação. Maceió, 2023.

Bibliografia: f. 50-56.

1. Teleoperação. 2. Robótica móvel. 3. *Framework* de teleoperação. 4. Aplicações web. 5. Robôs - Sistemas de controle. I. Título.

CDU: 004.896

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Programa de Pós-Graduação em Informática – PPGI
Instituto de Computação/UFAL
Campus A. C. Simões BR 104-Norte Km 14 BL 12 Tabuleiro do Martins
Maceió/AL - Brasil CEP: 57.072-970 | Telefone: (082) 3214-1401

Folha de Aprovação

ANDRESSA MARTINS OLIVEIRA

FRAMEWORK PARA TELEOPERAÇÃO DE ROBÔS MÓVEIS

Dissertação submetida ao corpo docente do Programa de Pós-Graduação em Informática da Universidade Federal de Alagoas e aprovada em 19 de julho de 2023.

Banca Examinadora:

gov.br Documento assinado digitalmente
ICARO BEZERRA QUEIROZ DE ARAUJO
Data: 19/07/2023 17:01:25-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. ICARO BEZERRA QUEIROZ DE ARAUJO
UFAL – Instituto de Computação
Orientador

gov.br Documento assinado digitalmente
TIAGO FIGUEIREDO VIEIRA
Data: 20/07/2023 09:46:07-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. TIAGO FIGUEIREDO VIEIRA
UFAL – Instituto de Computação
Examinador Interno

gov.br Documento assinado digitalmente
ALLAN DE MEDEIROS MARTINS
Data: 20/07/2023 10:42:41-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. ALLAN DE MEDEIROS MARTINS
UFRN- Universidade Federal do Rio Grande do Norte
Examinador Externo

Dedicatória

À minha família e amigos.

Agradecimentos

Agradeço aos meus pais Gilvanice e Josenildo por seu amor incondicional, apoio constante e pelos sacrifícios que fizeram para me proporcionar esta oportunidade. Seu encorajamento e orientação foram fundamentais em cada passo desta jornada. Sem vocês, eu não estaria aqui hoje.

Ao meu irmão Hugo, obrigado por sempre acreditar em mim e me motivar a buscar a excelência. Sua força e apoio foram um farol constante.

Minha gratidão se estende ao meu amor Felipe, cujo apoio inabalável me impulsionou nos momentos de desafio. Sua paciência e compreensão foram meu refúgio. Com você ao meu lado, esta jornada se tornou mais significativa.

Não poderia deixar de expressar minha profunda gratidão aos meus orientadores Ícaro e Heitor. Suas orientações sábias, *insights* valiosos e dedicação incansável foram essenciais para moldar esta dissertação. Seu compromisso com meu crescimento acadêmico deixou uma marca duradoura em mim.

Ao grupo de laboratório AI-SPARC, expresso minha profunda gratidão por possibilitar o aprimoramento do meu trabalho ao longo desta jornada. Foi um privilégio fazer parte deste grupo e ter a oportunidade de compartilhar meu desenvolvimento, bem como receber valiosas contribuições em cada apresentação que realizei.

Aos meus amigos Arthur, Bruno, Eduardo, Glauber, Gustavo, Luís, Maria e Winnie, que estiveram ao meu lado oferecendo apoio e amizade sincera, gostaria de expressar minha gratidão. Suas palavras de encorajamento, risos compartilhados e apoio constante tornaram esta jornada acadêmica muito mais significativa e memorável.

Andressa Martins Oliveira

Resumo

Apesar dos notáveis avanços em robôs autônomos, certas atividades ainda requerem supervisão humana devido a várias barreiras encontradas na interação complexa com ambientes em constante mudança e situações imprevisíveis. Por conseguinte, o desenvolvimento de sistemas telerobóticos tem ganhado destaque em diversas áreas de atuação. Nesse contexto, este trabalho propõe o desenvolvimento de um sistema de teleoperação para robô móvel, capaz de ser aplicado em diferentes arquiteturas de controle de robôs teleoperados. Esse sistema implementa os modos de controle compartilhado e supervisor, utilizando o framework ROS (Robot Operating System) e suas ferramentas de comunicação e visualização entre o operador e o robô. O sistema desenvolvido integra-se com a estrutura do ROS e aproveita suas funcionalidades para a teleoperação do robô móvel. O controle compartilhado permite que o operador forneça comandos de controle em tempo real, enquanto o modo supervisor permite que o robô execute tarefas com certa autonomia, utilizando dados do ambiente e do sensoriamento.

Palavras-chave*: *Teleoperação*; *Robótica móvel*; *Framework de teleoperação*; *Aplicação Web*; *ROS

Abstract

Despite the notable advancements in autonomous robots, certain activities still require human supervision due to various barriers encountered in complex interactions with ever-changing environments and unpredictable situations. Consequently, the development of telerobotic systems has gained prominence in various fields of application. In this context, this work proposes the development of a teleoperation system for a mobile robot, capable of being applied in different architectures for controlling teleoperated robots. This system implements shared control and supervisory control modes, using the ROS (Robot Operating System) framework and its communication and visualization tools between the operator and the robot. The developed system integrates with the ROS framework and leverages its functionalities for teleoperating the mobile robot. The shared control mode allows the operator to provide real-time control commands, while the supervisory mode enables the robot to perform tasks with a certain level of autonomy, utilizing environmental and sensory data.

Keywords: ***Teleoperation***; ***Mobile Robotics***; ***Telesystems***; ***Teleoperation Framework***; ***Web application***; ***ROS***

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AMCL	Adaptive Monte Carlo Localization
AMR	Autonomous Mobile Robot
API	Application Programming Interface
EKF	Extended Kalman Filter
KF	Kalman Filter
MVC	Model-View-Controller
MVP	Model-View-Presenter
MVVM	Model-View-ViewModel
PF	Particle Filter
ROS	Robot Operating System
SSR	Server-Side Rendering
SPA	Single-Page Application
VR	Virtual Reality

Sumário

Capítulo 1

Introdução

De acordo com um relatório recente da Federação Internacional de Robótica, prevê-se um crescimento significativo no número de robôs em operação nas fábricas em todo o mundo durante a década de 2020 [IFR, 2022]. Apesar dos desafios enfrentados recentemente, como interrupções na cadeia de suprimentos e mudanças nas demandas do mercado, espera-se que a adoção de robôs industriais continue a aumentar. Esse crescimento anual reflete o reconhecimento dos benefícios oferecidos pela automação robótica, como aumento da eficiência, redução de erros e melhorias na qualidade e consistência dos produtos. Além disso, a crescente demanda por flexibilidade e personalização impulsiona a necessidade de sistemas de robôs altamente adaptáveis e programáveis

A automação envolve a programação de robôs para executar tarefas de forma autônoma, sem a necessidade de intervenção humana direta. É amplamente utilizada em processos industriais, onde a repetitividade, precisão e velocidade são essenciais. Os robôs autônomos são programados para seguir um conjunto de instruções e realizar tarefas de forma eficiente e consistente. Isso permite um aumento significativo na produtividade e eficiência em muitos setores, como manufatura, logística e agricultura.

Em geral, embora as tendências nessas áreas nos levem a um cenário em busca de robôs totalmente autônomos, ainda existem processos nos quais ações diretas não são possíveis, inadequadas ou em cenários onde é impossível estabelecer rotinas de operação, seja por causa da imprevisibilidade, grau de risco ou complexidade [Vertut and Coiffet, 1985]. Por isso, a teleoperação refere-se ao controle remoto de um robô por um operador humano. Nesse caso, o operador está envolvido ativamente na tomada de decisões e no controle das ações do robô desempenhando um papel fundamental em diversas áreas, desde a exploração espacial até a manufatura de produtos e serviços em setores como a indústria automotiva e aeroespacial. Além disso, é utilizada em aplicações militares, segurança pública e resgate em situações de desastre, permitindo a realização de tarefas perigosas sem expor os operadores a riscos desnecessários.

1.1 Problema

A motivação para utilizar a teleoperação em robótica surge de várias necessidades e desafios específicos em diferentes áreas de aplicação. Existem situações em que a automação total não é viável ou adequada devido à complexidade da tarefa, à imprevisibilidade do ambiente ou à falta de capacidade dos robôs de tomar decisões autônomas. Nesses casos, a teleoperação permite que os operadores humanos intervenham e controlem o robô remotamente, utilizando suas habilidades cognitivas e experiência para lidar com situações complexas e tomar decisões em tempo real.

Um dos principais desafios é a falta de flexibilidade dos sistemas robóticos, que são frequentemente projetados para executar tarefas específicas e repetitivas, sem a capacidade de se adaptar a novas situações ou mudanças no ambiente. Isso se torna especialmente evidente em atividades que exigem coordenação entre a mão e o olho, bem como em tarefas que requerem reconhecimento de objetos, consciência situacional ou outros tipos de percepção [Murphy, 2019].

Assim, a teleoperação se torna uma ferramenta valiosa para superar os desafios da automação em ambientes complexos e dinâmicos, proporcionando uma combinação poderosa de habilidades humanas e capacidades robóticas. Ao permitir o controle humano em tempo real, a teleoperação desempenha um papel fundamental na execução de tarefas críticas e na garantia da segurança em diversas áreas, desde exploração espacial até operações de resgate.

É importante considerar cuidadosamente a interação entre a automação e a intervenção humana e implementação de sistemas robóticos, a fim de maximizar os benefícios da automação, ao mesmo tempo em que se superam suas limitações inerentes.

1.2 Proposta

Este trabalho propôs o desenvolvimento de um sistema supervisório para um robô móvel solo, capaz de lidar com os dois níveis de controle para realizar a teleoperação, por meio de uma interface baseada em *Single Page Application* (SPA), uma aplicação web que funciona dentro de uma única página, onde o conteúdo é atualizado dinamicamente sem a necessidade de recarregar a página inteira. Através dessa interface, foi possível monitorar as principais características do robô, como velocidades, nível de bateria e leituras dos sensores. Além disso, foram utilizados protocolos de comunicação bidirecional entre a interface e o robô, visando garantir uma interação eficiente e confiável. A implementação do sistema em ambiente web facilitou o acesso e o gerenciamento remoto do robô, ampliando as possibilidades de uso em diferentes cenários e ambientes. O desenvolvimento desse sistema contribuiu para a pesquisa em teleoperação de robôs móveis, oferecendo uma solução flexível para a realização de tarefas remotas, como a navegação de robôs.

1.3 Trabalhos relacionados

Uma das primeiras ocorrências de teleoperação robótica foi em 1948 em manuseio de materiais químicos e nucleares por Ray Goertz [Li et al., 2015]. Nos anos seguintes, sistemas teleoperados foram desenvolvidos para outras aplicações militares, como o controle remoto de tanques e aeronaves. Na década de 1960, a NASA começou a explorar o uso de robôs em ambientes espaciais, desenvolvendo sistemas teleoperados para operações em planetas e luas. O programa Apollo utilizou um sistema teleoperado para a primeira missão tripulada à Lua [Anderson et al., 2020].

A utilização de interfaces gráficas de usuário proporciona uma abordagem visual e interativa, permitindo que o operador visualize imagens em tempo real e as controle usando dispositivos de entrada como joysticks, mouses, teclados, telas sensíveis ao toque e simuladores de direção [Cynthia et al., 2021, Szymańska et al., 2021, Hainsworth, 2001, Law et al., 2022, Lee and Ham, 2022].

Em busca de uma experiência mais imersiva e precisa para o operador, os trabalhos em desenvolvimento têm explorado a utilização da Realidade Virtual (VR), onde o operador se sente como se estivesse no ambiente onde o robô está operando [Cynthia et al., 2021, Ni et al., 2017]. A VR também pode ser usada para fornecer feedback sensorial ao operador humano, como a sensação de toque e a temperatura do ambiente em locais desestruturados, como um campo de lavra [Chen et al., 2020].

A teleoperação de robôs combinada com VR também tem aplicações em indústrias como a manufatura, onde o controle remoto de robôs pode aumentar a eficiência e reduzir os riscos para os trabalhadores humanos [Matsas et al., 2018]. Além disso, existem aplicações que utilizam a realidade aumentada, trazendo muitos benefícios em áreas como manufatura, medicina [Lin et al., 2022], exploração espacial e outras, onde a operação remota e a visualização de informações adicionais são importantes [Okura et al., 2010, Walker et al., 2019, Johansson and de Vin, 2009].

Na indústria, a teleoperação é principalmente utilizada para controlar e inspecionar máquinas e equipamentos remotamente, o que pode ser especialmente útil em ambientes perigosos ou de difícil acesso. Uma das principais aplicações da teleoperação é na área da manutenção remota de equipamentos. Através desse método, técnicos têm a capacidade de realizar a manutenção de equipamentos de forma remota, evitando a necessidade de se deslocarem pessoalmente até os locais onde os equipamentos estão instalados. Nesse contexto, surgem novas interfaces humano-máquina com o objetivo de aprimorar a eficiência, reduzir os riscos e aumentar a precisão das atividades que requerem a presença humana no *loop* de controle [González et al., 2021, Tokatli et al., 2021, Balachandran et al., 2020].

Na área de inspeção de infraestrutura, a teleoperação pode ser empregada para inspecionar pontes, viadutos, oleodutos e outros tipos de infraestrutura crítica. Os operadores podem controlar robôs equipados com câmeras e sensores para examinar a condição es-

trutural, identificar danos ou desgastes, e auxiliar na tomada de decisões relacionadas à manutenção e reparos [Schmidt et al., 2014].

Existem vários tipos de interfaces para teleoperação, cada um com suas próprias vantagens e limitações. Uma opção é a utilização de interfaces web, que podem ser mais simples de se implementar em comparação com interfaces de Realidade Virtual (VR) ou Realidade Aumentada (AR). As interfaces web permitem o acesso e controle remoto do robô por meio de um navegador web, o que torna mais fácil e acessível para os operadores realizarem a teleoperação. Além disso, geralmente exigem menos recursos computacionais e *hardwares* mais baratos, o que pode ser vantajoso em termos de custo e eficiência. No entanto, é importante considerar as necessidades específicas de cada aplicação e escolher a interface adequada que melhor atenda aos requisitos da teleoperação.

Neste trabalho, apresentamos uma solução de teleoperação de robôs móveis que utiliza uma implementação baseada em web para controlar um robô em diferentes níveis que a teleoperação tem a oferecer, como o controle direto e supervisório. O uso da tecnologia React nos permite criar uma aplicação multiplataforma, garantindo sua versatilidade e compatibilidade com diversos dispositivos web. Além disso, o fato de ser um projeto de código aberto e desenvolvido com o ROS (*Robot Operating System*) aumenta sua replicabilidade e permite sua utilização em qualquer robô móvel que utilize as mensagens e tópicos de comandos do ROS. Essa abordagem abrangente e acessível tem o potencial de impulsionar avanços significativos na teleoperação de robôs móveis, tornando-a mais eficiente, segura e adaptável às necessidades específicas de cada aplicação.

1.4 Estrutura

Este trabalho é composto por quatro capítulos. O Capítulo 2 aborda definições básicas e importantes sobre teleoperação, incluindo componentes, arquiteturas de controle e aplicações de sistemas teleoperados. O Capítulo 3 discute as abordagens de SPA (*Single Application Page*), explorando arquiteturas de implementação e seu funcionamento. O Capítulo 4 apresenta as principais técnicas, conceitos e ferramentas que possibilitam a navegação autônoma de um robô móvel. O Capítulo 5 contém os resultados alcaçandos nesse trabalho e o Capítulo 6 discute sobre as conclusões e trabalhos futuros [1].

[1] Todos os algoritmos utilizados neste trabalho podem ser encontrados em https://github.com/AndressaM/robot-teleoperation.

Capítulo 2

Teleoperação

Este capítulo tem como objetivo fornecer uma visão abrangente dos sistemas teleoperados na área, abordando suas definições, arquiteturas, componentes e métodos de teleoperação.

De acordo com [Ferre et al., 2007] a teleoperação é conectar seres humanos e robôs para reproduzir as ações do operador à distância. Essa operação pode ser realizada por meio de uma variedade de métodos, cada um com suas próprias vantagens e limitações. O objetivo principal da teleoperação é executar tarefas em ambientes perigosos, difíceis ou inacessíveis para os seres humanos, ou estender as capacidades dos operadores humanos além de seus limites físicos. Nesse contexto, [Aracil and Ferre, 2007] desenvolveram um sistema teleoperado para realizar manutenção em linhas vivas de redes elétricas, removendo assim todo perigo o qual um humano estava submetido nessa atividade.

Um sistema teleoperado é recomendado em situações em que a tarefa a ser realizada é complexa, não pode ser totalmente especificada antecipadamente ou requer interação em tempo real com o ambiente.

Ao adotar a teleoperação, os operadores podem realizar atividades em locais remotos ou hostis, minimizando riscos à sua segurança e saúde. Isso permite que tarefas desafiadoras sejam executadas com maior precisão e eficiência, além de possibilitar a exploração de ambientes inacessíveis ou desconhecidos.

Em resumo, a teleoperação oferece uma solução viável para superar os desafios impostos por ambientes perigosos ou inacessíveis, além de ampliar as capacidades humanas. Com o avanço contínuo da tecnologia, a teleoperação continuará a desempenhar um papel importante em diversas áreas, desde a exploração espacial até a indústria, medicina, agricultura e muito mais.

De acordo com [Dorf and Nof, 1990] existem diretrizes que determinam se um sistema teleoperado é aplicável ou não:

- As tarefas não seguem um padrão e não são repetitivas
- Partes da atividade exigem habilidades de manipulação, especialmente a coordenação *hand-eye*

- As tarefas requerem reconhecimento de objeto ou consciência situacional, como ocorre geralmente em robôs que auxiliam médicos em cirurgias.
- As necessidades da tecnologia de visualização não excedem as limitações do link de comunicação (largura de banda, atrasos de tempo)

2.1 Componentes básicos para a teleoperação

No estudo realizado por Uttal em 1989 [Uttal, 1989], é estabelecido que um sistema de teleoperação é composto por sete componentes essenciais. Esses componentes são divididos em dois grupos: os presentes na estação de trabalho local e os presentes no robô. Além disso, é necessário um canal de comunicação para conectar esses componentes.

Na literatura, esses componentes são considerados fundamentais para viabilizar a teleoperação, permitindo o controle eficiente e seguro de um robô em um ambiente remoto. Os componentes incluem dispositivos de entrada, canal de comunicação, dispositivos de saída, sistema de controle, robô e ambiente remoto. Essa configuração colabora para o funcionamento integrado e efetivo do sistema de teleoperação. Vamos destacar brevemente os papéis desempenhados por cada um dos componentes principais:

Os dispositivos de entrada e saída desempenham um papel importante ao permitir que o operador humano capture informações sobre o ambiente e controle o robô. Conhecidas como interfaces homem-máquina, elas podem incluir elementos visuais, como telas e gráficos, bem como dispositivos de entrada, como *joysticks*, teclados ou até mesmo interfaces de realidade virtual. Além disso, as IHMs podem fornecer informações adicionais, como dados sensoriais, mapas ou visualizações 3D do ambiente. Uma interface bem projetada e intuitiva é essencial para garantir que o operador possa controlar o robô de forma precisa e eficiente, mesmo em situações desafiadoras. Estudos de casos de usos são realizados para aprimoramento dessas interfaces buscando cada vez mais a imersão do usuário na atividade de teleoperação [Graf and Hussmann, 2020, Hetrick et al., 2020].

O canal de comunicação é responsável pela transmissão das informações entre o operador humano e o robô. Pode ser estabelecido por meio de redes sem fio, conexões por cabo ou até mesmo tecnologias avançadas. Pesquisas sobre redes em teleoperação desempenham um papel importante na evolução desse campo. Com o avanço das tecnologias de comunicação, como o desenvolvimento da tecnologia 5G, há a possibilidade de alcançar latências ultra baixas e maior largura de banda, proporcionando uma transmissão de dados mais rápida e confiável[Chen et al., 2022]. Isso é crucial para a teleoperação, pois permite uma comunicação em tempo real entre o operador e o robô, garantindo uma resposta imediata aos comandos e *feedbacks* precisos sobre o ambiente.

O sistema de controle desempenha um papel central na teleoperação, sendo responsável pelo gerenciamento do robô e pela comunicação bidirecional entre o operador humano e

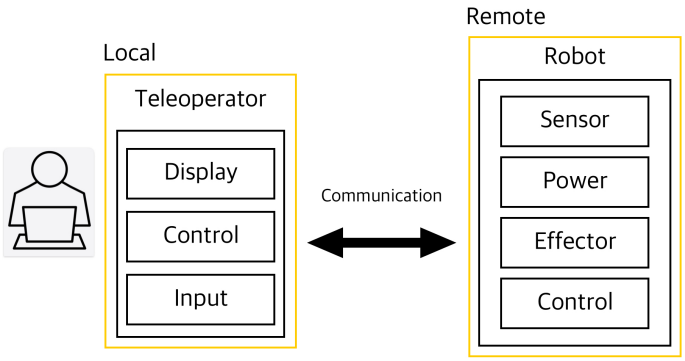
o robô. Ele inclui algoritmos e software que garantem uma resposta precisa e em tempo real aos comandos do operador, além de permitir o monitoramento do estado do robô e do ambiente remoto.

O robô é a entidade física que executa as tarefas designadas pelo operador humano. Pode variar desde um pequeno veículo terrestre até um braço robótico complexo, dependendo do propósito e das necessidades da teleoperação.

Por fim, o ambiente remoto é o local onde o robô opera e executa as tarefas atribuídas. Pode ser um ambiente perigoso, inacessível ou simplesmente distante do operador humano, tornando a teleoperação uma solução viável para realizar atividades nesses contextos desafiadores.

Em resumo, um sistema de teleoperação envolve uma interação complexa entre dispositivos de entrada, canal de comunicação, dispositivos de saída, sistema de controle, robô e ambiente remoto. Essa integração eficiente dos componentes é fundamental para garantir o sucesso e a segurança da teleoperação, permitindo que os operadores humanos controlem e executem tarefas em ambientes remotos. A Figura 2.1 representa como os componentes de teleoperação estão distribuídos em um sistema de teleoperação.

Figura 2.1: Representação dos componentes de um sistema de teleoperação



Fonte : Adaptada de [Murphy, 2019]

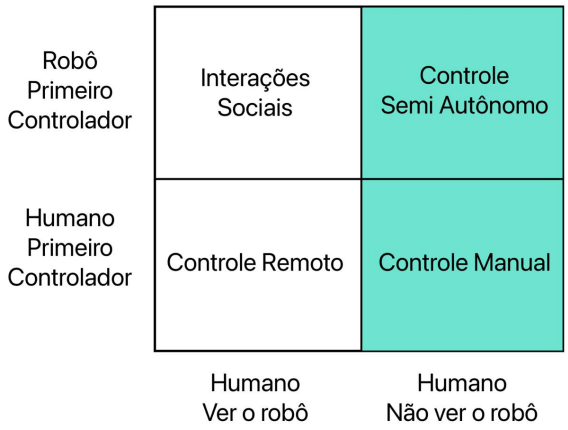
2.2 Métodos de teleoperação

Alguns métodos comuns de teleoperação, como teleoperação direta, controle de supervisão, controle negociado e controle compartilhado, apresentam vantagens e desvantagens distintas. Por exemplo, a teleoperação direta permite um controle mais imediato e preciso, mas requer habilidades especializadas do operador. O controle de supervisão oferece maior segurança e autonomia do robô, mas pode resultar em menor agilidade e resposta em tempo real. O controle negociado busca um equilíbrio entre o controle do operador e o controle autônomo do robô, mas pode exigir maior complexidade na interface. A teleprogramação permite uma especificação precisa de sequências de ações, mas pode li-

mitar a adaptabilidade a novas situações. A telepresença proporciona uma experiência mais imersiva para o operador, mas pode enfrentar desafios técnicos e latência na transmissão de dados. Cada método possui suas próprias características a serem consideradas, dependendo da aplicação e do ambiente em que é utilizado.

Os tipos de controle de um robô podem ser representados por uma matriz com dois eixos ortogonais os quais devem responder as seguintes questões : Você pode ver o robô no ambiente ? e Onde está a a maior parte da inteligência : local ou remoto?. [Murphy, 2019]

Figura 2.2: Tipos de controle divididos em quadrantes



Fonte : Adaptada de AI Robotics [Murphy, 2019]

Na coluna representada a esquerda temos dois tipos de controles que não são considerados como teleoperação pois em ambos o operador consegue ver o robô e como definido pela autora a teleoperação ocorre quando o humano não pode ver o robô, isto é quando há distância entre eles. A coluna à direita de cor azul apresenta os tipos de controle que encontram-se em sistemas teleoperados que serão detalhados a seguir.

2.2.1 Controle Manual

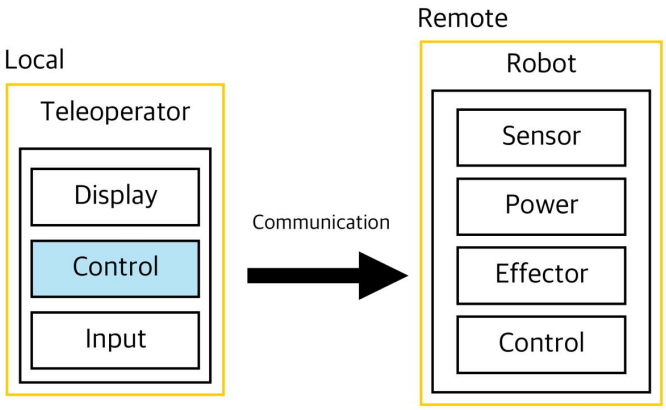
O controle manual ou direto é uma abordagem comum na teleoperação de robôs, em que o operador humano controla diretamente os movimentos do robô usando um dispositivo de entrada, como um *joystick* ou um dispositivo de realidade virtual.

Nesse tipo de controle, os movimentos do dispositivo de entrada são mapeados diretamente nos movimentos do robô, sem a necessidade de um sistema de controle intermediário. Isso permite que o operador humano tenha um controle preciso e intuitivo do robô, mas também pode apresentar desafios, como a necessidade de treinamento e habilidade para controlar o robô de forma eficaz, especialmente em ambientes complexos e dinâmicos.

O controle direto pode ser apropriado para tarefas simples e repetitivas, onde a precisão é essencial e o ambiente é estático e previsível. No entanto, em tarefas mais complexas

e dinâmicas, é mais comum utilizar uma abordagem de controle mais avançada, como o controle supervisório ou o controle baseado em metas, para auxiliar o operador humano no controle do robô e garantir a segurança do ambiente e do robô.

Figura 2.3: Controle direto



Fonte : Adaptada de AI Robotics

2.2.2 Controle Semi-Autônomo

Introduzido por [Ferrell and Sheridan, 1967], o controle supervisório representa uma abordagem de controle de robôs que estabelece uma relação mais abrangente entre o robô e o operador. Nesse contexto, o robô é capaz de executar atividades com maior autonomia, enquanto o operador desempenha um papel de supervisão e tomada de decisões de alto nível.

O controle supervisório proporciona uma interação mais sofisticada entre o robô e o operador, permitindo que o robô execute tarefas complexas de forma autônoma, enquanto o operador estabelece comandos e restrições gerais. Isso possibilita ao robô lidar com ambientes dinâmicos e imprevisíveis, adaptando seu comportamento de acordo com as circunstâncias

Essa abordagem de controle tem sido amplamente explorada em diversas áreas, como robótica industrial, robótica móvel, assistência médica e aplicações de pesquisa [González et al., 2021, Tokatli et al., 2021, Deng et al., 2021, Balachandran et al., 2020] Através do controle supervisório, é possível melhorar a eficiência e a flexibilidade do robô, aumentar a segurança em ambientes de trabalho complexos e permitir a realização de tarefas que exigem uma interação mais próxima entre o robô e o ambiente.

Controle compartilhado

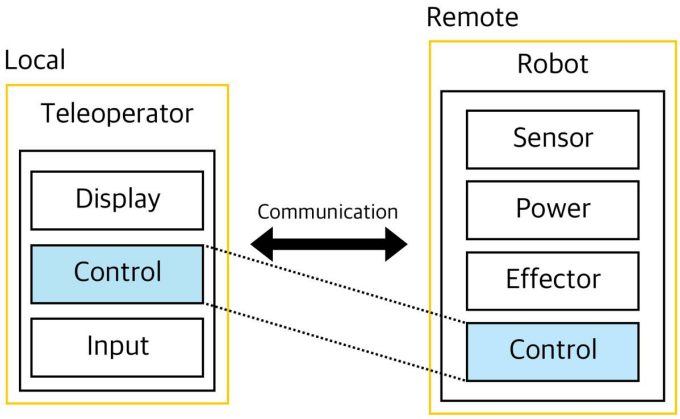
O controle compartilhado é uma abordagem em que o controle do robô é dividido entre o operador humano e um sistema de controle automático. Nessa abordagem, o operador hu-

mano tem controle direto sobre algumas partes do robô, enquanto outras são controladas automaticamente pelo sistema de controle. A Figura 2.4 mostra como está estabelecida a relação entre o operador e o robô sendo compartilhada pelos seus módulos de controle.

Por exemplo, o operador humano pode controlar diretamente os movimentos do braço do robô, enquanto o sistema de controle automático cuida do controle da câmera ou dos movimentos da base do robô. O objetivo do controle compartilhado é combinar a habilidade e a capacidade do operador humano com a precisão e a eficiência do controle automático, para melhorar a qualidade do controle do robô e reduzir o esforço e a fadiga do operador humano.

O controle compartilhado é comumente usado em situações em que o ambiente é complexo ou dinâmico, ou em tarefas que exigem grande precisão. A abordagem pode ajudar a reduzir o tempo de resposta do operador humano e minimizar o risco de erros, além de aumentar a eficiência e a segurança do controle do robô, como é observado pelo autor [Gottardi et al., 2022]. No entanto, também pode apresentar desafios, como a necessidade de sincronização precisa entre o operador humano e o sistema de controle automático.

Figura 2.4: Controle compartilhado

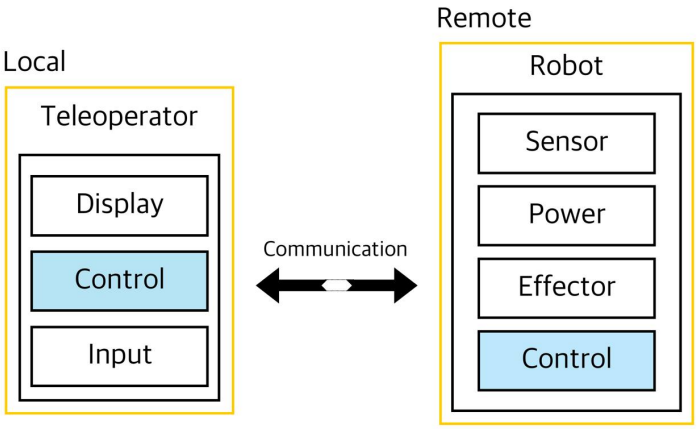


Fonte : Adaptada de [Murphy, 2019]

Uma variação desse método ocorre quando há uma discretização das atividades realizadas entre o operador e o robô, ou seja, eles não trabalham simultaneamente como representado na Figura 2.5, defindo como controle compartilhado, do inglês *traded control*. Essa abordagem é discutida pelos autores em [Oh et al., 2021], onde propõem a utilização de gestos para indicar uma sequência de objetos que o robô irá manipular, gerando movimentos de agarra ou otimização de trajetória de forma autônoma.

Quando o controle é semi autônomo e não há intervenção direta do operador, temos o que chamamos de automação. No entanto, vale ressaltar que o foco desta dissertação não está na automação, mas sim na teleoperação, que envolve a participação ativa e o controle do operador no processo de operação do sistema.

Figura 2.5: Controle negociado



Fonte : Adaptada de [Murphy, 2019]

2.2.3 Telepresença

A telepresença é um conceito que vai além da teleoperação de máquinas, incorporando também uma dimensão social. Ela se baseia na ideia de representação do operador e busca criar uma experiência imersiva no ambiente remoto. Ao contrário da teleoperação, que se concentra principalmente na operação remota de dispositivos, a telepresença visa aproximar e representar o humano em um ambiente específico.

Ela permite que uma pessoa se sinta presente e interaja em um local remoto como se estivesse fisicamente lá. Isso é possível graças a avanços em tecnologias de comunicação, como áudio, vídeo e realidade virtual, que possibilitam a transmissão de informações sensoriais em tempo real.

Ao proporcionar uma sensação de presença virtual, a telepresença tem potencial para transformar várias áreas, como reuniões remotas, educação a distância, assistência médica remota e turismo virtual. Ela permite que as pessoas superem as limitações físicas e geográficas, promovendo uma maior interação e colaboração à distância

As aplicações mais relevantes da telepresença estão relacionadas aos robôs sociais, projetados para interagir com seres humanos em ambientes sociais, como residências [Olatunji et al., 2022], escolas [Sharrab et al., 2023] e espaços públicos [De-Sheng and Qin-Yi, 2022]. Esses robôs são equipados com sensores, câmeras e microfones para perceber e responder aos sinais humanos, como expressões faciais, gestos e fala.

Capítulo 3

Single Page Applications

A seção seguinte abordará os principais aspectos relacionados às SPAs, do inglês *Single Page Applications*. Serão explorados os conceitos fundamentais, a arquitetura e as tecnologias utilizadas no desenvolvimento de SPAs. Ao compreender esses elementos, será possível apreciar a importância e a relevância das SPAs na construção de aplicações web modernas e interativas.

3.1 Motivação

As SPAs são uma tecnologia que gerenciam o estado e a lógica da aplicação principalmente no navegador. O autor [Sun, 2019] discorre que ao contrário das abordagens tradicionais de SSR (Server-Side Rendering), proporcionam uma experiência de usuário mais fluida, sem a necessidade de recarregamento completo da página a cada interação do usuário. Isso ocorre porque as interações do usuário, como cliques em botões e envios de formulários, enviam solicitações assíncronas para a API web, então, atualiza a página de forma parcial, proporcionando uma experiência mais suave, sem o incômodo do piscar de página em branco. Além disso, a carga no servidor é reduzida, pois não é mais necessário renderizar toda a página a cada interação. A sincronização do estado entre o navegador e o servidor também se torna mais simples, uma vez que a lógica da aplicação é executada no cliente. Com todas essas vantagens, as SPAs oferecem uma experiência de usuário mais rica, sem interrupções, e reduzem a carga no servidor, resultando em uma aplicação mais eficiente e responsiva.

A modularidade da arquitetura das SPAs simplifica a manutenção, permitindo alterações específicas em componentes individuais. Além disso, são altamente compatíveis com APIs e serviços web, possibilitando a criação de interfaces dinâmicas e interativas. Sua natureza baseada em tecnologias web também permite o desenvolvimento multiplataforma. Em suma, a motivação para a adoção de SPAs é proporcionar uma experiência de usuário aprimorada, com melhor desempenho, menor consumo de dados e maior flexibilidade de desenvolvimento.

3.2 Arquiteturas

As SPAs podem ser construídas utilizando diversas arquiteturas que ajudam a organizar e estruturar o código da aplicação de forma eficiente. Essas arquiteturas oferecem uma abordagem modular, permitindo a separação de responsabilidades e facilitando a manutenção e evolução do código.

As estruturas mais utilizadas em aplicações web são baseadas no modelo MVC, do inglês Model-View-Controller, que promove a separação de responsabilidades e a organização do código. De acordo com o autor [Scott Jr, 2015], a adoção do padrão MVC traz diversos benefícios para as aplicações. Além dos benefícios mencionados anteriormente, como a organização e separação clara das responsabilidades, o padrão MVC também contribui para a escalabilidade da aplicação. Ao dividir a lógica de negócio (Model), a apresentação (View) e o controle do fluxo (Controller), é possível desenvolver módulos independentes que podem ser facilmente estendidos ou substituídos, permitindo que a aplicação cresça de forma flexível e sustentável.

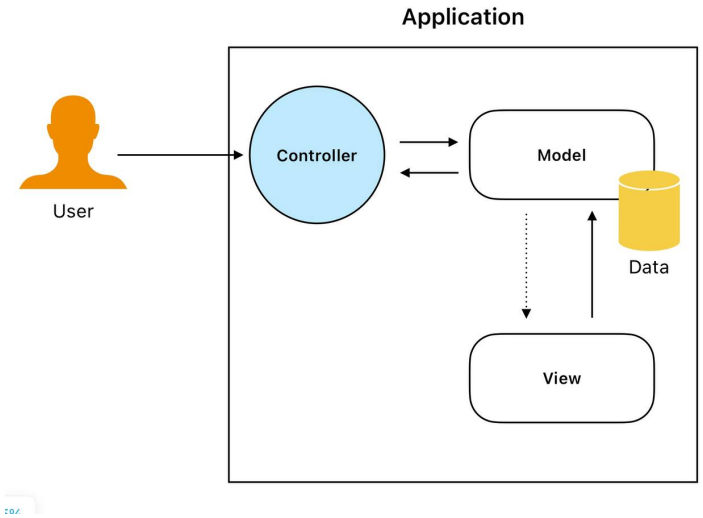
Outra vantagem do uso do padrão MVC é a padronização do código. Ao seguir as diretrizes e convenções do padrão, torna-se mais fácil para os desenvolvedores compreenderem e colaborarem no projeto. A divisão clara entre as partes do padrão também ajuda na manutenção do código, permitindo a identificação e solução de problemas de forma mais eficiente.

Além disso, o padrão MVC proporciona ganhos de produtividade. Com a separação de responsabilidades, diferentes membros da equipe podem trabalhar em paralelo, focando em suas áreas específicas. Isso possibilita o desenvolvimento ágil, com maior rapidez na implementação de novas funcionalidades e menor risco de introduzir erros no código existente. Existem três padrões utilizados na construção de aplicações web, baseados no modelo de arquitetura Model-View-Controller (MVC), Model-View-Presenter (MVP) e Model-View-ViewModel (MVVM).

3.2.1 Model-View-Controller

O padrão Model-View-Controller (MVC) é uma arquitetura de software que divide a aplicação em três componentes principais: o Model, que representa os dados e a lógica de negócio; a View, responsável pela apresentação da interface gráfica ao usuário; e o Controller, que gerencia as interações do usuário e atualiza o Model e a View conforme necessário. O MVC é conhecido por sua separação clara de responsabilidades e pela facilidade de manutenção e teste do código.

Figura 3.1: Diagrama do padrão MVC

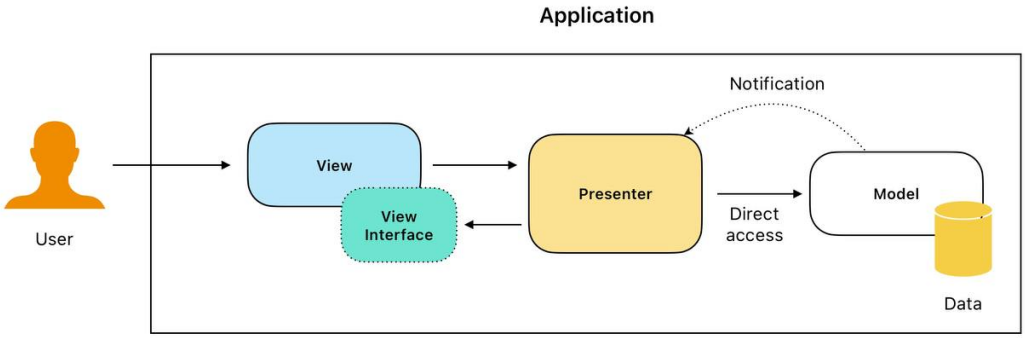


Fonte : Adaptada de [Scott Jr, 2015]

3.2.2 Model-View-Presenter

O padrão Model-View-Presenter (MVP) é uma variação do MVC, na qual o Presenter assume o papel de intermediário entre o Model e a View. O Presenter é responsável por atualizar a View com base nas alterações do Model e capturar as interações do usuário para atualizar o Model. Essa separação de responsabilidades permite uma maior testabilidade e facilita a substituição de componentes.

Figura 3.2: Diagrama do padrão MVP



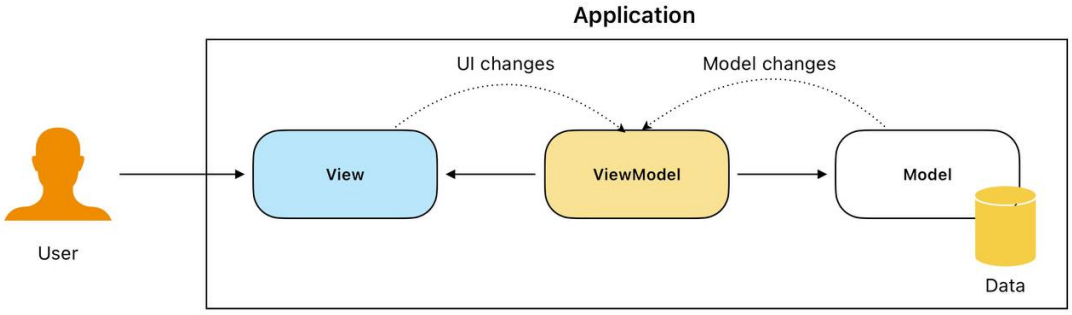
Fonte : Adaptada de [Scott Jr, 2015]

3.2.3 Model-View-ViewModel

O padrão Model-View-ViewModel (MVVM) é frequentemente usado em frameworks e bibliotecas de desenvolvimento de interfaces de usuário, como o Angular e o React. O ViewModel é responsável por fornecer dados e lógica de apresentação específica para a

View. A View se vincula ao ViewModel e atualiza automaticamente quando os dados do ViewModel são alterados. O MVVM facilita a criação de interfaces de usuário reativas e simplifica a manipulação de estados.

Figura 3.3: Diagrama do padrão MVVM



Fonte : Adaptada de [Scott Jr, 2015]

3.3 Funcionamento

O funcionamento de uma SPA é baseado em tecnologias como HTML, CSS e JavaScript, quando um usuário acessa a aplicação, o navegador carrega o arquivo HTML principal, que contém a estrutura básica da página e os scripts JavaScript necessários. Em seguida, o JavaScript assume o controle e manipula as interações do usuário e as requisições de dados.

A navegação ocorre em vez de carregar páginas separadas, o JavaScript é responsável por recuperar os dados necessários do servidor, geralmente usando APIs RESTful ou GraphQL. Esses dados são então renderizados e exibidos na página dinamicamente, sem a necessidade de recarregar a página inteira. Isso proporciona uma experiência de usuário mais rápida e suave, sem interrupções desnecessárias.

Além disso, as SPAs geralmente utilizam frameworks e bibliotecas JavaScript, como React, Angular ou Vue.js, para facilitar o desenvolvimento e a organização do código [Fink et al., 2014]. Essas ferramentas oferecem recursos avançados, como manipulação de estados, roteamento e componentização, que ajudam a construir SPAs escaláveis e de fácil manutenção.

É importante mencionar que, como toda a aplicação está concentrada em uma única página, o carregamento inicial pode exigir um tempo maior, especialmente se houver muitos recursos e dados a serem carregados. No entanto, uma vez carregada, a interação do usuário é mais ágil, pois as atualizações ocorrem de forma assíncrona, sem a necessidade de recarregar a página.

Capítulo 4

Navegação de Robôs Autônomos

De acordo com Meystel (1991), compreende-se que um sistema de Robôs Móveis Autônomos (AMR, na sigla em inglês) não requer qualquer forma de comunicação direta para compreender suas tarefas e executá-las de forma adequada. Portanto, para habilitar a autonomia de um robô, é necessário fazer uso de uma variedade de técnicas e ferramentas disponíveis na área. Essas abordagens e recursos são fundamentais para permitir que o robô tome decisões e realize suas tarefas de forma independente, sem a necessidade de intervenção humana constante. Neste capítulo, abordaremos as técnicas , conceitos e ferramentas que serão utilizadas nesta dissertação para viabilizar a navegação autônoma de um robô móvel diferencial.

4.1 Robôs diferenciais

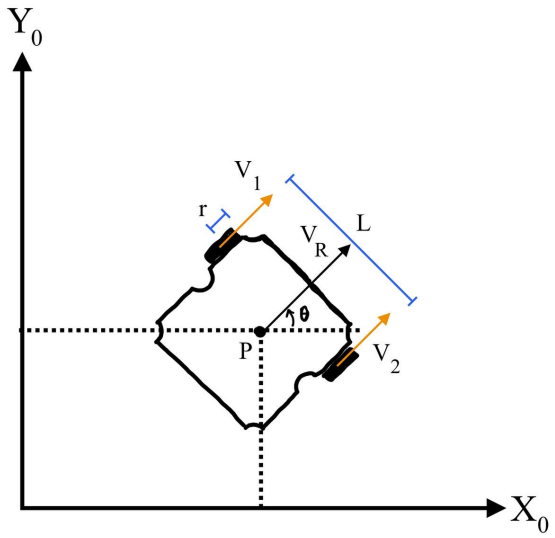
Robôs diferenciais são uma classe de robôs móveis que têm como sua principal característica a configuração mecânica de duas rodas independentes. Essa configuração permite que os robôs diferenciais se movam e controlem sua direção de forma diferencial, variando as velocidades das rodas esquerda e direita, como está representado na Figura 4.1.

Essa versatilidade de movimento torna os robôs diferenciais ideais para manobras precisas, curvas fechadas e rotações no próprio eixo. Além disso, a simplicidade mecânica dos robôs diferenciais oferece benefícios como maior robustez, menor custo de construção e manutenção simplificada, tornando-os amplamente utilizados em diversas aplicações, desde exploração espacial até sistemas de entrega autônoma.

Matematicamente, pode-se representar a cinemática direta de um robô diferencial através das seguinte equação analisando a Figura 4.1, temos que $x' = x_0 + V_R \cos(\theta)$, $y' = y_0 + V_R \sin(\theta)$ e $\theta(t) = \theta(0) + \omega * t$.

Onde x' e y' representam as velocidades do robô no tempo t, x(0) e y(0) são as coordenadas iniciais, θ(t) é o ângulo de orientação do robô no tempo t, $\theta(0)$ é o ângulo inicial, ω é a velocidade angular do robô e t é o tempo decorrido.

Figura 4.1: Representação do problema SLAM



Fonte : Elaborada pelo autor

As equações descrevem o movimento do robô diferencial em um plano bidimensional, levando em consideração a velocidade linear descrita pela Equação 4.1 e angular ω descrita pela Equação 4.2, determinadas a partir das velocidades das rodas $V1$ e $V2$, raio r e distâncias entre as rodas L.

$$V_R = r\frac{V1 + V2}{2} \tag{4.1}$$

$$\omega = r\frac{V2 - V1}{L} \tag{4.2}$$

4.2 SLAM

Os desafios centrais da navegação autônoma envolvem a resolução do sistema de localização do robô, o mapeamento do ambiente e o planejamento da trajetória. Com o intuito de abordar esses desafios de forma simultânea, em 1986, durante a Conferência de Robótica e Automação do IEEE em São Francisco, foi proposta uma técnica que visava realizar a localização e o mapeamento de maneira conjunta [Pritsker et al., 1986].

Matematicamente, o problema pode ser formulado como a estimação conjunta da trajetória do robô, denotada por $X_T = \{x_1, x_2, \cdots, x_T\}$, onde x_t representa a posição e orientação do robô no tempo t e o conjunto de todos os marcos de referência denotado por $M = \{m_1, m_2, \cdots, m_n\}$, onde é um vetor que descreve a localização do i-ésimo marco de referência, cuja posição verdadeira é considerada constante ao longo do tempo. [Durrant-Whyte and Bailey, 2006].

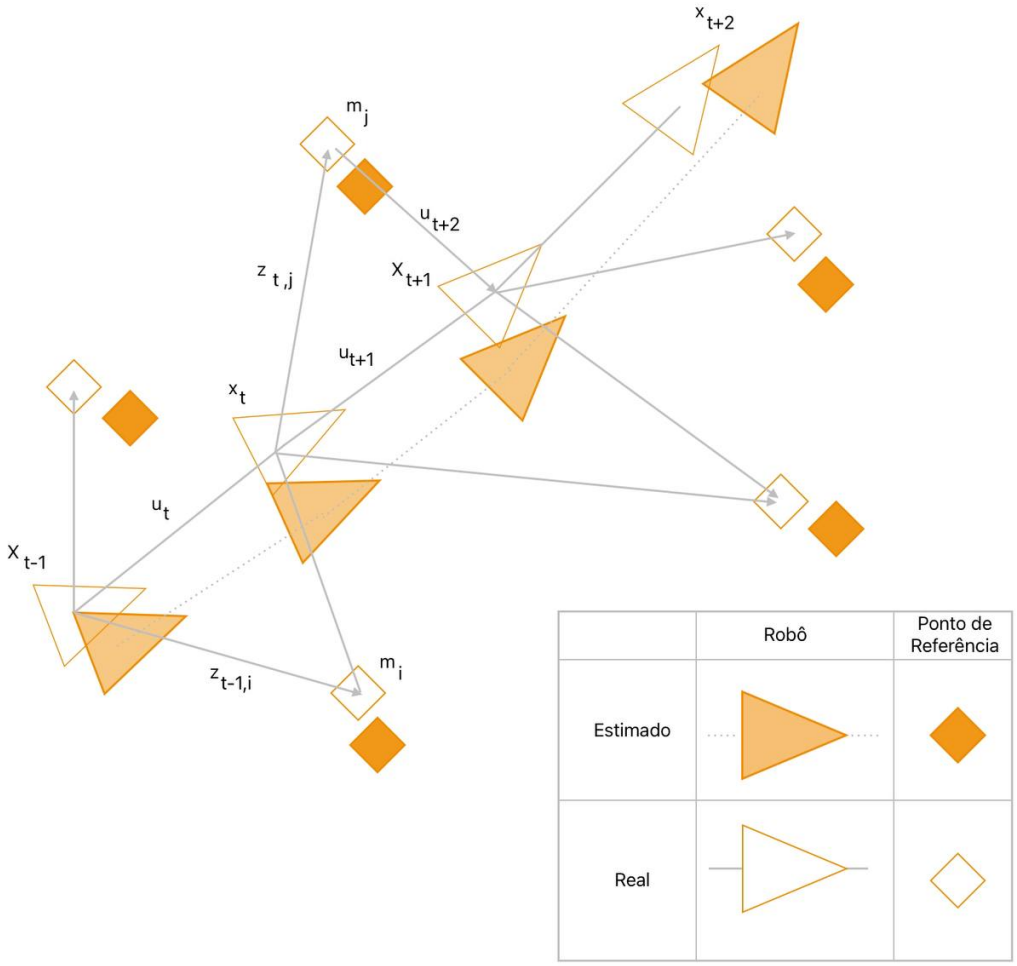
O problema pode ser resumido na seguinte expressão:

$$p(X_T, M|Z_T, U_T) \tag{4.3}$$

onde:

- $p(X_T, M|Z_T, U_T)$ é a distribuição de probabilidade conjunta da trajetória do robô e do mapa do ambiente, dado um conjunto de observações Z_T e comandos de controle U_T.
- $Z_T = z_1, z_2, ..., z_T$ são as observações coletadas pelo robô, como leituras de sensores, imagens ou dados de GPS.
- $U_T = u_1, u_2, ..., u_T$ são os comandos de controle aplicados ao robô, como velocidades lineares e angulares.
- T é o tempo final, tende ao ∞.

Figura 4.2: Representação do problema SLAM

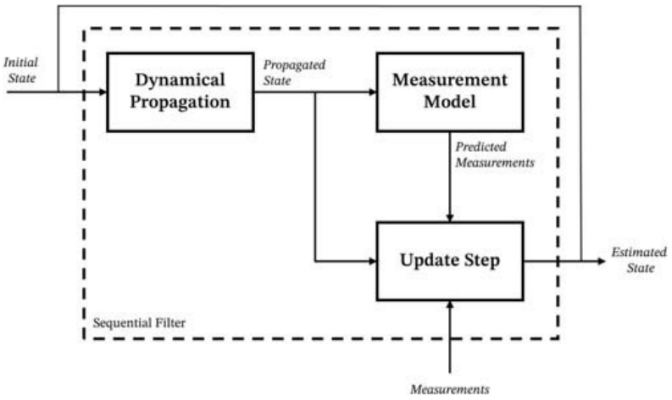


Fonte : Elaborada pelo autor

O objetivo é encontrar a estimativa de máxima verossimilhança ou a distribuição de probabilidade posterior de X_T e M, dada as observações e comandos de controle.

Há duas abordagens para tratar o problema *online*, onde o robô realiza a localização de maneira simultânea à medida que percorre o ambiente. Ele utiliza-se das informações dos sensores em tempo real para estimar a pose do robô e construir o mapa. O global, por outro lado armazena os dados dos sensores para calcular posteriormente de maneira offline utilizando de algoritmos mais complexos e recursos computacionais para construir um mapa mais preciso e consistente do ambiente, levando em consideração informações de várias posições e momentos no tempo [Bailey and Durrant-Whyte, 2006]. Isso é geralmente resolvido usando métodos de filtragem conhecidos na literatura como filtros sequenciais. Eles são uma classe de filtros Bayesianos que lidam com problemas de estimação em tempo real, onde as informações disponíveis são atualizadas sequencialmente ao longo do tempo. Esses filtros são particularmente úteis quando as características do sistema são não lineares e/ou o ruído é não gaussiano, uma representação geral desses filtros pode ser observada na figura 4.3.

Figura 4.3: Representação de um filtro sequencial



Fonte : [Pesce et al., 2023]

O processo de atualização é realizado em duas etapas principais: predição e correção. Na etapa de predição, o filtro estima o estado futuro do sistema com base nas informações anteriores e no modelo dinâmico do sistema. Em seguida, na etapa de correção, o filtro atualiza a estimativa do estado com base nas observações atuais. Essa atualização é realizada por meio da combinação das estimativas anteriores com as informações observadas, levando em consideração o modelo de medição do sistema. Através do processo iterativo de predição e correção, os filtros sequenciais refinam continuamente as estimativas do estado do sistema à medida que novas observações são obtidas. Isso permite que os filtros se adaptem a mudanças nas condições do sistema e forneçam estimativas mais precisas em tempo real.

Como discutido em [Siciliano et al., 2008], existem três principais abordagens para lidar com o problema de localização e mapeamento simultâneos. Essas abordagens são

amplamente utilizadas na área de robótica e são conhecidas como Filtro de Partículas, Filtro de Kalman Estendido e SLAM baseado em grafos. Cada uma dessas abordagens possui suas próprias características e técnicas específicas para estimar a posição do robô e construir um mapa do ambiente. Esses métodos têm sido amplamente estudados e aplicados em diferentes cenários [Placed et al., 2023].

4.2.1 Filtro de partículas

O filtro de partículas é um método amplamente utilizado na área de estimação de estado, sendo especialmente aplicado em problemas de localização e mapeamento simultâneos (SLAM) em robótica e em sistemas de rastreamento [Zhao et al., 2006, Hui-min et al., 2008, Yu et al., 2013, Ko and Kim, 2012, Yan et al., 2017]. Esse filtro pertence à classe dos filtros Bayesiano não lineares e é particularmente útil em situações em que o modelo do sistema e as medidas são não lineares ou apresentam ruído [Djuric et al., 2003]

O filtro de partículas funciona por meio de um conjunto de partículas que representam possíveis estados do sistema como apresentado pela equação 4.4

$$\chi_t := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{4.4}$$

onde M é o número de partículas.

A ideia principal de um filtro de partículas é utilizar um conjunto de partículas para representar possíveis estados de um sistema. Cada partícula possui uma estimativa do estado atual do sistema e é atualizada com base em medições e na propagação temporal. O filtro de partículas calcula a crença dos estados atuais, representada por $bel(x_t)$, a partir da crença dos estados anteriores, representada por $bel(x_{t-1})$. Esse cálculo é realizado através de um processo de duas etapas como apresentado na seção anterior

Na etapa de previsão, as partículas são propagadas no tempo utilizando o modelo dinâmico do sistema. Cada partícula é atualizada de acordo com as características do sistema e as incertezas associadas. Isso permite que as partículas capturem a evolução do estado do sistema ao longo do tempo.

Na etapa de correção, as partículas são atualizadas com base nas medições disponíveis. Cada partícula é ponderada de acordo com sua compatibilidade com as medições observadas. As partículas que se ajustam melhor às medições têm pesos maiores, enquanto as menos compatíveis têm pesos menores. Isso permite que as partículas representem de forma mais precisa o estado atual do sistema, levando em consideração as informações provenientes das medições.

Após a etapa de correção, é realizada uma re-amostragem para selecionar as partículas mais representativas, sendo baseada nos pesos das partículas, de modo que aquelas com pesos maiores têm maior probabilidade de serem selecionadas. Isso ajuda a evitar a

degeneração do conjunto de partículas, garantindo que as partículas mais relevantes sejam preservadas e as menos relevantes sejam descartadas.

4.2.2 Filtro de Kalman estendido (EFK)

O filtro de Kalman é uma técnica de estimativa estatística amplamente utilizada para estimar o estado de um sistema dinâmico a partir de medições ruidosas. Ele combina informações provenientes de duas fontes principais: a estimativa anterior do estado e a medição mais recente do sistema. Kalman assume que o sistema segue um modelo linear com ruído aditivo gaussiano e utiliza a teoria das probabilidades para calcular a estimativa mais provável do estado atual, levando em consideração a incerteza das medições e do modelo do sistema.

Em [Choset et al., 2005] descreve o sistema com as seguintes equações

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k) \tag{4.5}$$

$$y(k) = H(k)x(k) + w(k) \tag{4.6}$$

Considere o vetor $x(k)$ que representa o estado completo do sistema. O vetor $u(k)$ é a entrada do sistema, que pode incluir informações como velocidades, comandos de torque ou forças a serem aplicadas ao robô. O vetor $y(k)$ é a saída do sistema, que contém os valores dos sensores do sistema.

A matriz $F(k)$ descreve a dinâmica do sistema, representando a evolução do estado ao longo do tempo. A matriz $G(k)$ relaciona a entrada $u(k)$ com a mudança no estado do sistema. O vetor $v(k)$ é conhecido como ruído do sistema e é assumido como um ruído branco gaussiano com média zero. Ele representa as perturbações ou incertezas que afetam o sistema. A matriz de covariância $V(k)$ descreve a variabilidade ou dispersão desses ruídos ao longo do tempo. A matriz $H(k)$ descreve o mapeamento dos estados do motor para as saídas. Essa matriz relaciona os estados internos do motor, que podem incluir variáveis como posição, velocidade ou corrente, às medições obtidas pelos sensores. O vetor $w(k)$ representa o ruído de medição, que é assumido como um ruído branco gaussiano com média zero. Esse ruído afeta as medições obtidas pelos sensores e representa as incertezas e perturbações associadas ao processo de medição. A matriz de covariância do ruído de medição, geralmente denotada por $W(k)$, descreve a variabilidade ou dispersão desse ruído ao longo do tempo.

Assim como o filtro de partículas, o cálculo será realizado em duas etapas como mostram as equações abaixo, sumarizadas em

- Predição:

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k) \tag{4.7}$$

$$P(k+1|k) = F(k)P(k|k)F(k)^T + V(k) \tag{4.8}$$

- Correção

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k+1) + Rv \tag{4.9}$$

$$P(k+1|k+1) = P(k+1|k) - RH(k+1)P(k+1|k) \tag{4.10}$$

onde S é o resíduo da covariância e R é o ganho ótimo do Filtro de Kalman como mostram as Equações 4.11 e 4.12

$$v = y(k+1) - H(k+1)x(k+1|k)) \tag{4.11}$$

$$S = H(k+1)P(k+1|k)H(k+1)^T + W(k+1) \tag{4.12}$$

$$R = P(k+1|k)H(k+1)^T S^{-1} \tag{4.13}$$

O EKF (*Extended Kalman Filter*) é uma extensão do Filtro de Kalman que é aplicada quando o sistema dinâmico e as medições possuem características não lineares. O EKF lineariza o sistema e as equações de medição em torno da estimativa atual do estado e em seguida, é aplicado na versão linearizada do sistema para estimar o estado atual.

Considerando o seguinte sistema abaixo onde x,y,u,v são o mesmos das equações 4.5 e 4.6. E as funções f e h ambas são continuas e diferenciáveis em $x(k)$:

$$x(k+1) = f(x(k), u(k), k) + v(k) \tag{4.14}$$

$$y(k) = h(x(k), k) + w(k) \tag{4.15}$$

Temos que as equações de predição e correção para o filtro de kalman estendido são :

- Predição:

$$\hat{x}(k+1|k) = f(\hat{x}(k|k), u(k), k) \tag{4.16}$$

$$P(k+1|k) = F(k)P(k|k)F(k)^T + V(k) \tag{4.17}$$

onde

$$F(k) = \frac{\delta f}{\delta x}, x = \hat{x}(k|k) \tag{4.18}$$

- Correção

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k+1) + Rv \tag{4.19}$$

$$P(k+1|k+1) = P(k+1|k) - RH(k+1)P(k+1|k) \tag{4.20}$$

onde

$$v = y(k+1) - H(k+1)x(k+1|k)) \tag{4.21}$$

$$S = H(k+1)P(k+1|k)H(k+1)^T + W(k+1) \tag{4.22}$$

$$R = P(k+1|k)H(k+1)^T S^{-1} \tag{4.23}$$

e

$$H(k+1) = \frac{\delta h}{\delta x}, x = \hat{x}(k+1|k). \tag{4.24}$$

Embora o EKF seja amplamente utilizado em aplicações práticas, é importante destacar que a linearização introduz erros de aproximação, tornando a estimativa do estado menos precisa em comparação com o Filtro de Kalman convencional em sistemas lineares.

No entanto, é uma técnica valiosa quando se trata de estimar o estado de sistemas não lineares em tempo real, sendo amplamente aplicado em áreas como robótica, navegação, processamento de sinais e controle de sistemas não lineares [Bonnabel, 2012, Das et al., 2021, Zhang et al., 2018].

4.3 ROS

O ROS[1], do inglês *Robot Operating System*, é uma plataforma de software amplamente utilizada no desenvolvimento de robôs. Ele oferece um conjunto abrangente de ferramentas, bibliotecas e recursos que facilitam a criação de sistemas robóticos complexos. Através de uma ampla gama de funcionalidades e APIs, ele fornece um ecossistema robusto para desenvolvedores de robótica.

A Figura 4.4 ilustra a complexidade e diversidade do ecossistema da plataforma, ela engloba uma variedade de componentes, incluindo bibliotecas de código aberto, pacotes de software, ferramentas de simulação, ambientes de desenvolvimento e recursos de comunicação. Esses elementos se combinam para permitir o desenvolvimento e a operação eficiente de sistemas robóticos.

Um dos principais pontos fortes do ROS é sua arquitetura flexível e modular. Ele adota uma abordagem baseada em nós, em que cada componente funcional do sistema é encapsulado em um nó independente. Esses nós podem se comunicar uns com os outros por meio de troca de mensagens, serviços e tópicos.

Além disso, possui uma comunidade ativa e colaborativa, o que resulta em uma ampla gama de recursos disponíveis para os desenvolvedores. Essa comunidade contribui com pacotes de software, tutoriais, exemplos e suporte técnico, facilitando o desenvolvimento de soluções robóticas inovadoras [Estefo et al., 2019].

[1] https://www.ros.org

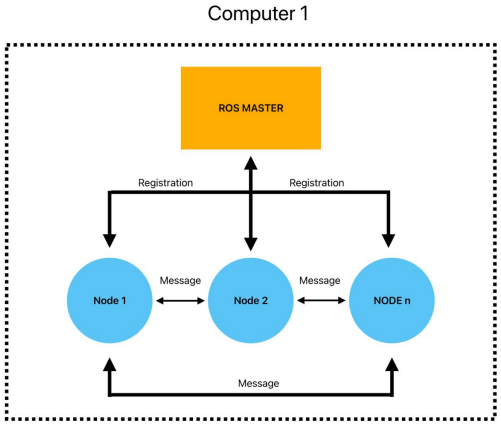
Figura 4.4: Representação do ecositema do ROS

Fonte : Website do ROS [2]

4.3.1 Arquitetura

Sua arquitetura é ilustrada de forma simplificada na Figura 4.5. No centro temos o ROS *Master*, que desempenha um papel fundamental ao permitir a localização e a comunicação ponto a ponto entre os diferentes componentes.

Figura 4.5: Representação de uma arquitetura do ROS



Fonte : Website do ROS [3]

Ao redor do ROS Master, encontramos os principais elementos. Os nós são os blocos de construção fundamentais do sistema, representando processos executáveis independentes que desempenham tarefas específicas. Eles podem se comunicar entre si através do uso de tópicos e serviços.

Os tópicos são canais de comunicação unidirecionais nos quais os nós podem publicar mensagens para compartilhar informações ou se inscrever para receber mensagens de outros nós. Essa forma de comunicação assíncrona permite a troca de informações em tempo real e facilita a integração e o compartilhamento de dados entre os nós.

Os serviços, por sua vez, permitem a comunicação síncrona entre os nós. Eles são semelhantes a chamadas de função remotas, onde um nó solicita a execução de um serviço específico em outro nó e espera pela resposta. Isso permite interações mais complexas e baseadas em solicitação-resposta entre os nós.

Além disso, o ROS utiliza o conceito de mensagens, que são estruturas de dados usadas para trocar informações entre os nós, fornecendo uma variedade de tipos de mensagem pré-definidos para diferentes finalidades, como dados de sensores, comandos de atuadores,

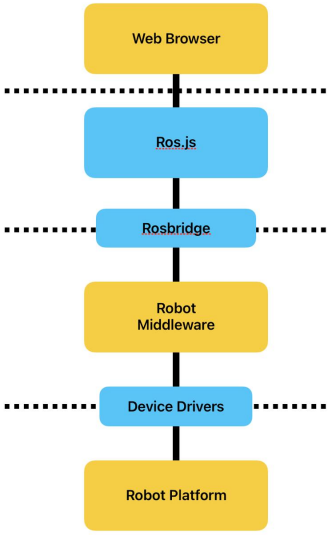
informações de localização, entre outros. Também é possível criar mensagens personalizadas para atender às necessidades específicas do sistema robótico.

Por ser uma arquitetura modular, o ROS permite a conexão de outros componentes em diferentes computadores, desde que estejam inscritos no mesmo *Master*.

4.3.2 Rosbridge

Apresentada em [Crick et al., 2017], o Rosbridge é uma biblioteca e um conjunto de ferramentas que permite a comunicação entre o ROS e outras aplicações não relacionadas, permitindo que aplicativos e sistemas externos se comuniquem com o ROS e acessem os recursos do sistema robótico. Ele utiliza protocolos de comunicação padrão, como WebSocket e TCP/IP, para fornecer uma interface de comunicação bidirecional entre o ROS e os clientes externos.

Figura 4.6: Fluxo de trabalho utilizando *rosbridge*



Fonte : Adaptada de [Alexander et al., 2012]

No diagrama da Figura 4.6 observa-se um exemplo de um fluxo de trabalho no qual o *rosbridge* pode atuar, oferecendo uma camada de abstração que simplifica a interação com o ROS no ambiente web. Ela permite que os desenvolvedores aproveitem as funcionalidades do ROS, como a comunicação baseada em tópicos e serviços, diretamente em aplicativos web implementados em JavaScript. Com o roslibjs, é possível criar interfaces de usuário interativas e personalizadas, facilitando o controle e o monitoramento remotos de robôs através de uma experiência web fluida [Lee, 2012, Crick et al., 2017, Krupke et al., 2016].

4.3.3 Robot Web Tools

O Robot Web Tools [4] é um conjunto de ferramentas e bibliotecas desenvolvidas para facilitar a integração e a interação entre robôs e aplicativos web. Ele fornece um conjunto de recursos e APIs que permitem a comunicação e o controle de robôs por meio de uma interface web.

Essas ferramentas permitem que os desenvolvedores criem interfaces de usuário baseadas em web para controlar e monitorar robôs de forma remota. Elas fornecem recursos como visualização em 3D, controle de movimento, interação com sensores e atuadores, e compartilhamento de dados em tempo real [Alexander et al., 2012].

Os autores em [Casañ et al., 2015] utilizaram a ferramenta para desenvolver uma extensão web com o objetivo de ser utilizada em laboratórios remotos e robôs online baseados em ROS, com foco específico em aplicações educacionais. Para controlar um robô humanoide foi construída uma aplicação web.

Para controle de um robô humanoide foi desenvolvida uma arquitetura de software que incluem o uso de um controlador de corpo inteiro que permite ao usuário trabalhar remotamente e uma interface do usuário inovadora que é acessível por meio de qualquer navegador web moderno, incluindo aqueles em dispositivos moveis [Fok et al., 2016].

Além disso permite que os desenvolvedores compartilhem e reutilizem bibliotecas e componentes para acelerar o desenvolvimento de novas aplicações robóticas baseadas na web.

4.3.4 Pilha de Navegação

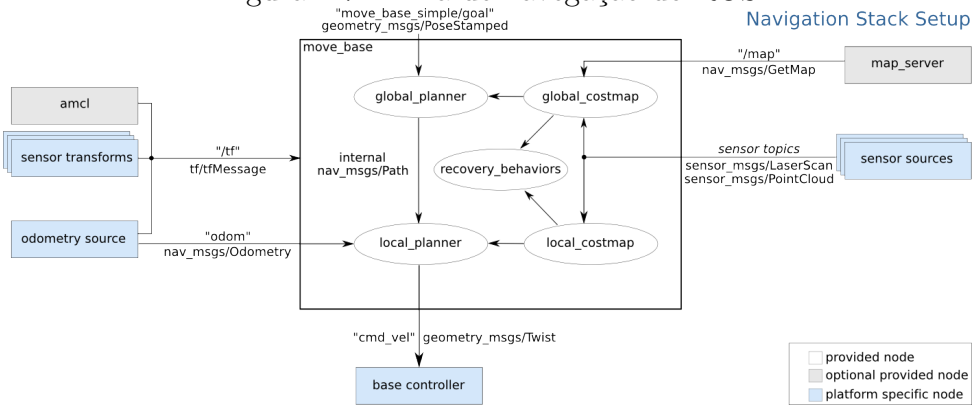
A pilha de navegação do ROS (Robot Operating System) é um conjunto de pacotes e algoritmos que fornecem recursos de navegação para robôs móveis, commo apresentada na Figura 4.7. Essa pilha é projetada para permitir que os robôs planejem trajetórias, evitem obstáculos e executem movimentos autônomos em ambientes desconhecidos ou dinâmicos.

A pilha de navegação é composta por vários componentes principais, incluindo:

- AMCL (*Adaptive Monte Carlo Localization*): É responsável pela localização do robô no ambiente, utilizando das medições dos sensores e cálculo da odometria do robô aplica-se filtro de partículas para estimar a posição e a orientação do robô em relação a um mapa do ambiente, criando uma nuvem de partículas como exemplificado nas Figura 4.8, 4.9 no inicio e no final do movimento respectivamente. Conforme o robô se desloca pelo mapa, é possível notar uma redução gradual nas incertezas associadas à sua localização. Isso ocorre devido à atualização contínua da confiança do robô em sua posição à medida que ele recebe informações perceptivas do ambiente.

[4] https://robotwebtools.github.io/

Figura 4.7: Pilha de navegação do ROS



Fonte : Wiki ROS

Figura 4.8: Visualização das nuvem de partículas gerados pelo tópico AMCL no inicio do movimento utilizando o Rviz.

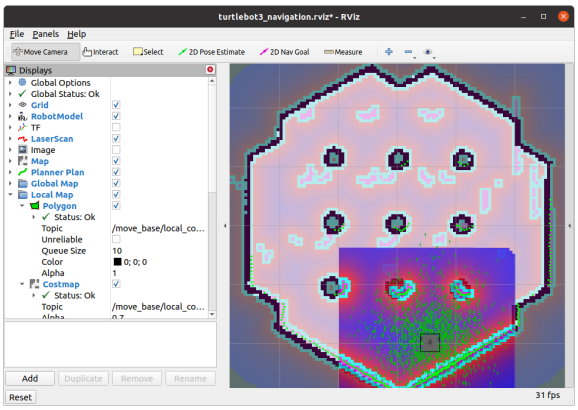
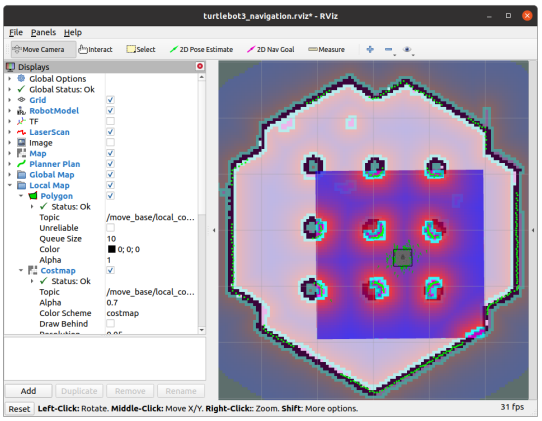


Figura 4.9: Visualização das nuvem de partículas gerados pelo tópico AMCL no final do movimento utilizando o Rviz.



Fonte : Wiki ROS

- Mapa de custo global e local (costmap) : Representa o mapa de custo do ambiente, que descreve a acessibilidade de diferentes regiões para o robô. O mapa de custo global é construído para permitir o planejamento de trajetórias (*global_planner*) de longo prazo em todo o ambiente, considerando obstáculos estáticos e restrições de movimento. Por outro lado, o mapa de custo local é utilizado para o planejamento (*base_local_planner*) de curto prazo em uma janela de atuação específica, levando em consideração a detecção e evasão de obstáculos dinâmicos. Esses mapas de custo são fundamentais para garantir a navegação segura e eficiente do robô, permitindo a geração de trajetórias livres de colisões e a tomada de decisões adaptativas em tempo real.

- Pacote *move_base*: É o componente central da pilha de navegação. Ele coordena todos os outros componentes e fornece uma interface de alto nível para controle de movimento do robô. O pacote recebe comandos de navegação, como um objetivo de destino, e utiliza os componentes mencionados anteriormente para planejar e

executar a trajetória do robô.

A pilha de navegação do ROS é altamente modular e flexível, permitindo a customização e configuração de acordo com as necessidades específicas do robô e do ambiente. Ela é amplamente utilizada em aplicações de robótica móvel, como robôs de serviço, robôs autônomos e robôs de inspeção, para realizar tarefas de navegação de forma autônoma e segura.

Capítulo 5

Resultados

Neste capítulo são fornecidos os detalhes da implementação para desenvolver uma aplicação web destinada à teleoperação de um robô móvel baseado no ROS (Robot Operating System). O *framework* de teleoperação é dividido em dois modos: compartilhado e supervisório. Ambos os modos utilizam o sensoriamento do robô, neste caso, o Lidar, para alertar o operador ou permitir que o robô execute tarefas com autonomia local. O trabalho descrito neste capítulo foi testado no robô modelo Waffle da linha Turtlebot 3. Foram realizados testes práticos para validar o funcionamento do framework, demonstrando sua capacidade de controlar o robô remotamente por meio da aplicação web. Os códigos implementados e vídeos da utilização estão disponíveis no repositório do github AndressaM[1].

5.1 Turtlebot 3 - Waffle

O Turtlebot3 Waffle é um robô móvel desenvolvido pela ROBOTIS[2] em colaboração com a Open Robotics e a OSRF (Open Source Robotics Foundation).

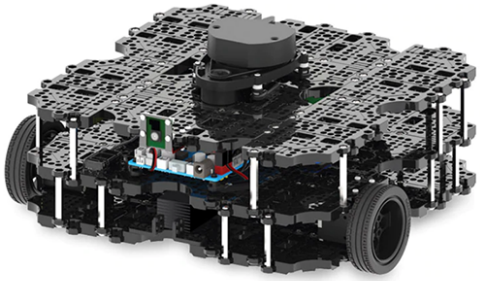


Figura 5.1: Turtlebot3 modelo *waffle*

O Turtlebot3 Waffle possui um formato circular com aproximadamente 22 centímetros

[1]https://github.com/AndressaM/robot-teleoperation
[2]https://www.robotis.us/

de diâmetro e 23 centímetros de altura. Sua estrutura é compacta e leve, o que facilita a mobilidade em ambientes internos. O robô é equipado com uma plataforma de movimentação diferencial, o que significa que ele possui duas rodas motorizadas independentes que permitem controlar sua velocidade linear e velocidade angular.

O sistema de sensoriamento do inclui um sensor LiDAR (Light Detection and Ranging) de 360 graus, que permite ao robô fazer leituras de distância ao seu redor e mapear o ambiente. Além disso, possui uma câmera montada em sua estrutura para capturar imagens e realizar tarefas de visão computacional.

O Waffle é projetado para ser compatível com o ROS [Martínez, 2021], devido às suas características e versatilidade, ele é amplamente utilizado em ambientes acadêmicos e de pesquisa, bem como em projetos de aprendizado de máquina e robótica educacional por fornecer uma plataforma acessível e flexível para estudantes e pesquisadores desenvolverem soluções robóticas em diversos campos de estudo [Safin et al., 2021, de Assis Brasil et al., 2020, Amsters and Slaets, 2020].

5.2 Implementação

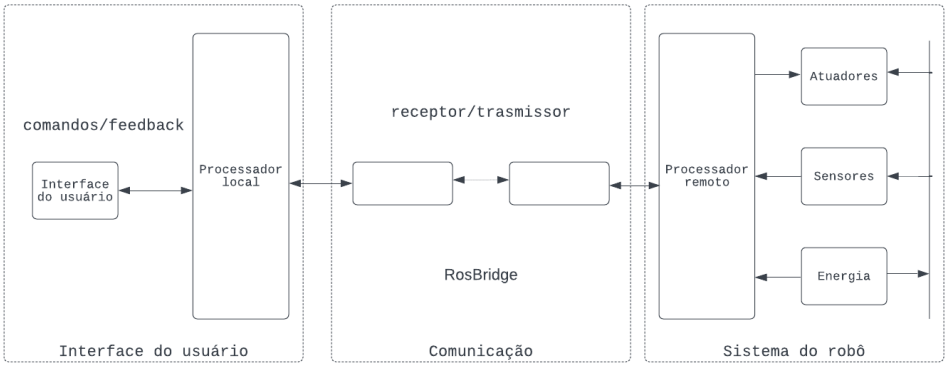
De acordo [Opiyo et al., 2021], a arquitetura de teleoperação consiste em três elementos: o robô, o canal de comunicação e a estação do operador. Alinhando essa arquitetura com o presente trabalho, a interface em que o usuário irá comandar e receber *feedback* do robô será desenvolvida em uma plataforma web, implementada utilizando o *React*, baseado na implementação disponível de Anis Koubaa [Koubaa, 2022]. A comunicação utilizará um WebSocket fornecido pelo ROS, especificamente o Rosbridge [3].

O Rosbridge irá fornecer um protocolo de comunicação que permite a qualquer cliente enviar mensagens JSON para publicar ou se inscrever em tópicos e chamar serviços do ROS. Por meio dessa comunicação, o robô receberá e enviará mensagens em seus tópicos ativos, como aqueles relacionados a sensores, atuadores e energia conforme ilustrado na Figura 5.2.

As bibliotecas utilizadas, incluindo aquelas que são executadas remotamente no robô, são implementadas no ROS Noetic. O ROS Noetic é uma versão específica do ROS lançada em maio de 2020. Essa versão é compatível com sistemas operacionais como Ubuntu 20.04 (Focal Fossa) e Debian Buster. No momento de desenvolvimento deste trabalho é a versão mais estável para utilização da coleção de ferramentas disponíveis pelo *robot web tools*.

[3] https://github.com/RobotWebTools/rosbridge_suite

Figura 5.2: Representação da arquitetura do sistema

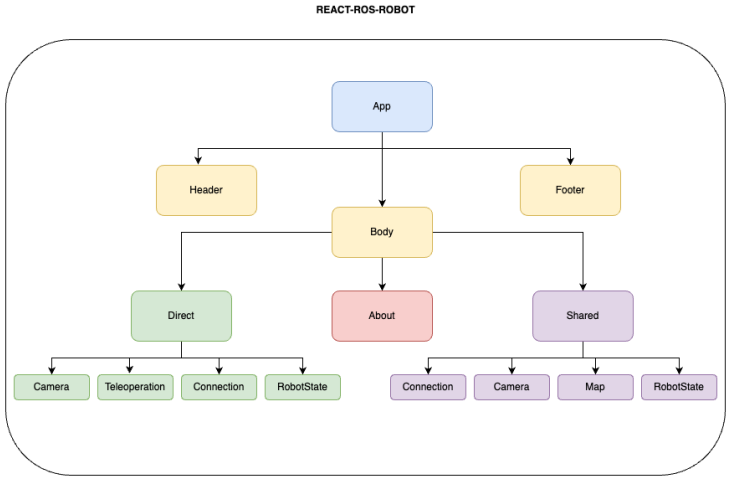


Fonte : Elaborada pelo autor

5.3 Aplicação Web

A aplicação foi desenvolvida utilizando a biblioteca React, que é uma biblioteca JavaScript que simplifica a criação de componentes de interface do usuário e a construção de interfaces complexas do tipo SPA. A Figura 5.3 mostra a hierarquia de componentes utilizados na construção da interface. É importante observar que alguns componentes foram reutilizados em várias páginas, graças à modularização, o que permite um maior reaproveitamento de código.

Figura 5.3: Visualização da aplicação em uma árvore de componentes



Fonte : Elaborada pelo autor

Além dos componentes, a aplicação também incorpora scripts em JavaScript, juntamente com módulos do ROS, que são utilizados para viabilizar a comunicação entre a interface e o robô. Essa abordagem resulta em uma estrutura completa da aplicação, a qual é exemplificada de forma visual na Figura 5.4.

Figura 5.4: Visualização da estrutura completa da aplicação

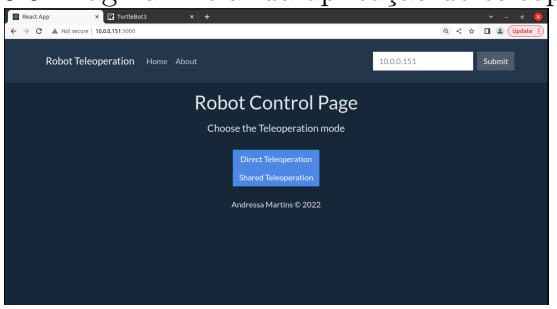
```
REACT-ROS-ROBOT
├── README.md
├── node_modules
├── package.json
├── react-ros-robot
├── .gitignore
├── public
│   ├── js
│   ├── ros-nav2d-js
│   │   ├── navigator
│   │   │   ├── ImageMapClientNav.js
│   │   │   ├── Navigator.js
│   │   │   └── OcuppancyGridClientNav.js
│   │   └── Nav2D.js
│   ├── easeljs
│   ├── eventemitter2.min
│   ├── mjpegcanvas.js
│   ├── mjpegcanvas.min.js
│   ├── ros2d.js
│   └── roslib.js
└── src
    ├── bootstrap
    │   ├── bootstrap.min.css
    │   └── bootstrap-superhero.css
    ├── components
    ├── scripts
    │   └── config.js
    ├── App.js
    ├── index.css
    ├── index.js
    ├── reportWebVitals.js
    └── setupTests.js
```

Fonte : Elaborada pelo autor

5.4 Interface

A interface desta aplicação está organizada em quatro rotas principais. A página inicial apresenta um menu que permite ao usuário selecionar o tipo de teleoperação a ser realizado. Além disso, uma barra superior exibe um menu principal, conforme ilustrado na Figura 5.5.

Figura 5.5: Página inicial da aplicação de teleoperação

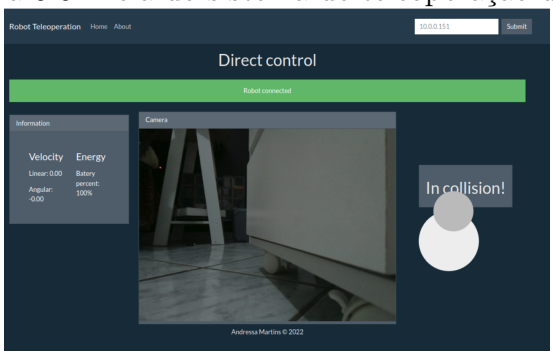


Fonte : Elaborada pelo autor

A página de teleoperação direta contém o estado do robô, isto é se está conectado ou não, a visualização da câmera do robô em tempo real e um *joystick* para controle de movimento do robô e um aviso em caso do robô estar próximo a obstáculo alertando sobre possível colisão como ilustrado na Figura 5.6.

Assim como na tela anterior, na tela de teleoperação compartilhada são apresentados o estado do robô e a visualização da câmera. No entanto, agora é possível observar um mapa pré-carregado por meio de um tópico externo do ROS. Além disso, nesse mapa, é exibida a posição atual do robô representada por um triângulo laranja juntamente com

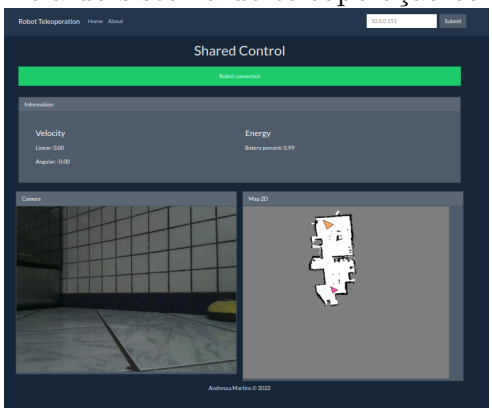
Figura 5.6: Tela do sistema de teleoperação direta



Fonte : Elaborada pelo autor

a posição final representada por um triângulo vermelho. Essas informações são dispostas no mapa com atualização em tempo real como mostra a Figura 5.7.

Figura 5.7: Tela do sistema de teleoperação compartilhado



Fonte : Elaborada pelo autor

Por ser uma aplicação baseada na web, ela é multiplataforma, oferecendo versatilidade no acesso aos recursos. A Figura 5.8 apresenta a mesma aplicação em um dispositivo móvel.

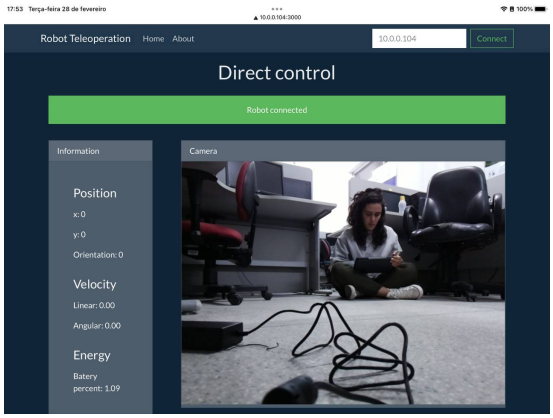
Quando não há conexão com o robô as telas apresentam a falta de conexão através da barra de estado como mostra as Figura 5.9 e 5.10, assim não sendo possível mais observar seus estados, mapa e câmera.

5.5 Teleoperação

Após a conclusão da aplicação, foram realizados testes de execução dos dois métodos propostos neste trabalho. Para realizar essas atividades, é necessário que o robô esteja com todos os tópicos ativos. Isso pode ser alcançado iniciando a chamada do arquivo de configuração (bringup) do robô, cujo procedimento é detalhado em TurtleBot 3[4].

[4]https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/

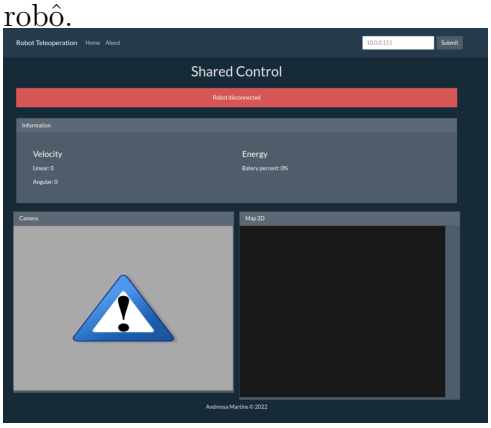
Figura 5.8: Página de controle direto da aplicação de teleoperação aberta em um dispositivo móvel



Fonte : Elaborada pelo autor

Figura 5.9: Tela de teleoperação direta sem conexão com o robô

Figura 5.10: Tela de teleoperação compartilhada sem conexão com o robô.



Fonte : Elaborada pelo autor

Após a inicialização do robô, o computador responsável por manter o nó master do ROS, que é o ponto central da comunicação de toda a aplicação, deve ter seus tópicos ativos. As chamadas desses tópicos estão presentes e descritas no repositório deste trabalho.

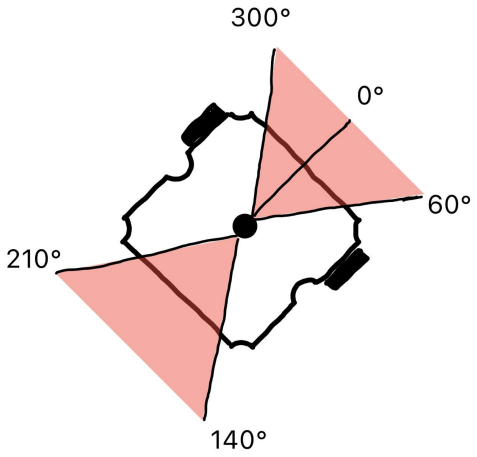
5.5.1 Controle do robô com um JoyStick virtual

Neste cenário, a interface está operando no modo de teleoperação direta, como ilustrado anteriormente na Figura 5.6. O usuário utiliza o *joystick* virtual para controlar o movimento do robô. A lógica implementada mapeia os movimentos do robô em termos de velocidades angulares e lineares, de acordo com a movimentação do *joystick*. Esses valores são então enviados para o nó *cmd_vel*, responsável por publicar as velocidades recebidas nos motores do robô.

A leitura dos dados do Lidar é constantemente disponibilizada pelo tópico *scan*, fornecendo informações sobre o ambiente ao redor do robô. Esses dados são analisados

para evitar colisões durante a teleoperação, alertando o operador sobre a presença de obstáculos. Além do alerta, foi implementada uma medida de segurança contra colisões, em que o sistema impede o movimento na direção em que um obstáculo é detectado. Essa verificação é realizada através da análise da distância mínima obtida pelo Lidar em intervalos angulares à frente e atrás do robô, conforme ilustrado na Figura 5.11. É importante ressaltar que, para o robô *Turtlebot3*, o ângulo 0 graus do Lidar corresponde à direção frontal do robô.

Figura 5.11: Intervalos de ângulos do lidar



Fonte : Elaborada pelo autor

A Figura 5.12 apresenta de maneira gráfica o sistema com todos os nós e tópicos ativos do sistema para este cenário, este gráfico foi gerado a partir de uma ferramenta do disponível no ROS conhecida como *rqt_graph*[5]. O nós são representados pelos círculos e os tópicos pelos itens quadrados. As linhas com direção apresenta o sentido da comunicação e as conexões entres os componentes do ROS.

A tabela 5.1 apresenta as descrições dos nós e tópicos do sistema.

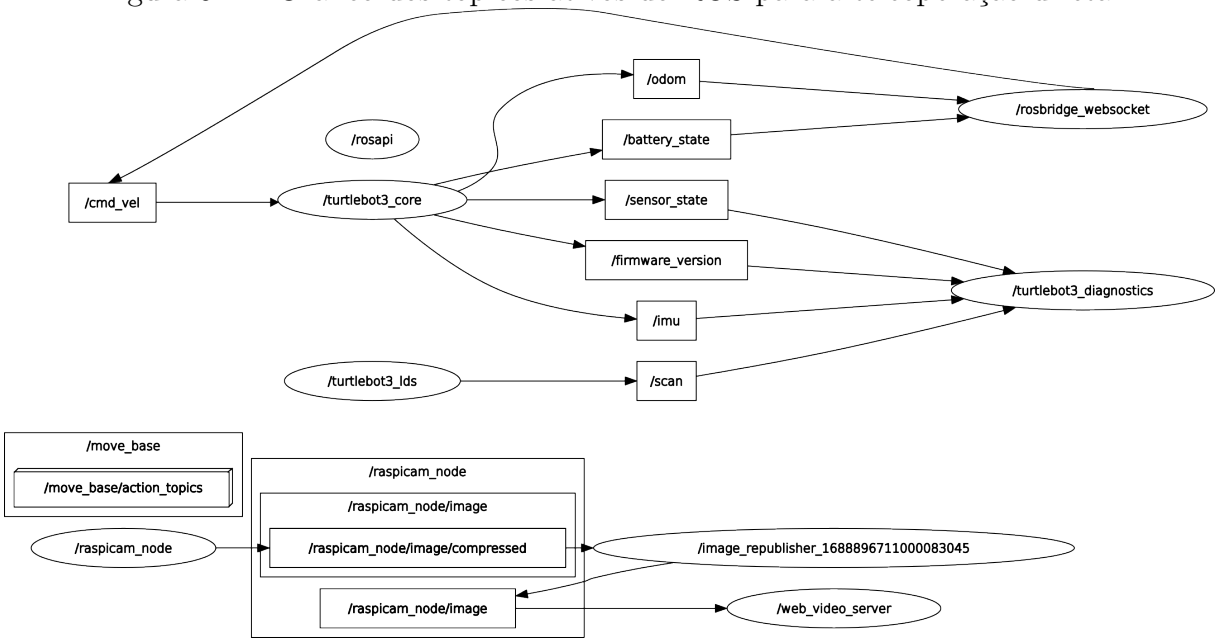
5.5.2 Controle supervisório

Neste cenário de teleoperação, o objetivo é realizar a navegação autônoma em um mapa conhecido utilizando o pacote *move_base*, que faz parte da pilha de navegação do ROS. O pacote é responsável por planejar trajetórias e controlar o movimento do robô de forma autônoma, levando em consideração o mapa fornecido como apresentado na Figura 5.13.

O usuário é habilitado a selecionar o objetivo final do robô com base nas informações visuais do mapa e na sua posição atual, por meio de interação direta ao clicar em um ponto válido do mapa. Após receber o comando, o sistema utiliza os recursos do ROS, como nós e serviços, para calcular a trajetória do robô em direção ao objetivo global. Durante

[5]http://wiki.ros.org/rqt_graph

Figura 5.12: Gráfico dos tópicos ativos do ROS para a teleoperação direta.



Fonte : Elaborada pelo autor

turtlebot3_core	sensor_state	Tópico que contém os valores dos sensores montados no Turtlebot3
	battery_state	Contém a voltagem e o status da bateria
	odom	Contém as informações de odometria do Turtlebot3 com base no codificador e na IMU
	imu	Tópico que inclui a atitude do robô com base no sensor de aceleração e giroscópio
	firmware_version	Contém as informações de hardware, firmware e software do Turtlebot3
	cmd_vel	Controle a velocidade de translação e rotação da unidade robótica em m/s, rad/s
turltebot3_lds	scan	Tópico que confirma os valores de varredura do LiDAR montado no Turtlebot3
turltebot3_diagnostics	diagnostics	Contém informações de autodiagnóstico
web_video_server		Este pacote fornece um fluxo de vídeo de um tópico de transporte de imagens ROS que pode ser acessado via HTTP.
rosbridge_websocket		Enquanto rosbridge_library fornece a conversão JSON para ROS, ela deixa a camada de transporte para outros.
raspicam_node	diagnostics	Status e informações sobre a raspicam

Tabela 5.1: Descrição de nós e tópicos do sistema

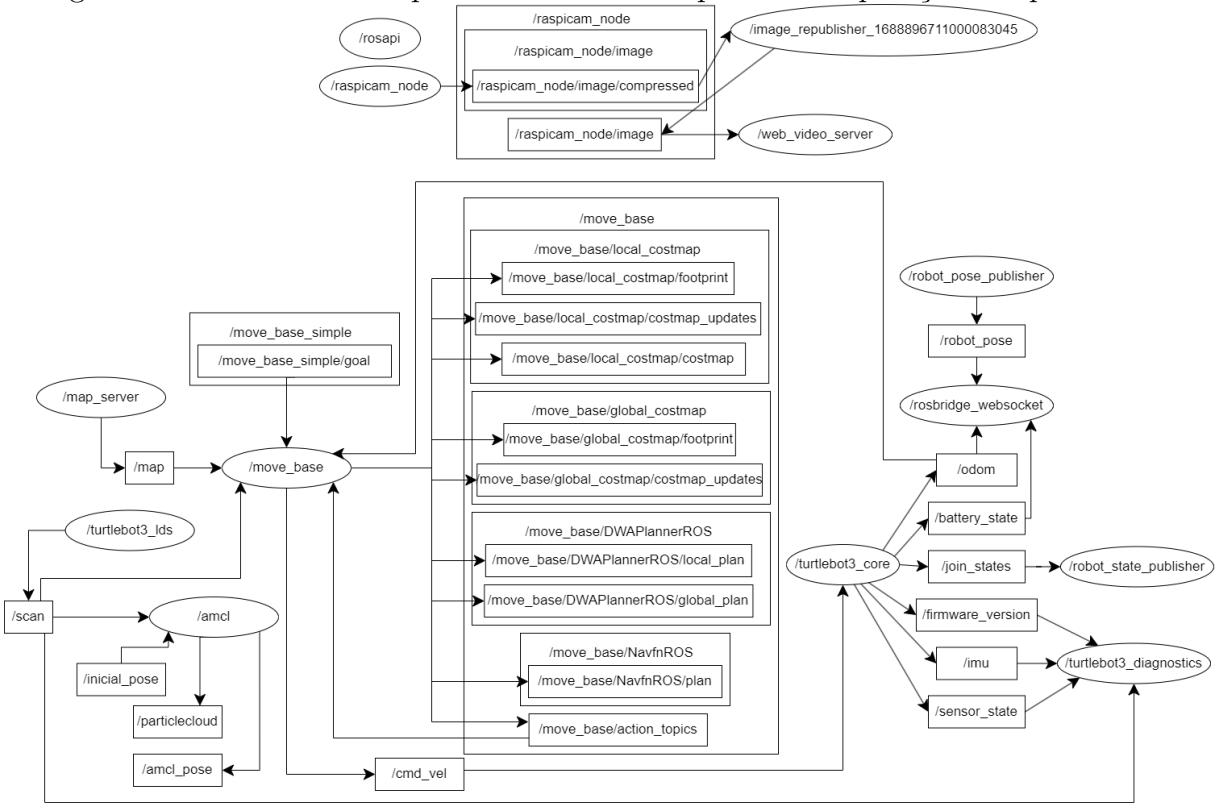
o percurso, o robô lida de forma autônoma com obstáculos locais, contornando-os de maneira adequada para realizar a atividade específica desejada.

A representação gráfica dos recursos utilizados pelo ROS pode ser visualizada na Figura 5.14, onde os nós e tópicos fazem parte do *move_base* e *robot_pose_publisher* representam os componentes disponíveis no modo de teleoperação em questão. Os demais componentes correspondem aos elementos previamente apresentados no método anteriormente discutido.

Figura 5.13: Visualização do mapa com as posições do robô e o objetivo

Fonte : Elaborada pelo autor

Figura 5.14: Gráfico dos tópicos ativos do ROS para a teleoperação compartilhada.



Fonte : Elaborada pelo autor

Capítulo 6

Conclusão e Discussão dos resultados

Em conclusão, os testes de teleoperação realizados para os dois modos apresentados neste estudo obtiveram êxito. A integração entre a aplicação web e o robô foi estabelecida de forma eficiente e ágil por meio do ROS. Observou-se que, para dados simples, como os estados do robô e os comandos de velocidade, a transmissão ocorreu em tempo real. No entanto, verificou-se um certo atraso na transmissão das imagens provenientes da câmera. Além disso, a utilização do pacote de navegação, conhecido como "navigation stack", permitiu que o robô executasse a atividade de forma eficaz, utilizando os principais algoritmos aplicados na literatura. Isso proporcionou uma teleoperação autônoma e eficiente, atendendo aos requisitos necessários para a realização da tarefa.

No entanto, durante o modo de teleoperação direta, foram notados alguns problemas de comunicação, como o atraso no tempo de resposta do robô. Embora essas questões tenham surgido, é importante destacar que elas podem ser abordadas em trabalhos futuros. A melhoria da comunicação em tempo real entre o operador e o robô é um aspecto crucial a ser explorado, visando aprimorar a experiência de teleoperação e permitir um controle mais eficiente e preciso em diferentes cenários. Essas considerações apontam para a necessidade de avanços contínuos na área, impulsionando futuras pesquisas e desenvolvimentos no campo da teleoperação

Bibliografia

[Alexander et al., 2012] Alexander, B., Hsiao, K., Jenkins, C., Suay, B., and Toris, R. (2012). Robot web tools [ros topics]. *IEEE Robotics Automation Magazine*, 19(4):20–23.

[Amsters and Slaets, 2020] Amsters, R. and Slaets, P. (2020). Turtlebot 3 as a robotics education platform. In *Robotics in Education: Current Research and Innovations 10*, pages 170–181. Springer.

[Anderson et al., 2020] Anderson, R. C., Adamo, D., Jones, T., Podnar, G., Artigas, J., Backes, P., Badger, J., Bailey, S. A., Bell, J., Bienhoff, D., et al. (2020). Space science opportunities augmented by exploration telepresence.

[Aracil and Ferre, 2007] Aracil, R. and Ferre, M. (2007). Telerobotics for aerial live power line maintenance. In *Advances in telerobotics*, pages 459–469. Springer.

[Bailey and Durrant-Whyte, 2006] Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): part ii. *IEEE Robotics Automation Magazine*, 13(3):108–117.

[Balachandran et al., 2020] Balachandran, R., Ryu, J.-H., Jorda, M., Ott, C., and Albu-Schaeffer, A. (2020). Closing the force loop to enhance transparency in time-delayed teleoperation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10198–10204. IEEE.

[Bonnabel, 2012] Bonnabel, S. (2012). Symmetries in observer design: Review of some recent results and applications to ekf-based slam. *Robot Motion and Control 2011*, pages 3–15.

[Casañ et al., 2015] Casañ, G. A., Cervera, E., Moughlbay, A. A., Alemany, J., and Martinet, P. (2015). Ros-based online robot programming for remote education and training. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6101–6106.

[Chen et al., 2022] Chen, X., Johannsmeier, L., Sadeghian, H., Shahriari, E., Danneberg, M., Nicklas, A., Wu, F., Fettweis, G., and Haddadin, S. (2022). On the communication

channel in bilateral teleoperation: An experimental study for ethernet, wifi, lte and 5g. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7712–7719.

[Chen et al., 2020] Chen, Y., Zhang, B., Zhou, J., and Wang, K. (2020). Real-time 3d unstructured environment reconstruction utilizing vr and kinect-based immersive teleoperation for agricultural field robots. *Computers and Electronics in Agriculture*, 175:105579.

[Choset et al., 2005] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G. A., and Burgard, W. (2005). *Principles of robot motion: theory, algorithms, and implementations*. MIT press.

[Crick et al., 2017] Crick, C., Jay, G., Osentoski, S., Pitzer, B., and Jenkins, O. C. (2017). Rosbridge: Ros for non-ros users. In *Robotics Research: The 15th International Symposium ISRR*, pages 493–504. Springer.

[Cynthia et al., 2021] Cynthia, J., Mohankumar, T., Arjun, T., and Naveenkumar, C. (2021). Development of python based ui application for tele-operated vehicles. In *2021 IEEE 6th International Conference on Computing, Communication and Automation (ICCCA)*, pages 383–388.

[Das et al., 2021] Das, S., Kumari, R., and Deepak Kumar, S. (2021). A review on applications of simultaneous localization and mapping method in autonomous vehicles. *Advances in Interdisciplinary Engineering: Select Proceedings of FLAME 2020*, pages 367–375.

[de Assis Brasil et al., 2020] de Assis Brasil, P. M., Pereira, F. U., Cuadros, M. A. d. S. L., Cukla, A. R., and Gamarra, D. F. T. (2020). A study on global path planners algorithms for the simulated turtlebot 3 robot in ros. In *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, pages 1–6. IEEE.

[De-Sheng and Qin-Yi, 2022] De-Sheng, L. and Qin-Yi, T. (2022). Research on avatar-like robot design based on virtual museum scene. *The Journal of China Universities of Posts and Telecommunications*, 29(2):43.

[Deng et al., 2021] Deng, Y., Tang, Y., Yang, B., Zheng, W., Liu, S., and Liu, C. (2021). A review of bilateral teleoperation control strategies with soft environment. In *2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 459–464. IEEE.

[Djuric et al., 2003] Djuric, P., Kotecha, J., Zhang, J., Huang, Y., Ghirmai, T., Bugallo, M., and Miguez, J. (2003). Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38.

[Dorf and Nof, 1990] Dorf, R. C. and Nof, S. Y. (1990). Concise international encyclopedia of robotics.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110.

[Estefo et al., 2019] Estefo, P., Simmonds, J., Robbes, R., and Fabry, J. (2019). The robot operating system: Package reuse and community dynamics. *Journal of Systems and Software*, 151:226–242.

[Ferre et al., 2007] Ferre, M., Buss, M., Aracil, R., Melchiorri, C., and Balaguer, C. (2007). *Introduction to advances in telerobotics*. Springer.

[Ferrell and Sheridan, 1967] Ferrell, W. R. and Sheridan, T. B. (1967). Supervisory control of remote manipulation. *IEEE Spectrum*, 4(10):81–88.

[Fink et al., 2014] Fink, G., Flatow, I., Group, S., et al. (2014). *Pro Single Page Application Development: Using Backbone. Js and ASP. Net*, volume 1. Springer.

[Fok et al., 2016] Fok, C. L., Sun, F., Mangum, M., Mok, A., He, B., and Sentis, L. (2016). Web based teleoperation of a humanoid robot. *arXiv preprint arXiv:1607.05402*.

[González et al., 2021] González, C., Solanes, J. E., Muñoz, A., Gracia, L., Girbés-Juan, V., and Tornero, J. (2021). Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback. *Journal of Manufacturing Systems*, 59:283–298.

[Gottardi et al., 2022] Gottardi, A., Tortora, S., Tosello, E., and Menegatti, E. (2022). Shared control in robot teleoperation with improved potential fields. *IEEE Transactions on Human-Machine Systems*, 52(3):410–422.

[Graf and Hussmann, 2020] Graf, G. and Hussmann, H. (2020). User requirements for remote teleoperation-based interfaces. In *12th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, AutomotiveUI '20, page 85–88, New York, NY, USA. Association for Computing Machinery.

[Hainsworth, 2001] Hainsworth, D. W. (2001). Teleoperation user interfaces for mining robotics. *Autonomous Robots*, 11(1):19–28.

[Hetrick et al., 2020] Hetrick, R., Amerson, N., Kim, B., Rosen, E., Visser, E. J. d., and Phillips, E. (2020). Comparing virtual reality interfaces for the teleoperation of robots. In *2020 Systems and Information Engineering Design Symposium (SIEDS)*, pages 1–7.

[Hui-min et al., 2008] Hui-min, H., Cheng-lin, W., and Xiao-bin, X. (2008). Particle filter for range-only tracking in airborne radar. In *2008 Chinese Control and Decision Conference*, pages 5053–5058.

[IFR, 2022] IFR (2022). Executive summary wr industrial robots 2022. urlhttps://ifr.org/img/worldrobotics/Executive$_S ummary_W R_I ndustrial_R obots_2$022.*pdf*.

[Johansson and de Vin, 2009] Johansson, D. and de Vin, L. J. (2009). Omnidirectional robotic telepresence through augmented virtuality for increased situation awareness in hazardous environments. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 6–11.

[Ko and Kim, 2012] Ko, N. Y. and Kim, T. G. (2012). Comparison of kalman filter and particle filter used for localization of an underwater vehicle. In *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 350–352.

[Koubaa, 2022] Koubaa, A. (2022). ROS-Teleoperation-Web-Interface. Acesso em: 1 de agosto de 2023, https://github.com/aniskoubaa/udemy-rosbridge-course.

[Krupke et al., 2016] Krupke, D., Einig, L., Langbehn, E., Zhang, J., and Steinicke, F. (2016). Immersive remote grasping: realtime gripper control by a heterogenous robot control system. In *Proceedings of the 22nd ACM Conference on Virtual Reality Software and Technology*, pages 337–338.

[Law et al., 2022] Law, W.-t., Li, K.-s., Fan, K.-w., Ko, W.-h., Mo, T., and Poon, C.-k. (2022). Two-way human-robot interaction in 5g tele-operation. In *2022 17th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 1198–1199. IEEE.

[Lee, 2012] Lee, J. (2012). Web applications for robots using rosbridge. *Brown University*.

[Lee and Ham, 2022] Lee, J. S. and Ham, Y. (2022). Exploring human-machine interfaces for teleoperation of excavator. In *Construction Research Congress 2022*, pages 757–765.

[Li et al., 2015] Li, Z., Xia, Y., and Su, C.-Y. (2015). Intelligent networked teleoperation control.

[Lin et al., 2022] Lin, Z., Zhang, T., Sun, Z., Gao, H., Ai, X., Chen, W., Yang, G.-Z., and Gao, A. (2022). Robotic telepresence based on augmented reality and human motion mapping for interventional medicine. *IEEE Transactions on Medical Robotics and Bionics*, 4(4):935–944.

[Martínez, 2021] Martínez, F. H. (2021). Turtlebot3 robot operation for navigation applications using ros. *Tekhnê*, 18(2):19–24.

[Matsas et al., 2018] Matsas, E., Vosniakos, G.-C., and Batras, D. (2018). Prototyping proactive and adaptive techniques for human-robot collaboration in manufacturing using virtual reality. *Robotics and Computer-Integrated Manufacturing*, 50:168–180.

[Murphy, 2019] Murphy, R. R. (2019). *Introduction to AI robotics*. MIT press.

[Ni et al., 2017] Ni, D., Song, A., and Li, H. (2017). Survey on robot teleoperation based on virtual reality. *Yi Qi Yi Biao Xue Bao/Chinese Journal of Scientific Instrument*, 38:2351–2363.

[Oh et al., 2021] Oh, Y., Schäfer, T., Rüther, B., Toussaint, M., and Mainprice, J. (2021). A system for traded control teleoperation of manipulation tasks using intent prediction from hand gestures. In *2021 30th IEEE International Conference on Robot Human Interactive Communication (RO-MAN)*, pages 503–508.

[Okura et al., 2010] Okura, F., Kanbara, M., and Yokoya, N. (2010). Augmented telepresence using autopilot airship and omni-directional camera. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 259–260.

[Olatunji et al., 2022] Olatunji, S. A., Potenza, A., Kiselev, A., Oron-Gilad, T., Loutfi, A., and Edan, Y. (2022). Levels of automation for a mobile robot teleoperated by a caregiver. *ACM Transactions on Human-Robot Interaction (THRI)*, 11(2):1–21.

[Opiyo et al., 2021] Opiyo, S., Zhou, J., Mwangi, E., Kai, W., and Sunusi, I. (2021). A review on teleoperation of mobile ground robots: Architecture and situation awareness. *International Journal of Control, Automation and Systems*, 19(3):1384–1407.

[Pesce et al., 2023] Pesce, V., Hermosin, P., Rivolta, A., Bhaskaran, S., Silvestrini, S., and Colagrossi, A. (2023). Navigation. In *Modern Spacecraft Guidance, Navigation, and Control*, pages 441–542. Elsevier.

[Placed et al., 2023] Placed, J. A., Strader, J., Carrillo, H., Atanasov, N., Indelman, V., Carlone, L., and Castellanos, J. A. (2023). A survey on active simultaneous localization and mapping: State of the art and new frontiers. *IEEE Transactions on Robotics*.

[Pritsker et al., 1986] Pritsker, A. A. B., Rolston, L. J., and Floss, P. (1986). *Solutions: introduction to Simulation and SLAM II*. Systems Publishing Corporation.

[Safin et al., 2021] Safin, R., Lavrenov, R., Hsia, K.-H., Maslak, E., Schiefermeier-Mach, N., and Magid, E. (2021). Modelling a turtlebot3 based delivery system for a smart hospital in gazebo. In *2021 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–6. IEEE.

[Schmidt et al., 2014] Schmidt, L., Hegenberg, J., and Cramar, L. (2014). User studies on teleoperation of robots for plant inspection. *Industrial Robot: An International Journal*, 41(1):6–14.

[Scott Jr, 2015] Scott Jr, E. A. (2015). *SPA Design and Architecture: Understanding single-page web applications*. Simon and Schuster.

[Sharrab et al., 2023] Sharrab, Y., Almutiri, N. T., Tarawneh, M., Alzyoud, F., Al-Ghuwairi, A.-R. F., and Al-Fraihat, D. (2023). Toward smart and immersive classroom based on ai, vr, and 6g. *International Journal of Emerging Technologies in Learning (Online)*, 18(2):4.

[Siciliano et al., 2008] Siciliano, B., Khatib, O., and Kröger, T. (2008). *Springer handbook of robotics*, volume 200. Springer.

[Sun, 2019] Sun, Y. (2019). Practical application development with apprun. *Building Reliable, High-Performance Web Apps Using Elm-Inspired Architecture, Event Pub-Sub, and Components. Berkeley, CA: Apress*.

[Szymańska et al., 2021] Szymańska, E., Petrović, L., Marković, I., and Petrović, I. (2021). Mobile robot teleoperation via android mobile device with udp communication. In *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, pages 1143–1148. IEEE.

[Tokatli et al., 2021] Tokatli, O., Das, P., Nath, R., Pangione, L., Altobelli, A., Burroughes, G., Jonasson, E. T., Turner, M. F., and Skilton, R. (2021). Robot-assisted glovebox teleoperation for nuclear industry. *Robotics*, 10(3):85.

[Uttal, 1989] Uttal, W. R. (1989). Teleoperators. *Scientific American*, 261(6):124–129.

[Vertut and Coiffet, 1985] Vertut, J. and Coiffet, P. (1985). Robot technology. vol. 3b. teleoperation and robotics: applications and technology.

[Walker et al., 2019] Walker, M. E., Szafir, D., and Rae, I. (2019). The influence of size in augmented reality telepresence avatars. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 538–546.

[Yan et al., 2017] Yan, H., Zhao, X., Gao, S., Zhang, J., and Shao, X. (2017). Adaptive robust unscented particle filter and its application in sins/sar integration navigation system. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 2364–2368.

[Yu et al., 2013] Yu, H., Wang, G., Cao, Q., and Sun, Y. (2013). A novel particle filtering algorithm based on state fusion. In *IET International Radar Conference 2013*, pages 1–5.

[Zhang et al., 2018] Zhang, J., Liu, W., Li, L., Li, Z., et al. (2018). The master adaptive impedance control and slave adaptive neural network control in underwater manipulator uncertainty teleoperation. *Ocean Engineering*, 165:465–479.

[Zhao et al., 2006] Zhao, M., Zhang, S., and Zhu, G. (2006). The application reserch of unscented particle filter algorithm to gps/dr. In *2006 6th World Congress on Intelligent Control and Automation*, volume 2, pages 8717–8721.